Template Final Exam

|  |  |  |
| --- | --- | --- |
| # | Full Name | Group |
| 1 | Satkanbay Ayaulym | CS-2122N |
| 2 | Esenova Gaukhar | CS-2122N |
|  |  |  |
|  |  |  |

|  |
| --- |
| Link to the repository: |

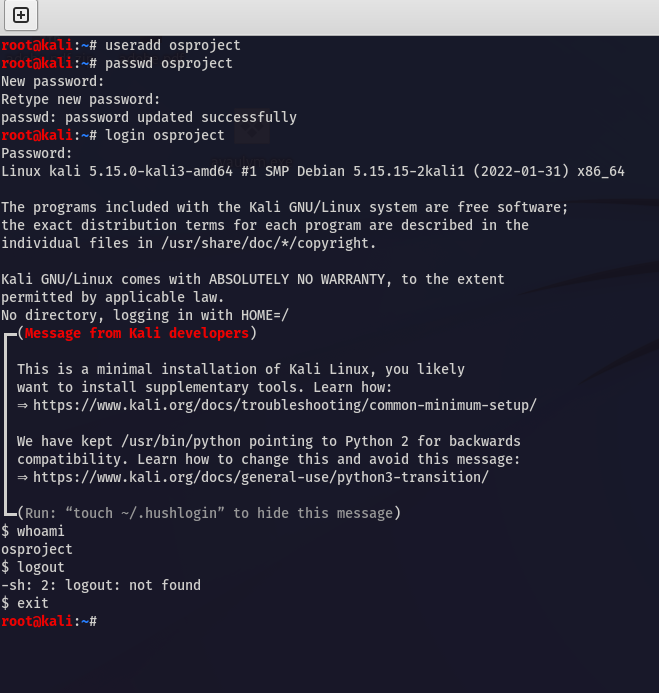
Step-by-step task completion:

Task 1:

First of all we added new user, password and login.

Then we checked with a help of “whoiam”

Last command “logout” and ‘’exit”



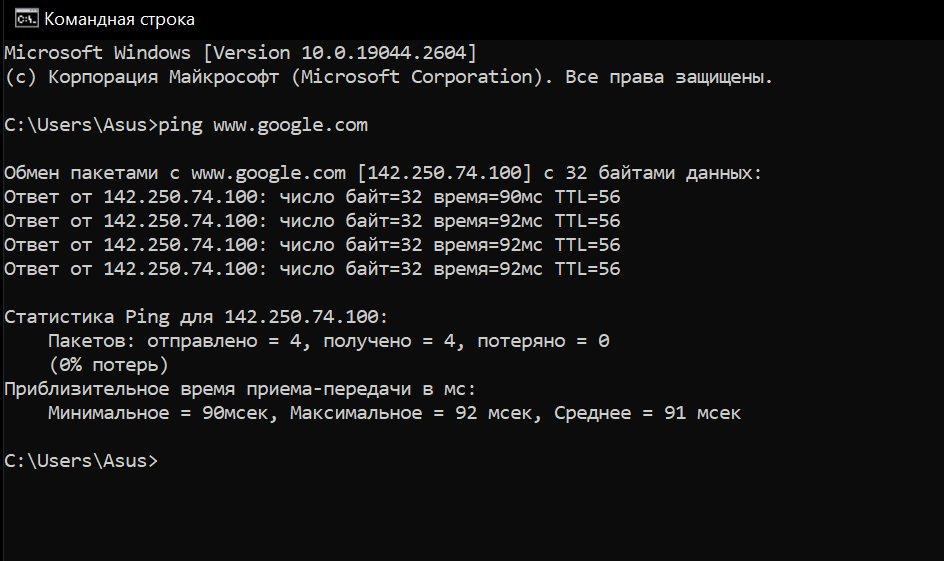
Task 2:

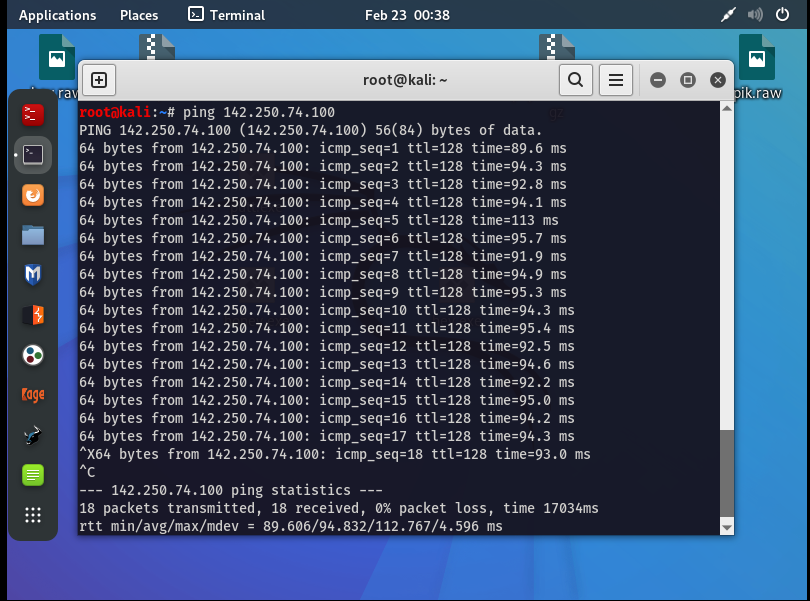
1. **Direct IP Connection.**
2. For start, Provide the virtual computer a direct public IP address.

For illustration, we'll use the ip domain of google.com which, ip address is - 142.250.74.100. We find it by pinging.

1. Because of public IP addresses are a limited resource, this method requires the virtual machine to have its own publicly available IP address, which may not always be possible.
2. The virtual machine can now have direct connection to the Internet.

As a result, by sending our traffic packets to this public IP, we can use the ping function to confirm it.

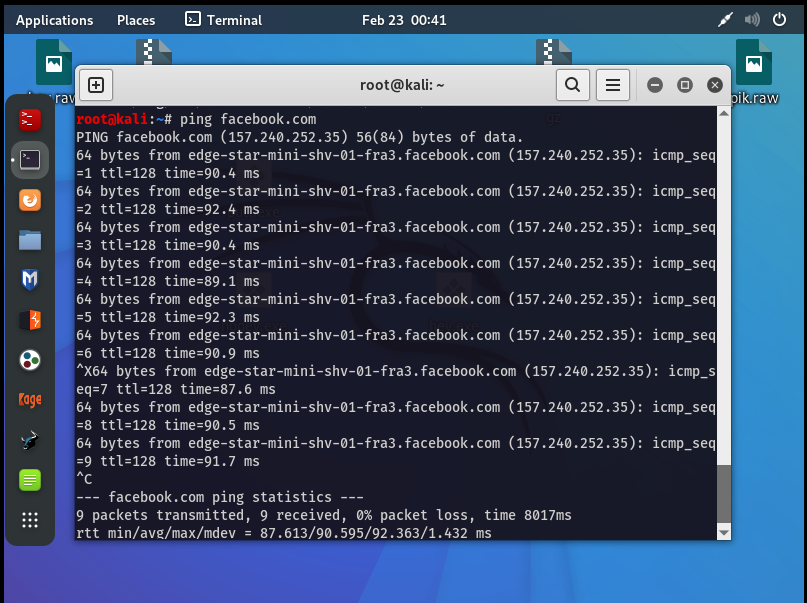
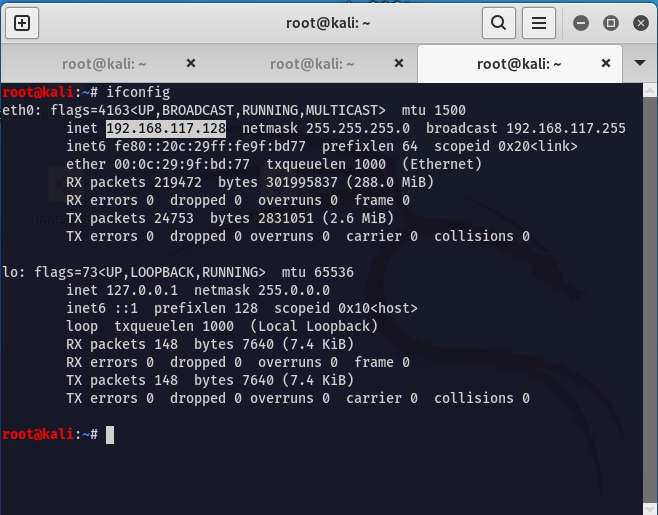


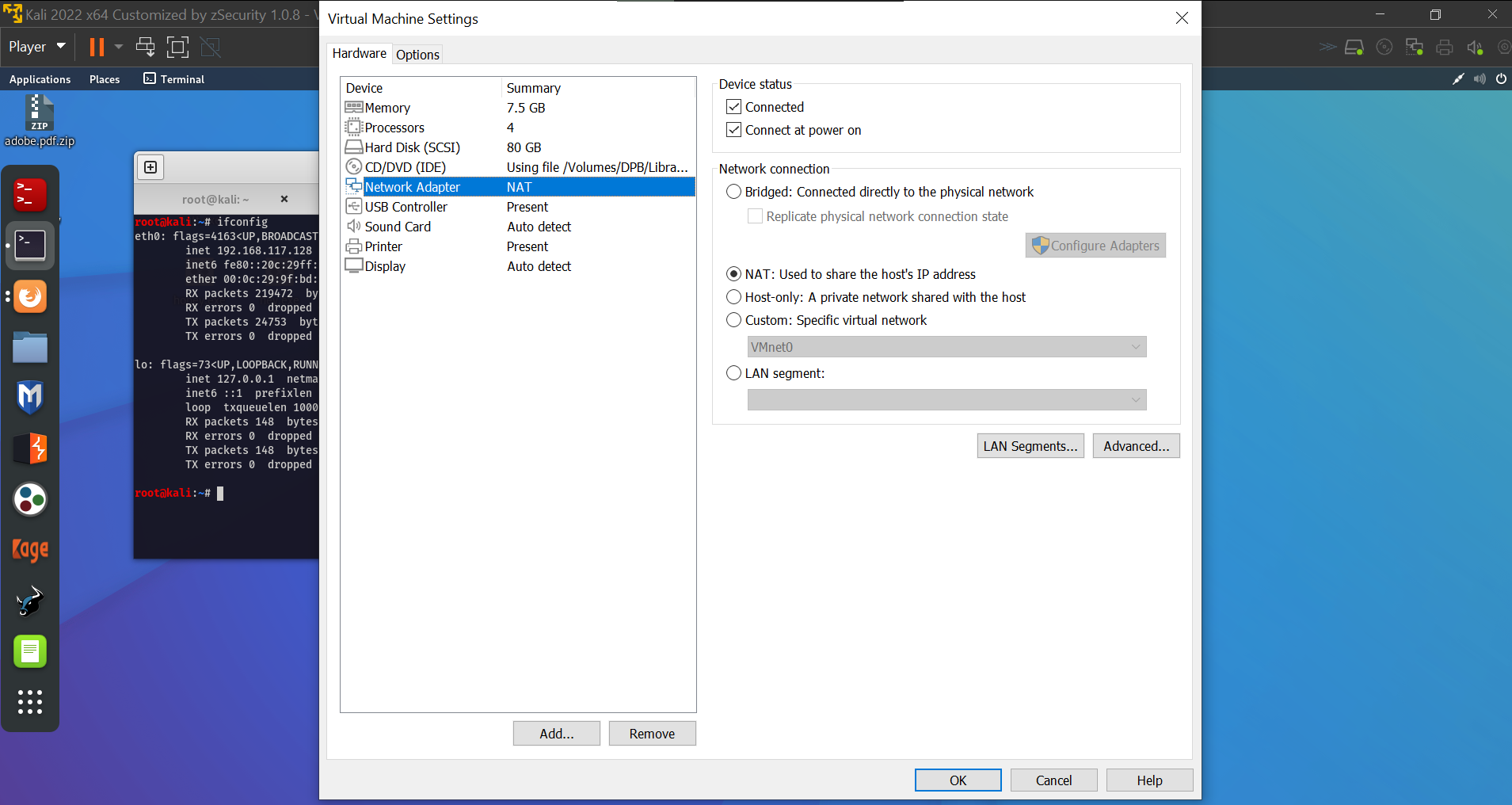


1. **NAT Connection.**
2. A private IP address is given to the virtual machine, and a NAT gateway is utilized to convert the private IP address to a public IP address.
3. This keeps the finite amount of public IP addresses while enabling the virtual machine to access the Internet.
4. Virtual private clouds and residential networks both frequently employ a NAT.

With the ping command, confirm the network to see if OS with NAT has an internet connection.

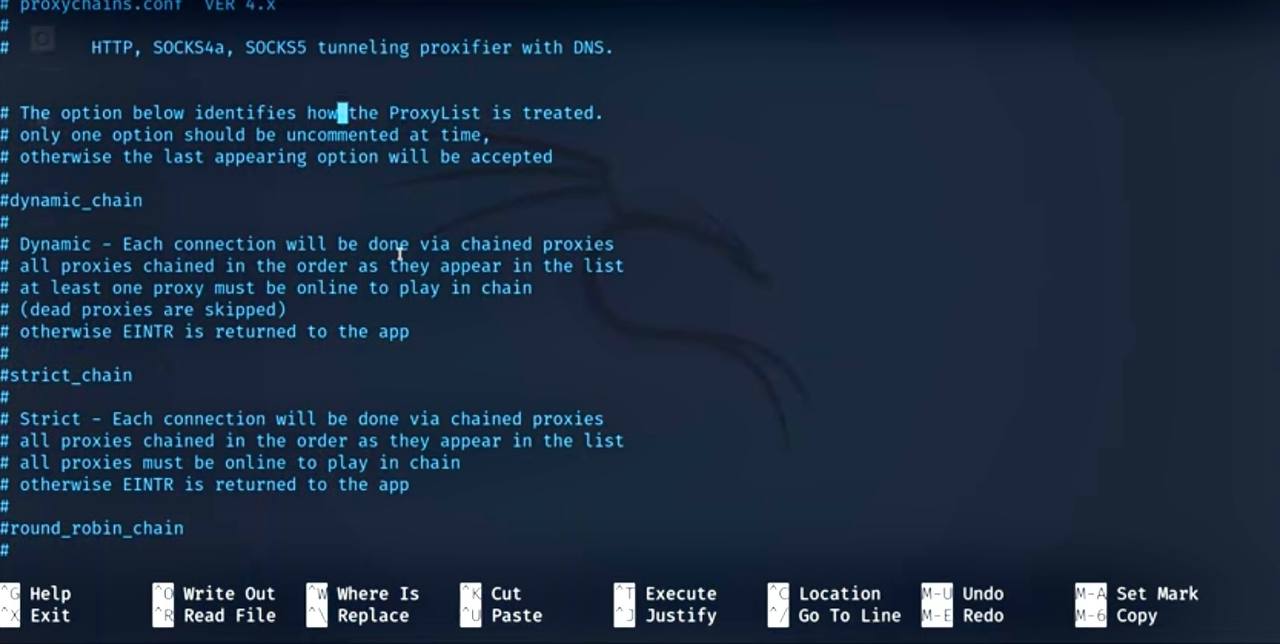
All of packet properly transmitted.

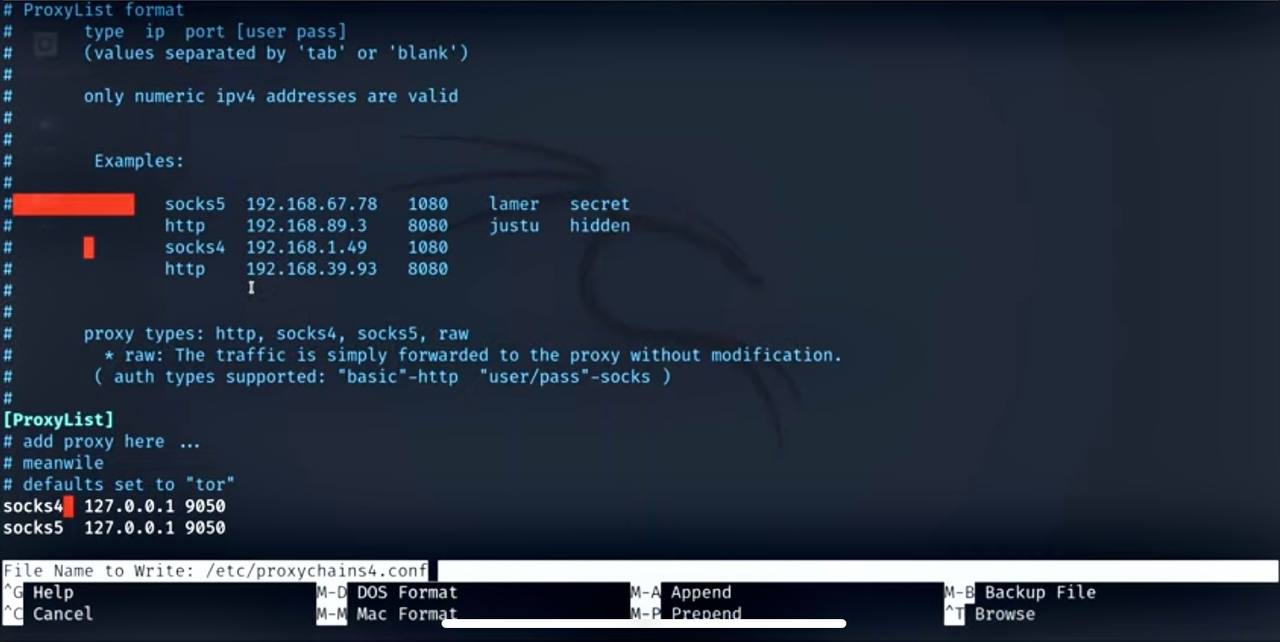




1. **Proxy Server Connection.**
2. Between the virtual machine and the Internet, a proxy server works as a middleman.
3. This proxy server could offer further protection & filtration functions, like the ability to restrict particular Web traffic types.
4. The virtual machine sends the proxy server its requests for Internet access, and the proxy server responds on the virtual machine's behalf.

Below, as you can see, we have used proxychains to access our website [www.google.com](http://www.google.com) in firefox, which hides our actual Internet address from the proxy server and sends requests using the another IP address.







Task 3:

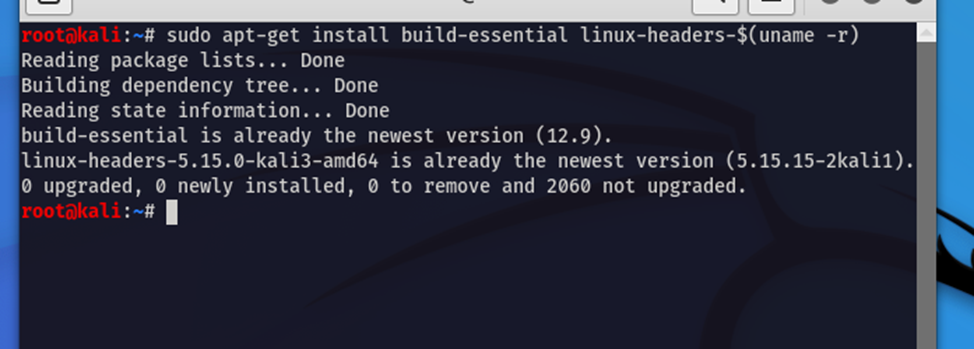
A Linux kernel module is a piece of software that can be dynamically loaded into the Linux kernel at runtime, allowing the kernel to gain additional functionality or device driver support without requiring a full kernel rebuild or reboot.

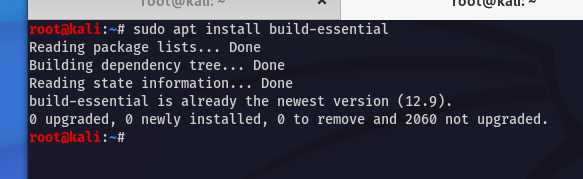
Developing a Linux kernel module involves writing code that interacts with the kernel's internal data structures and functions to achieve the desired functionality. This can involve understanding the Linux kernel's programming interfaces and data structures, as well as the specific requirements and limitations of the module being developed.

Some common steps involved in developing a Linux kernel module include:

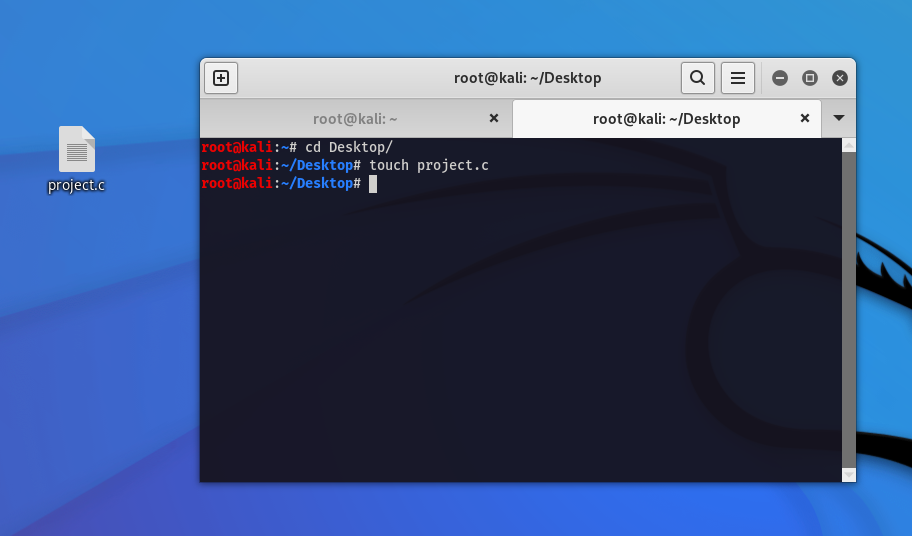
1. Setting up a development environment: This involves installing the necessary tools and libraries, such as the GNU Compiler Collection (GCC) and kernel headers, to build and test the module.
2. Writing the module code: This involves using a programming language such as C or C++ to implement the desired functionality, including any necessary data structures and algorithms.
3. Compiling the module: This involves using the development environment to compile the module code into a binary file that can be loaded into the kernel.

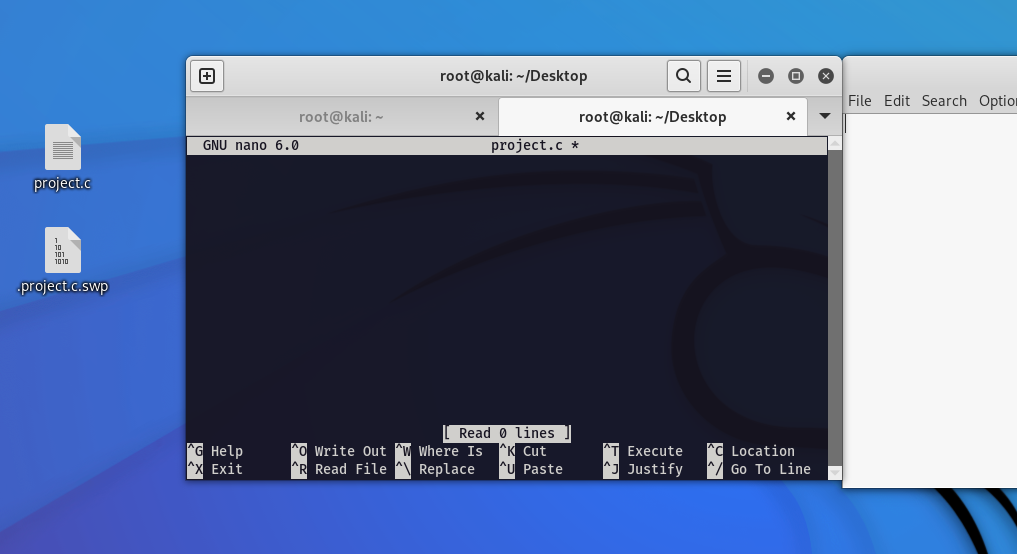
**At first we installed necessary packages for KMD:**

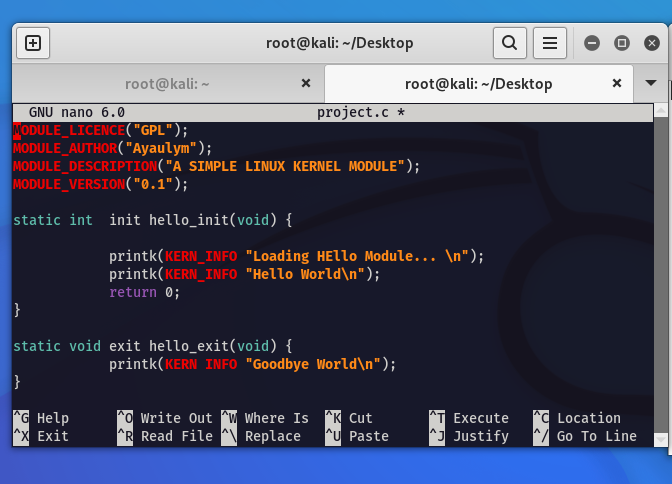




Then, we created .c file/program in Kali Linux







Testing the module: This involves using various testing techniques to ensure the module functions correctly and does not cause any system instability or crashes.

Packaging and distribution: Once the module is complete and tested, it can be packaged and distributed as part of a larger software project, such as a Linux distribution or a custom software package.