Assignment #1: Linux Kernel Installation

Muhammad Shafeen (22P-9278)

Step 1: System Update and Upgrade

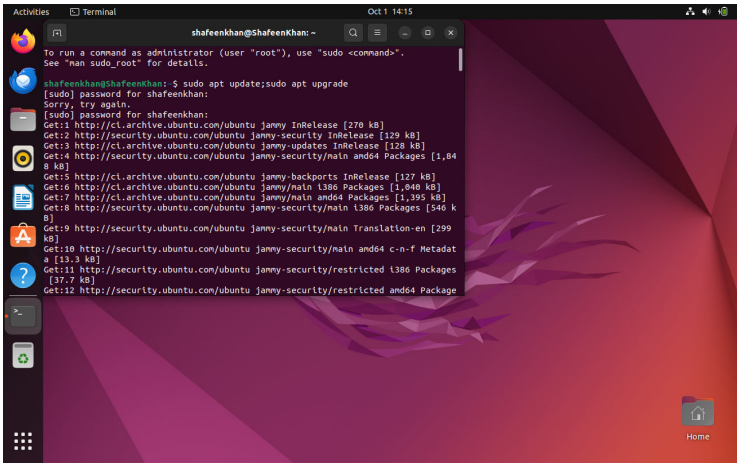

Figure 1: Updating and upgrading the system before kernel compilation.

Before starting the kernel installation process, updating the system using ‘sudo apt update‘ and ‘sudo apt upgrade‘ to ensure all packages are up to date.

Step 2: Installing Build Dependencies

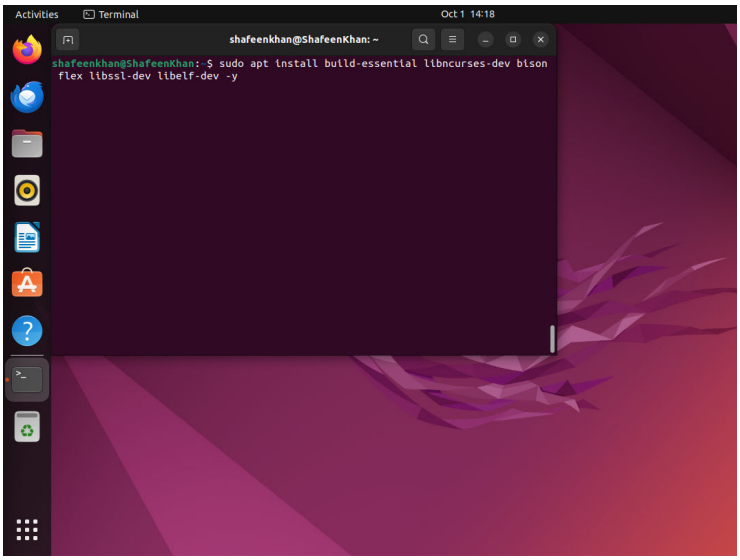

Figure 2: Installing the required build dependencies for kernel compilation.

The user installs essential build tools like ‘build-essential‘, ‘libncurses-dev‘, ‘bison‘, ‘flex‘, and others required for compiling the Linux kernel.

Step 3: Verifying Installed Packages

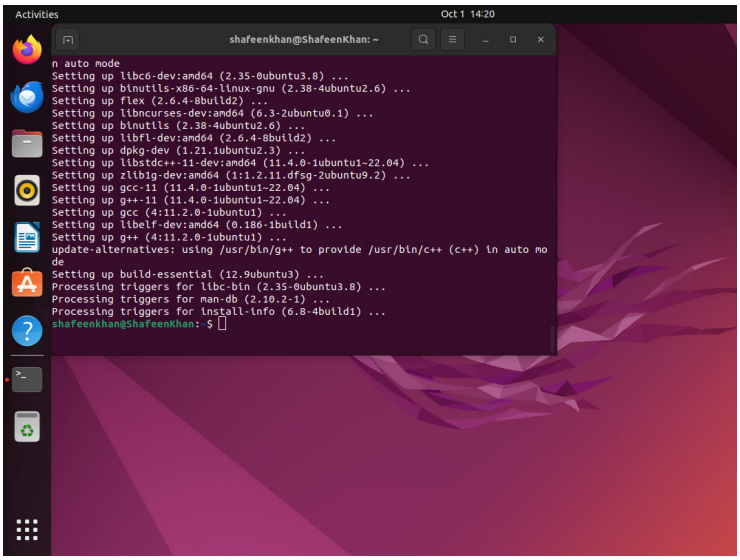

Figure 3: Successful installation of required packages for kernel compilation.

Step 4: Downloading Kernel from Kernel Archives

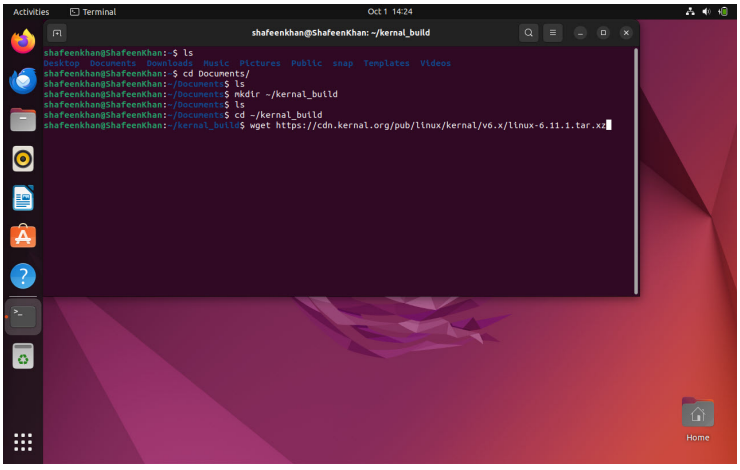

Figure 4: Accessing the Linux Kernel Archives to download the latest stable version.

Step 5: Extracting the Kernel Source

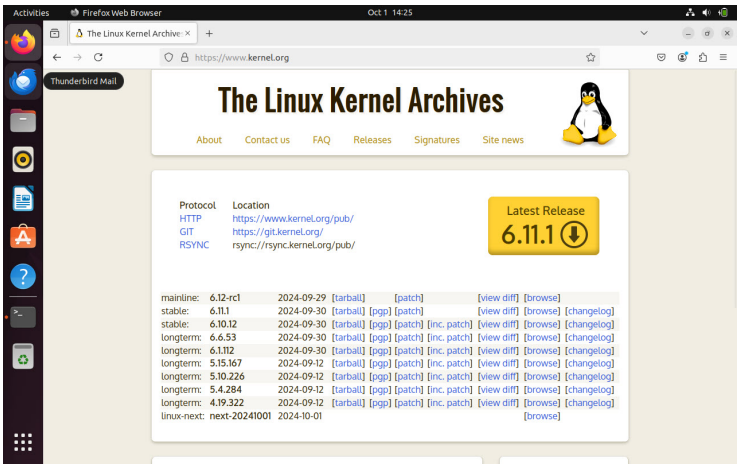

Figure 5: Downloading the linux kernal

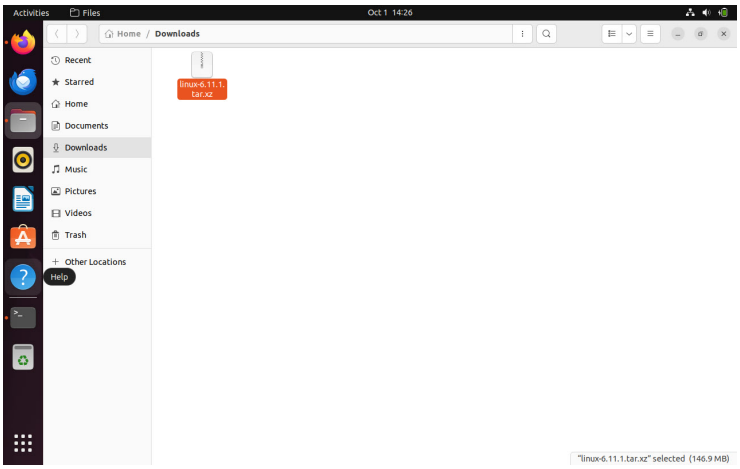

Figure 6: Displaying the downloaded file

Step 6: Extract the linux source code file

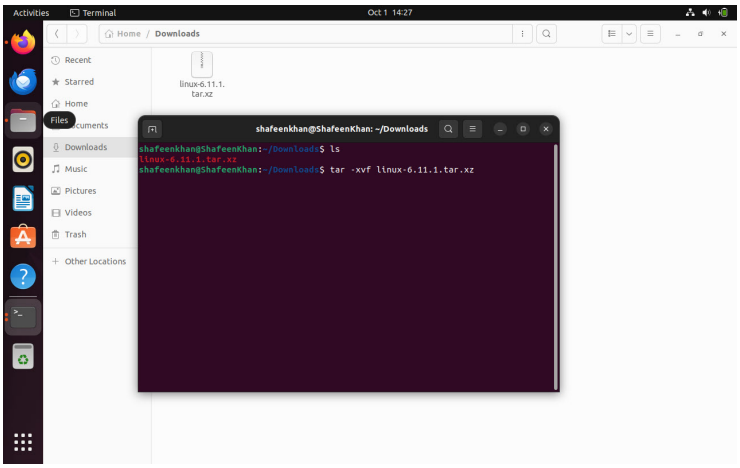

Figure 7: Extracting the kernel source file using the 'tar' command.

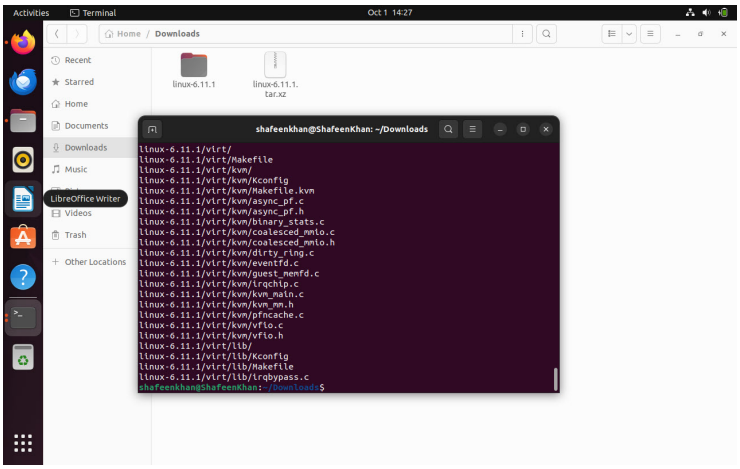

Figure 8: Extraction completed

Step 7: Kernal Configuration

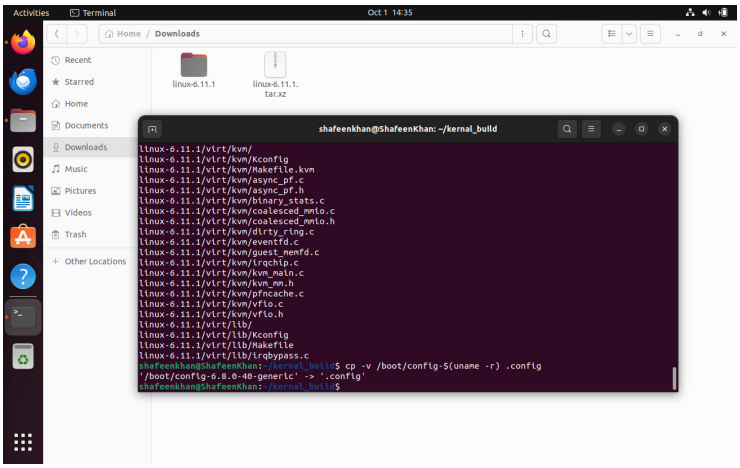

Figure 9: Ensure similar configuration for new kernal

0.1 Update Configuration Settings

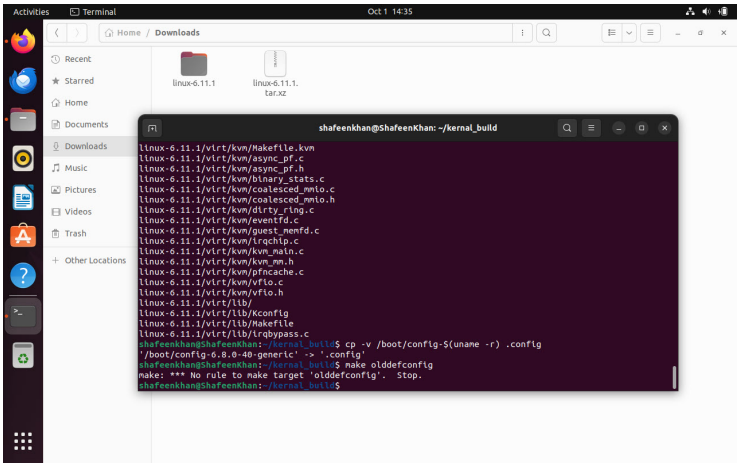

Figure 10: This updates the copied configuration to be compatible with the new kernel version

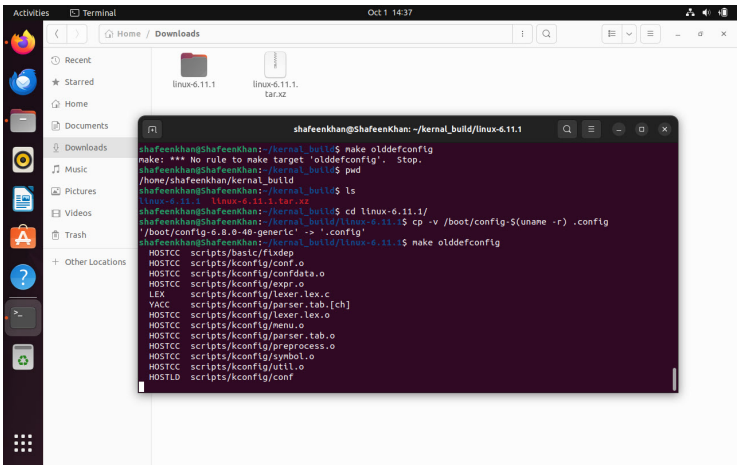

Figure 11: In process

0.2 Custom Kernal Configuration

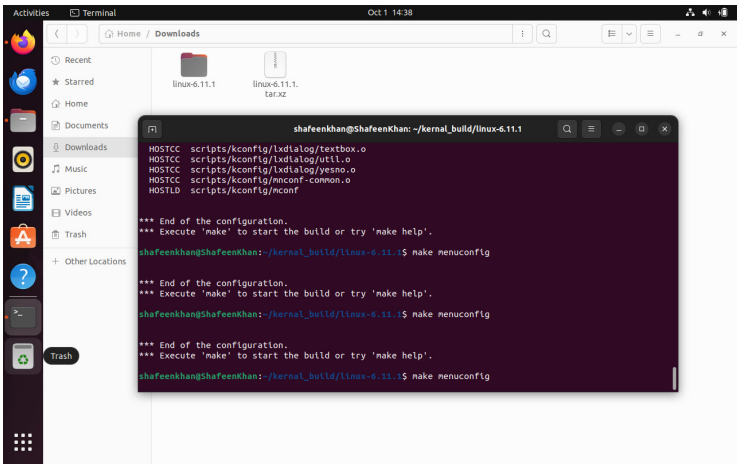

Figure 12: This opens a text-based UI where you can navigate and adjust kernel settings

0.3 Bonus Challenge
The UI - BASED interface for optional Tweaking

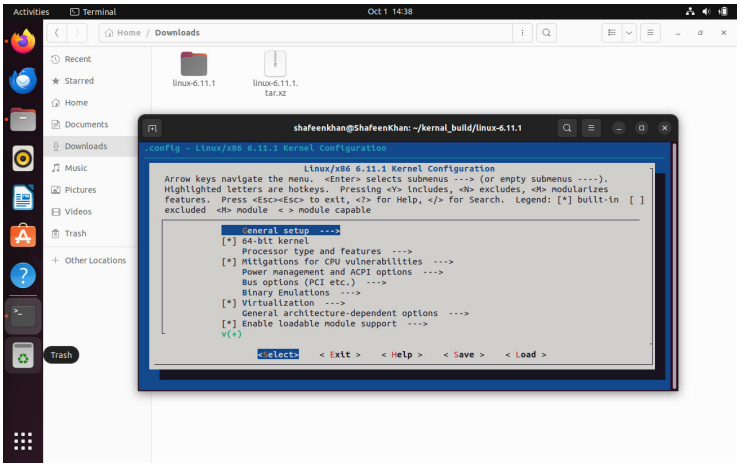

Figure 13: The text-based UI where you can navigate and adjust custom kernel settings for your benefit

0.4 Kernel Configuration Setup - File System Options

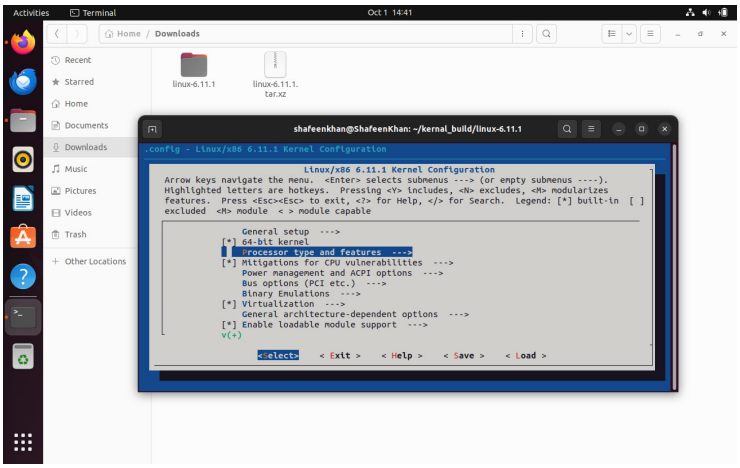

Figure 14: Navigating the kernel configuration menu to modify the file system options. The user is in the kernel configuration interface for Linux version 6.11.1. Various subsystems, including networking support and device drivers, are shown with a focus on selecting file systems for tweaking.

0.5 Kernel Configuration Setup - General Setup

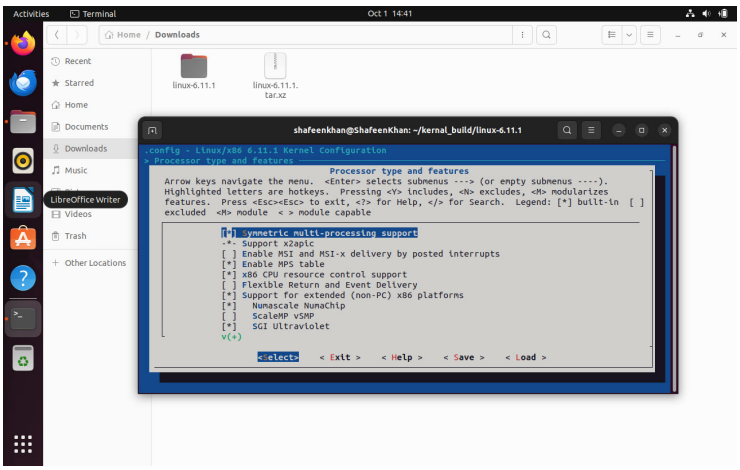

Figure 15: Turning on the Symmetric multi-processing support for faster computations and to increase speed

0.6 Kernel Configuration Setup - Optimizing for my own Specific Processor

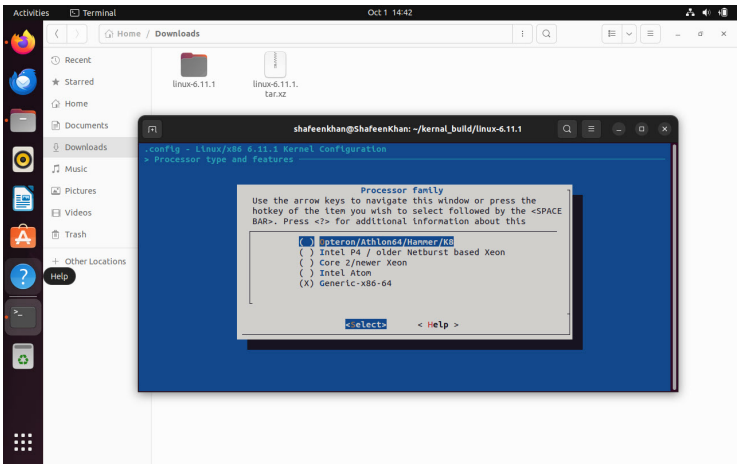

Figure 16: Choosing a specific processor family to optimize it further.

0.7 Kernel Configuration Setup - Optimizing for my own Specific Processor

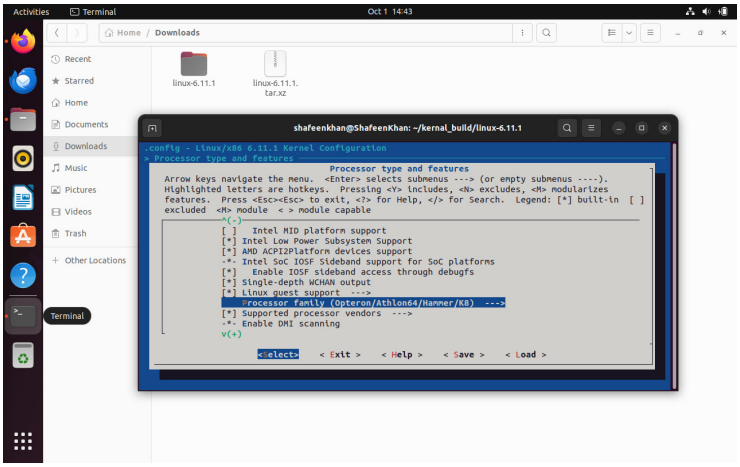

Figure 17: As you can see i chose the Opteron version because i am using an AMD ryzen processor which will give me an edge in processing speed as it is optimized

0.8 Kernel Configuration Setup - Customize Kernel Local Version

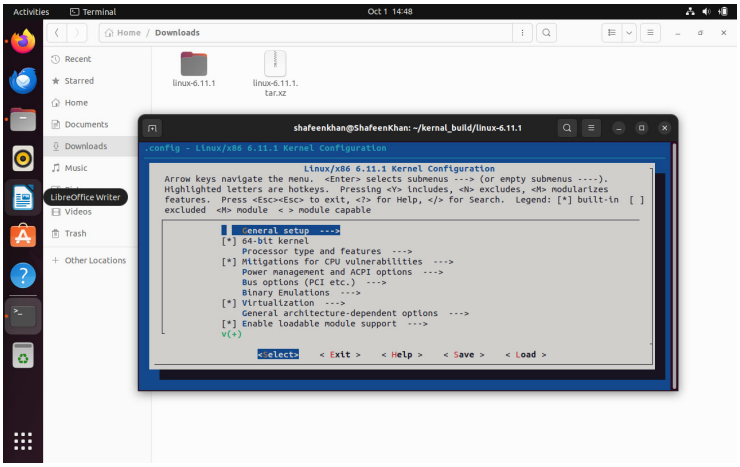

Figure 18: Going into general setup section to change the kernal version to easily identify it

0.9 Kernel Configuration Setup - Customize Kernel Local Version

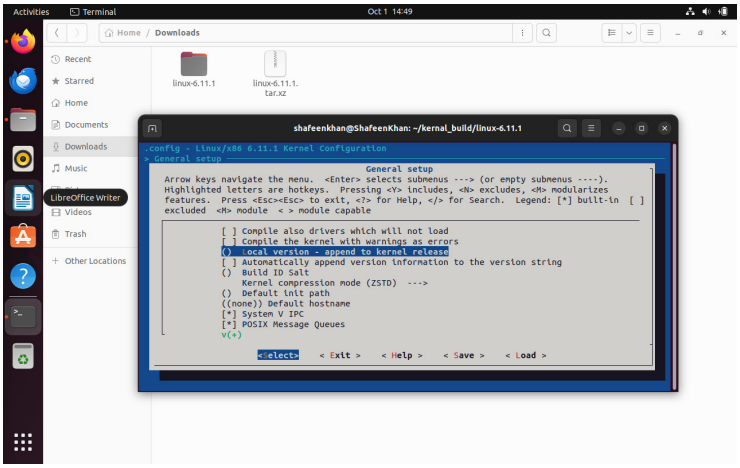

Figure 19: Continued

0.10 Kernel Configuration Setup - Customize Kernel Local Version

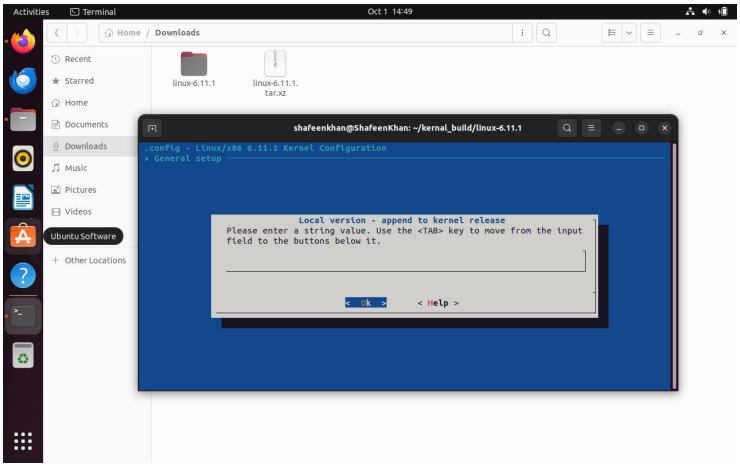

Figure 20: Continued

0.11 Kernel Configuration Setup - Customize Kernel Local Version

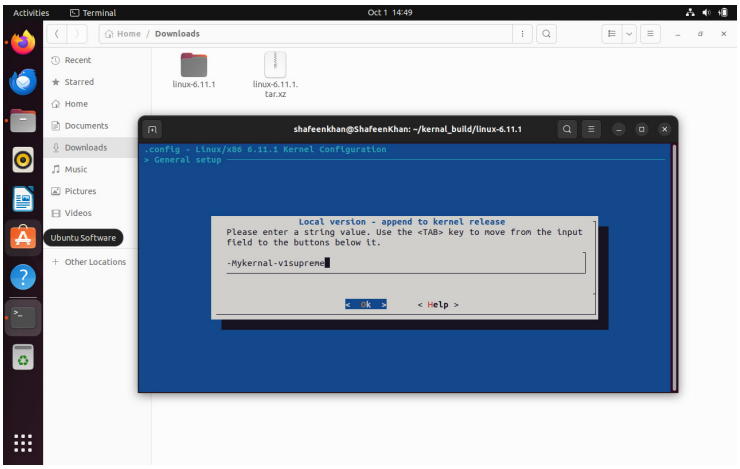

Figure 21: Continued

0.12 Kernel Configuration Setup - Customize Kernel Local Version

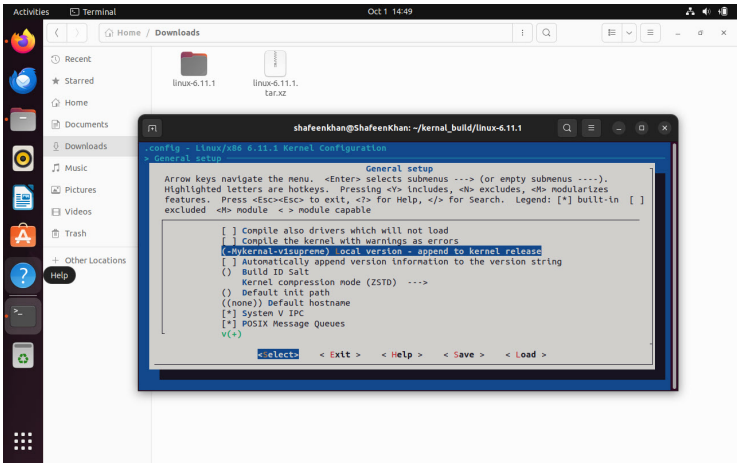

Figure 22: Continued

0.13 Kernel Configuration Setup - Disable Unnecessary Filesystems and Drivers

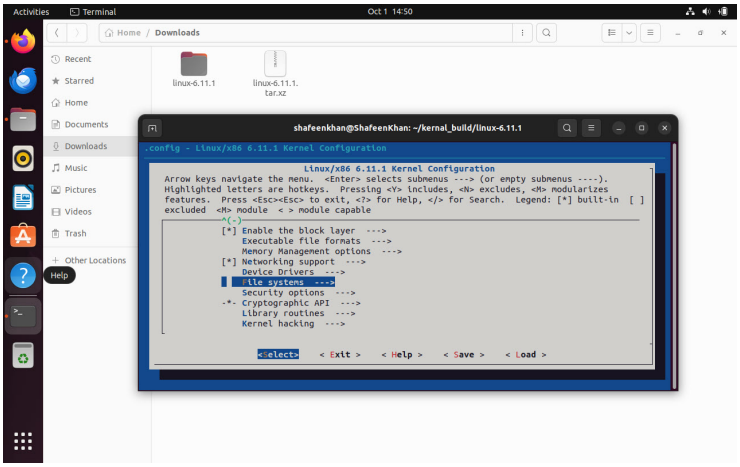

Figure 23: Removing support for filesystems and drivers you don't use can reduce kernel size and potentially speed up boot time

0.14 Kernel Configuration Setup - Disable Unnecessary Filesystems and Drivers

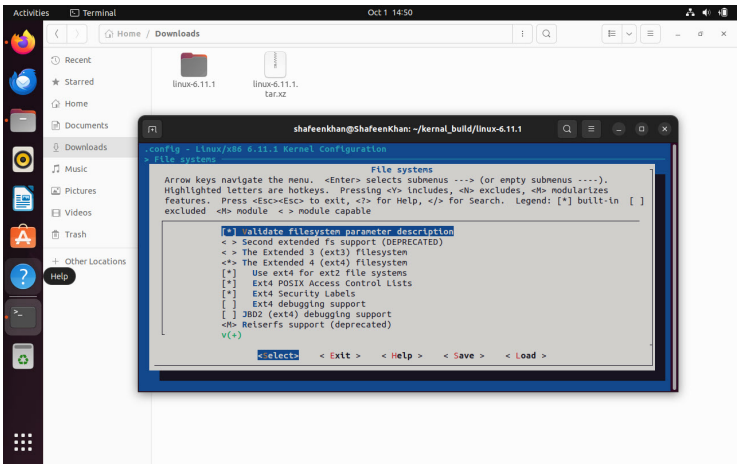

Figure 24: Continued

0.15 Kernel Configuration Setup - Disable Unnecessary Filesystems and Drivers

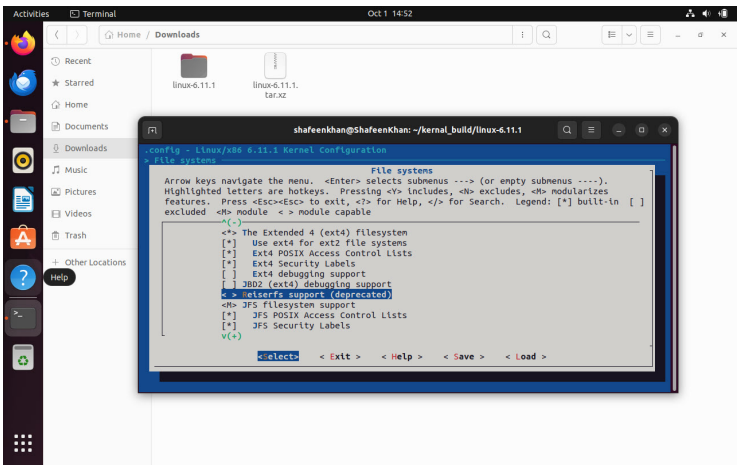

Figure 25: Continued

0.16 Kernel Configuration Setup - Disable Unnecessary Filesystems and Drivers

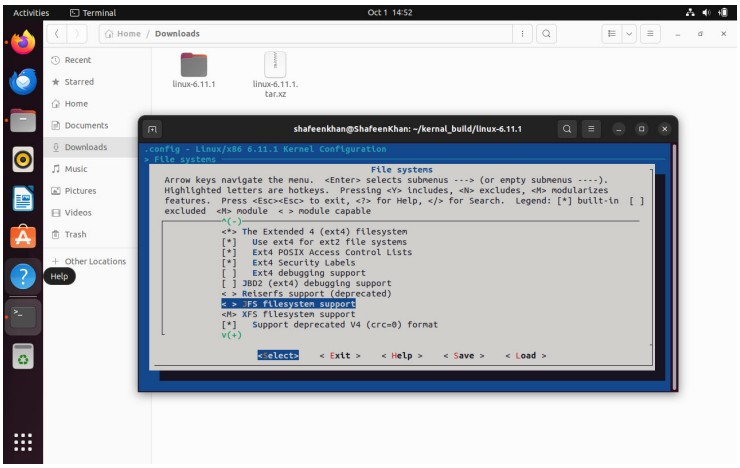

Figure 26: Continued

0.17 Kernel Configuration Setup - Disable RAID Controllers

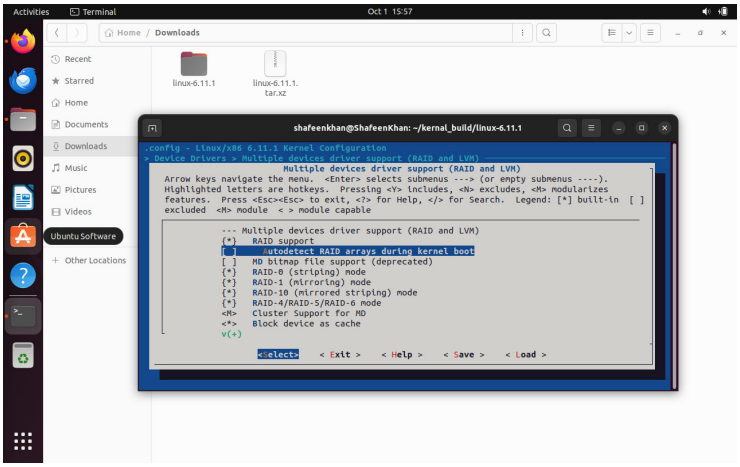

Figure 27: Disabling unused RAID support reduces the kernel size and boot time but also minimizes security risks by removing unnecessary code

0.18 Kernel Configuration Setup - Disable Camera

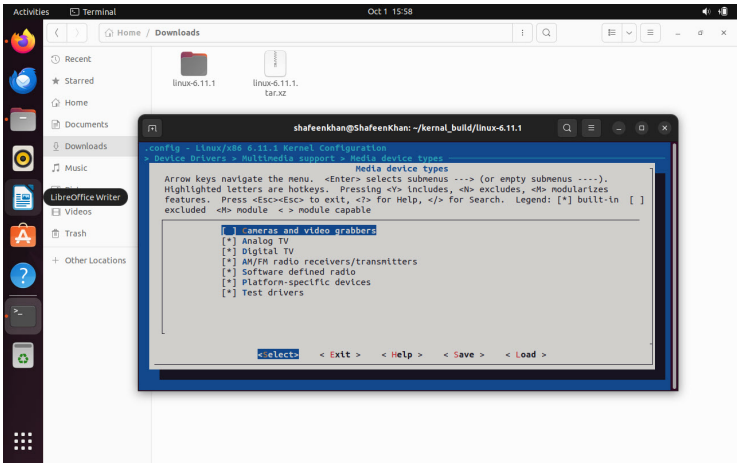

Figure 28: Disabling camera because it is of no use , to save resources / processes and to increase security

0.19 Enable Preemptive Kernel for Desktop Responsiveness

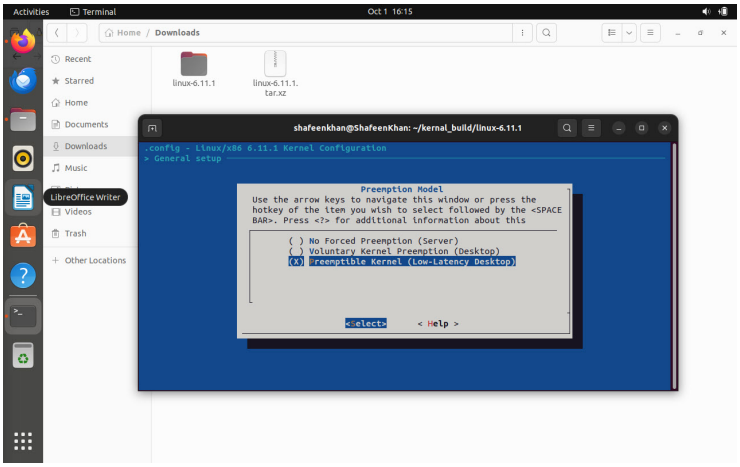

Figure 29: A preemptive kernel allows the system to interrupt tasks more frequently, improving responsiveness in desktop environments.

0.20 Enable Preemptive Kernel for Desktop Responsiveness

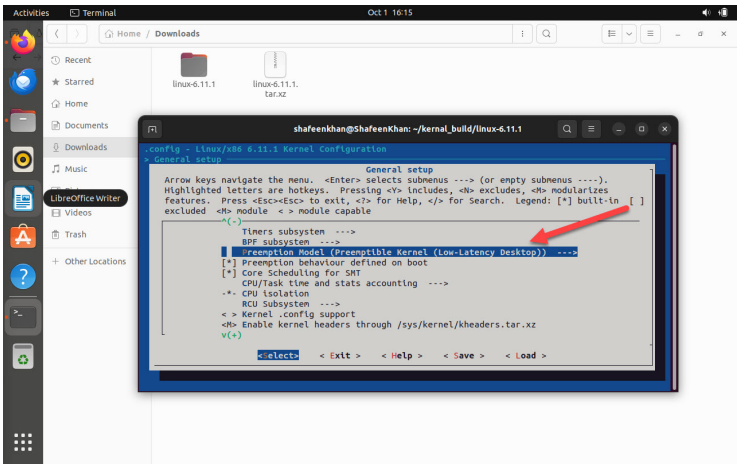

Figure 30: continued

0.21 Kernel Configuration Setup - Save Configuration

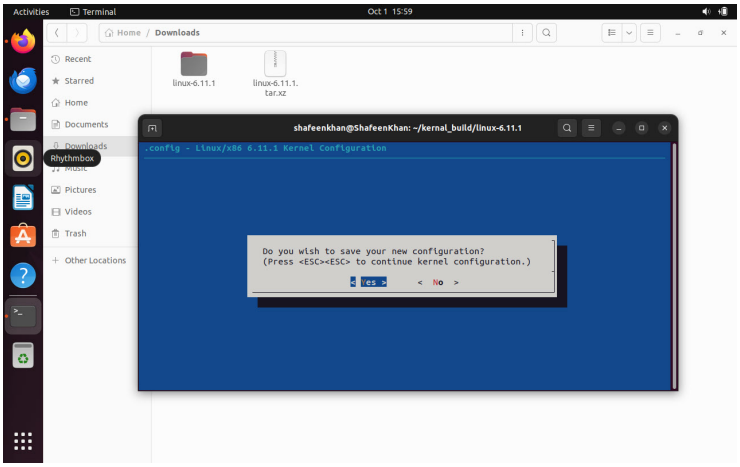

Figure 31

Step 8: Compiling the Kernel

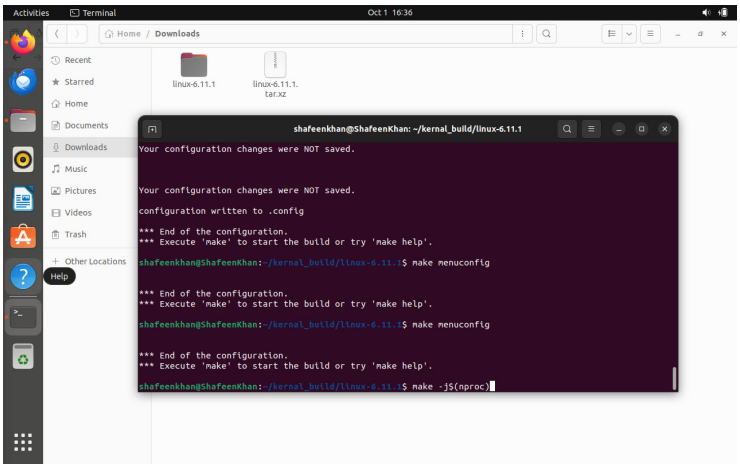

Figure 32: Using make -j$(nproc) , the command after make is used to utilize all cores for processing resulting in faster compilation

0.22 Compiling the Kernel - Continued

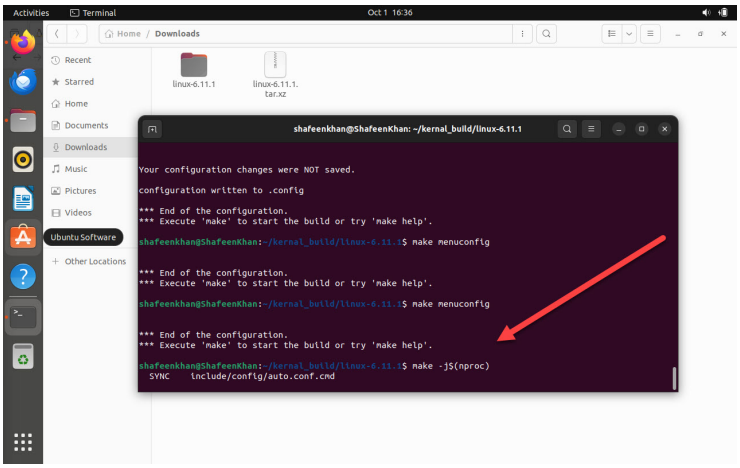

Figure 33

0.23 Compiling the Kernel - Continued

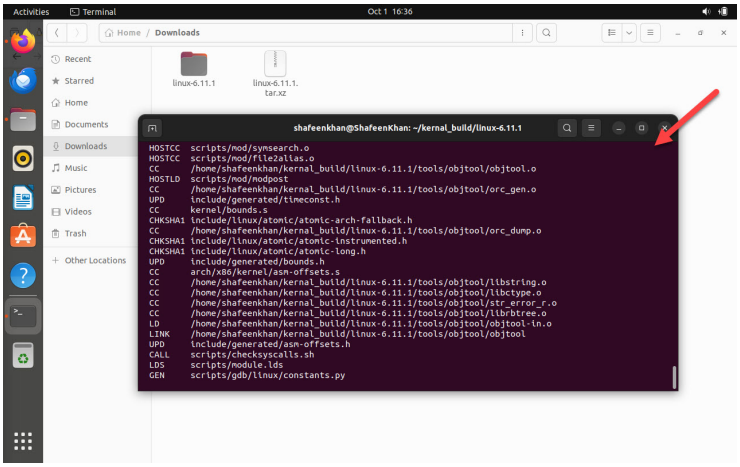

Figure 34

0.24 Compiling the Kernel - Continued

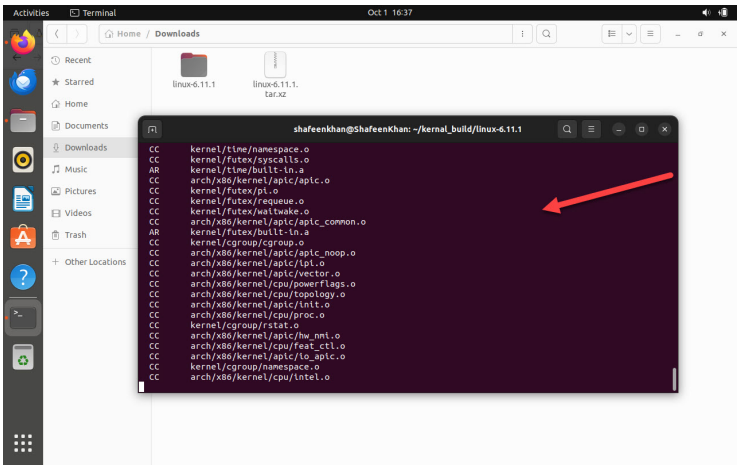

Figure 35

0.25 Compiling the Kernel - Continued

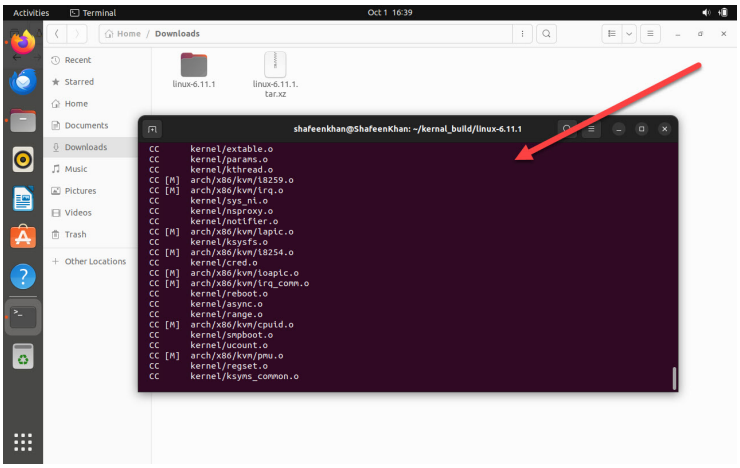

Figure 36

0.26 Compiling the Kernel - Continued

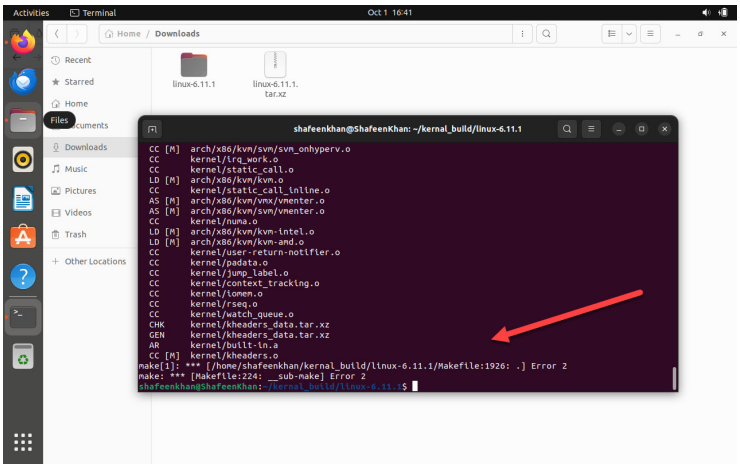

Figure 37: it gives this error to fix this we run to commands

Step 9: ERROR WHILE - Compiling the Kernel

This error is caused by security reasons from Linux operating system , it gives an error because installing custom linux it checks some predefined keys that

are linked to your current kernal hence the keys do not match which is why it throws an error , we are gonna use some commands to disable the predefined keys and use new signatures and certifications to make the compilation possible. ” scripts/config –disable SYSTEM_TRUSTED_KEYS: ” ” scripts/config –disable SYSTEM_REVOCATION_KEYS: ” These commands disable the kernel’s use of trusted and revoked cryptographic keys for module and firmware verification during the build process.

0.27 Compiling the Kernel - Continued

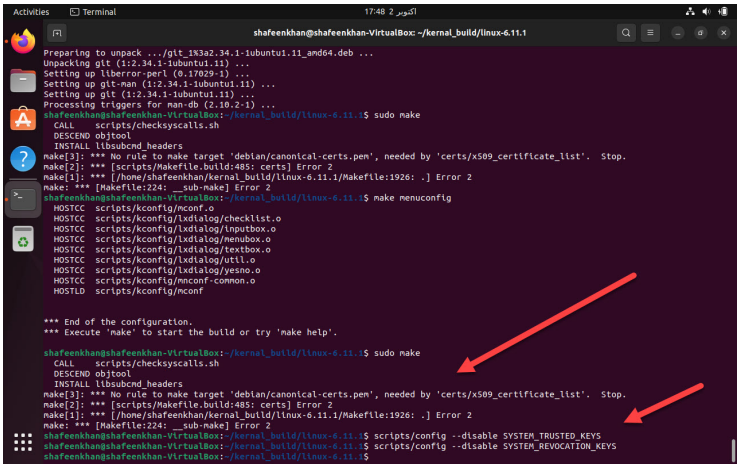

Figure 38: The above commands fixes the errors

1 Step 10: Installing the Kernel Modules

1.1 Compiling the Kernel - Continued

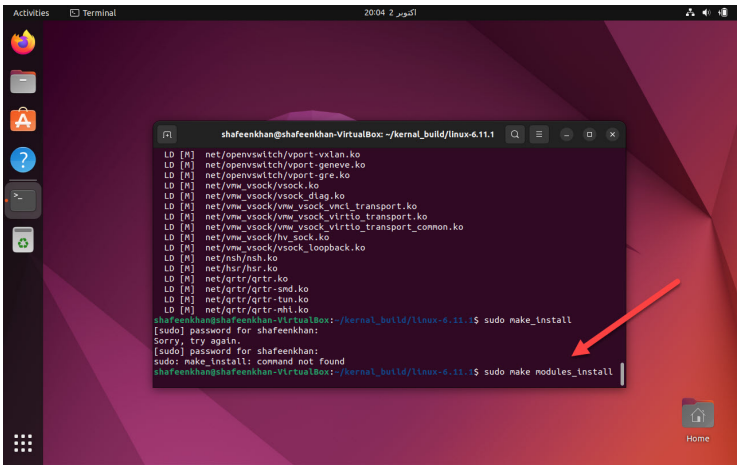

Figure 39: This installs kernel modules to /lib/modules/¡kernel-version¿.

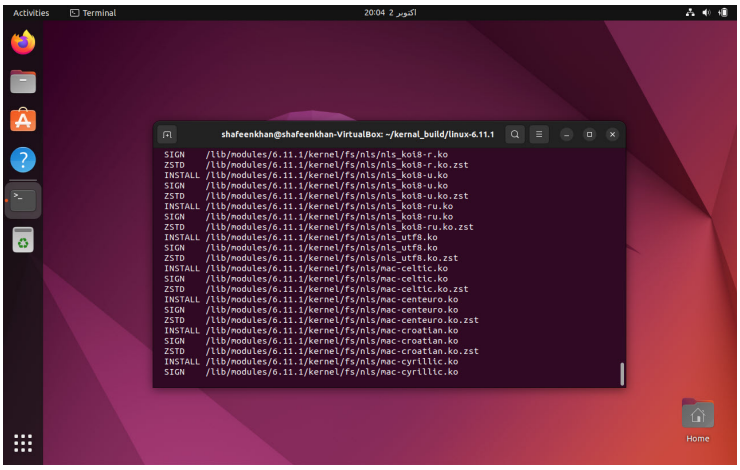

Figure 40

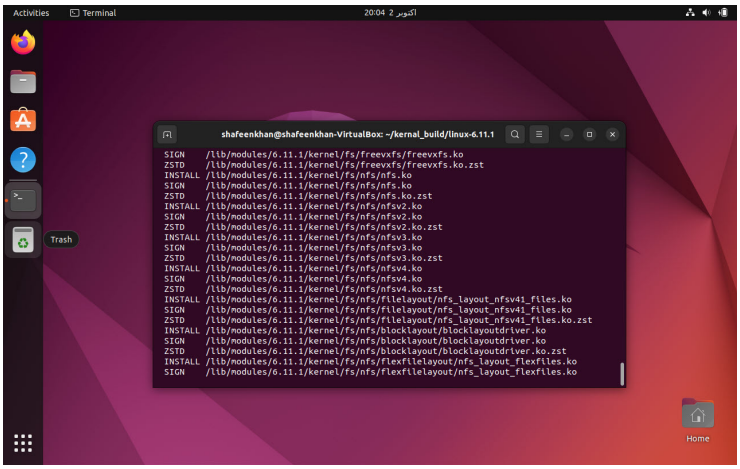

Figure 41

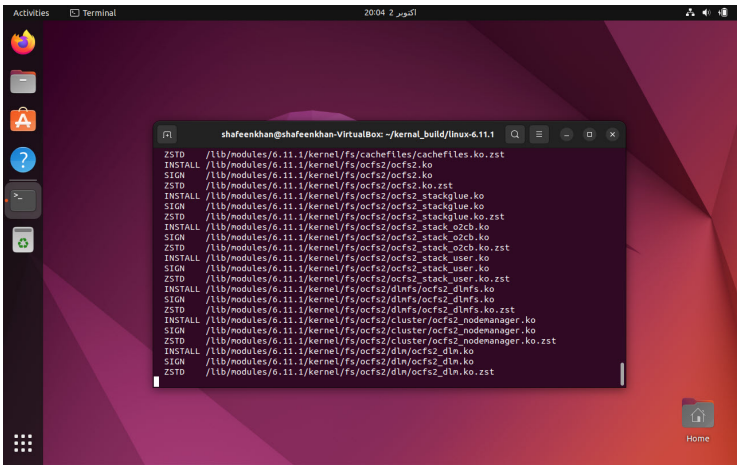

Figure 42

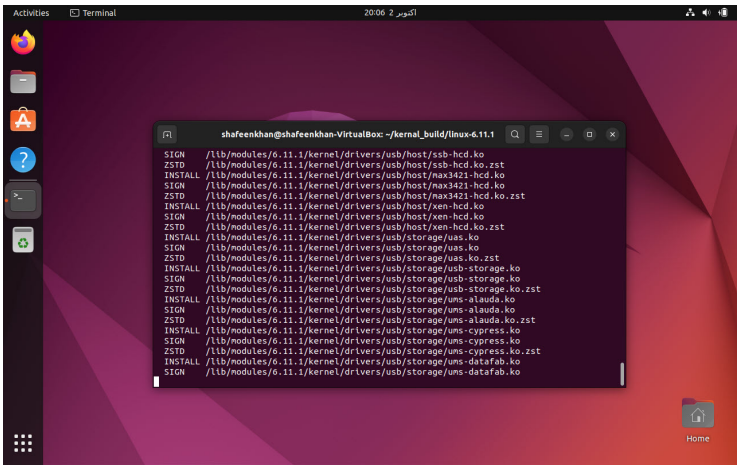

Figure 43

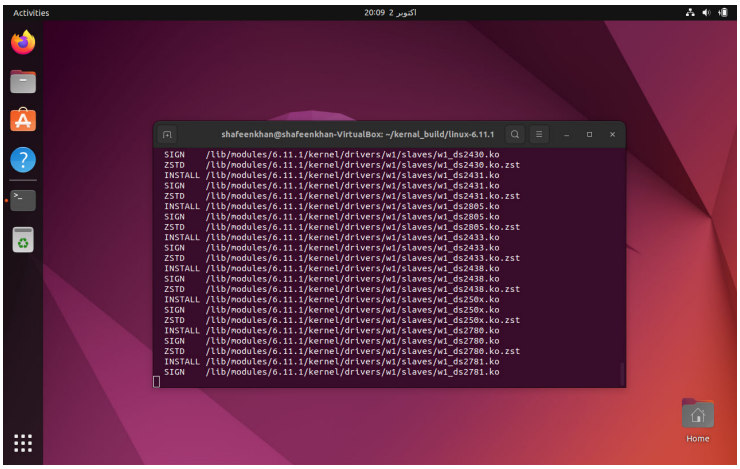

Figure 44

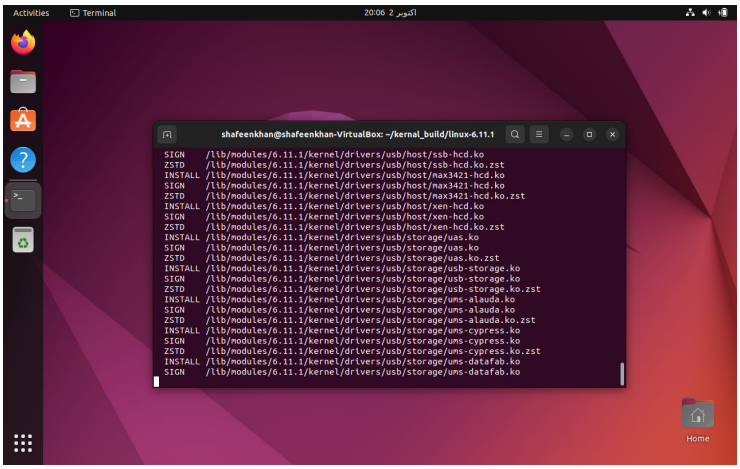

Figure 45: Completed.

2 Step 11: Installing the New Kernel

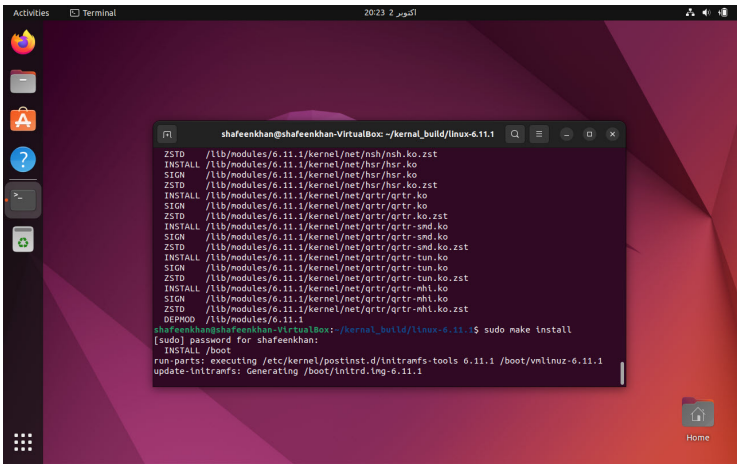

Figure 46: This copies the kernel image (vmlinuz), system map, and config file to /boot

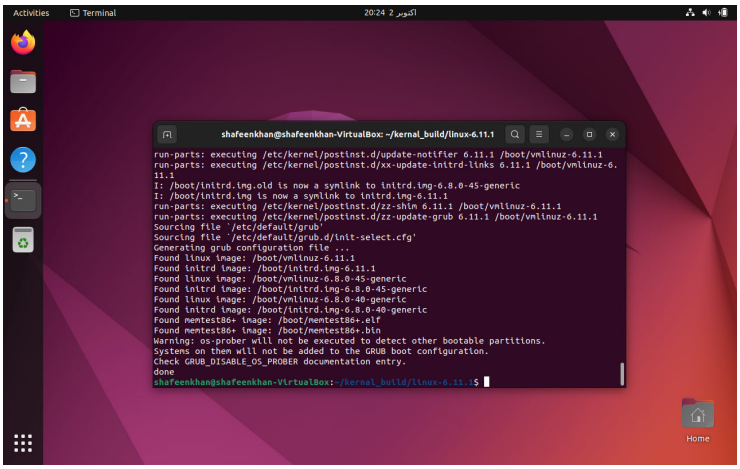

Figure 47: Completed.

3 Step 12: Updating the Bootloader

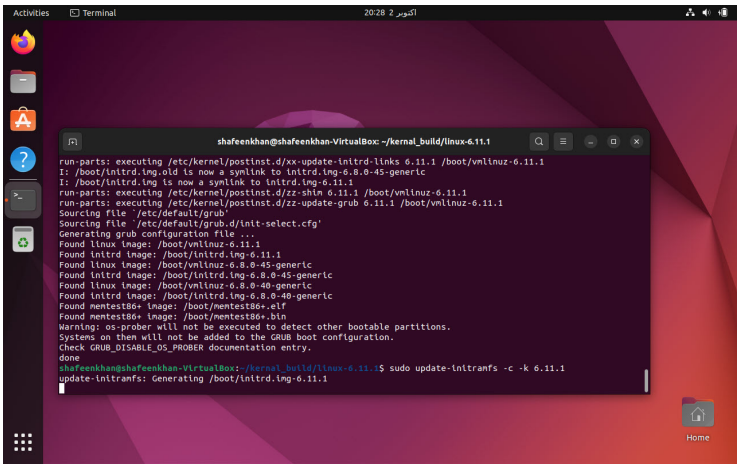

Figure 48: update-initramfs creates a new initrd image for the new kernel.

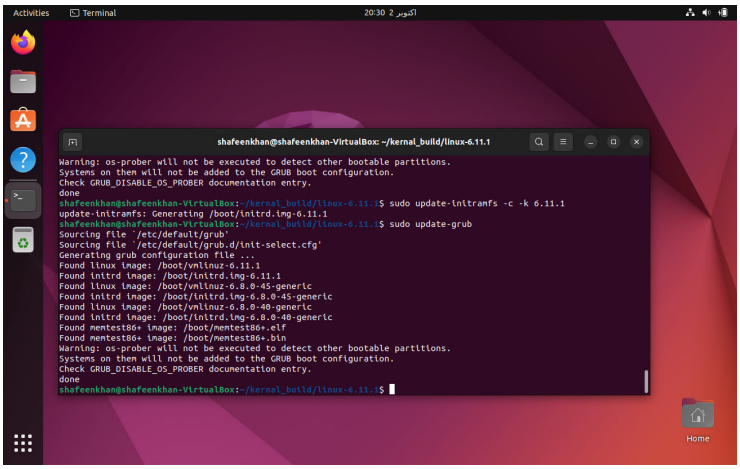

Figure 49: update-grub scans for kernels in /boot and updates the GRUB menu.

4 Step 13 : Step 9: Rebooting and Verify the new kernal

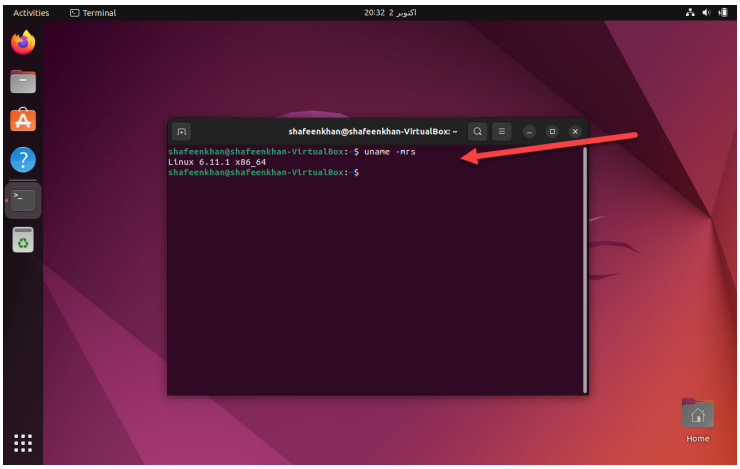

Figure 50: uname -r to display

5 Challanges Faced :

One of the challenges I faced was the **kernel compilation time**. The process of running ‘make‘ took significantly longer than expected, especially since I was working in a virtual machine with limited resources. Watching it compile, I had

to be patient, and it made me realize how resource-intensive this task is, even for modern systems.

Another challenge was the **kernel configuration** itself. When running ‘make menuconfig‘, I was overwhelmed by the sheer number of options available. Deciding what to enable or disable felt like navigating a maze, and I had to be careful not to accidentally disable something critical. This added complexity made the whole process more time-consuming than I initially anticipated.