COMPILING THE LATEST STABLE LINUX KERNEL AND DUAL BOOTING IT WITH THE CURRENT VERSION

Submitted by:

Panasa Teja
B191143CS

Problem Statement:
Download the latest stable Linux kernel from kernel.org, compile it, and dual boot it with your current Linux version. Your current version as well as the new version should be present in the grub-menu.

Abstract:
This report contains the complete process of compiling a latest stable Linux kernel and installing it alongside the current kernel version in Ubuntu 21.04 operating system. It demonstrates the process right from downloading the source code of the latest stable kernel v. 5.13.13, to dual booting it with the kernel v. 5.11.0-31-generic i.e., the compiled kernel can be loaded from the GRUB bootloader. Throughout the process, I have not used the install.sh script, instead, I have done everything manually. So, if you are using the commands make and make install then you might need to add and skip a few steps.

Table of Contents

1. Introduction

1.1 What is an Operating System?

An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs. Your computer's operating system (OS) manages all of the software and hardware on the computer. Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory, and storage. The operating system coordinates all of this to make sure each program gets what it needs. There are various types of Operating systems and Linux is one of them. Linux is an open-source operating system originally developed by Linus Torvalds.

1.2 What is a Linux Kernel?

The Linux® kernel is the main component of a Linux operating system (OS) and is the core interface between a computer’s hardware and its processes. It communicates between thee 2, managing resources as efficiently as possible. The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it’s a phone, laptop, server, or any other kind of computer.

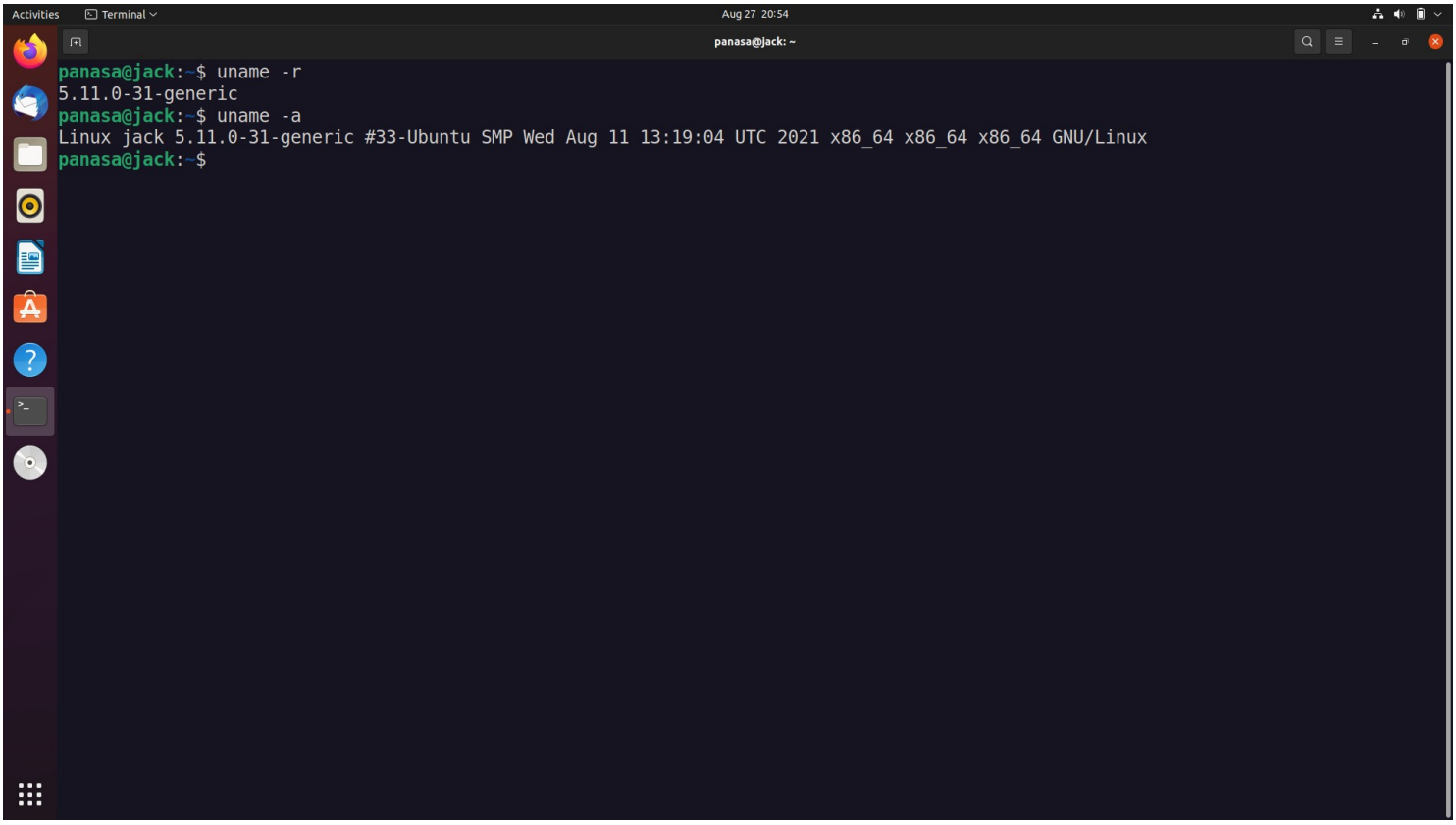
1.3 What does a kernel do?

The kernel has 4 jobs:
1. Memory management: Keep track of how much memory is used to store what, and where.
2. Process management: Determine which processes can use the central processing unit (CPU), when, and for how long.
3. Device drivers: Act as mediator/interpreter between the hardware and processes.
4. System calls and security: Receive requests for service from the processes.

System specifications:

Kernel name	Linux
Host name	jack
Kernel release	5.11.0-31-generic (kernel for the default Ubuntu 21.04 OS Image in Virtual box)
Kernel version	#33-Ubuntu SMP Wed Aug 11 13:19:04 UTC 2021
Machine, processor, hardware platform	x86_64
Operating system	GNU/Linux

The above details can be obtained from the command **uname -a:**



2. Explanation and Methodology

Linux kernel compilation is majorly divided into five sections, downloading the kernel source code, configuring the kernel, compiling the source code, installing the kernel and finally updating the grub. Let us get into each one of them in detail.

2.1 Downloading the kernel source code

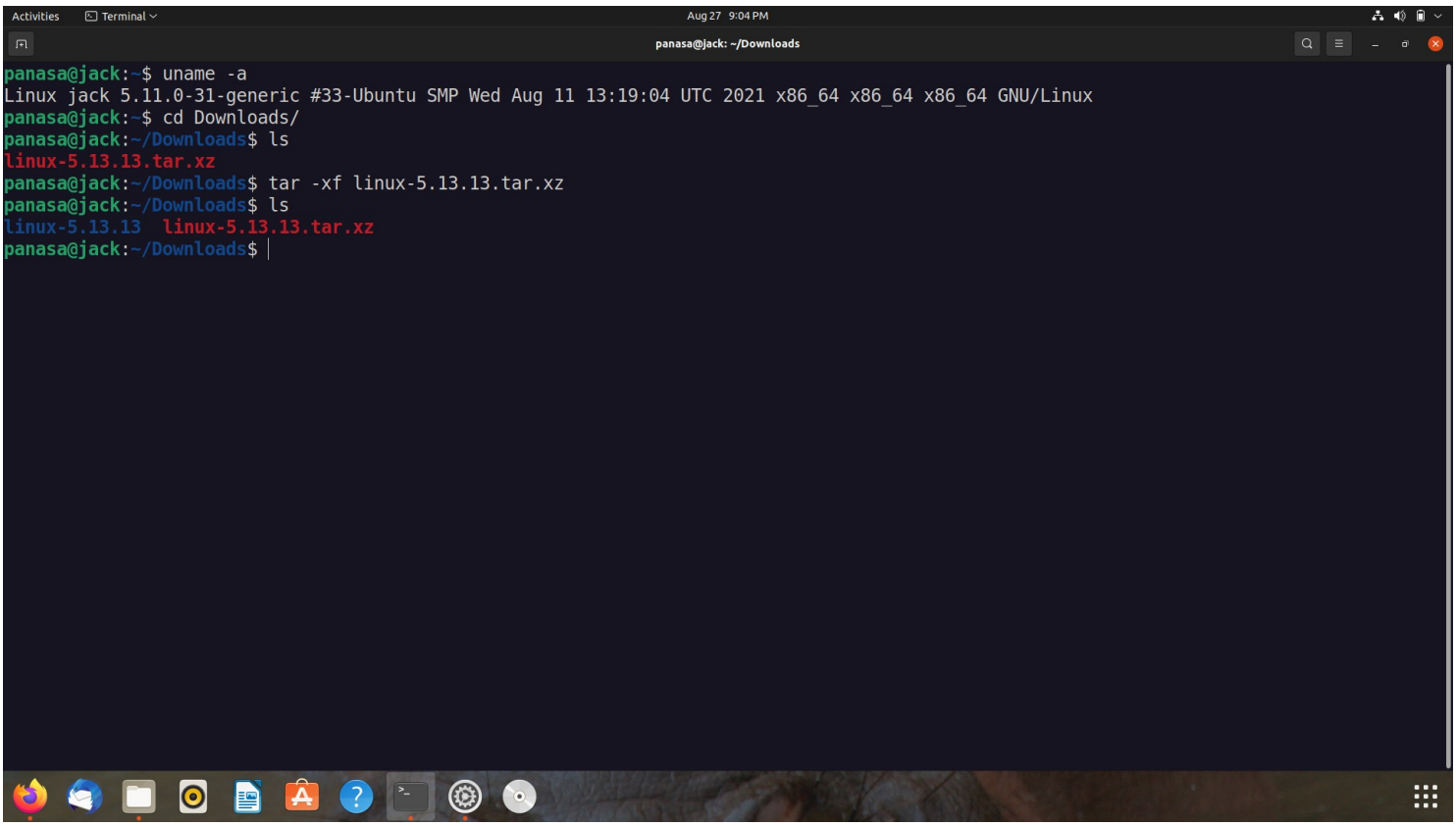
As stated earlier, Linux is an opensource operating system, which means the source codes are public and can be used, modified, distributed by everyone. The entities are termed as Linux distributions, shortly referred to as a "distro". Ex. Ubuntu, Monjaro, Pop!_OS, Fedora, CentOS, Arch Linux, Kali Linux etc.

These distros come with a default stable kernel within them. But what we are looking for is to install another kernel version manually on one such distro.
At the time of making this report the stable kernel version was 5.13.13

Download the source code using **wget:**

$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.13.13.tar.xz

Decompress the archive using **tar:**
$ tar -xf linux-5.13.12.tar.xz



2.2 Kernel configuration

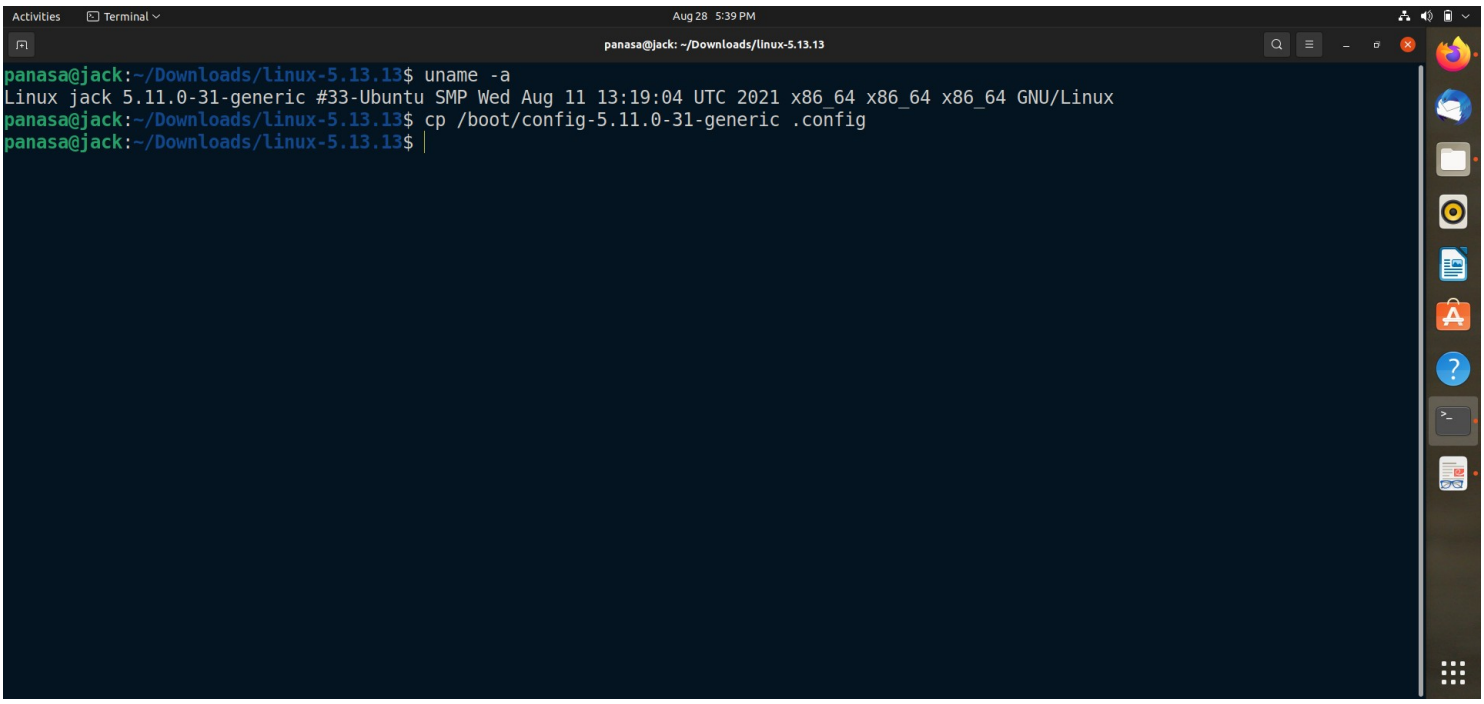
Before we actually compile the kernel, we must first configure which modules to include. There is actually a really easy way to do this. With a single command, you can copy the current kernel's config file and then use the tried and true menuconfig command to make any necessary changes. To do this

(a) Copy the existing kernel config file using the command:

Copy the existing kernel config file just in case you need to replace it in the event of a problem.

cp /boot/config-$(uname -r) .config

(Since the version might be slightly different, give uname -r).

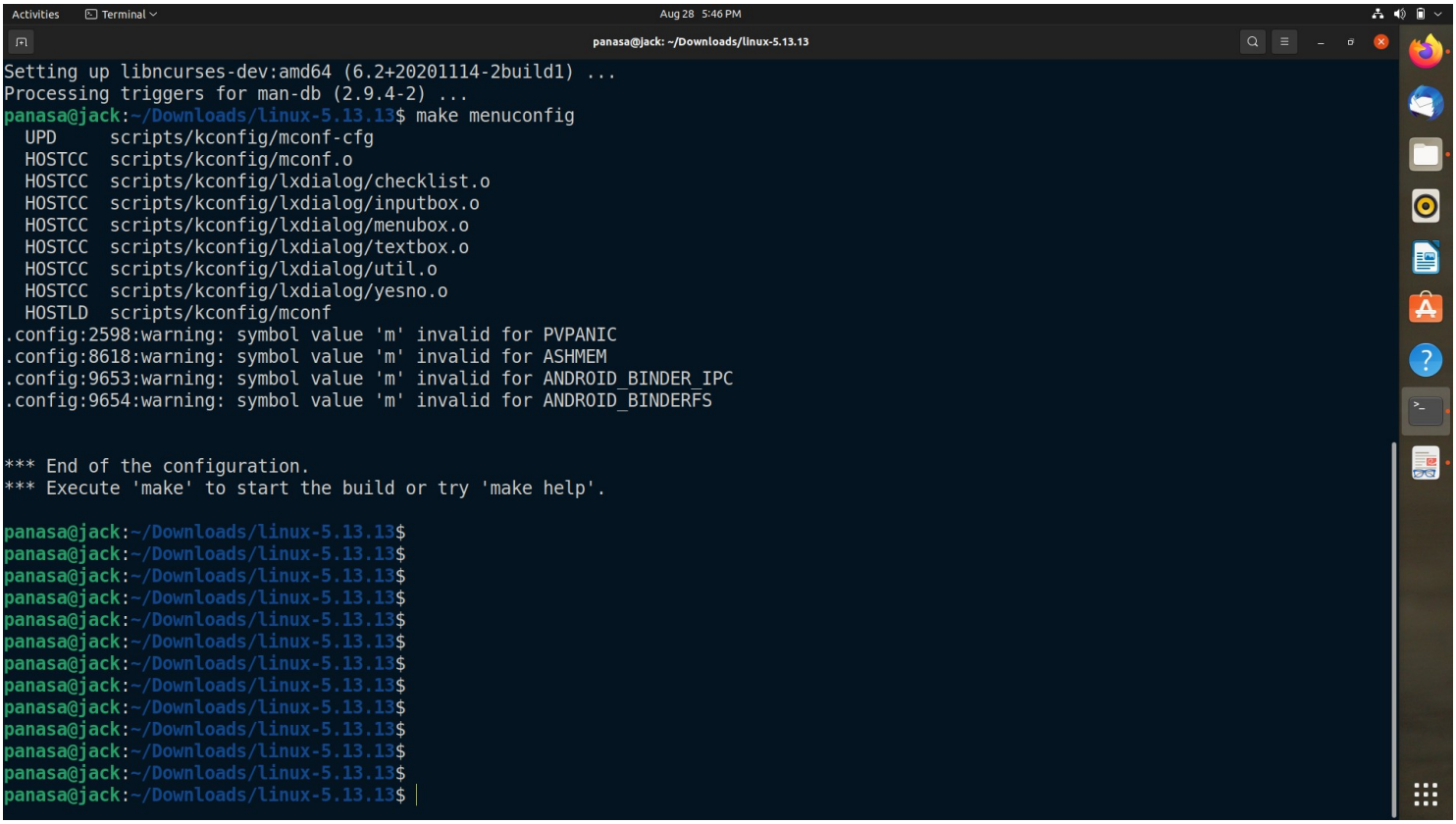


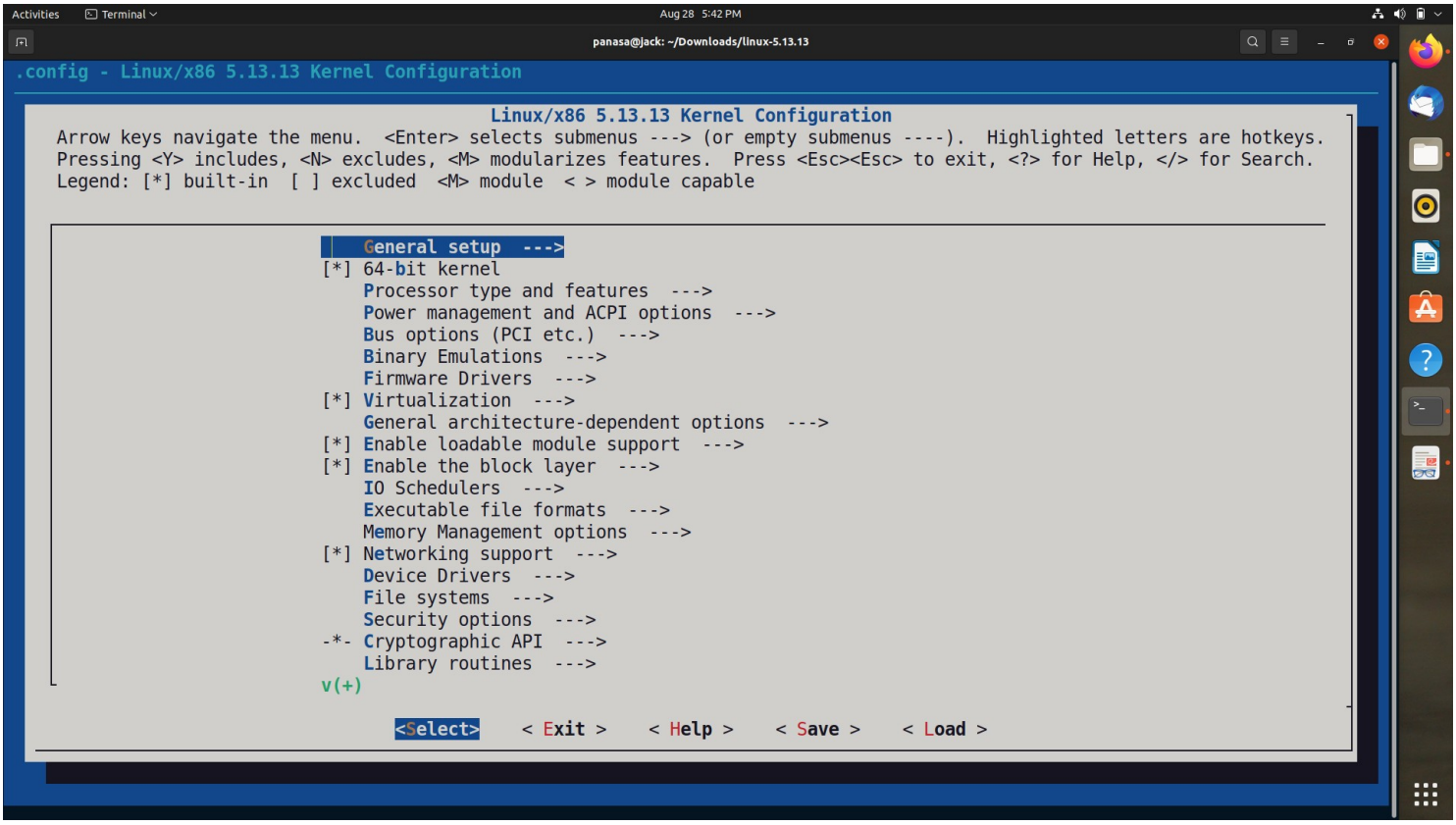
(b)To configure the kernel use the command:

Now, before we compile the kernel, we need to make sure we configure it. This is because you need to specify which modules you want to be installed and which to not. You have the option of just sticking with the configuration offered by the kernel itself or to alter the existing configuration or add your own, we will use the default configuration. Run the following command,

make menuconfig

This command will open up a configuration tool that allows you to go through every module available and enable or disable what you need or don’t need.

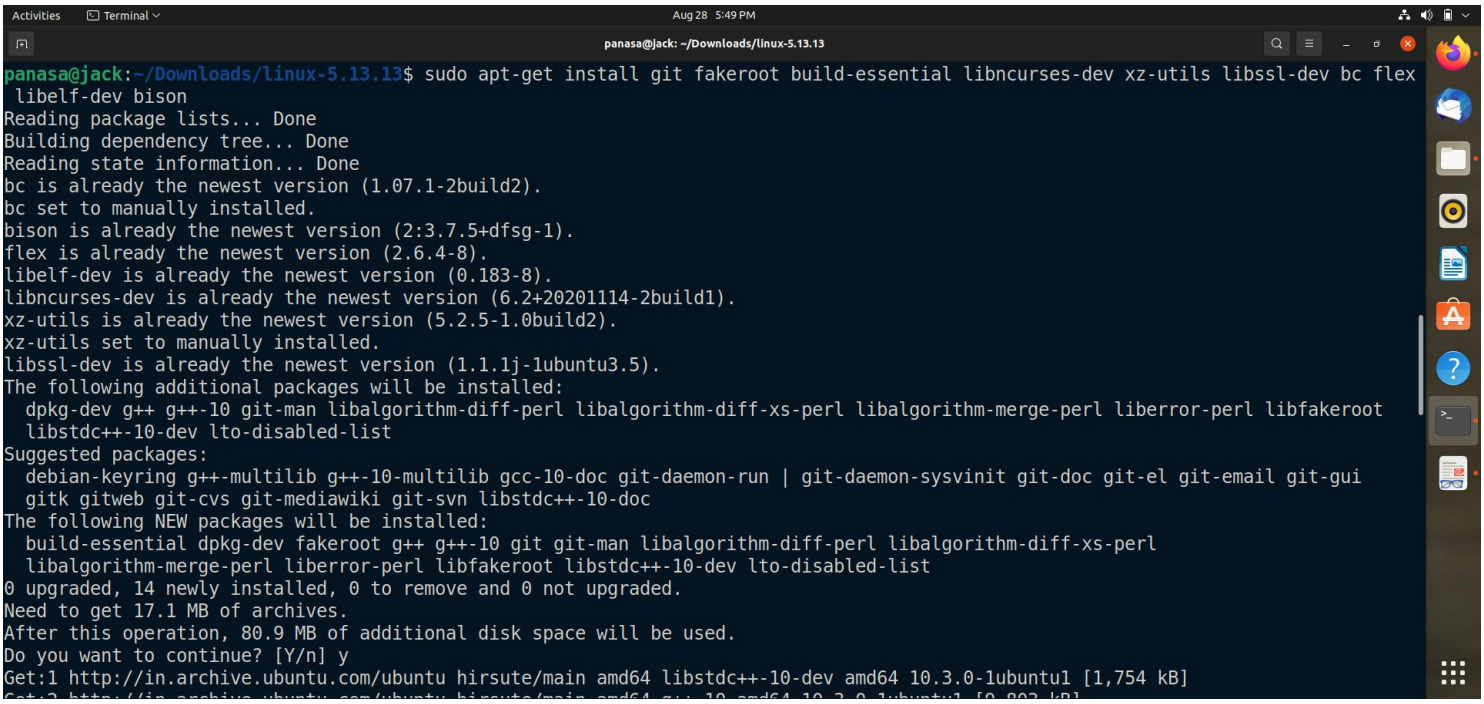

Activities
Terminal
Aug 28 5:46 PM
panasa@jack: ~/Downloads/linux-5.13.13
Setting up libncurses-dev:amd64 (6.2+20201114-2build1) ...
Processing triggers for man-db (2.9.4-2) ...
panasa@jack:~/Downloads/linux-5.13.13$ make menuconfig
UPD scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTLD scripts/kconfig/mconf
.config:2598:warning: symbol value 'm' invalid for PVPANIC
.config:8618:warning: symbol value 'm' invalid for ASHMEM
.config:9653:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
.config:9654:warning: symbol value 'm' invalid for ANDROID_BINDERFS
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
panasa@jack:~/Downloads/linux-5.13.13$


Activities
Terminal
Aug 28 5:42 PM
panasa@jack: ~/Downloads/linux-5.13.13
.config - Linux/x86 5.13.13 Kernel Configuration
Linux/x86 5.13.13 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] 64-bit kernel
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Binary Emulations --->
Firmware Drivers --->
[*] Virtualization --->
General architecture-dependent options --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
IO Schedulers --->
Executable file formats --->
Memory Management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Security options --->
-*- Cryptographic API --->
Library routines --->
v(+)
<Select> < Exit > < Help > < Save > < Load >

2.2.1 Installing the requirements

In order to compile the kernel, we'll need to first install a few requirements. This can be done with a single command:

sudo apt-get install git fakeroot build-essential libncurses-dev xz-utils libssl-dev bc flex libelf-dev bison



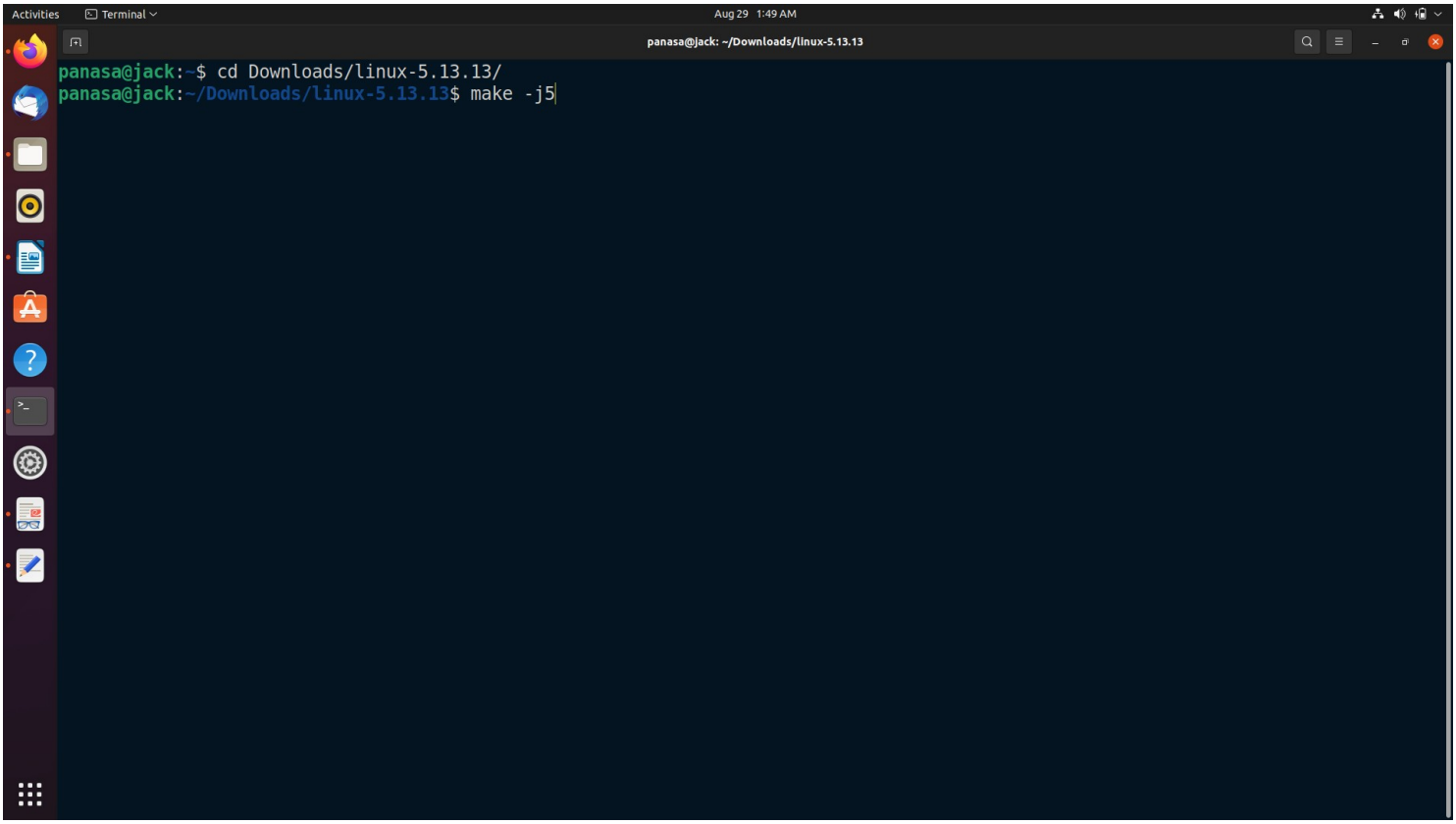
2.3 Kernel compilation and installation

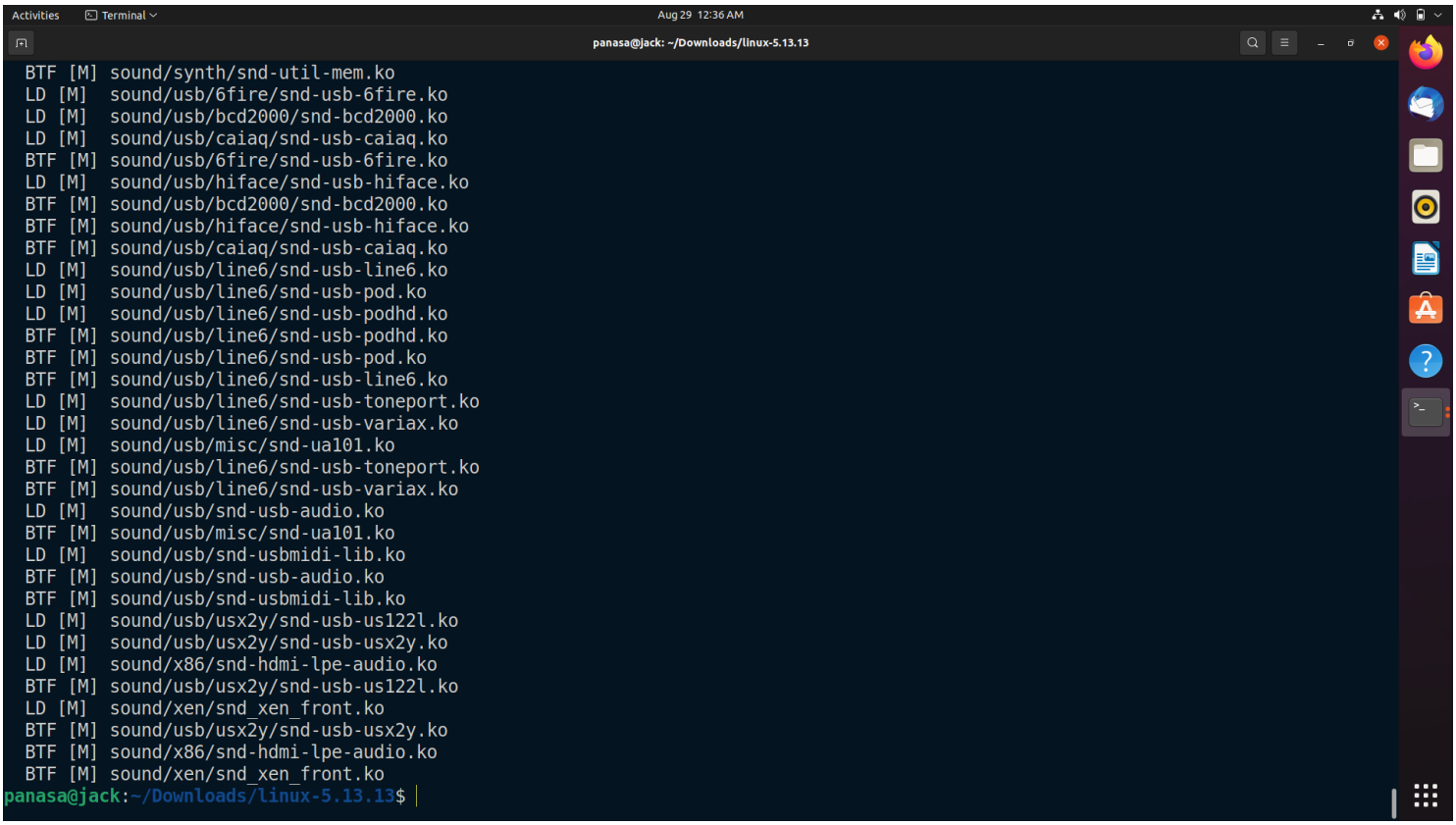
Now it's time to actually compile the kernel. The first step is to compile using the make command.Here we can customize the number of threads you are allocating for the compile/install process by using the -j parameter.

make -j 5

The -j parameter is to customize the number of threads you are allocating for the compile/install process.

So issue make and then answer the necessary questions (optional if the configuration was saved in the text-based menu). The questions asked will be determined by what kernel you're upgrading from and what kernel you're upgrading to. The compilation process will mostly look like this -

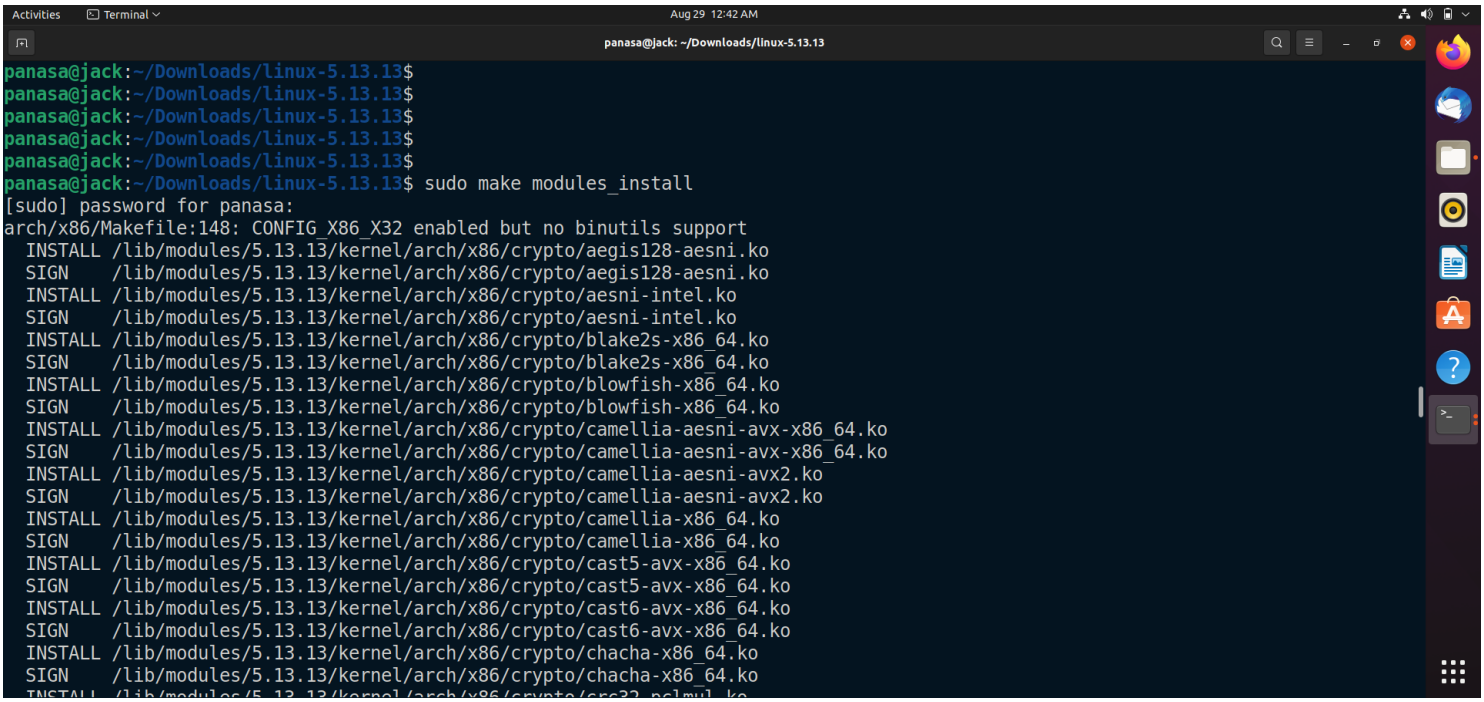


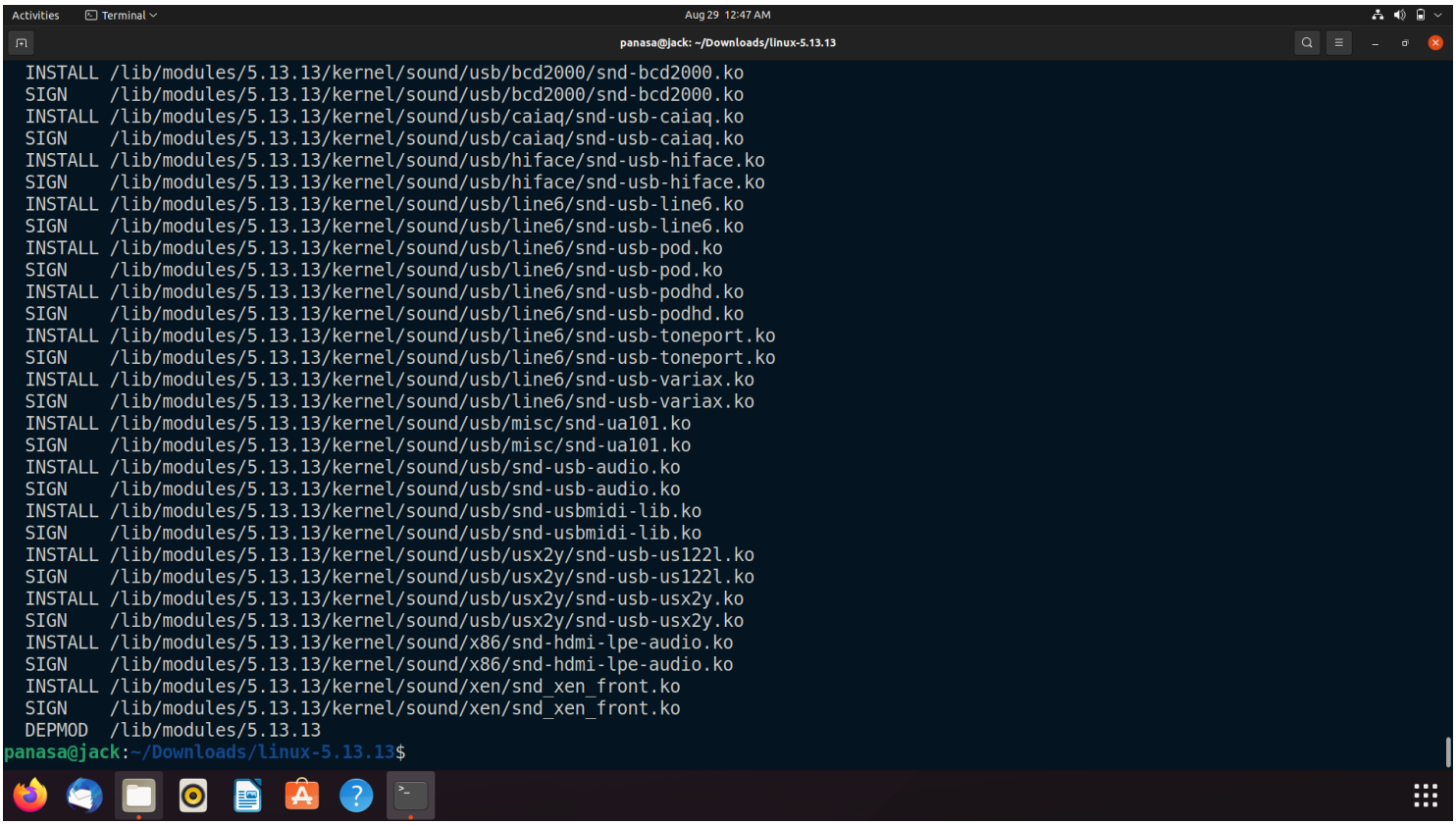


On completing this, you can install the modules you've enabled with the command:

2.4 sudo make modules_install

The installation process will mostly look like this -

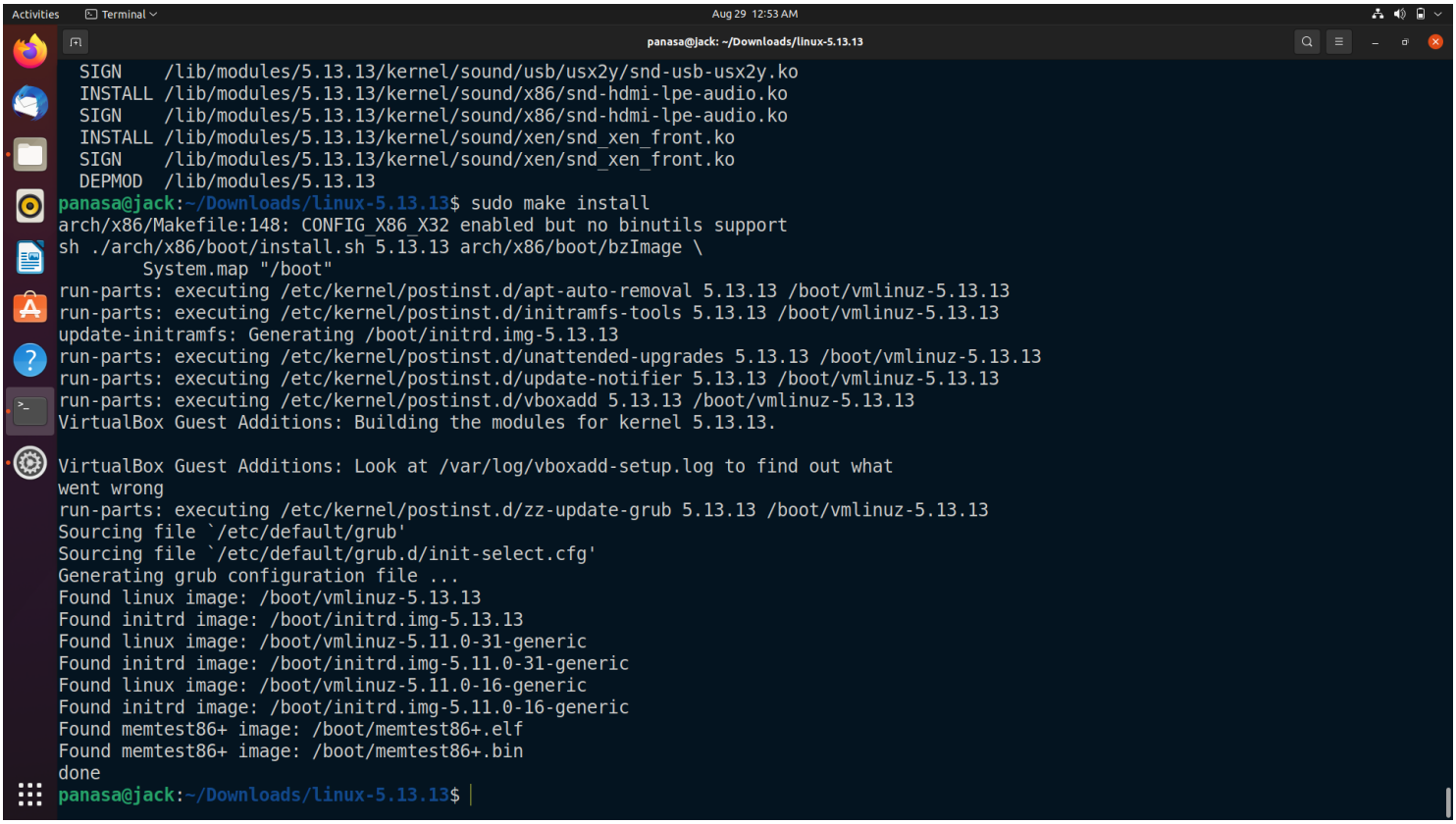




Now we install the kernel with the command:

2.5 sudo make install

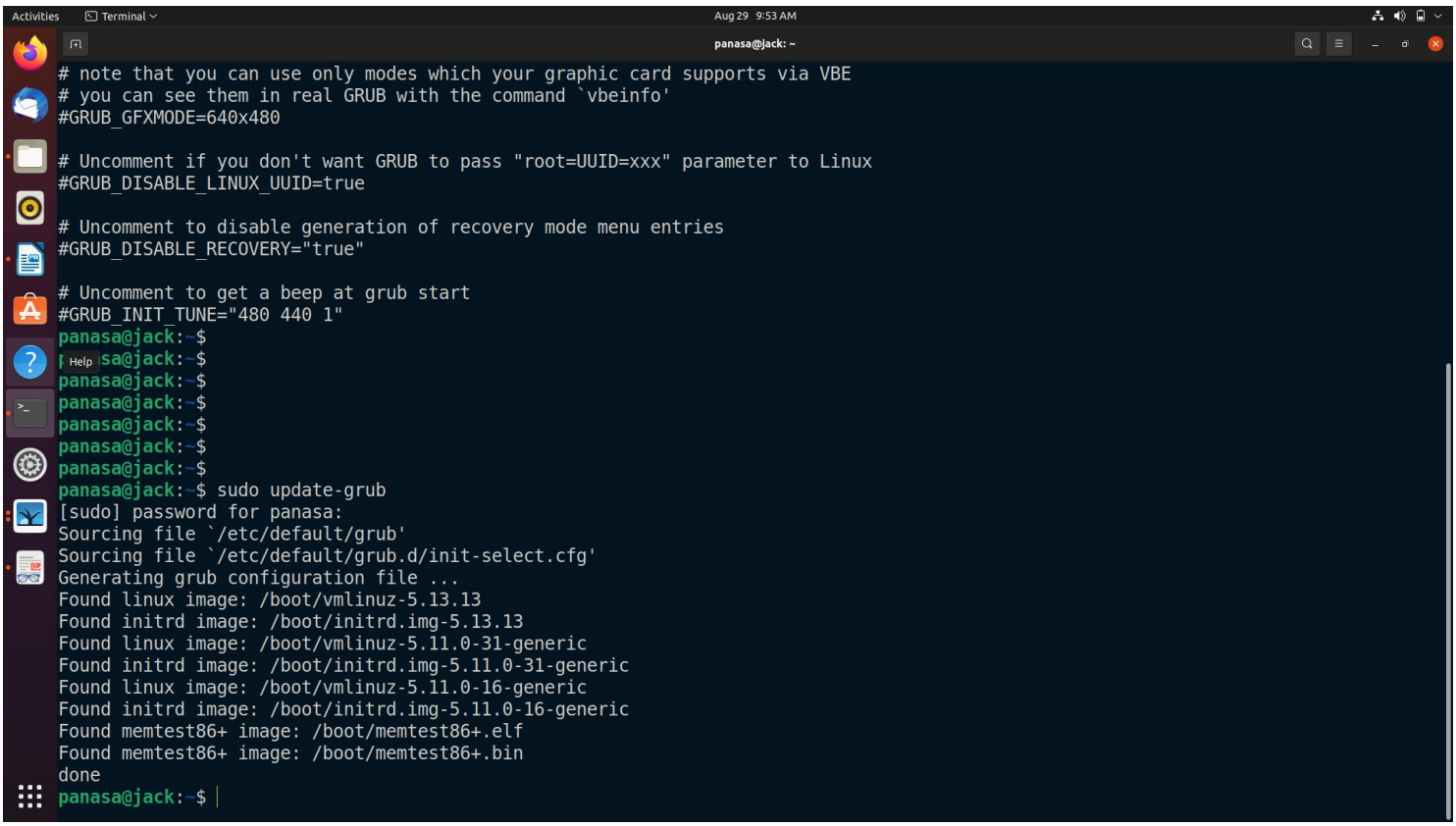
The above process in itself updates the initramfs, which is responsible to look for kernels in the /boot/ folder and add them to the grub's configuration file, the command "make install" will also update the grub.cfg by default. So we don't need to manually edit the grub.cfg file. (GRUB is a multi-boot bootloader for most linux distros).



2.6 Update Grub file

Update this file which is in the directory : sudo gedit */etc/default/grub*

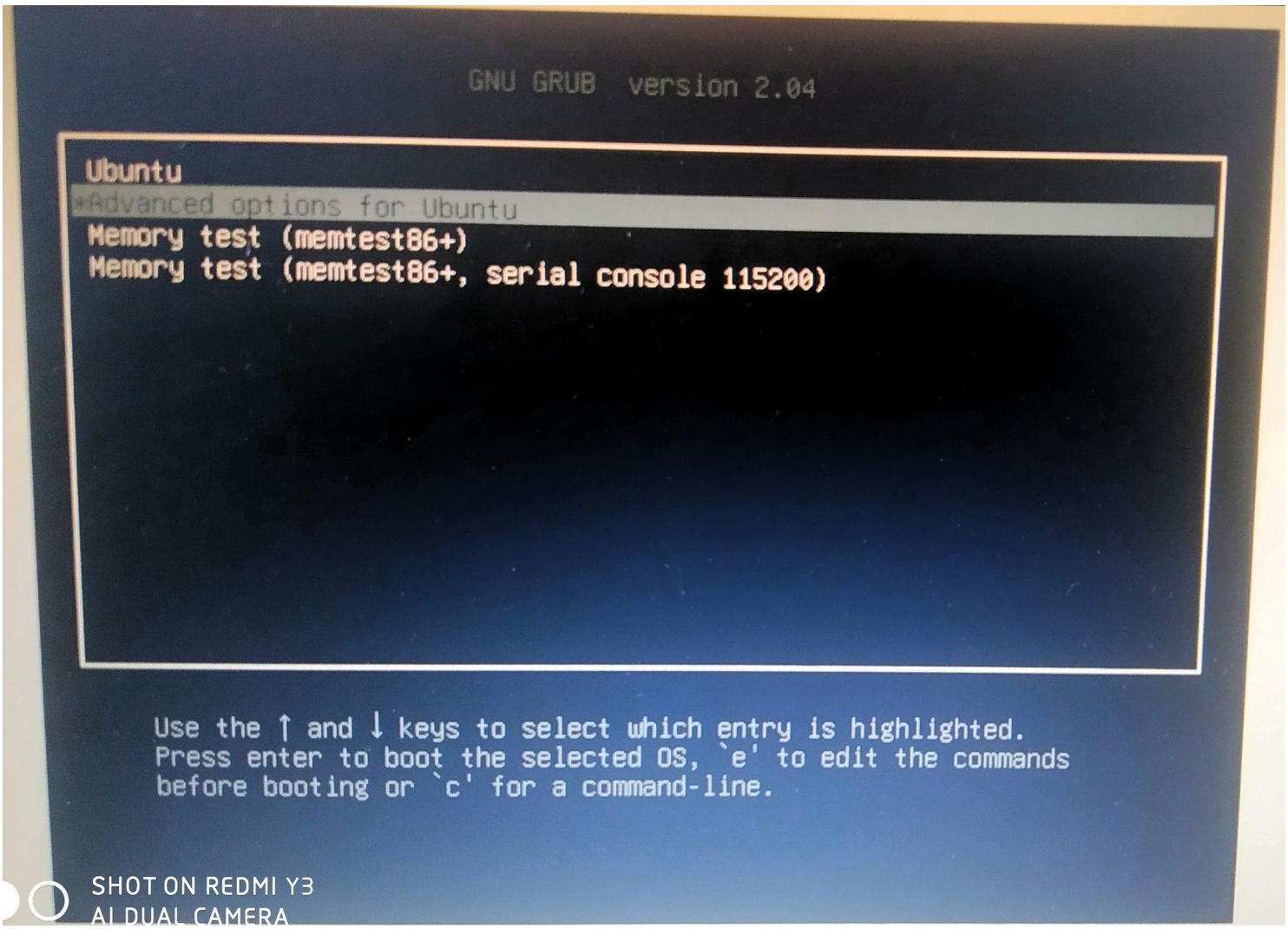
make GRU_TIMEOUT_STYLE=menu
make GRUB_TIMEOUT= k (where k>=10) for displaying purpose



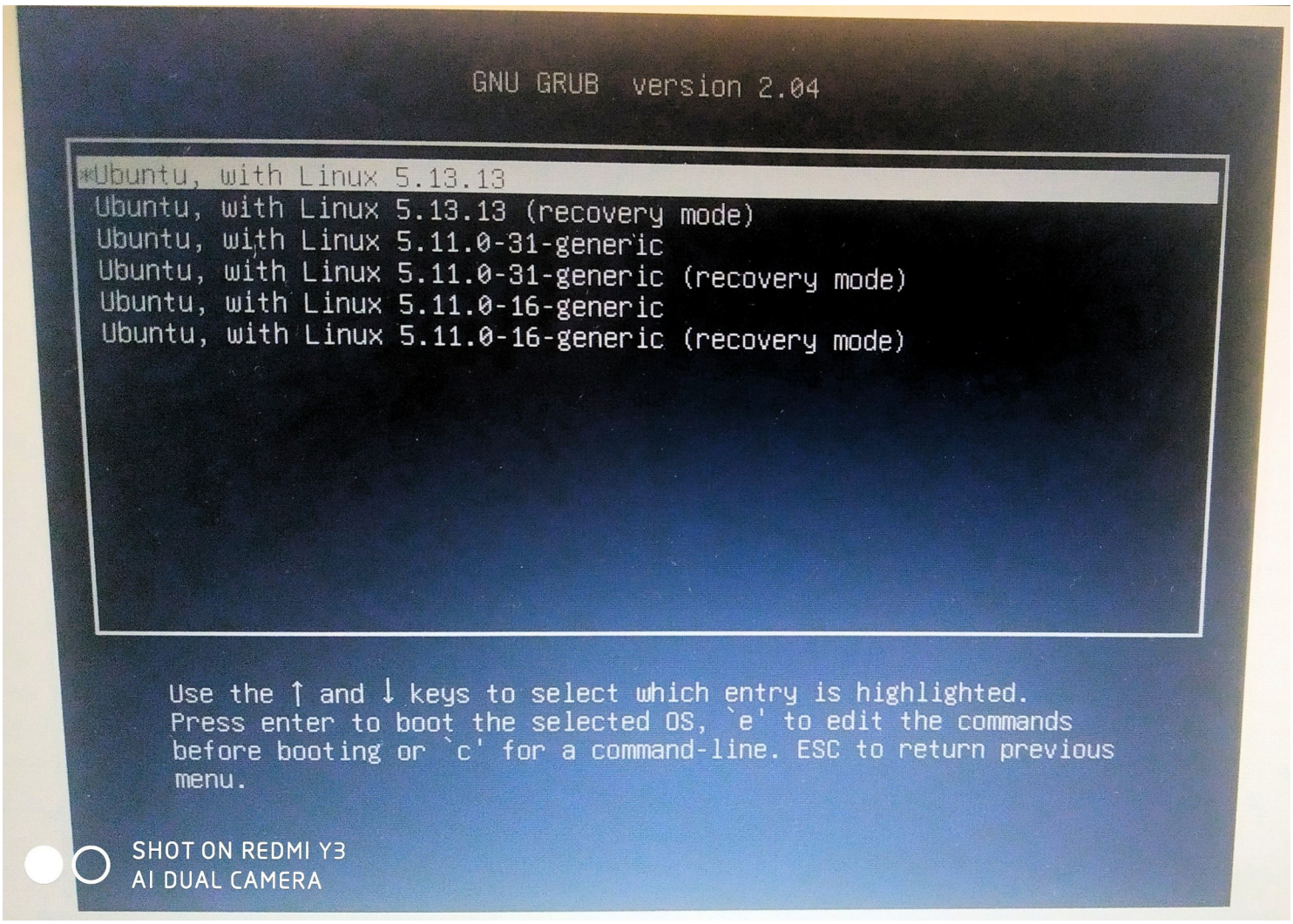
2.6 Reboot

You should ideally reboot the virtual machine after the completion of this stage.

Reboot your machine and hold the right shift key during boot process to access the GRUB bootloader
menu, which will look like this -

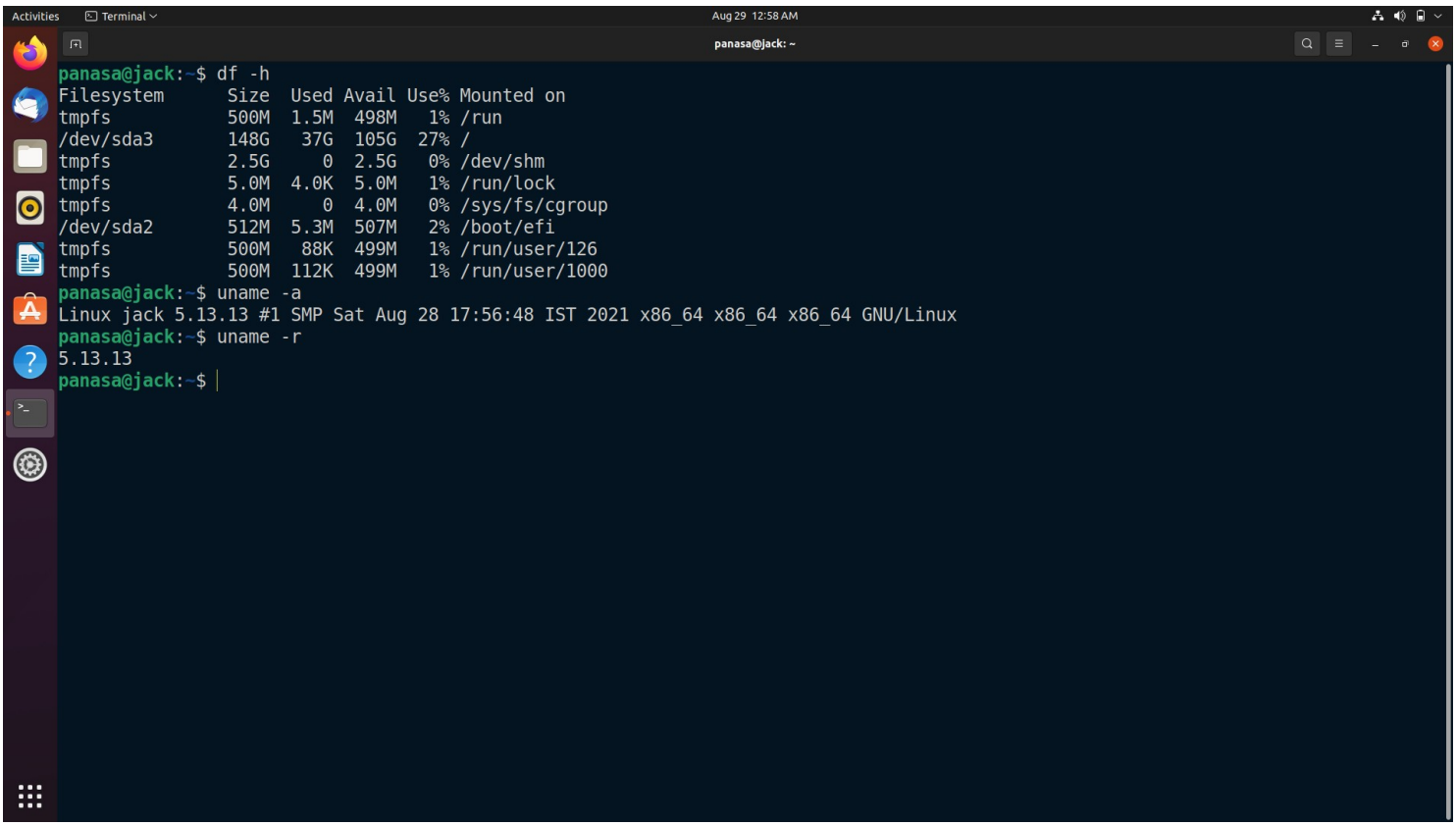


Choose “Advanced options for Ubuntu” and you will be directed to a menu with the list of kernels available (default as well as custom ones that have been installed). Since we installed the latest version , this will be the first option.



To show that the latest kernel is installed, run the command from the terminal:

uname -r



3. Conclusion

We have successfully compiled and installed a new kernel alongside our current kernel.

4. Bibliography

www.kernel.org
courses.linuxchix.org
www.linfo.org
linux.die.net
tldp.org
titanwolf.org
www.linuxquestions.org
askubuntu.com
unix.stackexchange.com
opensource.com
linuxdocs.org

**********THE END**********