

Problem Statement:

The objective of this project is to obtain and compile the most recent stable Linux kernel version available from the official kernel.org website. Subsequently, we aim to establish a dual-boot configuration on your Linux-based system, integrating the freshly compiled kernel alongside your existing Linux installation. The ultimate goal is to make both the current and new Linux kernel versions accessible and selectable via the GRUB bootloader menu during system startup.

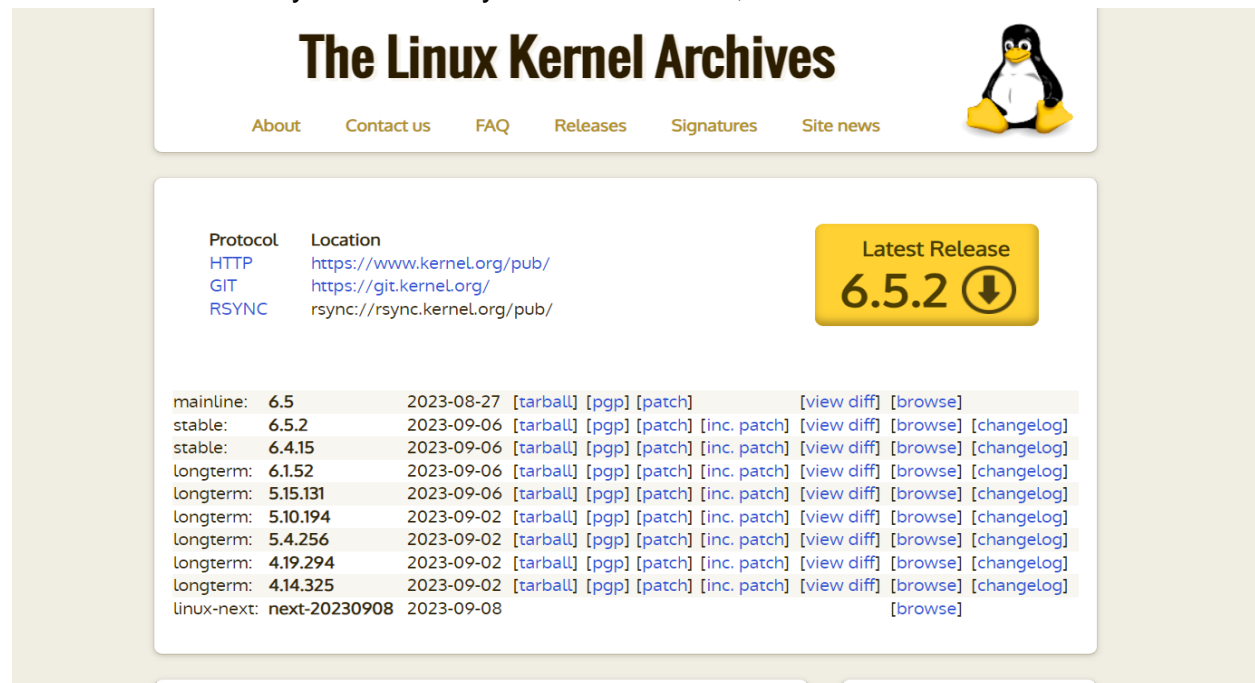
Methodology:

1. Download the Latest Stable Linux Kernel
2. Install Required Dependencies
3. Extract and Configure the Kernel
4. Compile the Kernel
5. Install Modules and kernel
6. Update GRUB
7. Reboot

1. Download the Latest Stable Linux Kernel

Visit the official Linux Kernel website at

<https://www.kernel.org/> to download the latest stable kernel source code. Identify the version you want to install, and note its version number.



The screenshot displays the 'The Linux Kernel Archives' website. At the top, there's a navigation bar with links: About, Contact us, FAQ, Releases, Signatures, and Site news. To the right is the Linux penguin logo. Below the navigation bar, a section titled 'Latest Release' features a large yellow button with the text '6.5.2' and a download icon. To the left of this button, there's a table with download links for HTTP, GIT, and RSYNC. Below this, a table lists various kernel versions and their release dates, along with links to download tarballs, view patches, and browse changelogs.

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

mainline:	6.5	2023-08-27	[tarball]	[pgp]	[patch]	[view diff]	[browse]
stable:	6.5.2	2023-09-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
stable:	6.4.15	2023-09-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	6.1.52	2023-09-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.15.131	2023-09-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.10.194	2023-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.4.256	2023-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.19.294	2023-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.14.325	2023-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
linux-next:	next-20230908	2023-09-08					[browse]

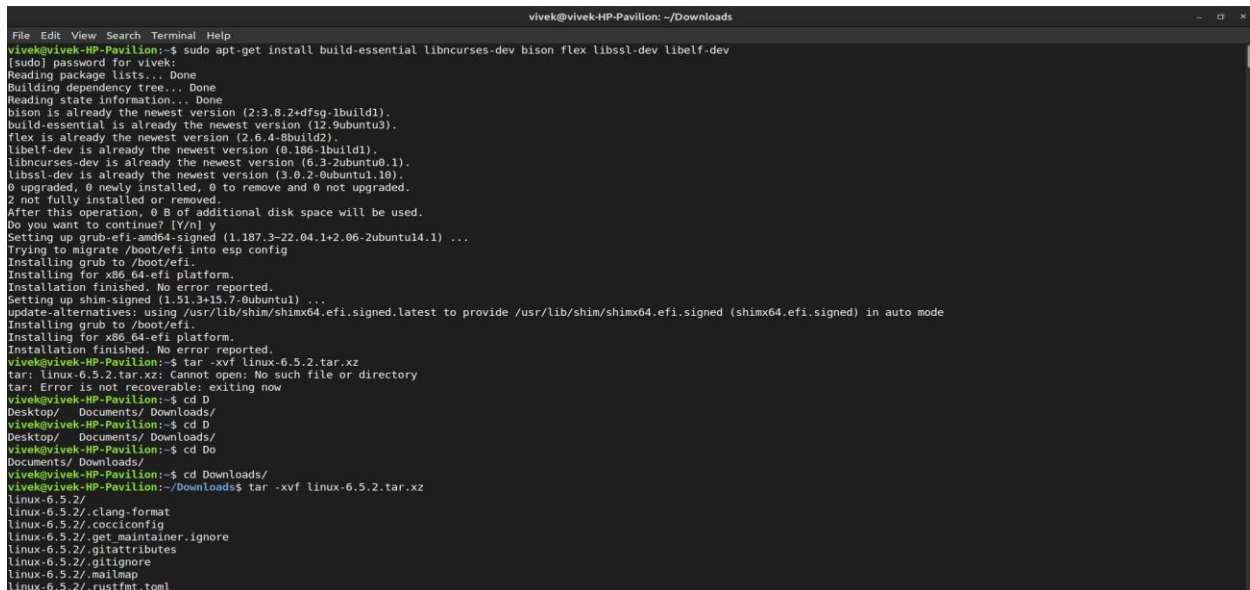
2. Install Required Dependencies

Before we can compile the Linux kernel, we will need to ensure that your system has all the necessary tools, libraries, and header files. These dependencies are crucial for a successful compilation process. Below is a breakdown of what each component is and why it's essential:

```

```
sudo apt-get install build-essential libncurses-dev bison flex libssl-dev libelf-dev
```

```



```
vivek@vivek-HP-Pavilion: ~/Downloads
File Edit View Search Terminal Help
vivek@vivek-HP-Pavilion:~$ sudo apt-get install build-essential libncurses-dev bison flex libssl-dev libelf-dev
[sudo] password for vivek:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
bison is already the newest version (2:3.8.2+dfsg-1build1).
build-essential is already the newest version (12.9ubuntu3).
flex is already the newest version (2.6.4-8build2).
libelf-dev is already the newest version (0.186-1build1).
libncurses-dev is already the newest version (6.3-2ubuntu0.1).
libssl-dev is already the newest version (3.0.2-0ubuntu10).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
2 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Setting up grub-efi-amd64-signed (1.187.3-22.04.1+2.06-2ubuntu14.1) ...
Trying to migrate /boot/efi into esp config
Installing grub to /boot/efi.
Installing for x86_64-efi platform.
Installation finished. No error reported.
Setting up shim-signed (1.51.3+15.7-0ubuntu1) ...
update-alternatives: using /usr/lib/shim/shimx64.efi.signed.latest to provide /usr/lib/shim/shimx64.efi.signed (shimx64.efi.signed) in auto mode
Installing grub to /boot/efi.
Installing for x86_64-efi platform.
Installation finished. No error reported.
vivek@vivek-HP-Pavilion:~$ tar -xvf linux-6.5.2.tar.xz
tar: linux-6.5.2.tar.xz: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
vivek@vivek-HP-Pavilion:~$ cd D
Desktop/ Documents/ Downloads/
vivek@vivek-HP-Pavilion:~$ cd D
Desktop/ Documents/ Downloads/
vivek@vivek-HP-Pavilion:~$ cd D
Documents/ Downloads/
vivek@vivek-HP-Pavilion:~$ cd Downloads/
vivek@vivek-HP-Pavilion:~/Downloads$ tar -xvf linux-6.5.2.tar.xz
linux-6.5.2/
linux-6.5.2/.clang-format
linux-6.5.2/.coqconfig
linux-6.5.2/.get_maintainer.ignore
linux-6.5.2/.gitattributes
linux-6.5.2/.gitignore
linux-6.5.2/.mailmap
linux-6.5.2/.rustfmt.toml
```

3.Extract and Configure the Kernel

Navigate to the directory where you downloaded the kernel source code and extract it. Then, configure the kernel.Replace `6.5.1` with the actual version number you downloaded.

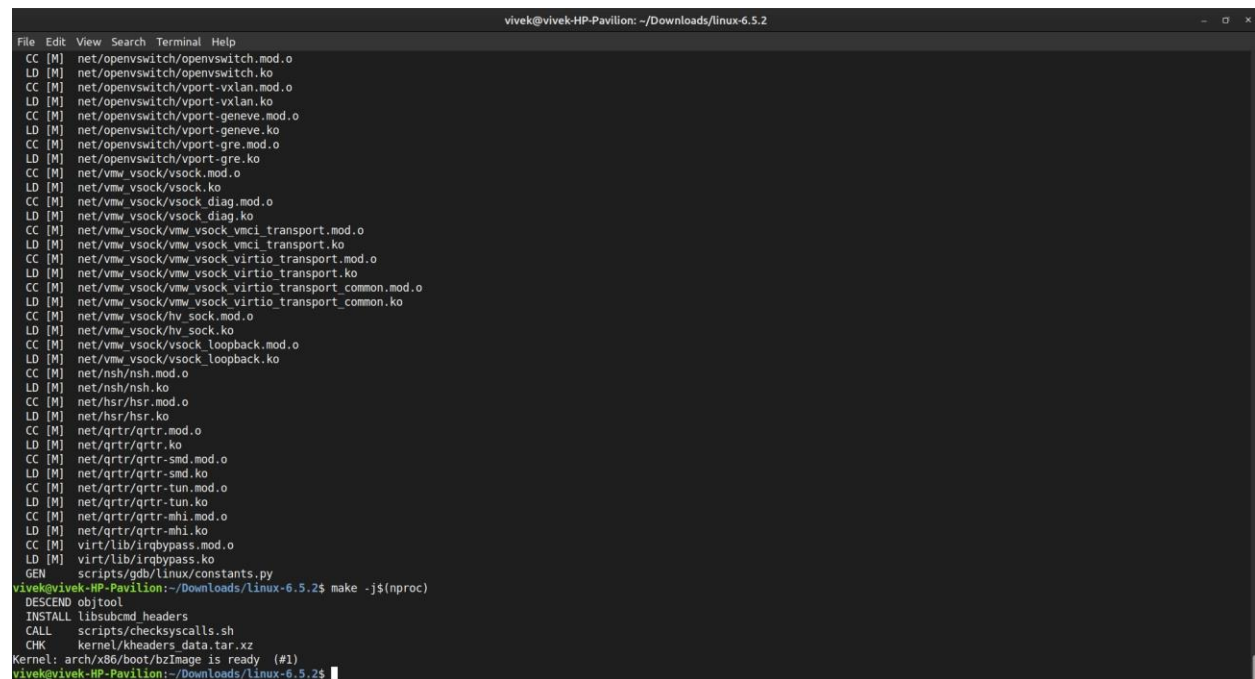
- Navigate to the Kernel Source Directory
- Extract the Kernel Source
- Configure the Kernel
- Save Configuration and Exit

```
tar -xvf linux-6.5.1.tar.gz
```

```
Installing grub to /boot/efi.  
Installing for x86_64-efi platform.  
Installation finished. No error reported.  
vivek@vivek-HP-Pavilion:~$ tar -xvf linux-6.5.2.tar.xz  
tar: linux-6.5.2.tar.xz: Cannot open: No such file or directory  
tar: Error is not recoverable: exiting now  
vivek@vivek-HP-Pavilion:~$ cd D  
Desktop/  Documents/ Downloads/  
vivek@vivek-HP-Pavilion:~$ cd D  
Desktop/  Documents/ Downloads/  
vivek@vivek-HP-Pavilion:~$ cd Do  
Documents/ Downloads/  
vivek@vivek-HP-Pavilion:~$ cd Downloads/  
vivek@vivek-HP-Pavilion:~/Downloads$ tar -xvf linux-6.5.2.tar.xz  
linux-6.5.2/  
linux-6.5.2/.clang-format  
linux-6.5.2/.cocciconfig  
linux-6.5.2/.get_maintainer.ignore  
linux-6.5.2/.gitattributes  
linux-6.5.2/.gitignore  
linux-6.5.2/.mailmap  
linux-6.5.2/.rustfmt.toml
```

```
cd linux-6.5.1
```

```
make menuconfig
```

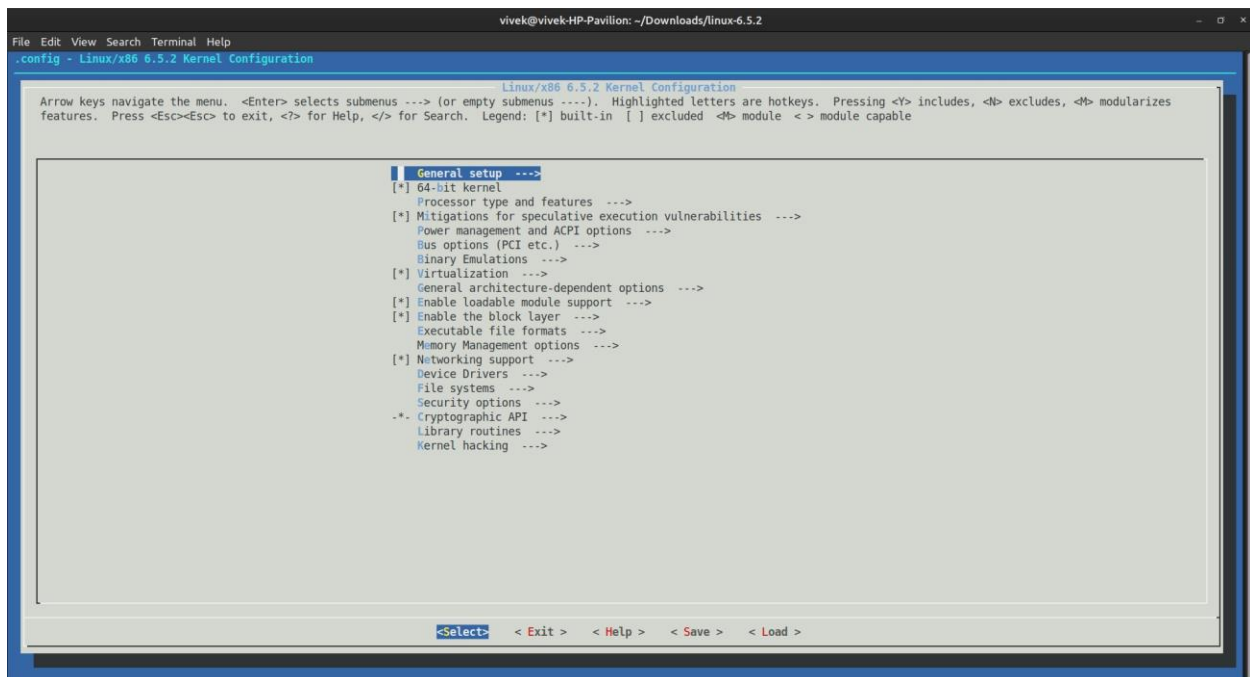


```
vivek@vivek-HP-Pavilion: ~/Downloads/linux-6.5.2  
File Edit View Search Terminal Help  
CC [M] net/openvswitch/openvswitch.mod.o  
LD [M] net/openvswitch/openvswitch.ko  
CC [M] net/openvswitch/vport-vxlan.mod.o  
LD [M] net/openvswitch/vport-vxlan.ko  
CC [M] net/openvswitch/vport-geneve.mod.o  
LD [M] net/openvswitch/vport-geneve.ko  
CC [M] net/openvswitch/vport-gre.mod.o  
LD [M] net/openvswitch/vport-gre.ko  
CC [M] net/vmw_vsock/vsock.mod.o  
LD [M] net/vmw_vsock/vsock.ko  
CC [M] net/vmw_vsock/vsock_diag.mod.o  
LD [M] net/vmw_vsock/vsock_diag.ko  
CC [M] net/vmw_vsock/vmw_vsock_vhci_transport.mod.o  
LD [M] net/vmw_vsock/vmw_vsock_vhci_transport.ko  
CC [M] net/vmw_vsock/vmw_vsock_virtio_transport.mod.o  
LD [M] net/vmw_vsock/vmw_vsock_virtio_transport.ko  
CC [M] net/vmw_vsock/vmw_vsock_virtio_transport_common.mod.o  
LD [M] net/vmw_vsock/vmw_vsock_virtio_transport_common.ko  
CC [M] net/vmw_vsock/hv_vsock.mod.o  
LD [M] net/vmw_vsock/hv_vsock.ko  
CC [M] net/vmw_vsock/vsock_loopback.mod.o  
LD [M] net/vmw_vsock/vsock_loopback.ko  
CC [M] net/nsh/nsh.mod.o  
LD [M] net/nsh/nsh.ko  
CC [M] net/hsr/hsr.mod.o  
LD [M] net/hsr/hsr.ko  
CC [M] net/qdrtr/qdrtr.mod.o  
LD [M] net/qdrtr/qdrtr.ko  
CC [M] net/qdrtr/qdrtr-smc.mod.o  
LD [M] net/qdrtr/qdrtr-smc.ko  
CC [M] net/qdrtr/qdrtr-tun.mod.o  
LD [M] net/qdrtr/qdrtr-tun.ko  
CC [M] net/qdrtr/qdrtr-ahb.mod.o  
LD [M] net/qdrtr/qdrtr-ahb.ko  
CC [M] virt/lib/irqbypass.mod.o  
LD [M] virt/lib/irqbypass.ko  
GEN scripts/gdb/linux/constants.py  
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ make -j$(nproc)  
DESCEND objtool  
INSTALL libsubcmd_headers  
CALL scripts/checksyscalls.sh  
CHK kernel/headers/data.tar.xz  
Kernel: arch/x86/boot/bzImage is ready (#1)  
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$
```

This command launches a text-based menu-driven configuration interface. Here's a basic overview of some key configuration sections:

General Setup: Configure fundamental settings such as the kernel version, local version information, and kernel compression method.

- **Processor type and features**: Customize CPU-related options, including processor family, CPU frequency scaling, and virtualization support.
- **Device Drivers**: Enable or disable support for various hardware devices and drivers, including storage devices, network adapters, and USB devices.
- **File systems**: Configure support for different file system types (e.g., ext4, XFS, Btrfs) and file system features.
- **Networking support**: Enable or disable networking options, including protocols and network device drivers.
- **Security options**: Set security-related options such as SELinux or AppArmor.
- **Kernel hacking**: Configure options for debugging and kernel tracing.



After configuring the kernel to your specifications, save your changes and exit the configuration interface. Your settings will be saved in a `.config` file in the kernel source directory.

4. Compile the Kernel

After you have configured the Linux kernel source code to match your system's requirements, the next step is to compile the kernel. Compiling the kernel involves translating the human-readable C source code into machine-executable binary code that your computer can understand and use as the operating system.

- when you run the `make -j\$(nproc)` command, it initiates the compilation process with the optimal number of simultaneous compilation tasks based on your CPU's core count
- After compiling the kernel successfully, the next steps typically involve installing the kernel modules and the kernel image, as well as updating the GRUB bootloader configuration to include the newly compiled kernel, as explained in previous responses.

```
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ make -j$(nproc)
DESCEND objtool
INSTALL libsubcmd_headers
CALL scripts/checksyscalls.sh
CHK kernel/kheaders_data.tar.xz
Kernel: arch/x86/boot/bzImage is ready (#1)
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ sudo make modules_install
[sudo] password for vivek:
INSTALL /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
SIGN /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
At main.c:170:
- SSL error:1E08010C:DECODER routines::unsupported: ../crypto/encode_decode/decoder_lib.c:101
sign-file: ./
make[2]: *** [scripts/Makefile.modinst:87: /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko] Error 1
make[2]: *** Deleting file '/lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko'
make[1]: *** [/home/vivek/Downloads/linux-6.5.2/Makefile:1964: modules_install] Error 2
make: *** [Makefile:234: __sub-make] Error 2
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ sudo make modules_install
sudo make install
INSTALL /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
```

This command will compile the kernel using the number of processor cores you have for parallel compilation, which speeds up the process.

5.Install Modules and Kernel

After successfully compiling the Linux kernel, you need to install the kernel modules and the kernel image onto your system. These are essential components for the proper functioning of the new kernel.

...

```
sudo make modules_install
sudo make install
```

...


```

CALL scripts/checksysctls.sh
CHK kernel/kheaders.data.tar.xz
Kernel: arch/x86/boot/bzImage is ready (#1)
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ sudo make modules_install
[sudo] password for vivek:
INSTALL /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
SIGN /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
At main.c:170:
- SSL error:1E08010C:DECODER routines::unsupported: ../crypto/encode_decode/decoder_lib.c:101
sign-file: ./
make[2]: *** [scripts/Makefile.modinst:87: /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko] Error 1
make[2]: *** Deleting file '/lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko'
make[1]: *** [/home/vivek/Downloads/linux-6.5.2/Makefile:1964: modules_install] Error 2
make: *** [Makefile:234: __sub-make] Error 2
vivek@vivek-HP-Pavilion:~/Downloads/linux-6.5.2$ sudo make modules_install
sudo make install
INSTALL /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
SIGN /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko
At main.c:170:
- SSL error:1E08010C:DECODER routines::unsupported: ../crypto/encode_decode/decoder_lib.c:101
sign-file: ./
make[2]: *** [scripts/Makefile.modinst:87: /lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko] Error 1
make[2]: *** Deleting file '/lib/modules/6.5.2/kernel/arch/x86/events/amd/amd-uncore.ko'
make[1]: *** [/home/vivek/Downloads/linux-6.5.2/Makefile:1964: modules_install] Error 2
make: *** [Makefile:234: __sub-make] Error 2
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/dkms 6.5.2 /boot/vmlinuz-6.5.2
* dkms: running auto installation service for kernel 6.5.2
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.5.2 /boot/vmlinuz-6.5.2
update-initramfs: Generating /boot/initrd.img-6.5.2
run-parts: executing /etc/kernel/postinst.d/pm-utils 6.5.2 /boot/vmlinuz-6.5.2

```

6.Update GRUB

Update the GRUB configuration to include the new kernel:

```

...

```

`sudo update-grub`

```

...

```

```

vivek@vivek-HP-Pavilion: ~
File Edit View Search Terminal Help
vivek@vivek-HP-Pavilion:~$ sudo update-grub
[sudo] password for vivek:
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/50_linuxmint.cfg'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.4.14-060414-generic
Found initrd image: /boot/initrd.img-6.4.14-060414-generic
Found linux image: /boot/vmlinuz-5.15.0-83-generic
Found initrd image: /boot/initrd.img-5.15.0-83-generic
Found linux image: /boot/vmlinuz-5.15.0-56-generic
Found initrd image: /boot/initrd.img-5.15.0-56-generic
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
Some pools couldn't be imported and will be ignored:
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1@EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
vivek@vivek-HP-Pavilion:~$

```

7.Reboot

Reboot your system. When you see the GRUB menu during boot, you should now have the option to boot into the new kernel alongside your existing one.

1. ****Save Your Work:**** Ensure that you have saved any unsaved work and closed all running applications. Rebooting will restart your computer, and any unsaved changes may be lost.
2. ****Issue the Reboot Command:**** Open a terminal if you don't already have one open and enter the following command to initiate the reboot:

```
`sudo reboot`
```

This command will prompt you for your password because it requires administrative privileges to reboot the system.

3. ****GRUB Boot Menu:**** During the boot process, your computer will display the GRUB (GRand Unified Bootloader) menu. The GRUB menu typically appears shortly after your system's BIOS/UEFI screen and lists the available boot options.

4. ****Select the Kernel:**** In the GRUB menu, you should now see multiple boot options. This includes your existing Linux kernel version and the newly compiled one. Use the arrow keys on your keyboard to highlight the desired kernel version, and then press Enter to boot into that kernel.

- ****Existing Kernel:**** This option represents your current, stable Linux kernel, which you were using before the kernel compilation process.

- ****Newly Compiled Kernel:**** This option represents the kernel you just compiled and installed.

5. ****Boot into the Chosen Kernel:**** After selecting the kernel version, your computer will proceed to boot into the selected kernel. You will see the familiar Linux boot sequence messages as the system initializes.

6. ****Login and Verify:**** Once the boot process is complete, log in to your system as usual. Now, you are running the kernel you selected from the GRUB menu. To verify that you are using the new kernel, you can open a terminal and use the following command:

GNU GRUB version

Linux Mint 21.1 Cinnamon, with Linux 6.5.2
Linux Mint 21.1 Cinnamon, with Linux 6.5.2 (recovery mode)
Linux Mint 21.1 Cinnamon, with Linux 6.4.14-060414-generic
Linux Mint 21.1 Cinnamon, with Linux 6.4.14-060414-generic (recovery mode)
Linux Mint 21.1 Cinnamon, with Linux 5.15.0-83-generic
Linux Mint 21.1 Cinnamon, with Linux 5.15.0-83-generic (recovery mode)
Linux Mint 21.1 Cinnamon, with Linux 5.15.0-56-generic
*Linux Mint 21.1 Cinnamon, with Linux 5.15.0-56-generic (recovery mode)