National University
Of Computer and Emerging Sciences

# Assignment 2

## Name – Muhammad Taha

## Roll NO – 23P-0559

## SECTION – BCS(4A)

## Subject – OPERATING SYSTEM- LAB

## Kernel Compilation in Gentoo

This is a continuation of the previous assignment where I will complete the kernel compilation process from start to finish.

# 1. Configuring the Kernel

The Linux kernel is the core component of your operating system responsible for managing hardware resources and enabling communication between software and your computer. In Gentoo the kernel configuration is done manually to ensure that it supports your specific hardware and meets your system's requirements. This step is crucial to optimize the system for performance and compatibility with your hardware.

## Steps:

## Install the Kernel Sources

## 1. Before configuring, install Gentoo's kernel source code:

**emerge -v sys-kernel/gentoo-source**

This fetches the latest kernel sources from Gentoo's repositories, or you can explicitly use the kernel version by writing the kernel version in place of sources like 6.12.6 (current version)

2. **Navigate to the Kernel Directory**
   Switch to the kernel source directory:

   **cd /usr/src/linux**

3. **Run menuconfig to Configure the Kernel**
   Launch the kernel configuration menu:

**make menuconfig**

- This opens a text-based interface where you can enable/disable kernel features.
- Essential settings to check:

1. **Set CPU Type (Core i3/i5/i7)**

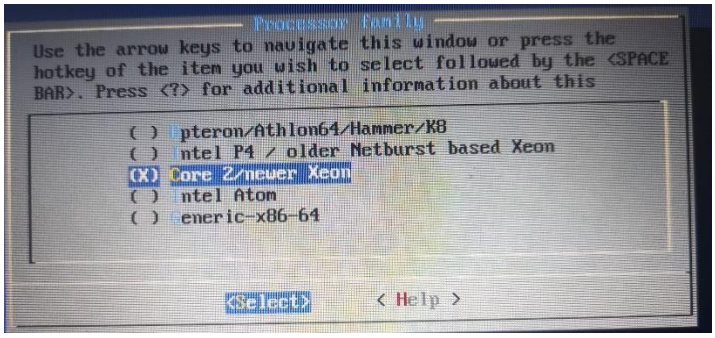
   Navigate to:

   Processor type and features --->

   Processor family --->

   (X) Core i3/i5/i7

   Select the correct CPU type.

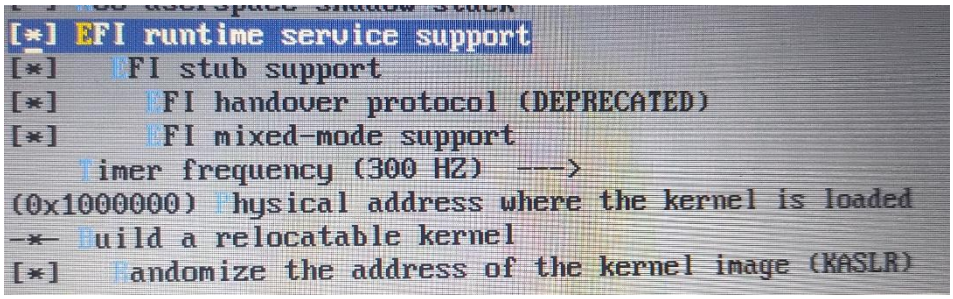



## 2. Enable EFI Support

Navigate to:

Processor type and features --->

[*] EFI runtime service support

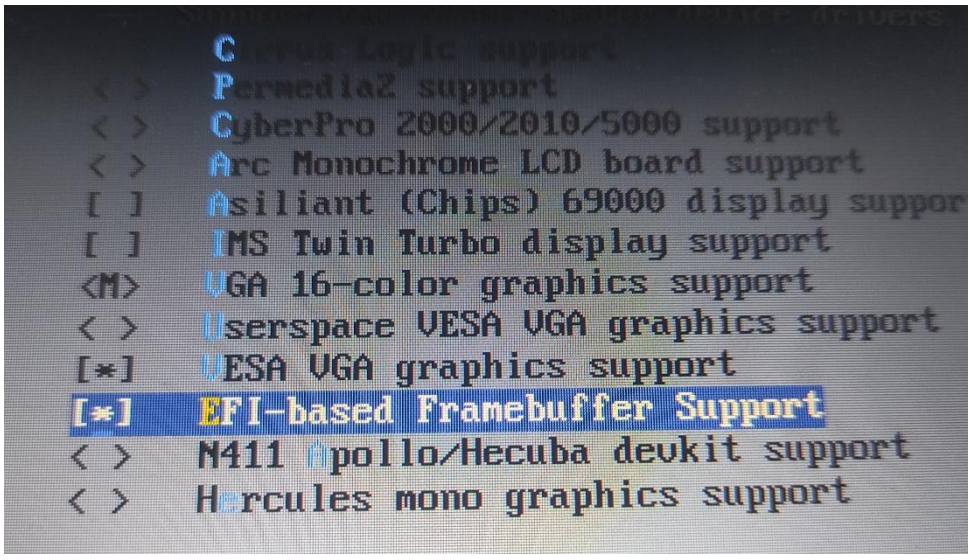



Enabling EFI runtime service support ensures that the kernel can properly interact with the EFI system, which is required for booting on modern systems with UEFI firmware.Also enable EFI framebuffer (needed for GRUB EFI boot):

Device Drivers --->

Graphics support --->

[*] EFI-based Framebuffer Support



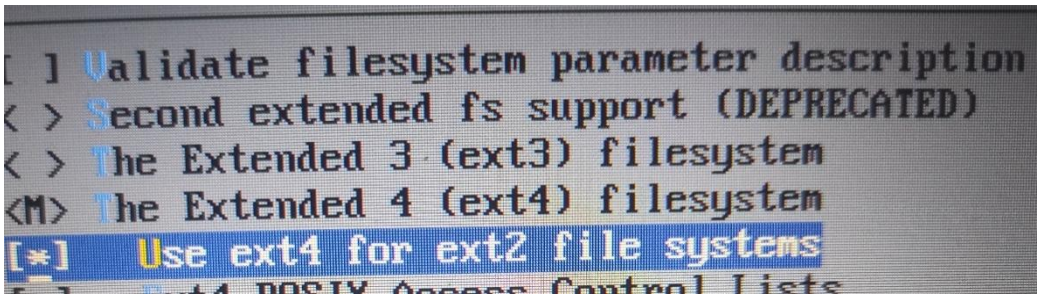
Enabling EFI framebuffer support is necessary for displaying graphics during boot with a UEFI system. It ensures that the bootloader (like GRUB) can display graphical content correctly before the kernel takes over.

3. **Enable Filesystem Support (ext4, FAT32, NTFS)**

   Navigate to:

   File systems  --->

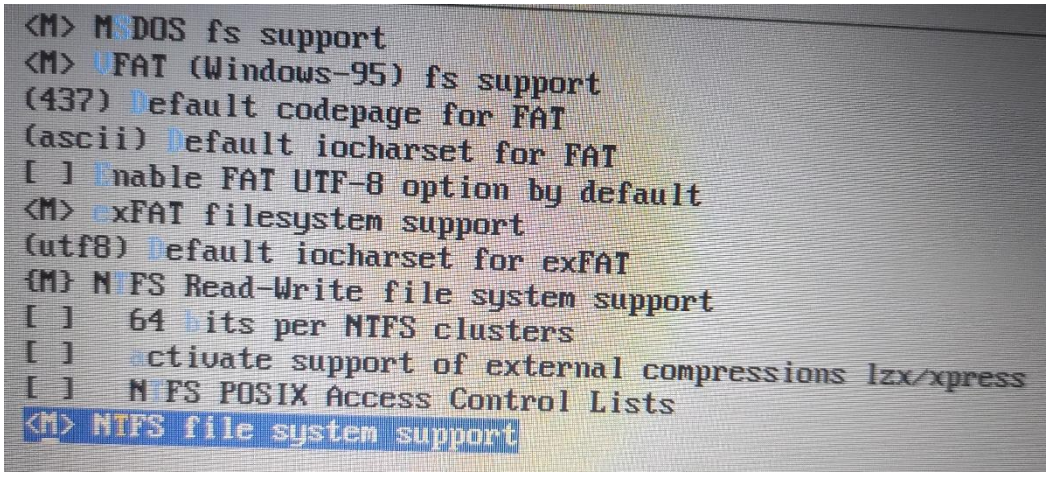
   <*> The Extended 4 (ext4) filesystem




   <*> DOS/FAT/NT Filesystems  --->

   <*> VFAT (Windows-95) fs support

   <*> NTFS file system support




   Use <*> instead of <M> if you want them built-in instead of modules.

4. **Enable Wi-Fi Drivers (**

   Navigate to:

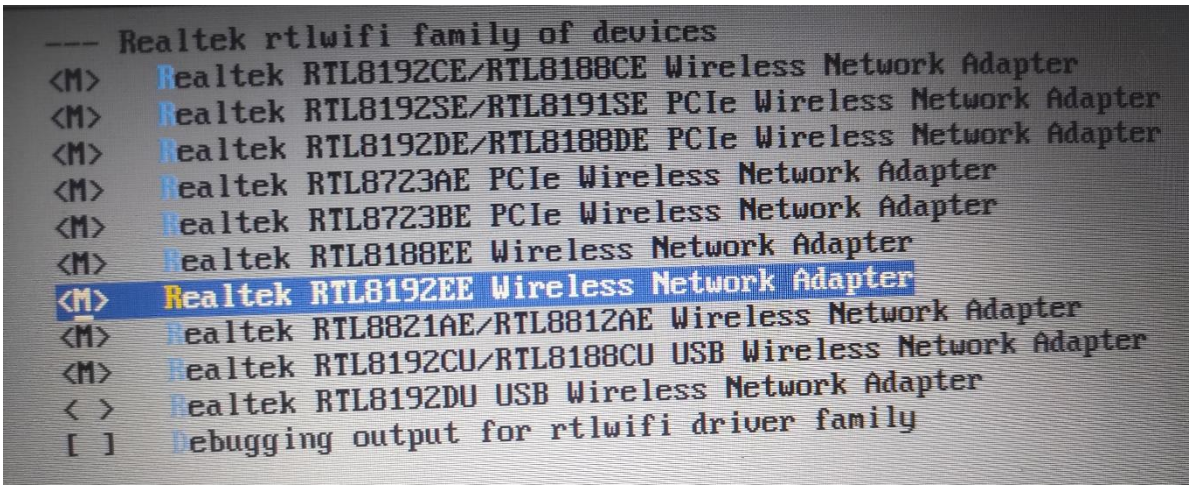
   Device Drivers  --->

   Network device support  --->

   Wireless LAN  --->

   find **Realtek**

**RTL8192EE** and enable it by selecting the correct driver




We enable this to enable for wifi adapter

5. **Enable Framebuffer Support (Fix Display Issues)**

Navigate to:

Device Drivers --->

Graphics support --->

[*] Support for frame buffer devices

[*] Simple framebuffer support (CONFIG_FB_SIMPLE)

[*] EFI-based Framebuffer Support (CONFIG_FB_EFI)

This fixes **"CONFIG_FB is not set"** errors. You can enable them by just enter forward slash(/) and then search the following name CONFIG_FB_SIMPLE CONFIG_FB_EFI and press 1 for each and then press y to enable them.

4. **Save Your Configuration**

After making changes, save and exit. The settings are stored in .config.

# 2. Compiling the Kernel

Compiling transforms your configured kernel into a bootable binary. This step is resource-intensive and may take time.

**Steps:**

**Start Compilation**
Use make with parallel jobs for faster compilation:

**make -j$(nproc)**

This automatically detects and uses all available CPU cores.

**How to Manually Set -j Value?**

- **Recommended formula:**

CPU cores×1.5

Example:

**4-core CPU** → Use -j6

**8-core CPU** → Use -j12

This ensures faster compilation without overloading the system and helps prevent errors like **make[2,3]**. If you encounter issues, you can adjust the -j value to better match your system's resources.

**3. Installing the Kernel**

The compiled kernel must be installed to /boot and modules to /lib/modules.

**Steps:**

**1. Install Kernel Modules**

make modules_install

**2. Install the Kernel**

make install

This copies the kernel (vmlinuz) and initial RAM disk (initramfs) to /boot.

When we do make install we usually seen the popup of the cannot/install lilo
so for this we manually paste in this folder

**cp arch/x86/boot/bzImage /boot/vmlinuz-6.12.6-gentoo**

**cp System.map /boot/System.map-6.12.6-gentoo**

**cp .config /boot/config-6.12.6-gentoo**

- This moves the **bzImage** file to /boot and renames it as vmlinuz, which is a common name for Linux kernels.
- GRUB expects the kernel file to be inside /boot, so this step ensures GRUB can load it properly.