

## Cross-compiler and kernel environment for rpi4

### I. Steps to flash Raspbian OS onto sd card:

On Host(ubuntu) :

1. Open terminal

```
$ sudo apt install rpi-imager
```

or

```
$ snap install rpi-imager
```

```
$ rpi-imager
```

Choose OS : Raspberry Pi OS (other) => raspberry pi OS LITE 32-bit

Choose storage : choose your sd card

Click on write and then click on yes - This will take some time.

2. After completing flashing image plug out sd card and insert sd card again.

```
$ cd /media/<user-name>/boot
```

```
$ touch ssh
```

```
$ touch wpa_supplicant.conf
```

```
$ vim wpa_supplicant.conf
```

Write the following code in wpa\_supplicant.conf file and save it.

```
country=IN
```

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
```

```
network={
```

```
    ssid="ShubhamMob"
```

```
    psk="Shubham@08"
```

```
    key_mgmt=WPA-PSK
```

```
}
```

3. Plug out the SD card and insert into your raspberry pi board.
4. Board will start booting and access it.

## II. Steps for cross-compiling kernel :

On Host (ubuntu) :

1. Install Required dependencies :

```
$ sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
```

2. Install 32-bit toolchain

```
$ sudo apt install crossbuild-essential-armhf
```

3. Download/clone kernel source

```
~ $ mkdir rpi
$ cd rpi
$ git clone --depth=1 --branch rpi-5.15.y https://github.com/raspberrypi/linux
$ cd linux
```

4. Apply the config file of rpi4 :

Check config file for your board(rpi4) using below command

```
$ ls arch/arm/configs
```

Default config file for rpi4 is bcm2711\_defconfig

Now apply config file using below command

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- bcm2711_defconfig
```

5. Build kernel image and kernel modules for rpi4 :

```
$ make -j8 ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage modules
```

Result of above command :

```
$ ls arch/arm/boot
```

zImage

6. Plug in your sd card to your HOST PC(ubuntu)

```
$ cp arch/arm/boot/zImage /media/<user_name>/boot
```

## 7. Install modules onto rootfs partition of SDcard

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-  
INSTALL_MOD_PATH=<path-to-sdcard rootfs partition> modules_install
```

Example In my pc :

```
$ make ARCH=arm  
CROSS_COMPILE=arm-linux-gnueabihf-INSTALL_MOD_PATH=/media/embedded/rootfs  
modules_install
```

Modules gets installed in rootfs/lib/modules path

## 8. Configuring config.txt to boot our new kernel

```
$ cd /media/<user-name>/boot
```

Open config.txt:

```
$ vim config.txt
```

Add below line at the end of the file and save file :

```
kernel=zImage
```

9. If “ssh” and “wpa\_supplicant.conf” files are not in your boot partition then follow steps of 2 of flashing raspbian OS.
10. Plug out sd card and insert into your raspberry pi board.
11. Board will start booting and access it .

### III. Cross compile module for the Raspberry Pi.

1. Write a source code on HOST Machine(Ubuntu)

hello.c

```
#include<linux/module.h>
#include<linux/init.h>

static int __init hello_init(void)
{
    pr_info("Hello World\n");
    return 0;
}

static void __exit hello_exit(void)
{
    pr_info("Good Bye\n");
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("CDAC");
MODULE_DESCRIPTION("A simple hello_world kernel module");
MODULE_INFO(board,"RASPBERRY PI 4");
```

Makefile for Cross Compilation.

```
obj-m := hello.o
KERN_DIR=/lib/modules/5.10.52-v7l+/build/

all:
    Make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C $(KERN_DIR)
    M=$(PWD) modules

clean:
    make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C $(KERN_DIR)
    M=$(PWD) clean
```

- After compiling the module, Copy the Kernel Object(.ko) file to the RaspberryPi.

- You can copy the contents from host to destination using command,

```
scp -r <path-to-source> <path-to-destination>
```

Example-

```
scp hello.ko /home/pi/<your-folder>
```

#### **IV. Native compile for the Raspberry Pi**

hello.c

- source code will be same - refer earlier hello.c file.

Makefile for the Native Compilation

```
obj-m := hello.o
```

```
KERN_DIR=/lib/modules/$(shell uname -r)/build/
```

```
all:
```

```
    make -C $(KERN_DIR) M=$(PWD) modules
```

```
clean:
```

```
    make -C $(KERN_DIR) M=$(PWD) clean
```