**Cross-compiler and kernel environment for rpi4**

1. **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.

1. 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=”PrasadMob”

psk=”prasad@08”

key\_mgmt=WPA-PSK

}

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

1. Install 32-bit toolchain

$ sudo apt install crossbuild-essential-armhf

1. Download/clone kernel source

~ $ mkdir rpi

$ cd rpi

$ git clone --depth=1 --branch rpi-5.15.y <https://github.com/raspberrypi/linux>

$ cd linux

1. 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-gnueabihf- bcm2711\_defconfig

1. Build kernel image and kernel modules for rpi4 :

$ make -j8 ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- zImage modules

Result of above command :

$ ls arch/arm/boot

zImage

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

$ cp arch/arm/boot/zImage /media/<user\_name>/boot

1. 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

1. 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

1. If “ssh” and “wpa\_supplicant.conf” files are not in your boot partition then follow steps of 2 of flashing raspbian OS.
2. Plug out sd card and insert into your raspberry pi board.
3. Board will start booting and access it .
4. **Cross compile module for the Raspberry Pi.**
5. 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>

1. **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