**ssh -p 5057 vectorrp7@devicedrivers.ddns.net**

**@rp57**

**scp -P 5057 timer vectorrp7@devicedrivers.ddns.net:/tmp/**

**This document created with the view of 32-bit Raspberry PI board.**

**Software**

Ubuntu 20.04

**Hardware**

Raspberry Pi

Micro SD card (16/32GB)

Micro SD card reader

**Experiments details**

1. Flash Raspberry Pi Image on SD card/USB. Boot with new Image

2. Cross compile Hello world application and execute it on RP board

3. Cross Compiling Kernel for Raspberry Pi board

4. Port Hello world kernel module to Raspberry Pi board

**Experiment -1:**

**Flash Raspberry Pi SD image**

1. Download the Raspberry-pi- os (Raspberry Pi OS with desktop) from below weblink

For 32-bit :-

[https://www.raspberrypi.com/software/operating-systems/#raspberry-pios-32-bit](https://www.raspberrypi.com/software/operating-systems/" \l "raspberry-pios-32-bit)

2. Extract the downloaded image file using below command. This command is applicable only if your downloaded file has extension (img.xz) in Linux

* **$unxz filename.img.xz**

Example:

root@rama-Inspiron-3501:/home/rama/Downloads# unxz 2023-05-03-raspios-bullseye-armhf.img.xz

root@rama-Inspiron-3501:/home/rama/Downloads# ls 2023-05-03-raspios-bullseye-armhf.img

3. Above command generates Raspberry pi Image file. Flash this Image file to micro sd card using the below tools By using the below weblink choose the respective OS

[**https://www.balena.io/etcher/**](https://www.balena.io/etcher/)

4. After flashing Image on SD card, Insert the SD card into Raspberry Pi. Connect Keyboard, mouse and HDMI display. Power up on board

**Experiment - 2 :**

**Cross compile Hello world application and Execute on RP board**

1. Install the required dependency tool chain

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

2. Install the 32-bit Toolchain for a 32-bit Kernel

* **sudo apt install crossbuild-essential-armhf**

3. Cross compile hellow\_world.c application on Host OS

using ARM cross compiler tool chain

* **vi hello.c**
* **arm-linux-gnueabihf-gcc hello.c -o arm**

3.Copy the application to RP board using scp/ usb/sd

card. Here using scp to copy to RP board. Ensure ssh is enabled on RP board

(use raspi-config utility/ under this utility to enable ssh)

* **$rama@host:~ scp arm pi@192.168.2.240:/tmp**

4. Execute application on RP board

* **$pi@raspberrypi:~ $ cp /tmp/arm**
* **$pi@raspberrypi:~ $ file arm**

ELF 32-bit LSB pie executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux- armhf.so.3,

* **$pi@raspberrypi:~ $ ./arm**

Hello world.

**Experiment -3**:

**Reference**

[**https://www.raspberrypi.com/documentation/computes/linux\_kernel.html**](https://www.raspberrypi.com/documentation/computes/linux_kernel.html)

1. Download Kernel source code for RP

* **git clone -b rpi-6.5.y https://github.com/raspberrypi/linux.git**

2. Build sources:-

For Raspberry Pi 2, 3, 3+ and Zero 2 W, and Raspberry Pi Compute Modules 3 and 3+:

* **cd linux**
* **KERNEL=kernel7**
* **make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- bcm2709\_defconfig**

For custom configuration

The menuconfig tool requires the ncurses development headers to compile properly. These can be installed with the following command:

* **sudo apt install libncurses5-dev**

If you’re cross-compiling a 32-bit kernel:

* **make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- menuconfig**

Build with Configs

For all 32-bit Builds

* **make ARCH=arm CROSS\_COMPILE=arm-linuxgnueabihf- zImage modules dtbs**

Install Kernel

Having built the kernel, you need to copy it onto your Raspberry Pi and install the modules;

this is best done directly using an SD card reader.

First, use **lsblk** before and after plugging in your SD card to identify it. You should end up with something a lot like this:

**sdb**

**sdb1**

**sdb2**

with **sdb1** being the **FAT** filesystem (boot) partition, and

**sdb2** being the ext4 filesystem (root) partition. Mount these first, adjusting the partition letter as necessary:

* **mkdir -p /mnt**
* **mkdir -p /mnt/fat32**
* **mkdir -p /mnt/ext4**
* **sudo mount /dev/sdb1 /mnt/fat32** // boot partition
* **sudo mount /dev/sdb2 /mnt/ext4** // root FS

Next, install the kernel modules onto the SD card:

For 32-bit

**sudo env PATH=$PATH make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- INSTALL\_MOD\_PATH=/mnt/ext4 modules\_install**

Finally, copy the kernel and Device Tree blobs onto the

SD card, making sure to back up your old kernel:

For 32-bit

* **sudo cp /mnt/fat32/$KERNEL.img /mnt/fat32/$KERNEL-backup.img**
* **sudo cp arch/arm/boot/zImage /mnt/fat32/$KERNEL.img**
* **sudo cp arch/arm/boot/dts/broadcom/\*.dtb /mnt/fat32/**
* **sudo cp arch/arm/boot/dts/overlays/\*.dtb\* /mnt/fat32/overlays/**
* **sudo cp arch/arm/boot/dts/overlays/README /mnt/fat32/overlays/**
* **sudo umount /mnt/fat32**
* **sudo umount /mnt/ext4**

**Experiment - 4 :**

Port Hello world kernel module to Raspberry Pi board

1. Write basic Hello world kernel module.

khello.c

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/sched.h>

#include <linux/slab.h>

static int \_\_init khello\_init(void)

{

printk(KERN\_EMERG"Hello world1\n");

printk("Hello world2\n");

printk("Hello world3\n");

return 0;

}

static void \_\_exit khello\_exit(void)

{

printk("Khello unloaded.\n");

}

module\_init(khello\_init);

module\_exit(khello\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("RAMA");

MODULE\_DESCRIPTION("A simple Hello world module.");

2. Change the Makefile content to cross-compile for RP board

obj-m := khello.o

PWD := $(shell pwd)

CCPREFIX := arm-linux-gnueabihf-

all:

make ARCH=arm CROSS\_COMPILE=${CCPREFIX} -C <path to Raspberry kernel> M=$(PWD) modules

clean:

make -C <path to Raspberry kernel> M=$(PWD) clean

3. Build the kernel module and copy that module to RP board

[root@rama-Inspiron-3501](mailto:root@rama-Inspiron-3501):/home/rama/linux/khello\_rp# make

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- -C /home/rama/linux M=/home/ rama/linux/khello\_rp modules

make[1]: Entering directory '/home/rama/linux'

$scp khello.ko pi[@192.168.0.4](mailto:rama@192.168.0.4):/tmp

pi@raspberrypi:~ $ cp /tmp/khello.ko .

pi@raspberrypi:~ $ modinfo khello.ko

filename:

/home/rama/khello.ko

description: A simple Hello world module.

Author: RAMA

license: GPL

srcversion: 3526A3396836880EBE7F7B6

depends:

name: khello

vermagic: 6.1.38-v7+ SMP mod\_unload modversions ARMv7 p2v8

rama@raspberrypi:~ $ insmod khello.ko