Building Linux kernel for Raspberry Pi

I. Overview

To compile code for your system, you need install compilation toolchain, what contain series tools such as : a compiler (ex: gcc), a standard C library (ex: GNU libc), …

If you use a toolchain runs on your system and generates code for this system, the toolchain will be called as “host compilation toolchain” and the building process will be called as “local building”. But in your embedded system, for some reasons, you must run toolchain in one system and generates code for other system, so you need a “cross compilation toolchain” and that process is called as “cross building”. In this guide, we choose cross building to build kernel for Rasp Pi.

Kernel components :

- Kernel image: in this case we use zImage (read more about zImage/uImage???)
- Device tree binary: low level device description for your device (*.dtb)
- Kernel modules: driver modules
- Device firmware

Kernel image and Device tree binary will be contained in boot partition of your system (boot – fat32), whereas the rest will be contained in root file system partition (rootfs – ext4).

I recommend that you should use linux virtual machine (Ubuntu) , that easy for you to both build kernel and connect to device. You can verify your building process by the way: download a completely raspbian image from internet, mount it to device and boot it; when success you can get kernel version (ex: 2016-09-23-raspbian-jessie have kernel version is 4.4.21); after you done building your custom kernel, you mount it to device and can get new version.

II. Build process

1. Step1: Prepare environment, get cross compilation toolchain and kernel source

You have to install some host packages to support building process (ex: git, make, ncurses5 …) when you perform build, if system requires any package, you must install it by:

sudo apt-get install package_name

Cross compilation toolchain for raspbian is available on Raspberry Pi server, get it by:

git clone https://github.com/raspberrypi/tools

Kernel source is similar to cross compilation toolchain, get it by:

git clone --depth=1 https://github.com/raspberrypi/linux

2. Step2: Build source

cd linux

First of all, you need config the kernel you want to build, this means you can choose what driver will be had in built kernel (turn on/off driver available in device). This very useful when you want to optimize kernel and save memory. For study and normally develop purpose, I think you should use default configuration (turn on all available driver). Your configuration is written into file .config and is loaded when you boot device.

Generate .config file:

Pi 1:

make ARCH=arm CROSS_COMPILE=../tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf- bcmrpi_defconfig

Pi 2/3:

make ARCH=arm CROSS_COMPILE==../tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf- bcm2709_defconfig

Option: If you wants to modify configuration (turn on/off driver) use :

make menuconfig

A GUI appear help you easy to setting (You need try more yourselt)

Next is make the makefile to generates object file, zImage (a compressed kernel image), device tree file

```
make ARCH=arm CROSS_COMPILE===../tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf- zImage modules dtbs
```

To speed up, if your host system is multiprocessor, you can use option –j n (n is number of processors) with the command above.

3. Step3: Install onto device (SD Card)

Insert SD Card to host system and use

```
lsblk
```

To check partition of this SD Card, usually same that

sdb – sdb1: boot, sdb2: rootfs

Mount SD Card to host system by following command:

```
mkdir mnt mnt/fat32 mnt/ext4

sudo mount /dev/sdb1 mnt/fat32

sudo mount /dev/sdb2 mnt/ext4
```

Next you install module (generate file .ko – kernel object) to folder mnt/ext4

```
sudo make ARCH=arm CROSS_COMPILE===../tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf- INSTALL_MOD_PATH=mnt/ext4 modules_install
```

Finally copy all device tree file and kernel image file to boot partition:

```
sudo cp mnt/fat32/kernel.img mnt/fat32/kernel-backup.img

sudo scripts/mkknlimg arch/arm/boot/zImage mnt/fat32/kernel.img

sudo cp arch/arm/boot/dts/*.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

At this time, you build your custom kernel success, remove SD Card and insert it into device then perform boot.

4. Step4: Generate file .img for re-use purpose

Insert SD Card into host system and run command:

sudo dd if=/dev/sdb of=my_kernel_image.img

On other hand, to load an image to SD Card you can use command

sudo dd if=my_kernel_image.img of=/dev/sdb bs=4M

Boot Pi can be performed through Serial connection by Putty or TeraTerm in Window

Login: pi

Password: raspberry

GOOD LUCK!