Kernel

This page introduce how you can download and compile the Linux kernel for **ODROID-N2**.

Cross Compile - HOST-PC/Ubuntu

Installing required packages

You will need install required packages before you start to build Linux kernel on your Ubuntu desktop.

```
$ sudo apt-get update
$ sudo apt-get install git lzop build-essential gcc bc libncurses5-dev
libc6-i386 lib32stdc++6 zlib1g:i386
```

Toolchain (6.3.1)

Click one of the site to download toolchain to build Linux kernel.

- Download

Once the download is done, extract the tarball to **/opt/toolchains/**.

```
$ sudo mkdir -p /opt/toolchains
$ sudo tar Jxvf gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu.tar.xz -C
/opt/toolchains/
```

In order to add the toolchain path to PATH, paste below lines to **$HOME/.bashrc**.

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
export PATH=/opt/toolchains/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-
gnu/bin/:$PATH
```

You can apply the change if you login again or import to apply this change, login again or evaluate **$HOME/.bashrc** with source command.

```
$ source ~/.bashrc
```

You can check if the toolchain installed above works properly while checking the version of toolchain. If you can find **gcc version 6.3.1 20170109** at the end of the line, the toolchain is well installed.

```
$ aarch64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/opt/toolchains/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-
```

```
linux-gnu/bin/../libexec/gcc/aarch64-linux-gnu/6.3.1/lto-wrapper
Target: aarch64-linux-gnu
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-
release/label/docker-trusty-amd64-tcwg-build/target/aarch64-linux-
gnu/snapshots/gcc-linaro-6.3-2017.02/configure SHELL=/bin/bash --with-
mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-
amd64-tcwg-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-
unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-
release/label/docker-trusty-amd64-tcwg-build/target/aarch64-linux-
gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-
buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-
build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-
gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-
objc-gc --enable-shared --without-included-gettext --enable-nls --disable-
sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --
disable-libstdcxx-pch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-
debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --
disable-multilib --enable-fix-cortex-a53-835769 --enable-fix-cortex-
a53-843419 --with-arch=armv8-a --enable-threads=posix --enable-multiarch --
enable-libstdcxx-time=yes --enable-gnu-indirect-function --with-build-
sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-
trusty-amd64-tcwg-build/target/aarch64-linux-gnu/_build/sysroots/aarch64-
linux-gnu --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-
release/label/docker-trusty-amd64-tcwg-build/target/aarch64-linux-
gnu/_build/builds/destdir/x86_64-unknown-linux-gnu/aarch64-linux-gnu/libc --
enable-checking=release --disable-bootstrap --enable-
languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-
unknown-linux-gnu --target=aarch64-linux-gnu --prefix=/home/tcwg-
buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-
build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-
gnu
Thread model: posix
gcc version 6.3.1 20170109 (Linaro GCC 6.3-2017.02)
```

Checkout

You can check out Linux kernel source tree from **Hardkernel's Github**, please note that we distribute the Linux kernel in different branches for Android and other Linux distributions.

Linux

```
$ git clone --depth 1 https://github.com/hardkernel/linux.git -b
odroidn2-4.9.y
$ cd linux
```

Android

Pie

```
$ git clone --depth 1 https://github.com/hardkernel/linux.git -b
odroidn2-4.9.y-android
$ cd linux
```

Compile

Note

- If you want to download & build only **Android Linux kernel** source code without another Android BSP, you face the error message as below:
  - *drivers/amlogic/wifi/Kconfig:26: can't open file "../hardware/wifi/realtek/drivers/8192cu/rtl8xxx_CU/Kconfig"*
- Please delete the last line as below in **'drivers/amlogic/wifi/Kconfig'** file if you will build only Android Linux kernel.
  - ~~source "../hardware/wifi/realtek/drivers/8192cu/rtl8xxx_CU/Kconfig"~~

You must do kernel configuration for **ODROID-N2**, then start to build. Adding **-j** option on make command will help you to finish compiling faster.

```
$ make odroidn2_defconfig
$ make [-j4] Image modules dtbs
```

You have done to compile the Linux kernel (Image), the device tree file (.dtb) and kernel modules (.ko).

Custom Kernel Build

If you have some kernel drivers wish to include for your custom build, you can select the drivers easily in Linux kernel tree. **make menuconfig** will show you text based menus help you to select kernel drivers.

```
$ make menuconfig
```

Once you done selecting the drivers, exit from the menu screen. Then you can start kernel build with **make** again.

```
$ make [-j4] Image modules
```

When you exit from kernel menu screen, you will have **.config** in the current directory what have changes for your custom build. You can back up this as a file, my_kernel.config for example, or can make a patch if you manage your own git.

```
$ cp .config arch/arm64/configs/odroidn2_defconfig
```

```
$ git add arch/arm64/configs/odroidn2_defconfig
$ git commit -s -m "Change the kernel config file for ODROID-N2"
$ git push
```

Installation

There are different instructions to install Linux kernel image and device tree for Android and Linux. Since Android loads both from a boot partition, we have to use **fastboot** to install into the dedicated partition. Please refer the partition table from here. In contrast, Linux boots by the instructions described in **boot.ini** the 1st FAT partition.

Android

This is the instruction to install kernel image, **Image**, to the boot card.
You should build boot image via android repo because boot iamge consist of root ramdisk and kernel image.
On the android build root directory.

```
$ make bootimage
```

This is to install boot image that include kernel image.

```
$ fastboot flash boot out/target/product/odroidn2/boot.img
```

This is to install a device tree file, **meson64_odroidn2.dtb**.

```
$ fastboot flash dtb <path/of/your/meson64_odroidn2.dtb>
```

Linux

> **This explanation assume that your USB memory CARD reader is assigned at /dev/sdc. Be careful!**

1. Plug the Boot-Device(eMMC or SD) into the USB memory CARD reader and Connect the USB memory CARD reader to your HOST PC(Linux OS).

2. Copy the Image and DT(meson64_odroidn2.dtb) to the FAT partition(1st partition) in the Boot-Device.

```
$ mkdir -p mount
$ sudo mount /dev/sdc1 ./mount
$ sudo cp arch/arm64/boot/Image arch/arm64/boot/dts/meson64_odroidn2.dtb
./mount && sync && sudo umount ./mount
```

3. Copy the driver modules to the EXT4 partition(2nd partition) in the Boot-Device.

```
$ sudo mount /dev/sdn2 ./mount
$ sudo make modules_install ARCH=arm64 INSTALL_MOD_PATH=./mount && sync &&
sudo umount ./mount
$ rm -rf mount
```

Native Compile - ODROID-N2/Ubuntu

Note

- 8GB eMMC/SD card have not enough space to build kernel source. In order to do native compile, the **5GB** of free space is required at least.

Installing required packages

You will need install required packages before you start to build Linux kernel on your Ubuntu ODROID-N2.

```
odroid@odroid64:~$ sudo apt update
odroid@odroid64:~$ sudo apt install git
```

Checkout

```
odroid@odroid64:~$ git clone --depth 1
https://github.com/hardkernel/linux.git -b odroidn2-4.9.y
odroid@odroid64:~$ cd linux
```

Compile & Installation

```
odroid@odroid64:~/linux$ make odroidn2_defconfig

You can edit .config file or run "make menuconfig" to add/remove Kernel
drivers.

odroid@odroid64:~/linux$ make -j4
odroid@odroid64:~/linux$ sudo make modules_install
odroid@odroid64:~/linux$ sudo cp -f arch/arm64/boot/Image.gz
arch/arm64/boot/dts/amlogic/meson64_odroidn2.dtb /media/boot/
odroid@odroid64:~/linux$ sudo sync
odroid@odroid64:~/linux$ sudo reboot
```