Installing required packages

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

host

```
$ sudo apt update
$ sudo apt install git
$ sudo apt install lib32stdc++6  lib32z1 lzop u-boot-tools
$ sudo apt install build-essential gcc
$ sudo apt install libncurses5-dev
```

Kernel

This page introduce how you can download and compile the Linux kernel for **ODROID-C1**. You can select one of two version, GCC 4.9.2 or GCC 4.7.3. Since Linux release **v1.1**, the official toolchain for **ODROID-C1** is switched to **GCC 4.9.2**.

Toolchain (4.9.2)

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

- Download #1
- Download #2

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

host

```
$ sudo mkdir -p /opt/toolchains
$ sudo tar xvf gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz
-C /opt/toolchains/
```

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

host

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
export PATH=/opt/toolchains/gcc-linaro-arm-linux-
gnueabihf-4.9-2014.09_linux/bin/:$PATH
```

You can check if the toolchain installed above works properly while checking the version of toolchain. If you can find **gcc version 4.9.2 20140904 (prerelease)** at the end of the line, the toolchain is well installed.

host

```
$ source ~/.bashrc
$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=/opt/toolchains/gcc-linaro-arm-linux-
gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/opt/toolchains/gcc-linaro-arm-linux-
gnueabihf-4.9-2014.09_linux/bin/../libexec/gcc/arm-linux-
gnueabihf/4.9.2/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-
gnueabihf-linux/.build/src/gcc-linaro-4.9-2014.09/configure --
build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --
target=arm-linux-gnueabihf --prefix=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-linux-gnueabihf-linux/install --with-
sysroot=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/install/arm-linux-gnueabihf/libc --enable-languages=c,c++,fortran
--disable-multilib --enable-multiarch --with-arch=armv7-a --with-
tune=cortex-a9 --with-fpu=vfpv3-d16 --with-float=hard --with-
pkgversion='crosstool-NG linaro-1.13.1-4.9-2014.09 - Linaro GCC
4.9-2014.09' --with-bugurl=https://bugs.launchpad.net/gcc-linaro --
enable-__cxa_atexit --enable-libmudflap --enable-libgomp --enable-
libssp --with-gmp=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-
gnueabihf-linux/.build/arm-linux-gnueabihf/build/static --with-
mpfr=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
mpc=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
isl=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
cloog=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
libelf=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --enable-threads=posix --
disable-libstdcxx-pch --enable-linker-build-id --enable-plugin --
enable-gold --with-local-prefix=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-linux-gnueabihf-linux/install/arm-linux-gnueabihf/libc --
enable-c99 --enable-long-long --with-mode=thumb --disable-multilib --
with-float=hard
Thread model: posix
gcc version 4.9.2 20140904 (prerelease) (crosstool-NG
linaro-1.13.1-4.9-2014.09 - Linaro GCC 4.9-2014.09)
```

Toolchain (4.7.3)

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

- Download #1
- Download #2

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

host

```
$ sudo mkdir -p /opt/toolchains
$ sudo tar jxvf gcc-linaro-arm-linux-
gnueabihf-4.7-2012.12-20121214_linux.tar.bz2 -C /opt/toolchains/
```

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

host

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
export PATH=/opt/toolchains/gcc-linaro-arm-linux-
gnueabihf-4.7-2013.04-20130415_linux/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.

host

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

You can check if the toolchain installed above works properly while checking the version of toolchain. If you can find **gcc version 4.7.3 20130328 (prerelease)** at the end of the line, the toolchain is well installed.

host

```
$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/opt/toolchains/gcc-linaro-arm-linux-
gnueabihf-4.7-2013.04-20130415_linux/bin/../libexec/gcc/arm-linux-
gnueabihf/4.7.3/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-
gnueabihf-linux/.build/src/gcc-linaro-4.7-2013.04/configure --
```

```
build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --
target=arm-linux-gnueabihf --prefix=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-linux-gnueabihf-linux/install --with-
sysroot=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/install/arm-linux-gnueabihf/libc --enable-languages=c,c++,fortran
--enable-multilib --with-arch=armv7-a --with-tune=cortex-a9 --with-
fpu=vfpv3-d16 --with-float=hard --with-pkgversion='crosstool-NG
linaro-1.13.1-4.7-2013.04-20130415 - Linaro GCC 2013.04' --with-
bugurl=https://bugs.launchpad.net/gcc-linaro --enable-__cxa_atexit --
enable-libmudflap --enable-libgomp --enable-libssp --with-
gmp=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
mpfr=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
mpc=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
ppl=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
cloog=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-
libelf=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static --with-host-libstdcxx='-
L/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-
linux/.build/arm-linux-gnueabihf/build/static/lib -lpwl' --enable-
threads=posix --disable-libstdcxx-pch --enable-linker-build-id --
enable-gold --with-local-prefix=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-linux-gnueabihf-linux/install/arm-linux-gnueabihf/libc --
enable-c99 --enable-long-long --with-mode=thumb
Thread model: posix
gcc version 4.7.3 20130328 (prerelease) (crosstool-NG
linaro-1.13.1-4.7-2013.04-20130415 - Linaro GCC 2013.04)
```

Checkout

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

Android

host

```
$ git clone --depth 1 https://github.com/hardkernel/linux.git -b
odroidc-3.10.y-android
$ cd linux
```

Linux

both

```
$ git clone --depth 1 https://github.com/hardkernel/linux.git -b
odroidc-3.10.y
$ cd linux
```

Compile

You must do kernel configuration for **ODROID-C1**, then start to build. Adding **-j** option on make command will help you to finish compiling faster.

both

```
$ make odroidc_defconfig
$ make <-j4>
$ make <-j4> modules
```

You have done to compile the Linux kernel (zImage), the device tree file (.dtb) and kernel modules (.ko). But in order to boot on **ODROID-C1**, you need **uImage** apart from **zImage** which have boot information and simply can make this.

both

```
$ make uImage
```

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.

both

```
$ make menuconfig
```

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

both

```
$ make <-j4>
$ make <-j4> 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.

both

```
$ cp .config arch/arm/configs/odroidc_defconfig
$ git add arch/arm/configs/odroidc_defconfig
$ git commit -s -m "blahblahblah"
$ 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, **uImage**, to the boot card.

host

```
$ sudo fastboot flash boot <path/of/your/uImage>
```

This is to install a device tree file, **meson8b_odroidc.dtb**.

host

```
$ sudo fastboot flash dtb <path/of/your/meson8b_odroidc.dtb>
```

Linux (Cross build)

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

A partition for kernel, dtb and initrd is necessary. Run fdisk <device> and create a **root partition** (partition 1).
If you plan to have your rootfs on the sdcard on another partition you can put a minimum of +60M for the last sector of this root partition.
Change type to FAT16 (“t” option, value 6) and type “w” to save changes. Format it with mkfs.vfat /dev/sd<device>1 and put your kernel, dtb, boot.ini and initrd there. You should check the start Address of root partition from here

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 uImage and DT((meson8b_odroidc.dtb) to the FAT partition(1st partition) in the Boot-Device.

both

```
$ mkdir -p mount
$ sudo mount /dev/sdc1 ./mount
$ sudo cp arch/arm/boot/uImage arch/arm/boot/dts/meson8b_odroidc.dtb
./mount && sync && sudo umount ./mount
```

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

both

```
$ sudo mount /dev/sdc2 ./mount
$ sudo make modules_install ARCH=arm INSTALL_MOD_PATH=./mount && sync
&& sudo umount ./mount
$ rm -rf mount
```