

[Get started](#)[Open in app](#)

Ilyas Hamadouche

41 Followers

[About](#)[Follow](#)

Creating a Cross-Platform Toolchain for Raspberry Pi 4

Learn how to build customized toolchains



[Ilyas Hamadouche](#) Dec 5, 2020 · 4 min read



Photo by [Todd Quackenbush](#) on [Unsplash](#)

I have a Raspberry Pi 4B running a 64-bit [Ubuntu Server 20.04 LTS](#). I would like to compile some C++ applications on my local PC and run them on the RPi. To do so, I need a **cross-platform toolchain**, that helps me transform the source code into executables compatible with the RPi.

A **toolchain** is the set of C/C++ run-time libraries and the tools (mainly the compiler, assemble, and linker) that turns the source code into binaries that the target hardware can run. **Cross-platform** means the tools run on one platform while building applications for another [1].

In this story, I will show you how I created one!

Crosstool-NG

One of the favorite and easy-to-use tools to create stand-alone toolchains on the embedded Linux domain is Crosstool-NG [2]. It's an open-source utility that supports different architectures including ARM, x86, PowerPC, and MPIC, and has a menuconfig-style interface similar to the one of Linux kernel.

1. To use it, first install the dependencies:

```
~$ sudo apt-get update
~$ sudo apt-get install automake bison chrpath flex g++ git gperf
gawk help2man libexpat1-dev libncurses5-dev libsdl1.2-dev libtool
libtool-bin libtool-doc python2.7-dev texinfo
```

2. Get the source code:

```
~$ git clone https://github.com/crosstool-ng/crosstool-ng.git
~$ cd crosstool-ng/
~/crosstool-ng$ git checkout crosstool-ng-1.24.0
```

3. Build and install:

```
~/crosstool-ng$ ./bootstrap
~/crosstool-ng$ ./configure --enable-local
~/crosstool-ng$ make
~/crosstool-ng$ sudo make install
```

Creating the toolchain for RPi 4

After building Crosstool-NG, the next step is to create the toolchain for RPi 4.

Selecting a base-line configuration

By default, Crosstool-NG comes with a few ready-to-use configurations. You can see the full list by typing:

```
~/crosstool-ng$ ./ct-ng list-samples
```

At the time of writing, there are no configurations for RPi 4. To avoid creating one from scratch, we can use an existing configuration of RPi 3 and modify it to match RPi 4.

The one from the list that is close to our target is **aarch64-rpi3-linux-gnu**. To get more details about it, run:

```
~/crosstool-ng$ ./ct-ng show-aarch64-rpi3-linux-gnu
```

You will see something like:

```
[L...] aarch64-rpi3-linux-gnu
  Languages      : C,C++
  OS             : linux-4.20.8
  Binutils       : binutils-2.32
  Compiler       : gcc-8.3.0
  C library      : glibc-2.29
  Debug tools    : gdb-8.2.1
  Companion libs : expat-2.2.6 gettext-0.19.8.1 gmp-6.1.2 isl-0.20
libiconv-1.15 mpc-1.1.0 mpfr-4.0.2 ncurses-6.1 zlib-1.2.11
  Companion tools :
```

Select **aarch64-rpi3-linux-gnu** as a base-line configuration by typing:

```
~/w/crosstool-ng$ ./ct-ng aarch64-rpi3-linux-gnu
```

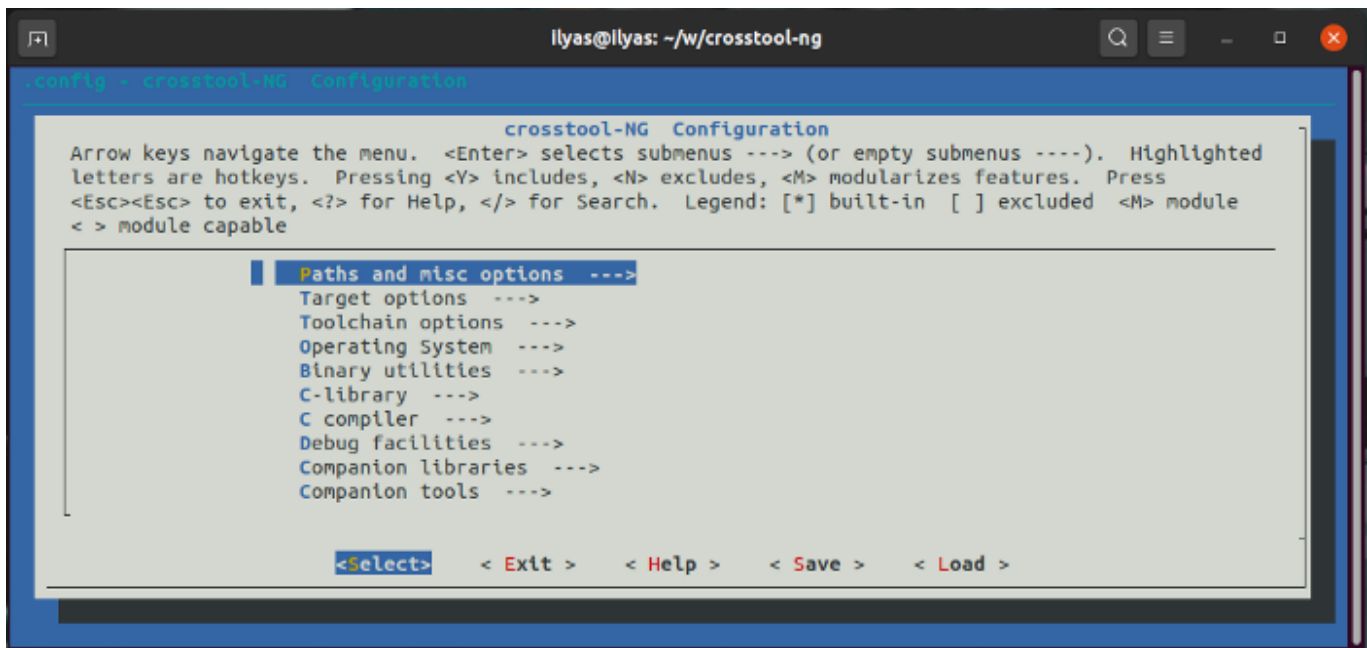
Customization

Before we do adjustments to the base-line configuration, we need information about the CPU being utilized on the RPi 4 board. From the [specifications](#), we can read that the processor being used is:

*Broadcom BCM2711, Quad core **Cortex-A72** (ARM v8) 64-bit SoC @ 1.5GHz*

To start the customization, open **menuconfig**:

```
~/w/crosstool-ng$ ./ct-ng menuconfig
```



Crosstool-NG interactive menu.

Make 3 changes:

1. Allow extending the toolchain after it is created (by default, it is created as read-only):

Paths and misc options -> Render the toolchain read-only -> false

2. Change the ARM Cortex core:

Target options -> Emit assembly for CPU: Change cortex-a53 to cortex-a72

3. Change the tuple's vendor string:

Toolchain options -> Tuple's vendor string: Change rpi3 to rpi4

Build

To create the toolchain, type:

```
~/crosstool-ng$ ./ct-ng build
```

The toolchain will be named **aarch64-rpi4-linux-gnu** and created in `${HOME}/x-tools/aarch64-rpi4-linux-gnu`.

Using the toolchain

Hello world!

To test the toolchain, I created a simple hello world program. If we can compile and run it on target, we can say that the toolchain works.

The code is as simple as:

```
1  #include<iostream>
2
3  int main(){
4      std::cout << "Hello world! \n";
5      return 0;
6  }
```

hello_world.cpp hosted with ❤ by GitHub

[view raw](#)

In order to compile it using our toolchain, first, we need to add the **bin** dir to PATH so that we can use the tools:

```
$ PATH=$PATH:~/x-tools/aarch64-rpi4-linux-gnu/bin
```

Compile using the following command:

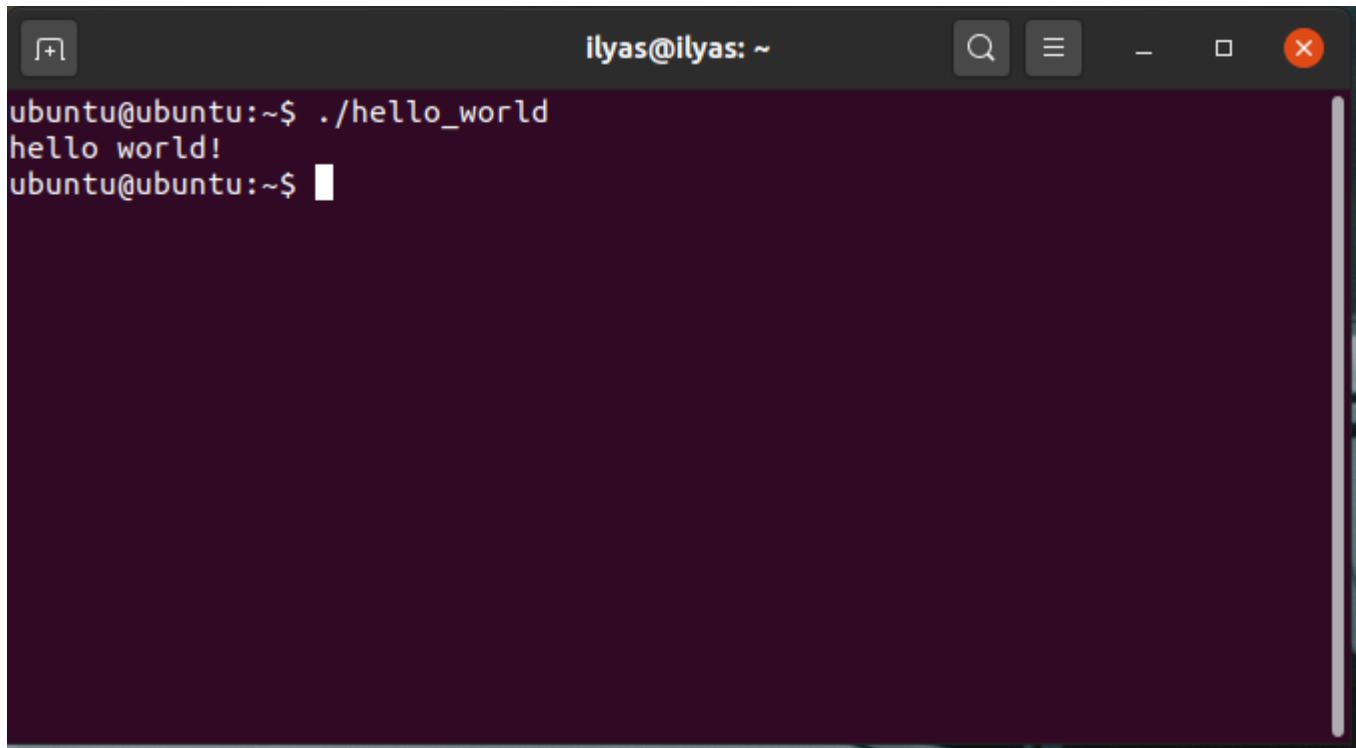
```
~/ $ aarch64-rpi4-linux-gnu-g++ hello_world.cpp -o hello_world
```

There should be no build errors and a file named **hello_world** is created.

Note: For compiling C files, you can use **aarch64-rpi4-linux-gnu-gcc** instead.

Testing on target

Copy **hello_world** to the target (running Ubuntu Server 20.04.1 LTS), and execute it. You should see the following output:

A terminal window titled 'ilyas@ilyas: ~' with standard window controls. The terminal shows the command './hello_world' being executed, resulting in the output 'hello world!'. The prompt 'ubuntu@ubuntu:~\$' is visible on both lines.

```
ilyas@ilyas: ~
ubuntu@ubuntu:~$ ./hello_world
hello world!
ubuntu@ubuntu:~$
```

This is a sign that the program is working! The toolchain can be used to compile other libraries and tools!

This is the end of the article. If you think this is useful, share it to spread the knowledge!

In case you have a board different than RPi4, you still can use the same steps described here to build your own customized cross-platform toolchain for your board.

References

- [1] Building Embedded Linux Systems 2nd edition, by Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum
- [2] Mastering Embedded Linux Programming 2nd edition, by Chris Simmonds
<https://www.amazon.de/Mastering-Embedded-Linux-Programming-potential/dp/1787283283>

- <https://crosstool-ng.github.io/>

About me

I am Ilyas Hamadouche, a Senior Software Engineer at [Elektrobit Automotive](#). I am interested in automotive software, embedded systems, and computer vision. Follow me on [Twitter](#) and [LinkedIn](#).

[Raspberry Pi](#)[Linux](#)[Open Source](#)[DIY](#)[Embedded Systems](#)[About](#) [Write](#) [Help](#) [Legal](#)

Get the Medium app

