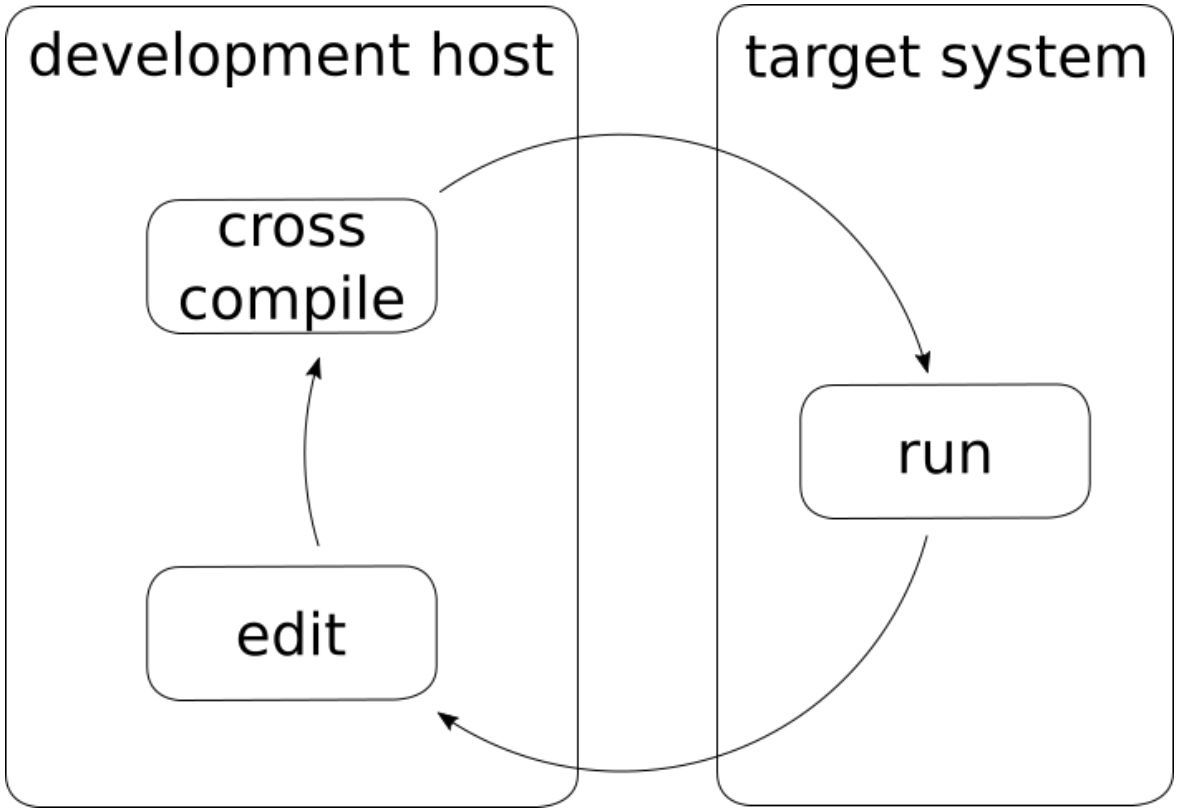
Cross Compiling For Embedded Debian Target Systems

Matthias Lüscher, 2. July 2018

Content

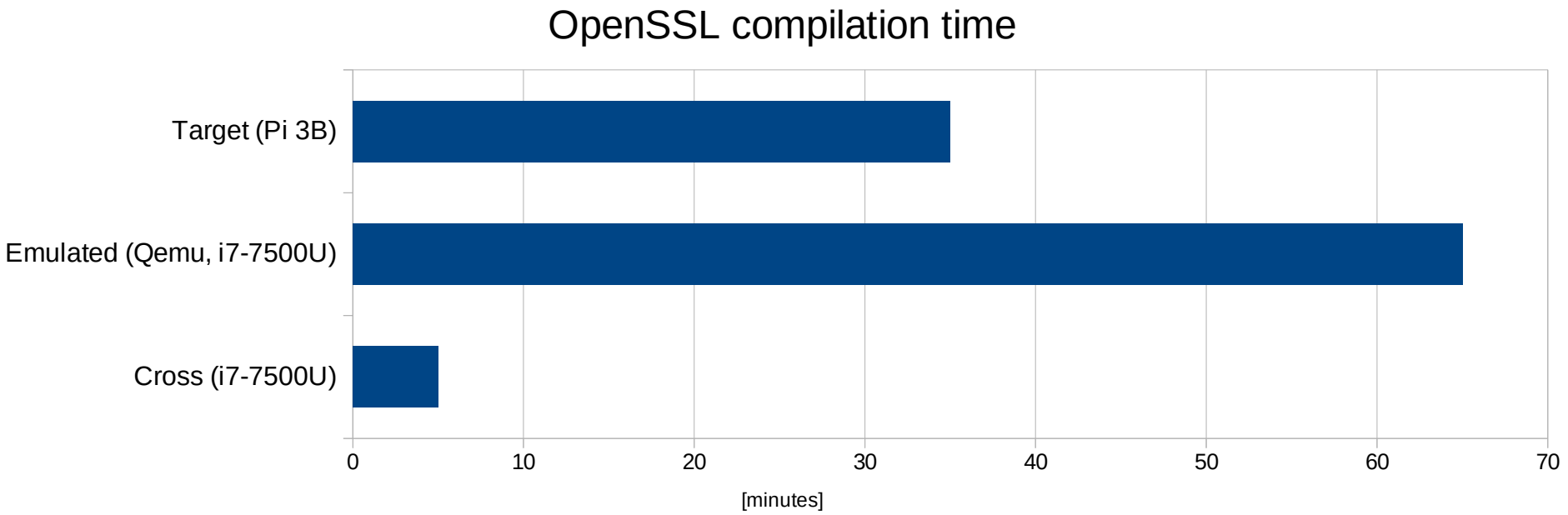
- What is cross compiling?
- Why should I cross compile?
- How should I cross compile with Debian?
- How can I manage my cross compilation tool chain?
- How can I integrate my favorite IDE?
- What are the best practices?
- Where can I get more information?

What is cross compiling?



- Wikipedia: A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running.
- Easy: Kernel (no dependencies, well prepared for cross compilation)
- Less easy: Libraries and executables (dependencies, maybe not cross compilation aware)

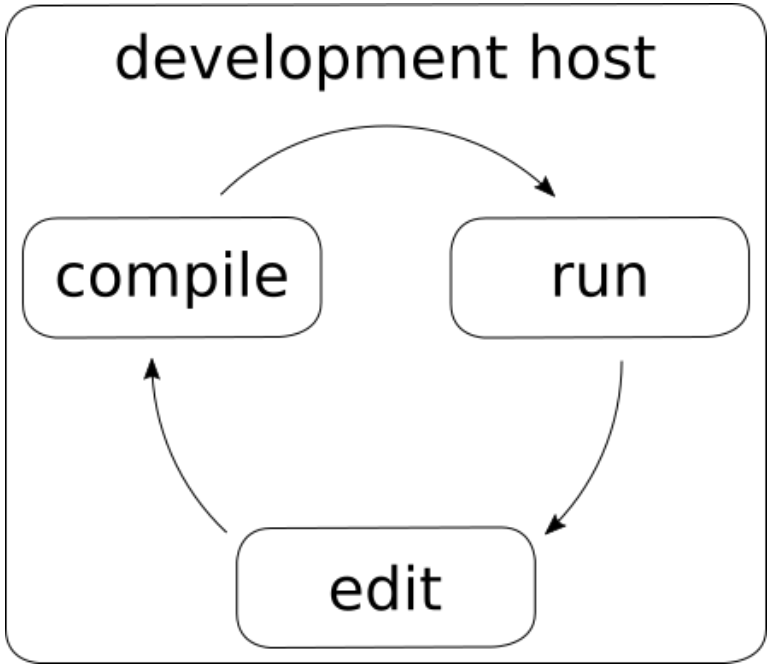
Why cross compile?

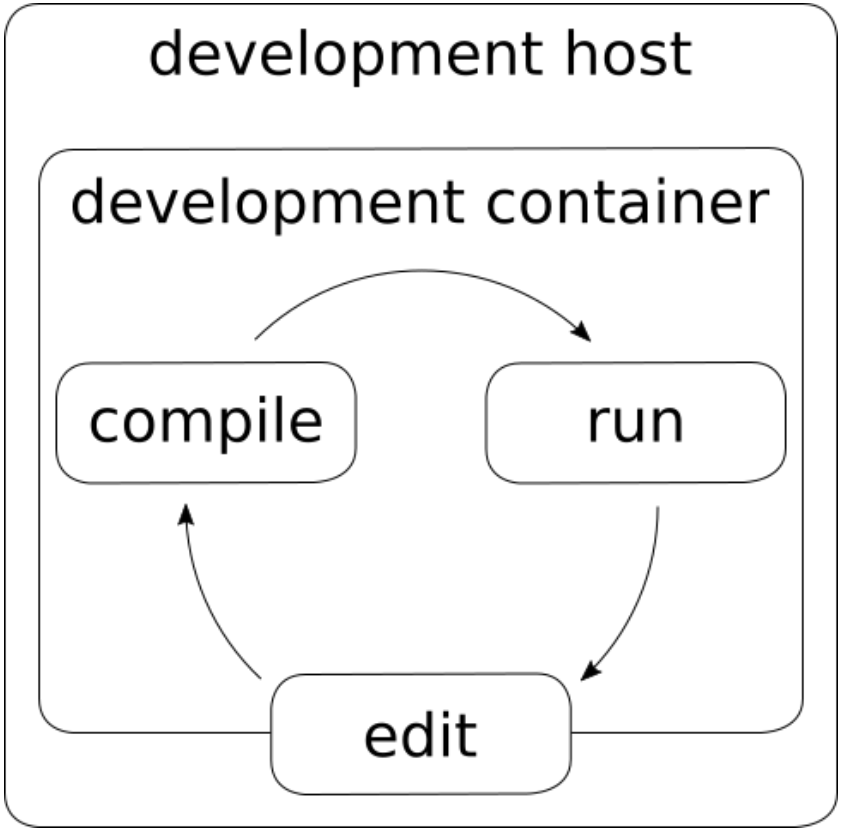


- Speed: Cross compilation is a lot faster!
- Flash: The target system might not have enough flash for compilation.
- Memory: The target system might run out of memory during compilation.

More information: http://www.get-edi.io/Compiling-for-Embedded-Debian-Target-Systems/

How should I cross compile with Debian?





- Environment is “self contained”: For Debian stretch you build within a Debian stretch.
- The Debian project does build ARM packages on ARM hardware.
- For a long time Debian was not well suited for cross compilation.

What has changed recently?

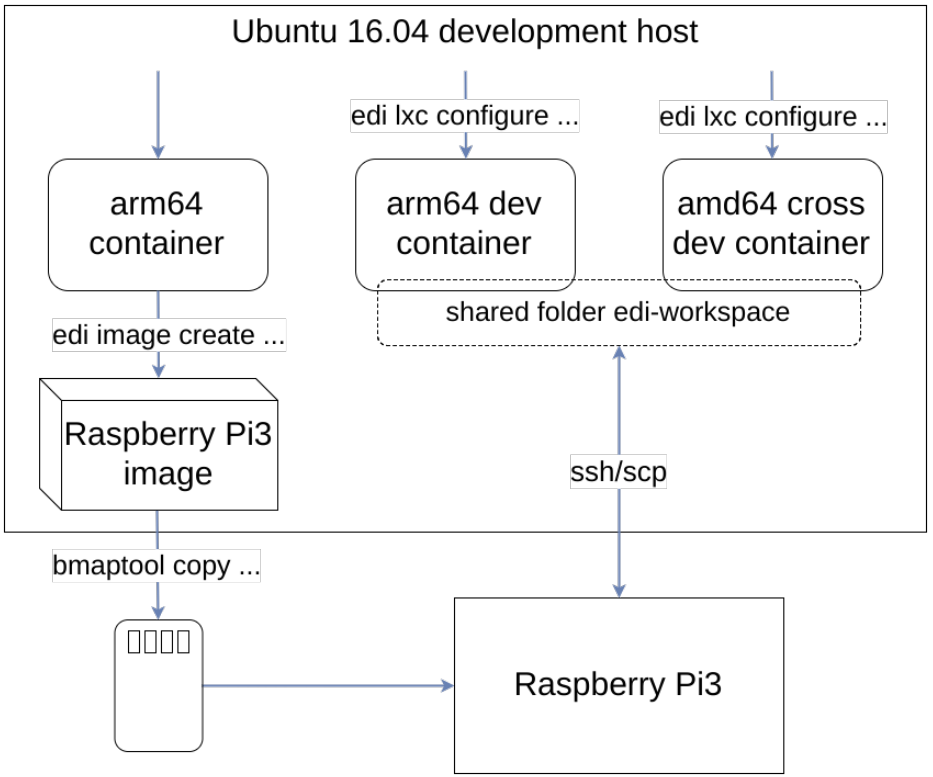
- Debian got broadly adopted for embedded devices.
- Multiarch and multilib got introduced in Debian wheezy: You can add a foreign architecture and install libraries and foreign libraries side by side:

```
sudo dpkg --add-architecture arm64
sudo apt update
sudo apt install <library>:arm64
```

- With Debian stretch the cross compilers became part of the main Debian repository:

```
sudo apt install crossbuild-essential-arm64
```

How can I manage my tool chain(s)?

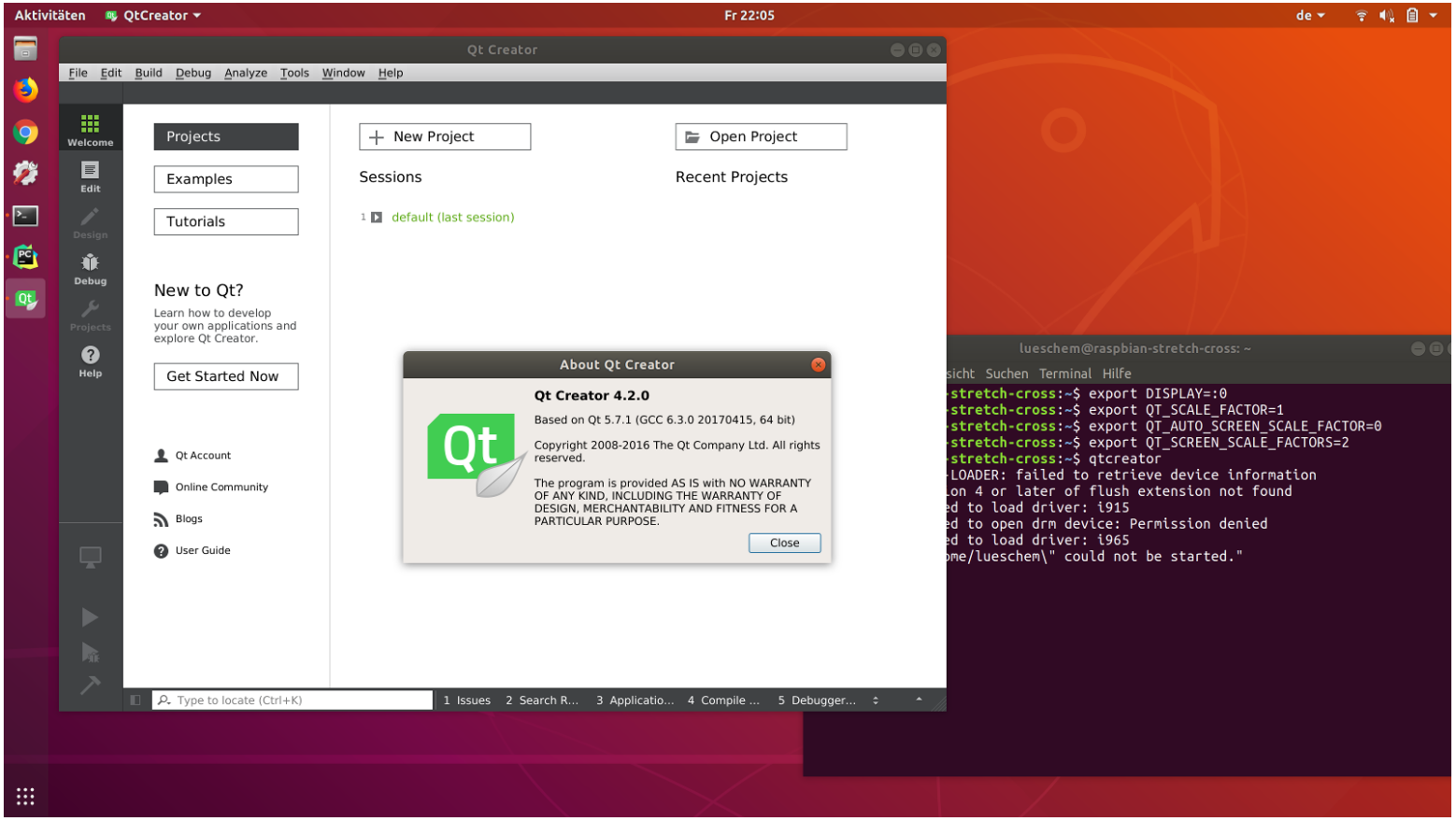


- Classical approach: Build a matching chroot (https://en.wikipedia.org/wiki/Chroot) environment and use it for cross compilation.
- Containerized approach: Build a Linux container containing a cross tool chain:

```
sudo edi -v lxc configure edi-pi-cross-dev pi3-stretch-arm64-cross-dev.yml
```

More information: http://www.get-edi.io/A-new-Approach-to-Operating-System-Image-Generation/

How can I integrate my favorite IDE?



- You can choose whatever IDE you like.
- To improve the overall handling, it is advisable to run the IDE within the development container.

More information: http://www.get-edi.io/Running-GUI-Applications-Within-LXD-Container/

What are the best practices?

- Do not take Debian for very resource constrained devices (consider using Yocto, ptxdist, buildroot etc. for such use cases).
- Make sure that the majority of your application can be developed and tested on the development host:

  - faster development cycle
  - easier to test (also in virtual environment)
  - portable to future hardware

- Use standard interfaces like USB and Ethernet to improve readiness for future hardware and emulated environments.

Conclusion

- Nowadays, Debian is a great choice for many embedded use cases.
- Since Debian stretch the cross compiler packages are part of the main repository.
- If your software is not a one-man business, it is advisable to automate the setup of the (cross) tool chain.
- Make sure that your software is for the most part hardware independent and enjoy the possibility to develop and test the software in a virtual amd64 environment.
- Take whatever IDE you like.

Where can I get more information?

- Debian wiki page about cross compilation: https://wiki.debian.org/CrossCompiling
- My personal (Debian biased) blog: http://www.get-edi.io/blog/
- sbuild: Tool for building Debian binary packages from Debian sources: https://wiki.debian.org/sbuild