Apertis Toolchain

Contents

The Apertis Toolchain provides a build environment to build Apertis for a foreign architecture from an operating system that is not Apertis.

The toolchain is intended for cross-compilation, targeting architectures that don’t match the CPU architecture of the build host (for instance, building ARM64 binaries from a Intel-based host). Currently Apertis provides toolchain for two architectures :

- 32 bits ARM (armhf)
- 64 bits ARM (AArch64)

It is meant to be used by teams that want to keep their development tools such as their preferred editor.

The toolchain only provides a compiler and minimal libraries to build code. The main supported language is C, but C++ is also supported. The toolchain provides a debugger with multiarch support for debugging foreign binaries. To build binaries using other libraries, either those libraries have to be built with the toolchain, or the developer can use a sysroot containing the libraries.

Sysroots are file system trees specifically meant for cross-compilation and remote debugging targeting a specific release image.

See Sysroots and devroots[1] for more information.

Usage of the toolchain

The toolchain for AArch64 and armhf can be downloaded on the Apertis release site[2].

We recommend following the guidance on product development on Apertis[3] when picking which release to use.

To extract the toolchain for AArch64 use the following commands:

```
$ wget https://images.apertis.org/release/v2020/v2020.2/toolchain/apertis-
aarch64-linux-gnu-toolchain.tar.xz
$ xz -d apertis-aarch64-linux-gnu-toolchain.tar.xz
$ tar xf apertis-aarch64-linux-gnu-toolchain.tar
$ cd apertis-aarch64-linux-gnu-toolchain/
```

Download and extract a sysroot. Ignore errors related to `mknod`. Be careful about the extraction of the sysroot. There is no directory prepended to the

[1] https://www.apertis.org/architecture/platform/sysroots-and-devroots/

[2] https://images.apertis.org/release

[3] https://www.apertis.org/policies/release-flow/#guidelines-for-product-development-on-top-of-apertis-and-its-direct-downstreams

extraction path of the sysroot archive. It can become problematic if extracted in the HOME directory. Make sure to create a work directory and change to it before extraction.

```
$ wget https://images.apertis.org/release/v2020/v2020.2/arm64/sysroot/sysroot-
apertis-v2020-arm64-v2020.2.tar.gz
mkdir ~/sysroot/
tar xf sysroot-apertis-v2020-arm64-v2020.2.tar.gz -C ~/sysroot/ --
exclude=./dev
```

Retrieve the project to build:

```
git clone git@gitlab.apertis.org:tests/bluez-phone-tester.git
```

From there, the steps are project dependent. Not all projects support cross compilation setup. It can usually be encompassed using environment variables.

```
$ ./autogen.sh
$ CC=~/apertis-aarch64-linux-gnu-toolchain/usr/bin/aarch64-linux-gnu-gcc-
8 PKG_CONFIG_SYSROOT_DIR=~/sysroot PKG_CONFIG_PATH=~/sysroot/usr/lib/aarch64-
linux-gnu/pkgconfig/ ./configure --host aarch64-linux-gnu
$ make CFLAGS="-I ~/sysroot/usr/include" LDFLAGS="-L~/sysroot/usr/lib/aarch64-
linux-gnu -L~/apertis-aarch64-linux-gnu-toolchain/usr/lib/../.. -L~/sysroot/usr/aarch64-
linux-gnu/lib/ --sysroot=~/sysroot"
$ file bluez-phone-tester
bluez-phone-tester: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /
linux-aarch64.so.1, for GNU/Linux 3.7.0, BuildID[sha1]=ca3fb7801ab26632208a6def82ac1dcf5cb40d10, not strippec
```

This project did not require specific dependencies, everything needed was in the sysroot. The sysroot are meant to be self sufficient for building the Apertis projects. If a new dependency is needed in the sysroot, the recipe should be modified. It is explicitly discouraged to install dependencies using different means because there should not be absolute symbolic links in the sysroot since they would not be pointing to the sysroot root folder. This has the potential to damage the system.