Speedrun

Pieter P

Setup of the necessary software on your computer

Install Docker

```
sudo apt update
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
# Add the Docker GPG key and PPA repository
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
source /etc/os-release
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu ${UBUNTU_CODENAME} stable"
# Install Docker
sudo apt install -y docker-ce
# Add your user to the docker group so you can run Docker without sudo
sudo usermod -aG docker ${USER}
# Log in again so group membership is up-to-date
su - ${USER}
```

Install some other tools

```
sudo apt install -y --no-install-recommends cmake make ninja-build git
```

Clone the RPi-Cpp-Toolchain repository

```
mkdir -p ~/GitHub
cd ~/GitHub
git clone --recursive https://github.com/tttapa/RPi-Cpp-Toolchain.git
cd RPi-Cpp-Toolchain
```

Pull the toolchain and the cross-compiled libraries from Docker Hub

```
./docker-arm-cross-build-scripts/build.sh # List available board configurations
# Replace the board name with one of the options printed by the previous command
./docker-arm-cross-build-scripts/build.sh rpi3-aarch64 --pull --export
```

Setup of the necessary software on the Raspberry Pi

OS installation and network configuration

If you haven't set up your Pi yet, follow the instructions on how to do so for Ubuntu or for Raspbian (TODO).

Make sure that you have the following set up correctly:

1. A working network connection to the same local network as your computer.
2. The hostname and DNS or mDNS configured correctly so you can find the Pi on the network as e.g. `rpi3.local`, even if its IP address changes.
3. An SSH configuration on your computer so you can SSH into the Pi using e.g. `ssh RPi3`.
4. An SSH key from your computer installed in the `authorized_keys` of the Pi so you can SSH into it without having to enter your password each time.

Installation of the staging area

We now have to install all libraries and tools that were cross-compiled in the previous steps to the Pi. This is as easy as just copying over the archive containing the staging area, and then extracting it to the root directory. All files in the staging area are either in `/usr/local` or in `/opt`, so it won't interfere with software you installed through the package manager.

```
# Copy the staging area archive to the Pi
# This command may take a while, depending on the speed of your SD card
scp ./docker-arm-cross-build-scripts/staging-aarch64-rpi3-linux-gnu.tar RPi3:/tmp
# Install everything to the root of the filesystem
# (will only install to /usr/local and /opt)
# Enter the sudo password of the Pi if necessary
# This command may take a while, depending on the speed of your SD card
ssh -t RPi3 "sudo tar xf /tmp/staging-aarch64-rpi3-linux-gnu.tar \
    -C / --strip-components=1 --keep-directory-symlink \
    --no-same-owner --no-same-permissions --no-overwrite-dir"
# Configure dynamic linker run-time bindings so newly installed libraries
# are found by the linker
# Enter the sudo password of the Pi if necessary
ssh -t RPi3 "sudo ldconfig"
```

To verify that the installation was successful, you can try running the newly installed Python interpreter:

```
# Check that Python 3.10 and OpenCV were installed correctly
ssh RPi3 "python3.10 -c 'import cv2; print(cv2.__version__)'"
```

Try out the C++ example project

```
# See what toolchain files are available, use the one that matches your board.
ls cmake
# Configure the project using the correct toolchain.
cmake -S. -Bbuild \
    -DCMAKE_TOOLCHAIN_FILE="cmake/aarch64-rpi3-linux-gnu.cmake" \
    -DCMAKE_BUILD_TYPE=Debug
# Build the project.
cmake --build build -j
# Package the project.
pushd build; cpack; popd
# Copy the project to the Raspberry Pi.
ssh RPi3 tar xz < build/greeter-1.0.0-Linux-arm64.tar.gz
# Run the hello world program on the Pi.
ssh -t RPi3 greeter-1.0.0-Linux-arm64/bin/hello-world
```