The Build Process

of (GNU Tools for Arm Embedded Processors 8-2018-q4-major)
2018-12

Table of Contents

Preface

This manual provides a step-by-step guide to help you build '`GNU Tools for Arm Embedded Processors`' on a newly installed Ubuntu 14.04 LTS 64-bit operating system.

Note that the steps below may most likely also work on an Ubuntu which is not newly installed or version other than 14.04 LTS, but it is not guaranteed. In this case please go through Appendix A [Known Issues], page 7 before you go, and you need to solve any other problems you may encounter by yourself. We highly appreciate if you could share the problems and solutions with us.

1 Build GNU Tools for Linux and Windows Platforms

1.1 Install Ubuntu

Ubuntu 14.04.5 ISO image is available from http://releases.ubuntu.com/14.04/ubuntu-14.04.5-desktop-amd64.iso. You can install it as a native system or a virtual machine.

1.2 Install Dependencies

1.2.1 Install dependencies available in Ubuntu's repositories

Execute the commands in this section to install the tools needed to build the toolchain. Lines starting with '$' denote commands that need to be input as is while lines starting with '#' are comments and as such do not need to be typed in.

Please note that the "Ignoring Provides line" and "unknown Multi-Arch type" warnings when executing `apt-get update` are harmless and can thus safely be ignored. Similarly, the warning about `update-alternatives` skipping the creation of symbolic links when executing `apt-get install` is also harmless and can therefore safely be ignored.

```
# Start root session
$ sudo su

# Add extra repositories to be used by APT
$ apt-get install software-properties-common
$ add-apt-repository universe
$ cat >/etc/apt/sources.list.d/xenial.list <<EOF
deb http://archive.ubuntu.com/ubuntu xenial main universe
deb-src http://archive.ubuntu.com/ubuntu xenial main universe
deb http://security.ubuntu.com/ubuntu xenial-security main
EOF

# Ensure package for Ubuntu Trusty are chosen by default
$ echo 'APT::Default-Release "trusty";' > /etc/apt/apt.conf.d/00default

# Enable use of 32bit packages
$ dpkg --add-architecture i386
$ apt-get update
```

```
# Install packages
$ apt-get install -y -t xenial \
    gcc-mingw-w64-i686 g++-mingw-w64-i686 binutils-mingw-w64-i686
$ apt-get -f install -y \
    build-essential \
    autoconf \
    autogen \
    bison \
    dejagnu \
    flex \
    flip \
    gawk \
    git \
    gperf \
    gzip \
    nsis \
    openssh-client \
    p7zip-full \
    perl \
    python-dev \
    libisl-dev \
    scons \
    tcl \
    texinfo \
    tofrodos \
    wget \
    zip \
    texlive \
    texlive-extra-utils \
    libncurses5-dev

# End root session
$exit
```

1.3 Build GNU Tools for Arm Embedded Processors

You are now ready to build the toolchain. Just follow the below instructions, substituting `~/toolchain` by the directory in which you wish to build the toolchain. Note that if you are not interested in the Windows toolchain, you can speed up the build by passing the option `--skip_steps=mingw32` to **all** of install-sources.sh, build-prerequisites.sh and build-toolchain.sh.

```
# Create a directory in which to build the toolchain and copy the source
# release package into it.
$ mkdir ~/toolchain
$ cp gcc-arm-none-eabi-8-2018-q4-major-src.tar.bz2 ~/toolchain

# Untar the source tarball.
$ cd ~/toolchain
$ tar -xjf gcc-arm-none-eabi-8-2018-q4-major-src.tar.bz2
$ cd ./gcc-arm-none-eabi-8-2018-q4-major
$ ./install-sources.sh

# Build the toolchain(s).
$ cd ../
$ ./build-prerequisites.sh
$ ./build-toolchain.sh
```

Once the build completes you can find the binary and source tarballs in '`~/toolchain/gcc-arm-none-eabi-8-2018-q4-major/pkg`' along with the md5 checksum.

2 Build GNU Tools on Mac OS X

In addition to the build on Ubuntu, the build scripts in same source package can also be used on Mac OS X to natively build a tool chain whose host is Mac OS X and target is arm-none-eabi. In this step we will describe how to install required software components and how to execute the build scripts. After this step you should be able to generate a same tool chain with the one released. This build process has been tested on Mac OS X 10.13.

2.1 Prepare a Mac OS X environment

The hardware should be an x86-based Mac machine like iMac. The installed OS should be Mac OS X which is updated to 10.13. The way to find out the Mac OS X version information is to click the **Apple** menu and choose **About This Mac**.

2.2 Install the latest Command Line Tools for Xcode

This component is originally part of Apple Xcode but can be installed separately without Xcode. It can be freely obtained from Apple's official website https://developer.apple.com/downloads/index.action. A valid Apple ID is required to login and download.

2.3 Install MacTeX to build PDF format documents

This is an optional step and can be skipped if PDF format documents aren't needed. Please refer to https://tug.org/mactex/mactex-download.html and follow the instructions given there.

2.4 Build the tool chain under Mac OS X

With all the dependent packages installed, we can start to natively build the tool chain on Mac OS. Following are the commands and steps we are using:

```
# Copy the src release package into ~/mac-build/ directory
$ cp gcc-arm-none-eabi-8-2018-q4-major-src.tar.bz2 ~/mac-build

# Prepare source codes
$ cd ~/mac-build
$ tar xjf gcc-arm-none-eabi-8-2018-q4-major-src.tar.bz2
$ cd ./gcc-arm-none-eabi-8-2018-q4-major
$ ./install-sources.sh

# Start building the toolchain.
$ ./build-prerequisites.sh
$ ./build-toolchain.sh
```

Appendix A Known Issues

- If you are using different build environment and tools, you might run into a problem where binutils can not be successfully built. This is probably caused by binutils bug 13036. For more information, please refer to http://sourceware.org/bugzilla/show_bug.cgi?id=13036.
- Some shell scripts in gcc and other packages are incompatible with the dash shell, which is the default /bin/sh for Ubuntu 14.04 LTS. You must make /bin/sh a symbolic link to one of the supported shells: saying bash. Here on Ubuntu 14.04 LTS system, this can be done by running following command:

  ```
  $ sudo dpkg-reconfigure -plow dash
  ```

  Then choose '`No`' in the '`Configuring dash`' popup dialog and press enter. You can run following command and check that /bin/sh points to '`bash`':

  ```
  $ ls -l /bin/sh
  ...... /bin/sh -> bash
  ```