The Build Process

of (GNU Tools for ARM Embedded Processors)
2014-01

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 9, 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

```
$ echo \
  "deb http://archive.ubuntu.com/ubuntu/ trusty main restricted universe multiverse" \
  > /etc/apt/sources.list

$ echo "deb http://security.ubuntu.com/ubuntu trusty-security main" \
     >> /etc/apt/sources.list

$ echo "deb http://archive.ubuntu.com/ubuntu xenial main universe"  \
     > /etc/apt/sources.list.d/xenial.list

$ echo "deb-src http://archive.ubuntu.com/ubuntu xenial main universe" \
     >> /etc/apt/sources.list.d/xenial.list

$ echo 'APT::Default-Release "trusty";' > /etc/apt/apt.conf.d/00default

$ dpkg --add-architecture i386

$ sudo apt-get update

$ sudo apt-get install -y -t xenial \
  gcc-mingw-w64-i686 g++-mingw-w64-i686 binutils-mingw-w64-i686

$ sudo apt-get -f install -y \
    apt-src \
    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
```

Note that there may be some warnings about Multi-Arch for some packages and update alternatives for MinGW32 GCC. Ignore these as they dont affect the build.

1.2.1 Download and deploy prebuilt native tools

In order to save effort to prepare the native build tools, we provide prebuilt ones at website `https://launchpad.net/gcc-arm-embedded-misc/native-build-tools/20150408`. The related source package and script are also provided. Please download the tools.

1.3 Build GNU Tools for ARM Embedded Processors

You are now ready to build the toolchain.

```
# Copy the src release package into ~/toolchain/ directory
$ cp gcc-arm-none-eabi-4_7-2013q1-20130106-src.tar.bz2 ~/toolchain

# Untar the source tarball.
$ cd ~/toolchain
$ tar -xjf gcc-arm-none-eabi-4_7-2013q1-20130106-src.tar.bz2
$ cd ./gcc-arm-none-eabi-4_7-2013q1-20130106/src
$ find -name '*.tar.*' | xargs -I% tar -xf %
$ wget http://nsis.sourceforge.net/mediawiki/images/a/ad/EnvVarUpdate.7z
$ tar xf <path to prebuilt tools>/prebuilt-native-tools.tar.lzma \
  --strip-components=1 prebuilt-native-tools/python-win
$ cd ../

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

# You can specify "--skip_steps=mingw32" option to skip building Windows host
# toolchain. If you specify this option when building prerequisites,
# you have to specify it when building toolchain too.
```

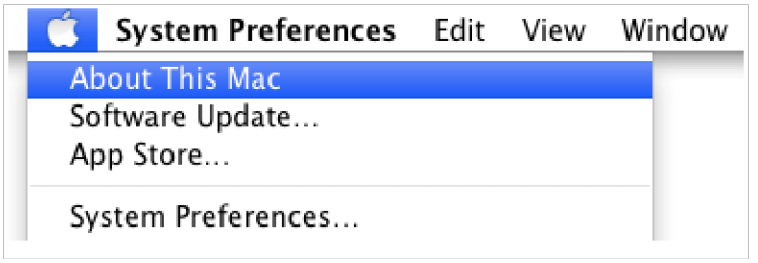
Once the build completes you can find the binary and source tarballs in '`~/toolchain/gcc-arm-none-eabi-4_7-2013q1-20130106/pkg`' along with the md5 checksum.

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. Due to resource limit, this build process is only tested against Mac OS X 10.7.3 along with components listed below.

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 10.7.3. The way to find out the Mac OS X version information is to click the **Apple** menu and choose **About This Mac**.



For the environment we are using, it looks as below:

2.2 Install the 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. The one we are using is in the item named '`Command Line Tools (OS X Lion) for Xcode - April 2013`'. After the download finishes, just double click the '`.dmg`' file and follow the instructions to install it.

2.3 Install a native version of gcc 5 using MacPorts

Download and install MacPorts for Mac OS X Lion following the steps on MacPorts website `https://www.macports.org/install.php#installing`. Then use MacPorts to install gcc5 with the following commands:

```
# Install gcc 5
$ sudo port install gcc5

# Make sure the newly installed gcc is the one reachable by PATH
$ sudo ln -s /opt/local/bin/{x86_64-apple-darwin11-gcc-5.*,gcc}
$ sudo ln -s /opt/local/bin/{g++-mp-5,g++}
$ export PATH=/opt/local/bin:$PATH
```

2.4 Install MacTeX to build PDF format documents

This is an optional step and can be skipped if PDF format documents aren't needed. The build process will use TeX engineer provided by MacTeX-2012 to generate PDF format documents. This component can be freely obtained from its official FTP server `ftp://ftp.tug.org/historic/systems/mactex/2012/MacTeX.pkg`. Its original size is approximately 2.1G. Once downloaded, just double click on the '`MacTeX.pkg`' file and follow the instructions to install it. By default the related TeX executable files won't be installed into the default path like '`/usr/bin`', so the Terminal need to be restarted before running the build scripts.

2.5 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-4_7-2013q1-20130106-src.tar.bz2 ~/mac-build

#Prepare source codes
$ cd ~/mac-build
$ tar xjf gcc-arm-none-eabi-4_7-2013q1-20130106-src.tar.bz2
$ cd ./gcc-arm-none-eabi-4_7-2013q1-20130106/src
$ find . -name '*.tar.*' | xargs -I% tar -xf %
$ cd ..

#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
  ```