The Build Process

of (GNU Tools for ARM Embedded Processors)
2016-06

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 8.10 operating system.

Note that the steps below may most likely also work on an Ubuntu which is not newly installed or version other than 8.10, but it is not guaranteed. In this case please go through Appendix A [Known Issues], page 10 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 on Ubuntu 8.10

1.1 Install Ubuntu

Ubuntu 8.10 ISO image is available from http://old-releases.ubuntu.com/releases/8.10/ubuntu-8.10-desktop-i386.iso. You can install it as a native system or a virtual machine. The command lines provided in this document are all using user id '`build`' as an example, so please create a new user called '`build`' in the system. Otherwise, you have to replace user id '`build`' with your own one.

1.2 Tune environment and install required softwares

1.2.1 Change /bin/sh to bash

Some shell scripts in gcc and other packages are incompatible with the dash shell, which is the default /bin/sh for Ubuntu 8.10. You must make /bin/sh a symbolic link to one of the supported shells: saying bash. Here on Ubuntu 8.10 system, this can be done by running following command firstly:

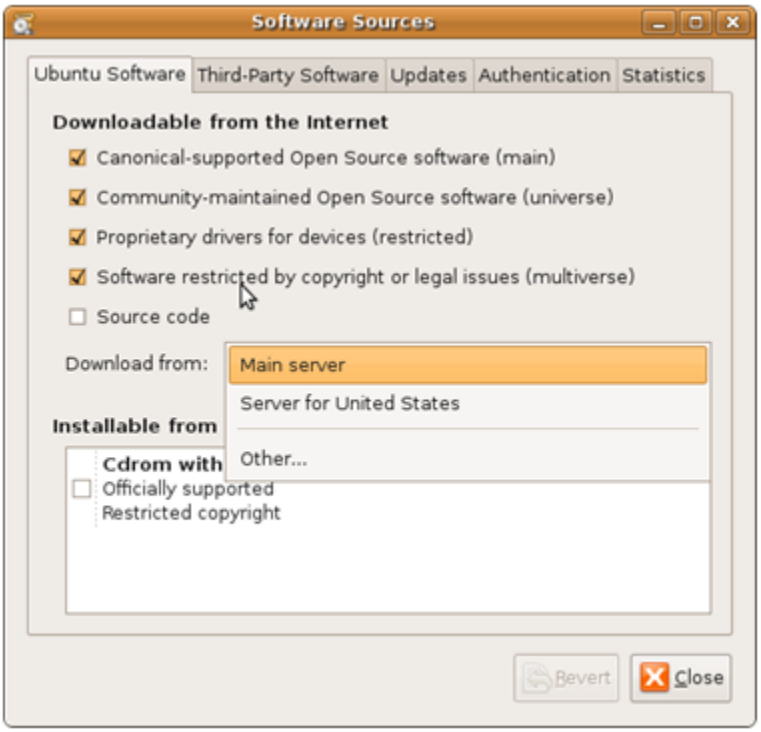
```
$ sudo dpkg-reconfigure -plow dash
```

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
```

1.2.2 Change software sources to Main server

On Ubuntu 8.10 system, click '`System->Administration->Software Sources`' to open '`Software Sources`' dialog, choose '`Main server`' in '`Download from:`' list box, then click '`close`'. You will be prompted by a window saying '`The information about available software is out-of-date`', please click '`Reload`'. And then there will be a warning message box popped up, which can be just ignored by clicking '`Close`'.



Edit the file using command line:

```
$ sudo vi /etc/apt/sources.list
```

replace all '`http://*.ubuntu.com`' with '`http://old-releases.ubuntu.com`' in that file, save and exit. Run following command to update package list. It should not fail, or else something has been wrong.

```
$ sudo apt-get update
```

1.2.3 Install common tools and libraries

Install common tools and libraries needed by build process with below command:

```
$ sudo apt-get install apt-src \
scons \
mingw32-runtime \
p7zip-full \
gawk \
gzip \
perl \
autoconf \
m4 \
automake \
libtool \
libncurses5-dev \
gettext \
gperf \
dejagnu \
expect \
tcl \
autogen \
guile-1.6 \
flex \
flip \
bison \
tofrodos \
texinfo \
g++ \
gcc-multilib \
libgmp3-dev \
libmpfr-dev \
debhelper \
texlive \
texlive-extra-utils
```

Note that the package management software might complain that several packages cannot be installed properly while installing texlive and texlive-extra-utils. It won't harm our building process, please just ignore it now. Some of those tools might be unnecessary, but it won't hurt if installed.

1.2.4 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 tool and decompress it to a proper place, it will be used in subsequent steps to build gcc arm embedded toolchain. The command to decompress it looks like:

```
tar xf prebuilt-native-tools.tar.lzma --lzma
```

Please be noted that those prebuilt tools are for Ubuntu 8.10 32-bit and not suitable for any other build platforms. For those working on other build platforms, please either prepare your own build tools and use them through option –build_tools of build script or just use the ones from your system by not specifying the –build_tools option.

1.3 Build GNU Tools for ARM Embedded Processors

If you download and decompress the prebuilt tools successfully, you have set up the building environment. You can now start to build the toolchain by yourself with below commands:

```
#Copy the src release package into ~/toolchain/ directory
$ cp gcc-arm-none-eabi-5_4-2016q2-20160622-src.tar.bz2 ~/toolchain
#Prepare source codes
$ cd ~/toolchain
$ tar -xjf gcc-arm-none-eabi-5_4-2016q2-20160622-src.tar.bz2
$ cd ./gcc-arm-none-eabi-5_4-2016q2-20160622/src
$ find -name '*.tar.*' | xargs -I% tar -xf %
$ cd ../
#Start building the toolchain.
#Can specify "--skip_steps=mingw32" option to skip building windows host
#toolchain, and if specify that option when building prerequisites,
#you have to specify it when building toolchain too.
#
#Without option --build_tools, the tools in current PATH will be used.
#You can download and deploy the provided prebuilt one, use it like
#./build-prerequisites.sh --build_tools=/home/build/prebuilt-native-tools
# or ./build-toolchain.sh --build_tools=/home/build/prebuilt-native-tools
$ ./build-prerequisites.sh --build_tools=YOUR_NATIVE_TOOLS_PATH
$ ./build-toolchain.sh --build_tools=YOUR_NATIVE_TOOLS_PATH
```

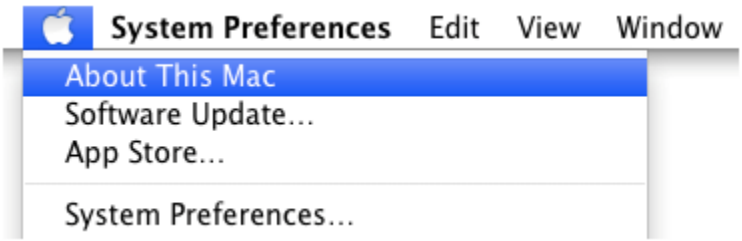
After this, you can '`cd`' into '`~/toolchain/gcc-arm-none-eabi-5_4-2016q2-20160622/pkg`' and find the built toolchain/source code packages and the md5 checksum file.

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 for Xcode - June 2012`'. After the download finishes, just double click the '`.dmg`' file and follow the instructions to install it.

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. 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.

2.4 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-5_4-2016q2-20160622-src.tar.bz2 ~/mac-build

#Prepare source codes
$ cd ~/mac-build
$ tar xjf gcc-arm-none-eabi-5_4-2016q2-20160622-src.tar.bz2
$ cd ./gcc-arm-none-eabi-5_4-2016q2-20160622/src
$ find . -name '*.tar.*' | xargs -I% tar -xf %
$ cd ..

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

Appendix A Known Issues

- The commands given in this document and build scripts sometimes have a problem when running on Ubuntu 9.10 due to a bug in the '`tar-1.22-1`' package. You can update '`tar`' to an higher version. For more information, please refer to https://bugs.launchpad.net/ubuntu/+source/tar/+bug/453330.
- 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.