

**Build Process
of
GNU Tools for ARM Embedded Processors**

2012-03-16

ARM Ltd.

Content

Step 0: Preface.....	4
Step 1: Install Ubuntu	5
1) Download Ubuntu-8.10.....	5
2) Install Ubuntu	5
Step 2: Tune environment and install some softwares.....	6
1) Change /bin/sh to bash	6
2) Change software sources to Main server.....	6
3) Install common tools and libraries	7
Step 3: Install MinGW and InstallJammer	8
1) Download mingw32 and related.....	8
2) Update/Build/Install mingw32 and related	9
3) Install InstallJammer	10
Step 4: Build new native gcc.....	12
Step 5: Build GNU Tools for ARM Embedded Processors	13
Appendix: Known Issues.....	14

Step 0: 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. But some of the steps can be ignored,

- 1) If you don't want to build the package running on windows, please ignore step 3.
- 2) If you don't want to build the package running on the platform which is different from your build machine, please ignore step 4.

Note that the steps below may most likely also work on an Ubuntu which is not newly installed or version above 8.10, but it is not guaranteed, so in any case you need to solve the problems you may encounter by yourself.

Step 1: Install Ubuntu

1) Download Ubuntu-8.10

You can download Ubuntu-8.10 iso image from the link below,

<http://old-releases.ubuntu.com/releases/8.10/ubuntu-8.10-desktop-i386.iso>

2) Install Ubuntu

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.

Step 2: Tune environment and install some softwares

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:

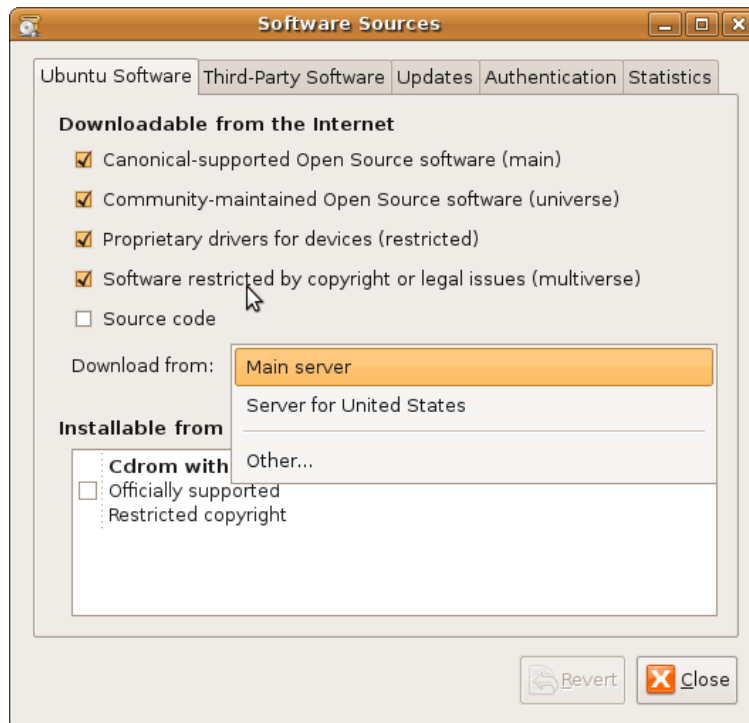
```
$ sudo dpkg-reconfigure -pnow dash
```

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
```

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
```

3) Install common tools and libraries

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

```
$ sudo apt-get install apt-src gawk gzip perl autoconf m4 automake libtool  
gettext gperf dejagnu expect tcl autogen guile-1.6 flex bison texinfo g++ gcc-  
multilib mingw32 mingw32-binutils mingw32-runtime 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.

Step 3: Install MinGW and InstallJammer

If you don't want to build package running on windows, please skip this step.

1) Download mingw32 and related

Mingw32 source and patch can be downloaded from following links,

* mingw32_4.2.1.dfsg-2ubuntu1.diff.gz:

https://launchpad.net/ubuntu/maverick/+source/mingw32/4.2.1.dfsg-2ubuntu1/+files/mingw32_4.2.1.dfsg-2ubuntu1.diff.gz

* mingw32_4.2.1.dfsg.orig.tar.gz:

https://answers.launchpad.net/ubuntu/+archive/primary/+files/mingw32_4.2.1.dfsg.orig.tar.gz

* mingw32_4.2.1.dfsg-2ubuntu1.dsc:

https://answers.launchpad.net/ubuntu/+archive/primary/+files/mingw32_4.2.1.dfsg-2ubuntu1.dsc

Save these files in ~/tools/mingw32/ directory.

Download the mingw32-binutils source and patch from following links:

* mingw32-binutils_2.20-0.1.diff.gz:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-binutils_2.20-0.1.diff.gz

* mingw32-binutils_2.20-0.1.dsc:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-binutils_2.20-0.1.dsc

* mingw32-binutils_2.20.orig.tar.gz:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-binutils_2.20.orig.tar.gz

Save these files in ~/tools/mingw32-binutils/ directory

Download the mingw32-runtime source and patch from following links:

* mingw32-runtime_3.15.2-0ubuntu1.diff.gz:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-runtime_3.15.2-0ubuntu1.diff.gz

* mingw32-runtime_3.15.2-0ubuntu1.dsc:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-runtime_3.15.2-0ubuntu1.dsc

* mingw32-runtime_3.15.2.orig.tar.gz:

https://launchpad.net/ubuntu/+archive/primary/+files/mingw32-runtime_3.15.2.orig.tar.gz

Save these files in ~/tools/mingw32-runtime/ directory

Note:

In this step, the *.diff.gz files are compressed patch files. But if you download these packages using internet explorer (such as IE, FireFox, etc.), it might automatically help you unzip these files. Please check the downloaded files using following commands:

```
#Here using mingw32_4.2.1.dfsg-2ubuntu1.diff.gz as an example,
#the same story stands for all *.diff.gz files
$ cd ~/tools/mingw32/
$ file mingw32_4.2.1.dfsg-2ubuntu1.diff.gz
mingw32_4.2.1.dfsg-2ubuntu1.diff.gz: ASCII English text
```

It says that 'mingw32_4.2.1.dfsg-2ubuntu1.diff.gz' is an ASCII text file, though file name has 'gz' suffix.

In this case, you could choose either following methods:

- a) Download the packages using "wget" command;
- b) Execute following commands to prepare the diff file:

```
$ cd ~/tools/mingw32/
$ mv mingw32_4.2.1.dfsg-2ubuntu1.diff.gz mingw32_4.2.1.dfsg-2ubuntu1.diff
```

Using this method, there is no need to unzip *.diff.gz in the next step.

2) Update/Build/Install mingw32 and related

Run the following commands to update mingw32 package

```
$ cd ~/tools/mingw32/
$ ls
mingw32_4.2.1.dfsg-2ubuntu1.diff.gz      mingw32_4.2.1.dfsg.orig.tar.gz
mingw32_4.2.1.dfsg-2ubuntu1.dsc
$ tar xzf mingw32_4.2.1.dfsg.orig.tar.gz
$ gzip -d mingw32_4.2.1.dfsg-2ubuntu1.diff.gz
$ mv mingw32-4.2.1.dfsg.orig mingw32-4.2.1.dfsg
$ cd mingw32-4.2.1.dfsg
$ patch -p1 <../mingw32_4.2.1.dfsg-2ubuntu1.diff
$ chmod a+x ./debian/rules
$ sudo dpkg-buildpackage
```

Everything should be ok and debian package “mingw32_4.2.1.dfsg-2ubuntu1_i386.deb” should be generated in ~/mingw32/ directory.

Install the generated package:

```
$ cd ~/tools/mingw32/  
$ sudo dpkg -i mingw32_4.2.1.dfsg-2ubuntu1_i386.deb
```

Run the following commands to update mingw32-binutils package

```
$ cd ~/tools/mingw32-binutils/  
$ ls  
mingw32-binutils_2.20-0.1.diff.gz      mingw32-binutils_2.20-0.1.dsc  
mingw32-binutils_2.20.orig.tar.gz  
$ tar xzf mingw32-binutils_2.20.orig.tar.gz  
$ gzip -d mingw32-binutils_2.20-0.1.diff.gz  
$ cd mingw32-binutils-2.20  
$ patch -p1 <../mingw32-binutils_2.20-0.1.diff  
$ chmod a+x ./debian/rules  
$ sudo dpkg-buildpackage
```

Install the generated package:

```
$ cd ~/tools/mingw32-binutils/  
$ sudo dpkg -i mingw32-binutils_2.20-0.1_i386.deb
```

Run the following commands to update mingw32-runtime package

```
$ cd ~/tools/mingw32-runtime/  
$ ls  
mingw32-runtime_3.15.2-0ubuntu1.diff.gz mingw32-runtime_3.15.2-0ubuntu1.dsc  
mingw32-runtime_3.15.2.orig.tar.gz  
$ tar xzf mingw32-runtime_3.15.2.orig.tar.gz  
$ gzip -d mingw32-runtime_3.15.2-0ubuntu1.diff.gz  
$ cd mingw32-runtime-3.15  
$ patch -p1 <../mingw32-runtime_3.15.2-0ubuntu1.diff  
$ chmod a+x ./debian/rules  
$ sudo dpkg-buildpackage
```

Install the generated package:

```
$ cd ~/tools/mingw32-runtime/  
$ sudo dpkg -i mingw32-runtime_3.15.2-0ubuntu1_all.deb
```

3) Install InstallJammer

Download latest InstallJammer installer (installjammer-1.2.15.tar.gz) from website:

<http://www.installjammer.com/>

Save it in “~/tools” directory, unpack it by running following command:

```
$ cd ~/tools
$ tar xzf installjammer-1.2.15.tar.gz
```

After unpacking, find the executable program named installjammer in the destination directory, for example “~/tools/installjammer”. Export the path in PATH environment variable by appending following lines at the end of ~/.bashrc file.

```
#For example, $YOUR_INSTALLJAMMER_PATH=/home/build/tools/installjammer
export PATH=$YOUR_INSTALLJAMMER_PATH:$PATH
```

Save .bashrc and quit, then restart or re-login your system. Check that installjammer is in your \$PATH by running below command,

```
$ which installjammer
$YOUR_INSTALLJAMMER_PATH/installjammer
#in my system, the output would be:
#/home/build/tools/installjammer/installjammer
```

Step 4: Build new native gcc

If you don't want to build the package running on the platform which is different from your build machine, please skip this step.

Reconfigure and compile native gcc/g++ on Ubuntu-8.10 system. Gcc distributed along with ubuntu-8.10 system supports ssp utility by default, which uses symbols of version GLIBC_2.8/GLIBC_2.7, etc. in glibc, resulting in programs compiled by this gcc/g++ may not be able to run on system with lower version glibc, like redhat5/redhat4. To provide greater application coverage of our toolchain, we have to reconfigure and compile gcc with libssp disabled.

Download gcc-4.3.6.tar.bz2 from website and save it in ~/tools/native-gcc/src directory:

<ftp://ftp.gnu.org/gnu/gcc/gcc-4.3.6/gcc-4.3.6.tar.bz2>

Reconfigure and compile gcc with following commands:

```
$ cd ~/tools/native-gcc/src
$ mkdir -p ~/tools/native-gcc/obj/gcc && mkdir ~/tools/native-gcc/target
$ tar -xjf gcc-4.3.6.tar.bz2
$ cd ../obj/gcc
$ ../../src/gcc-4.3.6/configure \
  --enable-languages=c,c++ \
  --enable-shared \
  --enable-threads=posix \
  --disable-decimal-float \
  --disable-libffi \
  --disable-libgomp \
  --disable-libmudflap \
  --disable-libssp \
  --disable-libstdcxx-pch \
  --disable-multilib \
  --disable-nls \
  --with-gnu-as \
  --with-gnu-ld \
  --enable-libstdcxx-debug \
  --enable-targets=all \
  --enable-checking=release \
  --prefix=/home/build/tools/native-gcc/target \
  --with-host-libstdcxx="-static-libgcc -L /usr/lib/gcc/i486-linux-
gnu/4.3.2/ -lstdc++ -lsupc++ -lm"
$ make
$ make install
```

After this, add path /home/build/tools/native-gcc/target/bin into your PATH environment, make sure to use the new gcc/g++ to compile our arm toolchain. You can do this by appending following line at the end of ~/.bashrc file:

```
export PATH=/home/build/tools/native-gcc/target/bin:$PATH
```

Then re-start or re-login the system.

Step 5: Build GNU Tools for ARM Embedded Processors

You have set up the building environment. You can now build the toolchain by yourself with below commands:

```
#Copy the src release package into ~/toolchain/ directory
$ cp gcc-arm-none-eabi-4_6-2012q1-20120316-src.tar.bz2 ~/toolchain
#Prepare source codes
$ cd ~/toolchain
$ tar -xjf gcc-arm-none-eabi-4_6-2012q1-20120316-src.tar.bz2
$ cd ./gcc-arm-none-eabi-4_6-2012q1-20120316/src
$ find -name '*.tar.*' | xargs -I% tar -xf %
#Since we should not modify zlib package,
#here provides a patch to compile it successfully.
$ cd zlib-1.2.5
$ patch -p1 <../zlib-1.2.5.patch
$ cd ../../
#Start building the toolchain.
#Can specify "--skip_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.
$ ./build-prerequisites.sh [--skip_mingw32]
$ ./build-toolchain.sh [--skip_mingw32]
```

After this, you can cd into “~/toolchain/gcc-arm-none-eabi-4_6-2012q1-20120316/pkg” and find the built toolchain/source code packages and the md5 checksum file.

Appendix: Known Issues

- 1) This document and build scripts sometimes have problem when running on ubuntu9.10 due to bug in tar-1.22-1 package. You can update 'tar' to higher version. For more information, please refer to <https://bugs.launchpad.net/ubuntu/+source/tar/+bug/453330> .
- 2) Due to binutils bug 13036, native gcc with version higher than 4.5.1 won't build binutils successfully. You should note this issue when trying to build the cross-toolchain in your specific host environment. For more information, please refer to http://sourceware.org/bugzilla/show_bug.cgi?id=13036.