Scratchbox toolchains

Ricardo Kekki

rkekki@movial.fi

Scratchbox toolchains
by Ricardo Kekki

Revision history

Version:	**Author:**	**Description:**
2005-04-01	Savola	Fixed a typo
2005-02-17	Savola	Fixed build option example
2005-02-07	Savola	Scratchbox 1.0 updates
2004-09-07	Kekki	Initial version

Table of Contents

Chapter 1. Introduction

Toolchain is a collection of tools used to develop software for a certain hardware target. Toolchains are based on particular versions of compiler, libraries, special headers and other tools. A cross-toolchain is a toolchain for compiling binaries for different CPU architecture than the host CPU.

Scratchbox is a cross-compilation toolkit for embedded Linux application development. It is designed for compiling software for different target CPU architectures. Scratchbox allows creating several target environments. Each target is a separate environment that has a selected toolchain, target CPU and an own file system.

Building toolchains is not always trivial. Scratchbox uses scripts for building predefined toolchains. After building a toolchain it should be tested to know that it works properly. Testing toolchains is harder than building toolchains.

Chapter 2. Scratchbox toolchains

2.1. General

Scratchbox toolchains provide the cross compilation tools for compiling binaries for the target environment. Each toolchain is built for a certain CPU target and they are based on certain gcc and C-library sources.

Scratchbox toolchains can be used both inside and outside Scratchbox. In Scratchbox each target uses a certain toolchain with a specific target CPU. Scratchbox uses wrappers to make the toolchains appear as if they were native toolchains. Outside Scratchbox the toolchains are used as any normal cross compilation toolchains.

All the installed toolchains can be found in the `/scratchbox/compilers` directory.

2.2. Toolchain design

The toolchains use either glibc or uClibc C-libraries. The glibc toolchain is based on Debian gcc-3.3, binutils, libc6 and linux-kernel-headers packages. Also some Scratchbox patches are applied to the packages. The uClibc build uses Erik Andersen’s uClibc toolchain build script which uses uClibc’s own patches.

Scratchbox.org [1] offers prebuilt toolchains for x86 and ARM targets. Only these targets are currently supported because too much work to support all different configurations. With the Scratchbox toolchain sources you can build your own custom toolchains.

Scratchbox toolchain building scripts allow you to build easily predefined toolchains and give the possibility to build custom toolchains for different targets. Changing the toolchain binutils, compiler or C-library packages can be a more demanding task.

2.3. Wrappers

Scratchbox uses a *gcc wrapper* for wrapping most of the toolchain command binaries. In the `/scratchbox/compilers/bin/` directory you can see the linked binaries. The wrapper knows how to handle each command and depending on the command it might change some of the command parameters. Then it runs the actual command from the correct path that depends from the selected target. The gcc wrapper reads all the target specific information from the target configuration files that can be found in the `/scratchbox/users/username/targets/` directory.

Scratchbox has also an *ld wrapper* for linking the binaries properly. When compiling inside Scratchbox a *fake-native* ld is used. Outside Scratchbox a normally behaving ld is used. For example inside Scratchbox the dynamically linked binaries are linked against the libraries that are in standard library paths, not in the toolchain's library path.

The gcc wrapper uses ccache by default. The default cache directory is `/scratchbox/ccache/`. Ccache can be disabled by setting the environment variable SBOX_USE_CCACHE to "no". The cache directory can be changed with the CCACHE_DIR environment variable.

2.4. Toolchain installation

The following binary packages are available:

- scratchbox-toolchain-arm-gcc3.3-glibc2.3
- scratchbox-toolchain-arm-gcc3.2-uclibc20040229
- scratchbox-toolchain-i686-gcc3.3-glibc2.3
- scratchbox-toolchain-i386-gcc3.2-uclibc20040229

For each toolchain that you want to use you need to create a Scratchbox target. For installation and target creation instructions see *Installing Scratchbox* [2].

Chapter 3. Building custom toolchains

3.1. Toolchain build system

Scratchbox toolchain build scripts are available in the *sb-toolchains* source package that is available at the Scratchbox download page [1] for each Scratchbox version.

Scratchbox toolchains are built with the GAR system [3]. It is a mechanism for automating the compilation and installation of third-party source code. It appears in the form of a tree of directories containing Makefiles and other ancillary bookkeeping files (such as installation manifests and checksum lists).

The `gcc/glibc` directory contains the glibc toolchain scripts and `gcc/uclibc` the uClibc toolchain scripts. The `gcc/glibc/versions` file has all the information of the debian package versions. It tells which debian packages are used for compiling the toolchains. The `gcc/glibc/files` directory contains the Scratchbox patches. Only some of them are in use.

The sb-toolchains source package is used also to build *arch tools* and *device tools* which are target dependent. The build instructions in this document automatically build all of them; there is currently no documented way to disable that.

3.2. Environment

Toolchains have to be built with an already existing compiler. Scratchbox has the HOST target for this. HOST's host-gcc toolchain is for compiling binaries for the host. It's configured to make the compiled binaries use the Scratchbox's host libraries.

When building toolchains you need write privileges to the `/scratchbox/compilers` and `/scratchbox/device_tools` directories. The *scratchbox* source package contains the `scripts/permhack` script for changing the privileges, but you can also do it like this:

```
# chown -R username:groupname /scratchbox/compilers/
# chown -R username:groupname /scratchbox/device_tools/
```

Toolchain building needs the Debian devkit package to be installed; the build script uses the dpatch and dpkg commands.

3.3. Compiling toolchains

1. Make sure you have the scratchbox-devkit-debian package installed on your system.
2. Download the sb-toolchains source package (see Section 3.1). Extract it into a directory that is visible inside Scratchbox (such as `/scratchbox/users/username/home/username`).
3. Change the permissions of the Scratchbox installation directories (see Section 3.2).
4. Login to Scratchbox.
5. Modify the HOST target to use the Debian devkit:

   ```
   [sbox-HOST: ~] > sb-conf setup HOST --devkits debian --force
   ```
6. Install `/etc/passwd` and Debian-specific files on the HOST target:

   ```
   [sbox-HOST: ~] > sb-conf install HOST --etc --devkits
   ```
7. Reselect the HOST target so that the environment gets updated:

   ```
   [sbox-HOST: ~] > sb-conf select HOST
   ```
8. If you want to build the default toolchains, run the `build` script in the `sb-toolchains` directory:

   ```
   [sbox-HOST: ~] > cd sb-toolchains
   [sbox-HOST: ~/sb-toolchains] > ./build --build
   ```

   If you want to build custom toolchains, use the `Makefile`. You need to pass the following variables to **make**:

   `TC_SOURCE`

   source directory; for example “gcc/glibc”

   `TC_NAME`

   compiler name; for example “i686-gcc-3.3.4-glibc-2.3.2”

   `TC_ARCH`

   architecture; for example “i386”

   `TC_SUBARCH`

   sub-architecture; for example “i686”

   `TC_CPU`

   CPU type (optional)

   Examples:

   ```
   [sbox-HOST: ~/sb-toolchains] > make TC_SOURCE=gcc/glibc \
   TC_NAME=armv3l-gcc-3.3.4-glibc-2.3.2 TC_ARCH=arm TC_SUBARCH=armv3l TC_CPU=arm7
   ```

   ```
   [sbox-HOST: ~/sb-toolchains] > cat my-compiler.config
   TC_SOURCE  = gcc/glibc
   ```

```
TC_NAME    = arm-gcc-3.3.4-glibc-2.3.2
TC_ARCH    = arm
TC_SUBARCH = arm
[sbox-HOST: ~/sb-toolchains] > make TOOLCHAIN=my-compiler.config
```

3.4. Changing source packages

Before changing the source package versions, test building the toolchains with the default configuration to see that the Scratchbox environment works properly.

The glibc and uClibc sources are fetched in different ways. Glibc toolchain build uses a version file for managing the package versions. Each package is downloaded separately from the download site that is defined by `gar.conf.mk`. (You can override the MASTER_SITES variable in the `.gar-local.conf` file.)

The uClibc toolchain source version is defined in the `gcc/uclibs/Makefile`. The same file also defines the sources download site.

The glibc toolchain packages can be updated with the following process:

1. Download the new source packages to the `gcc/glibc/files` directory. For example in Debian you can use this command:

   ```
   $ cd /scratchbox/users/username/home/username/sb-toolchain/gcc/glibc/files
   $ apt-get source gcc-3.3
   ```

2. Update the new package names in `versions` file.

   ```
   $ /scratchbox/login
   [sbox-HOST: ~] > cd sb-toolchains/gcc/glibc
   [sbox-HOST: ~/sb-toolchains/gcc/glibc] > vi versions
   ```

3. Generate MD5 checksums of the new packages.

   ```
   [sbox-HOST: ~/sb-toolchains/gcc/glibc] > make makesums
   ```

4. Test that all used patches get written properly.
5. Compile and test the new toolchain.

3.5. Building binary packages

The toolchains can be packaged outside or inside Scratchbox. You can create tarballs and Debian packages inside Scratchbox, but if you want to create RPMs you need the **rpmbuild** command which is not included in Scratchbox. If you are using a Debian GNU/Linux system, you can get rpmbuild from the “rpm” Debian package.

You can create all three package types of the default toolchains using the `build` script:

```
$ cd /scratchbox/users/username/home/username/sb-toolchains/
$ ./build --package
```

Custom toolchains are packaged using the **packages**, **tarball**, **deb** and/or **rpm** targets of `Makefile`. This time the following options are needed:

`TC_NAME`

compiler name (see Section 3.3)

`TC_PACKAGE`

name of the binary package; for example “scratchbox-toolchain-i686-gcc3.3-glibc2.3”

`TC_REPLACE`

name of a package this new package replaces (optional)

Examples:

```
[sbox-HOST: ~/sb-toolchains] > make tarball deb \
TC_NAME=armv3l-gcc-3.3.4-glibc-2.3.2 \
TC_PACKAGE=scratchbox-toolchain-armv3l-gcc3.3-glibc2.3
```

```
$ cat my-compiler.config
TC_SOURCE  = gcc/glibc
TC_NAME    = arm-gcc-3.3.4-glibc-2.3.2
TC_ARCH    = arm
TC_SUBARCH = arm
TC_PACKAGE = scratchbox-toolchain-arm-gcc3.3-glibc2.3
TC_REPLACE = scratchbox-toolchain-arm-glibc
$ make packages TOOLCHAIN=my-compiler.config
```

Chapter 4. Testing toolchains

4.1. Initial tests

All created compilers should be at least tested to compile, link and run a simple test program. The test program should be tested to link both statically and dynamically. These tests should be done with both a C and a C++ program.

The `sb-toolchains/test_tools/test_scripts/init_tests.sh` script can be used for these initial tests.

4.2. Regression tests

These tests are a collection of test for the C and C++ frontends of gcc. The tests can be found in gcc's directory gcc-VERSION/gcc/testsuite.

The result of running the testsuite are various *.sum and *.log files in the testsuite subdirectories. The *.log files contain a detailed log of the compiler invocations and the corresponding results, the *.sum files summarise the results. These summaries contain status codes for all tests:

- PASS: the test passed as expected
- XPASS: the test unexpectedly passed
- FAIL: the test unexpectedly failed
- XFAIL: the test failed as expected
- UNSUPPORTED: the test is not supported on this platform
- ERROR: the testsuite detected an error
- WARNING: the testsuite detected a possible problem

The Scratchbox toolchains should be tested with these tests. These tests can be run with `sb-toolchains/test_tools/test_scripts/reg_tests.sh` or manually as described in *GCC Testing* [5].

Runnig the gcc testsuite requires DejaGnu and Expect host tools. These can be compiled running **make install** in the `sb-toolchains/test_tools/dejagnu/` directory. This will also build Expect. Remember that these are host tools so they have to be compiled with the HOST target. You also need to change the permissions of the `/scratchbox/tools` directory.

```
# chown -R username:groupname /scratchbox/tools/
```

References

[1] *Scratchbox download page (http://scratchbox.org/download/)* .

[2] *Installing Scratchbox (http://scratchbox.org/documentation/docbook/installdoc.html)* , Valtteri Rahkonen.

[3] *GAR Architecture (http://www.lnx-bbc.org/garchitecture.html)* .

[4] *Scratchbox development team contacts (http://scratchbox.org/contact/)* .

[5] *GCC Testing (http://gcc.gnu.org/install/test.html)* .