Scratchbox release test plan

Katja Kääriä

katja.kaaria@movial.fi

Scratchbox release test plan

by Katja Kääriä

Revision history

Version:	Author:	Description:
2004-02-19	Kääriä	Draft version
2004-04-08	Kääriä	Initial version
2004-04-16	Kääriä	Final proposal
2004-06-10	Savola	Updated copyright information
2004-07-21	Kataja	Changed document format to DocBook

Table of Contents

Chapter 1. Introduction

This document is the test plan for the Scratchbox project. It is written to describe methods taken for ensuring the quality of the open source software before each software release.

1.1. Scratchbox cross-compilation toolkit

Scratchbox is a cross-compilation toolkit designed to make embedded Linux application development easier. It also provides a full set of tools to integrate and cross-compile an entire Linux distribution. Especially Debian is supported, but is been used also on other Linux distributions. Scratchbox provides a sandboxed build environment which offers a controlled set of tools and utilities needed for cross-compilation. It is an environment in which it is easy to assure that the intended versions of libraries, headers and other such files are used during the build.

1.2. Scope of the document

The test plan covers testing the software implemented or modified in the Scratchbox project before a release can be established. The reader is also useful to review documents published on project pages [1] and should also achieve the *Scratchbox install manual* [2] document at hand since design and functionality are described there and not duplicated in this document.

1.3. Structure of the document

The document comprises firstly the checklist to confirm the release is containing up-to-dated products for aiming products consistency. And secondly, the actual testing procedure is identified purposes to describe the testing methodology used in the project without going into details of the tests. The details and test instructions are covered in *Scratchbox install manual* [2] document.

Chapter 2. Generic test information

2.1. Defects

The delivery package shall be tested with different Linux distributions with latest, suitable third party packages. When differences or fault behavior on implementation (i.e. bugs) are found, one of the following alternatives should be executed.

- If the bug can be fixed inside the project scope it should be added to the *buglist* and later fixed.
- If the problem seems to be in 3rd party software it should be
  1. added to the *buglist*,
  2. described in the test report and
  3. reported to the maintainers of the broken software if the bug is not a known problem of the software.

Also minor defects that are not considered test failures should be reported as bugs, but with lower priority. Bugs are classified to the following priority classes based on the severity:

P1 Crash

: Application stops, crashes, or ceases to function.

P2 Inoperable

: Application is severely restrict, but can continue to run, or application has a security related issues.

P3 Impaired

: Specific functionality is missing or unexpected behavior occurs.

P4 Cosmetic

: Errors that have a minor impact on use of the software, e.g. usability enhancements.

2.2. Test report

Test results shall be reported in a test report. The report includes:

- **Test implementation status:** A summary of which tests have been written, test entity, status, original schedule, and latest estimate.
- **Test execution status:** A summary of which tests have been run, test entity, status, result, and date.
- **Buglist status:** Count of new, open, ignored, and closed bugs; statistics graph of open, closed, and closed+ignored bugs, and top-ten priorized bugs.

- **Effort put to testing** in man-hours.

In addition, the test report shall include the output of the tests, the buglist, and the implementation. The test report shall be delivered before each release report.

2.3. Item pass and test suspension criteria

The release test is succeeded if a release works correctly on Debian testing, Debian stable and the latest Fedora distributions, with PC and ARM glibc c-library choices. This is valid until further notice of a development progress. After succeeded testing on Debian stable environment, tests can be run also on other environments.

When the product auditing discovers a failure preventing testing, e.g. packet uncompliances, or the dynamic testing is to be prevented, because the application ceases to function, testing will be suspended and to be continued when failures are fixed.

Chapter 3. Software to be tested

3.1. 3rd party packages

The third party, open-source software packages of latest version are to be checked, compared and confirmed before a release testing. Each package source compared to, is found in Release notes and in CVS directories. The current versions of available packages, with the naming convention, can be found in Makefiles under the following named CVS/scratchbox/ directories:

- **/debian_tools/**
- **/doc_tools/**
- **/tools/**

The version of each 3rd party package used in Scratchbox, should be inspected on Debian web pages [4] and compared to latest corresponding Fedora package. The updated directory for Fedora distribution packages can be found, for example, in address [5]. The older package version of these two named sources should be picked to be a valid package for a Scratchbox release. Notice, often package deprecations, package splits or package removes are involved. Also, these should be inspected and reported.

3.2. Test platform

All new releases developed in the Scratchbox project shall be tested with the following **supported Linux distributions**:

- Debian stable
- Debian testing
- The latest Fedora

For the developing purposes some releases are tested infrequently also on **other Linux distributions to be tested with**, listed below.

- Debian unstable
- Red Hat Enterprise Linux 3 WS
- Fedora Core 1
- Linox 4.0
- Mandrake 9.2
- SuSE 9.0
- Slackware 9.1

- Gentoo 1.4

Needless to say that different Linux distributions have differences in function. Therefore the Scratchbox project documentation covers the *Scratchbox Distribution Compatibility testing* [3] document describing testing procedures for each upon named Linux distributions.

3.3. Test schedule

Tests shall be run before every release after the test report is generated and saved into CVS. All tests are rerun in case parts of tests fails.

All tests are meant to be run from inside the SDK. The tests which are run in several platforms should first pass in x86 with Debian stable.

Chapter 4. Static testing

Auditing of Scratchbox products (software packages, documentation) is particularly important for insuring the compliance and sanity of the product delivery. Defects or uncompliances should be detected in early phases. Also improvement suggestions should be possible to report, since Scratchbox is in continuous developing progress.

When making architectural changes in developing that deviates from the documentation, the developer should be able to make a bug of it into the Bugzilla under the documentation title.

Before a new release is to be tested and then established, the tasks on the following checklist should be signed, and deviations or absences should be reported promptly and sign as defects. Before evaluating, the personal Scratchbox CVS repositories should be updated when necessary.

- **Check the documentation**. Inspect that the documentation is contains up-to-dated names, numbers, tables or contexts etc. Whenever an architectural change or a new implementation occurs the project documents should be updated accordingly. Check the contents responds to the configuration.
  - *Check and update testplan:* when necessary update tests or package information e.g. and create a new version testplan.
  - *Check the webpage documentation:* generic view of documentation and particularly inspecting of the user documentation, comparing release notes and consulting for developers of new features.
  - *Check the CVS repository:* evaluate files in doc directories under CVS repository.
- **Check the CVS repository modules.**
  - *Acquire a list of latest software package versions:* (when necessary implement checks of suitable websites etc.)
  - *Compare packages on CVS and on web release notes:* the contents should respond to current configuration.
  - *Check the software packages versions:* make sure the older version of two package resources, Debian testing release and the latest Fedora release, is available.
  - *Report changes:* involve information in report about package updates, splits, removes or other relevant changes.
- **Evaluate**
  - *Evaluate bug fixe and open bugs:* compare bugs on previous version and evaluate risk spots.

Chapter 5. Dynamic testing

If auditing the products doesn't discover the suspension criteria release package testing is to be run. Tests should be run firstly on Debian stable distribution, if suspension criteria occurs here, no following tests are need to run on other platforms.

5.1. Acquiring suitable packages

Acquire and install the latest Debian and Fedora platforms and then the Scratchbox delivery. Since a release version of 0.9.8, the product delivery is divided into the following named package deliveries:

- scratchbox-core
- scratchbox-libs
- scratchbox-doctools
- scratchbox-debian-devkit
- scratchbox-devicetools

And support currently the following toolchains:

- scratchbox-toolchain-i686-gcc-ds5
- scratchbox-toolchain-i386-gcc-uclibc
- scratchbox-toolchain-arm-gcc-ds5
- scratchbox-toolchain-arm-gcc-uclibc
- scratchbox-toolchain-powerpc-gcc-ds5

The release should be tested with all toolchains though emphasizing firstly glibc c-library choices with PC and ARM.

5.2. Testing guidelines

The compiling should work in three different environments:

- i386, PC development environment
- QEMU, target environment emulator
- real hardware, in this case it may be IPAQ with ARM (when suitable release version is available, it could be tested also in other hardware)

The guideline is that the ARM should be compiled with *sbrsh* and *qemu*. And the PowerPC should compile with *qemu*. Refer to [6] for detailed instructions for using *qemu* CPU emulator. The user is able to select either *sbrsh* or *qemu* by setting environment variable **SBOX_CPUTRANSPARENCY_METHOD**.

Testing should be run on at least three different types of source software:

- 'Hello World' application delivered with Scratchbox
- GTK2 'Hello World' application delivered with Scratchbox
- Debian application package delivered with Scratchbox (See Chapter 4.2 in [2])

Detailed command line procedures for performing compilation and X application launching can be found e.g. in the [2].

5.3. Install, configure, compile, run, check

In this section generic testing procedures are described on supported Linux distribution platforms. Detailed instructions for performing tests addressed in here, can be found in the [2].

5.3.1. Installation

At first software is to be installed. The root privileges are needed for the next parts of Scratchbox installation.

1. Update the platform(latest Debian stable, Debian testing and Fedora).
2. Install the new Scratchbox version (Chapters 2, 5.2 [2]).
3. Start Scratchbox again and login into it.
4. Try to compile and run some simple software on PC and QEMU environment with different targets and toolchains(Chapter 2.4 [2]).
5. Check for compilers, targets and other relevant features for ensuring that everything works correctly.

5.3.2. CPU-transparency

To be able to cross-compile more complex software which uses 'configure' scripts (such as glib, gtk, etc.) the Scratchbox CPU Transparency feature has to be enabled. This is done after compiling some software for x86 using the Scratchbox.

Set-up the CPU transparency according to following steps that instructions are documented in [2]:

1. Configure Linux handheld for network settings(Chapter 3 [2]).
2. Set-up the NFS environment and start the NFS server (the method varies on different Linux platforms)(Chapter 3.1 [2]).
3. Configure Sbrsh feature on **.sbrsh** file(Chapter 3.2 and Appendix A. [2]).
4. Install Debian devkit and Debian rootstrap for GUI libraries to use (Chapters 4.1, 4.2 [2]).
5. Build and run simple and complex software with Sbrsh on different targets with different toolchains (Chapters 2.3, 4.3 [2]).
6. After launching programs check for transparency log file for detailed results.

Check that software compiles and runs correctly. If errors, check first the environment, e.g environment variables prior to reporting errors.

Uninstall the Scratchbox according to documented (Chapter 5.3 [2]) procedure and make a test report.

References

[1] *Scratchbox project web pages (http://www.scratchbox.org/)* .

[2] *Scratchbox install manual (http://www.scratchbox.org/documentation/docbook/installdoc.html)* .

[3] *Scratchbox Distribution Compatibility (http://www.scratchbox.org/documentation/docbook/portingdocument.html)* .

[4] *Debian packages (http://packages.debian.org/)* .

[5] *Fedora packages (ftp://ftp.funet.fi/pub/mirrors/ftp.redhat.com/pub/fedora/linux/core/updates/)* .

[6] *QEMU CPU emulator (http://fabrice.bellard.free.fr/qemu/)* .