

MOVIAL

Scratchbox Apophis r4 Release Test Plan

Jussi Hakala, Timo Savola

November 2, 2006

Contents

1 Scratchbox Apophis r4 release test plan	1
1.1 Architecture and platforms	1
2 Packages included in Scratchbox Apophis r4	1
2.1 Packages	1
2.1.1 Debian	1
2.1.2 Tarball	2
2.1.3 Source	2
2.2 Toolchains	2
2.3 Devkits	2
2.4 CPU Transparency methods (provided by devkits)	3
3 New features in Scratchbox Apophis r4	3
3.1 Tests for new features	4
3.2 Custom tasks in the testing of new features	6
3.2.1 Toolchain	6
3.2.2 QEMU	7
3.2.3 Debian devkit	8

Revision history

Version	Author	Description
2006-03-27	Jussi Hakala	First draft
2006-03-28	Jussi Hakala	Few definitions and corrections
2006-04-05	Jussi Hakala	New naming convention
2006-04-18	Jussi Hakala	Section describing included packages in the release
2006-04-19	Jussi Hakala	Refined structure of the document
2006-04-19	Jussi Hakala	Scratchbox Apophis r3
2006-04-26	Timo Savola	Toolchain terminology disambiguated, correct deb_list task description
2006-04-26	Jussi Hakala	Correct task descriptions
2006-05-05	Jussi Hakala	Own environment for sbxversion for printing correctly
2006-11-02	Timo Savola	Scratchbox Apophis r4

1 Scratchbox Apophis r4 release test plan

1.1 Architecture and platforms

Scratchbox Apophis r4 will be tested using *x86* architecture. The primary platform will be *Debian Sarge*, other platforms have limited support. All the tests described by the Scratchbox Release Test Suite and Scratchbox Release Test Plan will be executed in the primary platform manually. In other platforms, only a set of tests from Scratchbox Release Test suite will be executed using automated testing utilities.

List of supported platforms:

- Debian 3.1 (Sarge)
- Ubuntu 6.10 (Edgy Eft)

2 Packages included in Scratchbox Apophis r4

2.1 Packages

2.1.1 Debian

- `scratchbox-core_1.0.6_i386.deb`
- `scratchbox-libs_1.0.6_i386.deb`
- `scratchbox-devkit-cputransp_1.0.1_i386.deb`
- `scratchbox-devkit-debian_1.0.4_i386.deb`
- `scratchbox-devkit-doctools_1.0.5_i386.deb`
- `scratchbox-devkit-perl_1.0.4_i386.deb`
- `scratchbox-toolchain-host-gcc_1.0.6_i386.deb`
- `scratchbox-toolchain-arm-gcc3.4-uclibc0.9.28_1.0.4_i386.deb`
- `scratchbox-toolchain-arm-gcc4.1-uclibc20061004_1.0.4_i386.deb`
- `scratchbox-toolchain-arm-linux-2006q1-6_1.0.4_i386.deb`
- `scratchbox-toolchain-arm-linux-cs344-2.3_1.0.4_i386.deb`
- `scratchbox-toolchain-arm-linux-ct401-2.3_1.0.4_i386.deb`
- `scratchbox-toolchain-i686-linux-ct4.1.0-2.3.6tls_1.0.4_i386.deb`

2.1.2 Tarball

- `scratchbox-core-1.0.6-i386.tar.gz`
- `scratchbox-libs-1.0.6-i386.tar.gz`
- `scratchbox-devkit-cputransp-1.0.1-i386.tar.gz`
- `scratchbox-devkit-debian-1.0.4-i386.tar.gz`
- `scratchbox-devkit-doctools-1.0.5-i386.tar.gz`
- `scratchbox-devkit-perl-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-host-gcc-1.0.6-i386.tar.gz`
- `scratchbox-toolchain-arm-gcc3.4-uclibc0.9.28-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-arm-gcc4.1-uclibc20061004-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-arm-linux-2006q1-6-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-arm-linux-cs344-2.3-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-arm-linux-ct401-2.3-1.0.4-i386.tar.gz`
- `scratchbox-toolchain-i686-linux-ct4.1.0-2.3.6tls-1.0.4-i386.tar.gz`

2.1.3 Source

Source packages can be obtained from the repository at scratchbox.org.

2.2 Toolchains

- `arm-gcc3.4-uclibc0.9.28`
- `arm-gcc4.1-uclibc20061004`
- `arm-linux-2006q1-6`
- `arm-linux-cs344-2.3`
- `arm-linux-ct401-2.3`
- `i686-linux-ct4.1.0-2.3.6tls`

2.3 Devkits

- `cputransp`
- `debian`
- `debian-sarge`
- `doctools`
- `perl`

2.4 CPU Transparency methods (provided by devkits)

- `qemu-arm-0.7.0-sb2`
- `qemu-arm-0.8.0-m2`
- `qemu-arm-0.8.0-sb2`
- `qemu-arm-0.8.1-sb2`
- `qemu-armeb-0.8.1-sb2`
- `qemu-i386-0.7.0-sb2`
- `qemu-i386-0.8.1-sb2`
- `qemu-mips-0.8.1-sb2`
- `qemu-mipsel-0.8.1-sb2`
- `qemu-ppc-0.7.0-sb2`
- `qemu-ppc-0.8.0-m2`
- `qemu-ppc-0.8.1-sb2`
- `qemu-sparc-0.7.0-sb2`
- `qemu-sparc-0.8.1-sb2`
- `sbrsh`

3 New features in Scratchbox Apophis r4

In addition to the tests in the Scratchbox Release Test Suite, new features in the Scratchbox Apophis r4 will be tested as follows:

- **Enhanced legacy toolchain support.**
Execute test 3.1.1 from Scratchbox Release Test Plan using the `cs2005q3.2-glibc-arm` and `cs2005q3.2-glibc-i386` legacy toolchains.
- **Merged QEMU fixes from Scratchbox 0.9.8.**
Execute test 3.1.2 from Scratchbox Release Test Plan.
- **Debian devkit with armel architecture support.**
Execute test 3.1.3 from Scratchbox Release Test Plan.
- **Debian Sarge backward-compatibility devkit.**
Execute test 3.1.4 from Scratchbox Release Test Plan.
- Test 3.1.5 from Scratchbox Release Test Plan tests the new features in general. Execute it using the `cs2005q3.2-glibc-arm` and `cs2005q3.2-glibc-i386` legacy toolchains.

3.1 Tests for new features

Test 3.1.1 *Setup a target using a legacy toolchain*

Test steps:

1. Install Scratchbox [task 3.1.1 from Scratchbox Release Test Suite]
2. Install a legacy toolchain [task 3.2.1]
3. Create a new user [task 3.2.1 from Scratchbox Release Test Suite]
4. Login to Scratchbox with newly created user [task 3.2.2 from Scratchbox Release Test Suite]
5. Create and select a new target using the legacy toolchain [task 3.2.3]
6. Install files on the new target [task 3.2.4]
7. Logout [task 3.2.3 from Scratchbox Release Test Suite]

Test 3.1.2 *Test the new QEMU*

Test steps:

1. Install Scratchbox [task 3.1.1 from Scratchbox Release Test Suite]
2. Install the `cs2005q3.2-glibc-arm` toolchain [task 3.2.1]
3. Create a new user [task 3.2.1 from Scratchbox Release Test Suite]
4. Login to Scratchbox with newly created user [task 3.2.2 from Scratchbox Release Test Suite]
5. Create and select a new target [task 3.2.5]
6. Install files on the new target [task 3.2.4]
7. Build netcat [task 3.2.6]
8. Execute the netcat binary [task 3.5.1 from Scratchbox Release Test Suite]
9. Logout [task 3.2.3 from Scratchbox Release Test Suite]

Test 3.1.3 *Test the new dpkg*

Test steps:

1. Install Scratchbox [task 3.1.1 from Scratchbox Release Test Suite]
2. Install the `cs2005q3.2-glibc-arm` toolchain [task 3.2.1]
3. Create a new user [task 3.2.1 from Scratchbox Release Test Suite]
4. Login to Scratchbox with newly created user [task 3.2.2 from Scratchbox Release Test Suite]

5. Create and select a new target [task 3.2.8]
6. Install files on the new target [task 3.2.4]
7. Check dpkg version [task 3.2.10]
8. Check dpkg architecture [task 3.2.11]
9. Logout [task 3.2.3 from Scratchbox Release Test Suite]

Test 3.1.4 *Test the legacy dpkg*

Test steps:

1. Install Scratchbox [task 3.1.1 from Scratchbox Release Test Suite]
2. Install the `cs2005q3.2-glibc-arm` toolchain [task 3.2.1]
3. Create a new user [task 3.2.1 from Scratchbox Release Test Suite]
4. Login to Scratchbox with newly created user [task 3.2.2 from Scratchbox Release Test Suite]
5. Create and select a new target [task 3.2.9]
6. Install files on the new target [task 3.2.4]
7. Check dpkg version [task 3.2.12]
8. Check dpkg architecture [task 3.2.13]
9. Logout [task 3.2.3 from Scratchbox Release Test Suite]

Test 3.1.5 *Build Debian Sarge base system using Crocodile, and the Debian Sarge versions of GTK+, Tetex, Texinfo and Pango*

Test steps:

1. Install Scratchbox [task 3.1.1 from Scratchbox Release Test Suite]
2. Install a legacy toolchain repackaged for Scratchbox Apophis [task 3.2.2]
3. Create a new user [task 3.2.1 from Scratchbox Release Test Suite]
4. Login to Scratchbox with newly created user [task 3.2.2 from Scratchbox Release Test Suite]
5. Create and select a new target [task 3.2.8 if using an ARM toolchain; task 3.2.3 if using a native toolchain]
6. Install files on the new target [task 3.2.4]
7. Compile essential packages for the Debian environment using Crocodile [task 3.4.2 from Scratchbox Release Test Suite]
8. Install all build-dependencies of the `texinfo` source package

9. Build the `texinfo` source package
10. Install all build-dependencies of the `tetex-base` source package
11. Build the `tetex-base` source package
12. Install all build-dependencies of the `tetex-bin` source package
13. Build the `tetex-bin` source package
14. Install all build-dependencies of the `pango1.0` source package
15. Build the `pango1.0` source package
16. Install all build-dependencies of the `gtk+2.0` source package
17. Build the `gtk+2.0` source package
18. Logout [task 3.2.3 from Scratchbox Release Test Suite]

3.2 Custom tasks in the testing of new features

3.2.1 Toolchain

Task 3.2.1 *Install a legacy toolchain package*

Task steps:

1. Download toolchain package from <http://scratchbox.org/download/files/sbox-releases/legacy/>
2. Extract it where your scratchbox directory was extracted

How to determine if the task was passed:

- You can see the toolchain directory under the `scratchbox/compilers` directory

Task 3.2.2 *Install a repackaged legacy toolchain*

Task steps:

1. Download toolchain package from <http://scratchbox.org/download/files/sbox-releases/stable/>
2. Extract it where your scratchbox directory was extracted

How to determine if the task was passed:

- You can see the toolchain directory under the `scratchbox/compilers` directory

Task 3.2.3 *Setup and select a target using a legacy toolchain*

Task steps:

1. Run `sb-menu`
2. Choose to setup a target
3. Create a new target
4. Select the legacy toolchain
5. Do not select any devkits
6. Do not select a CPU-transparency method
7. Choose not to extract a rootstrap
8. Choose to install files
9. Accept the default installation settings
10. Answer yes or acknowledge all prompts and exit from the menu

How to determine if the task was passed:

- After step 9, the user was asked to select the target before installing files

Task 3.2.4 *Install files on a target using a legacy toolchain*

Task steps:

1. Run `sb-menu`
2. Choose to install files
3. Select a target using a legacy toolchain
4. Accept the default installation settings
5. Exit from the menu

How to determine if the task was passed:

- The list of installable files included “`fakeroot 1.3`”
- Fakeroot was not selected for installation by default

3.2.2 QEMU

Task 3.2.5 *Setup and select a target using QEMU*

Task steps:

1. Run `sb-menu`

2. Choose to setup a target
3. Create a new target
4. Select the `cs2005q3.2-glibc-arm` toolchain
5. Select the `cputransp` devkit
6. Select the `qemu-arm-0.8.0-m2` CPU-transparency method
7. Choose not to extract a rootstrap
8. Choose to install files
9. Accept the default installation settings
10. Answer yes or acknowledge all prompts and exit from the menu

How to determine if the task was passed:

- `qemu-arm-0.8.0-m2` was listed as a CPU-transparency method
- The target was successfully created and selected

Task 3.2.6 *Compile netcat*

Task steps:

1. Download netcat source package from <http://netcat.sourceforge.net/>
2. Run `make linux` in the netcat source tree

How to determine if the task was passed:

- Make was successful

Task 3.2.7 *Test netcat*

Task steps:

1. Create `/etc/hosts` if it does not exist
2. Run `echo -e "GET / HTTP/1.0\n" | ./nc www.google.com 80`

How to determine if the task was passed:

- `nc` did not crash or return with an error

3.2.3 Debian devkit

Task 3.2.8 *Setup and select an armel-target using Debian devkit*

Task steps:

1. Run `sb-menu`
2. Choose to setup a target
3. Create a new target
4. Select the `cs2005q3.2-glibc-arm` toolchain
5. Select the `debian`, `perl` and `cputransp` devkit
6. Select the `qemu-arm-0.8.0-m2` CPU-transparency method
7. Choose not to extract a rootstrap
8. Choose to install files
9. Accept the default installation settings
10. Answer yes or acknowledge all prompts and exit from the menu

How to determine if the task was passed:

- The target was successfully created and selected

Task 3.2.9 *Setup and select a target using Debian Sarge devkit*

Task steps:

1. Run `sb-menu`
2. Choose to setup a target
3. Create a new target
4. Select a non-eabi/armel ARM toolchain
5. Select the `debian-sarge`, `perl` and `cputransp` devkit
6. Select the `qemu-arm-0.8.0-m2` CPU-transparency method
7. Choose not to extract a rootstrap
8. Choose to install files
9. Accept the default installation settings
10. Answer yes or acknowledge all prompts and exit from the menu

How to determine if the task was passed:

- The `debian devkit` was selected automatically when `debian-sarge` was selected
- The target was successfully created and selected

Task 3.2.10 *Check dpkg version*

Task steps:

1. Run “`dpkg --version`”

How to determine if the task was passed:

- `dpkg` prints `version 1.13.18` (among other things)

Task 3.2.11 *Check dpkg-architecture output*

Task steps:

1. Run “`dpkg-architecture -qDEB_HOST_ARCH`”
2. Run “`dpkg-architecture -qDEB_HOST_GNU_TYPE`”

How to determine if the task was passed:

- `dpkg-architecture` prints `armel` and `arm-linux-gnueabi`

Task 3.2.12 *Check legacy dpkg version*

Task steps:

1. Run “`dpkg --version`”

How to determine if the task was passed:

- `dpkg` prints “`version 1.10.26`” (among other things)

Task 3.2.13 *Check legacy dpkg-architecture output*

Task steps:

1. Run “`dpkg-architecture -qDEB_HOST_ARCH`”
2. Run “`dpkg-architecture -qDEB_HOST_GNU_TYPE`”

How to determine if the task was passed:

- `dpkg-architecture` prints `arm` and `arm-linux`