pbuilder
Debian Conference 2004

Junichi Uekawa

May 2004

1 Introduction

`pbuilder`[1] is a tool that is used for Building Debian packages in a clean environment inside chroot[1]. In this paper, the background of pbuilder, usage of pbuilder, and the evolution of pbuilder is discussed.

2 Background

2.1 What does pbuilder do?

`pbuilder`[1] is an application to manage minimal Debian installed image inside chroot, with ability to build Debian packages inside the chroot with minimal required packages installed.

This automates the process of creating a minimal chroot image as specified by the Build-Dependends field in the package control file. This allows to verify what is required to build the package in addition to the build-essential packages.

2.2 Debian Project

Debian Project is a project for creating a complete Operating System from free software. Debian has a guideline for 'freeness', called DFSG, which documents the criteria for being 'free' for Debian. DFSG section 4 states that Debian consists of software from which derived works can be created.

From pbuilder viewpoint, it is special in following points:

- Source code is provided in uniform format
- Method to build is uniform (debian/rules)
- Build-time dependency is documented (Build-Depends)

[1]chroot is a command which executes an application using a directory as a new '/', effectively allowing running different version of Debian live.

2.3 Build daemons and FTBFS

Debian Project has 'ports' to several architectures. Most developers are using i386 architecture to build packages and upload, but these packages are built from source on other architectures by 'build daemons'[2], and uploaded to the archive.

When a package does not build from source on an architecture, and it has been built before, it is a regression, and it is classified as a serious bug. It is called a 'FTBFS (Fails-to-build-from-source)' bug, and Debian Developers are usually requested to fix the bug in order to get the package in a releasable state.

2.4 Free Software Should Be Buildable

Even without taking into consideration whether a package is buildable on many esoteric architectures, it is important that free software be buildable from source on i386. Debian has Debian Free Software Guidelines which states that all software in Debian should be modifiable. However, in practice, to ensure that a package can be modified freely, the source-code needs to be rebuildable. Otherwise, a user may modify the source-code, but will not be able to create an object code that is similar to the binary package the user obtained from Debian.

`pbuilder` is a tool that allows to ensure that packages do build, on a local machine of Debian developer.

3 Demonstrating Usage of pbuilder

3.1 pbuilder create – creating pbuilder base.tgz image

`pbuilder`'s create command creates a chroot image. A command-line such as the following will usually be used.

```
# pbuilder create --distribution sid --mirror
    http://http.us.debian.org/debian --basetgz base.tgz
```

This will create a file `base.tgz` which is a tarball containing the 'sid' chroot image[3]. The chroot image will have minimal packages which are considered 'base packages', and 'build-essential'[4].

3.2 pbuilder update – updating pbuilder base.tgz image

`pbuilder`'s update command updates the chroot image with running `apt-get update` inside chroot. It is usually required to run pbuilder update about once

[2]'build daemons' consist of servers of different architectures taking part of the buildd/wanna-build infrastructure.

[3]A 'chroot image' is a tarball of directory hierarchy that can be used with chroot command. The directory hierarchy is similar to what is contained from '/' directory, relative to the directory which chroot is constructed from.

[4]There is a package build-essential. Debian packages do not have to explicitly build-depend on packages which are included in the build-essential.

a day to reflect changes to the Debian archive, and updates to Packages file[5].

The command-line will look like the following:

```
# pbuilder update --basetgz base.tgz
```

3.3 pbuilder build – building package inside chroot

Using the created chroot, it's now possible to run `pbuilder` to build a package. `pbuilder` will take a source package .dsc file and build the package from there.

The command line will look like the following:

```
# pbuilder build --basetgz base.tgz package_0.00-1.dsc
```

3.4 pdebuild – a short cut to building package

If '`pbuilder` build' command-line is not convenient, it's possible to run pdebuild, which tries to mimic the behaviour of debuild command[6]. It requires sudo to be configured.

pdebuild is ran from within the Debian source directory. The source directory should have a `./debian/` directory. pdebuild parses the files under `./debian/` and invoke pbuilder accordingly, after creating a source archive[7]

The command-line will look like:

```
$ pdebuild -- --basetgz /path/to/base.tgz
```

3.5 pbuilder login – log inside chroot to test

To debug problems within `pbuilder` environment, you can log-in to the environment temporarily.

```
# pbuilder login --basetgz /path/to/base.tgz
```

Note that the environment will be cleaned when you exit. It is a scrap-and-build environment.

4 Impact of pbuilder to the Debian Society

4.1 Responsibility and ability delegated from buildd administrators to individual Debian Developers

Debian Developers used to experience lack of tools to check the correctness of build dependency. It was a right reserved almost exclusively to the buildd administrators.

[5]Debian releases a new version of 'unstable' distribution every day

[6]debuild command can be found in devscripts package, and is a convenient wrapper over dpkg-buildpackage

[7]`--use-pdebuild-internal` option takes a different approach, but that option is not quite full-featured.

`pbuilder` allows performing the build on a local system with a very simple predefined procedure. It is possible to quickly create a system for building packages modulo bugs. The power of source-building is delegated to individual developers.

4.2 Regression testing of builds

With `pbuilder` it is easier to create a reproducible environment to test builds. By having the ability to do such operation, a maintainer of a package can ensure that a package will build reliably on a known minimal environment.

This allows having a regression testing on Debian archive, to figure out what packages no longer build where it should have built once. Regression testing of package builds requires that packages have been built before in the environment. `pbuilder` is an attempt to provide such environment.

4.3 Ability to build on pure environment

Many developers use environment that is not pure-unstable. They may be using some back-ported packages in stable distributions, or they may have some packages installed from the experimental distribution. Packages should not be built against libraries from experimental distribution, since uploading the package to unstable will make that package uninstallable within unstable.

Developers can now use pbuilder to build packages targeted at specific distribution release. This has a side-effect of allowing easier stable-back-ports.

5 Other works

5.1 Using user-mode-linux

chroot shows its limitations when used with `pbuilder`. Largest of them being that chroot shares the process space, which makes it difficult to handle daemons inside the chroot.

User-mode-linux has another advantage in that a normal user can simulate root user privilege within the user-mode-linux environment.

user-mode-linux is a virtual machine, and thus needs some work to get network connected. There are methods such as tun/tap. Unlike pbuilder, pbuilder-uml will need some user tweaking to get network working, and installation working.

pbuilder on User-mode-linux is slower than raw pbuilder.

However, pbuilder-uml uses COW file-system, which makes the startup time of pbuilder faster. COW file-system is a system where modified blocks on the file-systems are written to a different file from the real file-system, so that the modifications are not written to the file-system. This is used instead of extracting a fresh chroot image from tar archive every time pbuilder is invoked, which results in a more prompt user-experience.

pbuilder-uml does have overheads on system calls due to user-mode-linux, but it may sometimes be faster than pbuilder in this respect.

5.2 Fakechroot

The methods of fakeroot[2] can be extended to implement chroot as normal user. An implementation of such is called fakechroot[3].

This method shares the advantage of user-mode-linux that normal users can run pbuilder. However, the problem with shared process space of chroot is not solved.

Also, fakechroot uses shared library mechanisms to override libc functions. It will not work on static binaries and binaries that call syscalls directly; and will probably need some tweaking on the Debian side to get working reliably. Hacks on important components such as 'ldd' are already done.

6 Future works on pbuilder

6.1 Tuning pbuilder run speed

pbuilder has a bottleneck in several points. tar extraction is one of the largest part. pbuilder-uml solves the problem using COW file-system.

There are snapshot functions in LVM, which can be used instead of trying to untar a clean chroot image every time.

There are several file-systems which implemented template-based virtual file system[4] on Linux. translucency[5], and mini_fo[6] sound promising. It should however be noted that using such file-system will introduce another variable into package building process.

6.2 Rebuilding from scratch for custom distribution

pbuilder can be used for rebuilding everything in Debian to create a customised distribution. Initial motivation behind creating pbuilder was the 'Debian on athlon'[7] project, which aimed at building a distribution compiled with optimization for Athlon CPU.

Currently, on a modern PC, building all of Debian takes about a week.

6.3 Making Debian build from source

There are many packages which don't build from source with pbuilder. There are some shortcomings with pbuilder that will need workarounds. One of the main problems is that pbuilder does not handle daemons well, so packages that require daemons to build will need some workaround.

Some parsing of Build-Depends is not strictly compliant with what the policy states; although it is close. The Build-Depends parser of `pbuilder` is stateful, and tries to parse from left-to-right.

6.4 Extending pbuilder without introducing new bugs

`pbuilder` has grown to be a large script that requires much testing. It was initially 20-line shell code, and it is now more. Testsuite is used to ensure pbuilder works correctly. The testsuite currently takes about 20 minutes on a PC with Athlon XP 1800+ CPU[8]

In pbuilder source the following will invoke the testsuite:

```
$ cd testsuite
$ ./run-test.sh
```

and it will create files under 'normal/*.log'. Currently the tests involve running 'pbuilder create', 'pbuilder build', and 'pbuilder execute' commands. A summary is created as 'run-test.log' that will have an entry like:

```
[OK]   create-sid
[OK]   build-sid-dsh
[OK]   pdebuild-sid-dsh
[OK]   pdebuild-internal-sid-dsh
[OK]   execute-sid
[OK]   create-sarge
[OK]   build-sarge-dsh
[OK]   pdebuild-sarge-dsh
[OK]   pdebuild-internal-sarge-dsh
[OK]   execute-sarge
```

Having a testsuite to check that there is no regression is quite useful. Having a mechanical method to check obvious errors reduces total time required for checking the changes.

References

[1] `http://www.netfort.gr.jp/~dancer/software/pbuilder.html`. *pbuilder*, 2004

[2] `http://packages.debian.org/fakeroot` *fakeroot package*

[3] `http://packages.debian.org/fakechroot` *fakechroot package*

[4] `http://vserver.13thfloor.at/TBVFS/index.php?page=Linux+Implementations` *template based virtual file system implementations*

[5] `http://packages.debian.org/translucency-source` *translucency-source package*

[6] `http://www.denx.de/e/index1.php` *Overlay Filesystem "mini_fo" for Embedded Systems Released*

[8]Most operation seems to be I/O bound, and CPU type may be irrelevant.

[7] `http://www.netfort.gr.jp/~dancer/software/athlon-debian.html.en`
Debian on Athlon Project