Building Debian Packages with darcs-buildpackage

John Goerzen

Building Debian Packages with darcs-buildpackage

by John Goerzen

Table of Contents

Chapter 1. Introduction

Welcome to darcs-buildpackage, a system that integrates the Debian[1] package build system with Darcs[2].

Purpose

darcs-buildpackage is designed to do this for you:

- Ability to track changes to both upstream and Debian sources through time
- Ability to retrieve any version of the sources from the archive, and build Debian binary and source packages from them.
- Generated packages can be built and used by people that have neither Darcs nor darcs-buildpackage installed.
- Provide an alternative to systems like dbs and dpatch to help manage patches that Debian applies to packages.
- Provide a way to ease collaboration between multiple Debian developers on a single package.
- Provide a way to use the merging (update and replay) features of Darcs to help merge Debian changes into new versions of upstream packages.
- Make it easy to try darcs-buildpackage and switch back to normal systems if you don't like it.

Features

darcs-buildpackage has these features:

- dbp-importdsc will import an entire Debian source package into your Darcs area, handling both upstream and Debian trees automatically. dbp-importdsc can be run repeatedly to import newer versions, maintaining a full change history as it goes along.
- dbp-importorig can import an upstream tar.gz or a directory into your archive with a single command.
- darcs-buildpackage can use an existing orig.tar.gz, or can build you one from the Darcs archive if necessary.
- darcs-buildpackage works for both Debian-native and normal (upstream) packages.
- darcs-buildpackage ensures that Darcs metadata (the _darcs directory) doesn't show up in tar.gz or diff.gz files.
- Tight integration with darcs_load_dirs for intelligent handling and versioning of file renames upstream.
- Autosensing of information wherever possible from Debian changelogs, dsc files, and the darcs environment.

1. http://www.debian.org/
2. http://www.abridgegame.org/darcs

- dbp-markdeb can be used to easily checkpoint development into Darcs.

Usage Overview

Here is a high-level overview of the Debian development process using darcs-buildpackage.

You'll start by creating your `~/.darcs-buildpackage` configuration file and making a directory to store your Darcs repositories, if you don't already have one.

Next, it's time to bring your packages into Darcs. If you have some existing Debian packages, you'll import as much history as you like by using dbp-importdsc. Otherwise, to start a new package in darcs-buildpackage, you'll use dbp-importorig to bring in the upstream sources. Then, you'll use **darcs get** to copy the upstream area to the Debian area.

As you work on your Debian sources, you'll use darcs-buildpackage in place of debuild or dpkg-buildpackage (darcs-buildpackage passes all its arguments to debuild for you). You can use standard Darcs commands, such as **darcs record** or **darcs changes**, to commit changes and view history. When you have a final version of a Debian package, you'll upload it and run dbp-markdeb to tag the version for future reference.

Chapter 2. Getting Started

Getting started with darcs-buildpackage is pretty simple. This chapter shows you how.

The darcs-buildpackage Repository Layout

darcs-buildpackage assumes that you have two "canonical" repositories for every Debian package. These repositories correspond to the upstream and the Debian versions of the package's source. The upstream repository tracks unmodified upstream versions. The Debian repository pulls in upstream patches, and also has the Debian-specific patches for your package. (Debian-native packages have no upstream version and thus no upstream repository.)

Normally, you would not work directly in the canonical repository storage area. Rather, you would use **darcs get** or dbp-get to make a copy of the sources, and **darcs push** to push changes back to your storage area when done.

The canonical repository need not be on your local machine. It could be any repository area that Darcs has access to.

Upstream Repository

The upstream repository tracks changes to the upstream program. You can use dbp-importorig or dbp-importdsc to import upstream sources from tarballs, or you can use Darcs to track a Darcs-based upstream with **darcs get** and **darcs pull**.

Each upstream version should be tagged with a tag nammed UPSTREAM_*`packagename_version`*. Note that the Debian revision doesn't appear in this tag. An example would be UPSTREAM_mypackage_1.0.5. If you use dbp-importorig or dbp-importdsc, this tag will be created for you automatically. If you manage your upstream repository manually, you should create the tag yourself using **darcs tag**

Debian Repository

The Debian repository tracks the Debian-specific package. The Debian package is represented as the upstream source plus a set of Debian patches. Upstream patches are normally pulled into the Debian repository before beginning on the Debian package.

Each Debian version carries a tag similar to the upstream version, but the Debian version tags begin with DEBIAN_ and contain a full Debian revision. An example is DEBIAN_mypackage_1.0.5-3.

The Configuration File

The first thing you must do is create your darcs-buildpackage configuration file. It is called `.darcs-buildpackage` and stored in your home directory. The main purpose of the configuration file is to specify the location and naming of your repositories.

There are two items required in the configuration file: upstreamrepo and debianrepo. The special token `%(package)s` represents the name of a specific package. Thus, a sample file could look like this:

```
upstreamrepo = /home/jgoerzen/debdarcs/%(package)s.upstream
debianrepo = /home/jgoerzen/debdarcs/%(package)s
```

More sophisticated configurations are possible. (FIXME: write about them)

Chapter 3. Importing Packages

You will need to import sources into your darcs-buildpackage archive on several different occasions:

- You have existing Debian packages that you would like to maintain with darcs-buildpackage. In this case, you'll want to import one or more full Debian source packages with **dbp-importdsc**.
- You want to package some previously-unpackaged software for Debian. In this case, you'll want to import an upstream tarball with **dbp-importorig**.
- You want to update your Debian package with a new version of upstream source. This case also calls for **dbp-importorig**.
- You want to import Debian sources from someone else (for instance, if someone NMU's a package you maintain) into your tree. **dbp-importdsc** can handle that.

There are two different programs that handle importing: **dbp-importdsc** and **dbp-importorig**. Both are covered in this chapter.

Importing Debian Source Packages

Importing a package is very easy. All you have to do is run **dbp-importdsc** with the name of a .dsc file to import:

```
$ dbp-importdsc ~/dpkg/rdiff-backup_0.12.3-1.dsc
FIXME: show output
```

In this example, I had never imported rdiff-backup before. **dbp-importdsc** therefore initialized the repository for me. It then added and committed the upstream version (the first commit message), committed the Debian diffs (second commit), and noted which version in the archive corresponds to 0.12.3 and 0.12.3-1 (third commit message). (FIXME: these are probably old)

I can run it again with a new version:

```
$ dbp-importdsc ~/dpkg/rdiff-backup_0.12.5-1.dsc
FIXME: show output
```

dbp-importdsc is smart enough to know not to import an upstream version twice. For instance, if I would now load 0.12.5-2, there would be only two commits: a patch-3 on the Debian tree and a patch-5 on the configs tree. (FIXME: these names are for arch)

dbp-importdsc has a few restrictions: you must always load packages in ascending order of package version. Please see dbp-importdsc(1) for more details.

Importing Upstream Sources

While **dbp-importdsc** can solve many problems, sometimes you need to import just upstream sources (the Debian orig.tar.gz file). For this task, **dbp-importorig** exists. You will often use it if you have already loaded all the Debian versions into your archive, upstream has released a new version, and you want to package that version.

Using **dbp-importorig** is simple and straightforward; please refer to dbp-importorig(1) for more details.

Chapter 4. Common Tasks

This chapter describes how to use darcs-buildpackage to carry out regular, every-day development activities.

Getting Sources

FIXME: write this. Please see dbp-get manpage for info about getting and mirrors!

Building and Working With Source

Before you start working on any source package with darcs-buildpackage, first check it out as described in the Section called *Fetching Source from History* to make sure it is in its proper location. Then, as you work, you will use **darcs record** to commit your changes to the Debian repository. (If you just want to build an old package, there's no need to do this.)

When you're ready to build a package, just run **darcs-buildpackage** to build it. The **darcs-buildpackage** command takes the same arguments as **debuild**, and passes them on to **debuild**, so give it your usual set. **darcs-buildpackage** will use your existing orig.tar.gz file for source generation, if it exists; otherwise, it will take care of automatically building it from the upstream sources if necessary.

FIXME: these examples are from arch and need to be updated for Darcs

Here's an example:

```
~/tree/debian$ cd +packages/rdiff-backup/rdiff-backup-0.12.5
~/tree/debian/+packages/rdiff-backup/rdiff-backup-0.12.5$ darcs-buildpackage \
        -rfakeroot -us -uc
...
Building .orig from archive.
 * tla buildcfg upstream/rdiff-backup/0.12.5
* from import revision: jgoerzen@complete.org--debian/rdiff-backup--head--1.0--base-0
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--head--1.0--patch-1
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--head--1.0--patch-2
* making pristine copy
* tree version set jgoerzen@complete.org--debian/rdiff-backup--head--1.0
 * tla inventory -s "rdiff-backup-0.12.5.orig" | tar -cSpf - -T- | gzip -9 > "rdiff-backup_
 *** Running build program
Running:  debuild ['-i\\+\\+pristine-trees|„*|\\{arch\\}|\\.arch-ids', '-rfakeroot', '-us',
...
```

In this case, I had not checked out the upstream source and did not have my orig.tar.gz file handy. Therefore, **darcs-buildpackage** checked out the upstream sources for me, generated the tar.gz file, and then called **debuild** to do the rest.

If you are building for Debian, you will want to keep your orig.tar.gz around so that future source uploads use the same MD5 sum in the .dsc file.

For more details, please see the manpage for darcs-buildpackage(1).

Handling New Upstream Versions

A common scenario for a Debian developer to deal with is that of a new upstream release. The Debian patches from the most recent Debian release must be merged into the upstream one.

There have been a few tools to do that: **uupdate** is one. However, now that you are using Darcs, you can use its built-in **pull** command to make this easier.

The first thing that you will do is to import the new upstream sources into your darcs-buildpackage archive. Please see the Section called *Importing Upstream Sources* in Chapter 3 for instructions.

Next, check out the latest Debian version if you don't already have it (see the Section called *Fetching Source from History*) and then cd into the Debian source directory.

[Note: I anticipate writing a program to automate the following steps.]

Merging In Upstream

Now, you are ready to merge in the new upstream. Run **darcs pull**, giving it the path to your canonical upstream repository.

FIXME: add a sect3 demonstrating

Examining the Merge

Now, you will want to examine the merge, especially if Darcs complained of any conflicts.

You'll also want to note the new version in `debian/changelog` and rename your directory based on the new version. I find it easy to run a command like **debchange -v 0.13.3-1**, then modify the changelog as appropriate. debchange will handle the rename for you.

Committing the Merge

Finally, you will want to commit the merge. If you want to just use a simple log message, a command like this will work:

```
$ darcs record -m "Merged in upstream 0.13.3"
FIXME: show output
```

The string supplied after the -m is the log message.

If you want to add a more detailed log, try this, just run **darcs record** and it will prompt you for one.

You can see what happened by using **darcs changes -s**:

```
$ darcs changes -s
FIXME: show output
```

Slick -- it shows exactly which upstream patches you used.

Finalizing New Debian Versions

When you have uploaded a Debian package to the archive, you should ask darcs-buildpackage to note this for you. That way, you can request this specific version later. Just run this:

```
$ dbp-markdeb
FIXME: show output.
```

The reason for this is that you might make several commits during the course of hacking on a given Debian version. This command lets you note the final version, and to run it again, you must update the changelog.

Technically speaking, this creates the DEBIAN_ tag.

Finding Historic Versions

FIXME: this is not yet updated for Darcs

To find the versions available in your darcs-buildpackage archive, first cd to its top level and then run:

```
$ ls configs/*/*
configs/debian/rdiff-backup:
0.12.3-1  0.12.5-1

configs/upstream/rdiff-backup:
0.12.3  0.12.5
```

This shows you that two Debian versions and two upstream versions of rdiff-backup are present.

Fetching Source from History

To fetch the source code from the darcs-buildpackage archive, first **cd** into your top-level working copy. Then, using the name of the config file (see the Section called *Finding Historic Versions*), use the **tla buildcfg** command:

```
$ tla buildcfg debian/rdiff-backup/0.12.5-1
* from import revision: jgoerzen@complete.org--debian/rdiff-backup--head--1.0--base-0
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--head--1.0--patch-1
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--debian--1.0--base-0
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--debian--1.0--patch-1
* patching for revision: jgoerzen@complete.org--debian/rdiff-backup--debian--1.0--patch-2
* making pristine copy
* tree version set jgoerzen@complete.org--debian/rdiff-backup--debian--1.0
```

This command will have created the directory `+packages/rdiff-backup/rdiff-backup-0.12.5` to contain the sources. Generating upstream sources works the same, and puts them into a directory ending in .orig.

If you don't know what version you want but just want the latest, you can simply use a command such as **tla buildcfg debian/rdiff-backup/latest**.

Appendix A. Command Reference

dbp-importdsc

<jgoerzen@complete.org>

John Goerzen

Name

`dbp-importdsc` — Import a Debian source package into darcs-buildpackage archive

Synopsis

dbp-importdsc *`dsc_name`*

Description

dbp-importdsc imports a Debian source package into a darcs-buildpackage archive, notes the package version in the configs, and commits the change. All information, including package name, version, Debian diffs, and upstream source, is automatically detected from the source package.

This program will automatically detect if the upstream sources have already been imported, and if so, will not attempt to re-import them again.

It is an error to run **dbp-importdsc** on a package whose Debian version already exists in the archive. It is also an error to run **dbp-importdsc** on a package when a newer version already exists in the archive. **dbp-importdsc** detects both conditions and will terminate without causing harm.

Options

dsc_name

Gives the path to the Debian .dsc file from the source package.

Bugs

- Neither GPG signatures nor MD5 sums are verified when unpacking Debian source packages.

- This program does not deal well with Debian native packages (ones that lack a Debian diff.gz). It will probably work, but building the package later might result in a non-native package.

Files

The following specific files are used by this application:

`~/.darcs-buildpackage`

Specifies the location of the canonical Debian and upstream repositories.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

darcs-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

darcs-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/darcs-buildpackage.

darcs-buildpackage may also be downloaded using Darcs by running **darcs get --partial http://darcs.complete.org/darcs-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the darcs-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/darcs-buildpackage`.

darcs-buildpackage applications have references in that manual and the following manpages: dbp-importdsc(1), dbp-importorig(1), dbp-markdeb(1), darcs-buildpackage(1).

Other related references include: darcs(1), dpkg-source(1).

dbp-importorig

`<jgoerzen@complete.org>`

John Goerzen

Name

`dbp-importorig` — Import an upstream source into a darcs-buildpackage archive

Synopsis

dbp-importorig *`filename_or_dir package version`*

Description

dbp-importorig imports the tar.gz file given by filename (or a directory, if filename is one) into the darcs-buildpackage archive. Since upstream tarballs do not contain the package and version information in a unified format, you must specify that on the command line as well.

It is an error to run **dbp-importorig** on a package when a newer version already exists in the archive. **dbp-importorig** detects this condition and will terminate without causing harm.

Options

filename_or_dir

Gives the path to a tar.gz file to import.

Alternatively, gives the path to a directory to import without unpacking anything.

package

Gives the Debian source package name corresponding to this upstream package.

version

Gives the package version corresponding to this upstream package. This should not contain the Debian part (-1, -2, etc).

Files

The following specific files are used by this application:

`~/.darcs-buildpackage`

Specifies the location of the canonical Debian and upstream repositories.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

darcs-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

darcs-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/darcs-buildpackage.

darcs-buildpackage may also be downloaded using Darcs by running **darcs get --partial http://darcs.complete.org/darcs-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the darcs-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/darcs-buildpackage`.

darcs-buildpackage applications have references in that manual and the following manpages: dbp-importdsc(1), dbp-importorig(1), dbp-markdeb(1), darcs-buildpackage(1).

Other related references include: darcs(1), dpkg-source(1), tar(1).

dbp-markdeb

<jgoerzen@complete.org>

John Goerzen

Name

`dbp-markdeb` — Mark the Debian version of the working directory in archive

Synopsis

dbp-markdeb

Description

dbp-markdeb is used to tell darcs-buildpackage that the version of a package represented in the current working directory is "official". The progam will find the Debian version from `debian/changelog` and record it as appropriate. No input is required and no arguments are required. **dbp-markdeb** will abort if it is run more than once for a single Debian version.

Files

The following specific files are used by this application:

`~/.darcs-buildpackage`

Specifies the location of the canonical Debian and upstream repositories.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

darcs-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

darcs-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/darcs-buildpackage.

darcs-buildpackage may also be downloaded using Darcs by running **darcs get --partial http://darcs.complete.org/darcs-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the darcs-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/darcs-buildpackage`.

darcs-buildpackage applications have references in that manual and the following manpages: dbp-importdsc(1), dbp-importorig(1), dbp-markdeb(1), darcs-buildpackage(1).

dbp-get

<jgoerzen@complete.org>

John Goerzen

Name

`dbp-get` — Check out a Debian package from a Darcs repository

Synopsis

dbp-get [*package*...]

Description

The dbp-get command is used to check out a copy of the Debian source for a package from Darcs. Additionally, it can automatically download a copy from any of several mirrors if the Darcs repository doesn't already exist locally.

Options

The one option specifies the name of the package to get.

Configuration

Configuration happens in `~/.darcs-buildpackage` like with all the other commands. The two new options are `debianmirrors` and `upstreammirrors`. They are whitespace-separated lists of mirrors to check if the given package isn't in the local area. Here's an example:

```
repobase = /home/jgoerzen/debdarcs
upstreamrepo = %(repobase)s/%(package)s.upstream
debianrepo = %(repobase)s/%(package)s

debianmirrors = http://darcs.complete.org/debian/%(package)s
        http://somewhereelse.example.com/debian/%(package)s
upstreammirrors = http://darcs.complete.org/debian/%(package)s.upstream
        http://foo.example.com/upstream/%(package)s
```

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the darcs-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/darcs-buildpackage`.

darcs-buildpackage applications have references in that manual and the following manpages: dbp-importdsc(1), dbp-importorig(1), dbp-markdeb(1), darcs-buildpackage(1).

Other related references include: darcs(1), debuild(1), dpkg-buildpackage(1).

darcs-buildpackage

`<jgoerzen@complete.org>`

John Goerzen

Name

`darcs-buildpackage` — Build Debian packages from a Darcs archive

Synopsis

darcs-buildpackage [*`args for debuild`*...]

Description

The **darcs-buildpackage** command is used to build Debian source and .deb packages from a checked-out version of a Debian source tree. Please refer to the darcs-buildpackage manual for instructions on how to check out such a tree.

darcs-buildpackage will, in order:

- Verify that it is being executed from the proper location.
- Check to see if a proper orig.tar.gz exists. If not, it will check out the proper upstream directory from the darcs-buildpackage repository (if necessary) and build the orig.tar.gz. The orig.tar.gz will not contain Darcs meta-data.

  If the upstream repository is not available locally, it will consult mirrors in the same manner as dbp-get(1).
- Call debuild(1) (or the application specified in DBP_BUILDER) with arguments instructing it to ignore Darcs meta-data in the diff.gz, passing along all arguments given to **darcs-buildpackage**.

Options

All options passed to **darcs-buildpackage** are sent directly to debuild(1).

ENVIRONMENT VARIABLES

If the DBP_BUILDER environment variable is set, it will be taken as the name of a command to use to build the Debian packages. It will be passed no arguments save those passed to darcs-buildpackage itself.

If DBP_BUILDER is not set, the default is to use debuild -i_darcs -I_darcs, which will build a package and ensure that that _darcs directory is not included in the generated Debian source package. If you set DBP_BUILDER, you should probably ensure that similar exclusions are passed to your preferred building tool.

Example

```
~$ cd ~/tree/debian
FIXME: show example
```

Files

The following specific files are used by this application:

`~/.darcs-buildpackage`

Specifies the location of the canonical Debian and upstream repositories.

Environment

This application uses no environment variables directly, but other programs it calls may do so.

Copyright

Author

darcs-buildpackage, its modules, documentation, executables, and all included files, except where noted, was written by John Goerzen <`jgoerzen@complete.org`> and copyright is held as stated in the COPYRIGHT section.

darcs-buildpackage may be downloaded from the Debian packaging system or at http://packages.debian.org/darcs-buildpackage.

darcs-buildpackage may also be downloaded using Darcs by running **darcs get --partial http://darcs.complete.org/darcs-buildpackage**.

See Also

Detailed usage information, including a background and a description of how the tools fit together, can be found in the darcs-buildpackage Manual. On Debian systems, you may find this at `/usr/share/doc/darcs-buildpackage`.

darcs-buildpackage applications have references in that manual and the following manpages: dbp-importdsc(1), dbp-importorig(1), dbp-markdeb(1), darcs-buildpackage(1).

Other related references include: darcs(1), debuild(1), dbp-get(1), dpkg-buildpackage(1).

Index