.Deb

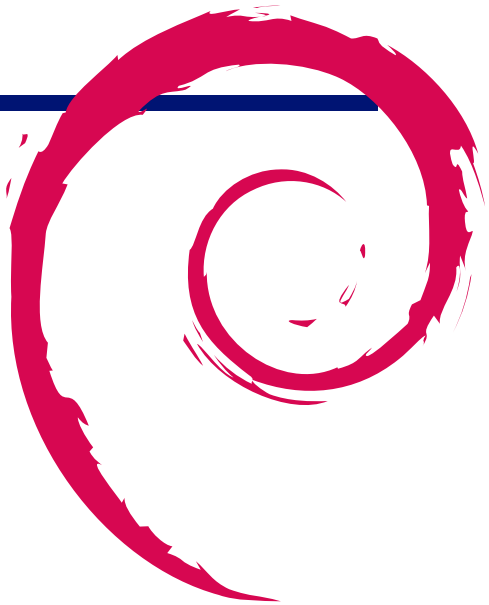
銀河系唯一のDebian専門誌

2022 年 3 月 19 日

git-buildpackage特集

デビアン勉強会

目次



1 最近の Debian 関連のミーティング報告

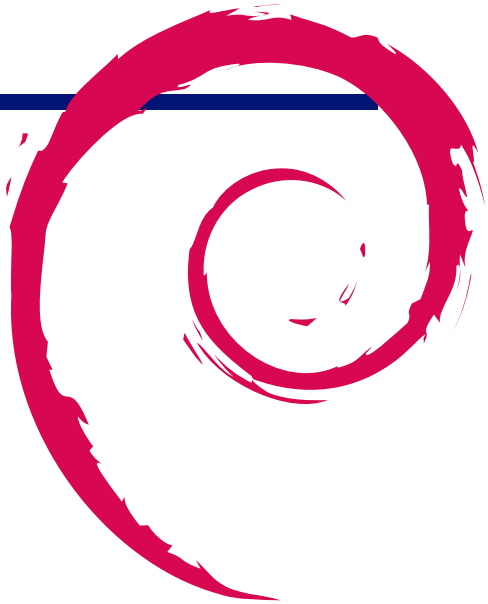
杉本 典充

1.1 2022 年 2 月度 東京エリア・関西合同 Debian 勉強会

2022 年 2 月 19 日 (土) に東京エリア Debian 勉強会と関西 Debian 勉強会の合同でオンラインによる Debian 勉強会を開催しました。参加者は 20 名でした。

セミナーは林さんの「技術書典 12 で Debian の薄い本を頒布した話」、taiseiyo さんの「Debian で始める Python Programming」の 2 本を発表しました。

勉強会の終了後、参加者同士で Debian や OSS に関する話の情報交換を行いました。



2 事前課題

杉本 典充

今回の事前課題は以下です。

1. git-buildpackage を知っていますか。
2. Debian に関連して知りたいこと、聞いてみたいことがあれば自由に書いてみてください。

2.1 dictoss

1. 知っているが使ったことはない
2. もっと git-buildpackage、X Windows System での日本語入力の仕組み、フォント

2.2 NOKUBI Takatsugu (knok)

1. 使ってパッケージ管理をしている
2. bookwarm リリースの課題

2.3 yosukesan (yosuke_san)

1. 知っているが使ったことはない
2. (回答なし)

2.4 takaswie

1. 知っているが使ったことはない
2. hinawa-utils v0.3.0 をリリースしたので、deb パッケージングについて相談したいです。

2.5 yy_y_ja_jp

1. 使ってパッケージ管理をしている
2. (回答なし)

2.6 Hiroyuki Yamamoto (yama1066)

1. 初めて知った
2. (回答なし)

2.7 レドレ (redred)

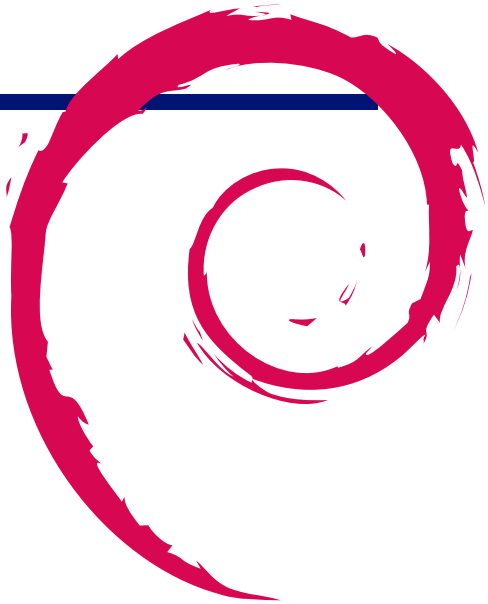
1. 初めて知った
2. 初見者です。

2.8 yukiy9999999999999999999999999

1. 使ったことがある
2. (回答なし)

2.9 YukiharuYABUKI

1. 使ってパッケージ管理をしている
2. (回答なし)



3 git-buildpackage を使ってみる

杉本典充

3.1 はじめに

debian パッケージを作成し管理する方法はいろいろあると思います。
今回は git でパッケージを管理し、ビルドする仕組みを提供する git-buildpackage (gbp) について調べてみました。

3.2 git-buildpackage とは

debian では「git-buildpackage」パッケージを提供しています。sid では git-buildpackage-0.9.25 を提供しており、gbp コマンド[*1]、git-pbuilder コマンドが含まれています。
gbp コマンドには debian パッケージを開発する上で様々な操作ができる便利な以下のサブコマンドがあります[*2]。

- gbp buildpackage
- gbp import-orig
- gbp export-orig
- gbp import-dsc,dscs
- gbp import-ref
- gbp dch
- gbp pq
- gbp pull,clone,push
- gbp create-remote-repo
- gbp tag
- gbp config
- gbp pristine-tar
- gbp setup-gitattributes

なお、git-buildpackage というパッケージ名になっていますが、他のバージョン管理システムに対応した cvs-buildpackage、svn-buildpackage、mercurial-buildpackage パッケージも存在します。

[*1] 中身は python3 で書かれています。
[*2] $ gbp –list-cmds

3.3 git-buildpackage を使って tarball を新規にパッケージ化する

ここでは例として、GNU hello[*3] の tarball を debian パッケージ化[*4]する作業を gbp コマンドを使って行ってみます。

なお、今回の作業環境は Debian sid とします。

3.3.1 必要な debian パッケージのインストール

まずは git-buildpackage パッケージを apt-get コマンドでインストールします。そして、debian パッケージの作成に必要な build-essential、大変便利な debhelper をインストールします。

```
$ sudo apt-get update
$ sudo apt-get install git-buildpackage
$ sudo apt-get install build-essential debhelper dh-make
```

今回 Debian パッケージを作成する GNU hello のビルドに使うツール、およびビルドに必要な依存パッケージをインストールしておきます。

```
$ sudo apt-get install wget
$ sudo apt-get install gnulib help2man
```

3.3.2 git リポジトリの作成と tarball のインポート

まずは GNU hello の tarball をダウンロードします。

```
$ wget https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

git のローカルブランチのディレクトリ名は tarball のファイル名と同じ ”hello” にすることとし、git リポジトリ ”hello” をローカルに作成します。最近の git の使い方としてデフォルトを main ブランチにするように変わってきていますので、ブランチ名を指定しています[*5]。

```
$ git init hello --initial-branch=main
$ cd hello
```

tarball を指定して gbp import-orig コマンドを実行すると、tarball を git に取り込んでコミットします。デフォルトでは master ブランチを作成しますが、`https://salsa.debian.org/` の git デフォルトブランチは main のため –debian-branch=main を指定しています。

```
$ gbp import-orig --debian-branch=main --pristine-tar ../hello-2.10.tar.gz

What will be the source package name? [hello]
What is the upstream version? [2.10]
gbp:info: Importing '../hello-2.10.tar.gz' to branch 'upstream'...
gbp:info: Source package is hello
gbp:info: Upstream version is 2.10
gbp:info: Successfully imported version 2.10 of ../hello_2.10.orig.tar.gz
```

gbp import-orig で tarball を取り込むと以下のブランチとタグが作成されます。

```
$ git branch
* main
  pristine-tar
  upstream
```

```
$ git tag
upstream/2.10
```

[*3] `https://www.gnu.org/software/hello/`

[*4] `https://packages.debian.org/ja/sid/hello` が既に存在します

[*5] git-buildpackage ではデフォルトのブランチ名は現状 master のままになっています。

git-buildpackage において git のブランチは以下の使い分けをします。

- master (または main) ブランチ：debian パッケージ全体の内容を含むブランチ
- upstream ブランチ：upstream の配布物をそのまま展開した内容を含むブランチ
- pristine-tar ブランチ：debian パッケージのビルドに使う upstream ブランチから生成する orig.tar.gz と upstream の tarball の差分をバイナリで保持するブランチ

gbp import-orig コマンドに –pristine-tar オプションを指定した場合は pristine-tar ブランチも作成されます。

pristine-tar ブランチの作成は必須ではありません。ただ、tarball を展開して upstream ブランチに保存して、逆に upstream ブランチのファイルから再生成した tarball ファイルのハッシュ値が、gbp import-orig で取り込んだ tarball のハッシュ値と同じにならないことがあります。pristine-tar ブランチにはその差分をバイナリで保存して、gbp buildpackage で debian パッケージをビルドするときに適用することでハッシュ値が同じになるように調整しています。

```
$ git checkout pristine-tar
Switched to branch 'pristine-tar'
Your branch is up to date with 'origin/pristine-tar'.

$ ls -l
合計 28
-rw-r--r-- 1 norimitu norimitu 6107  3 月 19 04:25 hello_2.10.orig.tar.gz.delta
-rw-r--r-- 1 norimitu norimitu   41  3 月 19 04:25 hello_2.10.orig.tar.gz.id

$ cat hello_2.11.orig.tar.gz.id
61732d00c5e503e9ced7f7f57f5b1c14a449c95c

$ file hello_2.11.orig.tar.gz.delta
hello_2.11.orig.tar.gz.delta: gzip compressed data, from Unix, original size modulo 2^32 30720

$ git checkout master
```

3.3.3 debian ディレクトリを生成する

debian パッケージを作成するのは、main ブランチに debian ディレクトリを作成します。dh_make コマンドを使って作成できます。

```
$ dh_make -p hello_2.10

Type of package: (single, indep, library, python)
[s/i/l/p]?
Maintainer Name     : Norimitsu Sugimoto
Email-Address       : dictoss@live.jp
Date                : Thu, 17 Mar 2022 13:27:56 +0000
Package Name        : hello
Version             : 2.10
License             : blank
Package Type        : single
Are the details correct? [Y/n/q]
Skipping creating ../hello_2.10.orig.tar.gz because it already exists
Done. Please edit the files in the debian/ subdirectory now.
```

dch コマンドで debian/changelog ファイルを開いて編集します[*6]。

```
$ dch

hello (2.10-1) UNRELEASED; urgency=medium

  * Initial release (Closes: #nnnn)  <nnnn is the bug number of your ITP>

 -- Norimitsu Sugimoto <dictoss@live.jp>  Thu, 17 Mar 2022 13:31:04 +0000
```

debian/control ファイルを編集し、パッケージのメタ情報を更新します。

[*6] 通常、debian パッケージを新規作成する場合は ITP のバグ報告を行っている前提があり、debian/changelog ファイルにそのバグ番号を含めます。

```
$ vim debian/control

Source: hello
Section: devel
Priority: optional
Maintainer: Norimitsu Sugimoto <dictoss@live.jp>
Build-Depends: debhelper-compat (= 13), autotools-dev, gnulib, help2man
Standards-Version: 4.6.0
Homepage: http://www.gnu.org/software/hello/
#Vcs-Browser: https://salsa.debian.org/debian/hello
#Vcs-Git: https://salsa.debian.org/debian/hello.git
Rules-Requires-Root: no

Package: hello
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: example package based on GNU hello
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's 'hello world' program
 (which is itself an example for the GNU Project).
```

debian/rules ファイルを編集し、debhelper を使って debian パッケージをビルドできるように調整します。

```
$ vim debian/rules

%:
	dh $@

override_dh_auto_clean:
	[ ! -f Makefile ] || $(MAKE) distclean
```

それでは gbp buildpackage コマンドを使って debian パッケージをビルドしてみます。しかし、コミットしていないファイルがあるとコマンドがエラーになります。まずは編集した debian ディレクトリを git へコミットします。

```
$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

gbp:error: You have uncommitted changes in your source tree:
gbp:error: On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
	debian/

nothing added to commit but untracked files present (use "git add" to track)

gbp:error: Use --git-ignore-new to ignore.
```

```
$ git add debian
$ git commit -m "add debian directory."
```

再度、gbp buildpackage コマンドを実行します。debian パッケージのビルドに成功しました。しかし、lintian の警告がたくさん出ていますのでリリースするまでに警告をできるだけ減らすパッケージを見直す必要があります。

```
$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

gbp:info: Performing the build
dpkg-buildpackage -us -uc -ui -i -I
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1
dpkg-buildpackage: info: source distribution UNRELEASED
dpkg-buildpackage: info: source changed by Norimitsu Sugimoto <dictoss@live.jp>
 dpkg-source -i -I --before-build .
dpkg-buildpackage: info: host architecture amd64
 debian/rules clean

  (省略)

dpkg-buildpackage: info: full upload (original source is included)
Now running lintian hello_2.10-1_amd64.changes ...
E: hello: changelog-is-dh_make-template
  (省略)
W: hello: wrong-bug-number-in-closes #nnnn in the installed changelog (line 4)
Finished running lintian.
```

これで debian パッケージのビルドができました。ビルド時に生成されたファイルを削除してから、main ブランチにタグを作成し、リモートの git リポジトリに push します。

```
$ gbp buildpackage --git-tag-only --git-debian-branch=main
gbp:info: Tagging Debian package 2.10-1 as debian/2.10-1 in git

$ git tag
debian/2.10-1
upstream/2.10
```

```
$ git remote add origin git@salsa.debian.org:dictoss-guest/hello.git
$ git remote -v
origin  git@salsa.debian.org:dictoss-guest/hello.git (fetch)
origin  git@salsa.debian.org:dictoss-guest/hello.git (push)

$ git push -u origin --all
$ git push -u origin --tag
```

3.4 upstream で新しいバージョンの tarball がリリースされた場合の作業

まず、更新された tarball をダウンロードします。

```
$ cd ..
$ wget https://ftp.gnu.org/gnu/hello/hello-2.11.tar.gz
```

gbp import-orig コマンドで tarball を git のローカルブランチに取り込みます。

```
$ gbp import-orig --debian-branch=main --pristine-tar ../hello-2.11.tar.gz

What is the upstream version? [2.11]
gbp:info: Importing '../hello-2.11.tar.gz' to branch 'upstream'...
gbp:info: Source package is hello
gbp:info: Upstream version is 2.11
gbp:info: Replacing upstream source on 'main'
gbp:info: Successfully imported version 2.11 of ../hello_2.11.orig.tar.gz
```

git branch を実行するとブランチの数に変化はありません。

```
$ git branch
* main
  pristine-tar
  upstream
```

git tag を実行すると "upstream/2.11" が追加されています。

```
$ git tag

debian/2.10-1
upstream/2.10
upstream/2.11
```

dch コマンドで debian/changelog ファイルを開いて編集します。

```
$ dch

hello (2.11-1) UNRELEASED; urgency=medium

  * New upstream version 2.11

 -- Norimitsu Sugimoto <dictoss@live.jp>  Fri, 19 Mar 2022 14:10:38 +0000

hello (2.10-1) UNRELEASED; urgency=medium

  * Initial release (Closes: #nnnn)  <nnnn is the bug number of your ITP>

 -- Norimitsu Sugimoto <dictoss@live.jp>  Fri, 18 Mar 2022 13:39:37 +0000
```

debian ディレクトリの変更内容をコミットします。

```
$ git add debian
$ git commit -m "New upstream version 2.11."
```

gbp buildpackage を実行して debian パッケージをビルドしてみるとエラーになりました。

```
$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

   （省略）
*** error: gettext infrastructure mismatch: using a Makefile.in.in from gettext
version 0.18 but the autoconf macros are from gettext version 0.20
make[3]: *** [Makefile:165: check-macro-version] Error 1
make[3]: *** Waiting for unfinished jobs....
（省略）
dh_auto_build: error: make -j3 returned exit code 2
make: *** [debian/rules:18: binary] エラー 25
dpkg-buildpackage: error: debian/rules binary subprocess returned exit status 2
debuild: fatal error at line 1182:
dpkg-buildpackage -us -uc -ui -i -I failed
gbp:error: 'debuild -i -I' failed: it exited with 29
```

gettext のバージョンが sid のバージョンと tarball のファイルに書いてあるバージョンが合わないためエラーになったようです。gbp pq コマンドを使ってパッチ作成用のブランチを作ります。gbp pq import コマンドを実行すると patch-queue/main ブランチが作成されチェックアウトした状態になります。

```
$ gbp pq import
gbp:info: Trying to apply patches at 'cacbfce059a86c836513ff70eb6af780d9255910'
gbp:info: 0 patches listed in 'debian/patches/series' imported on 'patch-queue/main'

$ git branch
  main
* patch-queue/main
  pristine-tar
  upstream
```

patch-queue/main ブランチ上で変更が必要なファイルを編集します。

```
$ vim configure.ac
$ git diff
diff --git a/configure.ac b/configure.ac
index 6d52c41..f101d7b 100644
--- a/configure.ac
+++ b/configure.ac
@@ -62,7 +62,7 @@ dnl Ensure VLAs are not used.
 AC_DEFINE([GNULIB_NO_VLA], [1], [Define to 1 to disable use of VLAs])

 dnl i18n support from GNU gettext.
-AM_GNU_GETTEXT_VERSION([0.18.1])
+AM_GNU_GETTEXT_VERSION([0.21])
 AM_GNU_GETTEXT([external])

 AC_CONFIG_FILES([Makefile
```

patch-queue/main ブランチに変更したファイルをコミットします。

```
$ git add configure.ac
$ git commit -m "change to gettext version to use with sid."
[patch-queue/main cc568aa] change to gettext version to use with sid.
 1 file changed, 1 insertion(+), 1 deletion(-)
```

すべてのファイルの変更が完了した後、gbp pq export コマンドを実行して編集を終了します。gbp pq export コマンドを実行すると main ブランチをチェックアウトした状態になり、main ブランチと patch-queue/main ブランチの差分が debian/patches/ ディレクトリに生成されます。

```
$ gbp pq export
gbp:info: On 'patch-queue/main', switching to 'main'
gbp:info: Generating patches from git (main..patch-queue/main)
```

```
$ git branch
* main
  patch-queue/main
  pristine-tar
  upstream
```

```
$ git status .
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        debian/patches/

nothing added to commit but untracked files present (use "git add" to track)

$ ls -l debian/patches/
合計 8
-rw-r--r-- 1 norimitu norimitu 596  3 月 19 02:28 0001-change-to-gettext-version-to-use-with-sid.patch
-rw-r--r-- 1 norimitu norimitu  53  3 月 19 02:28 series
```

生成された debian/patches ディレクトリを main ブランチにコミットします。

```
$ git add debian/patches
$ git commit -m "change to gettext version to use with sid."
[main 6ae3e38] change to gettext version to use with sid.
 2 files changed, 22 insertions(+)
 create mode 100644 debian/patches/0001-change-to-gettext-version-to-use-with-sid.patch
 create mode 100644 debian/patches/series
```

gbp buildpackage を実行すると、更新した debian パッケージをビルドできます。

```
$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

$ ls ..
hello                          hello_2.10-1.debian.tar.xz    hello_2.10-1_amd64.deb      hello_2.11-1_amd64.buildinfo
hello-2.10.tar.gz              hello_2.10-1.dsc              hello_2.10.orig.tar.gz      hello_2.11-1_amd64.changes
hello-2.11.tar.gz              hello_2.10-1_amd64.build      hello_2.11-1.debian.tar.xz  hello_2.11-1_amd64.deb
hello-dbgsym_2.10-1_amd64.deb  hello_2.10-1_amd64.buildinfo  hello_2.11-1.dsc            hello_2.11.orig.tar.gz
hello-dbgsym_2.11-1_amd64.deb  hello_2.10-1_amd64.changes    hello_2.11-1_amd64.build
```

ビルド時に生成されたファイルを削除してから、main ブランチにタグを作成し、リモートの git リポジトリに push します。

```
$ gbp buildpackage --git-tag-only --git-debian-branch=main
gbp:info: Tagging Debian package 2.11-1 as debian/2.11-1 in git

$ git tag
debian/2.10-1
debian/2.11-1
upstream/2.10
upstream/2.11
```

```
$ git push -u origin main upstream pristine-tar
$ git push -u origin --tag
```

作業を終えた patch-queue/main ブランチは不要のため削除します。

```
$ git branch -D patch-queue/main
```

3.5 今回取り扱えなかったこと

今回取り扱えなかったことはたくさんあります。

- upstream が git リポジトリのみで tarball を提供していない場合
- sid、stable、stable-backports、experimental を平行で管理する方法
- gbp buildpackage –git-pbuilder
- gbp.conf
- upstream のソースコードにすでに debian ディレクトリが存在している場合

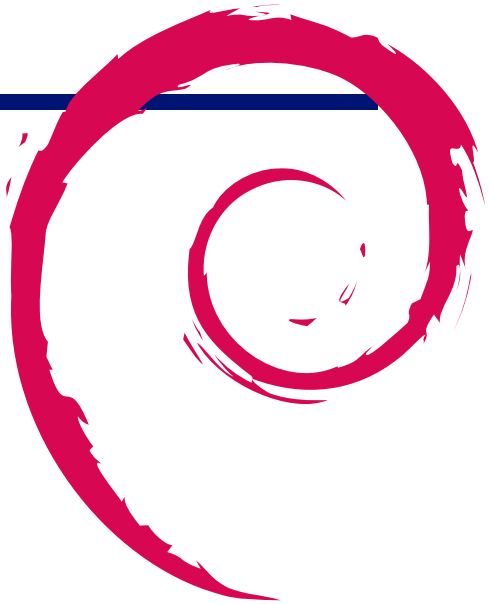
3.6 おわりに

git-buildpackage パッケージに入っている gbp コマンドを調べてみました。quilt コマンドを使ったパッチの作成を gbp pq コマンドでより便利に使えるのがよいと思いました。

実際にパッケージをメンテナンスするときには、複数のリリースを 1 つの git リポジトリで管理したいのだと思います。その場合にどのように git-buildpackage を使いこなせばよいか、まだまだ勉強が必要であると感じました。

3.7 参考文献

- Debian Wiki - PackagingWithGit `https://wiki.debian.org/PackagingWithGit`
- Debian パッケージング道場 git-buildpackage `https://tokyodebian-team.pages.debian.net/pdf2015/debianmeetingresume201509-presentation.pdf`
- Debian 新メンテナーガイド 第 6 章 パッケージのビルド `https://www.debian.org/doc/manuals/maint-guide/build.ja.html`
- git-buildpackage を用いた deb パッケージ管理方法の紹介 `https://blog.cybozu.io/entry/2019/04/11/110000`



4 メモ

Debian 勉強会資料

2022 年 3 月 19 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）