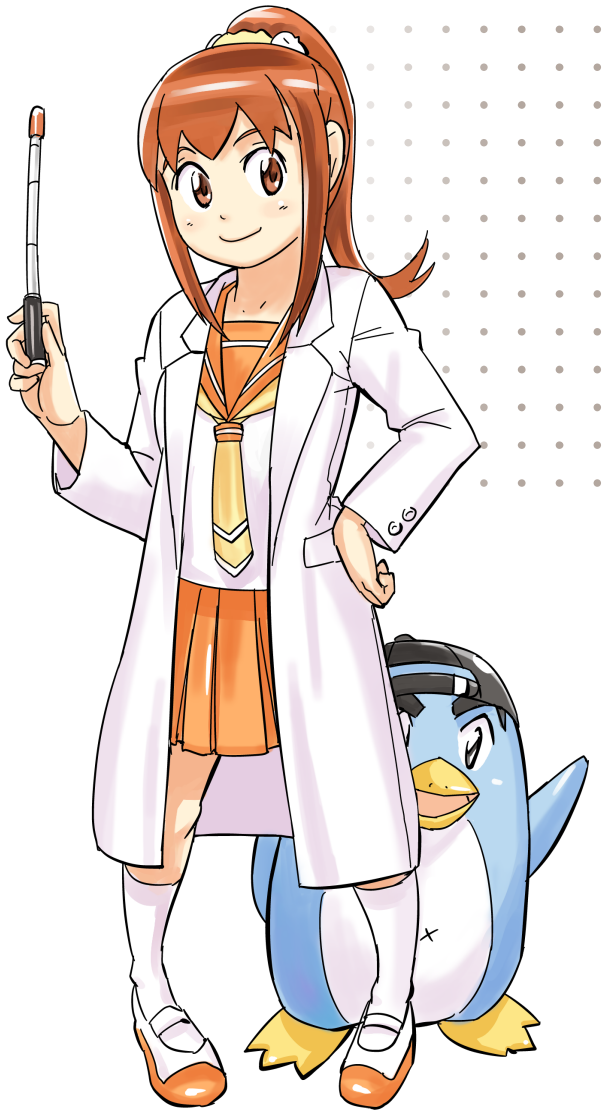
ubuntu

pbuilderを使ってみよう



2011/02/27

関西Debian勉強会

水野 源@Ubuntu Japanese Team

Self introduction

- Hajime MIZUNO
- Ubuntu Japanese Team所属
  - おもにイベント担当？
- 職業:モンスターハンター
- hajime.mizuno@gmail.com
- Twitter:@mizuno_as
- Launchpad:mizuno-as

今日のお題

- pbuilderの紹介
  - pbuilderってなに?
  - pbuilderのメリット
  - pbuilderの使い方
- cowbuilder
- Ubuntu的なpbuilder/cowbuilder

pbuilderってなに?

- Personal Debian Package Builder
  - シェルスクリプトで実装
  - debootstrap
  - chrootシステムを構築して、そのなかでパッケージのコンパイルができるしくみ
  - あるパッケージをコンパイルするために必要な、最小限のパッケージのみをインストールして、コンパイルを実行する

pbuilderのメリット

- クリーンな環境でパッケージをビルドできる!
- 依存関係のテストができる!
- 別アーキテクチャ、別リリース向けビルドができる!

→ なんのこっちゃ orz

理解しやすいようにおさらい

一般的なパッケージビルド手順

```
$ apt-get source PKGNAME
        ↑ソースパッケージの取得と展開

$ apt-get build-dep PKGNAME
        ↑ビルド依存パッケージのインストール

$ cd PKGDIRNAME
$ dpkg-buildpackage
        ↑ビルド実行
```

たとえばnanoの場合

```
$ apt-get source nano
(...略...)

$ ls
nano-2.2.4
nano_2.2.4-1.debian.tar.gz
nano_2.2.4-1.dsc
nano_2.2.4.orig.tar.gz
```

たとえばnanoの場合

```
$ cd nano-2.2.4
$ dpkg-buildpackage
(...略...)

dpkg-checkbuilddeps: Unmet build
dependencies: libncurses5-dev libncursesw5-dev
libslang2-dev

dpkg-buildpackage: warning: Build dependencies/
conflicts unsatisfied; aborting.
```

足りないパッケージがあってビルドできない！

たとえばnanoの場合

```
$ apt-get build-dep nano
(...略...)

The following NEW packages will be installed:
  bsdmainutils debhelper gettext gettext-base
  groff-base html2text intltool-debian
  libcroco3 libncurses5-dev libncursesw5-dev
  libpng12-dev libslang2-dev
  libunistring0 libxml2 man-db po-debconf
  Zlib1g-dev
```

nanoのビルドにはこれだけのパッケージが必要！
　＝ 開発マシンにインストールしておく必要がある

パッケージビルドのまとめ

- パッケージをビルドするには「ビルド時に」依存しているパッケージをインストールする必要がある
- apt-getはBuild-Dependsをもとに、依存しているパッケージを一括導入できる
- ビルド依存パッケージはdebian/controlにBuild-Dependsとして記述してある

パッケージビルドにつきまとう問題

- パッケージをビルドするには「ビルド時に」依存しているパッケージをインストールする必要がある
  - 開発用パッケージで環境が汚れていく
  - 他の人のマシンでビルドできる？
- ビルド依存パッケージはdebian/controlにBuild-Dependsとして記述してある
  - 依存関係は正しく記述されてる？

→ これを解決するのがpbuilder

実際に使ってみよう

pbuilderのインストール

- pbuilderパッケージをインストール

```
$ sudo apt-get install pbuilder
```

pbuilder環境の構築

```
$ sudo pbuilder --create ¥
--distribution sid ¥
--architecture i386 ¥
--mirror "http://ftp.riken.jp/Linux/debian/debian"

=> /var/cache/pbuilder/base.tgz が作成される
```

pbuilder環境でのビルド

- pbuilder –buildコマンドを実行する
  - 引数にはソースパッケージの*.dscファイル
- 一時ディレクトリを作成し、base.tgzを展開
  - ビルドが終了すると一時ディレクトリは削除
- /var/cache/pbuilder/resultにできあがり

```
$ sudo pbuilder --build nano_2.2.4-1.dsc
```

複数のpbuilder環境

- basetgzオプションで使用するファイルを指定

```
$ sudo pbuilder --create --distribution sid ¥
--architecture i386 ¥
--basetgz /var/cache/pbuilder/sid-i386.tgz

$ sudo pbuilder --build ¥
--basetgz /var/cache/pbuilder/sid-i386.tgz ¥
nano_2.2.4-1.dsc
```

pbuilder環境へログイン

- pbuilder環境へログインしてシェルを取る
- pbuilder –-loginコマンドを使用
- Squeeze上でsidをちょっと試す、などが可能
- 当然ログアウト後には変更は破棄

```
$ sudo pbuilder --login ¥
--basetgz /var/cache/pbuilder/sid-i386.tgz
```

pbuilder環境の更新

- ビルドするためには最新を保つ必要がある
- pbuilder --updateコマンドを使用

```
$ sudo pbuilder --update ¥
--basetgz /var/cache/pbuilder/sid-i386.tgz
```

pbuilder環境の更新

- 環境に手動で変更を加える
- ログアウト後にも変更を破棄せず、basetgzを再構築する
- pbuilder –login に --save-after-loginを追加

```
$ sudo pbuilder --login ¥
--save-after-login ¥
--basetgz /var/cache/pbuilder/sid-i386.tgz
```

pbuilderをさらに便利に使う

cowbuilder

cowbuilderとは

- より効率のよいpbuilder
- 環境をtarballに圧縮しない
- 一時ディレクトリ作成にハードリンクを使う

  ファイル展開、コピー、削除を省ける

Copy-On-Write

- コピーしたふりをして原本を参照させておく
- 変更が加わった時点ではじめて実際に書きこむ
- つまり変更が必要な部分のみコピーする

効率的!

→ これを実現しているのがcowdancer

Copy-On-Write

```
$ sudo cowbuilder --create --distribution sid ¥
--architecture i386
$ cd /var/cache/pbuilder/base.cow/
$ ls -i etc/debian-version
7020666 /etc/debian_version   ←原本
$ sudo cowbuilder --login
# ls -i /etc/debian-version
7020666 /etc/debian_version   ←一時ファイル
# vi /etc/debian_version
# ls -i /etc/debian_version
7070836 /etc/debian_version   ←変更後
```

Ubuntu的なpbuilder/cowbuilder

pbuilder-dist/cowbuilder-dist

pbuilder-dist/cowbuilder-dist

- pbuilder/cowbuilderのラッパスクリプト
- Pythonで実装
- ubuntu-dev-toolsパッケージに同梱
  - Debianでも使える!
- pbuilder-distribution.shのよりよい再実装

pbuilder-distの動作

```
$ pbuilder-dist sid i386 create
```

```
$ sudo /usr/sbin/pbuilder --create ¥
--basetgz "/home/mizuno/pbuilder/sid-i386-base.tgz" ¥
--distribution "sid" ¥
--buildresult "/home/mizuno/pbuilder/sid-i386_result/" ¥
--aptcache "/var/cache/apt/archives/" ¥
--override-config ¥
--logfile /home/mizuno/pbuilder/sid-i386_result/
last_operation.log ¥
--mirror "ftp://ftp.debian.org/debian" ¥
--components "main contrib non-free" ¥
--debootstrapopts --arch="i386"
```

pbuilder-distの動作

- Pbuilderに様々なオプション規定値を自動設定
- ユーザ権限で実行可能(内部でsudo)
- ホームディレクトリにファイル作成
- オプションが簡単
- 異なるバージョンを共存させやすい
  - .pbuilderrcより便利

pbuilder-distの呼び出し方

- シンボリックリンクを使って呼び出す

```
$ ln -s /usr/sbin/pbuilder-dist pbuilder-sid-amd64
$ ./pbuilder-sid-amd64 create
    ↑これは pbuilder-dist sid amd64 create と同等
```

```
$ sudo /usr/sbin/pbuilder --create ¥
--basetgz "/home/mizuno/pbuilder/sid-amd64-base.tgz" ¥
--distribution "sid" ¥
(...snip...)
```

pbuilder-distのメリット

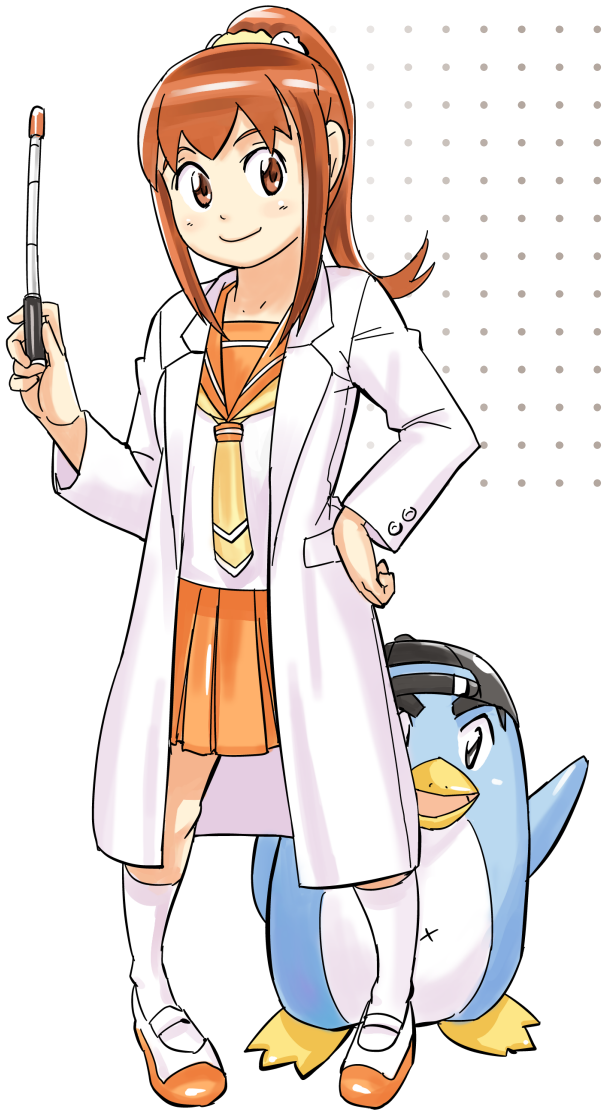
- 簡単に複数のpbuilder環境を使い分けられる
  - Ubuntuは複数のリリースや、上流であるDebianでのテストが必要なため、リリースやディストリを簡単に切り替えられることが大事
- ユーザのホームにファイルやパッケージを作成
  - ユーザ権限での管理がしやすい
- オプションではなく名前で対象を切り替えられる
  - 複数のリリースを管理する場合に便利

今日のまとめ

- pbuilder/cowbuilderを使えば、普段使っている環境を開発用パッケージで汚すことなく、パッケージのビルドを行うことができる
- 基本的なDebianシステムでビルドが可能なことを確認することができるので、作成したパッケージは必ずテストしよう
- cowbuilderを使えば、より高速にpbuilder環境を利用することができる
- pbuilder-distを使えば、リリース間、ディストリ間を渡り歩くのも簡単

ubuntu

質疑応答



以上

どうもありがとうございました

ubuntu

ライセンス

Write by Hajime MIZUNO <hajime.mizuno@gmail.com>
Illustration by Hiroshi SEO <seotch@gmail.com>