autopkgtest について調べてみた

東京エリア・関西合同 Debian 勉強会

Norimitsu SUGIMOTO (杉本 典充)
dictoss@live.jp

2024-06-22

アジェンダ

- 自己紹介
- おさらい：debian パッケージのビルド方法
- autopkgtest とは
- autopkgtest を用いたテストケースの作成と実行
- 仮想環境における autopkgtest の実行方法
- まとめ
- 参考資料

自己紹介

- Norimitsu SUGIMOTO (杉本 典充)
- dictoss@live.jp
- Twitter: @dictoss
- Debian を使い始めたのは 3.1 sarge が testing の頃
- 仕事はソフトウェア開発者をやってます
- python と Django の組み合わせで使うことが多いです

おさらい：debian パッケージのビルド方法

debian ディレクトリの中身

Debian 新メンテナーガイド[1] には.deb ファイルの元になる src パッケージの中の debian ディレクトリについて説明がある

- debian/control
- debian/copyright
- debian/changelog
- debian/rules
- debian/watch
- debian/source/format

Debian 新メンテナーガイドには書いていませんが、以下のファイルも扱える

- debian/upstream/metadata
- **debian/tests/control** ←今回はこのファイルの話

[1] https://www.debian.org/doc/manuals/maint-guide/

ビルド方法

debian パッケージのビルドに必要なツールをインストール

```
$ sudo apt-get install build-essential devscripts
```

既存の grep パッケージを自分でビルドしてみる場合

```
$ sudo apt-get build-dep grep
$ apt-get source grep
$ cd grep-3.11
$ debuild -uc -us
$ ls ../grep*.deb
../grep-dbgsym_3.11-4_amd64.deb  ../grep_3.11-4_amd64.deb
```

ビルドした debian のバイナリパッケージの中身がきちんと動くかを確認するには？

→ x 自分の環境にインストールして頑張ってテストする

→ o debian/tests/control を作成し、autopkgtest でテストする

autopkgtest とは

autopkgtest とは

- インストールされたバイナリパッケージをソースパッケージに付属するテストケースを用いてテストする方法
- 「# apt-get install autopkgtest」でインストールできる
- 詳しい説明
  - man autopkgtest
  - https://wiki.debian.org/ContinuousIntegration/autopkgtest
  - https://people.debian.org/~eriberto/README.package-tests.html
  - https://ci.debian.net/doc/file.TUTORIAL.html
- autopkgtest があると unstable から testing への移行が早い、unblock リクエストが不要な場合あり
  - https://release.debian.org/bookworm/freeze_policy.html

autopkgtest とは

- テストベッド：テスト対象のバイナリパッケージをインストールしたテスト環境
  - ローカル環境を使う方法
  - 仮想環境を使う方法 (schroot / lxc / qemu)
- autopkgtest のベストプラクティス[2]
  - 【重要】テストケースは src パッケージにあるが、テスト対象は src パッケージ内のビルド成果物ではない
  - テストは実行されるシステム (=テストベッド) を変更する可能性がある
  - upstream のテストが付属する場合はそれを使うことを推奨
  - 各プログラム言語で利用可能な標準的なテストソリューションを使用する (shunit2, nose, prove)
  - 1 件当たりのテストは短い時間で完了させるようにする (タイムアウトが起こります)

[2] https://wiki.debian.org/ContinuousIntegration/AutopkgtestBestPractices

autopkgtestを用いたテストケースの作成と実行

autopkgtest の定義の例

grep パッケージにおける src パッケージの debian/tests

```
$ ls -ln debian/tests/
合計 20
-rw-rw-r-- 1 1000 1000 309 11 月 23  2023 control
-rw-rw-r-- 1 1000 1000 234 11 月 23  2023 dirfd
-rw-rw-r-- 1 1000 1000  93  8 月 23  2023 egrep-fgrep
-rwxrwxr-x 1 1000 1000 318  8 月 23  2023 stray-backlash-warn
-rw-rw-r-- 1 1000 1000 240  1 月  4 20:56 upstream-test-suite
```

debian/tests/control の Tests キーの値の名前がテストケースで、debian/tests/ 配下のスクリプトファイル名に対応

```
$ cat debian/tests/control
Tests: upstream-test-suite
Depends: @builddeps@, fakeroot, libpcre2-8-0, locales-all
# TODO: remove allow-stderr. For now it is needed since for being able
#       to ignore expensive checks and for ignore unavailable locale
#       warnings.
Restrictions: allow-stderr

Tests: egrep-fgrep, stray-backlash-warn dirfd
```

autopkgtest の実行方法

- パッケージをビルド、インストールし、autopkgtest を実行する
- autopkgtest を実行して終了コードが 0 であればテストケースはすべて合格
- オプションの "null" は テストベッド環境を指定するオプションでローカル環境を差す

```
$ cd grep-3.11
$ debuild -uc -us
$ sudo dpkg -i ../grep_3.11-4_amd64.deb
$ autopkgtest --no-built-binaries -- null
  (テスト処理の出力を表示)
$ echo $?
0
```

autopkgtest のテストケースの作成 (1)

テストケースを作成する

- 実行するコマンドの終了コードが成功 (0) になるテストコードを書く
- set -e を指定するなど、途中のテストで失敗した場合は異常終了させる

```
$ vi debian/tests/mytest1
#!/bin/sh
set -e

# test egrep and -E option
printf 'aabbccdd' | grep -E 'a{2,}'
printf 'aabbccdd' | egrep 'a{2,}'
```

debian/tests/control に追加したテストを加える

```
Tests: egrep-fgrep, stray-backlash-warn dirfd, mytest1
```

autopkgtest のテストケースの作成 (2)

作成したテストケースを単体で実行 (テストに合格した場合)

```
$ autopkgtest --no-built-binaries --test-name=mytest1 -- null
autopkgtest [23:42:00]: starting date and time: 2024-06-20 23:42:00+0900
autopkgtest [23:42:00]: version 5.35
autopkgtest [23:42:00]: host nor-deb-sid-1; command line:
  /usr/bin/autopkgtest --no-built-binaries --test-name=mytest1 -- null
autopkgtest [23:42:00]: testbed dpkg architecture: amd64
autopkgtest [23:42:00]: testbed apt version: 2.9.5
autopkgtest [23:42:00]: testbed running kernel: Linux 6.8.12-amd64 #1
  SMP PREEMPT_DYNAMIC Debian 6.8.12-1 (2024-05-31)
autopkgtest [23:42:00]: @@@@@@@@@@@@@@@@@@@@ built-tree .
autopkgtest [23:42:00]: testing package grep version 3.11-4
autopkgtest [23:42:00]: test mytest1: preparing testbed
autopkgtest [23:42:00]: test mytest1: [----------------------
aabbccdd
aabbccdd
autopkgtest [23:42:00]: test mytest1: ----------------------]
autopkgtest [23:42:00]: test mytest1:  - - - - - - - - - - results - - - - -
mytest1                PASS
autopkgtest [23:42:00]: @@@@@@@@@@@@@@@@@@@@ summary
mytest1                PASS

$ echo $?
0
```

autopkgtest のテストケースの作成 (3)

作成したテストを単体で実行 (1 行目のテストが失敗した場合)

```
$ vi debian/tests/mytest1
#!/bin/sh
set -e

# test egrep and -E option
printf 'aabbccdd' | grep -E 'e{2,}'
printf 'aabbccdd' | egrep 'a{2,}'
```

```
$ autopkgtest --no-built-binaries --test-name=mytest1 -- null
  (省略)
autopkgtest [22:52:17]: test mytest1: [----------------------
autopkgtest [22:52:18]: test mytest1: ----------------------]
autopkgtest [22:52:18]: test mytest1:  - - - - - - - - - - results - - - - -
mytest1                FAIL non-zero exit status 1
autopkgtest [22:52:18]: @@@@@@@@@@@@@@@@@@@@ summary
mytest1                FAIL non-zero exit status 1

$ echo $?
4
```

仮想環境におけるautop-kgtestの実行方法

autopkgtest のテストベッド環境

- ローカル環境でテストを実行する場合の問題点
  - ローカル環境で実行する場合は　ビルド環境 = テストベッド環境　になる
  - バイナリパッケージをインストールしてテストする性質上、テストするパッケージが壊れていた場合は自分のビルド環境を壊す可能性あり
  - 重要なコマンドやライブラリを含むパッケージはテストしにくい
- ビルド環境とテストベッド環境を分離するために仮想環境を利用できる
  - schroot (準備するのが手軽)
  - lxc (`https://ci.debian.net/` で採用)
  - qemu (仮想サーバの上では仮想環境がネストになり実行できない可能性あり)

今回は schroot 環境で実行する手順を紹介します

schroot 環境の準備 (1)

コンテナ環境を debootstrap を使ってインストール

```
$ sudo apt-get install debootstrap schroot
$ sudo mkdir /srv/chroot
$ sudo debootstrap unstable /srv/chroot/sid1 http://deb.debian.org/debian/
$ ls /srv/chroot/sid1
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
```

/etc/schroot/chroot.d 配下の設定ファイルを作成 ([セクション] の中がコンテナ名)

```
$ sudo vi /etc/schroot/chroot.d/sid1.conf
[my-sid1-amd64]
description=Debian unstable amd64
directory=/srv/chroot/sid1
users=dictoss
root-groups=dictoss
aliases=unstable,default
type=directory
profile=desktop
union-type=overlay
```

schroot 環境の準備 (2)

schroot で利用できる環境一覧に表示されます

```
$ schroot -l
chroot:default
chroot:default-source
chroot:my-sid1-amd64
chroot:unstable
chroot:unstable-source
source:default
source:my-sid1-amd64
source:unstable
```

autopkgtest コマンドで 前は "null" だった部分を "schroot" に変更

```
$ cd grep-3.11
$ debuild -uc -us
$ autopkgtest --no-built-binaries -- schroot my-sid1-amd64
-> テストケースの実行に必要なパッケージのインストール、ビルド、テストが始まる
...
```

schroot 環境で autopkgtest

autopkgtest の実行出力

```
...
autopkgtest [00:15:37]: test mytest1: preparing testbed
Reading package lists... Done
Building dependency tree... Done
Starting pkgProblemResolver with broken count: 0
Starting 2 pkgProblemResolver with broken count: 0
Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
autopkgtest [00:15:37]: test mytest1: [-----------------------
aabbccdd
aabbccdd
autopkgtest [00:15:38]: test mytest1: -----------------------]
autopkgtest [00:15:38]: test mytest1:  - - - - - - - - - - results - - - - -
mytest1                PASS
autopkgtest [00:15:38]: @@@@@@@@@@@@@@@@@@@@ summary
upstream-test-suite    PASS
egrep-fgrep            PASS
stray-backlash-warn    PASS
dirfd                  PASS
mytest1                PASS
```

まとめ

まとめ

- autopkgtest のインストール方法、実行方法、テストの書き方を説明しました
- autopkgtest を仮想環境で実行する 1 つの例 (schroot) の方法を紹介しました
- 来年 2025 年は Debian 13 trixie のリリースに向けてフリーズが始まる見込みで、autopkgtest を書いておくと更新したパッケージの testing 移行が早い
- autopkgtest を書いて、品質のよいパッケージを作ってみましょう

参考文献

- Debian Wiki - ContinuousIntegration autopkgtest https://wiki.debian.org/ContinuousIntegration/autopkgtest
- man autopkgtest-virt-schroot(1) https://manpages.debian.org/testing/autopkgtest/autopkgtest-virt-schroot.1.en.html
- Charles Plessy. Debian Continuous Integration, 第 152 回東京エリア Debian 勉強会, 2017/06 https://tokyodebian-team.pages.debian.net/pdf2017/debianmeetingresume201706.pdf
- Youhei SASAKI. 「パッケージ作成環境: sbuild, lintian, piuparts, autopkgtest」 https://uwabami.github.io/cc-env/DebianPackaging.html