東京エリア デビアン 勉強会

Debian勉強会幹事 上川純一

2009 年 2 月 21 日

1 Introduction

上川 純一

今月の Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか？

Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での「開発に関する情報」を整理してまとめ、アップデートする。
- 場 の提供。
  - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
  - Debian のためになることを語る場を提供する。
  - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、「場」としての空間を提供するのが目的です。

2009 年の計画は仮です。

1. 新年の企画 (アンサンブル荻窪開催)
2. OSC
3. (東京大学?)
4. (千代田区都立図書館?[*1])
5. (東京大学?)
6.
7. スペインにて開催
8. Debconf 報告会
9.
10.
11.
12. 忘年会

会場候補としては下記があります:

- 大学
- 恵比寿 SGI ホール
- Google オフィス
- 公民館 (あんさんぶる荻窪等)
- 都立会議室 (無線 LAN)
- 健保の施設

[*1] http://www.library.chiyoda.tokyo.jp/

デビアン勉強会

目次

2 最近の Debian 関連のミーティング報告

上川 純一

2.1 東京エリア Debian 勉強会 47 回目報告

東京エリア Debian 勉強会報告。2009 年 1 月 17 日土曜日に東京エリア Debian 勉強会の第 48 回を開催しました。

今回の参加者は id774 さん、あけどさん、前田耕平さん、小室文さん、山本浩之さん、岩松信洋さん、やまだたくまさん、じつかたさん、キタハラさん、小林儀匡さん、藤沢理聡さん、上川の 12 名でした。

クイズについては、今回は小林さんが当初不在だったのとクイズを用意していなかったのでキャンセルです。何か別の企画を期待したいところです。

最初に最近のイベントの紹介をしました。前回の勉強会のおさらいと、Debian JP で行った IAX 会議について紹介しました。IAX での音声会議に興味をそそられた人もいたようです。

`http://pspunch.com/pd/article/asterisk_meetme_ja.html` にて今回利用した設定が紹介されています。

まず最初に事前課題を紹介しました。

2009 年にどういう内容を実施するのかについて議論しました。ブレインストーミングをしてそのあと 2009 年の毎月の計画をたてました。今年は無事にできるかな?

最後に冬休みの宿題を発表しあって終了。

上川は Git format-patch を利用したワークフローでいかにコンフリクトを発生させないかを紹介しました。前田さんは MacBook に Lenny をインストールしなおしたときにはまった内容。小室さんは、Google Ajax API の紹介。id774 さんは Aspire One にインストールしたときの話。山本浩之さんは 2ch ビューアーパッケージについて。岩松さんは Linux Kernel の .config 自動生成ツールについての紹介でした。

今回は会費は赤字。宴会は荻窪卯にて。Sony Type P をもってきたやまださんをかこんでわいわいと。

3 Debian パッケージングハンズオンの手引書

3.1 本日の目的

Debian パッケージ化されていないソフトウェアをパッケージ化して、ビルドテストとパッケージの変更までを体験します。ところどころにトラップがあるので注意しましょう。

3.2 本日の流れ

1. 講師紹介
2. 作業を始める前の前準備
3. ソフトウェアのコンパイル
4. パッケージの雛形
5. CDBS
6. debian ディレクトリ以下ファイルの編集
7. パッケージのビルド
8. パッケージのインストール
9. パッケージのビルドテスト
10. パッケージのインストール/アンインストールテスト
11. プログラムの編集
12. 質疑応答

3.3 記号の説明

$** が付いている場合は、コンソールからの入力を意味します。**$は入力せずにコマンドを入力してください。

コマンドラインやファイルの中身で \ が書かれている場所は行が続いている事を意味します。入力しないでください。

... は省略を意味します。実際には長い出力がある場合に省略している場合に利用しています。

3.4 エディタ

本ハンズオンでは、エディタとして **vi** および **mousepad** を使えるようにしています。**vi** が使えない人は、**mousepad** を使ってください。Windows のメモ帳と同じ機能を持ったエディタです。

3.5 ルート権限について

本ハンズオンでは、root 権限を使った作業を行う場合があります。その場合には sudo コマンドを使って作業をします。**sudo** コマンドが必要な場合にはコマンドラインの説明のところに **sudo** を指定しています。

3.6 前準備

3.6.1 パッケージメンテナ名の設定

パッケージメンテナの名前とメールアドレスを環境変数に設定します。適当なでエディタを使って、**/home/user/.bashrc** に以下の例のように変更して保存してください。各項目には自分の名前とメールアドレスをいれてください。

```
export DEBFULLNAME="Nobuhiro Iwamatsu"
export DEBEMAIL=iwamatsu@nigauri.org
```

保存できたら、ターミナルを起動し、

```
$ source ~/.bashrc
```

を実行してください。

3.6.2 web サーバの立ち上げ

コンソールから以下のコマンドを実行してください。これは Live-CD 環境で apt-get ができるようにするための対策として行っています。実際のパッケージ作成では必要ありません。

```
$ sudo ruby1.8 ./tools/web.rb
```

3.6.3 apt-line の変更

エディタを使い、**/etc/apt/sources.list** ファイルを以下のように変更してください。apt-line が書かれていますが、削除してください。

```
deb http://localhost/debian lenny main
```

3.6.4 リポジトリ情報のアップデート

リポジトリのアップデートを行います。以下のようにコマンドを実行します。

```
$ sudo apt-get update
```

3.6.5 /tmp のマウントオプションの変更

/tmp を **nodev** オプションなしで **remount** します。

以下のように実行します。

```
sudo mount -o remount,dev /tmp
```

3.7 今回のサンプル

今回は、**cwidget** を使ったサンプルプログラム **/live/image/osc/data/hello-cwidget-0.1.tar.gz** を用意しました。このサンプルプログラムを Debian パッケージ化します。**/live/image/osc/data** ディレクトリにソースファイルがあるので、ホームディレクトリに展開します。

```
$ cd
$ tar -xzf /live/image/osc/data/hello-cwidget-0.1.tar.gz
```

このソフトウェアは C++ で記述されており、コンパイルに必要なライブラリやソフトウェアがインストールされている場合には、./configure ; make ; make install でコンパイルおよびインストールまでができるようになっています。

3.8 パッケージング化開始

3.8.1 ソースを読んでみる

動作しないプログラムをパッケージ化してもしょうがないので、先にどのようなソフトウェアなのか理解するためにもパッケージング化する前にソースコードを読んで、ソフトウェアの中身を理解して置きましょう。

3.8.2 とりあえず、コンパイルしてみる

動かないプログラムをパッケージ化してもしょうがないので、動作確認をします。まずは最低限コンパイルに必要なパッケージをインストールする必要があります。それが **build-essential** パッケージです。これは、パッケージ化の場合にも必要です。以下のように実行し、インストールします。

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

先ほど解凍したディレクトリに移動します。移動したら、**configure** を実行します。

```
$ cd hello-cwidget-0.1
$ ./configure
...
Alternatively, you may set the environment variables \
SIGC_CFLAGS
and SIGC_LIBS to avoid the need to call pkg-config.
See the pkg-config man page for more details.
...
```

実行すると、エラーになります。

3.9 必要なライブラリを探す

Debian で特定のファイルが提供されているパッケージを探す場合には、**apt-file** を利用します。以下のように実行し、インストールします。

```
$ sudo apt-get install apt-file
```

通常は、 この後、**apt-file update** を実行し、ファイル情報データを取得しますが、既に Live-CD に入れているので省略します。ファイルを探すには以下のように実行します。

```
$ apt-file search pkg-config
...
nant: /usr/share/doc/nant/help/functions/pkg-config.\
     is-max-version.html
pkg-config: /usr/bin/pkg-config
pkg-config: /usr/share/doc/pkg-config/AUTHORS
...
```

実行すると、指定したファイルを提供しているパッケージ名が出力されます。出力されたパッケージをインストールします。

```
$ sudo apt-get install pkg-config
```

再度 **configure** を実行してみましょう。

```
$ ./configure
...
No package 'sigc++-2.0' found

Consider adjusting the PKG_CONFIG_PATH environment variable if you
installed software in a non-standard prefix.
...
```

まだ足りないパッケージがあるようです。先ほどと同じように **apt-file** を利用して検索し、インストールします。

```
$ apt-file search sigc++-2.0.pc
libsigc++-2.0-dev: /usr/lib/pkgconfig/sigc++-2.0.pc
$ sudo apt-get install libsigc++-2.0-dev
```

再度 configure を実行します。

```
$ ./configure
...
checking for CWIDGET... configure: error: Package \
  requirements(cwidget) were not met:

No package 'cwidget' found

Consider adjusting the PKG_CONFIG_PATH environment variable \
if you
installed software in a non-standard prefix.
...
```

エラーになります。まだ足りないようなので、再度検索してインストールします。

```
$ apt-file search cwidget.pc
libcwidget-dev: /usr/lib/pkgconfig/cwidget.pc
$ sudo apt-get install libcwidget-dev
```

```
./configure
...
config.status: WARNING:  Makefile.in seems to ignore the \
   --datarootdir setting
config.status: creating src/Makefile
config.status: WARNING:  src/Makefile.in seems to ignore the \
  --datarootdir setting
config.status: creating config.h
```

configure が正常に終了しました。終了すると、**Makefile** が作成されています。**make** を実行し、コンパイルします。

```
$ make
...
make[1]: ディレクトリ ‘/home/user/hello-cwidget-0.1’ \
に入ります
Making all in src
make[2]: ディレクトリ ‘/home/user/hello-cwidget-0.1/src’ \
に入ります
g++ -DHAVE_CONFIG_H -I. -I. -I..     -g -O2 -I/usr/ \
include/sigc++-2.0 \
-I/usr/lib/sigc++-2.0/include   -I/usr/lib/cwidget
 -I/usr/include/sigc++-2.0  -I/usr/lib/sigc++-2.0/include \
 -c hello.cc
g++  -g -O2 -I/usr/include/sigc++-2.0 -I/usr/lib/ \
sigc++-2.0/include   \
-I/usr/lib/cwidget -I/usr/include/sigc++-2.0
-I/usr/lib/sigc++-2.0/include \
-o hello  hello.o  -lsigc-2.0   -lcwidget -lncursesw \
-lsigc-2.0
make[2]: ディレクトリ ‘/home/user/hello-cwidget-0.1/src’ \
から出ます
make[2]: ディレクトリ ‘/home/user/hello-cwidget-0.1’ に入ります
make[2]: ディレクトリ ‘/home/user/hello-cwidget-0.1’ から出ます
make[1]: ディレクトリ ‘/home/user/hello-cwidget-0.1’ から出ます
```

コンパイルも正常に終了したので、試しに実行してみます。

```
$ ./src/hello
```

ここまではサンプルプログラムの動作確認です。動作しないプログラムをパッケージ化してもしょうがないので、先にどのようなソフトウェアなのか理解するためにもパッケージング化する前にソースコード等を読んでおくことをお勧めします。

3.10 Deban パッケージの雛形

dh_make コマンドでパッケージの雛形を作成することができます。**dh_make** は、**dh-make** パッケージで提供されています。以下のコマンドを実行し、インストールします。

```
$ sudo apt-get install dh-make
```

雛形の作成は以下のコマンドを実行します。

```
$ dh_make --createorig -s
```

–**createorig** オプションはオリジナルソースコードの tar.gz イメージを構築します。 今回はシングルバイナリパッケージ（一つのソースコードから一つのバイナリパッケージが作成される）なので**-s** を指定します。実行すると以下のようなメッセージが表示されるので、Enter キーを押します。

```
Maintainer name : Nobuhiro Iwamatsu
Email-Address   : iwamatsu@nigauri.org
Date            : Sun, 15 Feb 2009 23:51:58 +0900
Package Name    : hello-cwidget
Version         : 0.1
License         : blank
Using dpatch    : no
Using quilt     : no
Type of Package : Single
Hit <enter> to confirm:
```

3.10.1 debian ディレクトリ

うまく動作すると、**debian ディレクトリ**が作成され、この中に雛形が作成されます。パッケージメンテナはこのディレクトリの中以外は触りません。以下のような状態になっています。

```
.
|-- README.Debian  (Debian パッケージの README)
|-- changelog      (Debian パッケージのチェンジログ)
|-- compat         (Debian パッケージのバージョン)
|-- control        (Debian パッケージ情報)
|-- copyright      (コピーライト情報)
|-- cron.d.ex      (cron を使うパッケージ用設定ファイル)
|-- dirs           (作成するディレクトリ名を指定する)
|-- docs           (インストールするドキュメントファイルを指定する)
|-- emacsen-install.ex (emacs 用設定ファイル)
|-- emacsen-remove.ex  (emacs 用設定ファイル)
|-- emacsen-startup.ex (emacs 用設定ファイル)
|-- hello-cwidget.default.ex (debfonf 用)
|-- hello-cwidget.doc-base.EX (doc-base 用)
|-- init.d.ex      (init.d を使うパッケージ用設定ファイル)
|-- init.d.lsb.ex  (init.d を使うパッケージ用設定ファイル)
|-- manpage.1.ex   (manpage の雛形)
|-- manpage.sgml.ex(manpage の雛形)
|-- manpage.xml.ex (manpage の雛形)
|-- menu.ex        (メニューの雛形)
|-- postinst.ex    (postinst メンテナファイルの雛形)
|-- postrm.ex      (postrm メンテナファイルの雛形)
|-- preinst.ex     (preinst メンテナファイルの雛形)
|-- prerm.ex       (prerm メンテナファイルの雛形)
|-- rules          (パッケージビルドスクリプト)
`-- watch.ex       (アップストリームチェック用ファイル)
```

3.11 CDBS

./configure ; make ; make install でパッケージのコンパイルができるソフトウェアは cdbs を使った方が容易に Debian パッケージ化できます。

3.11.1 一回 hello-cwidget ディレクトリを削除する

現状では先ほどの **dh_make** の結果が残っているので一回、サンプルプログラムのディレクトリごと削除し、再度展開します。

```
$ cd
$ rm -rf hello-cwidget-0.1.*
$ tar -xzf /live/image/osc/data/hello-cwidget-0.1.tar.gz
$ cd hello-cwidget-0.1
```

3.11.2 dh_make を実行し、パッケージの雛形を作成する

CDBS を使う Debian パッケージの雛形作成は以下のコマンドを実行します。

```
$ dh_make --createorig -b
```

-b オプションを指定すると、CDBS を使った雛形を作成します。以下のようなメッセージが表示されるので、エンターキーを押します。

```
Maintainer name : Nobuhiro Iwamatsu
Email-Address   : iwamatsu@nigauri.org
Date            : Sun, 15 Feb 2009 23:51:58 +0900
Package Name    : hello-cwidget
Version         : 0.1
License         : blank
Using dpatch    : no
Using quilt     : no
Type of Package : cdbs
Hit <enter> to confirm:
```

3.11.3 不要なファイルの削除

今回のパッケージ化に必要ではないファイルを **debian** ディレクトリ以下から削除します。

```
$ rm -rf debian/*.ex debian/*.EX
```

3.11.4 debian/changelog ファイルの編集

debian/changelog ファイルには **ITP**(Intent To Package) のバグが既に書かれているで削除します。以下のように変更します。

```
hello-cwidget (0.1-1) unstable; urgency=low

  * Initial release.

 -- Nobuhiro Iwamatsu <iwamatsu@nigauri.org> \
                Wed, 18 Feb 2009 16:31:25 +0000
```

3.11.5 debian/copyright ファイルの編集

```
This package was debianized by Nobuhiro Iwamatsu \
                              <iwamatsu@nigauri.org> on
Wed, 18 Feb 2009 16:31:25 +0000.

It was downloaded from <http://www.nigauri.org/~iwamatsu/>

Upstream Author:

    Nobuhiro Iwamatsu <iwamatsu@nigauri.org>

Copyright:

    Copyright (C) 2009 Nobuhiro Iwamatsu <iwamatsu@nigauri.org>

License:

    GPLv2

The Debian packaging is (C) 2009, Nobuhiro Iwamatsu \
        <iwamatsu@nigauri.org> and
is licensed under the GPL, see '/usr/share/common-licenses/GPL'
```

3.11.6 debian/control ファイルの編集

```
Source: hello-cwidget
Section: devel
Priority: extra
Maintainer: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
Build-Depends: cdbs, debhelper (>= 7), autotools-dev
Standards-Version: 3.8.0
Homepage: http://www.nigauri.org/~iwamatsu/

Package: hello-cwidget
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: Debian Packaging Hands-on sample program
 This is sample program of Debian Hands-on done with
 OSC2009 TOKYO Spring.
 This is very easy program that uses CWidget.
```

3.11.7 パッケージのビルド

パッケージのビルドには **debuild** コマンド を使います。debuild コマンドは **devscripts** パッケージで提供されています。また、まだ **CDBS** パッケージをインストールしていないので、一緒にインストールします。パッケージをインストールしたら、パッケージのビルドをしてみましょう。

```
$ sudo apt-get install devscripts cdbs
$ debuild -us -uc
...
dpkg-buildpackage: full upload (original source is included)
Now running lintian...
W: hello-cwidget: binary-without-manpage usr/bin/hello
W: hello-cwidget: new-package-should-close-itp-bug
Finished running lintian.
```

3.12 パッケージのインストール

パッケージが無事ビルドできたら、実際にインストールしてみます。インストールには dpkg コマンドを使ってインストールします。インストールしたら、実際に動くか確認してみましょう。

```
$ sudo dpkg -i ../hello-cwidget_0.1-1_i386.deb
$ which hello
$ hello
```

3.13 パッケージのビルドテスト

パッケージができたあとにはパッケージのテストを行います。パッケージのビルドテストには **pbuilder** を使います。pbuilder は Debian に必要な最低限の環境からビルドを行い、依存関係等のチェックを行ってビルドテストを行うツールです。

3.13.1 pbuilder パッケージのインストール

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

3.13.2 pbuilder 環境の構築

ビルドテストを行う前に base システムイメージを構築する必要があります。通常は以下のように実行しますが、

```
$ sudo pbuilder --create --distribution lenny
```

今回はメモリの制限があるため、既に用意してある base システムイメージを利用します。イメージは **/live/image/osc/data/base.tgz** にあります。

3.13.3 パッケージのビルドテスト

pbuilder でテストする場合には作成されたパッケージの **dsc** ファイルを指定します。このファイルには、Debian パッケージの構成に必要なファイル名が書かれているので、その情報を元に再ビルドを行うことができます。また、実行前に **apt-get clean** コマンドを実行してキャッシュをクリアしてください。メモリが足りないためです。

```
$ cd ..
$ sudo apt-get clean
$ sudo pbuilder --build --distribution lenny \
     --basetgz /live/image/osc/data/base.tgz \
     --buildplace /tmp hello-cwidget_0.1-1.dsc
...
```

3.13.4 なぜエラーになるのか

先ほどの手順でやってもビルドエラーになります。なぜエラーになるのでしょうか。考えてみましょう。

3.13.5 再ビルドテスト

エラーになる理由は先にインストールしたパッケージ **libcwidget-dev** をパッケージビルド時の依存関係を記述するフィールド **Build-Depends** に追加していないためです。追加して、再ビルドしてみます。再ビルドには以下のように実行します。今度はうまくビルドができるはずです。

```
$ sudo pdebuild -- --distribution lenny --basetgz \
 /live/image/osc/data/base.tgz --buildplace /tmp
```

3.14 パッケージのインストール/アンインストールテスト

パッケージがビルドできただけでは喜んではいけません。インストール/アンインストールのテストも行いましょう。パッケージのインストール/アンインストールのテストには **piuparts** パッケージを使います。

3.14.1 piuparts のインストール

以下のように実行し、インストールします。

```
$ sudo apt-get install piuparts
```

3.14.2 パッケージのインストール/アンインストールテスト

piuparts も pbuilder と同様に最低限の環境からのインストールをチェックします。よって、base システムイメージが必要です。普段は指定する必要はありませんが、今回は**-b** オプションを付けて、**/live/image/osc/data/base.tgz** にある base システムイメージを指定して実行します。

```
$ cd ..
$ sudo piuparts -d lenny -b /live/image/osc/data/base.tgz \
   hello-cwidget_0.1-1_i386.deb
...
0m41.9s DEBUG: Removed directory tree at /tmp/tmpHliOKO
0m41.9s INFO: PASS: All tests.
0m41.9s INFO: piuparts run ends.
```

3.15 プログラムの編集

hello-cwidget を実行して、違和感のある方がおられたと思います。そう、**Lenny** がリリースされたというのに **Etch** になっていました。これはよくないので変更してみます。今回はよく利用されている **dpatch** を使って説明します。

3.15.1 dpatch のインストール

dpatch をインストールするには、以下のように実行します。

```
$ sudo apt-get install dpatch
```

3.15.2 dpatch を使うための準備

dpatch を使う前に、**debian/rules** ファイルに dpatch を使うように設定する必要があります。dpatch は一回、パッケージの状態を初期化してから行うためです。**hello-cwidget-0.1** ディレクトリに移動して、**debian/rules** を以下のように修正します。

```
$ cd  hello-cwidget-0.1
```

```
#!/usr/bin/make -f

include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/autotools.mk
include /usr/share/cdbs/1/rules/dpatch.mk
include /usr/share/dpatch/dpatch.make
```

3.15.3 dpatch の実行

dpatch は自パッケージを一回コピーし、dpatch 環境に移行します。その中で変更して、dpatch 環境を終了する時に差分を作成します。dpatch 環境に移行するには **dpatch-edit-patch** コマンドに作成する差分を保存するファイル名を指定して実行します。以下のように実行してください。

```
$ dpatch-edit-patch 01_change_dist
```

3.16 ファイルの変更

今回変更するファイルは **src/hello.cc** です。エディタを起動し、対象のファイルを変更します。mousepad の場合は以下のように実行します。

```
$ mousepad ./src/hello.cc
```

Etch の部分を **Lenny** に変更したあと、保存してエディタを終了します。

3.16.1 dpatch 環境を終了する

dpatch 環境を終了するには以下のように実行してください。実行すると、差分をファイルに保存して dpatch 環境を終了します。

```
$ exit
```

3.16.2 作成された差分 (patch) の中身

作成された差分は **debian/patches/01_change_dist.dpatch** として保存されています。以下のような内容になっているはずです。

```
#! /bin/sh /usr/share/dpatch/dpatch-run
## 01_change_dist.dpatch by Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
##
## All lines beginning with `## DP:' are a description of the patch.
## DP: No description.

@DPATCH@
diff -urNad hello-cwidget-0.1~/src/hello.cc hello-cwidget-0.1/src/hello.cc
--- hello-cwidget-0.1~/src/hello.cc 2009-02-15 06:56:01.000000000 +0000
+++ hello-cwidget-0.1/src/hello.cc  2009-02-18 16:54:40.668274925 +0000
@@ -26,7 +26,7 @@
    toplevel::init();

    widgets::widget_ref dialog =
-       dialogs::ok(L"Hello, Debian GNU/Linux Etch!",
+       dialogs::ok(L"Hello, Debian GNU/Linux Lenny!",
            util::arg(sigc::ptr_fun(toplevel::exitmain)));

    toplevel::settoplevel(dialog);
```

パッチにはなぜそのような説明をしたのか、説明を書く必要があります。**## DP: No description.** の部分に説明を書きます。以下のように変更するといいかもしれません。

```
## DP: Change distributin name from Etch to Lenny.
```

3.16.3 作成した差分をパッケージに反映させる

差分は作成されましたが、このままではパッケージ作成時に差分が適用されません。dpatch を使って差分をパッケージに適用させるには **debian/patches/00list** ファイルを作成し、パッケージにパッチをファイルに列挙する必要があります。**debian/patches/00list** を以下のように変更します。

```
01_change_dist.dpatch
```

3.16.4 差分を適用したパッケージを作成する

差分を適用したパッケージを作成するには通常のパッケージ作成と変わりません。**debuild** コマンドを使って作成します。

```
$ debuild -us -uc
....
```

3.16.5 パッケージ作成エラーになる

説明どおりに操作している人は、パッケージ作成エラーになると思います。理由は何なのか、考えてみましょう。原因が分かった人は、再ビルドした後に、実際にインストールして、差分が反映されているか確認してください。もちろん **pbuilder**/**piuparts** を使ってパッケージのテストを行う事も忘れずに。

3.17 質疑応答

以上で、本ハンズオンは終了です。何か質問等はありますか？

Debian 勉強会資料

2009 年 2 月 21 日　　初版第 1 刷発行
東京エリア Debian 勉強会 （編集・印刷・発行）