第 83 回 東京エリア Debian 勉強会資料

月刊 Deb専

日本唯一のDebian専門月刊誌

2011 年 12 月 17 日

特集1: 2011年の振り返り

特集2: quiltでportingしてみた

特集3: 月刊Debhelper

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?

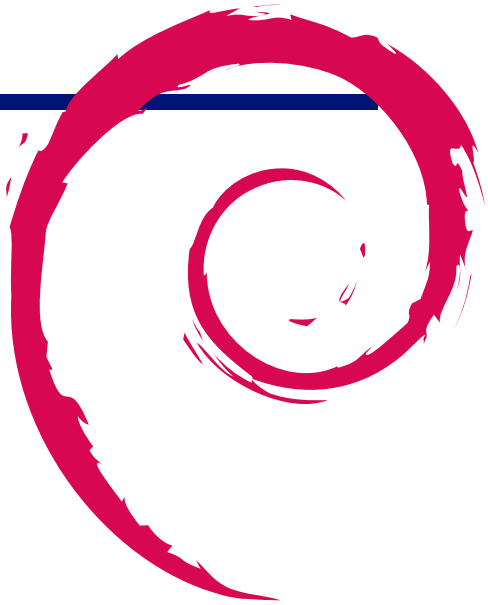
Debian 勉強会の目的は下記です。

- 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。

デビアン勉強会

目次



2 事前課題

野島 貴英

今回の事前課題は以下です:

1. 今年パッケージ化した、あるいは、使い込んだ Debian パッケージを 5 つまであげて、そこで出会った課題について 200 文字以内に解説してください。

この課題に対して提出いただいた内容は以下です。

2.1 吉野 (yy_y_ja_jp)

Twitter クライアント Polly を Ubuntu PPA が微妙だったのでパッケージ化してみました.

アイコンが Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported だったので削除しました. 自前で描くべきか微妙です。コードの多くが GPL-3+ で、一部アイコンセットが GPL-2 となっていて矛盾しているのでそのアイコンセットを削除しました。 Python で書かれており、 Python ライブラリを Build-Depends−*Indep* と Depends に手書きで追加しました。 Depends は自動生成できたらうれしいかもしれません。

2.2 dictoss(杉本 典充)

パッケージ化したものはないので、使い込んだ Debian パッケージだけです。

- kfreebsd-image-8-amd64
  FreeBSD 本家ではビルドされるのに、 Debian パッケージでは付属していないものが多く、その違いに悩んだ。
- gftp
  ssh コマンドみたくデフォルトの ssh 鍵認証→パスワード認証でログイン処理が進んでくれず、 ssh 鍵認証でログインに失敗すると画面が進んでくれない。アップストリームの問題かは不明。
- icewm

2.3 岩松 信洋

- 今年パッケージ化した Debian パッケージ:
  bluez-tools, fonts-ipamj-mincho, mozc, mtdev, xf86-input-mtrack, xf86-input-multitouch
- 今年使い込んだ Debian パッケージ:
  libpng, opencv, bluez, ruby, mozc, buildd, pbuilder
- 出会った課題: release チームとのやりとり

2.4 なかおけいすけ

- パッケージ化しているパッケージ
  stm32flash
- 課題
  man がないとか、ライセンスファイルの書き方がおかしいとか、パッケージングポリシーに従うこと、意外に大変なこと。

2.5 やまだ

Debian 構成や固有部分が絡んでるもので挙げてみました:

- busybox
  OK:ビルド設定が不足して job control できない
- initramfs-tools
  ??:rootwait があるから rootdelay は不要と思ったら MD+USB で起動障害
  - Linux 的には上は正だが、 udev も見てタイミング調整するというクイックハック状態なのだった
- grub-pc
  OK:grub2 でモニタとシリアルの同時有効化機能が一時落ちて、制御不能になった
- extlinux
  OK:設定バラバラ化のみならず、標準ではシリアル有効化の設定を仕込む箇所が存在せず、制御不能に
- dropbear
  ??:initramfs 内で dropbear を起動する荒業により、固

定 IP の DNS サーバが DHCP アドレスになり NW 死亡

- debhelper
  ??:CFLAGS（ や相当設定）の指定方法がわからず、 gcc -m32 や -f... の渡し方に悩む
- kexec
  OK:デフォで再起動が単なる kexec 実行に切り替わり、 BIOS が取れず悩んだ
- apt
  ??:実は security.d.o が 120ms-300ms の彼方で遠い・・・

2.6 Kazuo Ishii

emacs
常にカスタマイズを続けています。これをどれだけ使いこなすかが、課題です。

2.7 Aru

postfix のソースコードをいじってパッケージ化させてインストールしなおしました。

○ CN の回線上で SMTP サーバを構築する際は OP25B の関係上 OCN の SMTP サーバを通さなければならないのですが、その SMTP サーバが特殊な仕様のようで postfix の設定を変えるだけではサーバに弾かれてしまいます。そこで、○ CN のサーバ向けにソースコードをいじりました。

2.8 koedoyoshida

- Unbound:
  仕事で評価用 DNS を建てるのに使ったり、 dnstudy 等で発表の危険な LT ネタとして使用。 BIND に比べて良くできてるので特に課題等はなし
- zabbix:
  基本 stable を使っていること、かつ発展途上のソフトであることから豊富なはまりポイントが有った。が基本的にぐぐって解決したり、表示上の問題なので無視したりしてます。

2.9 henrich

今年はあまりパッケージをいじくってないです…パッケージ化したものといえば IRC クライアントの loqui とかぐらいでしょうか。それも upstream の方が、ほとんど雛形作っていただいていたので細かい所を直しただけでした。

DEP5 への対応や他の DEP への対応などが今後パッケージクオリティを挙げる上で課題でしょうか(主に面倒くさい意味で)

2.10 yamamoto

今年パッケージ化した Debian パッケージ: 無しうーむ、自分専用のパッチをあてたカーネルパッケージぐらいっすね。貢献はできてないな。

- 今年使い込んだ Debian パッケージ: devscripts pbuilder
  devscripts でポチポチと sbuild 用にパッケージを作り、 pbuilder で再ビルドしてました。今年は FTBFS を直してもらう BTS をたまにするのと、自分専用リポジトリを最新に保つだけで、いっぱいいっぱいでした。
- 来年使い込む予定の Debian パッケージ: sbuild buildd さて、弾ができたし、突撃じゃー。
- 出会った課題:
  base.tgz とかを最初に作る時、 arch=all なのはオフィシャルから、アーキテクチャ依存なパッケージはオレオレリポジトリから、という使い方ができなかった。ヘタレなおいらは、 build-essential なパッケージに依存する all なのは、全部オレオレリポジトリにつっこんで解決。

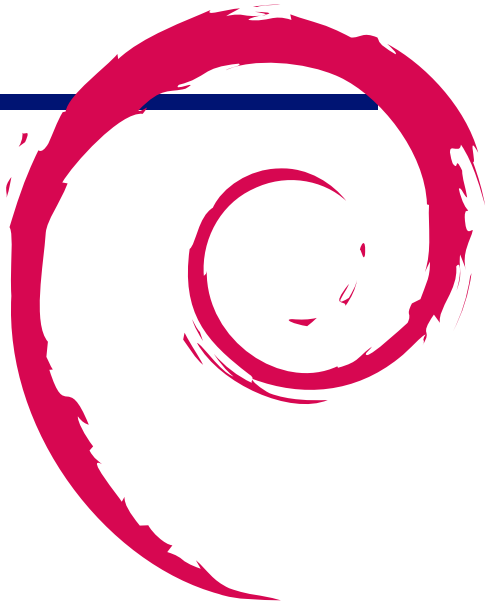
2.11 まえだこうへい

12/8 時点の状況

- パッケージング済み
  - python-funcparserlib : テストでコケる問題
  - python-webcolors
- パッケージング途中
  - python-ordereddict： Python2.6 のみ
  - python-blockdiag : Python 2.6 は python-ordereddict を、 2.7 は組み込みの ordereddict を使うようにパッケージングするには debian/control どう書けばいいんだ?
  - python-{sec,act,nw}diag および python-sphinxcontrib.{block,seq,act,nw}diag : python-blockdiag 待ち
  - python-tomahawk : nodetests でコケる問題

2.12 野島 貴英

- パッケージング途中
  - xmris
    score の others に rw 付与を止めさせたい。古いゲームは特に。
  - tracef
    dynamic loading を hook して、 symbol 再ロードして欲しい。
- 使い込んだパッケージ
  - totem,gstreamer-tools
    いくつか機能の動作が不完全な気が。例: 字幕ファイル利用とか。
  - gnome-shell
    いろいろドキュメントなさすぎ。
  - po-mode.el
    改行全部に \n を入れられてしまうので、 po4a 製 po ファイルの翻訳作業がきっつー。



3 最近の Debian 関連のミーティング報告

野島 貴英

3.1 東京エリア Debian 勉強会 81 回目報告

10 月の東京エリア Debian 勉強会は、筑波大学にて開催されました。 Debian をよく知らない学生の方々が参加される可能性があったため、若い人向けに Debian についておさらいの意味で、 Debian とは何かについて岩松さんが発表しました。この発表をきっかけに若い方々で Debian に興味を持っていただき、ますます Debian Developer 候補が増えて欲しいと思います。

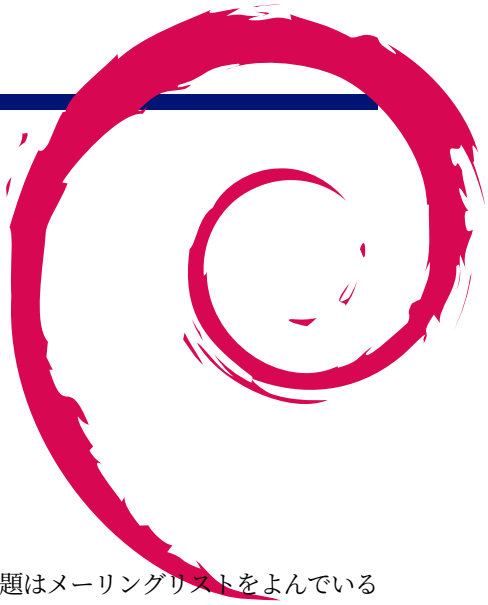
また、岡部さんには Debian 上で Haskell の開発について現状・問題点を説明していただきました。関数型言語の代表的な言語の一つである Haskell を Debian でやってみようという方が増えると良いなぁと思っています。坂口さんには、Debian 上での LaTeX 環境についての紹介と、学校の課題を自動生成する事について発表していただきました。坂口さんは学生ということで、今の学生さんがどのように LaTeX 環境と付き合っているのかについて非常に参考になりました。

最後に岩松さんから月刊 Debhelper の企画と発表がありました。 Debian のパッケージを開発する際に強力なツールである Debhelper のコマンドについて、毎月 2 コマンド以上持ち回りで説明が行われる事になりました。きっとこれで、Debian パッケージ開発者が益々増えると良いなぁと思います。

3.2 東京エリア Debian 勉強会 82 回目報告

11 月の東京エリア Debian 勉強会は OSC Tokyo/Fall にてセッション形式で行いました。場所は明星大学、第一日目の 11/19(土) の 11:00-11:45 にて行っています。 Debian JP Project の活動の宣伝と、最近の Debian 事情について岩松さんより発表していただきました。最近の Debian 事情について、これほどまとまった内容はないんじゃないかと思うほどの内容でした。

また、 OSC Tokyo/Fall では 11/19〜11/20 の 2 日間ブースも出しています。 LiveDVD の配布も行いました。11/19 はあいにくの雨にも関わらず多くの人が訪れたようです。これで Debian ユーザがもっともっと増える事を期待しています。



4 Debian Trivia Quiz

野島 貴英

ところで、みなさん Debian 関連の話題においついていますか? Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容と Debian Project News からです。

問題 1. 11 月終わり頃にルートファイルシステムの構造について議論を呼んでます。内容は?

☐ A /user を作る

☐ B /bin,/sbin,/lib の実体を/usr 以下に移動して、代わりにシンボリックリンクにする

☐ C /etc の実体を/usr 以下に移動して、代わりにシンボリックリンクにする

問題 2. sun-java6 が Debian パッケージとして配布できなくなりました。代わりに Debian で推奨される Java は?

☐ A openjdk

☐ B gcj-jdk

☐ C coco-java

問題 3. 11/19 に長らく活動を停止していたパッケージチームが復活宣言をしました。どれでしょう?

☐ A CORBA packaging team

☐ B Ham-radio packaging team

☐ C SDL packaging team

問題 4. 10/28〜30 で MiniDebconf2011 が開かれました。どこの国でしょう?

☐ A ニカラグア

☐ B インド

☐ C フランス

問題 5. Wheezy フリーズの為の BSP が各国で開かれました。ドイツとどこ?

☐ A フランス

☐ B ニカラグア

☐ C ポーランド

5　2011 年の振り返り

まえだこうへい

今月で 7 年目の Debian 勉強会が終了しました。

5.1　基本的な数値

Debian 勉強会は毎回事前課題事後課題を設定しており、予習復習を必要だと謳っている勉強会です。実際にどれくらいの人が出席しているのか、またその人たちがどれくらい事前課題・事後課題を提出しているのか、確認してみましょう。図 1 です。値は一年の移動平均です。

結果を見ると参加者数は下降傾向にあり、事前課題の提出率は昨年からは横ばい傾向です。事前課題をちゃんとやる常連参加者に収斂されてきているのでしょうか? 一方、事後課題 (ブログ) の率はさらに低下しています。昨年、「ブログはもう流行らないのでしょうか」との一言がありましたが、まさにその通りかもしれません。

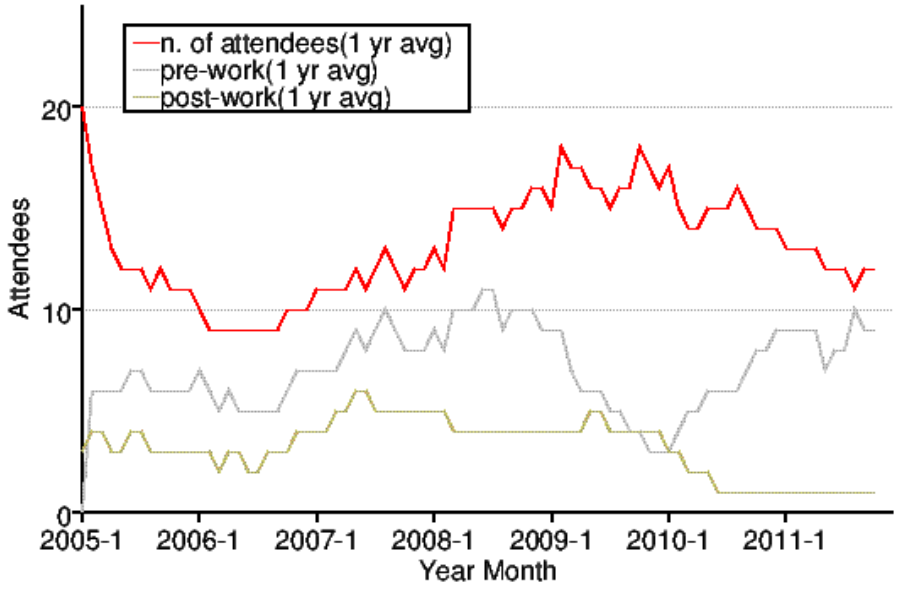


図 1　東京エリア Debian 勉強会事前課題・事後課題提出実績 (12 ヶ月移動平均)

毎回の参加者の人数と、その際のトピックを見てみます。今年も昨年同様、人数的には大きく増減はありませんが、10 月の筑波大での開催での参加者が、昨年の筑波大でのつくらぐさんとの合同勉強会の時と同じ参加者数でした。新参の参加者を期待した筑波大生の参加は思ったほどではなかった一方、初参加なのにわざわざ都心から筑波大まで参加してくれた方が意外と多かった回でした。2009 年から見ると、大学での開催はいつもより参加者数および初参加者が増えるので、来年以降も続けていくと良いでしょう。また、リストをみると、毎月数名は初参加者も毎月いて、そのうち一部は 2 回目以降も参加する人がいることを考えると、先ほどの参加数が下降傾向にありつつ、事前課題の提出数が横ばいであるのは、ちゃ

んと事前課題を提出する人が常連になる傾向にあるのではないか、とも見れます。事前課題の提出率は維持しつつ、参加者数は増やしていく為の対策は必要でしょう。

会場を見ると、今年の会場は、あんさんぶる荻窪以外の公民館などの利用が増えました。また、今年は 3 年前の草津温泉、昨年の木更津に続き、三回目の Debian 温泉を伊東の山喜温泉で開催しました。 Debian Hack Cafe も 6 月ごろから月 1 回程度のペースで再開しているようです。

表 1　東京エリア Debian 勉強会参加人数 (2005-2006 年)

	参加人数	内容
2005 年 1 月	21	秘密
2005 年 2 月	10	debhelper 1
2005 年 3 月	8	(早朝) debhelper 2、social contract
2005 年 4 月	6	debhelper 3
2005 年 5 月	8	DFSG、 dpkg-cross、lintian/linda
2005 年 6 月	12	alternatives、 d-i
2005 年 7 月	12	toolchain、 dpatch
2005 年 8 月	7	Debconf 参加報告、ITP からアップロードまで
2005 年 9 月	14	debconf
2005 年 10 月	9	apt-listbugs、 バグレポート、 debconf 翻訳、debbugs
2005 年 11 月	8	DWN 翻訳フロー、 stateoverride
2005 年 12 月	8	忘年会
2006 年 1 月	8	policy、 Debian 勉強会でやりたいこと
2006 年 2 月	7	policy、 multimedia
2006 年 3 月	30	OSC: debian 勉強会、sid
2006 年 4 月	15	policy、 LaTeX
2006 年 5 月	6	mexico
2006 年 6 月	16	debconf、 cowdancer
2006 年 7 月	40	OSC-Do: MacBook Debian
2006 年 8 月	17	13 執念
2006 年 9 月	12	翻訳、 Debian-specific、 oprofile
2006 年 10 月	23	network、 i18n 会議、Flash、 apt
2006 年 11 月	20	関西開催: bug、 sid、packaging
2006 年 12 月	14	忘年会

表 2　東京エリア Debian 勉強会参加人数 (2007-2008 年)

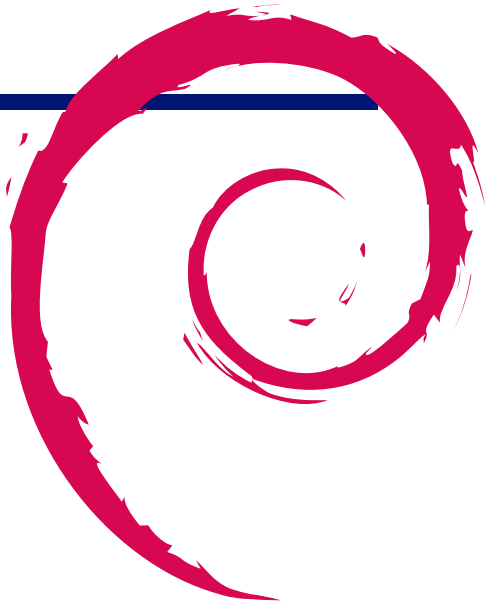
	参加人数	内容
2007 年 1 月	15	一年を企画する
2007 年 2 月	13	dbs, dpatch
2007 年 3 月	80	OSC 仮想化
2007 年 4 月	19	quilt, darcs, git
2007 年 5 月	23	etch, pbuilder, superh
2007 年 6 月	4	エジンバラ開催: Debconf7 実況中継
2007 年 7 月	18	Debconf7 参加報告
2007 年 8 月	25	cdn.debian.or.jp
2007 年 9 月	14	exim
2007 年 10 月	30	OSC Tokyo/Fall(CUPS)
2007 年 11 月	19	live-helper, tomoyo linux kernel patch, server
2007 年 12 月	11	忘年会
2008 年 1 月	23	一年を企画する
2008 年 2/29,3/1	36	OSC
2008 年 3 月	37	データだけのパッケージ、ライセンス
2008 年 4 月	17	バイナリパッケージ
2008 年 5 月	20	複数のバイナリパッケージ
2008 年 6 月	10	debhelper
2008 年 7 月	17	Linux kernel patch / module パッケージ
2008 年 8 月	10	Debconf IRC 会議と Debian 温泉
2008 年 9 月	17	po4a,「 Debian メンテナのお仕事」
2008 年 10 月	11?	OSC Tokyo/Fall
2008 年 11 月	17	「 その場で勉強会資料を作成しちゃえ」 Debian を使った LaTeX 原稿作成合宿
2008 年 12 月	12	忘年会

表 3　東京エリア Debian 勉強会参加人数 (2009-2010 年)

	参加人数	内容
2009 年 1 月	12	一年を企画する
2009 年 2 月	30	OSC パッケージハンズオン
2009 年 3 月	23	Common Lisp, パッケージ作成
2009 年 4 月	15	Java Policy, ocaml, 開発ワークフロー
2009 年 5 月	13	MC-MPI パッケージ化、 Erlang、 Android アプリ、 DDTP
2009 年 6 月	14	DDTP・ DDTSS、 bsdstats パッケージ、 Debian kFreeBSD
2009 年 7 月	4	スペインにて Debconf 9
2009 年 8 月	14	スペイン Debconf 9 参加報告
2009 年 9 月	26	GPG キーサインパーティー
2009 年 10 月	30	OSC Tokyo Fall
2009 年 11 月	12	Octave, R, gnuplot, auto-builder
2009 年 12 月	10	忘年会
2010 年 1 月	17	東京大学にて新年会
2010 年 2 月	11	Debian 温泉,ocaml,haskell
2010 年 3 月	12	weka,fftw,dpkg v3 quilt
2010 年 4 月	15	upstart,piuparts,debtags
2010 年 5 月	22	筑波大学,kernel
2010 年 6 月	12	OSC-Do リハーサル
2010 年 7 月	0	キャンセル
2010 年 8 月	3	Debconf (NYC)
2010 年 9 月	30	OSC Tokyo/Fall
2010 年 10 月	13	俺の Debian な一日
2010 年 11 月	15	ext4,btrfs,nilfs,ceph
2010 年 12 月	14	cacert, libsane

表 4　東京エリア Debian 勉強会参加人数 (2011 年)

	参加人数	内容
2011 年 1 月	12	荻窪,Kinect, アンケートシステム,CACert サイン会
2011 年 2 月	13	北新宿生涯学習館,HDFS,Debian Game Team
2011 年 3 月	?	OSC Tokyo/Spring,CACert ATE Tokyo
2011 年 4 月	12	IIJ,backports,initramfs, 月刊 PPC64
2011 年 5 月	15	戸山生涯学習館,Apache2 モジュール,Debian on ニフクラ,Debian/m68k, 月刊 PPC64
2011 年 6 月	17	東京オリンピックセンター, ドキュメント処理系,2011 再計画
2011 年 7 月	3	DebConf11
2011 年 8 月	12	荻窪, パッケージング関連, Debconf11 報告
2011 年 9 月	9	山喜旅館,Debian 温泉 2011
2011 年 10 月	22	筑波大学,Haskell,LaTeX, レポート自動生成, 月刊 Debhelper 開始
2011 年 11 月	?	OSC Tokyo/Fall
2011 年 12 月	9	スクウェア・エニックス,quilt で porting, 月刊 Debhelper, 振り返り



6 quilt で porting してみた

杉本典充

6.1 はじめに

本稿は debian のパッケージ作成で使用しているパッチ管理ツール quilt の紹介、および quilt を用いて kfreebsd の porting パッチを作成しましたので発表します。

6.2 Debian パッケージのフォーマット

現在 Debian で使用している主なソースパッケージフォーマットは 2 つです。[*1][*2]

- 3.0(native)： tarball 1 つで構成されたソースパッケージフォーマット
  - packagename-version.tar.ext
  - packagename-version.dsc
- 3.0(quilt)： あるアップストリームの tarball と Debian パッケージ作成に必要な tarball に分割しているソースパッケージフォーマット
  - packagename-upstreamversion.orig.tar.ext
  - packagename-upstreamversion.orig-component.tar.ext（ 任意）
  - packagename-debianversion.debian.tar.ext
  - packagename-debianversion.dsc

ソースパッケージフォーマットの詳細は ‘dpkg-source(1)’ コマンドで確認することができます。[*3]

アップストリームのソースコードを無修正のまま Debian パッケージを作成した場合、 lintian 処理でエラーになったり、 Debian の開発ポリシーを満たせない場合があります(アップストリームのソースコードは IPv6 で動作しない、など）。そのときは、 3.0(quilt) のソースパッケージフォーマットでパッケージを作成し、必要なパッチを当ててからソフトウェアのビルド処理を行うことで対応させます。

6.3 quilt パッケージ

quilt は debian パッケージを作成するときにパッチファイルを管理するツールとして採用しているソフトウェアです。Debian パッケージとして提供しており、 apt でインストールできます。

```
# apt-get update
# apt-get install quilt
```

[*1] 東京エリア Debian 勉強会 2010 年 03 月号「 dpkg ソース形式 “3.0(quilt)”」 吉野与志仁

[*2] `http://wiki.debian.org/Projects/DebSrc3.0`

[*3] man の記述には 3.0(custom)、 3.0(git)、 3.0(bzr) というパッケージ形式の定義があるようです。

quilt コマンドは複数の patch ファイルをスタックに積んだようなイメージで、パッチの適用順番を管理します。[*4]

quilt コマンドを使うことで、パッケージ作成に必要なパッチが複数ある場合も正しく適用でき、不要なパッチが出てきたときにそのパッチを取り除くことが容易になります。

6.4 quilt の使用例: Debian GNU/kFreeBSD 向けの porting パッチ作成

6.4.1 Debian GNU/kFreeBSD とは

Debian Project で開発している OS は Linux カーネルを用いた「Debian GNU/Linux」が有名ですが、Linux カーネル以外を用いた Debian があります。その 1 つとして FreeBSD カーネルを用いた「Debian GNU/kFreeBSD」[*5]があり、ユーザランドは Debian なので APT が使え、デバイスやシステムコールといったカーネル特有の機能は FreeBSD カーネルに準じる、という特徴があります。Debuan GNU/kFreeBSD は安定版のリリースには至っていませんが、日々開発が続けられています。

本稿では以降「Debian GNU/kFreeBSD」を「kfreebsd」と略記します。

6.4.2 kfreebsd の porting 作法

Debian の porting 関連情報および kfreebsd の porting 情報は以下にあります。

- `http://www.debian.org/ports/`
- `http://www.debian.org/ports/kfreebsd-gnu/`
- `http://glibc-bsd.alioth.debian.org/porting/`

「`http://glibc-bsd.alioth.debian.org/porting/PORTING`」を読むと kfreebsd 向けに porting するには以下の確認および修正が行うようにとあります。

- Add our system name to checks here and there
  - Makefile やスクリプト中の uname などをチェックしてください。
- debian/control files
  - debian/control ファイルで Architecture を linux 専用ソフトウェアの場合は「linux-any」等、CPU の違いのみで Linux、kfreebsd は関係なく使用できるソフトウェアの場合は「any-i386」等に変更してください。
- Libraries, your beloved enemy
  - libtool、aclocal.m4 周りに対応してください。
- Preprocessor Variables
  - kfreebsd のシステムマクロは「__FreeBSD_kernel__」、バージョンマクロは「__FreeBSD_kernel_version」なので対応してください。
- Writing to devfs (kFreeBSD)
  - FreeBSD カーネルでは(udev ではなく) devfs を使うようにしてください。
- RT signals
  - FreeBSD カーネルは「POSIX RT (realtime) signals」がないので変更してください。
- Get libc soname (6 or 6.1 on linux-gnu, 0.1 on kfreebsd-gnu, etc)
  - 使用する libc の名前をハードコードしているプログラムがあれば修正してください。

6.4.3 kfreebsd に対応させたいパッケージとその原因

今回 porting するパッケージは既に kfreebsd 用パッケージとして存在している icewm となります。icewm はウィンドウマネージャのソフトウェアで軽快な動作をするのが特徴です。

[*4] 東京エリア Debian 勉強会 2007 年 01 月号「パッチ管理ツール quilt の使い方」小林儀匡

[*5] `http://wiki.debian.org/Debian_GNU/kFreeBSD`

icewm ではタスクバーにバッテリー残量アイコンを表示する機能がありますが kfreebsd では表示されません。linux-i386 及び linux-amd64 では表示されるため kfreebsd のポーティングが不完全の可能性があります。

まずはビルド準備とソースコードのダウンロードを行います。

```
# apt-get update
# apt-get build-dep icewm
$ apt-get source icewm
```

ソースコードを「__FreeBSD__」及び「__linux__」grep すると、電源周りの処理で以下のマクロが検出されます。

```
$ cd icewm-1.3.7/src
$ grep -nr __FreeBSD__ *
aapm.cc:30:#ifdef __FreeBSD__
aapm.cc:74:#if defined(__FreeBSD__) && defined(i386)
aapm.cc:99:#if defined(__FreeBSD__) && defined(i386)
aapm.cc:273:#ifndef __FreeBSD__
aapm.cc:333:#ifndef __FreeBSD__
aapm.cc:418:#ifndef __FreeBSD__
aapm.cc:463:#ifdef __FreeBSD__
aapm.cc:885:#ifndef __FreeBSD__
aapm.h:2:#if defined(linux) || (defined (__FreeBSD__)) || (defined(__NetBSD__) && defined(i386))
aapm.h:7:#if defined(__FreeBSD__) || defined(__NetBSD__) || defined(__OpenBSD_
(以下略)

$ grep -nr __linux__ *
(なにもなし)
```

__FreeBSD__マクロは FreeBSD OS を示すマクロですが、kfreebsd では「__FreeBSD_kernel__」がシステム（厳密にはカーネル）を示すマクロです。そのためマクロでシステムを切り分けているはずが kfreebsd 向けのパッケージビルド時に Linux カーネル向けの処理が有効なコードとしてビルドされてしまいバッテリー残量アイコンの表示がうまく動作していないようです。

そのため、今回はこのマクロを「PORTING」の記述に従って以下のような修正を行います。

```
(修正前) #ifdef __FreeBSD__

(修正後) #if defined(__FreeBSD__) || defined(__FreeBSD_kernel__)
```

これで修正方針が定まりましたので、debian パッケージ作成に向けてパッチを作成していきます。

6.5 quilt の登場

今回は新規のパッチファイルとなるため、quilt のパッチスタックに新規追加します。

```
$ cat debian/patches/series
#cvs_fixes
package_build_fixes
#misc_fixes
debian_defaults
#compiler_defaults
#iconify_on_wm_hint
#move-to-screen
i18n_updates
tray_hotfixes
imap_unseen
ifstate_exact_check
debian-changes-1.3.7~pre2-1.1

$ quilt new kfreebsd_porting_aapm
Patch kfreebsd_porting_aapm is now on top

$ cat debian/patches/series
#cvs_fixes
package_build_fixes
#misc_fixes
debian_defaults
#compiler_defaults
#iconify_on_wm_hint
#move-to-screen
i18n_updates
tray_hotfixes
imap_unseen
ifstate_exact_check
debian-changes-1.3.7~pre2-1.1
kfreebsd_porting_aapm
```

これでパッチを新規に追加できました。

次に porting するために修正を行うソースファイルを quilt で管理するように登録処理をします。その後「 quilt edit ソースファイル」を実行すると、環境変数 EDITOR で登録したエディタが自動で起動しますのでソースファイルの修正作業を行います。

```
$ quilt add src/aapm.h
File src/aapm.h added to patch kfreebsd_porting_aapm

$ quilt edit src/aapm.h
File src/aapm.h is already in patch kfreebsd_porting_aapm

$ quilt refresh
Refreshed patch kfreebsd_porting_aapm

$ quilt add src/aapm.cc
File src/aapm.cc added to patch kfreebsd_porting_aapm

$ quilt edit src/aapm.cc
File src/aapm.cc is already in patch kfreebsd_porting_aapm

$ quilt refresh
Refreshed patch kfreebsd_porting_aapm
```

できたパッチファイル「 kfreebsd_porting_aapm」を確認します。

```
$ cat debian/patches/kfreebsd_porting_aapm
Index: icewm-1.3.7/src/aapm.h
===================================================================
--- icewm-1.3.7.orig/src/aapm.h 2010-10-31 23:09:36.000000000 +0900
+++ icewm-1.3.7/src/aapm.h      2011-12-10 23:17:15.000000000 +0900
@@ -1,10 +1,10 @@

-#if defined(linux) || (defined (__FreeBSD__)) || (defined(__NetBSD__) && defined(i386))
+#if defined(linux) || (defined (__FreeBSD__)) || (defined (__FreeBSD_kernel__)) || ((defined(__NetBSD__) && defined(i386))

 #include "ywindow.h"
 #include "ytimer.h"

-#if defined(__FreeBSD__) || defined(__NetBSD__) || defined(__OpenBSD__)
+#if defined(__FreeBSD__) || defined(__FreeBSD_kernel__) || defined(__NetBSD__) || defined(__OpenBSD__)
 #define APMDEV "/dev/apm"
 #else
 #define APMDEV "/proc/apm"
( 以下略)
```

あとはパッケージをビルドします。

```
$ dch
$ debuild -uc -us
```

作成したパッケージをインストールして確認します。

```
$ sudo dpkg -i icewm-common_1.3.7-1.1_kfreebsd-amd64.deb icewm_1.3.7-1.1_kfreebsd-amd64.deb
$ reboot
```

作成したパッチはバグレポートと共に BTS へ送信しておきましょう。

```
$ reportbug
$ w3m http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=650395
```

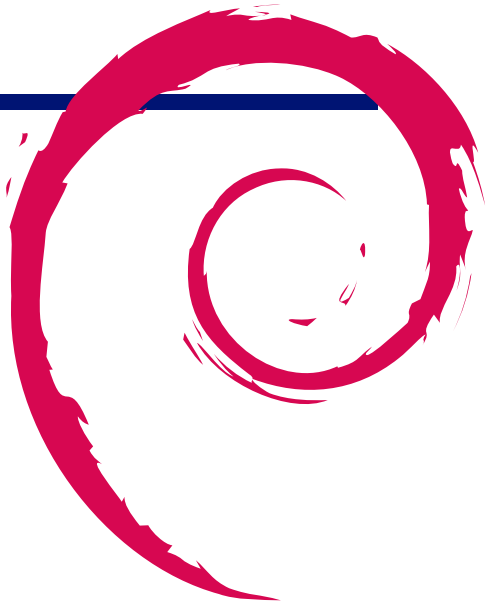
6.6 終わりに

これで kfreebsd の icewm ウィンドウマネージャ上でバッテリー残量アイコンを表示することができました。
みなさんも kfreebsd 含め様々なアーキテクチャで数多くのパッケージが Debian で動作するようにがんばりましょう。

6.7 参考情報

- 東京エリア Debian 勉強会 2007 年 01 月号「 パッチ管理ツール quilt の使い方」小林儀匡
- 東京エリア Debian 勉強会 2010 年 03 月号「 dpkg ソース形式”3.0(quilt)”」 吉野与志仁
- man dpkg-source(1), man quilt(1)
- http://wiki.debian.org/Projects/DebSrc3.0
- http://www.debian.org/doc/manuals/maint-guide/first.ja.html

- http://wiki.debian.org/Debian_GNU/kFreeBSD
- http://glibc-bsd.alioth.debian.org/porting/



7 月刊 Debhelper 第 2 回

野島 貴英

7.1 はじめに

Debian パッケージを作成する際、沢山の処理を debian/rule というファイルに GNU の make の makefile の形式で記述することになります。しかしながら、細かい処理を記載していくと膨大な量となってしまいます。これをできるだけ簡潔に記載できるように考えられたツールとして debhelper というコマンド群が存在します。

本企画はこの debhelper のコマンドについて、毎月持ち回りで解説していくというものです。ルールは、毎月 2 つ以上のコマンドを解説し、次回発表の立候補が無い場合は発表者が次の発表者を決めれるというルールの元に進めて行きます。

7.2 今月のコマンドその 1:dh

7.2.1 dh の動作概要

dh は引数に指定したシーケンス名に基づいて一連の debhelper を起動するコマンドとなります。実際の使い方では、以下の内容を debian/rules に記述して利用します。

```
$ cat debian/rules
#!/usr/bin/make -f
%:
        dh $@
```

7.2.2 dh に指定できるシーケンス名

dh に指定できるシーケンス名は表 6 の通りです。

なお、–with foo を指定すると、 dh に指定可能なシーケンスが増える場合があります (例: –with quilt の patch シーケンス等。)

7.2.3 dh のコマンドラインオプション

表 6 に dh のコマンドラインオプションを載せます。 (man dh より)

この他にも、 debhelper コマンド共通で使えるコマンドラインオプションが man debhelper に記載されており、 dh コマンドでも利用できます。こちらも参照ください。

7.2.4 廃止されたコマンドラインオプション

–until,–before,–after,–remaining がありましたが、これらは全部 dh が解釈する “override_*DH* **コマンド名**ターゲット” による動作に置き換えられた為、**廃止**となりました。

なので、昔の debian/rule にあるような、以下の用な書き方は**廃止**です。

シーケンス名	シーケンスの説明
binary	構築からパッケージ作成まで実行するシーケンスです。
binary-arch	arch 依存のパッケージの構築からパッケージ作成まで実行するシーケンスです。
binary-indep	arch 非依存のパッケージの構築からパッケージ作成まで実行するシーケンスです。
build	構築からテストまで実行するシーケンスです。
build-arch	arch 依存のパッケージの構築からパッケージ作成まで実行するシーケンスです。
build-indep	arch 非依存のパッケージの構築からパッケージ作成まで実行するシーケンスです。
clean	一度パッケージを構築したディレクトリから、パッケージ構築時に生成したものを取り除き、構築ディレクトリを綺麗にします。
install	構築から、パッケージ生成直前までの処理を行うシーケンスです。
install-arch	arch 依存のパッケージについて、構築から、パッケージ生成直前までの処理を行うシーケンスです。
install-indep	arch 非依存のパッケージについて、構築から、パッケージ生成直前までの処理を行うシーケンスです。

表 5　dh で指定できるシーケンス名一覧

オプション	説明
–with addon[,addon ...]	debhelper コマンドに適切な場所で一連のコマンドを実行するような付加機能 (addon) を指定します。
–without addon	–with とは逆の働きをします。指定された付加機能を使わないようにします。
–list, -l	利用可能な付加機能 (addon) 一覧です。
–no-act	指定された一連の処理の内容を表示するだけコマンドとなります。表示だけして実際にはコマンドを実行しません。
その他	dh に、先に記載した以外の何かオプションを渡すとそれはのちに実行する全コマンドへ引き渡されます。-v、-X、-N や、他の特別なオプションを指定するのに使われます。

表 6　コマンドラインオプション一覧

廃止された書き方

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

%:
        dh $@

build: build-stamp
build-stamp:
        dh build --before configure
        dh_auto_configure -- --with-gnu-ld --disable-nls
        dh build --after configure
        touch build-stamp
```

代わりの書き方は次の章で述べます。

7.2.5　“override_*debhelper* コマンド名” ターゲットについて

dh コマンドは”dh シーケンス名” により、そのシーケンスに必要な一連の debhelper コマンドを呼び出す機能があります。(どんな debhelper コマンドが呼び出されるかは、–no-act をオプションにつけて、 dh –no-act build とか、 dh –no-act install とかして見てください)

この呼び出されるコマンドを一部変更したい場合は以下のように書きます。

今時の書き方:

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

%:
        dh $@

override_dh_autoconfigre:
        dh_auto_configure -- --with-gnu-ld --disable-nls
```

こうすると、本来であれば、 dh_auto_configure がオプション無しで呼び出される場所が全部 “dh_auto_configure –

–with-gnu-ld –disable-nls” で呼び出されるようになります。

他の例として、 configure スクリプトが無く、代わりに Imakefile があるような古い X 用のプログラムをパッケージにする用な場合は以下のように書きます。

```
Imakefile を利用するような場合:
#!/usr/bin/make -f

%:
        dh $@ --with quilt

override_dh_auto_configure:
        xmkmf -a
```

こうすると、本来であれば、 dh_auto_configure が呼び出される場所全部で、 “xmkmf -a” を呼び出すようになります。

この”override_*debhelper* **コマンド名**” ターゲットは、コマンドを**実行したくない場合**にも利用可能です。 (“override_*debhelper* **コマンド名**” のアクションを空にする事がミソです。)

```
dh_auto_test,dh_compress,dh_fixperms を実行したく無い場合:
#!/usr/bin/make -f

%:
        dh $@

override_dh_auto_test override_dh_compress override_dh_fixperms:
```

また、 build-arch,binary-arch,build-indep,binary-indep ターゲットが dh に指定されるときにあわせて振る舞いを変更したい場合は”override_*debhelper* **コマンド名**-indep” や、”override_*debhelper* **コマンド名**-arch” を使って、それぞれの場合に dh によって呼び出されるコマンドを変更できます。以下の例では、ドキュメントパッケージの作成に時間がかかるので、 build-indep や、 binary-indep の時にだけドキュメントを作成してくれるようにする場合の debian/rule となります。

```
ドキュメント作成を分離して、時間のかかるドキュメント作成が何度も実行されないようにする:
#!/usr/bin/make -f
%:
        dh $@

override_dh_auto_build-indep:
        $(MAKE) -C docs

# No tests needed for docs
override_dh_auto_test-indep:

override_dh_auto_install-indep:
        $(MAKE) -C docs install
```

7.2.6 addon について

dh コマンドのオプション–with addon にて addon が提供するパッケージの作成方法を組み込む事ができます。お使いのシステムで現在どんな addon が使えるかは dh –list を実行すると一覧が出てきます。

```
$dh --list
bash-completion
dkms
python-central
python-support
python2
quilt
tex
$
( ... お使いのシステムによって表示される量が変わります...)
```

実はこれらは/usr/share/perl5/Debian/Debhelper/Sequence/”addon 名”.pm としてインストールされています。こちらを利用して、現在の Debian で、どんな addon が提供されているかを知りたければ、次のようにして調べる事ができます。

```
$ apt-file search Debhelper/Sqeuence
autotools-dev: /usr/share/perl5/Debian/Debhelper/Sequence/autotools_dev.pm
bash-completion: /usr/share/perl5/Debian/Debhelper/Sequence/bash_completion.pm
cli-common-dev: /usr/share/perl5/Debian/Debhelper/Sequence/cli.pm
... 中略...
sphinx-common: /usr/share/perl5/Debian/Debhelper/Sequence/sphinxdoc.pm
tex-common: /usr/share/perl5/Debian/Debhelper/Sequence/tex.pm
xserver-xorg-dev: /usr/share/perl5/Debian/Debhelper/Sequence/xsf.pm
xulrunner-dev: /usr/share/perl5/Debian/Debhelper/Sequence/xulrunner.pm
$ apt-file search Debhelper/Sqeuence | wc -l
43
$
```

全部で 43 個もありますね。(debian sid で実行)

複数 addon を指定したい場合は繰り返し–with オプションで指定したり、カンマで区切って指定します。

```
quilt 用の addon と、 autotools_dev 用の addon を併用したい時:
#!/usr/bin/make -f
%:
        dh $@ --with quilt --with autotools_dev
#       dh $@ --with quilt,autotools_dev   も OK
```

7.2.7 addon の構造について

addon が何をしているかは/usr/share/perl5/Debian/Debhelper/Sequence/”addon 名”.pm を覗くとピンときます。

例えば、–with quilt の場合、

1. dh clean にて、 dh_clean を呼び出す前に、 quilt パッケージが一緒に提供している dh_quilt_unpatch コマンドを呼び出すようになります。
2. dh build では、 dh_auto_configure の前に dh_quilt_patch を呼び出すようになります。
3. dh にシーケンス名 patch が追加され、 dh patch が使えるようになります。

addon を自分で書く場合は、 dh 内で定義されている表 7 の API を呼び出して書いてください。

API 名	API の説明
insert_before($existing,$new)	$existing で指定される debhelper コマンドを実行する直前に$new を実行します。
insert_after($existing,$new)	$existing で指定される debhelper コマンドを実行した直後に$new を実行します。
remove_command($command)	$command を dh が実行しないようにします。
add_command($command,$sequence)	$sequence で示されるシーケンスで実行されるコマンド群の最後に$command を付け加えます。また、本 API を使って 7.2.2 章で示されないシーケンスを新たに作成することができます。
add_command_options($command,@options)	$command に、配列@options で示される一連のオプションを付け加えて実行するようにします。
remove_command_options ($command,@options)	$command から配列@options で示される一連のオプションを取り除く。@options をまったく指定せずに remove_command_options($command) と呼び出すと、$command についてのオプション全部を取り除きます。

表 7 addon 用の API 一覧

量もそんなになく、非常にわかりやすいので、興味のある人は/usr/share/perl5/Debian/Debhelper/ Sequence/quilt.pm を試しに読んでみるとよいと思います。

なお、複数の addon を指定した場合、同じ内容の debhelper コマンドが意図せず複数回も同じシーケンスに挿入される事がありますが、きちんと 1 個の呼び出しにまとめてくれます。

7.2.8 dh の内部動作

dh コマンドは debian/rule が make ファイルである事を利用しながら、 make コマンドと協調して動作します。

図 2 に dpkg-buildpackage を呼び出したときの dh の内部動作を示します。

図 2 から判るように、 dpkg-buildpackage から make コマンドが起動され、次に dh が起動され、さらに dh から make が起動されるという関係になっている事が判ります。また、 override_*debhelper* **コマンド名**ターゲットの処理を行うのに、 make コマンドを使って処理をしているという事も判ります。

dh を使うと、 make コマンドは override_*debhelper* **コマンド名**ターゲットの処理をする役目だけを担当します。そのうち、 dh コマンドが進化すると、 make コマンドの力を借りなくてもパッケージ作成ができるようになるかもしれませんね。

7.2.9 “debian/**パッケージ名**.debhelper.log” **ファイルについて**

最近の dh コマンドを使う debian/rules には、ファイルの依存関係についての記載がありません。この為、パッケージビルド中で処理が中断した場合、どこから再開すれば良いかを debian/rules で make が判定する事はできません。

実は、 dh コマンドは clean 以外のシーケンスが指定されると、”debian/パッケージ名.debhelper.log” というファイルに処理を行った debhelper コマンドを記録しています。ここで、万一 dh の処理が中断した場合、処理をどこから始めれば良いかについてはこのログファイルを参照して処理の再開を行います。

“debian/パッケージ名.debhelper.log” の中身は以下のようになっています。

```
debian/パッケージ名.debhelper.log の中身:
dh_auto_test
dh_prep
dh_installdirs
... 中略...
dh_buiddeb
```

また、このファイルは dh clean によって消去されます。なお、 dh clean の時には、このログファイルは作成されません。つまり、 dh clean の処理を中断した場合は、 dh clean は呼び出される一連の debhelper コマンドは最初から実行されてしまいます。

なお、処理再開の場所は、このログファイルのみ参照して決める為、処理を中断した後に、パッケージのソースファイルを変更して再開させるような使い方はできません。例えば、ソースファイル中のあるファイルを変更した為、特定のパッケージのシーケンスについては再会時に全部やり直しが必要だったとしても、これを自動で検知することはできません。

7.2.10 dpkg-buildflag **との関係**

dh は互換性度合い (COMPATABLITY LEVEL) の v9 から、パッケージ構築の時に使う環境変数を設定するため、内部で dpkg-buildflag 相当の処理を呼び出します。

その為、 9 を debian/compat に指定すると、 debhelper コマンドに設定される環境変数は、

1. /etc/dpkg/buildflags.conf の中身
2. XDG_CONFIG_HOME/dpkg/buildflags.conf (XDG_CONFIG_HOME は環境変数です) の中身
3. HOME/.config/dpkg/buildflags.conf (HOME は環境変数です) の中身
4. DEB_flag_MAINT_SET, DEB_flag_MAINT_STRIP, DEB_flag_MAINT_APPEND, DEB_flag_MAINT_PREPEND, DEB_BUILD_MAINT_OPTINS（ 全部環境変数です）の値

により様々に変化します。どのように変わるかは man dpkg-buildflag を参照してください。

7.3 今月のコマンドその 2:dh_testroot

7.3.1 dh_testroot 動作詳細

現在の実行ユーザが root であるかどうかを確認するコマンドです。 root ユーザでは無い場合、エラーメッセージを出力して処理を中断します。

7.3.2 dh_testroot コマンドラインオプション

コマンドラインオプションは特にありません。何か指定しても無視されます。

7.3.3 dh_testroot を実行してみる

早速、実行してみましょう。

```
$ sudo dh_testroot
$ echo $?
0
$ dh_testroot
You must run this as root (or use fakeroot).
$ echo $?
255
$ fakeroot dh_testroot
$ echo $?
0
```

このように root 権限で実行するか、 fakeroot 経由で実行した時のみ 0 を返却します。

7.3.4 次回の発表について

次の発表者は勉強会で発表します。選ばれた人はよろしくおねがいします。

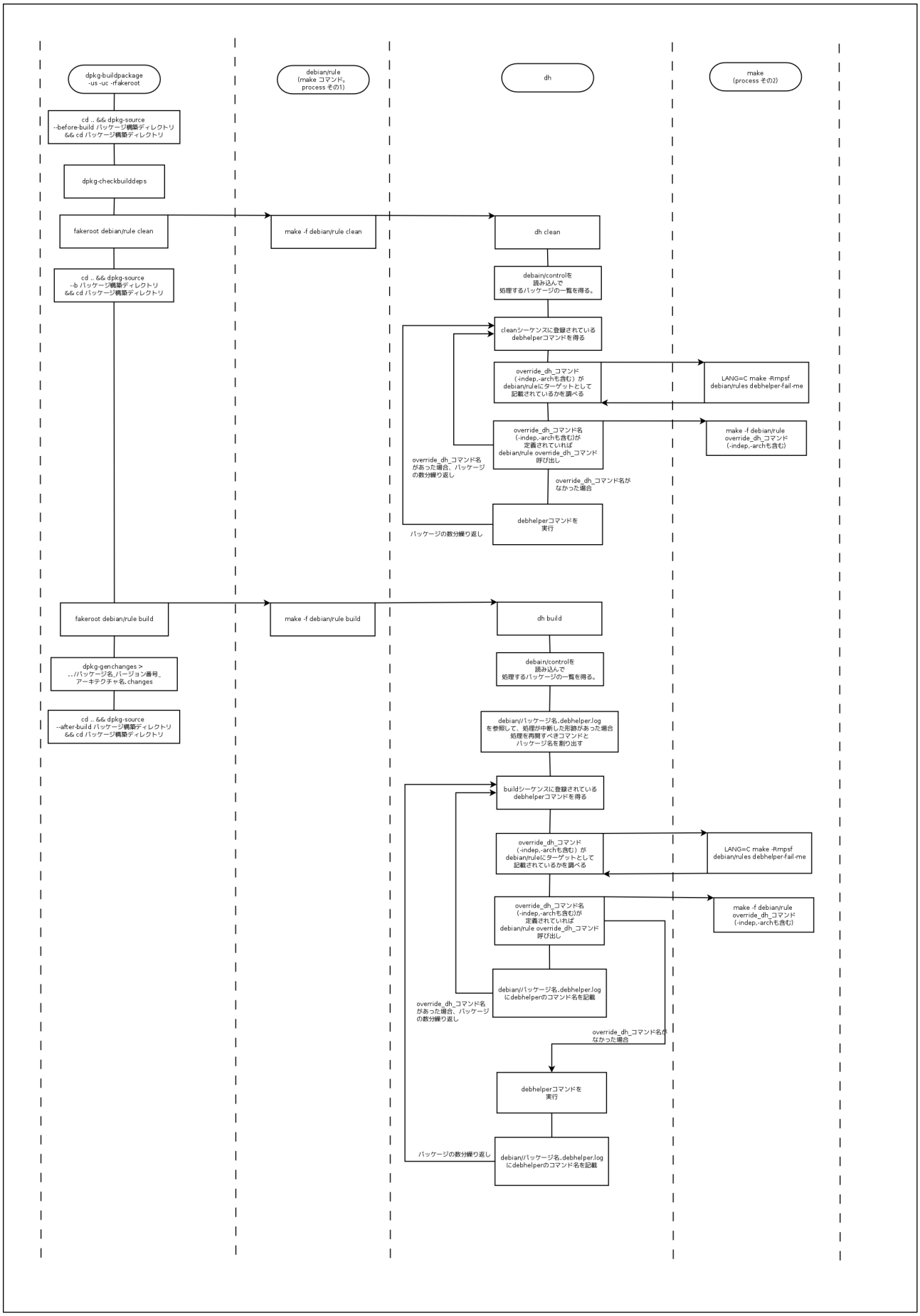

dpkg-buildpackage
-us-uc -rfakeroot
debian/rule
(make コマンド。
process その1)
dh
make
(process その2)
cd .. && dpkg-source
--before-build パッケージ構築ディレクトリ
&& cd パッケージ構築ディレクトリ
dpkg-checkbuilddeps
fakeroot debian/rule clean
make -f debian/rule clean
dh clean
debain/controlを
読み込んで
処理するパッケージの一覧を得る。
cleanシーケンスに登録されている
debhelperコマンドを得る
override_dh_コマンド
(-indep,-archも含む) が
debian/ruleにターゲットとして
記載されているかを調べる
LANG=C make -Rrnpsf
debian/rules debhelper-fail-me
override_dh_コマンド名
(-indep,-archも含む)が
定義されていれば
debian/rule override_dh_コマンド
呼び出し
make -f debian/rule
override_dh_コマンド
(-indep,-archも含む)
override_dh_コマンド名
があった場合、パッケージ
の数分繰り返し
override_dh_コマンド名が
なかった場合
debhelperコマンドを
実行
パッケージの数分繰り返し
cd .. && dpkg-source
-b パッケージ構築ディレクトリ
&& cd パッケージ構築ディレクトリ
fakeroot debian/rule build
make -f debian/rule build
dh build
dpkg-genchanges >
../パッケージ名_バージョン番号_
アーキテクチャ名.changes
debain/controlを
読み込んで
処理するパッケージの一覧を得る。
cd .. && dpkg-source
--after-build パッケージ構築ディレクトリ
&& cd パッケージ構築ディレクトリ
debian/パッケージ名.debhelper.log
を参照して、処理が中断した形跡があった場合
処理を再開すべきコマンドと
パッケージ名を割り出す
buildシーケンスに登録されている
debhelperコマンドを得る
override_dh_コマンド
(-indep,-archも含む) が
debian/ruleにターゲットとして
記載されているかを調べる
LANG=C make -Rrnpsf
debian/rules debhelper-fail-me
override_dh_コマンド名
(-indep,-archも含む)が
定義されていれば
debian/rule override_dh_コマンド
呼び出し
make -f debian/rule
override_dh_コマンド
(-indep,-archも含む)
debian/パッケージ名.debhelper.log
にdebhelperのコマンド名を記載
override_dh_コマンド名
があった場合、パッケージ
の数分繰り返し
override_dh_コマンド名が
なかった場合
debhelperコマンドを
実行
パッケージの数分繰り返し
debian/パッケージ名.debhelper.log
にdebhelperのコマンド名を記載

図 2　dh 内部動作

8 東京エリア Debian 勉強会の開催方法

野島 貴英

8.1 はじめに

東京エリア Debian 勉強会では、毎年 12 月号資料では勉強会の開催方法についてまとめています。今回は野島が一通りの流れをまとめます。これを読めば、あなたも東京エリア Debian 勉強会を開けるはず!

8.2 東京エリア Debian 勉強会を開く時のスケジュール

時期	作業内容
開催 2 ヶ月前	場所を確保します。会議室は 2 ヶ月前に予約しないと埋まりやすいです。
開催 2 ヶ月前から 1 ヶ月前	勉強会のテーマと、事前課題を決めます。
開催 1 ヶ月前	勉強会のテーマに沿った原稿を募集します。時間枠にあわせた発表が埋まらない場合、発表者を枠が埋まるまで、探す事になります。また、開催にあたり、作業の分担が必要な時は、この時ぐらいに割り当てを完了します。また、宴会の場所について、確保が困難な事が予想される場合は、あらかじめ席のみ予約しておきます。
開催 2 週間前までに	Debian 勉強会予約システム `http://debianmeeting.appspot.com` に予約登録ページを作成します。 Debian JP Blog `http://www.debian.or.jp/` 及び、 Debian 勉強会 Web サイト `http://tokyodebian.alioth.debian.org/` に勉強会の内容を提示します。メールで debian-users メーリングリストにアナウンスを投げます。 debian JP twitter や mixi Debian コミュニティーなどにもアナウンスすることが多いです。
開催 1 週間前	発表者は勉強会資料用リポジトリに資料をコミットします。
開催 2 日前	参加者の募集の締切りを行います。この時までに揃った事前課題を勉強会資料にマージします。
開催 1 日前	kinkos 等の印刷製本サービスに原稿印刷および製本を依頼します。
開催当日	1. 参加者から費用の回収と集計をします。 2. 勉強会を定刻どおりに開催します。 3. 司会者、発表者と協力して時間内に勉強会を終了します。 4. 宴会の場所の確保が困難でない時は、宴会の場所を抑えます。 5. 宴会の開催、宴会の会計をします。
開催後	Debian 勉強会予約システムからアンケートをおくります。
12 月	勉強会のアンケート結果などを一年分集計して発表します。

表 8　全体スケジュール

8.3 東京エリア Debian 勉強会の開催調整

東京エリア Debian 勉強会の開催調整は専用の ML があり、普段はこちらで調整が行われます。運営者専用の ML で、一部のコアメンバーのみが参加しています。運営に積極的に参加したい人は勉強会運営者に連絡を取る必要があります。

8.4 初めて東京エリア Debian 勉強会の開催を担当する場合の事前準備

初めて東京エリア Debian 勉強会の開催を担当する場合以下の項目が準備してあるとスムーズです。

1. Debian JP Project の提供するサーバーへログインできるアカウント
   Debian JP Blog を更新する際に必要になります。これは通常 Debian JP Project の会員になるときに付与されます。会員でない人は会員にコミットしてもらうよう依頼する必要があります。
2. `http://alioth.debian.org/` アカウント資料の編集等を行う際に必要になります。アカウントを取得したら、ML で tokyodebian グループ権限に所属させてほしい旨、取得したアカウント名を沿えて依頼します。このアカウントは誰でも取得可能です。

また、勉強会の幹事担当の場合に必要な知識とコンピュータの環境は以下になります。

1. Debian が動作し、インターネットが利用できる PC の用意
2. ssh,emacs,muse-el,git、 subversion の基本操作についての知識
3. ブラウザと HTML の記載の知識
4. `http://alioth.debian.org/`,`http://qwik.jp/` の使い方の知識
5. LaTeX の知識

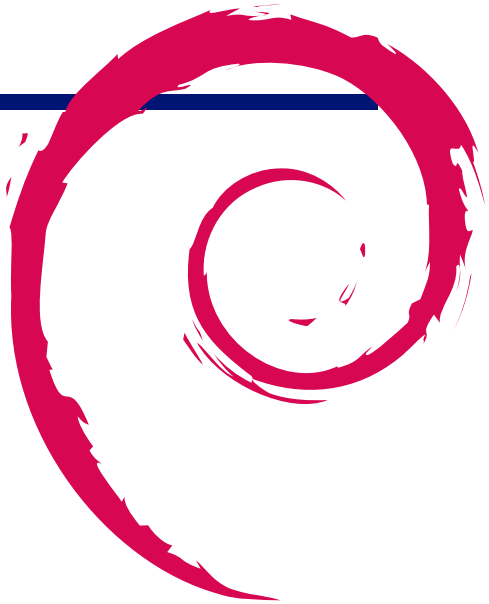
わからないことや困ったことがあればメールや ML で聞いてみましょう。

8.5 東京エリア Debian 勉強会の掲示の出し方

`http://www.debian.or.jp/` に掲示を出す場合の編集の仕方は、 `http://www.debian.or.jp/project/webmasters.html` にその記載があります。実際には、 svn でリポジトリをとってきて、 www.debian.or.jp/blosxom/data/events/tokyodebian-XX.d (XX は第 XX 回を示します）を作成します。

`http://debianmeeting.appspot.com/eventadmin/edit?eventid=` のフォームの編集はフォームに沿って情報を入れるだけです。ただ、細かい事は、 `http://tokyodebian.alioth.debian.org/YYYY-MM.html` へのリンクを URL の欄に指定し、細かい事はそちらを参照してもらいます。

`http://tokyodebian.alioth.debian.org/` へは、 `http://qwik.jp/ML` 名 のトップページに記載されているやり方に従います。実際には git で `git+ssh://git.debian.org/git/tokyodebian/muse.git`; リポジトリを clone して、 muse 形式で編集し、 make publish する事となります。



9 索引

Debian 勉強会資料

2011 年 12 月 17 日　　初版第 1 刷発行

東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）