



東京エリア Debian 勉強会

Debian 勉強会幹事 上川 純一
2007 年 2 月 17 日

1 Introduction

上川 純一

今月の Debian 勉強会へようこそ。これから Debian のあやしい世界に入るといふ方も、すでにどっぷりとつかっているといふ方も、月に一回 Debian について語りませんか？

目的として次の二つを考えています。

- メールではよみとれない、もしくはよみとってられないような情報について情報共有する場をつくる
- Debian を利用する際の情報をまとめて、ある程度の塊として整理するための場をつくる

Debian の勉強会といふ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。

Debian をこれからどうするといふ能動的な展開への土台としての空間を提供し、情報の共有をしたい、といふのが目的です。

目次

1	Introduction	1
2	事前課題	3
2.1	やまねさん	3
2.2	石原さん	3
2.3	澤田さん	4
2.4	キタハラさん	4
2.5	えとーさん	4
2.6	あけどさん	5
2.7	前田 耕平さん	5
2.8	小室 文さん	5
3	Debian Weekly News trivia quiz	6
3.1	2007 年 01 号	6
3.2	2007 年 02 号	6
3.3	2007 年 03 号	7
4	最近の Debian 関連のミーティング報告	8
4.1	東京エリア Debian 勉強会 24 回目報告	8
5	Debian 勉強会 2007 年度計画検討結果	9
6	dpatch についての小ネタ	10
6.1	インストールのしかた	10
6.2	作業のしかた	10
6.3	dpatch に含まれる各種ツールの紹介	10
6.4	debian/ディレクトリのみを展開したパッケージのメンテナンスをする	10
6.5	svn-buildpackage との連携	11
6.6	参考文献	11
7	dbx	13
7.1	dbx とはなにか	13
7.2	インストールのしかた	13
7.3	使いかた	13
7.4	パッチの変更	16
7.5	まとめ	16
8	次回	18

2 事前課題

上川 純一

今回の事前課題は「apt に足りなさそうな機能」と「パッケージングをしてみて感じたこと、または何故パッケージングをしないか」というタイトルで 200-800 文字程度の文章を書いてください。というものでした。その課題に対して下記の内容を提出いただきました。

2.1 やまねさん

「apt に足りなさそうな機能」

- 「apt の http では Redirect すらサポートしていない」らしいですよ？(<http://donrails.araki.net/archives/id/5580>)
- apt というかフレームワークというか判りませんが、言語・環境ごとに規定してのインストール。メタパッケージでもできることはありますが、なんというか面倒な事も多いような。

「パッケージングをしてみて感じたこと、または何故パッケージングをしないか」

パッケージは作っているの、感じたことを。

- パズルみたいで面白い :-)
- プログラム書けなくても、まあなんとかなるなる。
- distro specific なことをベタで書かれたものは扱づらい。

2.2 石原さん

apt に足りなさそうな機能

apt のことをすべて把握しているわけじゃないのでなんともいえません。使っているのは、`apt-get install hoge hoge` とか `apt-get update; apt-get upgrade` くらいで、たまに `apt-get remove hoge hoge` とかです。

こっこのへっぽこ設定で、時刻設定をハチャメチャなことにした状態で、`apt-get` をしてしまい、どうしようもなくなったことはありますが...^{*1}

あと、`apt-cache search` の結果 (パッケージ説明) が日本語だといいなあ、と思います。難しいのかなあ？

パッケージングをしてみて感じたこと、または何故パッケージングをしないか

パッケージングしたことないです。やり方がわかりません。そして、ホームディレクトリにパスを通して、`./configure ; make ; make install` で使えるし、ま、いっか、て感じなので調べていないですね...

^{*1} Windows Update の場合、ハチャメチャな時刻設定だと、たしかエラーになって出来なかったと思いますけど...

2.3 澤田さん

「パッケージングをしてみて感じたこと、または何故パッケージングをしないか」

apt に足りなさそうな機能とすると処理のロールバック機能だと思います。例えば、`dist-upgrade` してみた 壊れてるパッケージがあって `dist-upgrade` が完了できなかった とりあえず壊れてるパッケージを削除したくても中途半端な状態だと言われて削除できない orz `dist-upgrade` する前に戻れたら … といった感じです。ググってみるとそれっぽい試みはあるようですが `alioth` とかで引っ掛からないのだから形になっているものはないのかな？

「パッケージングをしてみて感じたこと、または何故パッケージングをしないか」

パッケージングして感じたこととすると…、いい言葉が思いつかないので具体例を示します。うちのサーバには日本語パッチを当ててパッケージングした `squirrelmail` が入っているのですが、DSA が出るたびに本家の最新とひとつ前を `diff` `diff` を見てローカルパッケージに手動パッチしてパッケージをビルド、しています。こちらへん、行番号は違って回りから判断してパッチを当ててくれるツールがあるとうれしいなと思います。どっちかという `more intelligent patch` な話ですが本家配布のものをいじったパッケージを入れてるって場合に重要な気がします。調べたことないからひょっとしたらあるのかも。もしかして今回のネタそのもの？

2.4 キタハラさん

apt に足りなさそうな機能

通常使用している範囲で、特に不足を感じる機能はありません。

パッケージングをしてみて感じたこと、または何故パッケージングをしないか

1. すみません、これまでパッケージングをした事がないので回答不能です。
2. 能力が無い&これまで必要なモノは既にあり、必要がなかった。

2.5 えとーさん

何個か思い付きました。一個思い着くと連鎖でがんがんでるもんですな。

1. ケーパビリティの細分化とその明確化

現状だとセキュア OS や Linux ケーパビリティを利用していてもパッケージ管理にはフル権限が必要になる。パッケージ信頼性は MD5sum の提供や署名によりある程度は担保されるもののシステム的な例外を作るしかない状況になっているので、なんとかしたい。運用で回避はできるのでまあいいといえいいのですが。。コストもそれなりに掛るし美しくない。

2. ユーザ単位での apt の利用

現状では root ユーザがシステムに対して行なうことが想定されているがそれ以外にユーザがシステムに影響を与えずにインストールした場合もあるだろう。こいつをどうにかできないか。

3. パッケージの集中管理

たとえサーバが二台しかなくても別々に見るの面倒なので一個のコンソールから管理したい。全ミラーじゃなくてインストールしているパッケージのみを選択的にミラーポリシーに従って自動的にアップデートを行なう。

4. セキュリティアップデートのタグ付け

サーバの場合はセキュリティアップデートとはいえ無造作に当てることはできないこともある。(できる場合は当然それでよいのだが) お仕事できっちりやる必要がある場合はアップデートがあればその差分ソースを精査し実験環境で動作確認を行なった上でアップデートを実行する必要がある、その場合はどうしようもないとし

でも、場合によっては管理コストから自動化したいしそれが認められる状況（条件）もある。これをサポートするためにセキュリティアップデートにタグで情報の付加を行ない事前に選択した条件にマッチした場合のみアップデートを行なうという機能があれば便利ではなからうか。また、バグの内容を解析して自動的に重みづけを行なうことや事前に重みづけを行なった機能との相関分析を行なうことによりアップデート条件の選択を簡易にできればより墮落できるであろう。

2.6 あけどさん

apt に足りなさそうな機能

apt を使っていてこんな機能が欲しいってのを挙げてみます。欲しい機能を持ったパッケージを探すのに apt-cache search とかで検索することがあるのですが grep とかで工夫しても見つけるのに手間がかかるのを何とかしたいです。で、考えてみました。まずは、

Google とかの検索エンジンと連動する機能（仕組み？）があればすごく便利になるかと思います。今の apt 系の検索ではライブラリとかアプリケーションなどのカテゴリズが今ひとつ整備できていない様に見えるので、それを補完する？機能が欲しいです。

もう一つ挙げてみますと、パッケージをインストールせずにマニュアルページだけ参照とか、パッケージ内の任意のファイルを参照（抽出）する機能ってのが apt で出来る様になれば良いかなと思います。

以上の機能ってのは dpkg とか他のコマンドの組み合わせで出来るかもしれませんがまとまった物としてあれば良いと考えています。

2.7 前田 耕平さん

「apt に足りなさそうな機能」

1) 足りなさそうな、というより実際足りない機能ですが、Vine Linux の apt だと、apt-line に書かずにローカルファイルも「apt-get install パッケージファイル名」でインストールできるのに、Debian ではできないことですね。dpkg を使えば、と言われればそれまでなのですが、

2) apt、というより dpkg の方が妥当かと思いますが、rpm だと、-q -changelog でパッケージの ChangeLog を表示できるのに、Deb パッケージだとオプションで ChangeLog を表示できないことでしょうか。（あるいは単に私が知らないだけ？）

「パッケージングをしてみて感じたこと、または何故パッケージングをしないか」

debian ディレクトリ以下のファイルの編集が大変だなあという感じがします。他には、普通にコンパイルするのはうまくいくのに、パッケージにするとうまくいかなかったりとか。

2.8 小室 文さん

apt に足りなさそうな機能

/etc/apt/source.list を触る機会はあまりないですけど、最初（インストール時）に apt-spy みたいなのが自動で動いてくれて sources.list に欲しい version のミラーサイト書き込んでくれたらインストールの時に便利だなと思います。

雑誌とか本についているインストール CD から入ると、設定していないのでそこでミラーサイトなんだっけ？と検索したりそもそも記述方法が分からなかったり、とインストール or アップグレード時につまづく（かもしれない）要素を自分の過去を振り返って思いました。

他の人はそうでもないのかもしれないけど …

3 Debian Weekly News trivia quiz

上川 純一



ところで、Debian Weekly News (DWN) は読んでいますか? Debian 界限でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、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 DWN を読んでみましょう。

漫然と読むだけではおもしろくないので、DWN の記事から出題した以下の質問にこたえてみてください。後で内容は解説します。

3.1 2007 年 01 号

<http://www.debian.org/News/weekly/2007/01/> にある 1 月 23 日版です。

問題 1. creative commons 2.5 は DFSG 互換か?

- A ちがう
- B DFSG 互換です
- C むしろ GPL とまったく同じ

問題 2. Kenshi Muto のアナウンスによると、Takeshi Yaegashi は何に Debian をインストールするのに成功した?

- A PLAYSTATION 3
- B Wii
- C XBox 360

問題 3. Florian Lohoff はどんな変化に気付いたか?

- A woody がミラーから取り除かれた
- B sarge のアーカイブに侵入された形跡がある
- C etch への開発者の興味が薄れている

問題 4. Joseph Smidt はリリースに関して何を提案したか?

- A unstable と testing だけをサポートするようにして保守作業を楽にしよう
- B etch はリリースされないまま古くなりつつあるので適当にリリースして lenny に全力を注ぎ込もう
- C そろそろ Debian も XP や Vista といったよくわからない名前をリリースにつけるようにしよう

3.2 2007 年 02 号

<http://www.debian.org/News/weekly/2007/02/> にある 1 月 30 日版です。

問題 5. Roberto C. Sanchez が提案したのは何か

- A 全員が自分の誕生日を Debian として祝うべきだ
- B Debian のスクリーンショットのレポジトリを準備して、それぞれのパッケージに対応させる
- C Debian を全員使うべきだ

問題 6. Robert Millan が作成した win32 用プログラムは?

- A Windows Vista を自動的にダウンロードし上書きインストールしてくれるインストーラ
- B ユーザの手間を省くために、Outlook Express のアドレス帳に載っているアドレスを lists.debian.org のメーリングリストすべてに自動登録してくれるプログラム
- C Debian Installer を自動的にダウンロード・起動してくれるランチャー

問題 7. 1 月 31 日に締め切られたのは?

- A 第 24 回東京エリア Debian 勉強会の blog での報告
- B Debian etch ベータ版のテスターの募集
- C Debian Conference 7 のスポンサー希望者の参加登録

問題 8. Luis Matos が etch のカーネルに関して提案したのは?

- A kvm が利用できるように 2.6.20 を使おう
- B Linux ではなく kFreeBSD を使おう
- C ハードウェアサポートのためにポイントリリースでカーネルの更新を行おう

3.3 2007 年 03 号

<http://www.debian.org/News/weekly/2007/03/> にある 2 月 13 日版です。

問題 9. Debian のウェブサイトについて Manoj Srivastava が気付いたのは?

- A デザインがやや古びており見た目がイマイチ
- B 投票ページのナビゲーションバーが長すぎる
- C 朝起きたらぐるぐるマークが逆回転になっていた

問題 10. 今年の Debian Project Leader 選挙のアナウンスから 4 時間。最初にノミネートしてきたのは?

- A Junichi Uekawa
- B Gustavo Franco
- C Bill Gates

問題 11. Debian Conference 2008 はどこの国で開催されることに決定したか?

- A イラク
- B アルゼンチン
- C パプアニューギニア

4 最近の Debian 関連のミーティング報告

上川 純一



4.1 東京エリア Debian 勉強会 24 回目報告

1 月の第 24 回 Debian 勉強会を実施しました。2007 年の Debian 勉強会を企画する会です。

今回の参加人数は 15 人でした。小林さん、小室さん、Henrich さん、石原さん、高杉さん、えとーさん、前田さん、河内さん、岩松さん、キタハラさん、吉田さん、あけどさん、吉藤さん、Charles Plessy さん、上川でした。

上川が最近の事情の紹介、事前課題の紹介をしました。参加者の方々の 2007 年へのいきごみが伝わってきて、圧倒されました。

岩松さんが apt-torrent について紹介しました。apt-get を bittorrent 経由で実施するアプリケーションの紹介です。アイデアはおもしろいのですが、現状のインフラが整備できていないので、そこを改善してみたいということだそうです。

上川が kvm という仮想マシンモニタについて紹介しました。Windows と RedHat Linux が Debian 上で動作しているデモをしました。2.6.20 カーネルがリリースされたら一般的になると思いますので、みなさまよろしくお願ひします。

グループワークで、2007 年度の Debian 勉強会の計画を立案してもらいました。熱い議論が展開されましたね。はたして今年を振り返ってみたときにこの成果が反映されているか、みものです。

時間がなくて、小林さんの quilt の話はきけませんでした。次回にもちこしということで。

宴会は今回も荻窪 うさぎにて。素敵なお店でした。次回の幹事は小林さんです。

5 Debian 勉強会 2007 年度 計画検討結果

上川 純一

2007 年度の計画を検討した結果次のような内容が出てきました。
Debian が現在市場に提供している付加価値としては下記がある。

- デプロイしやすいしくみであること、
- ライセンスが明確であること
- たくさんソフトウェアがあること
- アップグレードが平穩であること
- くだらないパッケージもはいつていること
- ソースコードが簡単にとってこれること
- ユーザが多いこと
- 学ぶことがおおいこと

また外部的な要因として

Windows VISTA の公開 テスト開始

Mac OSX iPhone, Leopard

他のディストリビューション FC, SUSE, RHEL の新リリースが出る、Ubuntu も出るが、宇宙に再度いつてしま
うのではないか？

ハードウェア Quad core, Wii/PS3/.. などが出る

ユーザの期待として 次はさすがに GUI で D-i できるだろう

というのがある。

ユーザの拡大方向としては、下記の案が出てきた。

主婦

学生 教育に Debian を利用すべきだ

また、今年の活動は、来年を考えて動くべきだろう。

以上をふまえての今年度の計画は下記で実施する。

- 2 小林さん幹事
- 3 岩松さん担当で OSC
- 4 えとーさんでエッチインストール大会
- 5 ごとむさん主催で google 開催 (!?)
- 6 やぶきさんでえじんばら開催
- 7 Debconf 参加報告会
- 8 Debian 14 周年

9以降 後で考える。

6 dpatch について的小ネタ

上川 純一

dpatch を subversion などといっしょに使うときにもしかすると便利かもしれない小ネタについて御紹介します。

6.1 インストールのしかた

まず最初にインストールのしかたですが、Debian システムであれば、

```
apt-get install dpatch
```

でインストールできます。そうでない場合のインストールについては想定外です。

6.2 作業のしかた

debian/rules を適切に変更したあと、dpatch-edit-patch で編集します。^{*2}

パッチは debian/patches 以下で管理されます。debian/patches/00list ファイルに適用するパッチの一覧があり、それを参照してパッチを適用します。

6.3 dpatch に含まれる各種ツールの紹介

各種ツールがあるので何をやるものなのか、見てみましょう。

dpatch メインのツールです。表立って直接利用することはありません。debian/rules などから呼び出されます。

dpatch-list-patch パッチの一覧を出力します。

dpatch-edit-patch 編集のツールです。パッチを作成、もしくは編集する際に利用します。dpatch-edit-patch パッチ名 適用するパッチ という形で指定すると、指定した適用するパッチまでが適用された状態のソースツリーが一時ディレクトリに展開された状態でシェルが起動します。そこで編集し、シェルを終了 (ctrl-D もしくは exit) するとパッチが作成され、debian/patches 以下にファイルが生成されます。

dpatch-convert-diffgz .diff.gz から dpatch ファイルを生成するためのツールです。

dpatch-get-origtargz dpatch-edit-patch が内部的に利用するツール。

6.4 debian/ディレクトリのみを展開したパッケージのメンテナンスをする

dpatch で管理すると、Debian のソースパッケージのうち、debian/以下のディレクトリに対しての変更しか発生しなくなります。orig.tar.gz を展開した状態でのパッケージのメンテナンスは実は必要なく、debian/ 以下だけをバージョン管理ツールで管理して、orig.tar.gz は必要に応じてアップストリームからダウンロードしてくるというスタイルで開発をすることができます。

dpatch-get-origtargz はそのためのツールで、現在の debian/ ディレクトリに対応した .orig.tar.gz を環境変数

^{*2} /usr/share/doc/dpatch/examples/rules/rules.dh.gz 参照

DPGO_ORIGTARDIR で指定したパス上に存在する .orig.tar.gz から探してきます。ローカルで見付からない場合には uscan や apt-get source などを利用して、ネットワークからダウンロードしてきてくれます。

```
[12:13:55]dancer64:ecatest> ls -l
合計 0
drwxr-xr-x 5 dancer dancer 700 2007-02-17 12:09 debian
[12:12:59]dancer64:ecatest> dpatch-edit-patch -b 12_users_guide.dpatch
dpatch-edit-patch: * /tmp/ecatest/debian/patches/12_users_guide.dpatch exists, this patch will be updated.
dpatch-edit-patch: * debian/-only layout selected
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています... 完了
1129kB のソースアーカイブを取得する必要があります。
取得:1 http://glantank sid/main ecasound2.2 2.4.4-6 (tar) [1129kB]
1129kB を 0s で取得しました (13.2MB/s)
dpatch-edit-patch: * unpacking ecasound2.2_2.4.4.orig.tar.gz
dpatch-edit-patch: * Copying /tmp/ecatest to reference directory.
dpatch-edit-patch: * Cleaning /tmp/dpep-ref.Nn8155/ecatest
dpatch deapply-all
12_users_guide not applied to ./ .
11_configure_in_alsa_fix not applied to ./ .
07_configure_in_maintainer_mode not applied to ./ .
rm -rf patch-stamp patch-stampT debian/patched
dh_testdir
dh_testroot
rm -f build-stamp configure-stamp
# Add here commands to clean up after the build process.

[中略]
12_users_guide not applied to ./ .
11_configure_in_alsa_fix not applied to ./ .
07_configure_in_maintainer_mode not applied to ./ .
dpatch-edit-patch: * Applying patches
applying patch 07_configure_in_maintainer_mode to ./ ... ok.
applying patch 11_configure_in_alsa_fix to ./ ... ok.
dpatch-edit-patch: * Copying reference directory /tmp/dpep-ref.Nn8155/ecatest to work directory.
dpatch-edit-patch: * Applying current 12_users_guide.dpatch for editing.
applying patch 12_users_guide to ./ ... ok.

dpatch-edit-patch:

Now launching an interactive shell in your work directory. Edit your files.
When you are done, exit the shell. When you exit the shell, your patch will be
automatically updated based on the changes in your work directory.

If you wish to abort the process, exit the shell such that it returns an exit
code of "230". This is typically done by exiting the shell with the command
'exit 230'.
[12:13:09]dancer64:ecatest>
```

6.5 svn-buildpackage との連携

svn-buildpackage と dpatch の連携については特に考慮されているわけではありません。ただ、連携して利用できないわけではありません。利用するには下記の点を設定すればよいでしょう。

svn-buildpackage も、debian/ ディレクトリのみを別管理にする方法をサポートしています。それを利用する際に注意する点として、orig.tar.gz の場所を指示するために、~/svn-buildpackage.conf に次のような行をいれます。

```
svn-override=origDir=$HOME/DEBIAN/svn/tarballs
```

この場所から svn-buildpackage は .orig.tar.gz を探してくれます。

この設定にあわせて、dpatch も同じ場所から orig.tar.gz を探すように環境変数を設定します。

```
DPGO_ORIGTARDIR=$HOME/DEBIAN/svn/tarballs
```

すると、svn-buildpackage と dpatch が連動して動作するようになります。

-b オプションを付けて dpatch-edit-patch を実行すると、orig.tar.gz を DPEP_GETORIGTARGZ を参照して探してくれるようになります。

6.6 参考文献

- 2005 年 7 月 Debian 勉強会資料
- svn-buildpackage HOWTO /usr/share/doc/svn-buildpackage/HOWTO.pdf
- svn-buildpackage のメモ <http://tach.arege.net/trac/wiki/Debian/svn-buildpackage>
- svn-buildpackage で .deb パッケージのバージョン管理 <http://www.j96.org/~yuya/d/20041128.html#>

p02



7 dbs

岩松 信洋

7.1 dbs とはなにか

dbs は Debian Build System の略です。dpatch や quilt はパッチを管理する方法に重点を置いているのですが、dbs は名前のとおり、ビルドまでの面倒を見るためのツールです。使いかたとしては、dbs も dpatch もあまり変わりません。debian/rules で専用のライブラリを include して使うだけです。ただ、作法がありところどころ違うところもあります。今回は dpatch と比べてどこが違うのかを比べてみようと思います。

7.2 インストールのしかた

```
# apt-get install dbs
```

7.3 使いかた

dbs の使うためのサンプルとして hello-dbs^{*3} というパッケージが存在します。これを使ってどのように dbs を使うのか見ていこうと思います。

7.3.1 hello-dbs を取得

hello-dbs のソースパッケージを取得します。

```
# apt-get source hello-dbs
```

7.3.2 展開されたソースパッケージ

展開されたソースパッケージ内をみると、以下のような構成になっています。

```
iwamatsu@chimagu:~/dev/debian/dbs/hello-dbs-1.3$ ls
debian hello-1.3.tar.gz
```

ここでわかる dbs を使ったパッケージの特徴として、ソースパッケージは tar.gz 形式で提供されている点です。dh_make を使って生成されたパッケージではこのような構成にはなっていません。upstream のソースパッケージは Debian 標準の形ではないということです。

しかし、debian ディレクトリは、他のパッケージの構成とあまり変わりません。

```
iwamatsu@chimagu:~/dev/debian/dbs/hello-dbs-1.3$ ls debian/
README.Debian README.build changelog compat control copyright dirs hello.1 patches rules
```

7.3.3 debian/rules ファイル

dbs では debian/rules に設定項目を書くことによって、細かい設定を行うことができます。hello-dbs では以下の設定を行っています。

^{*3} <http://packages.debian.org/unstable/dev/hello-dbs>

```

# DBS options

package      := hello-dbs
PWD          := $(shell pwd)
CFLAGS      := -O2 -Wall
INSTALL = install
INSTALL_DATA := $(INSTALL) -m644
INSTALL_DIR  := $(INSTALL) -p -d -o root -g root -m 755
INSTALL_FILE := $(INSTALL) -p -o root -g root -m 644
INSTALL_PROGRAM := $(INSTALL) -m755
INSTALL_SCRIPT := $(INSTALL) -p -o root -g root -m 755
SCRIPT_DIR   = /usr/share/dbs

# the dbs rules
TAR_DIR := hello-1.3.orig
include $(SCRIPT_DIR)/dbs-build.mk

```

以下に各設定項目の内容を示します。

- package
パッケージ名
- PWD
パッケージカレントディレクトリ
- CFLAGS
C コンパイラに設定するオプション
- INSTALL_DATA
インストールするデータのパーミッション
- INSTALL_DIR
インストール先ディレクトリのパーミッション
- INSTALL_FILE
インストールするファイルのパーミッション
- INSTALL_PROGRAM
インストールするプログラムのパーミッション
- INSTALL_SCRIPT
インストールするスクリプトのパーミッション
- SCRIPT_DIR
dbs スクリプトディレクトリパス
- TAR_DIR
tar 解凍後ディレクトリ名

`include $(SCRIPT_DIR)/dbs-build.mk` で `include` することにより、`dbs` を使うことができるようになります。

7.3.4 パッチ

`dbs` では `dpatch` と同様、`debian/patches` ディレクトリにパッチを格納する必要があります。パッチはユニファイド `diff` 形式 で書かれている必要があり、ファイル名の先頭には数字を付けます。数字とファイル名の間はアンダースコアで区切ります。特に拡張子等は必要ありません。`dbs` はこの数字が割り当てられている順にパッチをソースコードに適用していきます。

`dpatch` では `00list` というファイルにパッチ名を書き、書かれた順にパッチを適用していきます。パッチの順番が変わったときは、`00list` ファイルを修正すればいいだけですが、`dbs` の場合はパッチの順が変わったときはファイル名を変更しないといけません。

また、`hello-dbs` パッケージでは `00_maillocation` パッチがあります。

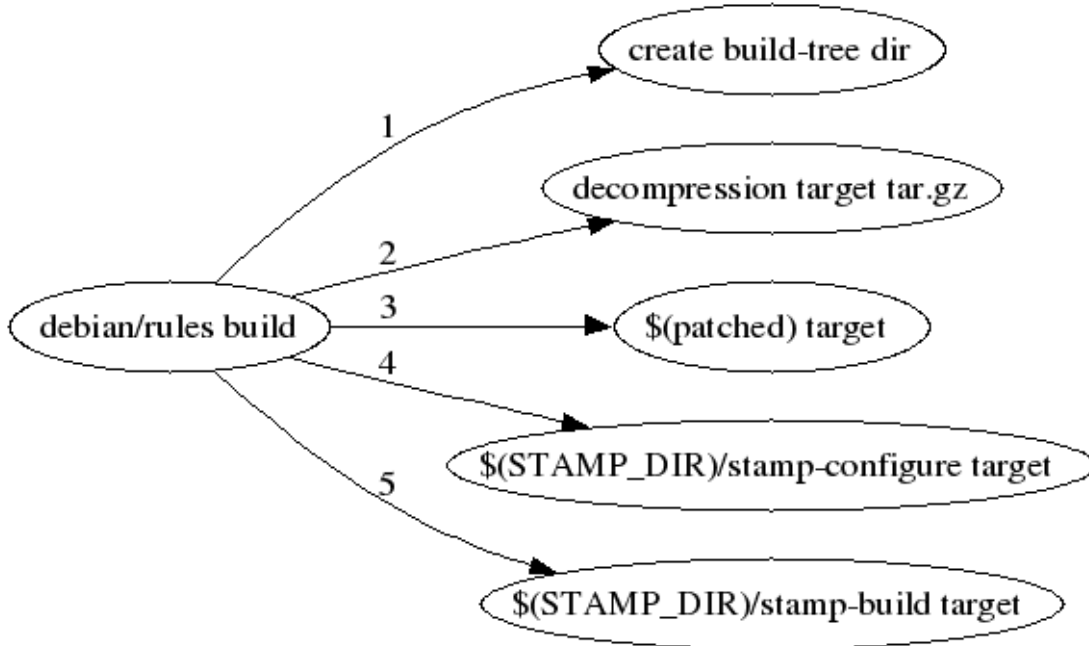
```

% ls -l debian/patches
00_maillocation

```

7.3.5 パッケージのビルド

パッケージのビルドが行われるとき、以下の図の順でターゲットが呼ばれます。



1. create build-tree ディレクトリ

`$BUILD_TREE` で指定されたディレクトリを作成します。デフォルトでは `build-tree` になっています。

2. decompression tar.gz

ソースコードが圧縮されている `tar.gz` を `build-tree` ディレクトリに解凍します。

3. \$(patched) target

`debian/patches` ディレクトリにあるパッチを適用します。

4. \$(STAMP_DIR)stamp-configure target

`stamp-configure` ターゲットを実行します。hello-dbs では `$BUILD_TREE` ディレクトリに移動し、`configure` を実行します。

5. \$(STAMP_DIR)/stamp-build target

`stamp-build` ターゲットを実行します。hello-dbs では `$BUILD_TREE` ディレクトリに移動し、`make` を実行します。

build 時の `dpatch` との違いは、

- パッチを当てるためのターゲット名が異なる

`dbs` は `$(patched)`、`dpatch` は `patch-stamp`。

`$(patched)` は `/usr/share/dbs/dbs-build.mk` 内で `$(STAMP_DIR)/patchapply` と宣言されています。

- マーク用のファイル名が異なる

`stamp-configure` や `stamp-build` というマーク用のファイル名が逆だったりします。(通常は `configure-stamp` / `build-stamp`)

7.3.6 パッケージの clean

`dbs` のソースの `clean` ターゲットはいたってシンプルです。`dpatch` は既に展開されているソースにパッチを適用するため、`clean` ターゲット時には適用されたパッチを外す処理が必要になりますが、`dbs` では、build 時に生成されたマークファイル用ディレクトリやソース格納先ディレクトリ (`$BUILD_TREE`) をばっさり削除します。これには、

適用されたパッチの管理等を行わずに済むというメリットがあります。

しかし、dbs は パッケージビルド毎に

1. ディレクトリを削除
2. tar.gz を解凍
3. パッチ適用

とするので、サイズの大きい tar.gz をパッケージ化するときは時間がかかるというデメリットもあります。

7.4 パッチの変更

dbs でパッチの変更を行うために以下のコマンドが提供されています。

7.4.1 dbs-edit-patch

このコマンドは現在のソースからパッチを作成する環境を構築してくれます。hello-dbs で新しいパッチを作りたいという状況になったとします。以下のコマンドを実行します。

```
% dbs-edit-patch 01mogeri_patch
```

実行すると /tmp ディレクトリに 01mogeri_patch というディレクトリが作成されます。ディレクトリの中は以下のようになっています。

```
% ls -l /tmp/01mogeri_patch/
-rwxr-xr-x 1 iwamatsu iwamatsu 546 2007-02-15 06:00 dbs-update-patch
drwxr-xr-x 2 iwamatsu iwamatsu 1024 1996-08-04 23:48 hello-1.3.orig
drwxr-xr-x 2 iwamatsu iwamatsu 1024 1996-08-04 23:48 hello-1.3.orig-old
```

hello-1.3.orig は修正対象のディレクトリで、hello-1.3.orig-old は修正前のディレクトリです。hello-1.3.orig を修正した内容と hello-1.3.orig-old で差分を取り、パッチを生成し、パッチをコピーしてくれるスクリプトが dbs-update-patch になっています。

```
#!/bin/sh -e
cd "/tmp/01mogeri_patch"
HOOK_DIR=/home/iwamatsu/dev/debian/dbs/hello-dbs-1.3/debian/dbs-hooks
test -d "$HOOK_DIR" && run-parts "$HOOK_DIR" --arg update-patch-prediff
find -name "*.bak" -print0 | xargs -0 --no-run-if-empty rm
find -name "*" -print0 | xargs -0 --no-run-if-empty rm
: > new_patch
diff -rUN hello-1.3.orig-old hello-1.3.orig >> new_patch || test $? -eq 1
mv new_patch "/home/iwamatsu/dev/debian/dbs/hello-dbs-1.3/debian/patches/01mogeri_patch"
test -d "$HOOK_DIR" && run-parts "$HOOK_DIR" --arg update-patch-postdiff
```

例えば、

```
cat /tmp/01mogeri_patch/hello-1.3.orig/MOGERI
mogemogeri
```

という修正を行い、/tmp/dbs-update-patch を実行した場合、

```
% /tmp/01mogeri_patch/dbs-update-patch
% ls -l debian/patches/01mogeri_patch
-rw-r--r-- 1 iwamatsu iwamatsu 212 2007-02-15 06:08 debian/patches/01mogeri_patch
% cat debian/patches/01mogeri_patch
diff -rUN hello-1.3.orig-old/MOGERI hello-1.3.orig/MOGERI
--- hello-1.3.orig-old/MOGERI    1970-01-01 09:00:00.000000000 +0900
+++ hello-1.3.orig/MOGERI        2007-02-15 06:06:54.000000000 +0900
@@ -0,0 +1 @@
+mogemogeri
```

となります。

7.5 まとめ

今回、dbs を触ってみたのですが、

- ソースが見えない
ソースが tar で固まっているため、見るができない。見るには コマンドを使って解凍する必要がある。
- dpatch と比べてソースの編集がしづらい

コマンドは用意されているんだけど、一回 tmp 等に持って行って修正して.... という作業が発生するので、めんどくさいところがある。

という感想です。また、dbs を使っていたパッケージも だんだん dpatch に移行しつつあるのでもう役目は追えたのではないかという気がします。

8 次回

未定です。内容は本日決定予定です。
参加者募集はまた後程。



下ヒアノ勉強会



Debian 勉強会資料

2007年2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）
