

月刊

Debian 専

日本唯一のDebian専門月刊誌

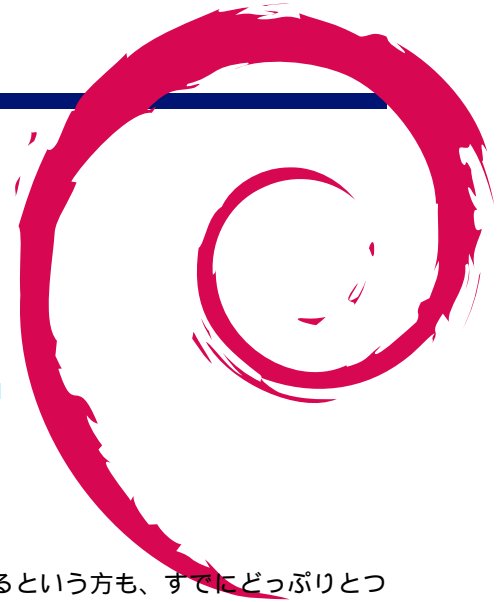
2012年1月21日

特集: クラウド時代に生きる



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？

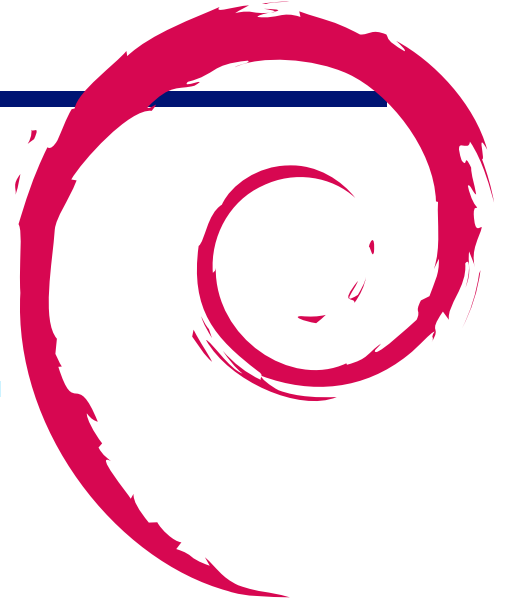
Debian 勉強会の目的は下記です。

- 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 をがりがりとす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。

今年 Debian 勉強会レポート

目次	
1	Introduction 1
2	事前課題 3
2.1	上川 純一 3
2.2	まえだこうへい 3
2.3	岩松 信洋 3
2.4	koedoyoshida 3
2.5	dictoss(杉本 典充) 3
2.6	yamamoto 3
2.7	野島 貴英 4
2.8	野首 4
3	最近の Debian 関連のミーテ ィング報告 5
3.1	東京エリア Debian 勉強会 83 回目報告 5
3.2	東京エリア Debian 勉強会ア ンケート集計 5
4	Debian Trivia Quiz 6
5	Debian 勉強会予約システム 再訪 8
5.1	脆弱性 8
5.2	可能性と脆弱性のバランス 9
6	Debian の使える VPS を使っ てみた 10
6.1	さくらインターネットの VPS 10
6.2	Amazon AWS EC2 10
6.3	S@@Ses 11
6.4	まとめ 11
6.5	さいごに 11
7	Debian で twitter 連携 12
7.1	Twitter API について 12
7.2	Debian での Twitter API サ ポート状況 12
7.3	Debian から Twitter API を 使ってみる 13
7.4	私が使っているツール紹介 13
7.5	まとめ 15
8	2012 年計画妄想会議 16
9	月刊 Debhelper 第 3 回 17
9.1	はじめに 17
9.2	今月のコマンド: dh&dh_auto_* - シーケンス とビルドシステム 17
9.3	今月のコマンド: dh_builddeb 20
10	索引 22



2 事前課題

岩松 信洋

今回の事前課題は以下です:

1. 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。
2. 自分がどのテーマを担当したいか提案してください。

この課題に対して提出いただいた内容は以下です。

2.1 上川 純一

テーマ案です。

- Android/iOS を Debian から使う (USB device driver, wifi, bluetooth, adb などなど)
- 電話で debian を使う
- タブレットで Debian を使う
- Debian からクラウドサービスを使う (EC2 API とか)
- ウェブアプリケーション開発を Debian でする
- Debian のウェブブラウザ
- nodejs
- Debian での javascript の開発・パッケージ
- clojure / Debian

2.2 まえだこうへい

- Debian で OpenStack
- Python2, 3、両方に対応したパッケージ作成
- Python のテストツール
- Vyatta ネット (Debian ベース、という意味で)
- webOS ネット (Ubuntu ベース、という意味で)

2.3 岩松 信洋

- Debian で AR
- Debian で Arduino
- Debian で動画配信
- Debian 専用ルール/方言をまとめてみた (Apache の設定など)

2.4 koedoyoshida

- 2.4.1 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。
予定は未定
- 2.4.2 自分がどのテーマを担当したいか提案してください。
出来そうなことを出来る範囲で

2.5 dictoss(杉本 典充)

- 2.5.1 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。
トランジションの流れ (自分がよくわかってないので) Debian インストーラ (kfreebsd だとよくコケるので要所をつかみたい)
- 2.5.2 自分がどのテーマを担当したいか提案してください。
Debian インストーラ?

2.6 yamamoto

- 2.6.1 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。
聞きたいネタ、ランキング 1 位 debhelper 9、ついにリリース。ずばり徹底解説。2 位 みんな、新しいフリーソフトウェア、どこで見つけてる? 3 位 ここが出る! DD 試験傾向

と対策。

2.6.2 自分がどのテーマを担当したいか提案してください。

むー、実に難しい課題ですな w

2.7 野島 貴英

2.7.1 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。

2月: Desktop KDE、3月:Desktop GNOME、4月:Desktop その他、5月: WEB とクラウド環境、6月: DB と KVS、7月: オフィススイート環境、8月:組み込み用途と変わったデバイス、9月: 学術系ソフト、10月: スマホアプリ開発、11月: マルチメディア/CG、12月: 作ったもの LT 大会と、キャッチーなテーマをいってみるテスト。あ、もちろん全

部 Debian な話で。

2.7.2 自分がどのテーマを担当したいか提案してください。

GNOME と、マルチメディアかな...

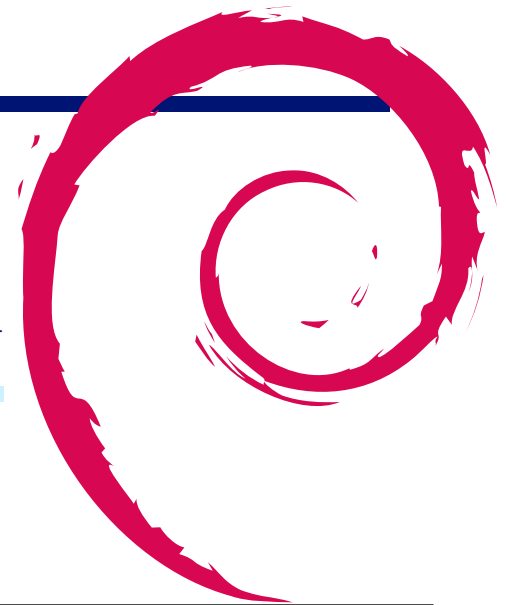
2.8 野首

2.8.1 2012 年の勉強会のテーマとして各月なにをするべきか提案してください。

VCS を使う buildpackage(svn-buildpackage, git-buildpackage 等) の話

2.8.2 自分がどのテーマを担当したいか提案してください。

Windows Sever との連携 (krb5 認証とか、LDAP 検索とか)



3 最近の Debian 関連のミーティング報告

上川 純一

3.1 東京エリア Debian 勉強会 83 回目報告

2011 年 12 月の勉強会は新宿のスクウェア・エニックスさんの会場を借りて忘年会を開催しました。Debian kFreeBSD への porting の話題、そして月刊 Debhelper の話題で盛り上がりました。

3.2 東京エリア Debian 勉強会アンケート集計

Debian 勉強会ではアンケートシステムを導入して、アンケートを集計するようにしました。その結果を集計してみました*1。アンケートの個人による評価値のばらつきを減らすために、アンケートの結果を個人において正規化し分散 1 平均 0 にし、その結果を平均してみました。アンケート参加人数の N が小さいのでこれが適正な処理なのか自信ないです。

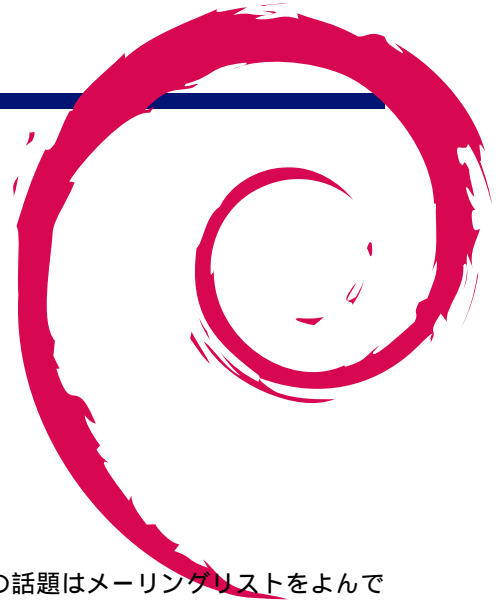
並べてみると、毎回やっている企画(事前課題など)については辛口の評価をつけてそうではない企画の内容についてはよい評価をつける傾向がある気がします。

タイトル	正規化後の平均スコア
Debian パッケージのビルド方法	1.36712095331843
Kinect	1.29031977439283
月刊.Debhelper	0.994361301686173
俺の libsane が火をふくぜ	0.884025671678169
CACert の準備に何が必要か	0.658609157051467
スポンサーアップロード入門	0.607053388754129
Debian とは何なのか.	0.593889881369359
quilt で porting してみた	0.584663154875175
Debian. で.sphinx. と.doxygen. を使う	0.557508487151354
月刊 Debhelper 第 2 回	0.292826734383766
DPN.trivia.quiz	0.268370655108359
事前課題紹介	0.268370655108359
2010 年の Debian を振り返って.2011 年を企画する	0.265782004752358
Debian.JP. 定例会議処理系に XSLT を使ってみた	0.264815114529638
Debian. で快適な.LaTeX. 作業環境	0.253402728051224
レポートの自動生成	0.253402728051224
Debian 勉強会アンケートシステム	0.00396377308743415
2011 年の振り返り	-0.101231927535827
クイズ	-0.1840838183099
最近のイベント紹介	-0.219801079954323
事前課題紹介	-0.325337934747546
CACert.Assure...GPG.keysigning	-0.339146226977589
Debconf11 レポート	-0.341629909296384
クイズ	-0.341629909296384
事前課題紹介	-0.341629909296384
事前課題紹介	-0.356155493492248
事前課題紹介	-0.49526151815082
事前課題発表	-0.86518061156894
DWN.trivia.quiz	-0.994688158876128
Haskell と Debian の辛くて甘い関係	-1.4294572162706
Debian.Miniconf. 企画	-1.67677847288666

*1 /enquete/showallresults が出力する CSV を R で処理

4 Debian Trivia Quiz

岩松 信洋



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. armhf が unstable にはいつだったのいつか

- A 2011-11-24 1952
- B 2013-11-24 1952
- C 2001-11-24 1952

問題 2. s390x が unstable にはいつだったのいつか

- A 2011-11-25 0152
- B 2013-11-25 0152
- C 2001-11-25 0152

問題 3. 1/17 に alioth になにがおきたか

- A `vasks.debian.org` が起動しなくなった
- B `wagner.debian.org` が起動しなくなった
- C SOPA の抗議をはじめた

問題 4. NM process の NM は何を意味することになったか

- A New Maintainer
- B New Member
- C New Moemoe

問題 5. REVU になにがおきるといつているか

- A universe を拡大
- B Debian を必要なくする
- C `mentors.debian.net` に統合

問題 6. トレードマークについての連絡先は

- A `trademark@debian.org`
- B `trade@debian.net`
- C `iwamatsu@debian.org`

問題 7. win32-loader.exe の新機能は

- A Debian GNU/Hurd のインストール
- B Debian GNU/kFreeBSD のインストール
- C Debian GNU/Linux のインストール

問題 8. `wiki.debian.org` の launchpad バグ対応を利用するにはどのタグを使うか

- A UbuntuBug
- B DebianBug
- C Hoge

問題 9. `dh-exec` とはなにか

- A 実行可能な設定ファイルの出力を使う仕組み
- B どんなものでも実行する仕組み
- C 実行、実行、実行

問題 10. Derivatives Census <http://wiki.debian.org/Derivatives/Census> にはなにがかいてあるか

- A Debian の正当な後継者の一覧
- B Debian からの派生物の一覧
- C Debian を disってる人の一覧

問題 11. <http://debtags.debian.net/> のリニューアルでは何をしたか

- A Django と jQuery での書き直し
- B Debian ベースでの再実装
- C ocaml で実装しなおした

問題 12. Debian の監査役としてがんばっているのは誰か

- A Nobuhiro Iwamatsu
- B Stefano Zacchiro
- C Martin Michlmayr

問題 13. kassia と liszt はいくらするのか

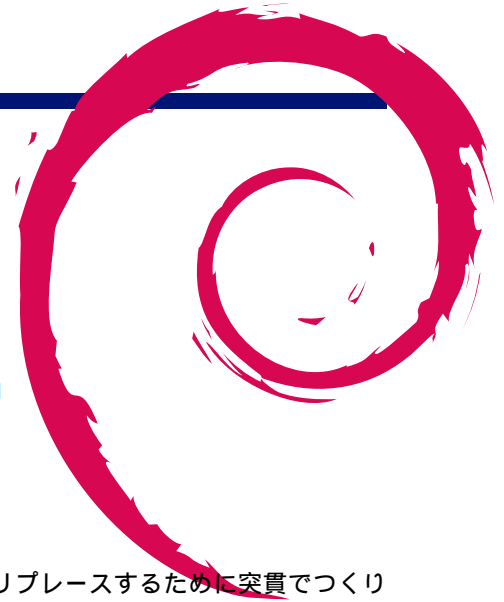
- A 10,000USD
- B 100 万円
- C 11'792.9 EUR

問題 14. Portland BSP で使った sbuild インスタンスはいくらしたか

- A 70USD
- B 700USD
- C 7000USD

5 Debian 勉強会予約システム再訪

上川 純一



Debian 勉強会予約システムの開発開始から 2 年経ちました。当初は宴会君をリプレースするために突貫で作りあげたものですがほぼそのままの状態です。面倒くさ過ぎて適当に実装した部分、ウェブアプリケーションで気をつけるべきところをよくわからずに書いていた部分があったのでいまから振り返ってどういう課題があったのかを検討します。

5.1 脆弱性

5.1.1 POST/GET の使い分け

当初めんどくさかったので post と get のハンドラを一緒にしてました。よって、あらゆるページが GET と POST の両方で動くようになっています。それを適切に GET と POST にわけるようにしました。

GET は情報の取得のみ、POST は登録などの副作用のある処理に利用します。

ブラウザのセキュリティモデルとして、same-origin じゃないと POST がやりにくいようになっています。GET はたとえば script タグや img タグなどを埋め込んでおけばいくらかでも発行できますが、POST は XMLHttpRequest の発行が必要で、XMLHttpRequest には same-origin policy があります。

以前は任意のページから script タグの埋め込みをして勝手に Debian 勉強会に登録することが可能でしたが、今はそれができないようになっています。これは勝手に予約する HTML ページの例です：

```
<html>
  <head>
    <title>auto reserve exploit</title>
    <script
      src="http://localhost:8080/eventregister?eventid=df24a1e1de11c067c461537dce6394e0e51df6ad
      &user_prework=&user_attend=attend&user_enkai_attend=enkai_attend&user_realname=myname">
    </script>
  </head>

  <body>
    <h1>auto reserve exploit</h1>
    <p>
      ぼくはまちちゃん
    </p>
```

5.1.2 HTML escaping

当時よくわからなかったのとめんどくさかったので入力文字列のサニタイズとかエスケープとかまったくしてませんでした。関西 Debian 勉強会では HTML タグを勉強会の予約ページの案内文に入力していたようで、それでこの脆弱性がそもそも存在するのに思い至りました。

任意の文字列を HTML として出力できると、任意の javascript のコードを実行することができます。たとえば、説明ページを開いた瞬間に予約登録するイベントを作成することができます。

```
<script>
var xhr = new XMLHttpRequest();
xhr.open('POST', 'http://localhost:8080/eventregister');
xhr.withCredentials = true
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.send('eventid=df24a1e1de11c067c461537dce6394e0e51df6ad&
+ 'user_prework=%E3%81%BC%E3%81%8F%E3%81%AF%E3%81%BE%E3%81%A1%E3%81%A1%E3%82%83%E3%82%93&
+ 'user_attend=attend&user_enkai_attend=enkai_attend&user_realname=hamachi');
</script>
```

しかたがないので文字列をエスケープするようにしました。副作用として HTML タグを説明文に打ち込むことができないようになっています。

5.1.3 任意の URL 表示

Debian 勉強会予約システムでは説明文の表示が脆弱な代わりに任意の URL を iframe で埋め込み表示できるようにしています。通常管理している勉強会の Wiki ページを埋め込むことで二重に情報を管理しなくて済むようにという意図です。

URL の中には特殊なものもあります。javascript: で始まる URL だとその Javascript のコードを実行することができます。できても普通の役には立たなさそうなので http:// 以外は許可しないようにしておきました。

また、iframe buster と呼ばれる手法により iframe から脱出できるように、そういうページを iframe の中でひらいているつもりになるとユーザが誤解する可能性があります。

すべてのブラウザでサポートされているわけではないのですが、iframe に sandbox 属性をつけておき javascript などの実行を制約しておきました。

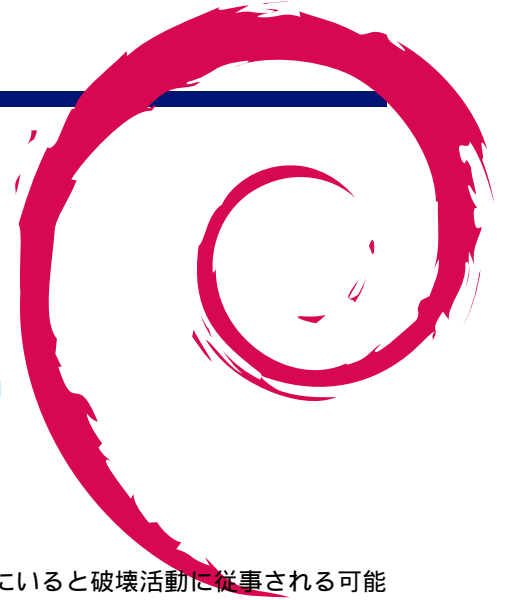
5.2 可能性と脆弱性のバランス

今回いろいろと脆弱性があったのでふさいでみました。しかし、脆弱性を塞ぐのも、ありうる被害とのバランスで考えるべきだと思います。Debian 勉強会に提出する事前課題が漏洩する、勝手に宴会に参加することになっている等の程度の脆弱性と、自由に HTML が記述できる自由とどちらが重要かと考えてみてください。すこし悩ましいと思いませんか？



6 Debian の使える VPS を使ってみた

上川 純一



自宅サーバ最強だと思っていた時期が僕にもありました、が幼少な子供が自宅にいると破壊活動に従事される可能性があり、自宅は設置したマシンを安定してサーバとして稼働させるのに適した環境とはいえません。また、つけっぱなしにしていると音がうるさいので静音 PC 化とかにこだわりますが、そろそろそれにも飽きてきました。静音のためにファンレスにしとくと夏気づいたら熱暴走してます。また、PC は自宅の消費電力の中でも大きな割合を占めてました。

しかし、常時走っているノードがないと、リグレーションテストを走らせることすらできません。我慢できなくなってきたので、Debian ノードで好きなようにいじれるようなサービスを探して試してみました。2012 年 1 月 15 日時点の情報です。近年 IaaS クラウドとかいうバズワードがあってよくわからなくなっているのですが、ここで僕の欲しいサービスは多分 VPS です。

6.1 さくらインターネットの VPS

KVM のノードがひとつ手に入ります。

デフォルトでは Cent OS がはいた状態で起動するのですが、Debian をインストールするメニューがブラウザで利用する管理画面で選択できます。Debian のインストールを選択すると、debian installer が起動し、java applet の VNC クライアント経由でコンソール画面を見ながらインストールします。

良くも悪くも DI ですが、一部カスタマイズされているようで、例えば、ssh がデフォルトでインストールされるようになっています。公開鍵認証にしたい場合はあとで自分で ssh の公開鍵を登録すればいいと思います。

ブラウザで利用する管理画面で再起動とか VNC でコンソール画面に接続するなどのメンテナンスが行えます。

使用料金は「さくらの VPS 512」で 980 円/月です。支払いはクレジットカード以外の方法もありますが、クレジットカードだと二週間の無料お試しが可能なようです。利用を開始すると住所の確認のためにハガキが送られてきてその中に書いてあるお知らせ番号を入力させられます。日本国内に住所があることが利用の条件です。

6.2 Amazon AWS EC2

Xen のノードがひとつ手にはりいます。既存の Debian の AMI ^{*2}[4] をクローンする感じでインスタンスが立ち上がります。自分で AMI をつくってもいいみたいです。

ブラウザで利用する管理画面からインスタンスの再起動などが行えます。

ブート時のログが制御コンソールから見れますが、コンソール接続する方法は見つけれませんでした。起動に失敗したら AMI からつくりなおせばいいという発想なんでしょうか。

初回時の接続は公開鍵認証の SSH で root ユーザとして接続します。ログインはインスタンスを作成するウェブインタフェースにて作成した公開鍵認証です。

^{*2} Amazon Machine Image: / のファイルシステムイメージのようです

使用料金は一時間あたり 10 セント程度で東京リージョンのsmallインスタンスが借りれます。

最初の開始の利用手続きでは、Amazon.com のアカウントをつかって登録します、その際に電話番号を確認のために電話がかかってきます。クレジットカードが必要です。

6.3 S@@Ses

Xen ベースでの VPS サービスを提供しています。LT サーバであれば月 450 円で利用できるよう。初期費用 3000 円かかるというのと 3ヶ月単位の契約だということで僕は試してません。最初に二週間のお試し期間があるようです。

ブラウザで利用する管理画面からインスタンスの再起動などが行えます。

OS の初期化メニューからの OS の選択肢に Lenny や Squeeze があり、それを選択するとインストール済みの Xen のイメージが立ち上がり、root で ssh ログインできるようになります。

6.4 まとめ

参考のため、各社の手頃っぽいエントリーモデルを並べてみました*3。

	さくらの VPS 512	AWS EC2 Small	S@@ses LT
使用料金	980 円 / 月	\$0.10/時間	450 / 月 (3ヶ月単位) + 3000 円初期料金
CPU	仮想 2 core	1 ECU	2.66GHz
メモリ	512MB	1.7GB	512MB
ディスク	20GB	160GB	50GB
仮想化技術	KVM	Xen	Xen
コンソールアクセス	VNC 経由	?	?
Debian 利用	ブラウザで利用する管理画面に D-I 起動するメニューあり、VNC 経由で D-I からインストール	Debian AMI をさがしてきてブラウザで利用する管理画面から起動すると root に ssh 可能なインスタンス	ブラウザで利用する管理画面のメニューから初期化を選択すると root に ssh 可能な状態のインスタンス

6.5 さいごに

ほかにもいろいろサービスがあるみたいですが、Debian が利用できる VPS サービスが日本でも充実してきたのだなという感想です。

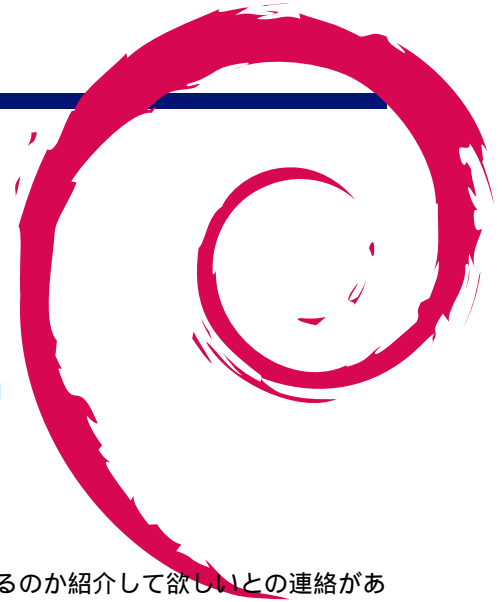
各社ストレージ構成がどうなっているのかバックアップ体制がどうなっているのかなどの記述がみあたらないので、障害発生の確率とか障害発生時の対応がどうなのか予想もつきません。そういう情報は価格競争が一旦収束して、業界が成熟してから重要になるのかもしれませんが。

個人的にはとりあえずさくらインターネットを使ってみることにしてみました。

参考文献

- [1] Amazon Web Services <http://aws.amazon.com/jp/>
- [2] VPS(仮想専用サーバ) のさくらインターネット <http://vps.sakura.ad.jp/>
- [3] SaaS Ses <http://www.saases.jp/>
- [4] Debian Wiki: Cloud Amazon EC2 Image <http://wiki.debian.org/Cloud/AmazonEC2Image>

*3 AWS EC2 が高く見えるけど、micro instance の spot instance にすればもっと安いです。



7 Debian で twitter 連携

岩松 信洋

Debian の作業内容を Twitter に投げている上川さんにどんなことやっているのか紹介して欲しいとの連絡がありました。今回は Debian からどのようにして Twitter を使っているのか、使うにはどのようにしたらいいのか説明します。

7.1 Twitter API について

Twitter は、ユーザーが「ツイート」と呼ばれる 140 文字の「つぶやき」を投稿し、そのツイートを閲覧したりツイートに対してさらにツイートしたりなど、コミュニケーションするためのサービスです。Twitter ではこの「ツイート」を投稿、削除、参照、検索などをプログラムから行えるように API を公開しています。これを Twitter API といいます。Twitter がサービスとして公開しており、利用するためには利用規約に同意する必要があります。

また Twitter API は大きく分けて、REST、Search、Streaming の 3 種類があります。REST API はツイートの更新や参照などを行う基本的な API、Search API はツイートを検索する API、Streaming API はタイムラインをリアルタイムに受け取るための API です。また、これらの API を使うためにはアクセス用の ID が必要で、利用できる API の回数なども決まっています。API を使うにはこの ID を使った認証処理が必要になります。API をラッパーし、Twitter API を使いやすくするための Twitter API 用のライブラリがいくつか存在します。

7.2 Debian での Twitter API サポート状況

まず、Debian での twitter 周りの整備状況を確認してみます。言語毎に Twitter API 用のパッケージが整備され、表 1 のようにパッケージが提供されています。メジャーな言語ではいくつかライブラリがあるようですが、手の足りていない言語チームでは整備が遅れているようです。興味のある方はメンテナンスに参加してみてもいいでしょうか。

表 1 Debian で提供されている各言語用のパッケージ

言語	パッケージ名
C, C++	libsosialweb, bitlbee, etc.
Perl	libnet-twitter-lite-perl, libnet-twitter-perl
Python	python-twyt, python-tweepy, python-twitter
Ruby	libtwitter-ruby1.x
Haskell	なし (パッケージになってない)
OCaml	なし (パッケージになってない)

7.3 Debian から Twitter API を使ってみる

次に Ruby の Twitter API 用ライブラリを使った簡単な例を紹介します。例えば、「test」というツイートをポストするアプリケーションを作成するには以下の順番で行います。

7.3.1 Twitter アプリケーション登録申請

<https://dev.twitter.com/> にアクセスし、OAuth を利用するにあたり必要となる、Consumer key、Consumer secret 等を取得します。これらのデータは Twitter アプリケーション毎に異なります。アプリケーション名を「api-test」とした場合、以下のような内容で適当なファイルに保存します。

```
api-test:
  login: iwamatsu
  oauth_consumer:
    key: XXXXXX
    secret: XXXXX
  oauth_access:
    key: XXXXX
    secret: XXXXX
```

7.3.2 パッケージをインストールする

インストールは apt-get で行えます。

```
$ sudo apt-get install libtwitter-ruby1.9.1
```

7.3.3 Twitter API ライブラリを使ったソースコードと実行

コードは以下のようになります。Twitter アプリケーション登録申請したときに取得した「Consumer key」等を保存したファイルを「/home/hoge/.twitter.yml」、アプリケーション名として指定しインスタンスを生成します。そして status メソッドで「test」をポストするようにします。

```
$ cat test.rb
#!/usr/bin/ruby

require 'twitter'
twitter = Twitter::Client.from_config("/home/hoge/.twitter.yml", "api-test")
twitter.status(:post, "test");

$ ruby ./test.rb
```

以上が Debian から Twitter API を使う例となります。

7.4 私が使っているツール紹介

Twitter はメモや作業内容などを通知をする場合に非常に便利なツールです。Debian の作業内容などを通知できないかなと思っていくつか Twitter 用ツールを作成したので紹介します。

7.4.1 Debian Hack Cafe 通知ツール

毎週東京/関西のどこかで行われていると言われている Debian Hack Cafe。Hack Cafe 開催通知を行うためのアカウントとして @debian_hackcafe があります。これは Debian Hack Cafe GPG キーリングに登録された人なら誰でもつぶやけるとい特徴があります。つぶやく場合、libwww-perl パッケージに含まれる lwp-request を使ってつぶやきをサーバに POST するとサーバで処理が行われ、問題がない場合 @debian_hackcafe アカウントとしてつぶやきます。

```
$ sudo apt-get install libwww-perl
$ echo "つぶやき" | gpg --clearsign | \
  lwp-request -m POST http://www.nigauri.org/debian_hackcafe_post
```

このシステムは以下のように処理されます。

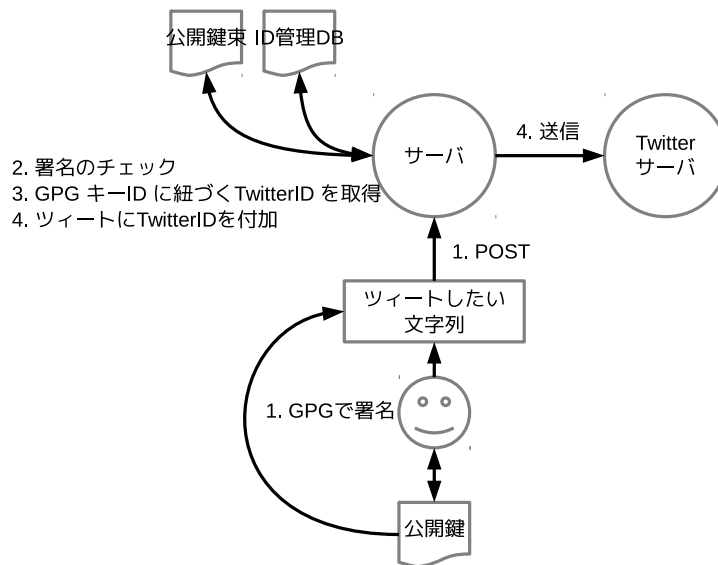


図 1 Debian Hack Cafe 通知ツール構成図

1. つぶやきを GPG サインしてサーバにポスト
2. サーバで鍵のサインをチェック
3. 署名から GPG キー ID を取得し、ID 管理 DB から GPG キー ID に紐づく TwitterID を取得
4. つぶやきに ID をいれて Twitter API を使ってつぶやく

このようにすることで Twitter のアカウントとパスワードを共有することなく、限定されたメンバで、つぶやくことができます。PGP/GnuPG を使って署名チェックするなんて Debian らしくてかっこいい! と個人的に思っています。

また、このシステムを Debian JP アカウント (@debianjp) 用にアップデートし、運営できるようにする予定です (今までは運営している人たちがアカウントとパスワードを共有していたようです)。また Debian JP の誰がつぶやいていたのかわからないという問題も解決する予定です。

7.4.2 パッケージがアップロードされたらつぶやく dput-tweet

最近スポンサーアップロードを行う事が多くなりました。またスポンサーしている人は Twitter のアカウントを持っているので、アップロードした旨を通知する方法の一つとして Twitter を使うようにしました。この通知を行うツールが dput-tweet です。Ruby の勉強用に作ったツールで、dput のラッパーになっており、dput したらパッケージ名とバージョンスポンサーした人の TwitterID をつぶやくというものです (図 2)。このツールのいけてない点として、dput しか対応できてない事と実行時にスポンサーする人の Twitter ID を指定する必要がある事です。

```

$ dput-tweet -s mkouhei ordereddict_1.1-1_amd64.changes
.....
「dput ordereddict_1.1-1 @mkouhei [dput-tweet] 」とつぶやきます。
  
```

もっと楽をしたいと思ったので、今は inotify を使って upload ファイルが作成されたらつぶやく機構にしました。これによって dput / dupload の両方に対応できます。また upload ファイルから changes ファイルを抽出し、Changed-By の行から 得たメールアドレスを元に TwitterID を DB から取得し、つぶやきに入れるようにしました (図 3)。

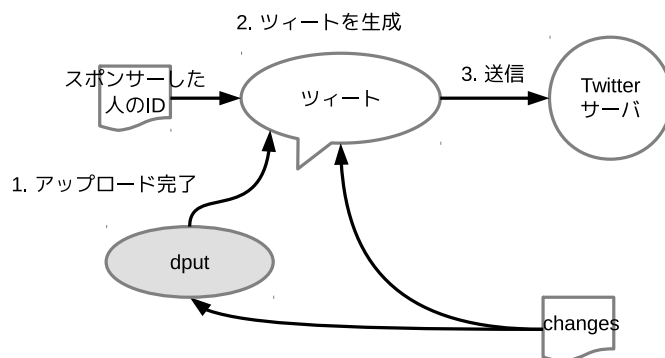


図2 パッケージアップロード通知ツール dput-tweet v1

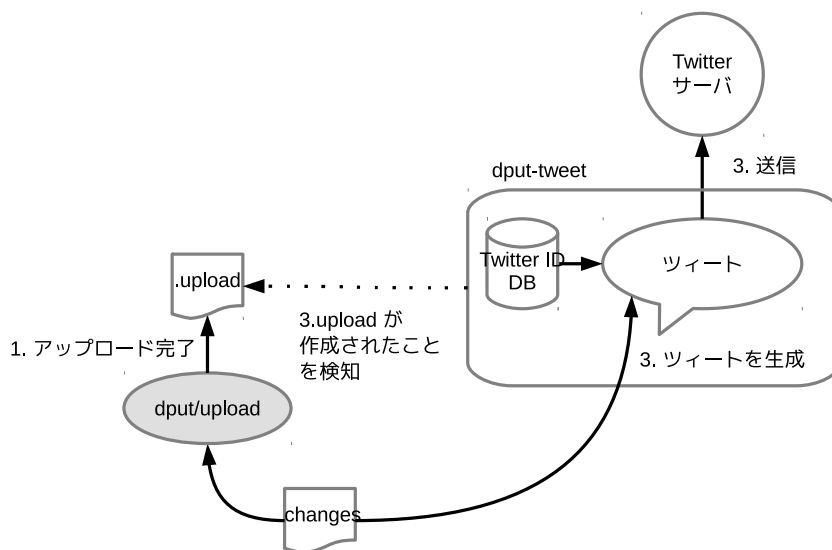


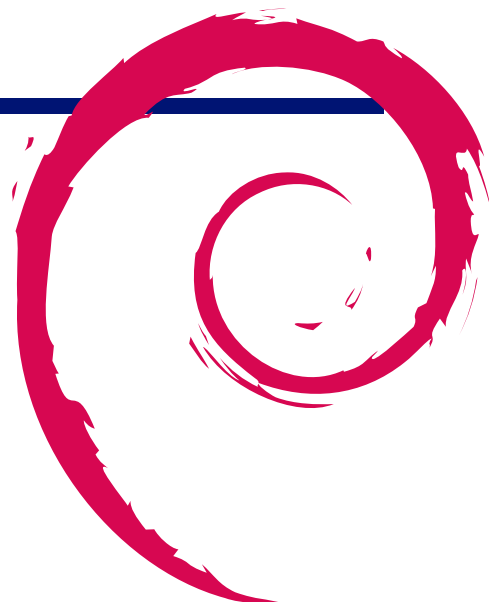
図3 パッケージアップロード通知ツール dput-tweet v2

7.5 まとめ

いくつかのツールを作ってみて、Twitter API の使い方がわかってきました。今度は Debian JP メンバの活動 Tweet などができるようにしたり、Twitter だけでなく Facebook などのサービスの API についても調べてみようと思います。

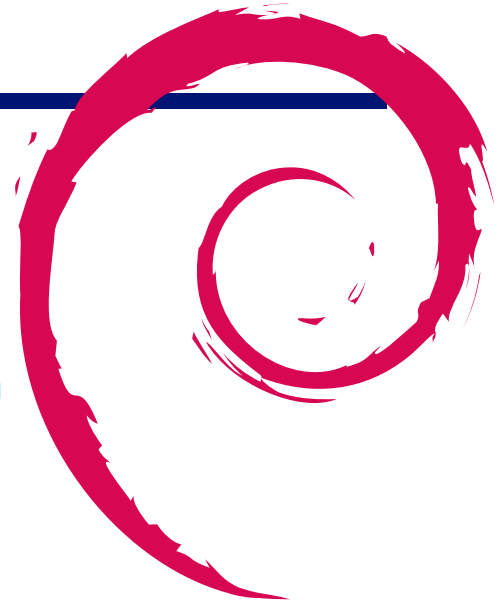
8 2012 年計画妄想会議

上川 純一



2012 年の Debian 勉強会の計画について妄想します。
なにをするべきか提案してください。

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____



9 月刊 Debhelper 第 3 回

山田 泰資

9.1 はじめに

パッケージビルド手順を記述する `debian/rule` ファイル。これを簡潔化するために `debhelper` コマンド群 (`dh_*`) がありますが、その一方で裏側で一体何が行なわれているのか掴み難くなってしまいました。

本企画ではこのコマンド群を、毎月持ち回りで解説します。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指名できるというルールで進めていきます。

9.2 今月のコマンド : `dh` & `dh_auto_*` - シーケンスとビルドシステム

`dh` コマンドの全体像については前回の野島さんの発表で既に解説されているのですが、そこで「試しに読んでみるとよいと思います」とあったので、実際に読みつついじってみました。その中でももう少し判ったことがあったので報告します。

9.2.1 `dh` の動作、再まとめ

`dh` コマンドを実行すると、デフォルトでは以下のコマンド群が各シーケンス毎に呼ばれます :

シーケンス名	シーケンスで実行されるコマンド
<code>clean</code>	<code>dh_testdir dh_auto_clean dh_clean</code>
<code>build</code>	<code>dh_testdir + (rules build-arch build-indep)</code>
<code>build-indep</code>	<code>dh_testdir dh_auto_configure dh_auto_build dh_auto_test</code>
<code>build-arch</code>	<code>dh_testdir dh_auto_configure dh_auto_build dh_auto_test</code>
<code>install</code>	<code>(rules build install-arch install-indep) + dh_testroot dh_prep dh_installdirs dh_auto_install dh_install dh_install* dh_bugfiles dh_ucf dh_lintian dh_gconf dh_icons dh_perl dh_usrlocal dh_link dh_compress dh_fixperms</code>
<code>install-indep</code>	<code>(rules install-indep) + <上の install と同じ></code>
<code>install-arch</code>	<code>(rules install-arch) + <上の install と同じ></code>
<code>binary</code>	<code>(rules install binary-arch binary-indep)</code>
<code>binary-indep</code>	<code>(rules install-indep) + dh_installdeb dh_gencontrol dh_md5sums dh_builddeb</code>
<code>binary-arch</code>	<code>(rules install-arch) + dh_strip dh_makeshlibs dh_shlibdeps + <上の binary-indep と同じ></code>

表 2 `dh` の各シーケンスで実行されるコマンド

かつては `rules`(`Makefile`) に羅列されていたコマンド群が、今は `dh` の中に Perl のリスト変数で管理されている形

になります。そして debhelper モジュール (*.pm) がロード時にリストの内容をいじって実行内容を変更することでビルド過程をカスタマイズします。

わざわざ make を使わず自前なのは、実行内容を変更するというモジュール機構と override 機構が make ベースでは依存関係ツリーの変更になり実現困難だったからでしょうか。たしかに拡張 make でも使わない限り難しそうです (拡張 make まで行きたくないから現状の実現方法 … なんでしょうか)。

さて、今回の話題は、このモジュール機構になります。普段何気なくどんなパッケージでもパッケージビルドされている訳ですが、言語環境や開発者の選択によってビルド方法は千差万別です。これはどうやって吸収されているのでしょうか？

9.2.2 2つのモジュール：シーケンスとビルドシステム

ここで登場するのが「シーケンス」とは別の、「ビルドシステム」モジュールになります。この2つは

- シーケンスモジュールは (主に) 前処理・後処理を追加し、適切なパッケージビルドが行われるようにする
- ビルドシステムモジュールは各ソースパッケージに内包されるビルド方式を自動認識し、それを駆動する

と異なる役割を持ちます。典型的な configure&make パターンで説明すると、

1. Sequence/autotools_dev.pm が config.sub/config.guess を最新に更新
2. Buildsystem/autoconf.pm が ./configure を発見・実行
3. Buildsystem/makefile.pm が Makefile を認識し、make でビルドやインストール

といった連携リレーになります (autotools_dev は-with で有効化された場合のみ)。

9.2.3 ビルドシステムの選択と連携

さて、ビルドシステムは以下のフローで自動選択されています：

1. まず、Buildsystem/*.pm は全部ロードする
2. 各モジュールの check_auto_buildable API にて「ビルドできる度」を照会
3. 同じクラス階層の中で一番大きい「ビルドできる度」を返したものを選択

ポイントは

- この自動選択は configure/build/test/install/clean の各段で毎回行われる
- 同じクラス階層縛りがある

の2点です。つまり、

- 毎回行われるので、各段で応答・不応答を変えてモジュール間連携を行う
- 自分が優先されるべき場合、親クラスの結果を上回るようにして勝つ

とする必要があり、「単純にビルドできるから真値を返す」という実装ではないのです。これは自前のビルドシステム拡張、特に他と連携する場合に必要な留意事項になります。具体的にどういうコードなのかというと cmake のものが参考になります：

```

=== Buildsystem/cmake.pm ===
sub check_auto_buildable {
    my $this=shift;
    my ($step)=@_;
    if (-e $this->get_sourcepath('CMakeLists.txt')) {
        my $ret = ($step eq 'configure' && 1) ||
            $this->SUPER::check_auto_buildable(@_);
        # Existence of CMakeCache.txt indicates cmake has already
        # been used by a prior build step, so should be used
        # instead of the parent makefile class.
        $ret++ if ($ret && -e $this->get_buildpath('CMakeCache.txt'));
        return $ret;
    }
    return 0;
}

```

Makefile ジェネレータとして configure ステージだけ担当のような顔をしつつ、ビルドキャッシュがある場合は自分が後段も担当すべきとして親である makefile.pm を押しのけて勝つ、というわけです。

9.2.4 ビルドシステムは誰が呼んでいるのか

ところでこのビルドシステム、誰が呼んでいるのでしょうか？例えば dh では

```
$ dh --buildsystem=perl_makemaker
```

のように指定できるのですが、dh にはどこにもビルドシステムに関する処理は書かれていません。

これは dh からオプションをそのままスルーされる形で^{*4}dh_auto_(build|clean|configure|install|test) の dh_auto_*系コマンド(だけ)が呼び出し元になっています。シーケンス中のすべての dh_*コマンドに同様にスルーパスは届くのですが、反応するのがこの5つだけ、という訳です (man debhelper の BUILD SYSTEM OPTIONS)。だからこそ独自処理を書く場合は rules に

```

override_dh_auto_build:
    ...

```

などと override_dh_auto_*ターゲットを書くという話になるわけです。dh_auto_*さえ止めれば、いかなるシーケンスが走ってもビルドシステム呼び出しが行われず、実際のビルドは行われなからです。

これらの dh_auto_*コマンドは上のロード 照会 (check_auto_buildable) API (configure|build|test|install|clean) コールのトリガを引いているだけです。このフローの詳細はライブラリ化されているので各コマンドは3行くらいしかありません。

9.2.5 ビルドシステムの追加方法

ビルドシステムの拡張は簡単で、以下の API を実装した*.pm を Buildsystem/ フォルダに置くだけです。基底クラスに空実装があるので全部書く必要はなく、実際に処理を追加したい API のみ実装すれば十分です。

check_auto_buildable(\$step)	# 必須
pre_building_step(\$step)	# オプション
configure() build() test() install(\$destdir) clean()	# いずれかを実装
post_building_step(\$step)	# オプション

各 API の処理は名称から想像される通りで、先に解説済みの c.a.b API 以外は返値もありません (使われていません)。

サンプルとして、今は懐かしき imake/xmkmf を使ったソースパッケージの自動検知 + ビルドに対応するように imake.pm モジュールを用意してみました：

^{*4} Debian は、ビルドコマンド調査の時も思いましたがコマンドライン引数の引き回し本当に多用しますね …

```

package Debian::Debhelper::Buildsystem::imake;
use strict;
use base 'Debian::Debhelper::Buildsystem::makefile';

sub DESCRIPTION { "imake (IMakefile)" }
sub new { shift->SUPER::new(@_); }
sub check_auto_buildable {
    my($self, $step) = @_;
    return 1 if ($step eq "configure" &&
        glob($self->get_sourcepath("I[Mm]akefile")));
    return 0;
}
sub configure { shift->doit_in_sourcedir("xmkmf", "-a", @_); }
1;

```

こんな簡単なものでも、kterm などの対象パッケージをビルドするには十分です。

なお、これを組み込むには Dh_Buildsystems.pm のソース中の自動判定リストの末尾に

```
our @BUILDSYSTEMS = ('autoconf', ..., 'imake');
```

のようにモジュール名を追加してやる必要があります。

9.2.6 まとめ

本解説では dh フレームワークを支えるビルドシステム部分を解説しました。これは dh_* コマンドとしては dh_auto_* の 5 コマンドに対応し、これらを通してビルドシステムが駆動されています。

9.3 今月のコマンド : dh_builddeb

dh_builddeb は、dh によって起動される一連のコマンドシーケンスの最後を飾るコマンドです (ちなみに最初は dh_testdir)

マニュアルは「dpkg-deb を呼ぶだけのかんたんなおしごと^{*5}」と一行だけの解説ですが、これが意外にも中で色々としていて dh_* コマンドの勉強になります。

9.3.1 何をしているの？

やっていること自体は以下の 3 つです :

1. debhelper(7) のファイル排除指定があれば、その除去処理をする
2. deb/udeb 形式の判定を行い、dpkg-deb の起動分けをする
3. さらに、DEB_BUILD_OPTIONS の parallel= 指定があれば、dpkg-deb を並列駆動する

マニュアルの解説が 1 行の割には、意外に仕事をしています。

9.3.2 疑問 : udeb って何？

実は udeb の存在を知りませんでした。udeb というのは Debian-Installer(d-i) で使用される *.deb 風のパッケージです。形式としては udeb も deb も同じで普通に dpkg で操作できるのですが、極小リソースでの限定的な利用を想定しているためドキュメントはおろかチェックサム機能などまで外されています。

```

=== debian/control ===
Section: debian-installer
...
XC-Package-Type: udeb
XB-Installer-Menu-Item: 1200 <- d-i menu での表示制御パラメータ

```

のように特殊なヘッダが入っている control がある場合、dh_builddeb は自動的に udeb ビルドモードで dpkg-deb を起動します。

他にもこういう特殊ヘッダはあるのだろうかとか、これを入れると具体的に何をどう変えられるのかなど udeb と

^{*5} dh_builddeb simply calls dpkg-deb(1) to build a Debian package or packages.

d-i の話は更に掘ると面白そうですが、今回は脱線ということでここまでしておきます*6。udeb 固有処理は他の dh_* コマンドにも多数含まれており、is_udeb() で様々な処理分けを裏側でしています。

9.3.3 debhelper(7) 系コマンドの実装パターン

dh_builddeb の中を覗くと、各所で \$dh... という変数へのアクセスが頻出しています。これは dh_* コマンドの debhelper(7) オプションのパーズや共通的な処理が Debian::Debhelper::Dh_Lib ライブラリで行われており、このライブラリとコマンド側の連携に %dh というグローバル変数が使われているためです。また %ENV も多用されています。

コードの流れとしては、debhelper の純正 (Perl 製) dh_* コマンドは概ね以下の実装パターンになっています：

```
use Debian::Debhelper::Dh_Lib; # init() 関数などがインポートされる
init(options => { 'myopt=s' => \&dh{MYOPT}, ... }); # @ARGV や %ENV の定型処理

# 含まれているパッケージの数だけ処理を反復
foreach my $package (@{ $dh{DOPACKAGES} }) {
    # 上の init() で取り込まれた結果を見ながら処理をする
    if ( $dh{...} ) { ... # Dh_Lib.pm の API を呼んだりするなど ... }
    # 自動的に取り込まれない環境変数 ( コマンド固有 ) は自分で処理する
    if ( $ENV{...} ) { ... # 上記同様 ... }
}
```

dh_builddeb の場合は、上のループの中が不要ファイルの削除と dpkg-deb の起動になり、これが例えば dh_strip の場合は各生成中パッケージのワーキングフォルダをスキャンして、しかるべきファイルを strip して回るといようになります。

9.3.4 まとめ

dh_builddeb コマンドの解説と、そこから出てきた疑問と共通的な構成の解説を行ってみました。dh_* コマンドは実質 1 行しかないものから 1000 行に迫るものまで色々ありますが、dh_builddeb はちょうど理解する上で手頃な大きさです。

*6 資料としては <http://d-i.alioth.debian.org/doc/talks/debconf6/paper/> かな？

10 索引

2012 年計画, 16

Amazon AWS EC2, 10

debhelper, 17

Debian 勉強会予約システム, 8

dh, 17

dh_builddeb, 20

dput-tweet, 14

twitter, 12

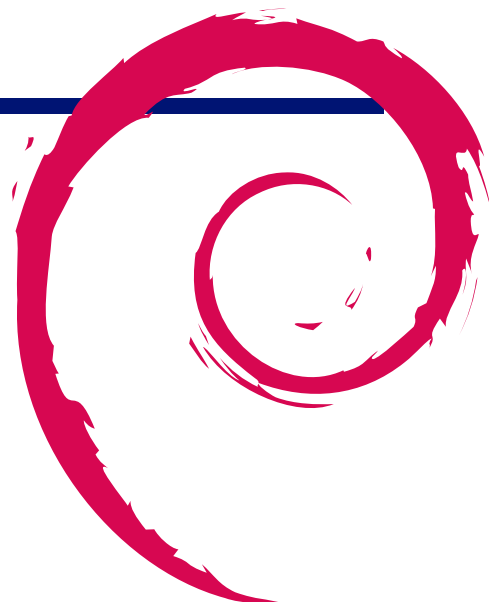
twitter api, 12

udeb, 20

vps, 10

月刊 Debhelper, 17

さくらインターネット, 10





Debian 勉強会資料

2012年1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）
