第 44 回 関西 Debian 勉強会資料

関西 Debian 勉強会担当者 佐々木・倉敷・のがた

2011 年 02 月 27 日

1 Introduction

Debian JP

関西 Debian 勉強会は Debian GNU/Linux のさまざまなトピック (新しいパッケージ, Debian 特有の機能の仕組, Debian 界隈で起こった出来事, などなど) について話し合う会です.

目的として次の三つを考えています.

- ML や掲示板ではなく, 直接顔を合わせる事での情報交換の促進
- 定期的に集まれる場所
- 資料の作成

それでは, 楽しい一時をお楽しみ下さい.

関西 Debian 勉強会

目次

2 最近の Debian 関係のイベント報告

Debian JP

2.1 Squeeze がリリースされました

カウントダウンバナーも用意され[*1] 、本当にカウントダウンどおりにリリースされるのか?と妙な心配もされた Debian 6.0 Squeeze ですが、 2 月 6 日に無事リリースされました。

なぜ 2 月 6 日に設定されたのかというと、ちょうどその時、 FOSDEM(Free and Open Source Development European Meeting) が開催されていて、それに合わせたみたいです。

リリースの様子は identi.ca(自由なソフトウェアで twitter のようなマイクロブロギングサービス) の Debian アカウント[*2]でも中継され、 identi.ca/twitter で、かなり盛り上がっていました。

2.2 第 43 回関西 Debian 勉強会

1 月 23 日、大阪港区民センターにて第 43 回関西 Debian 勉強会が開かれ、杉本さんによる Debian GNU/kFreeBSD についての発表「 Debian GNU/kFreeBSD で便利に暮らすための Tips」と、のがたのバグ報告についての発表「 バグ報告はバグのためだけじゃないよ」がありました。

杉本さんの kFreeBSD の発表は、常用できるとはいえ、細かなところではまだまだ問題が残っているので、もっと骨があるやつをいじりたいぜ!という人には、もってこいのプラットフォームですね。

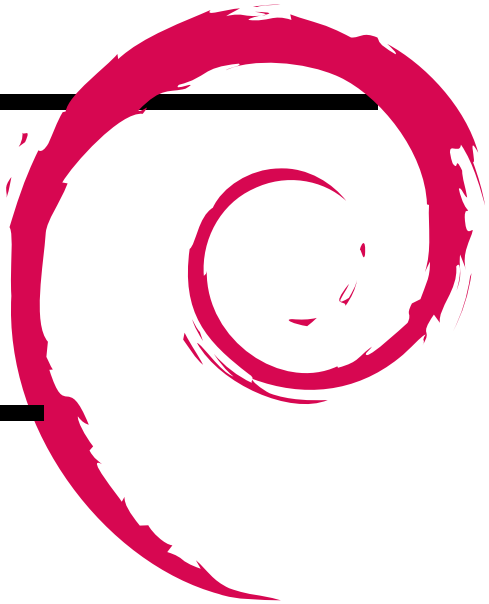
のがたのバグレポートの話は、ブラッシュアップして、バグ報告の入門ぐらいにはしたいですね。

2.3 第 73 回東京エリア Debian 勉強会

2 月 19 日に行われた第 73 回東京エリア Debian 勉強会では、 Hadoop と Debian Games Team 体験記、リリースされてばかりの Squeeze について、みんなで話し合われたそうです。

Debian Games Team で活動するきっかけとなった Xmris は、 Mr. Do クローンなんですね。懐かしくてちょっとやってみたいなと思いました。 CAcert についても、まだ盛り上がっているようです。

[*1] http://news.debian.net/2011/01/22/join-us-in-the-countdown-to-squeeze/
[*2] http://identi.ca/debian



3 事前課題

Debian JP

今回は以下の課題を設定しました.

> ついに Squeeze がリリースされました。そんなわけで Squeeze になって
> - よかったこと
> - うれしかったこと
> - (些細な事だけど) こんな所が変わった!
> - いきなりナニカ踏んだ!!
>
> なんて事柄を、なにか一つご報告下さい。

参加者の皆さんによる回答は以下の通りです.

3.1 甲斐正三

シリアル通信 (ttyUSB0) が一般ユーザーでは使えなくなってあせっています。

3.2 山下康成

rc が変わってはまりました。 LSB ヘッダ必須とか、 updaterc.d 必須とか、、

3.3 八津尾

リリースされて本当によかった

3.4 山下尊也

- よかったこと
  全体的にバージョンがあがったことで、比較的新しいソフトが動かせるようになりました。すいません。普段から sid なので、まったく思いつかなかったです...

3.5 のがたじゅん

- よかったこと
  デスクトップテーマの Space Fun がイカす! 起動が早くなった。 backports が正式に Debian になったので新しい

パッケージが欲しい人に backports を使ったら?と言いやすくなった。

- 残念だったこと
  Squeeze が残念、ではなく自分の取り組みで残念だったことですが Squeeze の Debian Live に時間が取れなかった事が心残りです。 Wheezey がんばります。

3.6 古川竜雄 (frkwtto@gmail.com)

すいません。まだインストールしてません。ごめんなさい。

3.7 川江

とにかく、リリース「 おめでとう」です。 25 日 1 時 55 分の時点でまだ、 Air に Squeeze をインストールできてませんが、日曜までになんとかインストールするつもりです。

3.8 木下達也

- よかったこと：リリースされたこと

3.9 山田 洋平

dwm のバージョンが 4.7 から 5.8 に上がり、仕様も新しくなりました。 Wikipedia にも載っていますがステータスバーに表示する仕方が変わったりなど。でも今見たらパッケージの説明文が古いままです。バグ報告しなきゃですかねこれは。

3.10 松澤二郎

- うれしかったこと： Space Fun がすてき

3.11 occult.zzz (立川勝宣)

- よかった事
  初めから二画面のドライバーが入っており使いやすかった。 leny の時は、ドライバーを探すのに入れすぎで何度もシステムを潰してしまい、結果それが、学ぶ一つになりました。やりはじめたばかりで、疑問多数なので、ぜひ参加させてください。よろしくお願い致します。

3.12 水野源

SheevaPlug でずっと testing な squeeze を使っていたので、正式リリースされて嬉しいです。今回の発表資料を作る際に squeeze の pbuilder で ubuntu 環境を作ろうとして、キーリングが自動で渡らないので構築に失敗する問題踏みましたょ。

3.13 かわだてつたろう

- first point release が三月に予定されている
- dropbox が入らなかった
- nodejs が入らなかった

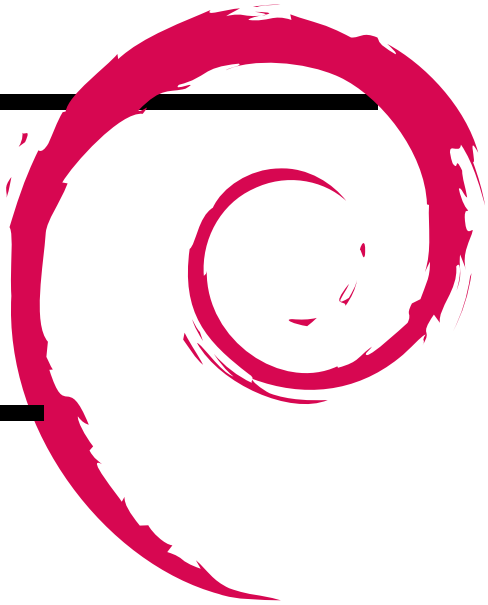
3.14 lurdan

何台か lenny からのアップグレードをしていますが、単機能サーバばかりなせいか、表紙抜けするくらいあっさりと完了しています。 dependency boot への移行がやたら失敗するくらいかな? 別に有り難くもないのでこれは気にしてません。

3.15 中川舟(しゅう)

- こんな所が変わった!
  レガシーなネットワーク接続の方法が先進的な設定方法に変わったと思います。 (GNOME の NetworkManager からの固定 IP アドレスの設定 or/etc/NetworkManager/以下のファイルに設定の要が記述される。) [*3]
- いきなりナニカ踏んだ!!: 無線でつまずきました。

[*3] 編注) NetworkManager で固定 IP の接続設定をするには、 nm-connection-editor を起動 (右クリック→ [接続を編集する]) して、有線の接続設定を追加すれば使えますよ。その際「 自動接続する」と「 すべてのユーザーに利用可能」のチェックをお忘れなく。



4 pbuilder を使ってみよう

水野 源

4.1 pbuilder とは?

4.1.1 pbuilder とはなにか?

pbuilder とは、「 Personal Debian Package Builder の略」で、 Debian パッケージをクリーンな chroot 環境内でビルドするためのツールです。シェルスクリプトで記述されており、内部では debootstrap という Debian の基本システムをインストールするためのユーティリティを利用しています。 pbuilder は debootstrap を用いて「 最小限な Debian 環境」をあらかじめ作成しておき、これをパッケージのビルド時に展開して、その中で実際のパッケージビルドを実行します。

もちろん pbuilder を使わなくてもバイナリパッケージを構築することはできます。ですが pbuilder を使うことで、パッケージビルドに伴ういくつかの問題を解決することができます。

4.1.2 Debian パッケージのビルド おさらい

pbuilder のメリットを理解しやすくするため、ソースパッケージからパッケージをリビルドする手順を例に、 Debian パッケージのビルドをおさらいしておきましょう。

apt-get source コマンドでソースの取得と展開を行います。 Debian のソースパッケージはオリジナルのソース tarball である*.orig.tar.gz、 Debian パッケージにする際に必要な差分である*.debian.tar.gz、署名ファイルである*.dsc の三つのファイルで構成されており、 apt-get source コマンドはこれらのファイルの取得と展開を行なうことができます。

展開後のディレクトリには、オリジナルのソースファイルとパッケージに必要な debian ディレクトリが存在し、ここで dpkg-buildpackage コマンドを実行することでパッケージのビルドが行えます。しかしこの例では、ビルド時の依存関係が満たせずビルドに失敗します。

```
$ apt-get source jd
$ cd jd-2.7.0~beta100627
$ dpkg-buildpackage
   (...snip...)
dpkg-checkbuilddeps: Unmet build dependencies: quilt (>= 0.46-7~) autoconf automake libtool libgnutls-dev
libgtkmm-2.4-dev zlib1g-dev hardening-wrapper libasound2-dev
dpkg-buildpackage: warning: Build dependencies/conflicts unsatisfied; aborting.
   (...snip...)
```

apt-get build-dep コマンドを実行すると、 debian/control ファイルの Build-Depends に列挙されているパッケージがインストールされます。これでビルド時の依存関係が満たされましたので、再度ビルドを行ないます。今度は問題なくビルドが完了するはずです。

```
$ apt-get build-dep jd
   (...snip...)
The following NEW packages will be installed:
  autoconf automake autotools-dev bsdmainutils debhelper defoma diffstat file
  fontconfig fontconfig-config gettext gettext-base groff-base
  hardening-wrapper html2text intltool-debian libasound2 libasound2-dev
   (...snip...)
$ dpkg-buildpackage
```

4.1.3 Debian パッケージのビルド まとめ

- パッケージをビルドするには「ビルド時に」依存しているパッケージをインストールする必要がある
- apt-get は Build-Depends をもとに、ビルド依存パッケージを一括導入できる
- ビルド依存パッケージは debian/control に Build-Depends として記述してある

4.1.4 pbuilder を使うことによるメリット

クリーンな環境でパッケージのビルドを行うことができる

前述の通りパッケージをビルドするためには、ビルド依存パッケージをあらかじめシステムにインストールしておく必要があります。通常のマシンでさまざまなパッケージのビルドを行なっていると、こういったパッケージたちがシステムに蓄積されていきます。 pbuilder はビルド時に最小限な Debian 環境を一時ディレクトリに展開し、必要なパッケージを追加インストールします。そしてビルド終了後に一時ディレクトリごと破棄するため、ビルド環境が「汚れていく」ことがありません。

依存関係のテストができる

上記のように pbuilder は、最小限な Debian 環境に、ビルドに必要なパッケージをその都度インストールします。この「必要なパッケージ」はパッケージの Build-Depends の記述をもとに行いますので、もしも Build-Depends の記述に不足があった場合は、ビルドに失敗する = 記述の間違いに気付くことができます。また Build-Depends が不足しているパッケージがリリースされてしまった場合、必要なパッケージが既に揃っているメンテナの環境では正常にビルドできるが、第三者の環境では apt-get build-dep で必要なパッケージがすべてインストールされないため、リビルドに失敗するという事態が発生する可能性があります。

別リリース、別アーキテクチャ向けのビルドを行うことができる

通常であれば、ビルドマシンを対象となるリリースやアーキテクチャごとにそれぞれ用意しなくてはなりません。しかし pbuilder であれば、 amd64 マシンの上に i386 の chroot 環境を作ることもできるため、 amd64 マシン一台で amd64/i386 両方のパッケージをビルドすることが可能です (もちろん arm なども)。それだけではなく、sid/squeeze/lenny といった別リリースや、 Ubuntu の chroot 環境を作ることも可能です (つまり Debian 上で Ubuntu パッケージが作れます。またその逆も可能!)。

4.2 pbuilder の使い方

4.2.1 pbuilder のインストール

それでは pbuilder をインストールして、実際に動かしてみましょう。 pbuilder 本体をインストールしたら、 pbuilder –create コマンドで chroot 環境を作成します。この例ではディストリビューションに sid、アーキテクチャは i386、ミラーサイトに理研を指定しています。 mirror オプションで指定したミラーサイトは chroot 内の apt-line にも反映されます。

```
$ sudo apt-get install pbuilder
$ sudo pbuilder --create --distribution sid --architecture i386 --mirror "http://ftp.riken.jp/Linux/debian/debian"
```

構築作業が完了すると、/var/cache/pbuilder/base.tgz というファイルが作成されています。これが i386 な sid の chroot 環境を固めた tarball です。

4.2.2 パッケージのビルド

pbuilder でパッケージをビルドするには、 pbuilder –build コマンドにソースパッケージの*.dsc ファイルを引数として渡します。すると pbuilder は、/var/cache/pbuilder/build/(pbuilder-buildpackage の PID) というディレクトリに base.tar.gz を展開し、この中でビルド依存パッケージのインストールと、パッケージのビルドを行ないます。ビルドが終了すると build ディレクトリは削除され、ビルド済みのバイナリパッケージが/var/cache/pbuilder/result 以下に作成されているはずです。

```
$ sudo pbuilder --build jd_2.7.0~beta100627-1.dsc
I: using fakeroot in build.
I: Current time: Tue Feb 15 19:48:46 JST 2011
I: pbuilder-time-stamp: 1297766926
I: Building the build Environment
I: extracting base tarball [/var/cache/pbuilder/base.tgz]
   (...snip...)
I: Installing the build-deps
 -> Attempting to satisfy build-dependencies
   (...snip...)
I: cleaning the build env
I: removing directory /var/cache/pbuilder/build//8063 and its subdirectories
```

4.2.3 複数の pbuilder 環境

pbuilder はデフォルトで/var/cache/pbuilder/base.tgz を使用しますが、–basetgz オプションを使用することで任意のパス/ファイルを指定することができます。これを利用して複数の pbuilder 環境を使いわけることが可能です。

```
$ sudo pbuilder --create --distribution sid --architecture i386 --basetgz /var/cache/pbuilder/sid-i386.tgz
$ sudo pbuilder --create --distribution squeeze --architecture amd64 --basetgz /var/cache/pbuilder/squeeze-amd64.tgz
$ sudo pbuilder --build jd_2.7.0~beta100627-1.dsc --basetgz /var/cache/pbuilder/sid-i386.tgz
$ sudo pbuilder --build jd_2.7.0~beta100627-1.dsc --basetgz /var/cache/pbuilder/squeeze-amd64.tgz
```

4.2.4 pbuilder 環境へのログイン

パッケージビルド時に自動的に使用される pbuilder の chroot 環境ですが、ここへ chroot してシェルを取ることも可能です。そのためには pbuilder –login コマンドを使用します。この機能を利用すれば、 squeeze 上で sid の環境をちょっと試す、といったことも可能です。

```
$ sudo pbuilder --login --basetgz /var/cache/pbuilder/sid-i386.tgz
```

4.2.5 pbuilder 環境の更新

パッケージをビルドするためには、 chroot 環境を常に最新に保っておく必要があります。既存の環境をアップデートするには pbuilder –update コマンドを使用します。

```
$ sudo pbuilder --update --basetgz /var/cache/pbuilder/sid-i386.tgz
```

また前述のように展開した pbuilder 環境は使い捨てで、環境に対して行なった変更はビルド終了後に破棄されます。これによって常にクリーンな環境が保たれるわけですが、場合によっては恒久的な変更を加えたい場合があるかもしれません。そんな場合は、–login –save-after-login コマンドでカスタマイズが可能です。–save-after–login つきで chroot 環境にログインして何らかの変更を加えると、ログアウト時に base.tgz を更新してくれます。 chroot 環境に「パッケージを追加しておきたい」「設定を変更しておきたい」などという時に便利です。

```
$ sudo pbuilder --login --save-after-login --basetgz /var/cache/pbuilder/sid-i386.tgz
I: Building the build Environment
I: extracting base tarball [/var/cache/pbuilder/sid-i386.tgz]
   (...snip...)
# exit
   (...snip...)
I: creating base tarball [/var/cache/pbuilder/sid-i386.tgz]
I: cleaning the build env
I: removing directory /var/cache/pbuilder/build//679 and its subdirectories
```

4.3 pbiulder をさらに便利に使う/cowbuilder

このように pbuilder は非常に便利ですが、毎回 tarball を展開し使い終ったら削除するというのは、展開や削除に時間がかかります。さらに tarball の中に格納されているファイルとビルド時に使用されるファイルは同じものですし、ほとんどのファイルは変更されないまま破棄されるわけですので、わざわざコピーするのは無駄です。それならば原本そのものを見ておけばいいよね、という発想で無駄を省いたのが cowbuilder です。

4.3.1 tarball からハードリンクへ

cowbuilder は pbuilder と同じように使用でき、同じように動作します。ただし、 pbuilder が chroot 環境を tarball で保存するのに対し、 cowbuilder は/var/cache/pbuilder/base.cow というディレクトリ以下に展開した状態で保存される点が異なります。 pbuilder では使用時に tarball を/var/cache/pbuilder/build 以下に展開しますが、 cowbuilder は cp -la コマンドで base.cow へのハードリンクを/var/cache/pbuilder/build 以下に作成しています。ファイル実体のコピーや展開は行なわれないため非常に高速で、余計なディスクスペースも消費しません。

ですが単にハードリンクしているだけでは、ビルド用の一時環境に変更が加わった時に原本が変更されてしまいます。そこで使用されているのが cowbuilder の名前の由来でもある「 Copy-On-Write」という技術です。 Copy-On-Write とは「 コピーしたふりをして原本を参照させておき、変更が加わった際に実際にコピーを行なって変更を書きこむ」技術で、 cowbuilder の内部では、 Copy-On-Write を実現するために cowdancer というプログラムが使用されています。これにより cowbuilder は、大部分のファイルはハードリンクで原本 (base.cow) をそのまま使い、変更が必要な部分のみコピーして書きかえるという効率的な動作が可能になっています。

```
$ sudo cowbuilder --create --distribution sid --architecture i386 --mirror "http://ftp.riken.jp/Linux/debian/debian"
$ cd /var/cache/pbuilder/base.cow/
$ ls -li etc/debian-version
7020666 /etc/debian_version
$ sudo cowbuilder --login
# ls -li /etc/debian-version
7020666 /etc/debian_version
# vi /etc/debian_version
# ls -i /etc/debian_version
7070836 /etc/debian_version
```

4.4 Ubuntu 的な pbuilder/cowbuilder

Ubuntu では pbuilder/cowbuilder を直接使わず、 pbuilder-dist/cowbuilder-dist というラッパースクリプトを経由して使うのが一般的です。これらのスクリプトは ubuntu-dev-tools パッケージに含まれており、 Debian でも使用することができます。これらのスクリプトは、 pbuilder における pbuilder-distribution.sh をよりよい形で Python で再実装したものと言えます。

4.4.1 pbuilder-dist/cowbuilder-dist の特徴

一言で言ってしまえば、 pbuilder/cowbuilder に、事前に設定したオプションを食わせるラッパーです。次のように実行します。

```
$ pbuilder-dist distribution [architecture] operation
```

–debug-echo オプションを使用すると、実際に起動される pbuilder のコマンドとオプションを確認することができます。 pbuilder は root か sudo つきで実行する必要がありましたが、 pbuilder-dist は内部で pbuilder が sudo つきで起動されるため、 pbuilder-dist 自身はユーザ権限で実行できます。また base tarball やログファイル、ビルドしたパッケージなどをユーザのホームディレクトリ以下に作成するため、さまざまな pbuilder のオプションが暗黙的に設定されているのがわかります。また tarball のファイル名も dist-arch-base.tgz という形になっており、複数の pbuilder 環境を共存させやすくなっています。

```
$ pbuilder-dist sid i386 create --debug-echo
sudo /usr/sbin/pbuilder --create --basetgz "/home/mizuno/pbuilder/sid-i386-base.tgz" --distribution "sid"
--buildresult "/home/mizuno/pbuilder/sid-i386_result/" --aptcache "/var/cache/apt/archives/"
--override-config --logfile /home/mizuno/pbuilder/sid-i386_result/last_operation.log
--mirror "ftp://ftp.debian.org/debian" --components "main contrib non-free" --debootstrapopts --arch="i386"
```

4.4.2 pbuilder-dist の呼び出し方

pbuilder-dist コマンドをそのまま呼び出してもよいのですが、シンボリックリンクで別名をつけて使うのも便利です。シンボリックリンクは pbuilder-[dist]-[arch] という形式で作成します。例えば pbuilder-sid-amd64 というリンクは、pbuilder-dist sid amd64 というコマンドと同等の働きをします。 i386/amd64 の sid と squeeze の環境、計 4 つを使いわけたいなどという場合は、それぞれの名前でリンクを作成しておけばわかりやすいでしょう。

```
$ ln -s /usr/sbin/pbuilder-dist pbuilder-sid-amd64
$ ./pbuilder-sid-amd64 create --debug-echo
sudo /usr/sbin/pbuilder --create --basetgz "/home/mizuno/pbuilder/sid-amd64-base.tgz" --distribution "sid"
--buildresult "/home/mizuno/pbuilder/sid-amd64_result/" --aptcache "/var/cache/apt/archives/"
--override-config --logfile /home/mizuno/pbuilder/sid-amd64_result/last_operation.log
--mirror "ftp://ftp.debian.org/debian" --components "main contrib non-free"
```

4.4.3 pbuilder-dist のメリット

pbuilder-dist では、簡単に複数の pbuilder 環境を使い分けることができます。特に Ubuntu は複数のリリースや、上流である Debian でのテストが必要なため、 release、 release+1、 sid を簡単に切り替えられることが大事です。

また、デフォルトでユーザのホーム以下に base tarball やビルドしたパッケージができるため、ユーザ権限での管理がしやすいという特徴があります。

4.5 まとめ

- pbuilder/cowbuilder を使うことで、普段使っている環境を開発用パッケージで汚すことなく、パッケージのビルドを行うことができます。
- 基本的な Debian システムでビルドが可能なことを確認することができるので、パッケージをいじった後は必ず確認しておくとよいでしょう。
- cowbuilder を使えば、より高速に pbuilder 環境を利用することができます。
- pbuilder-dist を使えば、リリース間、ディストリ間を渡り歩くのも簡単です。

5 Squeezeの変更点をみんなで見てみよう

倉敷悟

5.1 Squeeze の情報源

主な情報源としては下記のものがあります。

- Debian – ニュース – Debian 6.0 ”Squeeze” released: `http://www.debian.org/News/2011/20110205a`
- Debian GNU/Linux 6.0 (squeeze) リリースノート (32 ビット PC 用): `http://www.debian.org/releases/stable/i386/release-notes/index.ja.html`
- NewInSqueeze - Debian Wiki: `http://wiki.debian.org/NewInSqueeze`

このうち、リリースノートの読み方を簡単に紹介しつつ、目立った変更をざっくり眺めていこうと思います。

5.2 Debian リリースノートの読み方

読み方、なんて題名にしてますが、まさか読んでない……なんてことは……まさかね。

5.2.1 大まかな構造

さて、リリースノートは、構造はほぼ決まっていて、各リリースでの違いだけが反映されていくようになっています。

その章立ては次の通りです。これは、原文自体がこのような形で分割されているので、基本的には今後のリリースでも踏襲されると思います。

- 前置き
- 最新情報
- インストーラの変更点
- アップグレード
- 問題点
- 参照情報
- oldstable での事前準備
- 目録

まずは、「 最新情報」をざっくり読んだあと、「 問題点」にしっかり目を通すのが大切です。結局、 squeeze で何が変化してどうなったのか、という情報はここに集っているからです。

「 インストーラの変更点」は、新規にインストールする人は一応見ておきましょう。といっても、最近のインストーラ

は、わりと「 見ればわかる」ようになっているので、インストールするだけなら読まなくても問題になることは少ないと思います。ちなみに、今回からは、最新のインストーラが以前のバージョンもインストールできるようになったらしいです。最新のインストーラで woody とか sarge を入れたらどんな (目もあてられない) ことになるのか、興味深いですね。

「 アップグレード」と「 oldstable での事前準備」は、インストーラを使わずに以前のリリースから更新する人はとにかく必読です。これは言わば、「 既に踏まれた地雷」を考慮に入れたアップグレード作業の手順書です。従って、ここに書かれた内容にハマってしまうというのは、なんというか (政治的に正しくない言葉をお好みで) ということになってしまいますので注意してください。

ちなみに、「 なんだ用語集か」と思って見過ごしてしまいがちなのですが、「 目録」のページには、パッケージ名からの逆引きインデックスが含まれていたりしますので、「 あのパッケージに何か (わざわざリリースノートで触れないといけないような) 変化があるかな」といった見方をすることもできたりします。一度くらいは見てみるといいかも知れません。

5.2.2 どのように書かれるか

さて、最初に書いた通り、リリースノートは、これまでの文章に随時継ぎ足したり削ったりして、秘伝のタレのように作られていきます。

ここで、「 前置き」の部分に着目してみましょう。そういった変更は、 DDP (Debian Documentation Project) のメンバーが適宜行う他、主に BTS の release-notes に対するリクエストへの対応としてもなされます。あるパッケージのバグについてやりとりしている中で、「 あ、これはリリースノートに書いとかないと」という流れになることもあります。

既に書いた「 既に踏まれた地雷」はこういう形で集められたものですので、残念ながら網羅性があるわけではありません。皆さんも、何か新しい地雷を踏んじゃったなー、という時は、遠慮なく BTS へ報告してください (英語で)。

ここで注意が必要なのは、こういった BTS の動きもあって、「 リリースノートはリリースされた後にも更新される場合がある」ということです。今自分が読んでいるのが最新かどうか、ということにはちょっと注意しておくといいでしょう。http://svn.debian.org/viewsvn/ddp/manuals/trunk/release-otes/en/ を見て、各ファイルの最終更新日をチェックするのがお手軽です。

翻訳が追いついていないこともあるので、その場合は頑張って英語で読んだ後、 debian-doc あたりに原文更新を通知しましょう。翻訳済みのパッチも歓迎されます。

5.2.3 ともあれ Squeeze になった後

リリースノートの「 前置き」でも書かれていますが、基本的に、各パッケージ毎の些細な変更はわざわざリリースノートに書かれたりしません。

普段使いのパッケージについては、とりあえず基本に立ち帰って、「 /usr/share/doc/パッケージ名」に README.Debian や NEWS.Debian が追加されたり更新されたりしていないか、確認しておくようにしましょう。

おっかしーなー、想定と違う動き方をするなー、と思って散々ぐぐりまくったあげくに NEWS.Debian にちゃんと書いてあった、というのはよくある話です。

5.3 システムのコア部分の変更

Squeeze になって Linux カーネルとシステム起動部分に大きな変更がありました。

Linux カーネルでは、カーネルにバイナリの形で含まれていた含まれていたファームウェアが分離され、 non-free セクションに移動されました。これによって、 Debian の main セクションにある linux カーネルイメージは完全にフリーになりました. 移動された non-free のファームウェアには AMD Radeon と Broadcom のファームウェアなどが入っており、 NIC が使えない、インストール後に X が起動しないなどの影響が考えられます。

インストール中に non-free のファームウェアを読み込ませるには、 `http://cdimage.debian.org/cdimage/unofficial/non-free/firmware/` からファームウェアをダウンロードし、あらかじめ USB メモリなどに展開しておき、インストーラの指示に従って読み込ませます。 Debian インストール後に non-free のファームウェアをインストール

するには、 apt-line の non-free セクションを有効にして、 firmware-linux パッケージをインストールします。 (依存関係で firmware-linux-free と firmware-linux-non-free の二つがインストールされます。)

システムの起動では、起動シーケンスが依存関係ベース (insserv) になり、デーモンが並列起動されるようになりました。

デーモンの並列起動で問題が出た場合は、/etc/default/rcS に「 CONCURRENCY=none」と設定すると無効にできます。 (insserv は sysv-rc パッケージが依存しているので残念ながら外すことはできません。)

5.4 パッケージまわり

”libs”、”network”などのパッケージセクションが分割された一方で、”embedded”、”haskell”、”video” などの新しいセクションが追加されました。実際のところ、 debian を単に使うだけなら、セクションは余り意識しなくてもさほど困らないので、開発者向けの変更点と言えるかもしれません。

今まで backports.org で提供されていたバックポートされたパッケージが、 backports.debian.org から提供されるようになりました。 debian.org の公式サービスへと格上げされたため、これまでよりは敷居が下がって使いやすくなっっと思います。

volatile.debian.org で提供されていたパッケージが、 stable-updates という形で提供されるように変更されました。アンチウィルス系など、リリース後も継続してデータの更新が必要な種類のパッケージが volatile で提供されていましたが、パッケージ数も少なく、ほぼ機能していない状態でした。これが stable-updates となって状況が改善されればいいですね。

いろいろと細かい変更のある APT まわりですが、まずは推奨 (Recommends) パッケージを全部自動的にインストールされるようなっている、という点おおさえておきましょう。 RedHat を思わせる勢いであれもこれも入ってくるので、ストイックにいきたい人は、-R オプションを活用してください。

また、 apt-get と aptitude の位置付けがまた変更されています。コマンドラインでポンと投げる場合は apt-get を使います。また、ディストリビューションのアップグレードにも、 aptitude ではなく apt-get を使うことが推奨されています。 aptitude は引数なしで起動して、コンソール GUI で使うもの、ということになるようです。

5.5 ハードウェアまわり

Squeeze のハードウェア周辺の対応を見ていきましょう。

対応アーキテクチャですが、一番目を引くのはテクノロジープレビューという形ですが、 Linux カーネルではなく FreeBSD カーネルを使った、 kfreebsd-amd64 と kfreebsd-i386 の追加でしょう。 kFreeBSD については前回、杉本さんの資料もご覧ください。

そのほかには、 Lenny までサポートされていた HP PA-RISC (’hppa’)、 Alpha (’alpha’)、 ARM (’arm’) アーキテクチャが廃止になり、対応アーキテクチャは、 32 ビット PC (’i386’)、 64 ビット PC (’amd64’)、 PowerPC (’powerpc’)、 SPARC (’sparc’)、 Intel Itanium (’ia64’)、 ARM EABI (’armel’)、 MIPS (’mips’ (ビッグエンディアン)、’mipsel’ (リトルエンディアン))、 S/390 (’s390’) の 9 つ +2 の 11 種類のサポートになりました。[*4]

X 関係のサポートを見ると、 Intel, AMD, NVIDIA のメジャーなグラフィックチップにおいてカーネル内でグラフィック設定を行う、カーネルモードセッティング (KMS) が有効化されています。

これらにより X の起動が早くなったり、サスペンド・レジュームで X が死ななくなるなど良い面もある反面、コンソール画面の文字が出なくなったりするなどの影響もあります。その場合は/etc/modprobe.d/(i915—radeon—nouveau)-kms.conf の設定を変更する必要があります。[*5]

細かいところでは X とコンソールのキーボード設定が統一化されました。

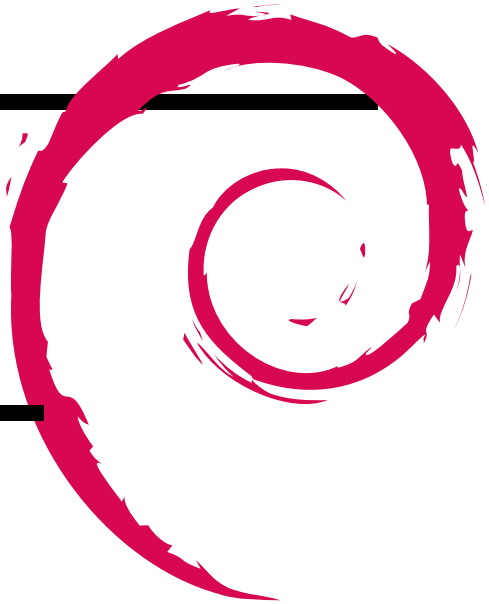
/etc/default/keyboard にキーボードの設定をしておくと、コンソールと X の設定は上書きされて同じ設定になり

[*4] sh は wheezy?

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

ます。

lenny から基本的に xorg.conf は使わなくなったので影響は少ないと思いますが、 xorg.conf を書いている人は注意が必要です。



6 今後の予定

Debian JP

6.1 次回の関西 Debian 勉強会

次回の関西 Debian 勉強会は 3 月 27 日 (日)、大阪港区区民センターで開催する予定です。発表については未定ですので、みなさまの発表をお待ちしております。

6.2 4 月神戸 IT フェスティバル + オープンソースカンファレンス 2011Kansai@Kobe

4 月 15 日 (金)、 16 日 (土) に JR 神戸駅すぐの神戸市産業振興センターにて、 神戸 IT フェスティバル + オープンソースカンファレンス 2011Kansai@Kobe が開催されます。

セッションは「 第 46 回関西 Debian 勉強会 - 遂にリリースされた (?) Squeeze について - 」で、講演者調整中 (多分佐々木?)、内容は「 リリースされた Debian 安定版「 Squeeze」の紹介と Debian 界隈の話題などについてお話しします。」です。ブース要員が足りないので協力して下さる方大歓迎です。

7 メモ

関西 Debian 勉強会

Debian 勉強会資料

2011 年 02 月 27 日　　初版第 1 刷発行
関西 Debian 勉強会 (編集・印刷・発行)