東京エリア でびあん 勉強会

Debian勉強会幹事　上川純一

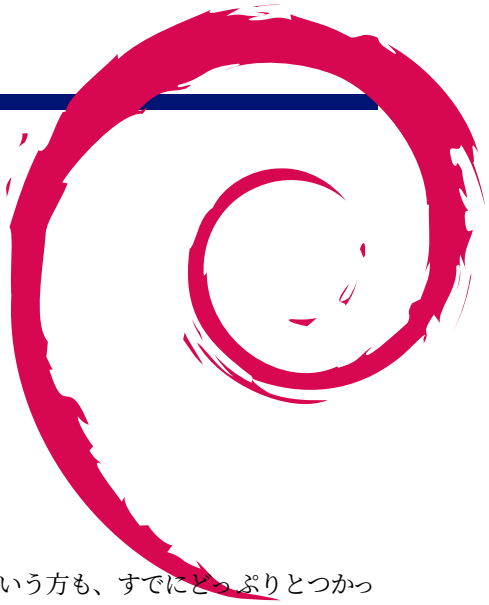
2010 年 4 月 17 日

特集 1: piuparts **の使い方**

特集 2: debtags **入門**

デビアン勉強会

目次



1 Introduction

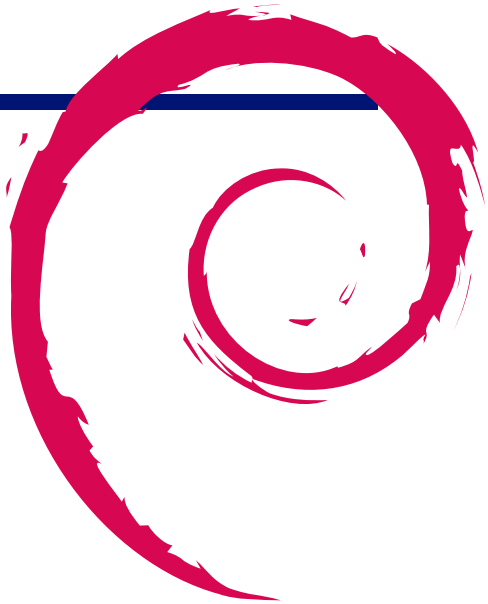
上川 純一

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。



2 事前課題

岩松 信洋

今回の事前課題は以下です:

1. あなたの使っている init は何ですか? 使っている理由を教えてください。
2. あなたのパッケージ検索方法を教えてください。

この課題に対して提出いただいた内容は以下です。

2.1 原口秀康

1. デフォルトの 2 を使用しています。通常のインストールをしてそのまま使用しているからです。 2-5 全て同じなのに他のランレベルを使用する意味はわかっていません
2. apt-get apptitude、「パッケージの内容を検索」を使用します。明確に名前が分かるものは apt-get や apptitude でこういうものがほしいと思うときは「パッケージの内容を検索」を使います。キーワードが使われていないパッケージでも検索することが可能だからです

2.2 キタハラ

1. デフォルトで組み込まれる「init」、特に不都合がないため。
2. 最近は、「Google 先生」に尋ねる事のほうが多い。

2.3 yama1066

1. sysvinit まだ移行してないから。 upstart に移行するメリットが議論できればいいなと思います。
2. dselect でひたすらゴリゴリ…、嘘です。 apt-cache search でほげほげ。

2.4 henrich

1. sysvinit を使ってます。昔は initng を使ったこともありましたが…面倒を避けるため、ですかね。

2.5 koedoyoshida

1. 安定志向なので lenny 標準の init です。仕事で使ってるマシンは upstart や init が混在しています。
   私は基本的に Linux はサーバ使用です。滅多に再起動しないのであまり恩恵を受けていません。
   特にメーカー系のサーバ機は (再) 起動時に SAS,FC 等を含めた BIOS チェックだけで数分かかるのもざらなので、

起動高速化のメリットは受けにくいというのが正直なところです。
init 以外を使用することによる、起動時以外のメリットが有れば知りたいです。

2.6 鈴木崇文

1. sysvinit。特にスピードを求めようと思ったことがないことと、サービス起動が非同期だと何か問題が起きたときに調査しにくいイメージがあるため。とはいえ、 Ubuntu のマシンでは Upstart 使っているので、ただ流されているだけだと思います。

2.7 藤沢理聡 (risou)

1. 今使用しているのは sysvinit です。理由は、 lenny のデフォルトになっているからです……。＃このあたり、違いがよくわかってないので、わざわざ変更してないです。

2.8 村田信人

1. sysvinit。少し前に Squeeze をインストールしたらデフォルトだったから。
2. apt-cache search でざっくりと見当をつけてから apt-cache show で詳細を確認。

2.9 akedon

1. sysvinit です。現在、デフォルトなのとメカニズムが単純なのでこれで良いかなと思っています。
2. aptitude search 〜 と apt-file search 〜 を使っています。

2.10 Hirotaka Kawata

1. init。 Debian 標準の init (ふつうの init)
2. aptitude search ”keyword”

2.11 opentaka

1. デフォルトの init。デフォルトで入っていたので。
2. aptitude search ”package name”

2.12 松澤二郎

1. あまり意識していませんでしたが、デフォルトのものを使っています。 sysvinit?
2. aptitude search hoge

2.13 まえだこうへい

1. sysvinit。テスト環境とかでは upstart も試してますが、 MacBook の Sid 環境とかでは切り替えるのまだ怖いですね。
2. apt-cache

2.14 Yasunori Higashiyama

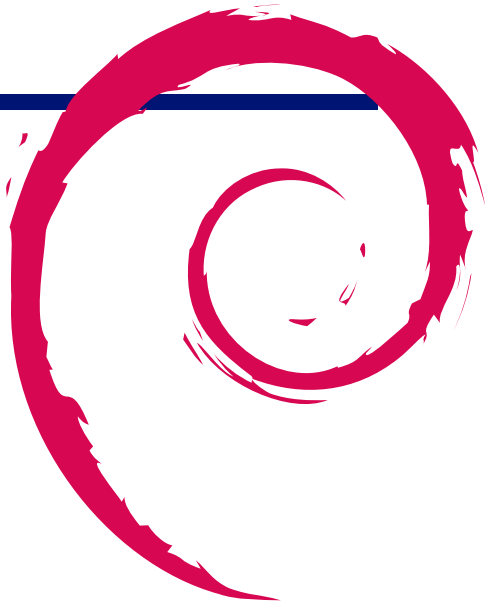
1. sysvinit。特に困っていないのでそのまま。
2. web で検索か aptitude search

2.15 岩松 信洋

1. sysvinit です。 Debian のデフォルトだからです。
2. aptitude で検索しています 。タグを使います。パッケージのインストールは apt ですが....。あとは、 iceweasel や Google Chrome の検索ボックスで検索することもあります。

2.16 google-account@rolf.leggewie.biz

今回もよろしくお願いします。



3 最近の Debian 関連のミーティング報告

前田耕平

3.1 東京エリア Debian 勉強会 62 回目報告

2010 年 3 月の東京エリア Debian 勉強会、参加者は山本さん、日比野さん、岩松信洋さん、吉野与志仁さん、本庄さん、 henrich さん、あけどさん、前田耕平さん、荒木 (靖) さん、吉田@小江戸さん、中尾さん、上川の 12 人でした。

本庄さんがニューラルネットワークを使って書籍スキャンを便利にしてみた話。次回は libsvm と libsane を使っていろいろ拡張するそうです。

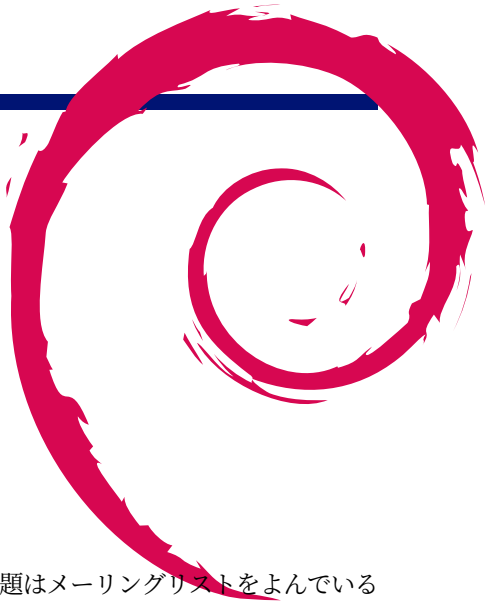
前田さんが weka を使ってみて、今月の生活費を予測した内容でした。回帰分析の結果とか、おもしろすぎです。

上川が libfftw と libsndfile を使ってみた話をしました。

日比野さんが、日本語 man がうまく表示されない件について発表してくれました。 groff の深みについて語り合う会です。 dvi サポートとかを考えるとめんどくさすぎですが、どうしたものか。

吉野さんが、 dpkg の新しいソースファイル形式、 3.0 quilt について紹介してくれました。

宴会は銀座ライオン。



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` に投稿された内容と Debian Project News からです。

問題 1. DPL 2010 に立候補しているのは誰?

☐ A Yasuhiro Araki

☐ B Charles Plessy

☐ C Kurt Roeckx

問題 2. Debian policy 3.8.4.0 で追加された項目は?

☐ A /sys と /selinux の FHS に対する例外ポリシー

☐ B kFreeBSD と Linux を共存するポリシー

☐ C スーパ牛さんパワーに関するポリシー

問題 3. 最近ハードウェアトラブルがあったサーバは?

☐ A rie.debian.org

☐ B ries.debian.org

☐ C rise.debian.org

問題 4. buildd.debian.org のあるサーバが移動しました。どこに移動したでしょう。

☐ A peri.debian.org

☐ B cimarosa.debian.org

☐ C grieg.debian.org

問題 5. squeeze のインストーラで追加された機能は

☐ A Recommends をインストールするようにします

☐ B インストーラ上でパッケージがビルドできます

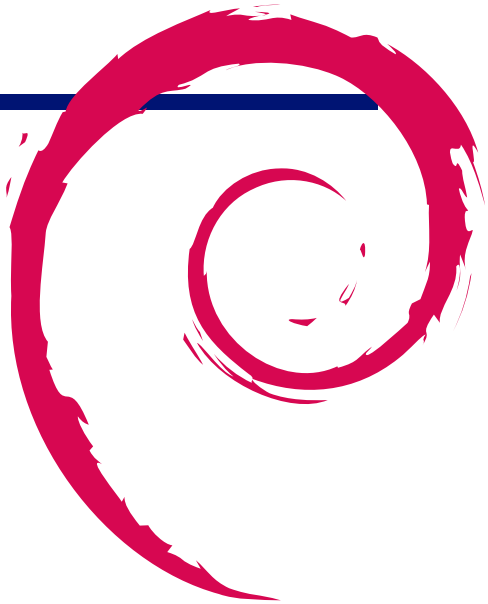
☐ C クロスアーキテクチャインストール機能を追加しました

問題 6. 新しく mips 用 porterbox が追加されました。CPU コア数はいくつでしょうか。

☐ A 64

☐ B 32

☐ C 16



5 piuparts の使い方

岩松

5.1 はじめに

piuparts は 次期リリース squeeze の目標の一つに挙げられている Package clean install/uninstall を達成するためのサポートツールです。既に構築された Debian パッケージのインストール、アンインストール、アップグレードのチェックを行います。通常、パッケージ構築時の依存関係チェックや実際の構築には pbuilder/cowbuilder を使います。パッケージのインストール、動作確認までは行いますが、アンインストールまでの確認を行っているパッケージメンテナは少ないようで(実際にそのまま使う人が多いためと考えられる) 、アンインストールできない事が稀にありました。最悪の場合、パッケージを作ってテストせずにアップーロードしてしまう事もあるようです。また、 stable からのアップグレードチェックもパッケージメンテナはあまりやってないのではないでしょうか。このような問題をチェックするためのツールとして piuparts は作られました。では、 piuparts はどのように動き、どのように使うのか見ていきましょう。

5.2 piuparts の使い方

piuparts を使ってパッケージのチェックを行う場合には、 piuparts コマンドにチェックしたいパッケージを指定します。例えば、 libcv4_2.0.0-4_i386.deb パッケージをチェックしてその結果を/tmp/libcv4_2.0.0-4_i386.piuparts-log に保存する場合には以下のように実行します。実行するとログが標準出力にも出力されますが、-l オプションで指定したログ指定先にも保存されます。出力されるログからパッケージのインストール、アンインストール、アップグレードを確認することができます。

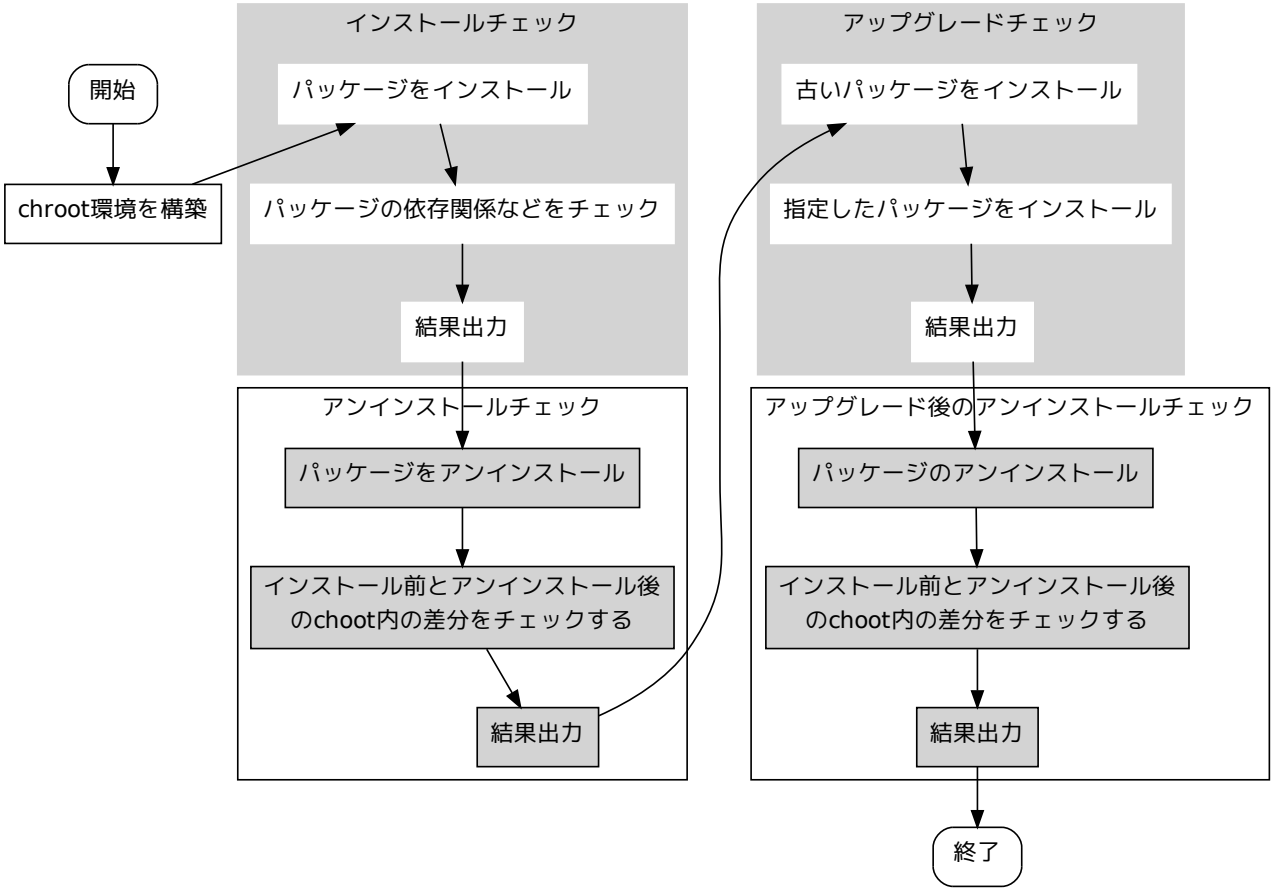
```
$ sudo piuparts libcv4_2.0.0-4_i386.deb -l /tmp/libcv4_2.0.0-4_i386.piuparts-log
0m0.0s INFO: --------------------------------------------------------------------------------
0m0.0s INFO: To quickly glance what went wrong, scroll down to the bottom of this logfile.
0m0.0s INFO: FAQ available at http://wiki.debian.org/piuparts/FAQ
0m0.0s INFO: --------------------------------------------------------------------------------
0m0.0s INFO: piuparts version 0.38 starting up.
0m0.0s INFO: Command line arguments: /usr/sbin/piuparts libcv4_2.0.0-4_i386.deb -l /tmp/libcv4_2.0.0-4_i386.piuparts-log
0m0.0s INFO: Running on: Linux chimagu 2.6.31-1-686 #1
 SMP Sun Nov 15 20:39:33 UTC 2009 i686
0m0.0s DEBUG: Starting command: ['dpkg', '--info', 'libcv4_2.0.0-4_i386.deb']
0m0.2s DUMP:
.... 省略 .....
6m32.6s DEBUG: Starting command: ['chroot',
 '/tmp/tmplunhrZ', 'umount', '/proc']
6m32.6s DEBUG: Command ok: ['chroot',
 '/tmp/tmplunhrZ', 'umount', '/proc']
6m33.0s DEBUG: Removed directory tree at /tmp/tmplunhrZ
6m33.0s INFO: PASS: All tests.
6m33.0s INFO: piuparts run ends.
```

piuparts の使い方は大きくわけて 2 つあります。一つはローカル PC にあるパッケージをチェックする場合、もう一つは既に Debian にインストールされているパッケージをチェックする場合に使います。前者はパッケージメンテナがよく使う方法です。パッケージをアップロードする前にパッケージを指定してテストします。後者の場合はあまりメンテナはあまり使う機会はないと思いますが、後で説明する piuparts.debian.org で使われています。

5.3 piuparts の動作

では、 piuparts の動作を見てみましょう。状態遷移図を図 1 に示します。非常にシンプルなチェック方法になっていることがわかります。

図 1 piuparts の動作



大まかな動きは以上になりますが、内部では dpkg、 apt の動きを利用したものになっています。例えば、パッケージのインストールチェックは以下のような動きになっています。

1. dpkg -i で 指定されたパッケージをインストールする。
   依存するパッケージがあるばあい、これは失敗する。
2. apt-get -yf --no-remove でパッケージの依存関係を apt で回避してインストールする。

これは、パッケージ依存関係をパッケージ情報から抽出して指定する方法を取らず、 apt のチェック機構を用いて依存関係を回避しようとしています。

5.4 ログの見方

piuparts はログが多いので正しい動きをしているのか非常にわかりずらいです。どのように見たらよいのか簡単に説明します。

piuparts のログは、基本的に以下の順で出力されます。

1. コマンド実行 (**DEBUG: Starting command:**)
2. 実行開始タグ (**DUMP:**)
3. 実行時のログ
4. コマンド結果 (**ERROR:** or **DEBUG: Command ok**)

そして、最後にテスト結果が出力されます。次にテストが正常に終了した場合と、問題がある場合を見てみます。

5.4.1 テストが正常に終了した場合

テストに問題がない場合には、以下のように出力されます。インストール → アンインストール、インストール → アップグレード → アンインストール のチェックが正常に終了していることがわかります。

```
..... 省略.....
6m13.7s INFO: PASS: Installation and purging test.
..... 省略.....
6m32.6s INFO: PASS: Installation, upgrade and purging tests.
..... 省略.....
6m33.0s INFO: PASS: All tests.
6m33.0s INFO: piuparts run ends.
```

5.4.2 エラーがある場合

エラーがある場合には、 **ERROR:**の次にエラー内容がされます。以下に、実際のエラー内容を示します。これは、upstart のチェック結果ですが、 essential パッケージである、 sysvinit をアンインストールしようとして、エラーになっています。

```
0m6.0s DEBUG: Starting command: ['chroot', '/org/piuparts.debian.org/tmp/tmpZ-SX9D', 'apt-get', '-y', 'install', 'upstart']
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^: コマンド実行
0m6.3s DUMP:
^^^^^^^^^^^^: コマンド実行時の情報
  Reading package lists...
  Building dependency tree...
  The following extra packages will be installed:
    libdbus-1-3
  Recommended packages:
    dbus
  The following packages will be REMOVED:
    sysvinit
  The following NEW packages will be installed:
    libdbus-1-3 upstart
  WARNING: The following essential packages will be removed.
  This should NOT be done unless you know exactly what you are doing!
    sysvinit
  0 upgraded, 2 newly installed, 1 to remove and 0 not upgraded.
  Need to get 636kB of archives.
  After this operation, 1196kB of additional disk space will be used.
  E: There are problems and -y was used without --force-yes
0m6.3s ERROR: Command failed (status=100): ['chroot', '/org/piuparts.debian.org/tmp/tmpZ-SX9D', 'apt-get', '-y', 'install', 'upstart']
^^^^^^^^^^^^^: コマンド結果
```

5.5 piuparts のオプション

普通の使い方では使いづらいので、 piuparts で提供されているオプションを自分が使っている開発環境に合わせて使うのが普通のようです。以下ではよく使うオプションを紹介します。

5.5.1 pbuilder の base.tgz を piuparts で利用する

piuparts はテストする度に base イメージを構成するパッケージ群をミラーサーバから取得し、 base イメージを構築します。キャッシュする機構はいまのところ存在せず、毎回取得する仕様になっています。これ

ではサーバに負荷がかかります。そこで、-p オプションを使います。 このオプションは pbuilder の base イメージ/var/cache/pbuilder/base.tgz を使ってテストを行うオプションです。通常、パッケージメンテナは pbuilder/cowbuilder を使ってパッケージビルドチェックを行うので、便利なオプションの一つです。また、 pbuilder を使ってないが、 base.tgz を独自のスクリプトで保持している人もいるでしょう。この場合には--basetgz オプションで tgz ファイルを指定することによって、利用できます。

5.5.2 ディストリビューションの指定

piuparts は unstable(sid) だけでなく、現在サポートされている Debian のディストリビューションと Ubuntu のディストリビューションをサポートしています。ディストリビューションの指定には-d オプションを指定します。これは複数指定することができ、指定した順にテストが実行されます。以下の例では、 sid 環境を構築し、テストした後、 squeeze の環境を構築し、テストします。

```
$ sudo piuparts -d sid -d squeeze libcv4_2.0.0-4_i386.deb
.....
```

5.6 debian にインストールされているパッケージのテスト

既に debian にインストールされているパッケージのテストを行う場合には、--apt オプションを使います。特定のディストリビューションにあるパッケージのテストを行いたい場合には、-d オプションでディストリビューションを指定する必要があります。

```
$ sudo piuparts --apt -d squeeze libcv4
```

5.6.1 ミラーサーバの指定

デフォルトでは、 piuparts が使う Debian リポジトリは、/etc/apt/sources.list からパーサして利用します。一家に一台 Debian ミラーの時代です。グローバルなミラーサーバを使用せずにローカルにあるミラーサーバを指定する場合には、--mirror オプションを使って指定します。

```
$ sudo piuparts -p libcv4_2.0.0-4_i386.deb --mirror http://debmirror.example.org/debian
```

5.6.2 依存するパッケージの回避方法

通常、ライブラリソースパッケージは libfoo0、 libfoo-dev など複数のバイナリパッケージを生成します。この場合、libfoo-dev パッケージは libfoo0 パッケージに依存しているので、 libfoo-dev パッケージだけでテストしても、 libfoo0 がないのでテストでエラーになります。この場合には、パッケージを 2 つ (libfoo0, libfoo-dev) 指定することで回避できます。

```
$ sudo piuparts -p libcv-dev_2.0.0-4_i386.deb libcv4_2.0.0-4_i386.deb
```

また、ソースパッケージから作成されたバイナリパッケージのテストを行う場合には、*.changes ファイルを指定します。

```
$ sudo piuparts -p opencv_2.0.0-4_i386.changes
```

5.7 piuparts.debian.org

piuparts がパッケージとして用意されたとしても、まだ利用しているパッケージメンテナは少ないようです。また、パッケージ作成時には問題がなかったが、依存関係があるパッケージが更新および削除され、正常にインストール等ができない状態になる場合があります。そこで、 QA チームは既に Debian にインストールされたパッケージをチェックするためのサービス piuparts.debian.org を立ち上げました。これでチェックされた内容は、 qa.debian.org で提供され

ている情報の一部として、表示されています。

5.7.1 現在チェックエラーになっているパッケージ

piuparts でチェックでエラーになっているパッケージはタグとユーザダグで検索できます。

- タグ：piuparts
- ユーザタグ：debian-qa@lists.debian.org

BTS では以下の URL で参照できます。 http://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=piuparts;users=debian-qa@lists.debian.org

5.8 現在の問題点

現在、 piuparts にはいくつかの問題点があります。 http://packages.qa.debian.org で表示されるチェック結果が誤解を受けやすいという点です。依存しているパッケージが問題を持っているのに、それが自分のパッケージにエラーとなって表示されます。情報を追えばどのパッケージでエラーになっているのか分かりますが、情報を追うのがめんどうです。

細かいところでは、-B の使い方がわかりません。

```
$ piuparts --help
.... 中略 ....
 -B FILE, --end-meta=FILE
                     XXX
```

ちなみに #560050 で 報告されていますが、開発者本人が報告しているので修正する気がないのかもしれません。

5.9 まとめ

今回は基本的な使い方と、パッケージメンテナから利用する場合に使うオプションを説明しました。パッケージメンテナの方は自分でもっているパッケージテストに組み込んでみてはいかがでしょうか。パッケージの基本的な部分の問題が減るのでよいと思います。

また、現在 piuparts v2 を開発中です。開発は bzr 上で行われています。 http://code.liw.fi/piuparts2/bzr/trunk/ ソースコードも多くなく、やっていることも単純です。興味のある方は参加してみてはいかがでしょうか。

6 upstart 再入門

前田耕平

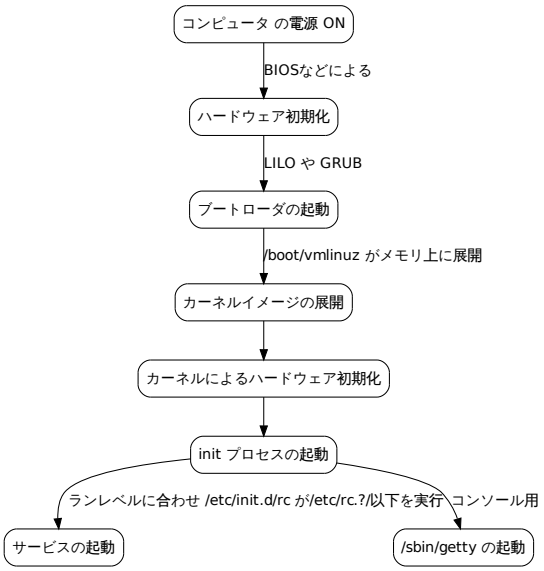
6.1 はじめに

今年の 2 月の Debian 勉強会(Debian 温泉) で一度扱ったテーマですが、新年度も始まり、新入生、新入社員、新入 Debian 開発者予備軍の皆さんも、気分も新たに、いずれやってくるであろう、 Squeeze のリリースに備えもう一度おさらいしておきましょう。[*1]

6.2 従来の init

まず従来の init のおさらいです。従来の init は System V 系の Unix 由来の起動の仕組みで、 sysvinit といい、Unix/Linux システムにおいて、カーネルがブートした後、ユーザプログラムが起動するための仕組みです。マシンに電源を入れてからログインプロンプトが表示されるまの流れは大まかには次のようになります。

図 2 ブートの流れ



init プロセスが起動すると、 init は /etc/inittab の内容に従って、プロセスのを生成や停止を行います。 inittab の書

[*1] 書き下ろすつもりでしたが、個人的な都合で余裕がなく、今回の事前配布資料は基本的に、 2010 年 2 月の Debian 勉強会の資料「 ブート方法が変わるよ」を再編したものです。

式は次のとおりです。

```
id:runlevels:action:process
```

Debian システムのデフォルトランレベルは 2 なので、プロセスの生成に関わる基本的なエントリは次の 4 行です。

```
id:2:initdefault:                       ←デフォルトランレベルは 2
si::sysinit:/etc/init.d/rcS             ←起動時は必ず実行
l2:2:wait:/etc/init.d/rc 2              ←ランレベル 2 で実行。
1:2345:respawn:/sbin/getty 38400 tty1 ← getty を常駐
```

2 行目のエントリは、ランレベルが指定されてませんが、 action に sysinit が指定されているためです。これはシステムブート中に他のブート用の action よりも優先して実行されます。ここで実行される /etc/init.d/rcS では

```
exec /etc/init.d/rc S
```

とだけが実行されます。これはランレベル S のシングルユーザモードのときのものです。上記 3 行目でランレベル 2 のときに実行されるエントリがあることからも分かるとおり、

1. ランレベル S のプロセスが実行
2. ランレベル 2 のプロセスが実行

の順で起動プロセスが実行されます。ランレベル S 用の起動処理が終わってから、ランレベル 2 用の起動処理が実行されるのですから、/etc/rcS.d/ 以下と/etc/rc2.d/ 以下を比較しても分かるとおり、これが逐次実行されるのはかなり時間がかかるでしょう。

6.3 upstart の特徴

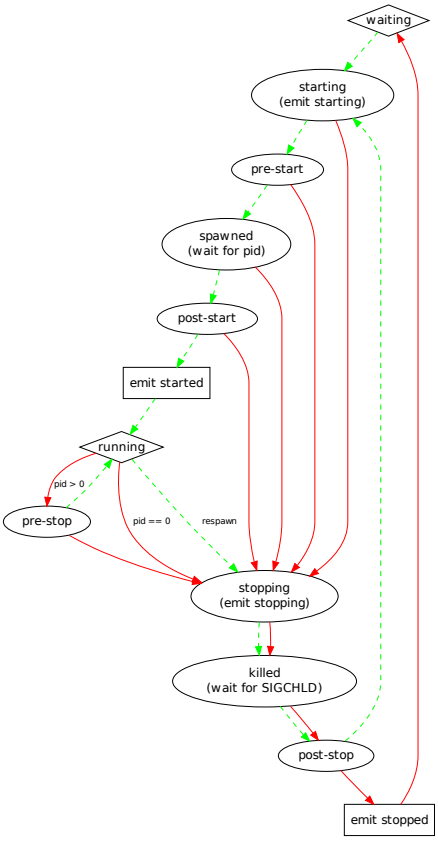
upstart は sysvinit をイベントベースに置き換えたもので、サービスの開始と停止はイベントの通信にもとづきます。

upstart の特徴は次の 6 つです。

- イベントドリブンでタスクやサービスを起動・停止する。
- タスクやサービスが起動・停止することでイベントが発生する。
- イベントはシステム上の他のプロセスから受け取ることができる。
- サービスが予期せず突然終了しても再起動することができる。
- デーモンの監視と再起動は親プロセスから分離できる。
- D-Bus を通じて init デーモンと通信できる。

upstart の状態遷移は次の図のようになります。

図 3　upstart 状態遷移



upstart は、 sysvinit と同様な機能を提供しますが、**非同期イベントに応じて自律的に動作する点**がもっとも異なります。そのため、 sysvinit に対する upstart のメリットには、

- 利用可能なハードウェアだけでブートするため、 runlevel が必要ない。これは存在しないハードウェアを必要とするジョブをトリガーとしないため。
- ホットプラグデバイスに対応

といったことが挙げられます。

例えば、システムがブート後、 NIC を挿すと

1. network-interface-add イベント生成
2. DHCP ジョブがネットワークカードを構成
3. network-interface-up イベントが生成
4. デフォルトルートが新しいインタフェースに割り当て
5. default-route-up イベントが生成
6. NIC を必要とするジョブ (各種サーバ) が自動的に開始される

逆にネットワークカードがなくなった場合は自動的に停止されます。

6.3.1 upstart へ の切り替え

切り替えには upstart パッケージをインストールします。通常のパッケージのインストールとは異なり、続行する場合は、 **Yes, do as I say** と入力しなさい、というメッセージが表示されます。これはうまくいかない場合は Debian システムが起動できなくなるなどのリスクがあるためです。

```
$ sudo apt-get install upstart
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
状態情報を読み取っています... 完了
以下の特別パッケージがインストールされます:
  dbus libdbus-1-3 libexpat1
提案パッケージ:
  dbus-x11
以下のパッケージは「削除」されます:
  sysvinit
以下のパッケージが新たにインストールされます:
  dbus libdbus-1-3 libexpat1 upstart
警告: 以下の不可欠パッケージが削除されます。
何をしようとしているか本当にわかっていない場合は、実行してはいけません!
  sysvinit
アップグレード: 0 個、新規インストール: 4 個、削除: 1 個、保留: 9 個。
1,005kB のアーカイブを取得する必要があります。
この操作後に追加で 2,105kB のディスク容量が消費されます。
重大な問題を引き起こす可能性のあることをしようとしています。
続行するには、'Yes, do as I say!' というフレーズをタイプしてください。
 ?] Yes, do as I say!
```

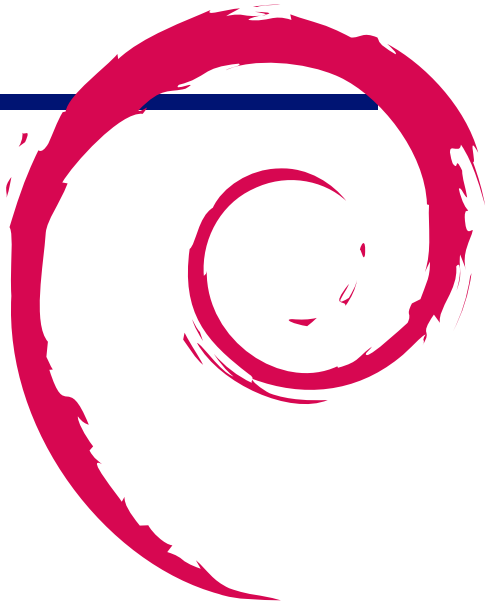
2 月の Debian 勉強会時点で、 lxc の環境で試した際[*2]は、 getty がうまく動かず、起動しては突然死して、再起動されて、また突然死、というのを繰り返してしまい、コンソールからのログインは出来ない状態でしたが、今回の資料作成時点[*3]では、 KVM 環境で問題なく起動します。バージョンが変わってませんが、環境が異なるので原因は後者にありそうではあります。

6.4 参考資料

http://www.ibm.com/developerworks/jp/linux/library/l-boot-faster/index.html

[*2] 2010 年 2 月 9 日現在

[*3] 2010 年 4 月 11 日に、 Squeeze Official Snapshot amd64 BC Binary-1 20100322-03:30 の ISO イメージを使用した最小構成。



7 debtags 入門

やまねひでき

7.1 概要

今日ここでは、「 debtags 便利だぜ! 」というのと「 もっと便利にするためにオラに元気を分けてくれ! 」というのを説明します。

7.1.1 探し物は何ですか?

Debian には大量のパッケージが用意されていますが、最初から全てがインストールされているわけではないので、必要に応じて導入を行います。で、必要なパッケージがわかっているのならば良いですがそうでない場合も度々です。その場合にはパッケージの検索を行ってパッケージを見つけてインストールという流れになります。

欲しいファイル・機能を思いつく→適当なキーワードで検索→見つけたパッケージをインストール

これが多くの Debian ユーザが一般的に行っている流れではないかと思います。

7.1.2 見つけにくいモノですか?

通常の apt-cache/aptitude/synaptic の検索 (search オプション) では、単純なキーワードマッチ検索を行います。しかし、キーワードにマッチするパッケージが少ない場合はともかく、大量にマッチしてしまう一般的なキーワードしか思いつかない場合は、ノイズ混じりの検索結果になって一苦労します。

例えば、ウェブブラウザを普段使っているものから変更したいと思ってパッケージを探してみましょう。

```
$ apt-cache search web browser| wc -l
295
```

いくら Debian にはパッケージが多いといってもこれは多すぎですね。その中身をちょっと見てみると…?

```
<snip>
tor - anonymizing overlay network for TCP
torrentflux - web based, feature-rich BitTorrent download manager
trac-graphviz - Graphs printing plugin for Trac
unhtml - Remove the markup tags from an HTML file
uzbl - Lightweight Webkit browser following the UNIX philosophy
vdetelweb - Telnet and Web interface for VDE 2.x
iceweasel-vimperator - Iceweasel extension to make it have vim look and feel.
<snip>
wwwoffle - World Wide Web OFFline Explorer
xdg-utils - desktop integration utilities from freedesktop.org
xemacs21-bin - highly customizable text editor -- support binaries
xemacs21-gnome-mule-canna-wnn - highly customizable text editor -- transitional package
xemacs21-gnome-mule - highly customizable text editor -- transitional package
xemacs21-gnome-nomule - highly customizable text editor -- transitional package
xemacs21-mule-canna-wnn - highly customizable text editor -- Mule binary compiled with Canna and Wnn
xemacs21-mule - highly customizable text editor -- Mule binary
xemacs21-nomule - highly customizable text editor -- Non-mule binary
xemacs21-support - highly customizable text editor -- architecture independent support files
xemacs21-supportel - highly customizable text editor -- non-required library files
xemacs21 - highly customizable text editor
<snip>
```

ネット接続の匿名化ツールである tor やブラウザのプラグインである iceweasel-vimperator、さらには xemacs21 のパッケージ群までもが検索一覧に含まれてしまっています。これは期待している「ウェブブラウザ」の検索結果ではありません。

ではどうすれば効率的にできるのか? そこで debtags ですよ。

7.2 debtags を導入してつかってみよう

では早速 debtags を導入してみましょう。

```
$ sudo aptitude install debtags
```

これだけです。簡単ですね。では実際の検索を。

```
$ sudo debtags update
$ debtags search web::browser
arora - simple cross platform web browser
chimera2 - Web browser for X
conkeror - keyboard focused web browser with Emacs look and feel
dillo - Small and fast web browser
edbrowse - A /bin/ed-alike webbrowser written in C
elinks - advanced text-mode WWW browser
elinks-lite - advanced text-mode WWW browser - lightweight version
epiphany-browser - Intuitive GNOME web browser
epiphany-extensions - Extensions for Epiphany web browser
epiphany-gecko - Dummy, transitional package
epiphany-webkit - Dummy, transitional package
ezmlm-browse - Web browser for ezmlm-idx archives
galeon - GNOME web browser for advanced users
galeon-common - data for the galeon web browser
iceape-browser - Iceape Navigator (Internet browser) and Composer
iceweasel - Web browser based on Firefox
(snip)
w3m - WWW browsable pager with excellent tables/frames support
w3m-el - simple Emacs interface of w3m
w3m-el-snapshot - simple Emacs interface of w3m (development version)
w3m-img - inline image extension support utilities for w3m
wapua - Web browser for WAP WML pages
```

先ほどよりはずっとまともな情報が検索できるようになっています。キーワードの組み合わせ方が思いつかないという方も、auto complete があるので安心です。

```
$ debtags search web(タブキーで補完)
$ debtags search web\:\:(さらにタブキーで補完)
web::TODO           web::browser        web::forum          web::server
web::application    web::cgi            web::portal         web::wiki
web::appserver      web::cms            web::scripting
web::blog           web::commerce       web::search-engine
```

さらに複数のキーワードを組み合わせての検索もお手の物です。” でくくって && で組み合わせるだけです。

```
$ debtags search 'web::browser && x11::application'
arora - simple cross platform web browser
chimera2 - Web browser for X
conkeror - keyboard focused web browser with Emacs look and feel
dillo - Small and fast web browser
epiphany-browser - Intuitive GNOME web browser
epiphany-extensions - Extensions for Epiphany web browser
epiphany-gecko - Dummy, transitional package
epiphany-webkit - Dummy, transitional package
galeon - GNOME web browser for advanced users
galeon-common - data for the galeon web browser
iceape-browser - Iceape Navigator (Internet browser) and Composer
iceweasel - Web browser based on Firefox
junior-internet - Debian Jr. Internet tools
kazehakase - GTK+-based web browser that allows pluggable rendering engines
konq-plugins - plugins for Konqueror, the KDE file/web/doc browser
konqueror - KDE 4's advanced file manager, web browser and document viewer
links2 - Web browser running in both graphics and text mode
midori - fast, lightweight graphical web browser
netsurf-gtk - Small portable web browser with CSS and Unicode support - GTK version
w3m-el-snapshot - simple Emacs interface of w3m (development version)
wapua - Web browser for WAP WML pages
```

これで「X 上のウェブブラウザっていったらどんなのがあるの? 」が検索できました。他に役立ちそうな例としては「ruby の deb パッケージ開発環境を整えたい」などはいかがでしょう?

```
$ debtags search 'devel::lang:ruby && devel::packaging'
dpkg-ruby - ruby interface for dpkg
ruby-pkg-tools - Tools for building Debian Ruby packages
rubygems1.8 - package management framework for Ruby libraries/applications
```

便利さがいまいちわからない場合は、同じ検索を apt-cache/aptitude のキーワード検索で行ってみてください。debtags rocks!

ちなみに、

```
aptitude ~Gdevel::lang:ruby
```

などとして aptitude をインターフェイスとしてタグ検索が可能になっています(ので、そちらが主に使われるようです)。さらについ先日から、新しいインターフェイスとして axi-cache search が使えるようになりました。

```
$ axi-cache search implemented-in::ruby devel::packaging
182 results found.
Results 1-20:
100% libcairo-ruby1.8 - Cairo bindings for the Ruby language
100% tictactoe - tic-tac-toe game written in Ruby
100% libgnomecanvas2-ruby1.8 - GNOME Canvas 2 bindings for the Ruby language
100% flvtool2 - a manipulation tool for flash video files
100% dpkg-ruby - ruby interface for dpkg
100% libfilesystem-ruby1.8 - Ruby1.8 extension for file-system information
100% libexif-ruby1.9.1 - EXIF tag parsing Library for ruby1.9.1
100% xmms2-scrobbler - Audioscrobbler/Last.FM client for XMMS2
100% ri1.9 - Ruby Interactive reference (for Ruby 1.9)
100% libdbm-ruby - DBM interface for Ruby
100% schleuder - GnuPG enabled mailing list manager with remailer-capabilities
100% libactiveldap-ruby1.8 - an object-oriented interface to LDAP for Ruby
100% libhttp-access2-ruby1.8 - HTTP accessing library for ruby (transitional package)
100% quickml - Very-easy-to-use mailing list system
100% libreadline-ruby1.9.1 - Readline interface for Ruby 1.9.1
100% docdiff - Compares two files word by word / char by char
100% zonecheck-cgi - DNS configuration checker (web interface)
100% libactiveldap-ruby - an object-oriented interface to LDAP for Ruby
100% ohai - Detects data about your operating system and reports it in JSON
100% ditz - distributed issue tracker
More terms: ruby ruby1.8 dependency libactiveldap activeldap interface oriented
More tags: devel::lang:ruby role::program mail::list role::shared-lib devel::library works-with::db interface::commandline
'axi-cache more' will give more results
```

おまけ

```
$ debtags search 'devel::lang:lisp && devel::packaging'
common-lisp-controller - Common Lisp source and compiler manager
dh-lisp - Debhelper to support Common Lisp related packages
$
$ debtags search 'devel::lang:haskell && devel::packaging'
haskell-devscripts - Tools to help Debian developers build Haskell packages
libhugs-cabal-bundled - A framework for packaging Haskell software
libhugs-quickcheck-bundled - Automatic testing of Haskell programs
$
$ debtags search 'devel::lang:ocaml && devel::packaging'
$
```

7.3 で、debtags というのは何ぞや?

Enrico Zini さんによって開発された「パッケージごとに予めリストアップしてある『カテゴリタグ』をつけておき、後々検索しやすくしよう」という取り組みです。ウェブサイトを見ると、どうやらランガナータンの図書コロン分類法からヒントを得て作ったようです&ポイントは複数のタグが付けられるところ。

- 任意のパッケージに対してタグをつけまくる
- タグデータが Alioth に保存される。
  `http://debtags.alioth.debian.org/tags/tags-current.gz` から現在のデータを取得できる(この時点でタグデータは未レビュー)。
- 定期的に Enrico さんと David Paleino さんが登録されたものをすべてレビューし、コミットする。
  コミット先は `http://svn.debian.org/wsvn/debtags/tagdb/tags` で確認できる。使っているスクリプトなどは `http://svn.debian.org/wsvn/debtags/tagdb/sessions/#_tagdb_sessions_`
- コミットされたデータを override ファイルに変換する。

(override ファイルについては https://wiki.debian.org/FtpMaster/Override 参照)

- 変換したデータをアップロードする。
  自動処理スクリプトによってミラーされる。
- ミラーしたデータが apt-get/aptitude update 時に取得され、/var/lib/debtags/package-tags に保存される。
  (apt-xapian-index がインストールされてる場合は、インデックスが作成され /var/lib/apt-xapian-index/ に保存される。インデックス作成時には大体 40MB ほどメモリを食う模様。)
- debtags パッケージのインストール debtags/axi-cache search で検索
- ウマー (AA 略

となります。すばらしいですね。

しかし、最初にタグをつけまくらないと検索しても引っかかりません。ということで、これからタグ付けのお時間です。現在、タグ付けされているパッケージとそうでないパッケージはどの程度あるのでしょうか?

```
$ debtags stats
Total count of packages: 38476
Total count of packages (according to APT): 38476
Total count of packages (according to Debtags): 22816
Number of facets: 31
Number of tags: 583
Number of packages with tags, but no special::not-yet-tagged tags: 22816 (100.0%)
Number of packages with special::not-yet-tagged tags: 0 (0.0%)
Number of packages with only special::not-yet-tagged tags: 0 (0.0%)
Number of packages with no tags: 0 (0.0%)
```

やりがいがありますね!

7.4 どうやってタグを付けるの?

タグをつけるには 3 種類のやり方があります

- CLI (debtags tag)
- GUI (debtags-edit)
- web インターフェイス

このうち、一番取っ付き易いと思われるのは web インターフェイスなので、こちらの説明を中心に行います。

図 4 debtags タグ付け web インターフェイスの様子

- パッケージメンテナの人向け

  まず、 `http://debtags.alioth.debian.org/todo.html?maint=<your_mail_address>` にアクセスしてください。メンテナンスしているパッケージとつけられているタグの一覧が表示されます。自分のパッケージをいい状態にメンテナンスする作業の一環ですよ！ 忘れないで。
- ユーザの方向け

  debtags のサイト (`http://debtags.alioth.debian.org/todo.html`) にアクセス、 Current View を full text search にして自分が良く使っているパッケージの名前を入れます。特に debtags grep で検索してみて「このパッケージが何でこのキーワードで引っかからないんだ！」というのがあれば、それは要改善点なわけなので入力してみるのが良いでしょう。

7.5 実際のタグ付け

適当なパッケージを選んだらタグ付けに入りましょう。サイトの画面は大きく４つに分けられます。

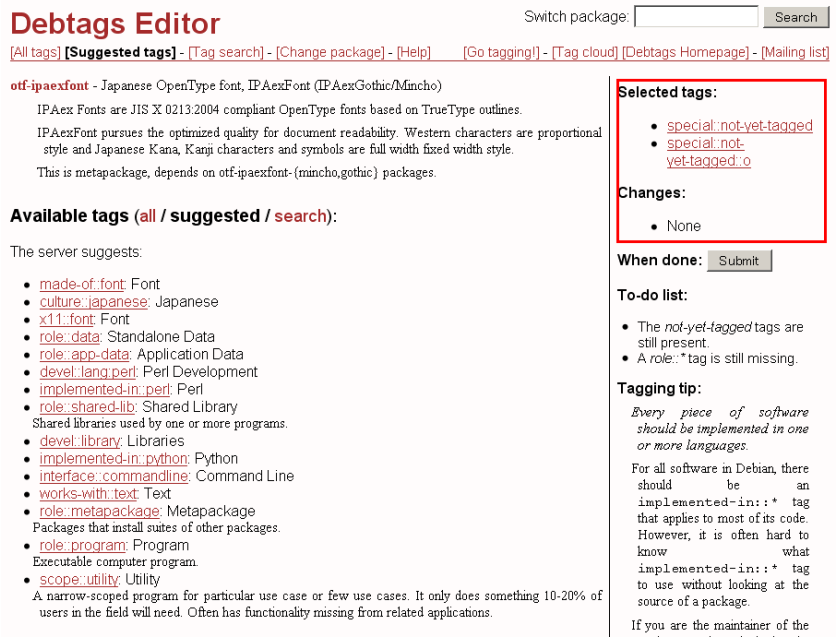


図５　debtags タグ付け web インターフェイスの様子

- 選択したパッケージの説明(画面左上)
- 利用可能なタグの区別:　all (すべて) / suggested (おすすめ) / search (検索)
- タグの一覧(画面左下)
- 既に付けられた・付けられるタグ(画面右)

7.6 修正が必要なタグ

まだキチンとタグ付けがされていないパッケージには、赤い×印と共に以下のような注意が表示されます。ここからまず直していくことを考えましょう。

それぞれ以下のような意味合いです。

表示される注意	意味合い
The not-yet-tagged tags are still present.	このパッケージはまだタグ付けが終わってないよ、のタグが残ってます。
An implemented-in::* tag seems to be missing.	このソフトはほげほげ言語で実装されています、ってタグ付けしようね
A role::* tag is still missing.	このソフトの役割をタグ付けせよ(role は必須です)
A devel::lang:* tag seems to be missing.	ほげほげ言語開発用のタグを付けましょう

表 1　タグ付けがされていないパッケージの注意書き

これを踏まえて以下のような作業をします。

- まだパッケージを指定していないなら、パッケージ名をクリック
- 提示されるタグや検索したタグをクリックして追加
- 必要ないタグが付けられている場合はクリックして削除
- 画面右側の「 Selected tags:」と「 Changes:」を見て、問題がないことを確認する
- 最後に submit

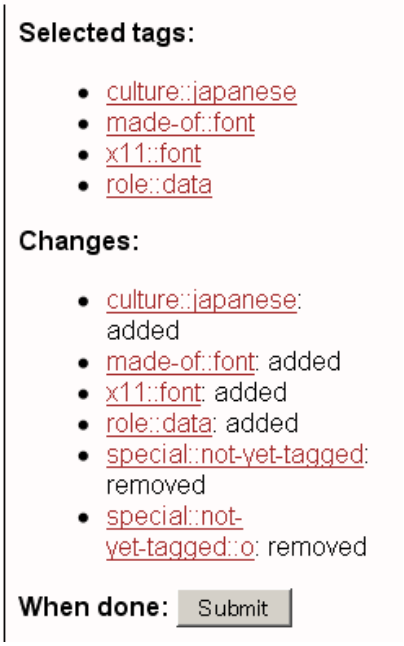


図 6　画面右側の「 Selected tags:」と「 Changes:」を確認!

これだけで作業は終わりです。簡単ですね!

7.7 どんなタグがあるの?

タグ付け自体は簡単なものなので、パッケージに対して適切なタグを付けることが肝要なのですが、すべてのタグを覚えることは大変すぎるのであきらめて、サイトが適宜提示してくれるものの中から選択しましょう。あと一応ガイドラインも

あります (http://wiki.debian.org/DebTaggingGuidelines)
で、最初に「推奨」タグが表示されています。適当なものがあればここから選ぶのもいいですが、お勧めは

- 他の似たパッケージを見てみる→おんなじタグ使う
- all を選んで、検索窓からキーワードでタグを検索してみる

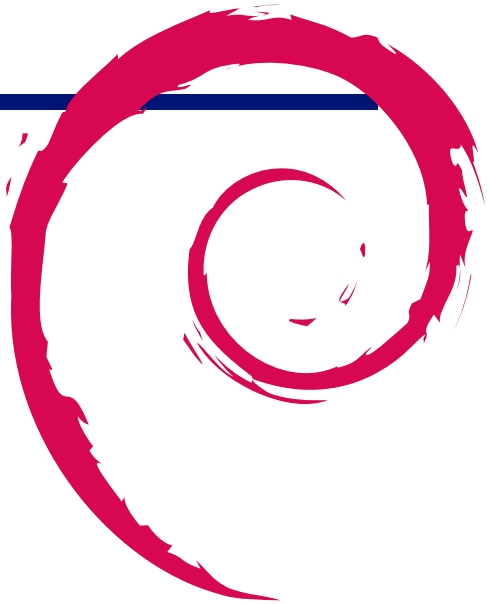
です。

7.8 その他疑問点?

- 悪意のあるコミットについては? spammer などは大丈夫?:
  コミットされたタグについては一応ブラウザのクッキーが紐付けられていて、レビュー時に重宝している模様です。
- コミットは誰でもできる? レビューは? :
  レビューするには Alioth の 'debtags' グループに参加する必要があるそうです。また少なくとも Debian を使っていないと、レビューの分類をする際、細かな点でうまく判断できないだろうということでした。

7.9 最後に

Happy tagging!



8 索引

Debian 勉強会資料

2010 年 4 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）