# Deb専

日本唯一のDebian専門誌

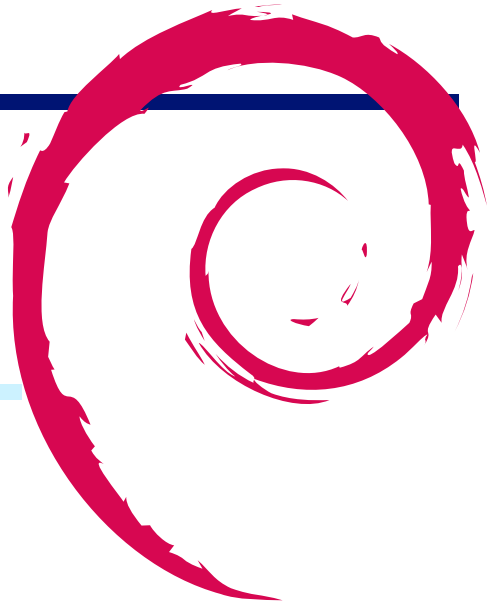
2011 年 08 月 13 日 初版発行

東京エリア Debian 勉強会

関西 Debian 勉強会

# デビアン勉強会

# 目次



# 1 Introduction

上川 純一, 山下 尊也

## 1.1 東京エリア Debian 勉強会

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。

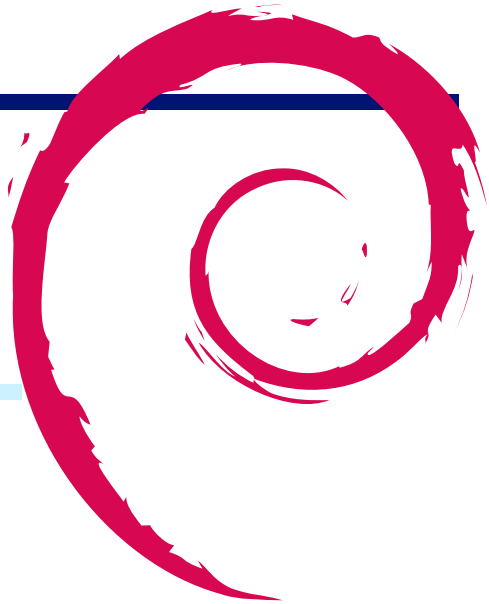
## 1.2 関西 Debian 勉強会

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。



# 2 「フリー」って、どういうこと?

木下 達也

## 2.1 フリーソフトウェアとは

Debian はフリーソフトウェアで構成されたオペレーティングシステムです。ここで言う「フリー」とは「無料」ではなく「自由」を意味しています。

Free Software Foundation (FSF) による『フリーソフトウェアの定義』では次のように説明されています。

> 「フリーソフトウェア」のフリーとはそのソフトウェアのユーザに与えられる 4 種類の自由を意味しています。
> - 目的を問わず、プログラムを実行する自由 (第 0 の自由)。
> - プログラムがどのように動作しているか研究し、そのプログラムに あなたの必要に応じて修正を加え、採り入れる自由 (第 1 の自由)。ソースコードが入手可能であることはこの前提条件となります。
> - 身近な人を助けられるよう、コピーを再頒布する自由 (第 2 の自由)。
> - プログラムを改良し、コミュニティ全体がその恩恵を受けられるよう あなたの改良点を公衆に発表する自由 (第 3 の自由)。ソースコードが入手可能であることはここでも前提条件となります。
>
> (`http://www.gnu.org/philosophy/free-sw.ja.html`)

## 2.2 著作権とライセンス

著作物には、基本的に著作者に対して著作権が発生しており、著作物を変更したりコピーして配ったりするには、著作者からの許可 (ライセンス) が必要になります。

フリーかどうかを判定する際には、変更や配布が許可されているか、それらに付随する制約はどうか、といった点を確認します。

そのほかに、特許、商標など個別の事情についても検討します。

## 2.3 Debian フリーソフトウェアガイドライン

ある著作物がフリーかどうか、Debian の構成要素として適しているかどうかを判定する際の基準、それが Debian フリーソフトウェアガイドライン (DFSG) です。

1. 自由な再配布
2. ソースコード
3. 派生ソフトウェア
4. 原作者によるソースコードの整合性維持
5. すべての個人、団体の平等

6. 目標分野の平等
7. ライセンスの配布
8. ライセンスは Debian に限定されない
9. ライセンスは他のソフトウエアを侵害しない
10. フリーなライセンスの例

(`http://www.debian.org/social_contract#guidelines`)

DFSG の各条項について簡単に説明します。

1. 自由な再配布 (Free Redistribution)
   有償・無償を問わず、別途の許可を必要とすることなく、プログラムを複数まとめて配布できます。
2. ソースコード (Source Code)
   ソースコード (変更に適した形式) が必要です。実行形式だけでなくソースコードでも配布できます。
3. 派生ソフトウェア (Derived Works)
   同様のライセンスで変更版を配布できます。
4. 原作者によるソースコードの整合性維持 (Integrity of The Author's Source Code)
   変更版のソースコードを配布する場合に、元のソースコードと差分 (パッチ) という形式のみ許可という制約は許容します。ただし非推奨です。
5. すべての個人、団体の平等 (No Discrimination Against Persons or Groups)
   いかなる個人・団体も差別せずに許可します。
6. 目標分野の平等 (No Discrimination Against Fields of Endeavor)
   商用・非商用など、用途を制限しません。
7. ライセンスの配布 (Distribution of License)
   ライセンスは、再配布されたすべての人々に、別途の許可を必要とすることなく適用されます。
8. ライセンスは Debian に限定されない (License Must Not Be Specific to Debian)
   Debian の一部としてのみの許可ではなく、他のシステムにも適用できるようにします。
9. ライセンスは他のソフトウエアを侵害しない (License Must Not Contaminate Other Software)
   たとえば、同じ媒体で配布されるソフトウエアすべてがフリーであることを要求しないようにします。
10. フリーなライセンスの例 (Example Licenses)
    GNU General Public License (GPL), BSD License, Artistic License が例として挙げられています。

## 2.4 オープンソースの定義

Open Source Initiative (OSI) による『オープンソースの定義』(OSD) は DFSG を元に作られました。何度か改訂されていますが、現状でも内容は DFSG とほぼ同等のものです。

ここでは、一見して異なる OSD 第 10 条についてのみ説明します。

10. ライセンスは技術中立的でなければならない

(`http://opensource.jp/osd/osd-japanese.html`)

たとえば、GUI 環境で「同意する」ボタンを押すことが必須であってはならない、ということになります。内容としては DFSG/OSD の第 7 条に含まれているものとも言えそうですが、OSD では、いわゆるクリックラップについて、この条項で特に注意を促しています。

## 2.5 コピーレフト

コピーレフトとは、著作物やその変更版について、フリーであることを要求するための方法のことです。

GNU GPL は代表的なコピーレフトライセンスです。GNU GPL で配布されるソフトウェアは、その変更版の配布についても GNU GPL が適用され、フリーソフトウェアであり続けます。

著作物全体への適用が「弱い」コピーレフトライセンスもあります。たとえば、GNU Lesser General Public License (LGPL) で配布されるライブラリは、リンクするプログラム全体についてはコピーレフトが適用されません。そのほかに、Mozilla Public License (MPL)、Eclipse Public License (EPL)、Common Development and Distribution License (CDDL) など、ファイル・モジュール単位でのコピーレフトライセンスもあります。

## 2.6 ライセンスの互換性

複数の著作物を組み合わせて一つの著作物にする場合、それぞれのライセンスによる要求が矛盾しないかどうか、互換性が問題になることがあります。

たとえば、CDDL では GNU GPL とは違った要求をしているので、GNU GPL で配布されるソフトウェアと CDDL で配布されるソフトウェアを組み合わせた著作物は配布できません。

ライセンスの非互換を回避するには、複数のライセンスを適用する (デュアルライセンス)、例外条項を設ける、単純で何でも許可するようなライセンスにする、といった方法があります。

## 2.7 ASP ループホール

GNU GPL には、アプリケーションサービスプロバイダ (ASP) の抜け穴 (loophole) が存在することが知られています。

手元のコンピュータではなく「あちら側」のコンピュータで実行されるウェブアプリケーションの場合、そのウェブアプリケーションが GNU GPL で配布されたソフトウェアを含んでいたとしても、ネットワークユーザーにソースコード込みで再配布されるわけではありません。

その抜け穴をふさぐためのライセンスが GNU Affero General Public License (AGPL) です。ネットワークユーザーへのソースコード提供を求める条文が GNU GPL に加えられたものになっています。

## 2.8 自由と協力

なぜソフトウェアをフリーにするのでしょうか。フリーソフトウェア、オープンソースの関係者それぞれに様々な思惑があることと思いますが、ここでは一つの考えを挙げておきます。

個人の自由を促進し人々が協力し合うこと、自分のためにみんなのために実践できること、それがソフトウェアに関することなのであれば、フリーソフトウェアがよく似合います。

Debian プロジェクトはフリーソフトウェアを強く支持しています。

Happy Hacking!

## 2.9 参考資料

# 参考文献


[1] フリーとは何だろう？ あるいはフリーソフトウェアとは何を意味するのか？, http://www.debian.org/intro/free.ja.html, http://www.debian.org/intro/free.en.html

[2] フリーソフトウェアの定義, http://www.gnu.org/philosophy/free-sw.ja.html, http://www.gnu.org/philosophy/free-sw.html

[3] Debian 社会契約 / Debian フリーソフトウェアガイドライン (DFSG), http://www.debian.org/social_contract.ja.html, http://www.debian.org/social_contract.en.html

[4] Debian – ライセンス情報, http://www.debian.org/legal/licenses/index.ja.html, http://www.debian.org/legal/licenses/index.en.html

[5] DFSG and Software License FAQ (Draft), http://people.debian.org/~bap/dfsg-faq.html
[6] オープンソースの定義, http://opensource.jp/osd/osd-japanese.html, http://opensource.org/docs/osd
[7] Open Source Licenses, http://opensource.org/licenses/index.html
[8] コピーレフトって何?, http://www.gnu.org/copyleft/copyleft.ja.html, http://www.gnu.org/copyleft/copyleft.html
[9] さまざまなライセンスとそれらについての解説, http://www.gnu.org/licenses/license-list.ja.html, http://www.gnu.org/licenses/license-list.html
[10] GNU GPL に関して良く聞かれる質問, http://www.gnu.org/licenses/gpl-faq.ja.html, http://www.gnu.org/licenses/gpl-faq.html
[11] Debian について http://www.debian.org/intro/about.ja.html, http://www.debian.org/intro/about.en.html

# 3 Squeeze の変更点をみんなで見てみよう

倉敷 悟

## 3.1 Squeeze の情報源

主な情報源としては下記のものがあります。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。

- 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`

## 3.2 Debian リリースノートの読み方

### 3.2.1 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。

### 3.2.2 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。

### 3.2.3 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。

## 3.3 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。)

## 3.4 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。

## 3.5 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。[*1]

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。[*2]

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。

[*1] sh は wheezy ?

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

# 4 Squeeze をみんなインストールした結果を語る会

まえだ こうへい

事前課題では Squeeze になってうれしい点、変わった点を挙げてもらいましたが、実際にインストールしてみた結果を語りましょう。

## 4.1 インストールした機器

| メーカー | 機種名 | 誰 |
|---|---|---|
| HP | MicroServer | まえだ |

## 4.2 インストールで困ったこと

| 困ったこと | 解決したか？ | 解決方法 | 担当 |
|---|---|---|---|
| プリンタの設定に少々悩んだ (詳細忘れた) | 解決? | 解決方法を思い出して情報共有する。 | キタハラ |

## 4.3 改善した方が良いと思ったこと

| 改善した方良いと思ったこと | Sid でも再現するか？ | あなたの宣言 | 担当 |
|---|---|---|---|
| libvirt の nwfilter の設定によって、自分で設定している iptables のルールが上書きされる | 未検証 | /etc/libvirt/nwfilter にルールを追加するか、その下をすべて削除して自分で作っているスクリプトに集約する。前者はルールの順番がよく分からないので仕組みを理解する。 | まえだ |
| Klipper の挙動が変わり、中ボタンでのペースが面倒になった。 | | 「クリップボードのアクションを有効にする」で解決するか確認する。しなければ改善案を考えて BTS。 | 山本 |
| 心残りがあること | そのまま Sid にある? | すっきり解決する。 | 岩松 |
| Debian Installer で LVM を一度設定すると削除できない。 | Yes。シェルモードで lvremove, vgremove, pvremove すれば削除できる。 | メニューからも普通に削除できた方が良いんじゃね。 | まえだ |

# 5 Debian のドキュメントをみてみよう

かわだ てつたろう

## 5.1 はじめに

Debian プロジェクトはユーザに向けたドキュメントを提供しています。ドキュメントパッケージを管理する doc-base のクライアントとともに Debian を使っていくうえで有用な Debian 固有のドキュメントをいくつか紹介します。

## 5.2 doc-base

doc-base とは Debian システムにインストールされたドキュメントを管理するユーティリティとドキュメントのメタデータのデータベースを提供するパッケージです。単なる man ページだけでなくオンラインドキュメントを提供する Debian パッケージはインストール時に doc-base に登録し、アンインストール時に登録を解除することが推奨されています。[*3] doc-base に登録されたメタデータを利用するクライアントとして dwww, dhelp, doc-central などが提供されています。これらクライアントを利用すればドキュメントの検索、閲覧が容易に行えます。

## 5.3 doc-base クライアント

doc-base クライアントをいくつか紹介します。

### 5.3.1 dwww

dwww はウェブベースのドキュメントリーダです。

doc-base に登録されたドキュメント、man、info を参照することができます。ドキュメントはインデックス化されており検索することができます。

#### インストール

apt を使用してインストールするだけですが info を表示する info2www の設定が少し必要です。

info2www のシンボリックリンクを作成して info2www をドキュメントルート下 (ここでは /var/www) に公開させます。info2www を公開させなくても動作しますが info を表示する場合に画像などが表示されません。[*4]

またインストール直後はインデックスの作成に少し時間がかかりますのでしばらく待ってから dwww を使用してください。

[*3] Debian Policy Manual: 9.10 Registering Documents using doc-base

[*4] /usr/share/doc/info2www/README.Debian 参照

```
$ sudo aptitude install dwww
$ sudo ln -s /var/lib/info2www /var/www/info2www
```

#### 使い方

WEB ブラウザで `http://localhost/dwww` にアクセスするかコマンドラインで dwww と実行してください。ブラウザに dwww のホームが表示されます。後は WEB ブラウジングするようにリンクを辿るか入力フォームにキーワードを入力して検索を行ない目的のドキュメントを探してください。残念ながら日本語での検索はできません。

検索はコマンドラインから行なうこともできます。例えば debian という検索をコマンドラインから行ないたい場合は以下のように実行すれば WEB ブラウザに debian と検索した結果が表示されます。

```
$ dwww debian
```

### 5.3.2 dhelp

dhelp はウェブベースのドキュメントリーダです。

WEB ブラウザから doc-base に登録されたドキュメントを参照することができます。ドキュメントはインデックス化されており検索することができます。

#### インストール

apt を使用してインストールするだけです。dhelp は WEB サーバを必要としていませんのでここで WEB サーバはインストールされません。インストール直後はインデックスの作成に少し時間がかかりますのでしばらく待ってから dhelp を使用してください。

```
$ sudo aptitude install dhelp
```

#### 使い方

WEB ブラウザで `file://localhost/usr/share/doc/HTML/index.html` にアクセスするかコマンドラインで dhelp と実行してください。ブラウザに dhelp のホームが表示されます。後は WEB ブラウジングするようにリンクを辿ってドキュメントを探してください。この状態では info pages, man pages の参照、ブラウザからの検索を行なうことはできません。

検索はコマンドラインから行なうことになります。例えば debian という検索をコマンドラインから行ないたい場合は以下のように実行すれば WEB ブラウザに debian と検索した結果が表示されます。残念ながら日本語での検索はできません。

```
$ dhelp debian
```

#### WEB サーバの導入

WEB サーバを必要としないのは一つのメリットですが制限もあり不便ですので WEB サーバをインストールして WEB ブラウザからの検索、info と man の参照ができるようにします。dhelp パッケージが Suggests しているパッケージを参考に apache2 と info2www, man2html を追加インストールします。先と同様に info2www の設定を少し行ないます。

```
$ sudo aptitude install apache2 info2www man2html
$ sudo ln -s /var/lib/info2www /var/www/info2www
```

これで WEB サーバを経由で dhelp を使用できるようになりました。dhelp コマンドの結果は WEB サーバ経由で表示されるようになりますが WEB ブラウザに直接アドレスを入力する場合はアドレスを `http://localhost/doc/HTML/index.html` に変更してください。

### 5.3.3 doc-central

doc-central はウェブベースのシンプルなドキュメントリーダです。

WEB ブラウザから doc-base に登録されたドキュメントを参照することができます。doc-central はドキュメントをインデックス化しませんが doc-base のメタデータに登録されたドキュメント名、作者、概要を対象として検索することができます。man は参照することができません。

#### インストール

apt を使用してインストールするだけですが info2www の設定が少し必要です。

```
$ sudo aptitude install doc-central
$ sudo ln -s /var/lib/info2www /var/www/info2www
```

#### 使い方

WEB ブラウザで `http://localhost/dc/` にアクセスするかコマンドラインで doccentral と実行してください。doc-central ではなく doccentral です。ブラウザに Doc-Central のホームが表示されます。後は WEB ブラウジングするようにリンクを辿るか入力フォーム[*5]にキーワードを入力して検索を行ない目的のドキュメントを探してください。

検索はコマンドラインから行なうこともできます。例えば debian という検索をコマンドラインから行ないたい場合は以下のように実行すれば WEB ブラウザに debian と検索した結果が表示されます。

```
$ doccentral debian
```

### 5.3.4 その他に

#### Yelp

Yelp は Gnome のヘルプブラウザです。

Yelp からは man と info そして Gnome のヘルプが参照できますが各パッケージのドキュメントは参照できません。

#### khelpcenter

khelpcenter4 は KDE のヘルプセンタです。

khelpcenter4 からは man と KDE のヘルプが参照できますが info や各パッケージのドキュメントは参照できません。

## 5.4 The Debian Documention Project のドキュメント

doc-base クライアントが使えるようになったところで、パッケージとして提供されている The Debian Documention Project (DDP) のドキュメントをいくつかざっとみていきます。

[*5] 左側フレームの最下部にあります

### 5.4.1 doc-debian

Debian の基礎となる Debian 社会契約や Debian フリーソフトウェアガイドラインといった Debian プロジェクトの基礎となる文書群です。

### 5.4.2 installation-guide

インストールガイドです。

squeeze では 12 種類のアーキテクチャ毎にパッケージが用意されています。[*6]i386 アーキテクチャ用だとパッケージ名は installation-guide-i386 になります。

初めて Debian をインストールする場合に参照することになるでしょう。CD-ROM/DVD-ROM や USB メモリといったメディアからのインストール以外にも debootstrap やネットブートを使用したインストール方法、preseed を利用したインストールの自動化についても説明されています。ちょっと違ったインストールをする場合にも参考になることが記載されています。

手早くインストールの流れを確認するだけなら「付録 A. インストール Howto」を参照してください。もしもの場合に供えて「5.4. インストールプロセスのトラブルシューティング」も見ておくことをおすすめします。

### 5.4.3 debian-reference

Debian システムの広範な概論です。

日本語訳は debian-reference-ja として別パッケージで提供されています。

内容は大変多岐に渡りコンソールの基礎から Debian でのパッケージ管理、ネットワーク設定、プログラミングまで扱っています。DDP の中でも最も充実したユーザマニュアルです。

Unix ライクなシステムを使った経験があまり無いのであればぜひ「1. GNU/Linux チュートリアル」を参照してください。Debian のパッケージについては「2. Debian パッケージ管理」で詳細に説明されています。unstable を使いはじめようという時には「2.6. 壊れたシステムからの復元」が手助けになるかもしれません。その他 Debian を使っている各局面にあわせて項目を参照してみるとよいでしょう。

### 5.4.4 maint-guide

新たにパッケージメンテナになろうという人向けの解説です。

「Debian 新メンテナガイド」、「Debian メンテナ入門」、「ニューメンテナガイド」、「新規メンテナのためのガイド」などと呼ばれています。日本語訳は maint-guide-ja として別パッケージで提供されています。

gentoo パッケージを例にとってパッケージの作成、アップロード、更新手順が説明されています。内容の大半がパッケージ作成に割かれて詳しく説明されていますのでパッケージメンテナになろうという人だけでなく、Debian のパッケージを作ってみようと思う方も是非参照してください。

### 5.4.5 developers-reference

Debian 開発者向けのリファレンスです。

日本語訳は developers-reference-ja として提供されていますが翻訳途中です。

新規メンテナプロセスから始まり Debian Developer が使用できるリソース、パッケージの扱い方が説明されています。特に技術的な事柄以外のパッケージ化についてありとあらゆる情報が記載されています。Debian Developer の活動の一部を垣間見られるドキュメントです。

### 5.4.6 debian-policy

Debian ポリシーマニュアルとポリシーを補足する文書群です。

[*6] squeeze では hppa のサポートが外れ 11 種類のアーキテクチャがサポート対象ですが installation-guide-hppa パッケージが存在します。

Debian ポリシーマニュアルには Debian パッケージの内部構造 (アーカイブ構成、パッケージの各書式など) やオペレーティングシステムとして必要な設計部 (ファイルシステムの階層構造、システムや各種プログラムの設定など) について記載されています。日々議論され更新されています。

日本語訳されたパッケージはありませんが Debian JP Project で Debian ポリシーマニュアル (Version 3.8.3.0) を翻訳したものが WEB で公開されています。[*7] 過去に東京エリア Debian 勉強会でとりあげられていますのでその資料も参考にしてみてください。[*8]

### 5.4.7 その他

**debian-faq**

Debian に関する FAQ がまとめられています。

**harden-doc**

Debian のセキュリティに関することをまとめたマニュアルです。システムのセキュリティに関することからプロジェクトのセキュリティ取り組みについても記載されています。現在、日本語への翻訳が進められています。

**debian-refcard**

いわゆるリファレンスカードです。一枚の紙 (A4 裏表) に重要なコマンドの一覧や参考となる情報をまとめてあります。

## 5.5 最後に

ドキュメントは随時更新されており最新ドキュメントは Debian のユーザ文書[*9]にまとめられていますので確認してみてください。ここにはパッケージになっていないドキュメント (例えばリリースノートなど) もあります。ドキュメントソースは Subversion で管理、公開されておりどなたでも確認することができます。興味があればこちらも覗いてみてください。[*10]

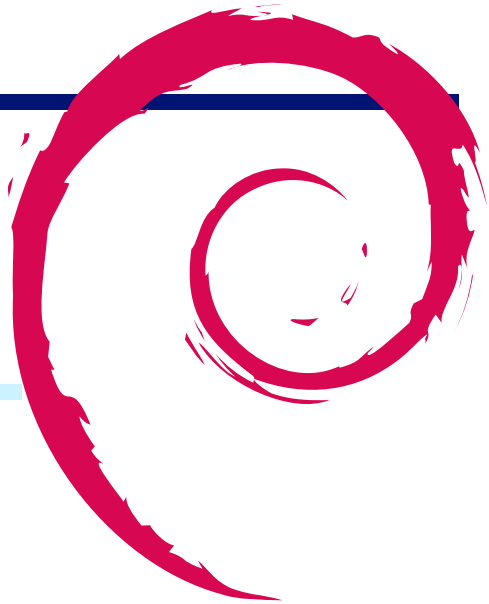
ドキュメントに間違いを見つけた場合は BTS するか、日本語訳に関することであれば `debian-doc@debian.or.jp` に報告するとよいでしょう。

また、The Debian Documention Project が提供するドキュメント以外にもソフトウェアのドキュメントパッケージがたくさんありますのでパッケージの “doc” セクションも覗いてみてください。

## 5.6 参考資料

- Debian ユーザ文書: `http://www.debian.org/doc/`
- doc-base: `http://wiki.debian.org/doc-base/`

[*7] `http://www.debian.or.jp/community/devel/debian-policy-ja/policy.ja.html/`
[*8] 第 12, 13, 15 回東京エリア Debian 勉強会事前資料参照
[*9] `http://www.debian.org/doc/`
[*10] `http://www.debian.org/doc/cvs.ja.html`



# 6　ハッカーに一歩近づく Tips：vi 編

山下 康成

## 6.1　はじめに

vi は、初心者は初心者なりに使え、使い込めば使い込むほど味の出るエディタだということは皆さんご存じの通りかと思います。皆さんも当たり前のように vi を使ってらっしゃると思いますが、さて、ハッカーのように vi を使いこなせてますでしょうか？ハッカーに近づくために、vi 使いこなしのちょっとした Tips を紹介します。

## 6.2　こんな経験ありませんか？

```
    k
    ↑
h ← → l
    ↓
    j
```

w　一語右へカーソルを移動
G　ファイルの末尾へカーソルを移動
x　1 文字消す
dd　1 行消す
etc　……

こんな教え方するから、バラバラなコマンドを覚えないとならず、なかなか「使い込めば味の出る」ところまで行けないのでしょう。今日はもうちょっと体系立てて説明してみます

## 6.3　コマンドを整理してみる

### 6.3.1　繰り返し回数－コマンド 型

回数に続いてコマンドを入力すると、指定した回数繰り返されるコマンド。5yy として 5 行コピーとかは良く使われる。

例:

2l　2 文字分右にカーソルを移動する
2h　2 文字分右にカーソルを移動する
2x　2 文字消す
2X　カーソルの左の 2 文字を消す

2r　2 文字を次に入力する文字と置き換える
2~　2 文字を英大文字 <- >小文字を切り替える
3w　3 ワード右にカーソルを移動する
3W　空白区切りで 3 ワード右にカーソルを移動する
3b　3 ワード左にカーソルを移動する
3B　空白区切りで 3 ワード左にカーソルを移動する
4yy　4 行コピーする
4dd　4 行削除する
5p　5 回ペーストする
5P　カーソルの前に 5 回ペーストする
5i <string >[ESC　] カーソルの左に <string >を 5 回入力する
5I <string >[ESC　] 行頭に <string >を 5 回入力する
5a <string >[ESC　] カーソルの右に <string >を 5 回入力する
5A <string >[ESC　] 行末に <string >を 5 回入力する
5H　画面上から 5 行目にカーソルを移動する
5L　画面下から 5 行目にカーソルを移動する

### 6.3.2 事前課題

以下、それぞれどのようにキー操作をすれば良いか、最も少ないキー操作を考えてください。

- 現在行から最終行（の行頭）にカーソルを移動する→ G
- 現在行から最終行までを消去する→ dG
- 現在行から最終行までをコピー (yank) する→ yG
- 現在行から最終行までを “x” 1 文字で置き換える→ cGx <ESC >

「カーソル移動」というコマンドが省略されていると考えれば、［コマンド］G という形になっていることに気が付いていただいただろうか？

### 6.3.3 コマンドー目的地 型

コマンド：

**省略**「目的地」にカーソル移動「カーソル移動」は「“目的地” までカーソルを移動する」ということ
d　「目的地」までを削除
y　「目的地」までをコピー
c　「目的地」までを変更

例：

d3l　3 文字左まで削除
c3w　3 ワードを変更
d0　行頭までを削除
d^　空白をのぞく行頭までを削除
d$　行末までを削除 (= D)
y5j　5 行下までをコピー
d%　対応するカッコまでを削除
dgg　先頭行までを削除
d/<string >　“string” があるところまでを削除

## 6.4 そのほかの便利な機能

### 6.4.1 名前つきバッファ

例：

"ayy　バッファ a に 1 行コピー
"by$　バッファ b に行末までをコピー
"ap　バッファ a をペースト
"bP　バッファ b をカーソルの前にペースト

### 6.4.2 コマンドの実行結果を追加

:r！ls　ls の実行結果をカーソル位置に挿入

## 6.5 最後に

vi のコマンドは、バラバラのように見えて

- 繰り返し回数－コマンド 型
- コマンド－目的地 型

と整理してみると、結構直行しているということがわかっていただけたかと思います。
皆さんの知ってるコマンドも、この型にあてはめてみると知らなかった用途が見えてきたのではないでしょうか。
vi を使いこなして、ハッカーに一歩近づこう！

# 7 Debian GNU/kFreeBSD で便利に暮らすための Tips

杉本 典充

## 7.1 Debian GNU/kFreeBSD について

Debian GNU/kFreeBSD とは, Debian Project で開発しているオペレーティングシステムのひとつです. Squeeze にて技術プレビュー版としてリリースされることになっています. [*11]

### 7.1.1 Debian GNU/kFreeBSD の現状

Debian GNU/kFreeBSD は現在次のような状況です.

- カーネルは FreeBSD 8.1 ベースのものに一本化された. (7.3 カーネルは廃止).[*12]
- まだ Linux バイナリ互換機能は動かない模様. (linux.ko のロード自体に失敗する).[*13]

## 7.2 ZFS 環境下でリアルタイム系アプリケーションを使用する Tips

### 7.2.1 現象

私の PC 環境では, sftp を実行してサーバから 1GB のファイルをダウンロードすると, audacious で再生中の MP3 ファイルが音飛びします. ディスクのアクセス LED は「消灯, 数秒間連続点灯, 消灯」を繰り返すようなディスクアクセスをしており, 音飛びは HDD LED が点灯中に発生します.

なお, PC の環境は以下です.

- ノート PC の VAIO TZ. 2.5inch HDD (SATA150 接続) の 250GB.
- / は UFS, /home は ZFS としている.
- ZFS の設定で, 圧縮はしない (compression=off) 設定にしている.
- 再生する MP3 ファイルと sftp でダウンロードする 1GB のファイルは双方 ZFS 領域に読み書きする.

### 7.2.2 原因の推定

ディスクのアクセス LED が点灯しているときに MP3 ファイルの再生が音飛びするため, 1GB のファイルを書き込み中に MP3 ファイルの読み込みが行えなかったためと推測します.

そのため, 大きいファイルを書き込む場合はディスクへの連続書き込み時間を短くするように ZFS のパラメータを

[*11] `http://www.debian.org/ports/kfreebsd-gnu/`
[*12] 2011 年 6 月末現在、unstable では 8.2 系カーネル、experimental では 9.0 系カーネルをベースに開発が行われています.
[*13] 2011 年 6 月末現在、まだ動作していません.

調整すれば現象は解決すると考えます.

ZFS のパラメータ設定については FreeBSD を使う人たちで研究されており, その結果が公開されています. [*14]

### 7.2.3 ZFS のパラメータを調整する

ZFS のパラメータを表示するには “sysctl” コマンドを使用します. 今回調整するパラメータは `vfs.zfs.arc_max` のため, 現在の値を確認します.

```
$ sysctl -a | grep arc_max
vfs.zfs.arc_max: 428705280
```

デフォルト値は約 400MB に設定されています.

FreeBSD における ZFS のパラメータ設定は`/boot/loader.conf` に記述するのですが, Debian GNU/kFeeeBSD では`/boot/loader.conf` に記述しても効きません. また, カーネルの起動後に `sysctl` コマンドを直接実行してもすでに ZFS subsystem が起動しているため変更できない状況です.

そのため, grub の起動パラメータとして設定する方法をとりました.

```
$ sudo vim /etc/grub.d/10_kfreebsd
# (省略)
# 元から設定しているパラメータ
set kFreeBSD.vfs.root.mountfrom=${kfreebsd_fs}:${kfreebsd_device}
set kFreeBSD.vfs.root.mountfrom.options=rw

# 今回新しい設定値を追加します.
set kFreeBSD.vfs.zfs.arc_max="100M"
# (省略)
$ sudo update-grub
$ sudo reboot
```

### 7.2.4 確認

```
$ sysctl -a | grep arc_max
vfs.zfs.arc_max: 104857600
```

設定した通り, 100MB になりました. これでテストしてみると音飛びがしなくなりました.

このパラメータ値はハードウェア環境, ディスク負荷で個々に違ってくるため, 皆さんの使い方に合わせてチューニングしてみてください.

## 7.3 USB メモリのマウントに関する Tips

### 7.3.1 まず普通にやってみる

私の Debian GNU/kFreeBSD 環境は `ja_JP.UTF-8` の locale を使用しています. FAT32 でフォーマットしている USB メモリに日本語ファイル名をもつファイルがある場合は mount 時に文字コードの変換が必要になります. そのため, 文字コード変換のオプションを指定してマウントを試みます.

すると `libiconv.so` がないといわれ, エラーになります.

```
$ sudo mount_msdosfs -L ja_JP.UTF-8 -D CP932 /dev/da0 /mnt/usb
mount_msdosfs: Unable to load iconv library: libiconv.so: cannot open shared object file: No such file or directory
: No such file or directory
mount_msdosfs: msdosfs_iconv: No such file or directory
```

### 7.3.2 libiconv のビルド

しかし Debian には libiconv.so というライブラリファイルは存在しません. そのため, GNU のサイトから tarball をダウンロードして build することにします.

[*14] `http://wiki.freebsd.org/ZFSTuningGuide`

```
$ cd
$ mkdir tmp
$ wget http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.13.1.tar.gz
$ tar zxvf libiconv-1.13.1.tar.gz
$ cd libiconv-1.13.1
$ ./configure
$ make
$ sudo make install
$ ls /usr/local/lib
charset.alias  libcharset.so        libiconv.la    libiconv.so.2.5.0
libcharset.a   libcharset.so.1      libiconv.so    python2.6
libcharset.la  libcharset.so.1.0.0  libiconv.so.2  site_ruby
```

再度 mount を試みます. コマンド自体はエラーになりませんが, /mnt/usb の中を ls してみると, 日本語部分が? で表示されています. 文字コードの変換に失敗したようです.

```
$ sudo mount_msdosfs -L ja_JP.UTF-8 -D CP932 /dev/da0 /mnt/usb
```

### 7.3.3 文字コード変換の失敗原因

FreeBSD 本家のカーネルでも mount_msdosfs で UTF-8 へ文字コードを変換する処理は動作しないようです. [*15] そのため, 考えられる方法は以下の 3 つありそうです.

- USB メモリに日本語ファイル名を使わない。(他人からもらったデータの場合は都合が悪い場合もある。)
- locale の設定を ja_JP.eucJP で運用する。(eucJP への変換は動作するため)
- 一時的に locale を ja_JP.eucJP に設定して USB メモリから HDD にコピーし、convmv で UTF-8 へファイル名を変換する。その後、locale を ja_JP.UTF-8 に戻す。

私の環境ではすでに UTF-8 の文字コード環境を使っている都合上, convmv を使った方法を紹介します.
まずは, ja_JP.eucJP 環境で起動します.

```
$ vim .xinitrc

  # UTF-8 の設定があったらコメントアウトして、 eucJP 設定を追加する
  # export LANGUAGE='ja_JP.UTF-8'
  # export LC_ALL='ja_JP.UTF-8'
  # export LANG='ja_JP.UTF-8'
  export LANGUAGE='ja_JP.eucJP'
  export LC_ALL='ja_JP.eucJP'
  export LANG='ja_JP.eucJP'

$ startx
```

ファイルを一度 HDD にコピーし, コピーしたファイル名の文字コードを変換します.

```
$ sudo mount_msdosfs -L ja_JP.eucJP -D CP932 /dev/da0 /mnt/usb
$ mkdir ~/usbtmp
$ cd ~/usbtmp
$ cp -r /mnt/usb/* ./
$ convmv -r -f eucjp -t utf8 * --notest
```

locale を元に戻します.

```
$ vim .xinitrc

  export LANGUAGE='ja_JP.UTF-8'
  export LC_ALL='ja_JP.UTF-8'
  export LANG='ja_JP.UTF-8'
  # export LANGUAGE='ja_JP.eucJP'
  # export LC_ALL='ja_JP.eucJP'
  # export LANG='ja_JP.eucJP'

$ startx
```

一手間必要ですが, locale を ja_JP.UTF-8 で使う場合はこのような回避策を行うことで対応することができます.

[*15] 日本語の UTF-8 へファイル名を変換して mount できるようにするパッチは FreeBSD を使う有志によって作成されているようですが, GENERIC カーネルに取り込まれていないようです.

## 7.4 今後の課題

日常生活で使えると便利な以下の事項については, 現在調査中です.

- iceweasel がよくフリーズする.
- Flash はやはり鬼門で安定して動作できていない. [*16]
- VAIO TZ の液晶の明るさ変更 (Fn キーから音量は何もしなくても変更できているが…).[*17]
- 無線 LAN (iwn0 として内蔵無線 LAN カードの認識はできているがエラーになる)
- IS03 を使ってテザリングする. (まずはドライバをなんとかしないと)

## 7.5 終わりに

Debian GNU/kFreeBSD はこれから更に完成度を高めていけるオペレーティングシステムのため, オペレーティングシステムの開発に関わりたい人には打ってつけのプロジェクトです. 今後も Debian GNU/kFreeBSD を使い, デバッグしながらいろいろ勉強していきたいです.

[*16] 2011 年 6 月末現在、iceweasel + gnash の組み合わせで Flash の再生ができています.

[*17] その後の調査でカーネルのパッケージビルド時に acpi_sony.ko がビルドされていないことが判明しました. カーネルのパッケージを自前でビルドし、その後に手操作で acpi_sony.ko のみをビルドすることで液晶の明るさを変更できるようになっています.

# 8 Debian Games Team 体験記

野島 貴英

ひょんな事から Debian Games Team Project に参加してしまったドタバタの体験記となります。パッケージ初心者の初心者視点の右往左往を楽しんでいただければ幸いです。

## 8.1 あれ？Xmris が無い !?

学生の頃、初めて触ったライセンス Free の UNIX（linux。当時は Slackware）で遊びまくっていたゲームに、Xmris というのがあります。Xmris は、当時、計算機科学の追求の為に導入されていたはずのあらゆる商用 UNIX マシンに何故か必ず入っていたゲーム でした。自分の学校とは全く無関係の施設の UNIX マシンにすら入っていたゲームだったので、思わず MIT X の配布物に梱包されているんだとずっと誤解していたりしました。

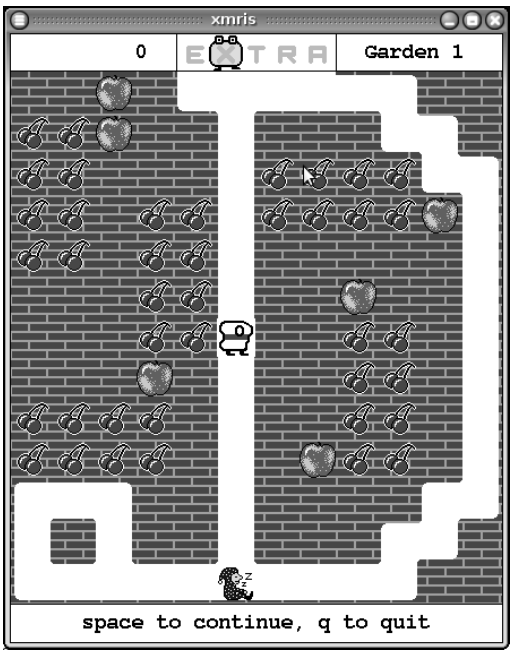



図 1　Xmirs の画面

で、昔を懐しんでこの Xmris がやりたくなり、早速 aptitude search xmris してみたところ、

ごはっ、Xmris がどこにも無いっ !?

でした。で、「もしや？」と思い、方々捜し回ったところ、Debian Wiki の Games/Suggested のページ (`http://wiki.debian.org/Games/Suggested`) のパッケージにして欲しいリストに Xmirs の名前が燦然と輝いているで

はありませんか。

## 8.2 Debian Games Team へ

Xmris がとにかくやりたかった為、完璧にパッケージ作成初心者ではありましたが、Debian Policy 文章、apt-get source コマンド、lintian コマンドのお陰で苦労なくパッケージを作成できました。

が、パッケージは作ったものの、当時、自分の周りには DD は全くおらず、「どうやったらパッケージを取り込んでもらえるのだろう？」と言う事で悩みまくることになりました。で、辿り着いたのが Debian Game Team でした。

右も左も分からない状況で Debian Games Team の ML（`debian-devel-games@lists.debian.org` 以下 ML）でおそるおそるやり取りしたところ以下の事がわかりました。

**参加資格**

実際に Game のパッケージをつくろうとしている人はだれでも (初心者ならなおさら入って他の人のパッケージの作り方とやりとりを見てくれと言われた)

**参加方法**

こんな感じで進めます。

Step 1. まず ML に参加したい意志表明を流す（例：これこれのパッケージを作りたいので入れて欲しい等）

Step 2. 次に alioth.debian.org（以下 alioth）にアカウントを作り（`http://alioth.debian.org/account/register.php`）、alioth 上の Debian Games Team のページ (`http://alioth.debian.org/projects/pkg-games/`) にある “Request to join” のリンクから参加のリクエストを送る。

Step 3. 後日 Debian Games Team の管理者から Wellcome レターが届き、alioth 上の Debian Games Team のページの Project Members の欄に名前が登録される。

**実際の開発**

マニュアル（`http://wiki.debian.org/Games/VCS`,`http://wiki.debian.org/Games/VCS/git`）を見て VCS/ML/IRC を活用しながらパッケージ開発を進めることになります。(IRC は、irc.debian.org サーバー上の#debian-games チャンネル)

Step 1. マニュアルに従い作ろうとしているパッケージのリポジトリを堀る。（勝手にバンバン掘って良いとの事です）

Step 2. git コマンドを使ってどんどんパッケージの修正をアップロードしていく。

Step 3. パッケージが完成したら、mentor.debian.net にアップロードして、mentor を募る。うまく mentor がつけば、Debian のパッケージリリースのプロセスにのるハズ。

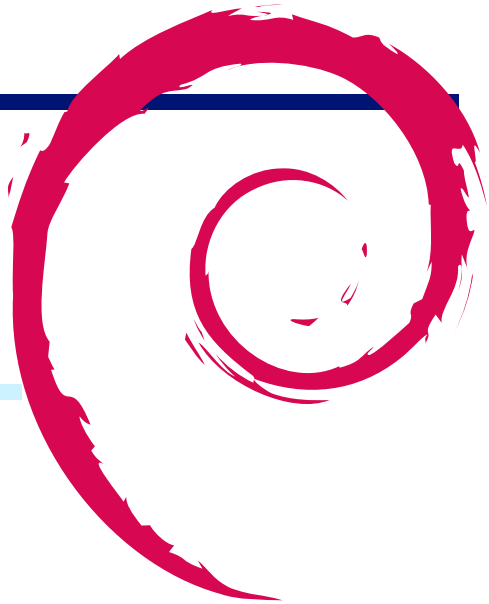
## 8.3 私見

以下に Debian Game Team に関する私見をば。

- ゲームを debian パッケージにしたい人が情報交換をする為に単に集まる寄合い場所のような位置づけなので、Debian Game Team として、ゲームのパッケージを作成する以外の明確なビジョンとかルールがあるわけではない。その為、 Debian Game Team のエリアに自分の git リポジトリを掘るときに「こうしていいか？ああしていいか？」といちいち許可を求めると、ウザがられるのは驚いた（最後はあなたの好きにしてくれ、本当にまずかったら指摘を受けるからその時に覚えてくれといわれてしまった。まあ、聞き方の問題だったのかもしれない...）
- きっと Debian Game Team の関係者と面識が在るといろいろやりやすいと思う。
- Debian Game Team 以外でも身の回りに相談できる DD がいると大変違う（Debian で常識、Debian Game Team で常識をうまく切り分けられる）
- 規模大き目のゲームをパッケージにまとめる際、付属しているバイナリデータ群に別のライセンスが適用/良

くわからないライセンスが適用されていることがあり、こちらの扱いで頭を抱える事がある。（キャラクタの画像、背景の画像、ゲーム内部で使われているフォント、サウンド etc...）

- 手のかかる問題の解決/ちょっと揉めるリスクのある問題の解決（ライセンス問題、以前のパッケージメンテナ/アップストリームに結局連絡取れない、upload に関してスポンサーがつかない等）については、消極的。結局、こちらはパッケージメンテしようとする側にて解決が求められる。
- ML に流れるやりとりよりも、IRC でのやり取りの方が多い印象。また、話も早い事が多い。ML で回答が得られなくてもがっかりせず、その場合は IRC へ突撃をお勧め。
- 空気を読むと package を作成する方法、debian policy などは基礎知識としてある事が前提。
- チームに所属している人の生活があふれるやり取りが IRC で日常的に行われているのは新鮮だった。例：今、「会社でたよー。ちょっとまってねー」→ 「はい家についたから繋いだよー。〜の件はねー」と会話が続く。日常生活の一部として、Contribution 活動が組み込まれてしまっているのは自分にとっては感動ものでした。



# 9　Kinect を Debian で使ってみた

山田 泰資

## 9.1　Kinect って何？

Kinect は Microsoft が 2010/11 に XBox360 の新しい操作デバイスとして発売したゲームデバイスで、強力な画像認識機能を持ち、一切コントローラなどを手にしないゲームプレイを実現します。

ハードウェアとしての最大の特徴は、画像認識と言っても単なるカメラではなく、深度センサを搭載していることです。この種の深度センサ付カメラは従来もあったのですが、Kinect は大量生産されることもあって価格が 1 万 4 千円前後と従来と比べてまさに桁違いの安さで、このため PC などの周辺機器として流用できないかと発売前から高い注目を集めました。

## 9.2　発売、そして Kinect ハックへ

そして発売されると普通に USB 接続ができることが判明（単品販売品のみ。XBox360 同梱版はケーブルを加工する必要あり）し、すかさず画像・深度データのキャプチャ方法が解析されます。こうしてまず生まれたのが libfreenect です。libfreenect は急速に開発が進み、

1. カメラ（通常、深度、赤外線）画像の取得
2. カメラのチルト制御
3. 加速度センサの読み出し
4. 筐体 LED の制御

とサウンド関係以外はほぼ制御できるようになり、各種言語へのラッパーが作られたり OpenCV などの既成の画像認識ライブラリと組み合わせてデモが作成されるなど、あっというまに普及が進みました。ここまでが発売後 1 ヶ月程の話になります。

しかし、libfreenect だけでは限界がありました。XBox360 では kinect を使ったゼスチャ認識などの高度な機能を持ったアプリケーション（ゲーム）を作成できる環境が用意されているのですが、libfreenect はあくまでハードウェアにアクセスするまでなので、その部分が欠けていたのです。

実は前評判では「本体側の負荷を下げるため、Kinect 側でジェスチャ認識などの高水準認識も分担する能力がある」という話もあったのですが、最終的には深度センサまでがハードウェアで、残りは XBox360 向けのソフトウェアライブラリで実現する形となっていたのでした。

しかし、ここで新展開が起こります。Kinect の深度センサと上記ライブラリの原型提供を行った PrimeSense 社から、

1. Linux, Windows 用の Kinect 搭載センサ用ドライバを OSS として公開する
2. ゼスチャ認識機能を備えた OpenNI フレームワークを OSS として公開する

3. 骨格認識および高度なジェスチャ認識を行う NITE ライブラリを無償提供する[*18]

という発表が行われ、純正の XBox360 開発環境と遜色がなくなり、人気がさらに高まりました。

これを生かして年末にかけては 3D モデリングツール（MikuMikuDance）にモーションキャプチャエンジンとして組み込んだり、年明けには AR ウルトラセブン（画面中の自分の姿にウルトラセブンのコスプレをオーバレイし、ジェスチャにあわせて様々な必殺技を繰り出すことができる）が登場するなど、日本では現在進行形で爆発的な盛り上がりを見せています。

ここでは OpenNI および NITE の利用方法を紹介しつつ、それを元に Debian 勉強会のプレゼンテーションをジェスチャで行う支援ツールの作成までを紹介します。

## 9.3 OpenNI と NITE の導入

まずはインストールです。

1. 独自に拡張が始まっている OpenNI とドライバ (SensorKinect) は github から
2. NITE はバイナリのため PrimeSense 社サイト (`http://www.openni.org/`) から

それぞれ入手します。

現段階ではバージョンの組み合わせによっては動作しないことがあり、ここでは以下の組み合わせで導入します：

1. OpenNI (stable) / `https://github.com/OpenNI/OpenNI.git`
2. SensorKinect (avin2 version) / `https://github.com/avin2/SensorKinect.git`
3. NITE-Bin-Ubuntu-x64-1.3.0.17.tar.bz2

いくつかのパッケージを入れるなどビルド環境を整えておく必要はあります[*19]が、後は付属のドキュメントの通りに進めればトラブルなく導入できました。

まずは OpenNI 本体をインストールします：

```
$ git-clone https://github.com/OpenNI/OpenNI.git
$ cd OpenNI/Platform/Linux-x86/CreateRedist
$ ./RedistMaker # ビルドおよびインストール用バイナリセットを作ります
...
$ cd ../Redist
$ pwd
/d/src/openni/OpenNI/Platform/Linux-x86/Redist
$ sudo ./install.sh
copying shared libraries...OK
copying executables...OK
copying include files...OK
creating database directory...OK
registering module 'libnimMockNodes.so'...OK
registering module 'libnimCodecs.so'...OK
registering module 'libnimRecorder.so'...OK

*** DONE ***
$
```

次にドライバ（といっても、Linux 版のものは単なる共有ライブラリです）：

[*18] OSS ではなく、無償利用のためのライセンスコードが公開された

[*19] apt-get install gcc python2.6 libusb-1.0-0-dev freeglut3-dev doxygen graphviz あたりで整うかと思います

```
$ git-clone https://github.com/avin2/SensorKinect.git
$ cd SensorKinect/Platform/Linux-x86/CreateRedist
$ ./RedistMaker # ビルドおよびインストール用バイナリセットを作ります
...
$ cd ../Redist
$ pwd
/d/src/openni/SensorKinect/Platform/Linux-x86/Redist
$ sudo ./install.sh
creating config dir /usr/etc/primesense...OK
copying shared libraries...OK
copying executables...OK
registering module 'libXnDeviceSensorV2.so' with OpenNI...OK
registering module 'libXnDeviceFile.so' with OpenNI...OK
copying server config file...OK
setting uid of server...OK
creating server logs dir...OK
installing usb rules...OK

*** DONE ***
$
```

そして最後に NITE のインストールです：

```
$ tar xf NITE-Bin-Ubuntu-x86-1.3.0.17.tar.bz2 -C /d/src/openni/
$ cd Nite-1.3.0.17
$ sudo ./install.bash -l=0KOIk2JeIBYClPWVnMoRKn5cdY4=
... バイナリを/usr/lib 等にコピーした後、サンプルコードをビルドする...
$
```

最後の NITE の導入で指定した「0KOIk2JeIBYClPWVnMoRKn5cdY4=」が無料利用のための公開ライセンスキーになります。

これらのインストールが終わると

1. /usr/lib に libnim*.so, libXn*.so, libXnV*.so
2. /usr/include/ni,nite に各種ヘッダファイル
3. /usr/bin に niReg, niLicense コマンド（モジュールやライセンスの管理用）
4. /var/lib/ni にモジュールやライセンスの管理用 XML ファイル
5. /usr/etc/primesense に OpenNI やモジュールの設定・データファイル

という構成になります。/usr/etc/primesense は LFS 的におかしな位置のため、/var/lib/ni/modules.xml から参照しているパス設定を変更の上、/etc に移動させるとよいかもしれません。[*20]

あと、サンプルコードを動かせるよう一部ファイルを修正しておきましょう。NITE を展開した Nite-1.3.0.17/Data/ 下の XML ファイルを以下の内容にします。ライセンスキーの書き込みと、キャプチャ設定の修正・追加になります。

Sample-Scene.xml の内容：

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="true" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="false"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="Scene" />
  </ProductionNodes>
</OpenNI>
```

Sample-Tracking.xml の内容：

[*20] OpenNI のパッケージを作成予定ですが、このあたりは検証後直す予定

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="false" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="true"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth" name="Depth1">
      <Configuration>
      <MapOutputMode xRes="640" yRes="480" FPS="30"/>
      <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="Gesture" />
    <Node type="Hands" />
  </ProductionNodes>
</OpenNI>
```

Sample-User.xml の内容：

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="true" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="false"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth" name="Depth1">
      <Configuration>
      <MapOutputMode xRes="640" yRes="480" FPS="30"/>
      <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="User" />
  </ProductionNodes>
</OpenNI>
```

これらの内容の意味については次項で解説します。

## 9.4 Kinect を叩いてみる - OpenNI を使ってみよう

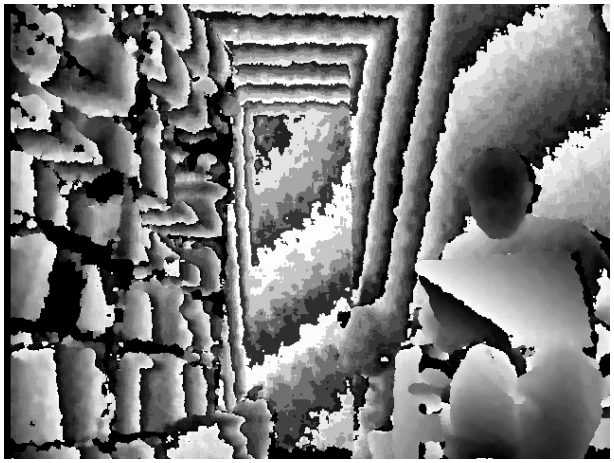
それでは OpenNI から使ってみましょう。以下はサンプルを少しいじって、深度カメラを叩いて深度情報を取得し、それを PPM 画像の形でダンプするようにしてみたものです。

```
/*BINFMTCXX: -Wall -I/usr/include/ni -I/usr/include/nite -lOpenNI
 *
 * 深度情報のキャプチャを行い、PGM 画像として保存する。
 * カメラからの距離がピクセルの濃淡で表される画像になる。
 *
 */

#include <XnCppWrapper.h>

#define log(...) do { \
        fprintf(stderr, __VA_ARGS__); fprintf(stderr, "\n");    \
    } while (0)

#define die(...) do { \
        fprintf(stderr, __VA_ARGS__); fprintf(stderr, "\n"); exit(1);   \
    } while (0)

// 深度センサが返すデータは深度が 16bit 値で左上から並んでいるので、
// そのまま 16bit PGM としてダンプすれば画像になる。
void
ppmsave(xn::DepthMetaData& meta, const char *outfile) {
    const XnDepthPixel* pix = meta.Data();

    XnUInt32 xmax = meta.XRes();
    XnUInt32 ymax = meta.YRes();

    // dump 16bit depth data as 16bit PGM
    FILE* out = fopen(outfile, "w+");
    fprintf(out, "P5\n%d %d 65535\n", xmax, ymax);
    fwrite(pix, sizeof(XnDepthPixel), xmax * ymax, out);
    fclose(out);
}

int
main(int argc, char **argv) {
    xn::Context context;
    xn::EnumerationErrors errors;
    xn::DepthGenerator dgen;
    xn::DepthMetaData meta;

    if (argc < 3) {
        die("Usage: %s kinect.xml depth.pgm", argv[0]);
    }

    // XML 構成ファイルでの初期化、深度カメラへの参照取得、そしてキャプチャ
    context.InitFromXmlFile(argv[1], &errors);
    context.FindExistingNode(XN_NODE_TYPE_DEPTH, dgen);
    context.WaitOneUpdateAll(dgen);

    // キャプチャデータの取得と保存
    dgen.GetMetaData(meta);
    ppmsave(meta, argv[2]);

    context.Shutdown();
    return 0;
}
```

これを実行すると、[*21]

```
$ ./camera3d.cc kinect.xml depth.pgm
```

次のように各ピクセルの深さが画像として取れます：



[*21] binfmtc パッケージをインストールしておくと、実行権限をつけておいた C++ プログラムを直接シェルから実行するとコンパイル・実行してくれるようになります。

正直何がなんだか？という画像ですが、右側にいるのがノート PC を膝において座っている私です。

ここでの中核部分は以下の 3 行です。

```
context.InitFromXmlFile(argv[1], &errors);
context.FindExistingNode(XN_NODE_TYPE_DEPTH, dgen);
context.WaitOneUpdateAll(dgen);
```

OpenNI ではシステム中にどのようなデバイスが存在するか、また、それらの構成（カメラなら解像度など）はどうなるかを外部の XML で記述でき、それを元に初期化を行います。コードだけでも書くことはできるのですが、その場合は構成を変更するとコードの書き換えが必要になります。

このコンテキスト（context）を作成し、そこにデバイスやフィルタ（未登場ですが、センサデータの入力を受けてゼスチャ認識を行い、登録されているフックをトリガするものなどがあります）といった OpenNI モジュール群を接続する、というのが OpenNI/NITE のプログラムフローになります。

最後に、上で利用した kinect.xml も引用しておきます：

```
<OpenNI>
  <!-- ログ設定。以下の設定で標準出力は使わず Log/*.log に書く -->
  <Log writeToConsole="false" writeToFile="true">
    <LogLevel value="0"/>
    <Masks>
      <Mask name="ALL" on="true"/>
    </Masks>
    <Dumps />
  </Log>

  <ProductionNodes>
    <Node type="Image" name="Kin2D">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>

    <Node type="Depth" name="Kin3D">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>

    <Node type="Scene" />
    <Node type="User" />
    <Node type="Gesture" />
    <Node type="Hands" />
  </ProductionNodes>
</OpenNI>
```

以後の例もすべてこの XML を使って動かしています。

## 9.5 ジェスチャの検出方法 - NITE を使う

それでは次はジェスチャの認識をさせてみましょう。こちらでは NITE ライブラリを利用します。

これも OpenNI フレームワーク上のプログラミングなのでコンテキストを生成する点は同じですが、NITE ではコンテキストを介した深度センサデータのやりとりのような低レベルの操作は隠され、

1. ジェスチャー操作の流れを管理する、コンテキストをラップするセッションマネージャ
2. セッションマネージャに登録される各種のジェスチャー検出器
3. それら検出器に登録される、操作検知時に呼び出されるコールバック

という構造のコードになります。操作中・操作終了をセッションマネージャが判定し、「セッション中」の動作のみが検出器にかかるよう調整しています。

実際にどのようなものになるか、見てみましょう：

```
/*BINFMTCXX: -Wall -I/usr/include/ni -I/usr/include/nite -lXnVNite -lOpenNI
 *
 * 簡単なジェスチャ認識のテスト。
 * 手を振ってセッション開始すると、プッシュとスワイプ動作を認識する。
 *
 */

#include <XnCppWrapper.h>
#include <XnVNite.h>

/***************************************************************************
 * Utility Functions
 ***************************************************************************/

#define log(...) do {                       \
        fprintf(stderr, __VA_ARGS__);       \
        fprintf(stderr, "\n");              \
    } while (0)

#define die(...) do {                       \
        fprintf(stderr, __VA_ARGS__);       \
        fprintf(stderr, "\n");              \
        exit(1);                            \
    } while (0)

/***************************************************************************
 * セッションマネージャや検出器に登録するコールバック関数
 * 現在は個々のジェスチャーを認識した等のログを出すのみ。
 ***************************************************************************/

static void
OnSessionStart(const XnPoint3D &p, void *data) {
    log("session start");
}

static void
OnSessionEnd(void *data) {
    log("session end");
}

static void
OnFocusStartDetected(const XnChar *focus,
                     const XnPoint3D &p, XnFloat progress, void *data) {
    log("focus start");
}

static void
OnSwipe(XnVDirection dir, XnFloat speed, XnFloat angle, void *data) {
    log("swipe detected. dir=%d, speed=%f, angle=%f", dir, speed, angle);
}

static void
OnPush(XnFloat speed, XnFloat angle, void *data) {
    log("push detected. speed=%f, angle=%f", speed, angle);
}

/***************************************************************************
 * Main
 ***************************************************************************/

int
main(int argc, char **argv) {
    xn::Context context;

    if (argc < 2) {
        die("Usage: %s kinect.xml", argv[0]);
    }

    context.InitFromXmlFile(argv[1]);

    // セッションマネージャと各種検出器の生成
    XnVSessionManager* sm = new XnVSessionManager;
    XnVSwipeDetector*  sd = new XnVSwipeDetector;
    XnVPushDetector*   pd = new XnVPushDetector;

    // セッションマネージャの初期化
    //
    // 手を振って（Wave）セッションに入り、休止状態からのセッション再開は
    // 手を上げる（RaiseHand)。セッション中は下で登録された検出器が操作を
    // 検出するようユーザの動きをそれらに伝達する。
    sm->Initialize(&context, "Wave", "RaiseHand");
    sm->RegisterSession(NULL,
                        OnSessionStart, OnSessionEnd, OnFocusStartDetected);

    // ジェスチャー検出器と検知した際のコールバックを登録
    sm->AddListener(sd);
    sm->AddListener(pd);
    sd->RegisterSwipe(NULL, OnSwipe);
    pd->RegisterPush(NULL, OnPush);

    // 後はずっと入力を待ってはイベント処理するループ
    context.StartGeneratingAll();
    for (;;) {
        context.WaitAndUpdateAll();
        sm->Update(&context);
    }
    return 0;
}
```

最初の深度データの画像を保存する例よりも簡単になりました。上では XnVSwipeDetector と XnVPushDetector の 2 種類だけを使いましたが、NITE には 10 を超える様々な基本ジェスチャーの検出器があり、また、複数のジェスチャーを組み合わせた操作に対応するための複合処理などを行う補助クラスも用意されています。

このように、多種多様な検出器を用意して、煩雑な体の各部の動きのトラッキングや照合の手間を NITE は省いてくれます。

各種の操作と認識率

なお、動かしてみるとわかりますが、動作によって「きちんと認識される」度が結構違います。スワイプは「一瞬手を止めて、それからさっと流す」動作なのですが非常に認識率が悪いです。一方、プッシュ操作はほぼ確実に認識されます。プッシュ（Push, Click）動作は手振り（Wave）動作と並んで「セッション開始のシグナル動作（focus gesture）」として実用的とマニュアルに書かれていますが、よくわかります。快適な操作感のためには結局カスタムの検出器を作成する必要がありそうです。

## 9.6 アプリケーションの制御方法 - XTest の話

さて、これまで OpenNI, NITE と例を出してきましたが、次はこれで認識したジェスチャでどう他のアプリケーションを制御するかです。今回は X11 の xtst(XTest) 拡張でマウスやキーイベントを送り、それで（この資料の）プレゼンテーションが行えるようにします。

XTest 自体の使い方は以下のようになります。

```
/*BINFMTCXX:-Wall -lXtst -lX11
 * XTest を使ったキーとマウスの制御テスト
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/extensions/XTest.h>

#define log(...) fprintf(stderr, __VA_ARGS__)
#define die(...) do { fprintf(stderr, __VA_ARGS__); exit(1); } while (0)

enum { DOWN = 1, UP, DOWNUP };

void
sendkey(Display *disp, unsigned int keysym, int mask) {
    unsigned int kc = XKeysymToKeycode(disp, keysym);
    if (mask & DOWN) { XTestFakeKeyEvent(disp, kc, True, CurrentTime); }
    if (mask & UP)   { XTestFakeKeyEvent(disp, kc, False, CurrentTime); }
    XSync(disp, False);
}
int
main(int argc, char **argv) {
    Display *disp;

    if ((disp = XOpenDisplay(NULL)) == NULL) {
        die("Unable to open display\n");
    }

    // キー入力のテスト - とりあえず ls させてみる
    unsigned int key[] = { XK_L, XK_S, XK_Return };
    for (int i = 0; i < (int)(sizeof(key) / sizeof(*key)); i++) {
        sendkey(disp, key[i], DOWNUP);
    }

    // 相対位置移動
    XTestFakeRelativeMotionEvent(disp, -100, -100, CurrentTime);
    XSync(disp, False);
    sleep(3);

    // 絶対位置移動
    XTestFakeMotionEvent(disp, -1, 100, 100, CurrentTime);
    XSync(disp, False);
    sleep(3);
```

```
    // XTest の FakeMotion でなぜか XMing 上のポインターが動かないので、
    // XWarpPointer を使ってみるテスト … これだと動くようだ。
    XWarpPointer(disp, None, None, 0, 0, 0, 0, 200, 200);
    XSync(disp, False);
    sleep(3);

    // Control 押しながら右クリックを実行（メニューが出るか確認）
    sendkey(disp, XK_Control_L, DOWN);
    XTestFakeButtonEvent(disp, 3, True,  CurrentTime);
    XTestFakeButtonEvent(disp, 3, False, CurrentTime);
    sendkey(disp, XK_Control_L, UP);
    XSync(disp, False);

    // 終了
    XTestDiscard(disp);
    XCloseDisplay(disp);

    return 0;
}
```

操作したいアプリケーションに合わせて

1. XTestFakeKeyEvent
2. XTestFakeButtonEvent
3. XTestFakeMotionEvent
4. XTestFakeRelativeMotionEvent

で操作イベントを送りながら、XSync で同期を取る（ポインタを移動させて入力、のような操作で、移動が完了してから入力がされるようにする）というのが基本的な処理になります。

あとはこれを先の OpenNI/NITE のジェスチャ検知をトリガに発行してやれば、今回のデモは完成になります。

## 9.7 デモ

さて当日うまくいくか …（当日の結果：いくら Kinect の前で踊っても反応が鈍すぎで、スライドを 3 枚めくった所で聴衆からお役御免を言い渡されたのだった … 。NUI 難しい）

# 10　俺の libsane が火を噴くぜ！

本庄 弘典

SANE は Scanner Access Now Easy の頭文字を取ったもので、各種デバイスから画像を取り出すための汎用インターフェースです。主にスキャナから画像を取得するため、Linux や FreeBSD などフリーな OS で利用されます。

今回は SANE の仕組みと libsane を用いたプログラムの作成方法について解説します。

## 10.1　SANE の構造

SANE を利用するアプリケーションは libsane.so という共有ライブラリをリンクします。libsane.so は dll と呼ばれる仮想デバイスドライバを通し、各デバイスごとに作られたドライバにアクセスすることでデバイスを制御します。また図に示されるように、net 仮想ドライバで他のマシンの saned に接続することで、リモートスキャナへのアクセスを実現します。[1, 2, 3]

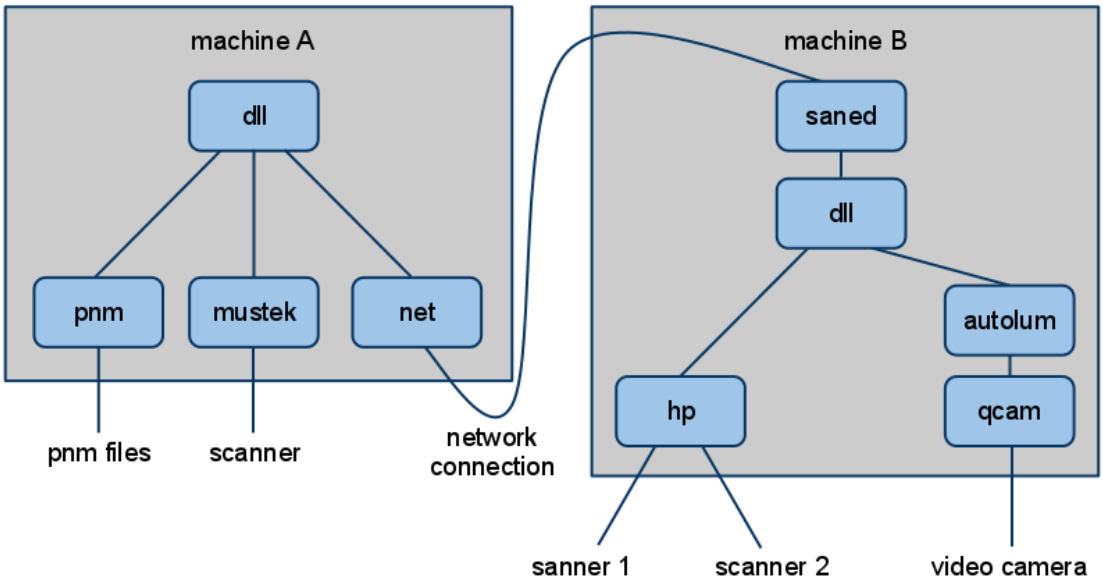



図 2　SANE の構造

なお、SANE を利用するユーザは scanner グループに所属させる必要があります。

## 10.2 Code Flow

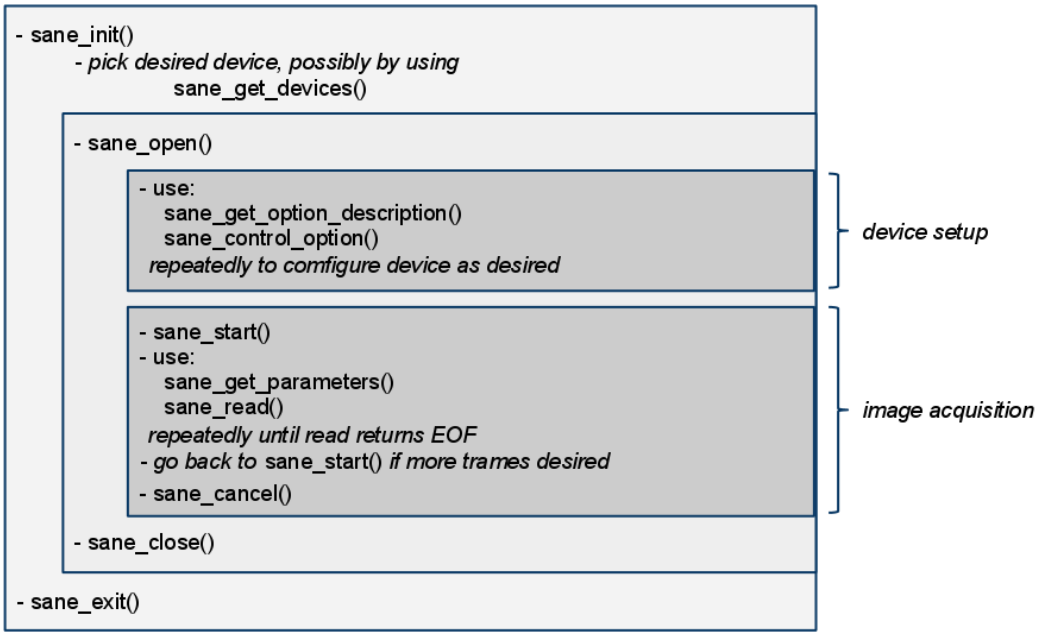



図 3 Code Flow

## 10.3 スキャンするためのコード

とりあえずスキャンするだけのコードは次のようになります。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sane/sane.h>


#define MAXLEN 32768

int main()
{
  SANE_Status stat;
  SANE_Handle hndl;
  SANE_Byte buf[MAXLEN];
  SANE_Int len = MAXLEN;
  FILE *fp;

  stat = sane_init(NULL, NULL);
  printf("init: %d\n", stat);
  stat = sane_open("fujitsu:libusb:008:002", &hndl);
  printf("open: %d\n", stat);

  fp = fopen("foo.bin", "wb");
  stat = sane_start(hndl);
  printf("start: %d\n", stat);
  while (len == MAXLEN) {
    stat = sane_read(hndl, buf, MAXLEN, &len);
    printf("read: %d\n", stat);
    printf("read_len: %d\n", len);
    fwrite(buf, len, 1, fp);
  }
  fclose(fp);

  sane_close(hndl);
  printf("close: %d\n", stat);
  sane_exit();
  printf("exit: %d\n", stat);
}
```

このコードで保存される foo.bin は、ScanSnap S1500 の場合、ヘッダ無しの pbm ファイルとして保存されました。スキャナによっては pgm で保存されるものもあり、デフォルトで保存される画像のモードは特に決まっていないようです。

## 10.4 カラーのスキャン

オプションの設定にはいくつか注意点があるようです。

- sane_get_option_descriptor() でオプションの説明文やフォーマットの取得が可能。
- sane_control_option(hndl, 0, SANE_ACTION_GET_VALUE, &num, &info) でオプションの総数を取得できる。
- sane_control_option() で Action に SANE_ACTION_GET_VALUE を指定することで現在設定されているオプションを読み出すことが可能。
- sane_control_option() で Action に SANE_ACTION_SET_VALUE を指定することでオプションの設定が可能。
- sane_control_option() を呼ぶ際には事前に sane_get_option_descriptor() を呼び出しておく必要がある。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sane/sane.h>

#define MAXLEN 32768

int main()
{
  SANE_Status stat;
  SANE_Handle hndl;
  const SANE_Option_Descriptor *opt;
  SANE_Int info = 0;
  SANE_Byte buf[MAXLEN];
  SANE_Int len = MAXLEN;
  SANE_Parameters param;
  FILE *fp;

  stat = sane_init(NULL, NULL);
  printf("stat: %d\n", stat);
  stat = sane_open("fujitsu:libusb:008:004", &hndl);
  printf("open: %d\n", stat);

  int x=300, y=300;
  opt = sane_get_option_descriptor(hndl, 4);
  printf("get_option_descriptor: %03d: %d\n", 0, opt->size);
  stat = sane_control_option(hndl, 4, SANE_ACTION_SET_VALUE, &x, &info);
  printf("control_option: %d\n", stat);
  printf("info: %d\n", info);

  opt = sane_get_option_descriptor(hndl, 5);
  printf("get_option_descriptor: %03d: %d\n", 0, opt->size);
  stat = sane_control_option(hndl, 5, SANE_ACTION_SET_VALUE, &y, &info);
  printf("control_option: %d\n", stat);
  printf("info: %d\n", info);

  strcpy(buf, "Color");
  opt = sane_get_option_descriptor(hndl, 3);
  printf("get_option_descriptor: %03d: %d\n", 0, opt->size);
  stat = sane_control_option(hndl, 3, SANE_ACTION_SET_VALUE, buf, &info);
  printf("control_option: %d\n", stat);
  printf("info: %d\n", info);

  stat = sane_start(hndl);
  printf("%d\n", stat);
  fp = fopen("foo.ppm", "wb");

  stat = sane_get_parameters(hndl, &param);
  printf("get_parameters: %d\n", stat);
  sprintf(buf, "P6\x0a# SANE data follows\x0a%d %d\x0a%d\x0a",
    param.pixels_per_line, param.lines, 255);
  fwrite(buf, strlen(buf), 1, fp);


  while (len == MAXLEN) {
    stat = sane_read(hndl, buf, MAXLEN, &len);
    printf("read: %d\n", stat);
    printf("read_len: %d\n", len);
    fwrite(buf, len, 1, fp);
  }
  fclose(fp);

  sane_close(hndl);
  printf("close: %d\n", stat);
  sane_exit();
  printf("exit: %d\n", stat);
}
```


図 4　スキャン結果

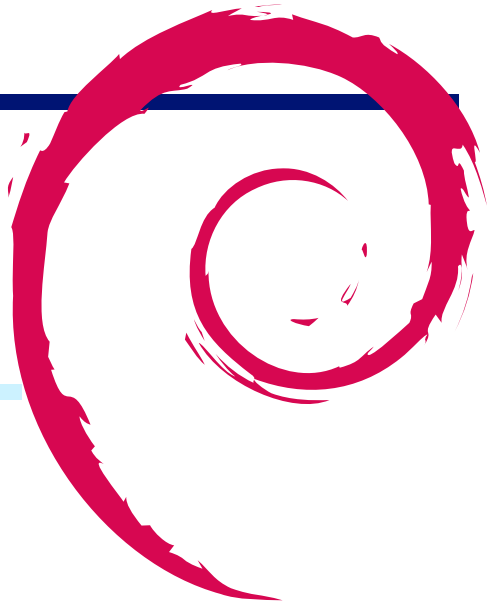
モノクロではわかりづらいと思いますが、ScanSnap S1500 でスキャンした結果、R と B が入れ替わってスキャンされました。

### 10.5　まとめ

- やや不安定 (特に権限周り)
  - scanimage -L でたまに segmentation fault する
- スキャナによってデフォルトの動作がまちまち
  - ScanSnap は.pbm でスキャン
  - CanoScan は.pgm でスキャン
- スキャナによって RGB の順番すらまちまち
  - でもこれは ScanSnap が悪いのかも
  - ScanSnap は TWAIN 未対応だし
- ドキュメント化されていない罠の存在
- スキャナのボタンをイベントとして使えないっぽい

## 参考文献

[1] SANE - Scanner Access Now Easy
`http://www.sane-project.org/`
[2] SANE-tutorial-JP / 著者:David Mosberger / 翻訳:川岸 良治
`http://archive.linux.or.jp/JF/JFdocs/SANE-tutorial-JP.html`
[3] SANE Standard Version 2.0 proposal 0.08 - rauch/beck
`http://www.sane-project.org/sane2/0.08/`



# 11 Proxmox VE の紹介

森山 京平

## 11.1 自己紹介・経歴・概要

森山 京平（23）

NAIST（奈良の森の中の大学院）
Ubuntu User
Debian は、サーバ用途でしか触ったことがない。
最近は、もっぱら、仮想化技術に興味をもち、KVM を触って自己満足の世界にひたっている。某プロジェクト[*22]のネットワークエンジニア兼サーバエンジニアに任命され、四苦八苦しながらネットワーク・サーバの運用と管理を行っている。今年 AI3 Project では、10 年ほど運用してきたサーバ（物理マシン）のリプレイスを行うようになった。そこで、近年話題になっている、仮想化技術を用いて、サーバの台数を減らし、可用性を増すことを試みた。機器の設置や、dom0（仮想化のベースとなるサーバ）の設定は、ほぼ終了。仮想マシンで実行するサービスのテストも無事に終わった。今後、次々に移行していく予定である。KVM を用いて完全仮想化を行うという要求があったので、どうしようと迷っていた所、Debian ベースで開発が進んでいる、ProxmoxVE という OSS に出会い、それを採用させて頂いた。今回は、その ProxmoxVE について簡単にご説明させていただこうと思う。

## 11.2 Proxmoc VE とは

ProxmoxVE とは、Proxmox Server Solutions GmbH によって開発されている、Debian ベースでかつ web、cli から仮想マシンを操作できる KVM&OpenVZ マネージメントツールである。perl と shell を使い qemu や kvm コマンドをうまく組み合わせて web 上からの操作を可能にしている。競合しているツールとして同じく web から操作できる KVM のマネージメントツールの karesansui がある。こちらは国内産であり、100%PurePython で書かれている。優位性云々は、karesansui をあまりあたったことがないのでわからない。ProxmoxVE のスタートは、メールゲートウェイの仮想化を行い、セキュリティを高める目的の仮想化環境である proxmox という OSS からスタートしている。proxmox から得た知見を proxmox VE にも適用している。現在の最新バージョンは、1.7 であり、2.0 のリリースが予定されている。ダウンロードは `http://www.proxmox.com/` からできる。

## 11.3 インストール

さて、インストールの方法を説明したい。そのまえに、dom0 マシンに対しての要求がある。KVM による完全仮想化を実現するために、お持ちのマシンの CPU が Intel-VT もしくは AMD-V に対応しておかなければならない。

[*22] AI3 `http://www.ai3.net`

もしも、OpenVZ のみを使うのであれば、それらは、必須ではない。しかしながら、パフォーマンスとしては、完全仮想化であるほうが、優位であると参考までに覚えておいていただきたい。

### 11.3.1 ダウンロード

ProxmoxVE のイメージをダウンロードする。`http://www.proxmox.com/` ここから donwnload リンクをたどって行くと iso をダウンロードできる。ダウンロードした iso を CD なり DVD に焼く。

### 11.3.2 ProxmoVE のインストール

インストール手順は、至ってシンプル。マシンの性能にもよるが、15 分程度で仮想環境が手に入る。

## 11.4 web インターフェースからの各種操作

ProxmoxVE では、基本的に操作を web インターフェース上から行うように設計されている。ログイン画面からログインを行うことで、Web 管理コンソールに入ることができる。ログインを終えると、ホストマシンのカレントステータスが表示される。[*23]

### 11.4.1 VM マネージャ

VM マネージャの欄には、仮想マシン、テンプレート、ISO イメージという項目が存在する。仮想マシンでは、仮想マシンの状態、作成、マイグレートの操作を行うことができ、テンプレートでは、OpenVZ のテンプレートに関する操作ができる。また、ISO イメージにおいては、OS のインストールディスクのイメージをアップロードできる。テンプレートは OpenVZ から、ISO イメージは KVM から用いる。

**■リスト**　ここから、仮想マシンのそれぞれのメンテナンスタスク、カレントステータスを確認することができる。
**仮想マシンの設定**　実行中の仮想マシンの欄をクリックすることで仮想マシンの設定画面に遷移する。仮想マシンの操作や設定はここから行う。

- 仮想マシンの設定 [仮想マシンのディスク領域、メモリ、スワップなどの設定]
- 仮想マシン削除 [仮想マシンの削除をおこなう（復元不可）]
- 仮想マシン起動 [仮想マシンの起動をおこなう]
- 仮想マシン再起動 [仮想マシンの強制再起動をおこなう]
- 仮想マシンシャットダウン [仮想マシンの強制シャットダウンを行う]
- 仮想マシン停止（サスペンド？）[仮想マシンのサスペンドをおこなう（未実装？）]
- VNC コンソールへのアクセス [JAVA アプレット-VNC でコンソールへアクセスする]

**■作成**　仮想マシンの作成をおこなう。KVM に対応していれば、KVM の項目が出てくる、そうでなければ、タイプの欄に OpenVZ の項目しか出てこない。それぞれのパラメータの設定おこない、”作成する ”をクリックすることで、仮想マシンの作成をおこなえる。

**■マイグレート**　ProxmoxVE では、CLI から pveca コマンドを用いることでクラスタリングを行うことができる。クラスタリングを行った物理マシン間で仮想マシンのマイグレートが可能である。今回は、端折らせていただくが、物理マシンのメンテナンスの際に役立つ機能である。

仮想マシンの項目から、ほとんどの管理を行うことができる、その点が ProxmoxVE の素晴らしいとこである。詳細な管理や設定を行いたい場合には、CLI（ssh によるアクセス）でおこなう。CLI のベースは debian であるので、かなり使いやすい印象をうけた。

[*23] CPU、メモリ、カーネルのバージョンなど

### 11.4.2 設定

設定には、システム、ストレージ、バックアップの項目が存在する。

**■システム**　ネットワーク、DNS、時刻、管理、オプションの項目があり、ネットワークでは、おもにネットワークインターフェースの設定を行う。DNS では、DNS の設定を行い、サード DNS サーバまで設定を行える。時刻は、時間帯の設定（JST など）の設定を行う。NTP サーバの設定ができるかと思ったらそうではなく、もし特定の NTP サーバの設定を行いたい場合は、CLI からの設定が必要であるように思う（未確認）。管理の項目からは、管理者の E-mail、パスワードの設定が行える。オプションからは、言語とキーボード、HTTP プロキシの設定を行える。
**ストレージ**　ストレージの設定を行う。ストレージの追加削除を行う。iSCSI、NFS、LVM グループ、ディレクトリの参照をおこなうことが可能である。
**バックアップ**　ストレージの項目から、バックアップストレージの指定があれば、ここから、バックアップの設定をおこなうことができる。定期的な仮想マシンのバックアップを行うことができる。

**■管理**　管理の項目には、サーバ、ログ、クラスタがある。
**サーバ**　サーバのサービスの起動、停止、再起動を行うことができる。SSL 証明書もここからダウンロードできる。またサーバそのものの再起動、シャットダウンも行える。
**ログ**　タスクのログ、syslog の確認を行えることは、認識しており、仮想化初心者ユーザの敷居を低くする有用な OSS であるかと思う。もし、詳細をしりたい方がいらっしゃれば、ProxmoxVE の Wiki[1] を参照していただけたらと思う。

# 参考文献

[1] ProxmoxVE Wiki, http://pve.proxmox.com/wiki/Main_Page

# 12 Apache2 のモジュールをつくってみた

上川 純一

Apache httpd というウェブサーバがあります。おそらく定番と言われるウェブサーバで、多くの人が利用していると思います。ウェブサーバとして多くの機能を提供している Apache ですが、モジュールという仕組みで機能拡張可能です。今回は Debian 上で Apache モジュールを作る方法をさぐってみることにします。

## 12.1 Apache モジュールをつくりたいとき

Apache モジュールが作りたいときはどういうときでしょうか。作りたいと思ったときがつくりどき。cgi で fork する手間が惜しい、インタプリタ言語でインタプリタが走っている時間がもったいない、GC のある言語でときどき反応が悪くなるのが気にくわない。いろんな理由があるとおもいますが、C 言語でがりがりとウェブリクエストを処理したいとかそういう要望があるのではないでしょうか。

Apache モジュールでできることはどういうことでしょうか。Apache の受け取る入力 (HTTP Request) から出力 (HTTP Response) のほぼ任意の場所にフックとして割り込むことができて、入出力のデータを加工することが可能です。

## 12.2 Debian の Apache パッケージの構成

ところで、httpd と Debian apache2 はどれくらい違うかご存知でしょうか。
Debian パッケージの README.Debian を見てみましょう。[1]
ざっと眺めただけでも以下の点で特徴があります。

- apache2 コマンドが環境変数に依存している。
- apache2ctl / /etc/init.d/apache2 を利用しないと起動とかできない。
- 設定ファイルの配置が全然違う。
- a2enmod / a2dismod コマンドが追加されている。
- apxs コマンドが apxs2 という名前になっている。

## 12.3 Apache モジュールの作成

普通の Apache モジュールの作成の流れをみてみましょう。

### 12.3.1 パッケージの準備

とりあえず開発環境をインストールしましょう。

```
# apt-get install apache2-threaded-dev
```

### 12.3.2 テンプレート生成

apxs2 コマンドを利用してテンプレートを作成します。モジュール名でディレクトリが作成されます。

```
$ apxs2 -g -n dancerqps
$ cd dancerqps
$ ls
Makefile  mod_dancerqps.c  modules.mk
```

### 12.3.3 ソースコード編集

ソースコードを編集しましょう。この場合、自動で生成された mod_dancerqps.c がメインのモジュールソースコードです。

まず一番下に一番重要な構造が定義されています。ここで重要なのは dancerqps_register_hooks を登録しているところでしょうか。

```
module AP_MODULE_DECLARE_DATA dancerqps_module = {
    STANDARD20_MODULE_STUFF,
    NULL,                  /* create per-dir    config structures */
    NULL,                  /* merge  per-dir    config structures */
    NULL,                  /* create per-server config structures */
    NULL,                  /* merge  per-server config structures */
    NULL,                  /* table of config file commands       */
    dancerqps_register_hooks  /* register hooks                      */
};
```

dancerqps_register_hooks で実際にフックの登録を行います。ここでは、アウトプットフィルタを登録しています。

```
static void dancerqps_register_hooks(apr_pool_t *p) {
  ap_register_output_filter(dancerqps_name, dancerqps_filter,
                            NULL, AP_FTYPE_RESOURCE);
}
```

そして、実際に各リクエストのアウトプットに対して実効するフィルタのコードを書きます。

```
static apr_status_t dancerqps_filter(ap_filter_t *f, apr_bucket_brigade *bb) {
  /* ここでなにか処理をする */
  ap_log_rerror(APLOG_MARK, APLOG_ERR, 0, f->r,
                "dancerqps processing was here!");
  return ap_pass_brigade(f->next, bb);
}
```

### 12.3.4 モジュールのファイルインストール

モジュールをビルドしてインストールする一連の手順をしてくれるコマンドがあります。

```
$ sudo apxs2 -c -i mod_dancerqps.c
```

### 12.3.5 モジュールを有効にする

apxs2 コマンドでモジュールを有効にする方法は Debian 独自の a2enmod などのコマンドの存在を無視しているようなので a2enmod などと整合性のとれるような方法をとるのがよいでしょう。で、それでは Debian 的な方法とはなんでしょうか。

#### Debian way 1

めんどくさい方法だけどシステム運用者としては便利かもしれない方法。a2enmod とかができます。

/etc/apache2/mods-available に hoge.conf, hoge.load ファイルを作成します。hoge.conf は設定が不要なら不要ですが、httpd.conf に書くような内容を記述します。hoge.load には LoadModule 行を記述します。

この方法でファイルを作成しておけば、/etc/apache2/mods-enabled ディレクトリにシンボリックリンクが作成さ

れます。

**Debian way 2**

デフォルトでは空のファイルだけど、/etc/apache2/httpd.conf に書き込むとその設定が有効になります。

```
LoadModule test_module /usr/lib/apache2/modules/mod_test.so
<Location "/hoge/">
   SetHandler test
</Location>
```

**Debian 的な方法から乖離しているっぽい方法**

apache をコマンドラインから設定ファイルを指定して起動することができます。Debian の Apache は環境変数を設定しないと直接は起動できなくなっているので、面倒な手続きが必要になります。

まず、適当にテスト用の http.conf をでっち上げます。

```
Listen 8080

LockFile /home/test/tmp/apache.1.lock
PidFile /home/test/tmp/apache.1.pid

# log configuration.
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "/home/test/log/access_log" common
ErrorLog "/home/test/log/error_log"

# Order, Allow.
LoadModule authz_host_module /usr/lib/apache2/modules/mod_authz_host.so
# map from / -> /index.html
LoadModule dir_module /usr/lib/apache2/modules/mod_dir.so
DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
# .html -> content-type: text/html
LoadModule mime_module /usr/lib/apache2/modules/mod_mime.so
TypesConfig /etc/mime.types

# Document root
DocumentRoot "/home/test/hoge"
<Directory "/home/test/hoge">
    Options Indexes FollowSymLinks

    AllowOverride None

    Order allow,deny
    Allow from all

</Directory>

# Load my custom filter.
LoadModule dancerqps_module /usr/lib/apache2/modules/mod_dancerqps.so
SetOutputFilter DANCERQPS
```

適当な設定ファイルを指定して Apache を起動してみます。

```
APACHE_RUN_USER=dancer   \
 APACHE_RUN_GROUP=dancer \
 /usr/sbin/apache2 -f $(readlink -f ./httpd.conf) -k restart
```

### 12.3.6 負荷テスト

ab (apachebench) ツールが標準で入っているのでそれで計測します。適当に確認しましょう。

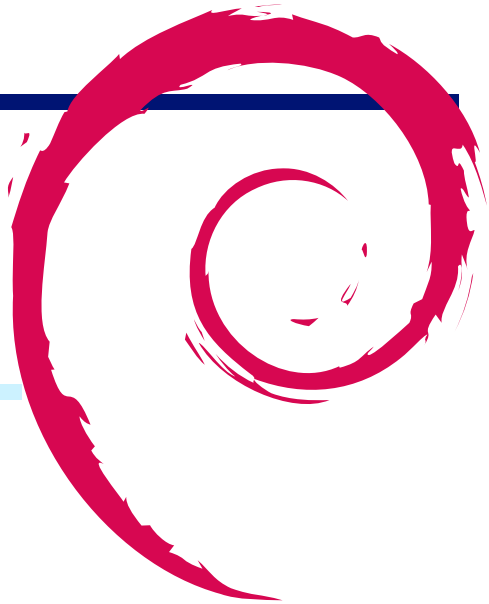
```
$ /usr/sbin/ab -c 100 -n 100 http://localhost:8080/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:        Apache/2.2.9
Server Hostname:        localhost
Server Port:            8080

Document Path:          /
Document Length:        44 bytes

Concurrency Level:      100
Time taken for tests:   0.056 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      29600 bytes
HTML transferred:       4400 bytes
Requests per second:    1796.17 [#/sec] (mean)
Time per request:       55.674 [ms] (mean)
Time per request:       0.557 [ms] (mean, across all concurrent requests)
Transfer rate:          519.21 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        7    9   0.5      9      10
Processing:     9   26   8.9     27      40
Waiting:        6   26   9.3     27      40
Total:         16   36   8.9     37      49

Percentage of the requests served within a certain time (ms)
  50%     37
  66%     41
  75%     43
  80%     45
  90%     47
  95%     48
  98%     49
  99%     49
 100%     49 (longest request)
```

### 12.3.7 まとめ

はじめて Apache モジュールを作成して、インストールして動かすところまでやってみました。

# 参考文献

[1] Debian の Apache2 パッケージの README.Debian ファイ, `/usr/share/doc/apache2.2-common/README.Debian.gz`



# 13　CAcert の準備に何が必要か

山田 泰資

## 13.1　CAcertって何？

CAcert プロジェクトとは誰もが低コストに証明書を持てるようにすることで、安全な環境・通信を広く実現しようというプロジェクトです。2002 年に発足し、2003 年には非営利法人の CAcert Inc. がオーストラリアで設立され、以来７万人のメンバーに対して１８万件の証明書（クライアント認証用、サーバ認証用、S/MIME 署名用、コード署名用、etc）を発行してきています。

通常、個人や組織内で自己発行した証明書は「オレオレ証明書」などと呼ばれてあくまで閉じた世界での利用になります。しかし、このプロジェクトでは CAcert がルート証明機関として、そのルート証明書が各種 OS やブラウザに組み込まれ、VeriSign 社などによる商用サービスに近い水準で世界的に信頼される状態を目指しています。このため、コミュニティベースのプロジェクトとは言っても、厳格かつ高い水準の認証基準や運営ポリシが定められています[*24]。

この高い水準の認証局運営を低コストで行うため、CAcert は次の２つを運営の柱としています：

1.「信頼関係の構築・維持」と「証明書発行」の分離
2.「信頼度・経験値のポイント化」と、「ポイントによる発行制限」

まずは前者から説明します。通常、証明局の運営では申請者の身元確認などを法的文書に基づいて行い、その上で証明書を発行します。この前者が高コストで、

1. 提出された申請内容が真正かどうかの確認
2. 後日の紛争に備えての関係書類の（法的に有効な）紙での保管

などで事務コストがかかります。一方、証明書の発行は秘密鍵の安全な保管などセキュリティ面のコストは必要なものの、比較的安価です。そこで、CAcert では

1. 本人確認と関係書類の保管はコミュニティベースで各自が実施
2. それに基づいた証明書の発行は CAcert がルート CA となって実施

という認証と発行の分離を行っています。コミュニティ側で行った認証結果をポイントという形で CAcert に登録し、CAcert はポイントに応じて各種の証明書を発行する訳です。

そしてこのポイント制度も CAcert の特色の１つです。CAcert Inc. 自身は本人確認を行っていないので、認証をする側もされる側もコミュニティメンバーという事になります。この仕組みの中で十分な認証を行うため、

1. 認証「される」人が申請できる証明書の形式・機能はポイントに応じる

[*24] 実は途中沈滞していたものの、ここ数年で急速に整備されてきました

2. 認証「できる」ようになるのは 100pt に到達してからで、更に試験合格が必要
3. 認証時に付与できるポイントも 100pt の成り立てでは少なく、経験に応じて増える

と 1 回の認証ではなく多人数間で認証を行う（Web of Trust と呼びます）方式を導入しています。

## 13.2 CAcert でできること - 証明書の作成、認証者としての活動

さて、まずは一番利用するであろう証明書の作成について紹介します。実際のところ、認証を受け、利用するのみのメンバとしてであれば、これが CAcert の唯一の機能です。

これは `https://cacert.org/` に行き、メンバ登録を行うことで作成できます。ただし、認証可能メンバ（CAcert Asurer）による認証・ポイント付与を受けていない場合、作成できるのは最低限の機能を持つ証明書のみとなります。以下が保有ポイントによる作成可能な証明書などメンバ権限の一覧です：

| 保有ポイント | ステータス | できること |
|---:|---|---|
| $\geq 0$ | Member | 実名抜き（メールアドレスのみ入る）のサーバ・クライアント用証明書の発行（半年有効） |
| $\geq 1$ | Member | 上と同様だが、署名元のルート証明書が Member Root 証明書に格上げされる |
| $\geq 50$ | Assured Member | 実名入りのサーバ・クライアント用証明書の発行（2 年有効） |
| $\geq 100$ | Prospective Assurer | 上に加え、コードサイニング証明書の発行（1 年有効） |
| $\geq 100$ + 試験通過 | Assurer | 上に加え、オンライン試験に合格した場合、認証可能メンバとして他者を認証し、最大 10 ポイントまで付与できる。また、自身は以後の認証の度に 0-2EP を受領する |
| $\geq 100$ + 10EP + 試験通過 | Assurer | 上と同様だが、付与可能ポイント上限が 15 ポイントとなる |
| $\geq 100$ + 20EP + 試験通過 | Assurer | 上と同様だが、付与可能ポイント上限が 20 ポイントとなる |
| $\geq 100$ + 30EP + 試験通過 | Assurer | 上と同様だが、付与可能ポイント上限が 25 ポイントとなる |
| $\geq 100$ + 40EP + 試験通過 | Assurer | 上と同様だが、付与可能ポイント上限が 30 ポイントとなる |
| $\geq 100$ + 50EP + 試験通過 | Assurer | 上と同様だが、付与可能ポイント上限が 35 ポイントとなる。また、コミュニティ内において Experienced Assurer / Senior Assurer としての役割に進む入口に達したと見なされる |

表 1　ポイントによるアカウントステータスと権限の一覧

上の通り、証明書の発行を受ける利用者としては 100pt が上限となります。これ以上はどれだけ認証を受けても Assurance Point は増えません（システム的に増えたように見えても無効）。しかし、一方で試験を受け、Assurer として他者を認証すると、その度に最大 2EP が与えられます。表中の 10EP/20EP/... とある部分がそれで、これを Experience Point と呼びます。以後はこちらを蓄積することで、更に上位のステータスを得る形になります。

ちなみに試験は選択式（25 問中 20 問正解で合格）で、実はそんなに難しくありません。100pt に達する前から受けることができ、合格するまで何度やっても構いません（トレーニング的な意味で定期再受験が推奨されている）。合格しておけば 100pt になった瞬間から Assurer になれるので、皆さん受けておいてはいかがでしょうか？[*25]以下が試験画面です：

[*25] https://cats.cacert.org/ から受けられますが、クライアント証明書が必要です

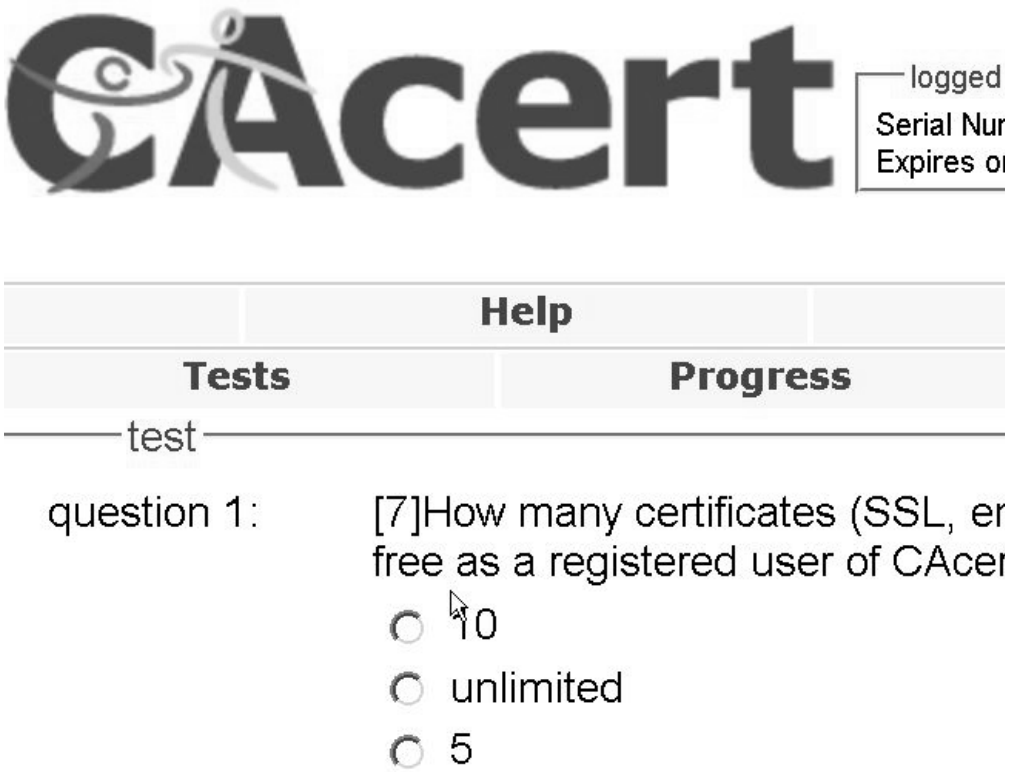



図 5　オンライン試験（選択式）の画面（全２５問）

最後に、さらに上位の Assurer として Experienced / Senior Assurer というものがあります。これらはイベント内での認証会の開催では Experienced Assurer が最低３名必要であったり、コミュニティ内での調停やトレーナーには Senior Assurer が指名されるなど、より幅広い関わりを持つ上で必須の重要な立場となります。Experienced Assurer は 50EP 以上（つまり、100pt 分の認証を受けた後、最低 25 人以上を認証した状態）が条件ですが、SeniorAssurer の認定には

1. Experienced Assurer であること
2. さらに、co-auditor[*26]の試問を受け、合格すること
3. さらに、各国[*27]で開催される ATE(Assurer Training Event) に参加し、トレーニングを受けていること
4. さらに、宣誓（CARS - CAcert Assurer Reliable Statement）を行うこと

という高いハードルがあります。組織としての CAcert が一通り機能するには Senior Assurer の存在がキーとなるため、日本で CAcert が完全に立ち上がったと言えるのは、この Senior Assurer が生まれた時になるでしょう。

## 13.3　認証を（して｜されて）みよう

最後までのハードルは高いのですが、まずは第一歩からということで、１対１で認証を（する｜される）時の手順を紹介します。

まずは双方で以下を用意します（標準的な手順の場合）：

[*26] 認証ポリシの策定や監査を統括する Assurance Office の委託を受けて活動している経験豊富な Experienced Assurer
[*27] 現時点ではほぼ欧州でのみ開催 …

認証する側（Assurer）が用意するもの

1. 相手に名乗るための名刺（ただし、口頭でもよい[a]）
2. 白紙・未署名の CAP(CAcert Assurance Program) Form（相手が忘れた時用）[b]
3. 紫外灯などの、証明書の偽造検出のための機材[c]
4. 法的責任や義務を説明するための CCA(CAcert Community Agreement)（できれば渡せるよう紙で）

[a] 不特定多数のイベントでは、0点付与/調停となる可能性もあるので、口頭に留めるのが正解？
[b] https://secure.cacert.org/cap.php
[c] 日本政府発行のパスポートなどをこれで確認する

認証される側（Member）が用意するもの

1. CAcert に登録したメールアドレス（複数登録の場合はプライマリのもの）
2. 身元証明書。2つ以上が望ましく、最低1つは政府発行かつ誕生日入りである必要がある。また、最低1つは写真入りである必要がある
3. 未署名の CAP Form（署名以外は事前記入・印刷 OK）[a]

[a] https://secure.cacert.org/cap.php - なお、ログインすれば事前記入済みのものをメニューから選択・印刷もできる

中央集権型の CAcert では、証明書の階層構造の安全を守るためと、認証手続きが実際に法的責任が生じる契約であることから、GPG キーサイン会よりも高い基準での偽造検出や確認を要請しています。上の紫外灯での真正性確認と、未署名書類へ目の前で署名することの確認義務などがそれにあたります。

さて、続きです。お互いの準備が整ったら、以下の手順で説明・確認・署名を行います。

## !! 手順は説明・確認・署名の順番です !!

大事なことなので二回書きました。

1. まず、Assurer から CCA の概要、特に法的に発生する義務と責任について明確に説明して下さい。骨子としては以下の3点ですが、Assurer は下記の背景や CAcert の意義を合わせて説明する義務があります。
   (a) 係争となった場合、各国内の裁判所に提訴する権利は放棄し、CAcert が Assurer の中から任命する調停者の裁定に従う義務があること
   (b) 誤用や誤認証を行い損害が発生した場合、補償責任があること。ただし、最高でも 1000EUR（2010/12 の相場で約 10 万円強）となる
   (c) 証明書の運用・保証範囲やプライバシ保護などに関して、公式文書群（COD - CAcert Official Document）の規定に従うこと
2. Member が上記の義務・責任に同意するなら次に進みます
3. 次に、Assurer は Member に CAP Form への記入と署名を求め、併せて証明書を受け取って下さい。なお、署名と提出は目の前で行われる必要があります（事前署名は不可）
4. Assurer は内容を確認して下さい。確認ポイントは以下の通りです：
   (a) 責任・義務を規定する CCA に同意する旨が明記・チェックされていること
   (b) 氏名・誕生日が一致し、また、写真からも当人であることが確認できること
   (c) メールアドレスが cacert.org に確かに登録されていること
   (d) 書類への署名が証明書に記載の署名（あれば）と同じであること
   (e) 証明書が失効しておらず、真正と見えること（紫外灯チェック、写真の上にハンコがかかっているか、等）
   (f) 相手が氏名・誕生日など、証明書記載事項の質問に対して正答すること
5. 不備があるなどで確信が持てない場合、以下の対応を行います：
   (a) 仕組み上、付与ポイントが少なくなったり、調停が必要となる可能性があることを伝えて下さい
   (b) 表記違いがある場合、証明書の記載内容が優先されます。CAcert 登録内容の修正について当人と調整し、

dispute（調停依頼[*28]）を Assurer より提起して下さい。ポイント付与は修正後に行います。

（c）複数の証明書で氏名表記に揺れがあるものの同一人物と判断できる場合は、CAP Form にすべて列挙の上で認証して下さい。

（d）芸名であっても、それが規定の証明書で証明可能であれば、認証して構いません

（e）称号（PhD 等）は使わないで下さい。証明書の記載内容以上のものを認証してはなりません

6. 以上で両者の対面での手続きは終了です。ここで別れて下さい。
7. 最後に、CAP Form の裏面にメモを残します。困った点、書類が古かった、妙に急かされた等の引っかかっる点があったらそれを、また、面白い用法や、相手が運営参加希望者だった場合はその得意分野などです。後日の連絡や調停時に使います。

以上が当日のやりとりとなります･･･が、Assurer の仕事はまだ終わっていません！認証パーティが終わったら、Assurer の手元には署名入り CAP Form が沢山残るはずです。これらを元に、以下の事後処理を行います：

1. CAP Form に対して認証者として署名する
2. cacert.org より、CAP Form 記載のメールアドレスに対してポイントを付与する。付与するポイント数は、裏面メモなどを見ての確信度に応じて 0 点から最高点まで調整
3. 処理を終えた CAP Form を、7 年間保管できる場所にしまう ← ← ← 説明後述

さて、ここで「0 点を付与」と「手続き自体をしない」は意味が異なります。「0 点」は「相手を信頼できなかった」ではなく

書類が未知すぎて、点数を付けたくてもできませんでしたー！mOm

という意味になります。0 点を貰った人は、誤解しないようにしましょう。一方、手続きをしないというのは

いや、この（人｜書類）はおかしいだろ JK

という、どう見ても本物ではなかったなどの疑念が大きい場合です。この場合は裏面メモを取っておき、ポイント付与ではなく調停依頼を support@cacert.org に要請することも考えましょう。

## 13.4 7 年間の保管義務って？

おそらく GPG キーサインとの最大の違いは、この点になります。
単に認証基準が高いというだけでなく、CAcert は

デジタル空間の証明階層を実世界の法的文書でバックアップする

という意義を持っています。このため、紙ベースで署名・作成された CAP Form は Assurer に保管義務があるだけでなく、その電子化も禁止されています[*29]。

もし自分では保管し続けることが困難になった場合は、`support@cacert.org` に調停依頼を要請し、指示を仰いでください。

## 13.5 公式 CAcert イベントの開催方法

1 対 1 での認証と違い、正式な認証会の開催となると多くのものが必要となります。基本要件は先の 1 対 1 と同じなのですが、公式 CAcert 認定イベントとして開催する場合、各種の規定があります。要件としてあるのか、単に「円滑な開催、信頼性の高い認証」のために推奨されているものか判然としない部分もありますが、マクロレベルで

[*28] この単語だけだと紛争・係争ですが、このような修正依頼なども含まれるため、調停を用語としました
[*29] CAP Form 毎に署名を付加して各自で保管とか考えましたが、再びデジタル空間に基盤が戻ってしまうので不可のようです…

の開催の流れは以下のようになります。

1. Event Office へ開催提案を送る (`http://wiki.cacert.org/Officers`)
2. CAcert との窓口になる主催者（EventOffier）を協議して決める
3. 開催日・場所を決め、スポンサーを付ける場合は予算案を詰める[*30]
4. 計画策定。出展内容や必要資材・搬入の調整、更には当日に Assurer として参加を求める場合は事前の Assurer 人員の教育や宿泊の手配を行う
5. 事前準備。ブースの設営、配布物の用意、買い出しなど
6. イベント実施
7. 後片付け。撤収の他、再利用可能なバナーなどの資材は回収・整理する
8. 完了報告。EventOfficer よりレポートを Event Office に送って終了

これはあくまで CAcert 公式イベントとする場合なので、そうでない場合は 1 対 1 の認証を基本として、この章の内容は準備内容の参考に使って下さい。

### 13.5.1 準備チェックリスト

ここでは推奨されている確保すべき人員・資材の一覧を列挙しておきます。これを満たさなくては正規の認証手順を経たと認定されないということはなさそうですが、後から不備・無効となるのを避けるため、慣れるまでは心配な点は CAcert 側の Event Office に相談するとよいでしょう。

| 役割 | 人数 | 注記 |
|---:|---|---|
| Event Organiser | 1 | 主催者兼進行役。一番の経験者がよい。Event Office と相談して決める |
| Experienced Assurer | 3+ | 認証担当。他ブースを訪問して証明して回る巡回証明役もいるとよい |
| Assurer | 2+ | EA の補助として CAP Form の事前確認を行い、EA をフル回転させる |
| ガイド役 | 2+ | 行列の整理や質疑、派生して起こる問題の対処など |

表 2　CAcert イベント開催に必要な人員

| 資材 | 個数 | 注記 |
|---:|---|---|
| 紫外灯 | 1+ | 並行処理する認証手続きの個数だけ用意 |
| 机 | 1+ | 並行処理する認証手続きの個数だけ用意 |
| 椅子 | 1+ | 並行処理する認証手続きの個数だけ用意 |
| ネット回線 | 0+ | 事前印刷が十分できていれば必須ではないが推奨 |
| PC | 1+ | 資料閲覧、ポイント付与作業、新規登録用（Knoppix 推奨） |
| プリンタ | 1 | CAP Form 不足や配布資料の現場量産用 |
| ペン | 10+ | 参加人数や進行方法に基づいて本数は考える |
| 募金箱 | 1 | 経費のカバー用（事前告知があれば、正式に徴収してもよい |

表 3　CAcert イベント開催に必要な資材

[*30] 本気で公式イベント化する場合、どうも色々支援がある？

| 資料 | 個数 | 注記 |
|---:|---|---|
| Assurer 連絡名簿 | 1 | 認証に関する疑義対応や後日のフォローアップ用 |
| CAP Form（1p, 記入済） | 30+ | 未署名。開催日、場所、Assurer 名まで記入 |
| CAP Form（1p, 白紙） | 20+ | 新規 Assurer に手伝ってもらう用 |
| CCA (4p) | 50+ | Assurance に伴う契約義務・責任の解説用 |
| CCA (巨大張り出し用) | 1 | 壁に張り出して使う |
| Assurance Policy (8p) | 1+ | 質疑回答、参照用 |
| Assurance Handbook (29p) | 1+ | 質疑回答、参照用 |
| Root Distribution License (1p) | 1 | 質疑回答、参照用 |
| Dispute Resolution Policy (6p) | 1 | 質疑回答、参照用 |
| Practice On Names (4p) | 1+ | 各 Assurer に 1 部。表記揺れへの対応ガイド |
| Practice On ID Checking (4p) | 1+ | 各 Assurer に 1 部。証明書内容の検査ガイド |
| PoJAM (4p) | 1 | 未成年への認証、未成年による認証に関する規定 |
| CPS(28p) | 1 | 発行証明書の内容・用法の規定 |
| その他 | 0+ | CAcert Office と協力する場合、IR 資料や景品が多数用意される |

表 4　CAcert イベント開催に必要な資料

多人数を相手にあまり時間を取らずに説明・認証をするため、CCA(CAcert Community Agreement) の印刷・配布や書類不備の防止のための確認員の準備が成功のキモになるようです。日本で開催する場合は、説明会の形式を取って、各座席に一部ずつ置いておき、解説の進行と共に記入して貰うような形がよいのではないでしょうか。

### 13.5.2　当日の進行手順（案）

具体的な進行方法については特に案内が見当たらなかったため、開催未経験者ですが進行案をここに書き出しておきます。

1. まず、クラスルーム形式で CAcert および発生する義務・責任の説明を行う。入室時に各自に 3-5 枚[*31]CAP Form を渡し、説明中に署名以外は全部記入してもらう。
2. 次に、Assurer と Member(予定者) で横並びに列を組んで、以下の図の通り Member にはカニ歩きをしてもらいながら認証をする。
3. 認証で新たな Assurer が生まれたら、Experience Point 獲得のためできるだけ Assurance 側に回ってもらう。

[*31] その場にいる Assurer のランクや人数による

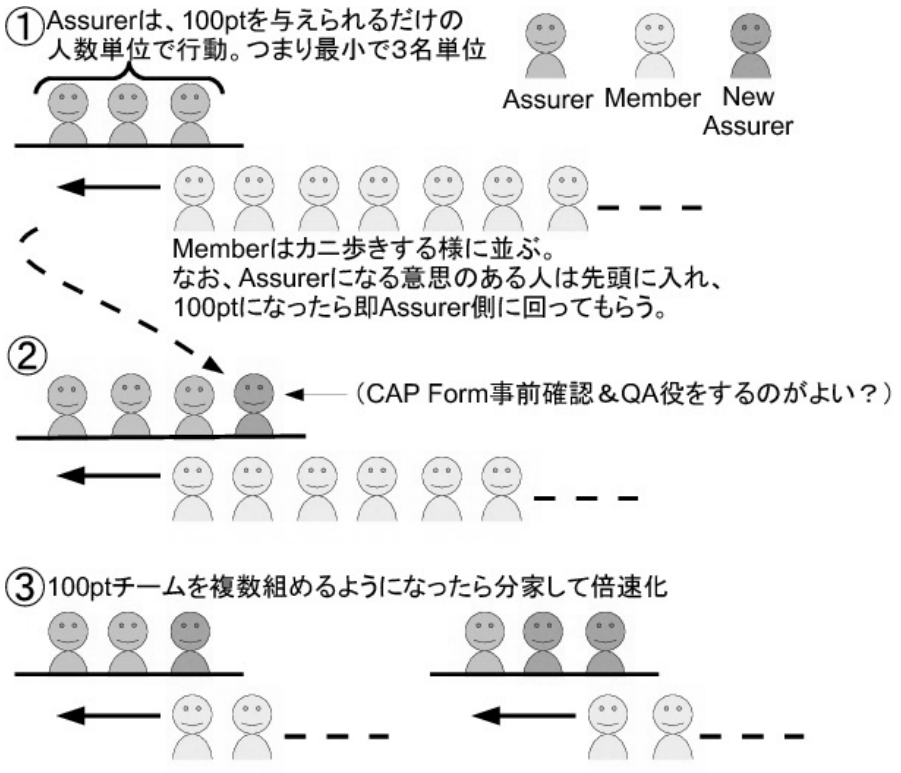



図 6　CAcert サイン会の進行案

以上の進行で Assurer の数を早く増やすことができます。

## 13.6　CAcert の主要資料と組織構成

CAcert Wiki などを読んでも資料が大量にあって大変、しかも組織構成も訳がわからなかった、という方もいるのではないかと思います（←自分のこと）。これは法的に監査を受け、商用の証明局並みの信頼度を獲得して普及を図るというゴールのために整備を重ねてそうなってきたのですが、理解しやすくなるように CAcert の主要ポリシ・資料の関係図と位置付け、また組織構成をまとめてみました。

まずは主要文書の一覧から。なお、公式文書（組織内の文書として法的に有効なもの）はすべて COD (CAcert Official Document) というコード番号が振られています。

| 文書番号 | タイトル（略称） | 状態 | 内容 |
|---|---|---|---|
| COD1 | Policy on Policy (PoP) | POLICY | ポリシ策定に置ける IETF 風のコンセンサスベースの運営方針およびドキュメント状態の管理方法を定める |
| COD2 | Configuration-Control Specification(CCS) | DRAFT | 監査対象となる DOC/HW/SW またはそれらへのポインタを定める |
| COD3 | CAcert Official Documents Policy (COD) | 破棄予定 | ドキュメント形式を定める予定だったが、不要として破棄予定 |
| COD4 | Non-Related Persons – Disclaimer and Licence (NRP-DaL) | 破棄済 | 外部の非メンバとの関係を規定する文書だったが、COD14(Root Distribution License) にてルート証明書の配布ライセンスを別途定めたので破棄された |
| COD5 | Privacy Policy (PP) | POLICY | サイトおよび証明書の利用においての個人情報の扱いを規定する |
| COD6 | Certification Practice Statement (CPS) | DRAFT | 認証局および証明書の提供機能・管理方針・保証範囲など全側面に渡る運営方針を規定する |
| COD7 | Dispute Resolution Policy (DRP) | POLICY | CAcert の運営およびメンバ間において発生する各種の係争に関する裁定方法を定める |
| COD8 | Security Policy (SP) | DRAFT | システムおよび鍵の管理システムが耐障害性・安全性・可用性の各面で満たすべき事項を定める |
| COD9 | CAcert Community Agreement (CCA) | 有効 | CAcert メンバーとして認証を受ける際に法的な束縛を行うための合意書 |
| COD10 | 欠番 | NA | |
| COD11 | Oranisation Assurance Policy(OAP) | POLICY | 組織体が CAcert 認証を受けるにあたって必要な手順と要件を定める |
| COD12 | 欠番 | NA | |
| COD13 | Assurance Policy (AP) | POLICY | 認証の保証範囲、ポイントの付与基準、留意事項など認証活動の全範囲を規定する |
| COD14 | Root Distribution License (RDL) | DRAFT | CAcert のルート証明書の配布ライセンスを規定する |
| - | Assurer Handbook | 有効 | 認証手順および実施要件の実務解説 |

各文書は WiP(Work in Progress - 策定中)、DRAFT（草案段階)、POLICY（正式ポリシ）の三段階を経て策定されます。ちなみに、今年（2010 年）はすべての文書が出揃い、CAcert の最終ゴールである証明局としての法的監査への準備が大きく進んだ年でした。

また、上記文書の中でも Member/Assurer の活動に特に密接なものを抜き出して関係を図示すると、以下の様になります：

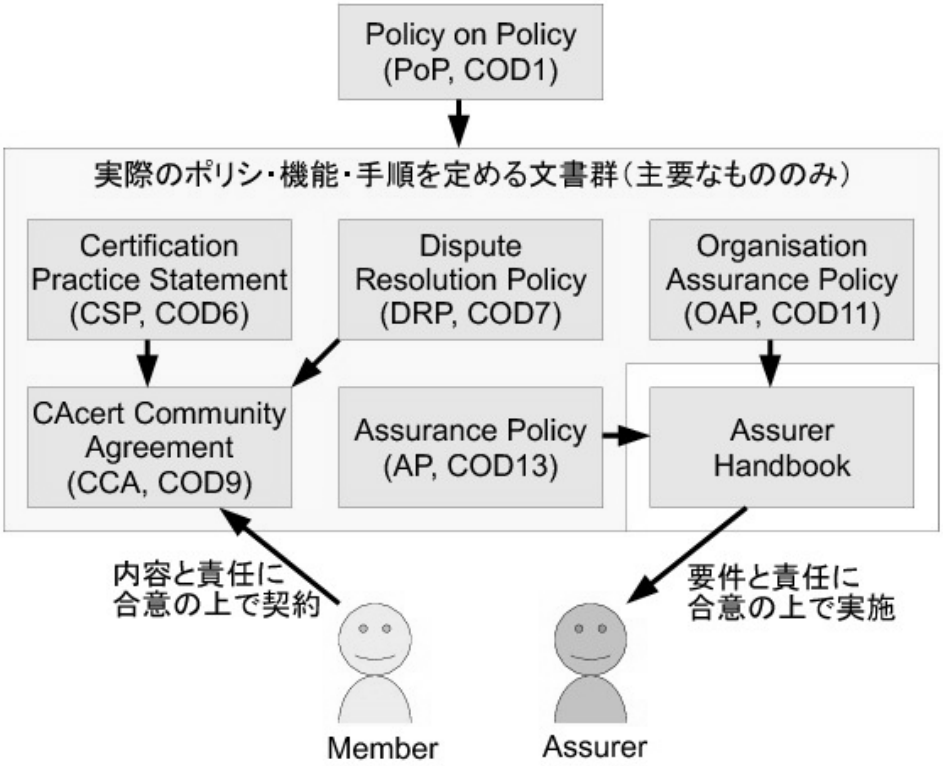



図 7　CAcert の主要ドキュメントの関係

各種文書では COD[0-9]+ や PoP/CCS/CCA/PP/DRL/... のような略称が頻繁に使われており初めて読むときは混乱してしまいがちです。まずは、図中の DRP(調停規則)/CCS(証明書運用規定)/CCA（会則）/Handbook（実務解説書）を押さえておくのがよいでしょう。

それでは次は組織構成です。問い合わせ先が見えにくいためまとめたのですが、実際に図にするとそう複雑でもなく、協同組合の CAcert Inc. を母体として、組織的な代表は通常組合員から選出・任命し、運営はコミュニティと共同という形態になっています。

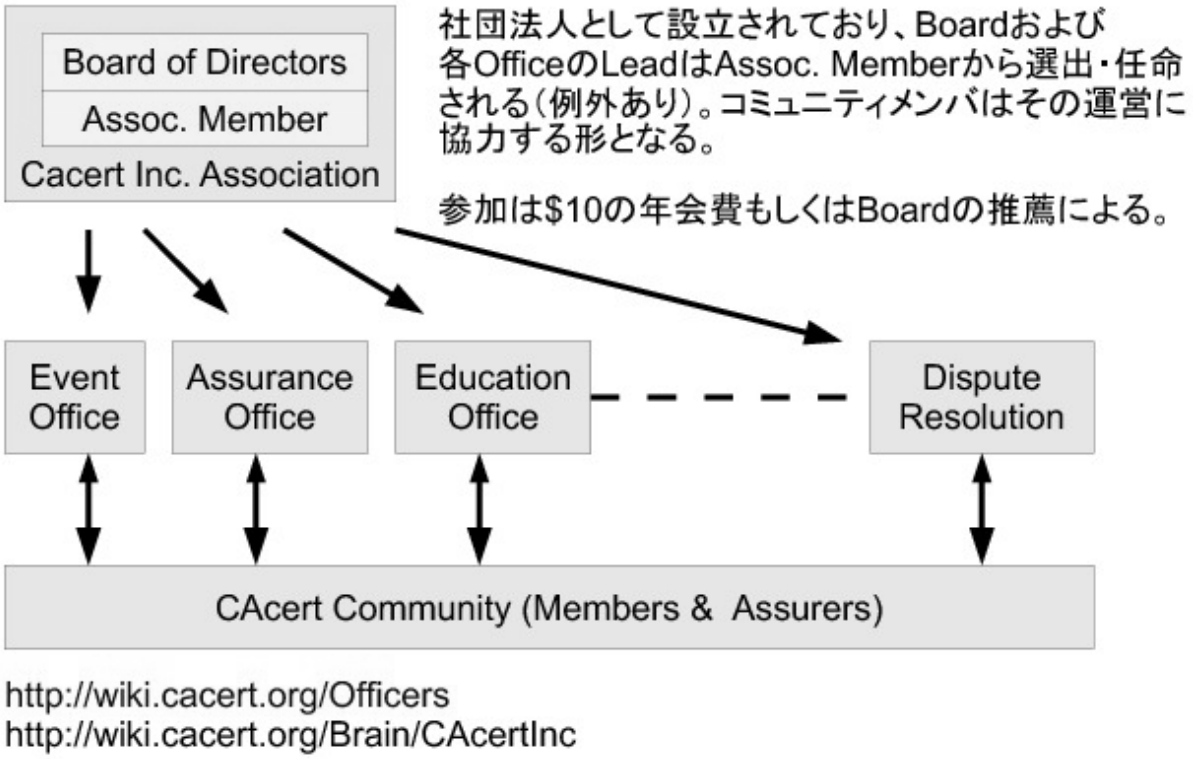



図 8　CAcert の組織構成（REF: `http://wiki.cacert.org/Officers`）

なお、連絡を取る場合は CAcert Wiki で該当するオフィス（担当チーム）を探す、というのが「正しい」方法ですが、情報が散らばっていたりして判断できないこともあるので、

とにかく `support@cacert.org` に連絡して、振り分けしてもらう

というのがよいでしょう。この ML はトリアージ役の人がウォッチしている（ことになっている）ので、適切な所に誘導して貰えるはずです。

## 13.7　まとめ

CAcert は組織も文書も、本格的な証明機関を目指しているだけあってかなり複雑になっていますが、利用するだけであればオンライン登録ですぐ証明書の発行を行えるので手軽に使うことができます。

これまでは欧州を中心[*32]に展開してきているため日本ではそもそも Assurer が少ない状況ですが、CAcert も世界巡業をするなど徐々に展開を図っているようです。あなたも CAcert に参加してみませんか？

## 13.8　おまけ

この文書のために CAcert について調べなおしていたら、

1. Using Secure DNS to Associate Certificates with Domain Names For TLS
   `https://datatracker.ietf.org/doc/draft-ietf-dane-protocol/`
2. 資料 - DNS に関わる技術動向 - KIDNS (Keys in DNS) -

[*32] CAcert Inc. は豪州ですが、サーバの実体や活動は欧州が中心

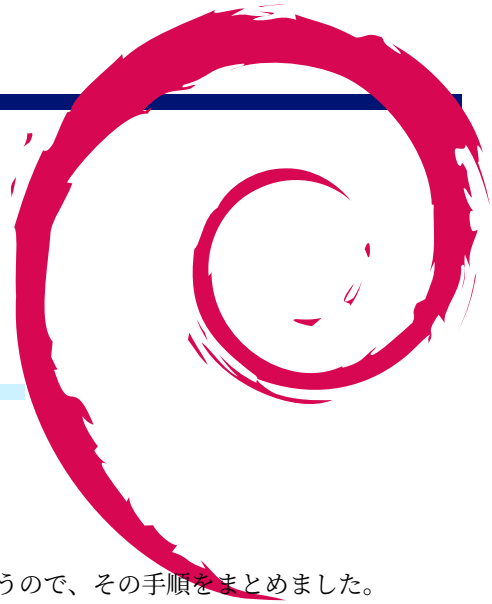
http://dnsops.jp/bof/20101125/201011-DNSOPS-KIDNSv5.pdf

というものを芋蔓式に見つけました。

要はアクセス先から証明書が送られてきたら、アクセス先ドメインの CERT/TLSA レコードを引いて、そちらから得られた証明書（またはそのハッシュ）で検証する仕組みや、同様の手法を SSH/PGP に適用する仕組みになります[*33]。これが普及したらドメインが絡むものについては、現行のルート CA からの信頼の階層という枠組み自体が不要になります。

実は 10 年近く前から検討されていたものの、最近ようやく DNSSEC が普及し始めた副産物として実用になってきたということですが、こういうものもあるということで、紹介しておきます。

[*33] 実は GnuPG はすでに対応しているそうです … 知らなかった



# 14 CACert Assurance

山田 泰資

先月の CAcert 報告に引き続き、今回は CAcert の assurance 会[*34]を行うので、その手順をまとめました。

また、手順について cacert.org の ML で質問していたところ、色々と話が転がって、3/4-5 のオープンソースカンファレンス（OSC2011 Tokyo/Spring @ 早大）で、関係者が来日の上、CAcert の公式イベントとして CAcert ATE (Assurer Training Event) を開催できることになりました。日本初となります。これについても報告＆協力募集します。

## 14.1 今日の手順

今日は練習を兼ねたミニサイン会を行います。必要な資材は用意してあるので、今日参加された方は政府発行の身分証明書があり、cacert.org にアカウントがあればポイントを受けられます。

以下、手順：

1. 認証を受けたい方は、CAP Form に以下の内容をまず記入して下さい
   （a）表面は、**署名および Assurer 用フィールド以外の各欄 に記入。名前や誕生日はローマ字と西暦で**
   **（b）裏面は、白紙部分に名前（漢字＆カタカナ）と和暦での誕生日を補記として記入**[*35]
2. Assurer は、以下の確認を行って下さい
   （a）CCA(CAcert Community Agreement) [*36]に同意することの口頭確認、および、その旨のチェックボックスにチェックがあること
   （b）CAP Form の内容がすべて記入されており、また、提示された身分証明書の通りであること
   （c）身分証明書が真正であり、内容が以下を満たすこと[*37]
      i. 誕生日記載の政府発行の物が 1 つはあること
      ii. 写真確認できるものが 1 つはあること
3. 最後に、**CAP Form に目の前で署名をし、そのまま Assurer に渡して別れて下さい。署名がないと無効です！**
4. **Assurer は、以下の追加作業を行って下さい**
   **（a）CAP Form に Assurer として署名し、それを最低 7 年間保管する**
   **（b）CAP Form 記載のメールアドレスを使って `http://cacert.org` で検索し、各人にポイントを付与する**

ポイントが 100pt を超えた方、超えそうな方は `https://cats.cacert.org/` のオンライン試験を通しておくと、100pt 以上になった時点から他の人を認証できるようになります。なお、アクセスには CAcert 発行のクライアント

[*34] 認証会？サイン会？もう GPG と合わせてサイン会でいいか？
[*35] これは来る OSC2011 で外国の方に認証してもらうための手順として策定中の方法です
[*36] CAcert に参加するにあたっての遵守事項
[*37] 可能ならば紫外灯などでの偽造チェックや、他の証明書（政府・民間発行は問わない）での追加確認をします

証明書が必要です。

## 14.2 CAcert@OSC2011 の計画

元々OSC2011 の Debian 枠の空き時間で GPG キーサイン兼 CAcert サイン会を企画していましたが、冒頭の通り cacert.org の方より打診があり、

> 日本初の CAcert 公式トレーニングイベント（ATE Tokyo）

を開催できることになりました。

ATE(Assurance Training Event) というのは通常のサイン会とは若干異なり、主に Assurer になる・なれる人を対象にした、上位 Assurer 養成（& CAcert の信頼性向上）のためのサイン会を兼ねたトレーニングイベントです。

たまたま期間中に来日する cacert.org の方（Peter Yuill 氏）がおり、OSC2011 の枠も別途確保できたため、日本での CAcert 立ち上げの好機として緊急で開催決定となりました。

検討中の日程は以下の通りです：

1. 3/5 昼に OSC2011 の正式セッションとしてトレーニング講座を実施する（サインも残り時間やブースで実施）
2. その上で、3/5 夕刻から追加サイン会を実施する。実施場所・時間は以下の 2 案で検討中
   （a）近隣の会議スペースや会場の空きスペースでサイン会をする
   （b）飲み屋やレストランのスペースを借りて、その中でサイン会をする

OSC2011 は 17:00 に終了なので、サイン会は前者なら 17:00-19:00、後者なら 18:00-エンドレス (酔っ払うまで) で考えています。飲み会内開催案は

> 「酔っ払いそうだが大丈夫か？」「大丈夫だ、問題ない」[*38]

と、先にサインをするなど手順を工夫すれば、場所や時間の確保問題をクリアする案としてよいかもという事です（ドイツの ATE でもやってみるとか）。

なお、準備のため、特に運営やサインに協力して下さる方を絶賛大募集中です。現在の準備状況は以下の通り：

| 必要なもの | 準備状況 |
|---:|---|
| 当日参加できる Assurer の方 | 未調整（来場者に 100pt 渡せるとベスト） |
| 当日参加できる運営の方 | 未調整（Assurer 担当と別に、CAP Form 確認や引率に数名） |
| 場外会場の確保 | 未了（隣接の区のスポーツセンタの優先権がある新宿区の人求む） |
| 事前告知＆申込ページの用意 | 未了（wiki.cacert.org で行う？atnd 等を使える？） |
| CAP Form や CCA などの紙資料 | 準備可能（各 100 部程度なら市の印刷機で激安） |
| ATE プレゼンや CCA の翻訳 | 未了（これが次の山か？頑張ります … ） |
| 案内チラシ（場所案内など） | 未了（内容が未作成。印刷は可能） |
| ブラックライト（Assurer 人数分） | 未了（製作計画中） |
| プリンタ（紙資料印刷用） | 準備 OK（持って行けます） |
| プロジェクタ（場外でプレゼンの場合） | 準備 OK（持って行けます） |
| 終了後レポート | 未了（実施レポートを cacert.org に出すまでが公式イベント、らしい。これはやります） |

なお、開催規模ですが、CAcert の方には、「Assurer または 100pt 近い Assurer 見込み」の状態の ATE 参加者は 10 名内外、また、その中で 150pt 達成済みの Experienced Assurer は数名だろうという見込みを伝えています。

[*38] 意訳

# 15　月刊 PPC64 ポーティング 2011 年 04 月, 05 月

山本 浩之

## 15.1　2011 年 04 月編

ppc64 porting は、2005 年頃に amd64 porting に貢献した Andreas Jochens によって alioth を使って行われていたのですが、ppc64 は amd64 と比較してネイティブサポートのメリットがそれほど大きくないとの結論に達し、2007 年頃、一度捨てられたプロジェクトです。

メリットが少ないとの結論に達したのには、

- PowerPC32 から PowerPC64 へは、amd64 と違い、レジスタ数が変わっていない
- PowerPC の命令は、PowerPC32 も PowerPC64 も、共に 32 bit の固定長のままである

という理由があります。

特に問題になったのが 32 bit の固定長命令で、以下のようになっています。

```
-------------------------------------------------------------------------
|    opcode    | src register | dest register |     immediate value      |
|    6 bits    |    5 bits    |    5 bits     |         16 bits          |
-------------------------------------------------------------------------
```

この中で即値が使えるのは「immediate value」である下位 16 bit だけです。

そのため、PowerPC32 では 32 bit の即値をレジスタにロードするためには、

```
16 bit 送る → 16 bit 送る
```

の 2 つの命令だけでした。

しかし PowerPC64 では、64 bit GPR の上位ワードに直接ロードするための命令が無いため、64 bit の即値をレジスタにロードするのに、

```
下位に 16 bit 送る → 下位に 16 bit 送る → 下位ワードを上位ワードにシフト → 下位に 16 bit 送る → 下位に 16 bit 送る
```

と、少なくとも 5 命令が必要です。

このため、通常だと、パフォーマンスが低くなることがあっても高くなることは無いのではないか、ということになり、プロジェクトは破棄されました。まあこれを趣味で拾ったのが私でして、なんとか根本的にパフォーマンスを上げていかなければならないという宿命を背負っています。

現在のところ、64 bit を使える CPU は PowerPC 970、POWER4、POWER5、POWER6、POWER7、Cell、PX などがあります。32 bit 専用の CPU より新しいため、POWER4 と POWER5 を除き、全て VMX (IBM & モトローラ) または AltiVec (モトローラ) と言うベクトル演算ユニットを備えており、それぞれを動かす命令は、同一な VMX 命令セットとして、完全に互換性があります。

さらに POWER6、POWER7、Cell、PX では VMX を拡張し、レジスタを 128 本とした VMX-128 ユニットが搭載されていますが、現在の gcc では、VMX-128 ユニットを使いこなすオプションはまだありません。

着手してかれこれ 1 年なんですが、とりあえず VMX ユニットを持たない POWER4 と POWER5 は、既に公式にある powerpc port を使ってもらうとして、ppc64 port ではこの VMX 命令をサポートする方向で、パフォーマンスが上がらないかと試しています。

ちなみに VMX 命令は、powerpcspe port でサポートされている、SPE 命令と全く同じところにデザインされていて、完全に排他的になっています。

## 15.2 2011 年 05 月編

週末ハッカーなおいらですが、今月の進捗状況を報告します。

今月は perl 2.12 投入、eglibc 2.13 投入、gcc-4.6 のデフォルト化などと、公式でも FTBFS 続出な月でした。しかしそれにもめげず、debian-ports への応募の前段階として、buildd 環境に最低限必要なパッケージ群を公開し始めました。

```
http://yama.fam.cx/debian/dists/unreleased/main/binary-ppc64
```

でも、まだ全ては揃っていません。足りないのは以下のパッケージです。

表 5 ppc64 で問題があるパッケージ（2011/05 の時点）

| パッケージ名 | 問題 |
|---|---|
| aptitude | build-deps なパッケージが FTBFS でビルドできていない |
| corutils | どうやら公式でも FTBFS らしいです。パッチが一ヶ月以上放置されています。 |
| debconf | all なパッケージなんですが、ppc64 では FTBFS でした。 |
| pam | なんのパッケージが悪さをしているのかはっきりとしていませんが、ある時点から Segment fault するようなパッケージしかビルドできなくなりました。 |

また、先月の宿題、「nbench で、ppc64 port を powerpc port と比較してみよ」ですが、以下の結果になりました。

```
ppc64 port:
-----
BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          762.72  :      19.56  :       6.42
STRING SORT         :          162.35  :      72.54  :      11.23
BITFIELD            :      1.4628e+08  :      25.09  :       5.24
FP EMULATION        :          165.36  :      79.35  :      18.31
FOURIER             :           12004  :      13.65  :       7.67
ASSIGNMENT          :          16.388  :      62.36  :      16.17
IDEA                :            2479  :      37.92  :      11.26
HUFFMAN             :          1095.6  :      30.38  :       9.70
NEURAL NET          :          25.628  :      41.17  :      17.32
LU DECOMPOSITION    :          806.48  :      41.78  :      30.17
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 41.242
FLOATING-POINT INDEX: 28.635
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
============================LINUX DATA BELOW============================
CPU                 : Dual
L2 Cache            :
OS                  : Linux 2.6.38-2-powerpc64
C compiler          : gcc version 4.6.1 20110428 (prerelease) (Debian 4.6.0-6)
libc                : libc-2.13.so
MEMORY INDEX        : 9.837
INTEGER INDEX       : 10.646
FLOATING-POINT INDEX: 15.882
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
-----

powerpc port:
-----

BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          781.12  :      20.03  :       6.58
STRING SORT         :           99.52  :      44.47  :       6.88
BITFIELD            :      1.8306e+08  :      31.40  :       6.56
FP EMULATION        :          168.36  :      80.79  :      18.64
FOURIER             :           11881  :      13.51  :       7.59
ASSIGNMENT          :          17.011  :      64.73  :      16.79
IDEA                :          3354.6  :      51.31  :      15.23
HUFFMAN             :            1149  :      31.86  :      10.17
NEURAL NET          :          23.981  :      38.52  :      16.20
LU DECOMPOSITION    :             731  :      37.87  :      27.35
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 42.220
FLOATING-POINT INDEX: 27.013
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
============================LINUX DATA BELOW============================
CPU                 : Dual
L2 Cache            :
OS                  : Linux 2.6.38-1-powerpc64
C compiler          : gcc version 4.5.2 (Debian 4.5.2-7)
libc                : libc-2.11.2.so
MEMORY INDEX        : 9.118
INTEGER INDEX       : 11.742
FLOATING-POINT INDEX: 14.982
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
-----
```

概要を言うと、ppc64 port は powerpc port と比べ、誤差範囲程度で同じです。ただし、「STRING SORT」で飛躍的に良くなり、「IDEA」で若干悪く、「LU DECOMPOSITION」で若干良くなっている結果となっています。

また、VMX 命令サポートの効果ですが、どうやら nbench は CPU 性能を主に計測するベンチマークソフトらしく、各アプリケーション自体のベンチマークは測定できませんでした。

# 16 Debian/m68k 開発

岩松 信洋

Debian/m68k 開発環境を構築する機会があったのでまとめてみました。

## 16.1 m68k とは？

m68k とはなにか？Motorola 680x0/m68000/68000 の事。省略して m68k。32bit で CISC。エンディアンはビッグ。世界中で未だに人気にある CPU の一つ。今はフリースケール・セミコンダクタによって、Coldfire という名前で製造および販売されています。なので今は 68k と言うことが多いです（モトローラじゃないから）。Apple 社の Macintosh SE やシャープの X68000、Palm Pilot の CPU として活躍していました。もちろん Linux でもサポートされており、Debian では hamm から正式にサポートアーキテクチャとして採用されましたが、etch から脱落しました。Debian に最初にポーティングされ、最初に脱落したアーキテクチャということで覚えておくと良いでしょう。

## 16.2 Debian/m68k の現状

表 6　m68k を使った主な機器

| メーカ | ハードウェア |
|---|---|
| ATARI | Atari Falcon |
| HP | HP 9000 Series 200 |
| SUN | Sun-1 |
| DEC | VAXstation 100 |
| SGI | RIS 1000 |
| SEGA | メガドライブ |
| SNK | ネオジオ |

etch から脱落した後も開発は続けられており、今は debian-ports.org 上で開発しています。1 年前に開発が停滞しましたが、Thorsten Glaser 氏[*39]が拾い上げ、数名の開発者と共に楽しく開発を続けています。

ポーティング開始当時は Macintosh や ATARI 社の Amiga 上で開発していましたが、既にこれらのハードウェアは入手が難しくなっているので主にエミュレータを使って開発しています。Debian の bootstrap が行える程度のパッケージは常に最新に近い状態が維持されているので、X や GUI を使わない環境程度ならすぐに構築可能です。

ちなみに、Debian に再度取り込むことは目標にしておらず、linux/m68k(68k) の開発用として生きる道を選んだようです。もし Debian で クロスコンパイルやエミュレータによる開発が許可されるようになったら、復活するかもしれません。

開発議論は ML（`http://lists.debian.org/debian-68k/`）と IRC（debian-68k@oftc）で行われています。

## 16.3 なぜ m68k に手を出してしまったのか？

先月、Ruby のコミッタになってしまったので、他になにかか自分でもできることないかなと思ってバグを見ていたら、Ruby1.9.1 パッケージのバグ #611691 （m68k が FTBFS）を見つけたのが事の始まりです。

[*39] MirOS の開発者。OpenWRT の開発者でもある。

## 16.4 開発環境設定方法

先にも説明したように、実機での開発は行われておらずエミュレータを使って開発が行われています。エミュレータといえば、ARM や SH などが使える qemu が有名ですが、qemu の 68k は不具合が多いので、Debian では ARAnyM [*40] という 68k エミュレータを使って開発しています。ここでは ARAnyM を使った開発環境の構築方法を説明します。

### 16.4.1 ARAnyM とは

ARAnyM（Atari Running on Any Machine） は 68040 + MMU + FPU(68882) を実装したエミュレータです。全ての 68k をサポートしているわけではなく、人気のあった CPU、特に Atari のハードウェアがサポートしていた CPU をサポートしています。グラフィックス、ディスクドライブ、CDROM、ネットワークもサポートしており、特徴として、OpenGL を使った高速なグラフィックと 4GB のメモリを扱えることがあります。

### 16.4.2 ホスト側の設定

まず、ARAnyM をインストールします。Debian パッケージになっているので apt で簡単にインストールできます。また、後で必要になるパッケージもインストールしておきます。

```
$ sudo apt-get install aranym p7zip
```

Debian m68k の開発に必要なカーネル、ユーザランドイメージのダウンロードします。

カーネルは linux-image-2.6.38-2-atari カーネルパッケージ[*41] を展開した vmlinuz を使います。

```
$ wget http://debian.nctu.edu.tw/debian-ports/pool-m68k/main/l/linux-2.6/linux-image-2.6.38-2-atari_2.6.38-5_m68k.deb
$ ar -x linux-image-2.6.38-2-atari_2.6.38-5_m68k.deb
$ tar -xzf data.tar.gz
$ ls boot/vmlinuz-2.6.38-2-atari
-rw-r--r-- 1 iwamatsu iwamatsu 1767311 2011-05-12 00:48 boot/vmlinuz-2.6.38-2-atari
```

次のユーザランドイメージをダウンロードします。build-essentail がインストールされたイメージが既にあるので、これを活用します。

```
$ wget http://people.debian.org/~smarenka/aranym/sid/disk.tar.7z
$ 7zr x -so disk.tar.7z | tar xvf -
$ ls -l disk.img
-rw-r--r-- 1 iwamatsu iwamatsu 10737377280 2011-05-18 00:37 disk.img
```

次にネットワークを設定します。以下で説明するネットワークは図 9 のようなネットワーク構成になるようにしています。

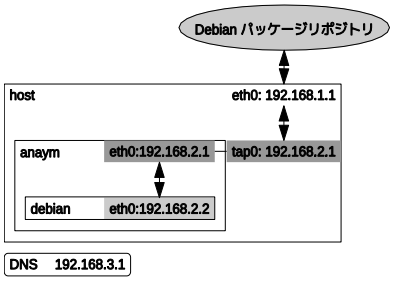



図 9 ARAnyM のネットワーク構成図

今回の ARAnyM 環境では tun を使うので uml-utilities パッケージをインストールします。

[*40] http://aranym.org/

[*41] http://packages.debian.org/search?keywords=linux-image-2.6.38-2-atari

```
$ sudo apt-get install uml-utilities
```

そして、tun および ARAnyM を使うユーザを uml-net に追加します。

```
$ sudo gpasswd -a iwamatsu uml-net
```

ホスト側の ネットワークを以下のように設定します。

```
$ cat /etc/network/interfaces
auto tap0
iface tap0 inet static
address 192.168.2.1
pointopoint 192.168.2.2
netmask 255.255.255.255
tunctl_user iwamatsu
up iptables -t nat -A POSTROUTING -s 192.168.2.2 -j MASQUERADE
down iptables -t nat -D POSTROUTING -s 192.168.2.2 -j MASQUERADE
```

フォワーディングを有効にして、tap0 ネットワークデバイスを上げます。

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
$ sudo ifup tap0
```

次に ARAnyM の設定を行います。ARAnyM は 特に指定しない場合には ~/.aranym/config を読み込みます。

```
$ cat aranym.config
[GLOBAL]
FastRAM = 768 # メモリサイズ。単位は MB。
Floppy =
TOS =
EmuTOS =
AutoGrabMouse = No
GMTime = Yes
[LILO]
# Linux カーネルイメージ
Kernel = vmlinuz-2.6.38-2-atari
# these Args for normal X operation
# カーネルコマンドライン
Args = root=/dev/hda1 console=tty debug=par
# these Args for headless
#Args = root=/dev/hda1 console=nfcon
# ネットワーク設定
[ETH0]
Type = bridge
Tunnel = tap0
# エミュレータで使う仮想ネットワークデバイスの Mac アドレス
Mac = XX:XX:XX:XX:XX:XX
[STARTUP]
GrabMouse = No
Debugger = No
[IDE0]
Present = Yes
IsCDROM = No
ByteSwap = No
ReadOnly = No
# ディスクイメージ
Path = disk.img
Cylinders = 20805
Heads = 16
SectorsPerTrack = 63
ModelName = Master
[VIDEO]
FullScreen = No
BootColorDepth = 8
VidelRefresh = 1
```

以上で設定は終わりなので、ARAnyM を使って、Debian OS を立ち上げます。

```
$ aranym-mmu -l -c aranym.config
```

```
$ uname -a
Linux aranym 2.6.38-2-atari #1 Mon May 9 16:39:31 UTC
 2011 m68k GNU/Linux
$ cat /proc/cpuinfo
 CPU:68040
 MMU:68040
 FPU:68040
 Clocking:73.5MHz
 BogoMips:49.04
 Calibration:245248 loops
```

### 16.4.3 ターゲットでの設定

Debian OS が立ち上がったら、root ユーザでログイン（パスワードは無し）し、ネットワーク設定を行います。起動時に ARAnyM の仮想ネットワークデバイス（nfeth:nat-feature) を eth0 として認識します。認識されている場合には ARAnyM の設定した MAC アドレスで eth0 が認識されています。

```
# dmesg  | grep eth0
eth0: nfeth addr:192.168.0.1 (192.168.0.2) HWaddr:XX:XX:XX:XX:XX:XX
```

もしホスト側の設定が間違っている場合、eth0 が存在しない状態になります。このような場合には、ホスト側の設定を見直してください。

eth0 が認識されているのなら、`/etc/network/interfaces` と `/etc/resolv.conf` を以下のように変更します。

```
# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.2.2
netmask 255.255.255.0
gateway 192.168.2.1
# cat /etc/resolv.conf
nameserver 192.168.3.1
```

設定が終わったら、各デバイスを上げネットワークがつながることを確認できれば、apt を使って最新の環境にアップデートし開発環境環境は完成です。

```
# ifup lo
# ifup eth0
# ping 192.168.2.1 # gateway へのチェック
# ping 192.168.3.1 # DNS へのチェック
# apt-get update   # apt-get update
# apt-get install debian-ports-archive-keyring
# apt-get update
# apt-get dist-upgrade
```

### 16.4.4 その他開発環境

エミュレータを使って開発できるのはすごく良いことなのですが、エミュレータだけでは遅いのでクロスツールチェインが欲しくなります。Debian でのクロス toolchain は emdebian プロジェクトが提供していますが、m68k のものは提供されていません。しかし、amd64 バイナリは Thorsten Glaser 氏が以下の apt-line で提供しています。

```
deb http://www.freewrt.org/~tg/debs68k/ cross main
```

## 16.5 ARAnyM 上での開発

動作しているのが エミュレータ上というだけで通常の開発と変わりません。cowbuilder も使えるので、遅いという以外には問題はないでしょう。開発速度を上げたい場合には、distcc/icecc/ccache などを駆使すれば快適な開発ができるようになります。このあたりの話はまた今度。

## 16.6 Ruby の FTBFS バグはどうなったのか？

Debian/m68k の開発環境は構築できましたが、Ruby のバグはどうなったのかというと、`http://redmine.ruby-lang.org/issues/4745` としてバグレポートし、r31646 でコミットしておきました。

# 参考文献

[1] wiki.debian.org の m68k の項目,`https://wiki.debian.org/M68k`

# 17 バグ報告はバグのためだけじゃないよ

のがた じゅん

## 17.1 はじめに

Debian を使っている時、なにか不具合を見つけたとします。そんな時、あなたはどういう行動をとるでしょうか。

Sid を使っていてよくあるケースは、パッケージのアップデート直後に以前と違う動きをしてバグに気がつくというケースでしょう。

そんな時、/var/log/apt/term.log や/var/log/aptitude のログを見て、アップデートしたパッケージにアタリをつけ、Debian バグ管理システム (BTS:Bug Tracking System)[*42]にそのパッケージのバグ情報を見に行きます。

すると、あなたの遭遇したバグはすでに報告されていて、結局何もすることなかった。で終わる人も多いかと思います。

そんなあなたのために、今回は不具合報告以外に焦点を当てたバグ報告について書いてみたいと思います。

## 17.2 おさらいも兼ねてバグ報告の基本

バグ報告以外の BTS の使い方、といっても基本的なバグ報告について知らなければ応用もできないので、おさらいも兼ねてバグ報告の基本について述べます。

Debian の BTS は、debbugs と呼ばれる Debian に特化したバグ報告システムを使っています。debbugs はメールベースの BTS で、submit@bugs.debian.org 宛にメールを送ることでバグを登録できます。

メールの送り方は、メールの Subject にバグの内容についての要約を書きます。メール本文には BTS として受け付けてもらうための擬似ヘッダを書きます。書き方は本文 1 行目にパッケージ名、2 行目にパッケージのバージョンを書きます。必要に応じて重要度とタグを書き、その後にバグの内容を書いていきます。

```
Subject: バグについて
----
Package: <パッケージ名>
Version: <パッケージバージョン>
Severity: <重要度 critical|grave|serious|important|normal|minor|wishlist>
Tags: <タグ>

以下バグの内容を書く
```

### 17.2.1 reportbug/reportbug-ng/debian-el を使う

Debian のバグ報告の基本は以上ですが、バグ報告のたびに毎回手で書くのは非常に面倒です。

reportbug や reportbu-ng を使うと定型化された部分は自動で埋めて、バグ修正に必要な情報も付記してくれるので、バグ報告にはこれらを利用しましょう。

[*42] http://bugs.debian.org/

reportbug を初めて使う場合はターミナルからは reportbug、GUI からは [メニュー] → [システムツール] → [reportbug] を実行すると、reportbug の初期設定が始まります。

詳しい初期設定の方法については「あんどきゅめんてっどでびあん 2009 年冬号」の「GUI がついて格好良くなった reportbug を使ってみよう」[9] に書いてあるので参考にしてください。

reportbug-ng を使う場合は、普段使うメーラー (MUA) と連携して利用するので、reportbug-ng のメニューバー [Edit] の Mail Client で MUA を確認しておきます。

emacs の人は debian-el パッケージを入れておいて、M-x debian-bug でいいと思います。

### 17.2.2 reportbug を使ったバグ報告の仕方

reportbug を使ってバグを報告するには、reportbug をそのまま起動するとバグ報告をしたいパッケージ名を尋ねられるので入力するか、reportbug にパッケージ名をつけて起動します。

```
$ reportbug (パッケージ名)
```

パッケージ名を入力すると、そのパッケージの既存のバグ一覧が表示されるので、目を通して、同じものがなければバグ報告に進みます。

reportbug の一連の流れも「あんどきゅめんてっどでびあん 2009 年冬号」の「GUI がついて格好良くなった reportbug を使ってみよう」[9] に書いてあるので参考にしてください。

## 17.3 Severity(バグの重要度) と Tags(タグ)

バグ報告の基本で擬似ヘッダに書く Severity(バグの重要度) と Tags(タグ) について触れましたが、ここでは「Debian – Debian BTS — 開発者情報」[*43]から引用して、簡単に説明します。

### 17.3.1 Severity(バグの重要度)

バグの重要度は critical(致命的) から、wishlist(要望) までの 7 段階に分かれています。通常使うのは normal が多いと思います。

critical (**致命的**) システム上の関係のないソフトウェア (またはシステム全体) を破壊する、重大なデータの欠落を引き起こす、または、そのパッケージをインストールしたシステム上でセキュリティホールが生じる場合。

grave (**重大**) 問題のあるパッケージが使用できない、またはほとんど使用できない。またはデータの欠落を引き起こす、そのパッケージを使用するユーザのアカウントにアクセスを許してしまうセキュリティホールが生じる場合。

serious (**深刻**) Debian ポリシーに対して見すごせない違反がある (大まかに言うと、「must」や「required」の要件に違反している)、またはパッケージメンテナあるいはリリースマネージャの意見としてそのパッケージがリリースに適していないと判断された場合。

important (**重要**) バグがパッケージの有用性を大きく損なっている場合 (ただし、誰にとっても完全に使用できなくなっている場合を除く)。

normal (**通常**) デフォルト値。通常のバグ。

minor (**軽度**) 問題がパッケージの利用に影響しない、かつ修正はたいした事がないと思われる場合。

wishlist (**要望**) 将来的な要望、主に設計上の理由により修正が非常に困難なバグ。

### 17.3.2 Tags(バグのタグ)

バグ報告には決められたタグをつけることができます。

とりあえず抜き書きをしてみましたが、個人的にはパッチをつけたときに patch を使ったことがあるような気がし

[*43] http://www.debian.org/Bugs/Developer

ますが、そんなに意識して使ったことがないです。

patch (**パッチ**)　バグ報告に、バグを修正するためのパッチや簡単な手順が含まれています。パッチがあってもバグを適切に解決できない場合や別の問題を生じる場合は、 このタグは使うべきではありません。

security (**セキュリティ**)　このバグはパッケージのセキュリティ問題を説明します (例: アクセスされてはいけないデータへのアクセスを許可する不正な許可属性がある、 やれるべきではない方法でシステムを制御できるバッファオーバーフローがある、 修正すべき DoS 攻撃の穴がある、等)。ほとんどのセキュリティバグは、critical (致命的) や grave (重大) の severity (重要度) も設定すべきです。

upstream (**上流**)　このバグは、パッケージの上流の部分に影響します。

l10n　このバグは、パッケージの地域化に関するものです。

sid/squeeze/lenny　ディストリビューションに関するタグです。

## 17.4　不具合報告ではないバグ報告とは

バグ報告についてひと通り書きましたが、タイトルにもあるように Debian はパッケージに由来する不具合のほかに、Debian に関連する問題があれば、すべて BTS に登録し管理します。

話だけではわかりにくいですが、reportbug を起動してパッケージ名の代わりに「other」と入力すると、このように登録するカテゴリ一覧が表示されます。

```
 1 base                   General bugs in the base system
 2 bugs.debian.org        The bug tracking system, @bugs.debian.org
 3 buildd.debian.org      Problems and requests related to the Debian Buildds
 4 buildd.emdebian.org    Problems related to building packages for Emdebian
 5 cdimage.debian.org     CD Image issues
 6 cdrom                  Problems with installation from CD-ROMs
 7 debian-i18n            Requests regarding Internationalization (i18n) of the
                          distribution
 8 debian-maintainers     Problems and requests related to Debian Maintainers
 9 debian-policy          Proposed changes in the Debian policy documentation
10 ftp.debian.org         Problems with the FTP site and Package removal
                          requests
11 general                General problems (e.g., that many manpages are mode
                          755)
12 installation-reports   Problems with installing Debian
13 listarchives           Problems with the WWW mailing list archives
14 lists.debian.org       The mailing lists, debian-*@lists.debian.org.
15 mirrors                Problems with Debian archive mirrors.
16 nm.debian.org          New Maintainer process and nm.debian.org website
17 press                  Press release issues
18 project                Problems related to project administration
19 qa.debian.org          Problems related to the quality assurance group
20 release-notes          Problems with the release notes
21 release.debian.org     Requests regarding Debian releases and release team
                          tools
22 security-tracker       The Debian Security Bug Tracker
23 security.debian.org    Problems with the security updates server
24 snapshot.debian.org    Issues with the snapshot.debian.org service
25 spam                   Spam (reassign spam to here so we can complain about
                          it)
26 tech-ctte              Issues to be referred to the technical committee
27 upgrade-reports        Reports of successful and unsucessful upgrades
28 wiki.debian.org        Problems with the Debian wiki
29 wnpp                   Work-Needing and Prospective Packages list
30 www.debian.org         Problems with the WWW site (including other
                          *.debian.org sites)
```

インストーラやプレスリリース、サイトについての問題といった多岐に渡る問題について扱っていることがわかると思います。

## 17.5　wishlist を使う

もうひとつ。不具合ではないバグ報告としては wishlist(要望) があります。

wishlist(要望) とは、パッケージに対しての要望がある場合に使用し、たとえばパッケージのソフトに対して何か機能を追加してもらいたい場合や、アップストリームで新しいバージョンのソフトがリリースされたときに、バージョンアップをお願いする場合などに使われます。

wishlist の使い方は簡単で、バグの重要度 (severity) に wishlist を指定して通常のバグ報告をするだけです。

```
$ reportbug gpsbabel-gui (reportbug にパッケージ名をつけて起動します)

Please briefly describe your problem (max. 100 characters allowed; you can
elaborate in a moment). This will be the bug email subject, so write a concise
summary of what is wrong with the package, for example, "fails to send email"
or "does not start with -q option specified" (type Ctrl+c to exit).
> Please include icon and desktop file. (要望の要約を書きます)
```

reportbug にパッケージ名をつけて起動します。起動すると問題の要約を尋ねられるので要望の要約を書きます。

```
Rewriting subject to 'gpsbabel-gui: Please include icon and desktop file.'
Removing release critical severities, since running in 'novice' mode.
How would you rate the severity of this problem or report?

1 important  a bug which has a major effect on the usability of a package,
             without rendering it completely unusable to everyone.

2 normal     a bug that does not undermine the usability of the whole package;
             for example, a problem with a particular option or menu item.

3 minor      things like spelling mistakes and other minor cosmetic errors
             that do not affect the core functionality of the package.

4 wishlist   suggestions and requests for new features.

Please select a severity level: [normal] 4 (wishlist を指定します。)
```

バグの重要度を尋ねられるので、4 番の wishlist を選びます。

```
Spawning sensible-editor... (エディタが起動します)

Subject: gpsbabel-gui: Please include icon and desktop file.
Package: gpsbabel-gui
Version: 1.4.2-1
Severity: wishlist

*** Please type your report below this line ***
(ここに要望の内容を書きます)

(以下略)
```

「*** Please type your report below this line ***」以下に要望を書き終えれば、通常のバグ報告と同じように submit@bugs.debian.org にメールを送るだけです。

## 17.6 パッケージ作成にまつわるバグ報告

パッケージ作成にまつわるバグ報告と書きましたが、「パッケージがないのにバグ報告とははなんぞや?」と思われるかもしれません。

パッケージを作成するためには、「こんなパッケージを作って欲しい」というパッケージ作成希望をバグ報告として出すことからスタートします。

「不具合ではないバグ報告」の章を思い出すと、こういうセクションがあったと思います。

```
29 wnpp                 Work-Needing and Prospective Packages list
```

この WNPP(Work-Needing and Prospective Packages) からパッケージ作成が始まります。

(と偉そうに書いていますが WNPP をしたことがないので、Debian – 作業が望まれるパッケージ: `http://www.debian.org/devel/wnpp/` に書いてあることをなぞっているだけです。何かあればツッコミをお願いします。)

```
$ reportbug
Please enter the name of the package in which you have found a problem, or
type 'other' to report a more general problem.
> other
```

reportbug を起動して「other」と入力します。

```
Please enter the name of the package in which you have found a problem, or
choose one of these bug categories:

 1 base                  General bugs in the base system

(中略)

29 wnpp                  Work-Needing and Prospective Packages list
30 www.debian.org        Problems with the WWW site (including other
                         *.debian.org sites)

Enter a package: 29
```

バグのカテゴリーを尋ねられるので 29 番の WNPP を選びます。

```
Are you sure you want to file a WNPP report? [y|N|q|?]? y

1 ITP  This is an 'Intent To Package'. Please submit a package description
       along with copyright and URL in such a report.
2 O    The package has been 'Orphaned'. It needs a new maintainer as soon as
       possible.
3 RFA  This is a 'Request for Adoption'. Due to lack of time, resources,
       interest or something similar, the current maintainer is asking for
       someone else to maintain this package. They will maintain it in the
       meantime, but perhaps not in the best possible way. In short: the
       package needs a new maintainer.
4 RFH  This is a 'Request For Help'. The current maintainer wants to continue
       to maintain this package, but they needs some help to do this, because
       their time is limited or the package is quite big and needs several
       maintainers.
5 RFP  This is a 'Request For Package'. You have found an interesting piece of
       software and would like someone else to maintain it for Debian. Please
       submit a package description along with copyright and URL in such a
       report.

Choose the request type: 5
```

WNPP の中で何をするのか尋ねられます。

- ITP: パッケージを作成する (Intent To Package)
- O: パッケージを手放す (Orphaned)
- RFA: パッケージを引き取って欲しい (Request for Adoption)
- RFH: パッケージの維持を手伝って欲しい (Request For Help)
- RFP: パッケージをつくって欲しい (Request For Package)

なので 5 の RFP を選びます。

```
Please enter the proposed package name: sahana-eden
Checking status database...
Please briefly describe this package; this should be an appropriate short
description for the eventual package:
> Disaster management system made from web2py
```

パッケージの簡単な説明を書きます。書き終えるとエディタが起動します。

```
Spawning sensible-editor...

Subject: RFP: sahana-eden -- Disaster Management System made by web2py
Package: wnpp
Severity: wishlist

*** Please type your report below this line ***

* Package name    : sahana-eden
  Version         : 0.5.3
  Upstream Author : Name <somebody@example.org>
* URL             : https://launchpad.net/sahana-eden
* License         : MIT/X
  Programming Lang: Python
  Description     : Disaster Management System made by web2py

Sahana Eden is an Emergency Development Environment which allows the
rapid deployment of customisable tools to support the 4 phases of the
Emergency Management Cycle as well as Development & Environment
projects. It is an example of an HFOSS project (Humanitarian FOSS)
```

書き終われば保存してエディタを終了します。

```
Report will be sent to "Debian Bug Tracking System" <submit@bugs.debian.org>
Wrong line:   Upstream Author : Name <somebody@example.org>
ERROR: you have composed an ITP with fields unchanged from the template; this will NOT be submitted. You should
edit all fields so they contain correct values for your ITP (e to edit) [E|q|?]?
```

適当に書いてると怒られてしまいました。ということで E で戻って書きなおして保存、submit@bugs.debian.org 宛に送れば RFP はおしまいです。

### 17.6.1 RFP を ITP にする

この辺りになると自分でも未知の領域ですが、「作業が望まれるパッケージ」によると

> これをパッケージ化することにしたら、このプログラムが パッケージ化の作業中ことをほかの人に知らせ、さらにあなたがそのバグの 所有者となるために、 バグレポートを「RFP」から「ITP」に改称します。それから ソフトウェアをパッケージ化し、アップロードして、そのパッケージが インストールされたらこのバグを閉じます。

とのことなので、control@bugs.debian.org 宛に「Debian BTS — 制御サーバ」http://www.debian.org/Bugs/server-control.ja.html を参照してコントロールメールを送ればよいのでしょうか。

```
owner (バグ番号) !
retitle (バグ番号) ITP: sahana-eden -- Disaster Management System made by web2py
thanks
```

調べきれず終わってしまった…。

## 17.7 最後に

不具合報告だけではないバグ報告を書きましたが、いかがでしたでしょうか。あまり自分も使うことがない事をまとめたので大変でした。

# 参考文献


[1] Debian – Debian BTS - バグを報告する http://debian.org/Bugs/Reporting

[2] Debian – 作業が望まれるパッケージ http://www.debian.org/devel/wnpp/

[3] Debian 開発者リファレンス「第 5 章 パッケージの取扱い方」http://www.jp.debian.org/doc/manuals/developers-reference/pkgs.html#newpackage

[4] 岩松伸洋「ITP の仕方からアップロードまでの流れ」(あんどきゅめんてっどでびあん 2005 年冬号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2005-fuyu.pdf pp. 3 – 7

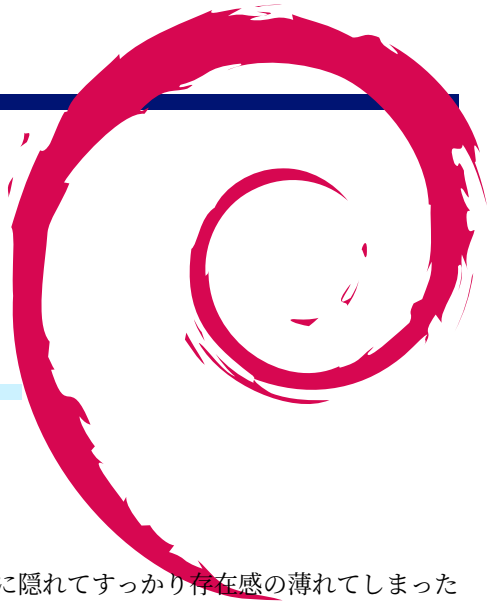
[5] やまねひでき「claim makes Debian better」(あんどきゅめんてっどでびあん 2005 年冬号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2005-fuyu.pdf pp. 26 – 29

[6] 上川純一「debbugs internal」(あんどきゅめんてっどでびあん 2005 年冬号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2005-fuyu.pdf pp. 31 – 38

[7] 木下達也「バグレポートから参加する Debian パッケージ開発」(あんどきゅめんてっどでびあん 2008 年夏号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2008-natsu.pdf pp. 76 – 78

[8] 藤澤徹「MC-MPI/GXP 公式パッケージへの道」(あんどきゅめんてっどでびあん 2009 年夏号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2009-natsu.pdf pp. 77 – 83

[9] のがたじゅん「GUI がついて格好良くなった reportbug を使ってみよう」(あんどきゅめんてっどでびあん 2009 年冬号) http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2009-fuyu.pdf pp. 61 – 66



# 18 dpkg のおさらい

倉敷 悟

今となっては、APT やその他いろいろ綺麗で便利な GUI ツールのかげに隠れてすっかり存在感の薄れてしまった dpkg。ですが debian システムの根幹を支えてくれている屋台骨であることには何の変化もありません。

というわけで、今回は新年度特集として、誰でも知っている (べき) dpkg のおさらいをしてみようと思います。

## 18.1 普段使いの dpkg

まずは文字通り、日常的に debian システムで作業をする上で dpkg コマンドを使う場面を振り返ってみましょう。事前課題として皆さんそれぞれの利用シーンも整理してもらいましたが、やはりある程度パターンは定まってくるようです。

### 18.1.1 deb ファイルの操作

基本中の基本ですが、バイナリパッケージファイルそのものを操作するコマンドです。

「ダウンロードした *.deb ファイルをインストールしよう」という時は：

```
dpkg -i <package filename>
```

APT と違って依存関係をひっぱってくれたりはしないので、個人的には–ignore-depends= <package name >を多用したりします。

「このパッケージファイルには何が入ってるんだろ？」という時は：

```
dpkg --contents <package filename>
```

### 18.1.2 パッケージ DB の操作

インストールされてパッケージ DB に登録された情報を照会するコマンド達です。実際の使用頻度が一番高いのはこのあたりでしょう。

「今どんなパッケージがインストールされてるんだっけ？」という時は：

```
dpkg -l [<string>]
```

「このパッケージでインストールされるファイルは何だっけ？」という時は：

```
dpkg -L <package name>
```

「このファイルはどのパッケージから来てるんだっけ？」という時は：

```
dpkg -S <file path>
```

「新しい PC に今のと同じパッケージをまるっと引っ越そうかな」という時は：

```
dpkg --get-selections > tmpfile
dpkg --clear-selections ; dpkg --set-selections < tmpfile
```

–clear-selections は、set-selection ができる状態でのみ実行するようにしておいた方がいいでしょう。

### 18.1.3 設定ファイルなど

そんなものの存在自体が初耳だ、という人がほとんどじゃないかという気もしますが、実は dpkg にも設定ファイルがあります。

```
/etc/dpkg/dpkg.cfg
/etc/dpkg/dpkg.cfg.d
~/.dpkg.cfg
```

書式は簡単で、dpkg コマンドのコマンドラインオプションからダッシュを省いたものを書いておくだけです。毎回こだわりの定番オプションを手打ちしている方は是非どうぞ。

ちなみに、手元の sid で自動登録されていた内容は次のような感じです。

```
no-debsig
log /var/log/dpkg.log
```

man dpkg して、どういう内容なのか見てみると、また新鮮な発見があるかもしれません。

## 18.2 パッケージの中身をのぞいてみる

さて、dpkg のことをだいぶ思い出してきたところではないかと思います。引き続き、今度は操作対象となるバイナリパッケージの中身を少し眺めてみることにしましょう。

### 18.2.1 バイナリパッケージの中身

実は、deb パッケージは単なる ar 書庫ファイルだったりします。なので、

```
ar t <package filename>
```

で書庫の中身を表示することができますし、同様に ar で中身を解凍して取り出すこともできます。出てくるファイルはこんな感じです。

```
debian-binary
data.tar.gz
control.tar.gz
```

この 3 つのファイル名はどんなパッケージでも同じです。それぞれ、次のような内容になっています。

debian-binary　バイナリパッケージのバージョンが記載されている。現時点では 2.0 のはず

data.tar.gz　パッケージに含まれるファイルの実体をまとめたディレクトリツリー。これだけ取り出せば checkinstall あたりと同じかも

control.tar.gz　パッケージのメタ情報とパッケージ前後処理のスクリプトなど

### 18.2.2 メタ情報

data.tar.gz は明確なのでひとまず置いておくことにして、control.tar.gz の中身を見てみましょう。

debconf **関連**　config templates

post/pre **関連**　preinst postinst prerm postrm

**パッケージ情報**　control conffiles md5sum

これらのファイルに記載されている情報は、下記の場所にそれぞれ適当に配置されます。

/var/lib/dpkg/info/　post/pre 関連、conffiles、debconf 関連などなど
/var/lib/dpkg/status　パッケージの導入ステータス

### 18.2.3 バイナリパッケージの分解と合成

先程 ar 書庫のことを書きましたが、実際にバイナリパッケージの中身をとりだしたい時は、別のコマンドで一度に解凍した方がラクかもしれません。

```
dpkg-deb --control <package filename> [<target directory>]
dpkg-deb --extract <package filename> [<target directory>]
```

指定した target directory (指定しなかった場合はカレントディレクトリ) に、それぞれパッケージ制御ファイルは DEBIAN/ ディレクトリ以下にまとめて、展開するツリーは直下をルートとしてそのまま、展開してくれます。

逆に、こういう状態で展開されているディレクトリを (DEBIAN/ と usr/ なりがある状態で) 指定して、

```
dpkg-deb --build <target directory>
```

を実行すると、バイナリパッケージ形式の ar アーカイブに固めることもできます。

## 18.3 dpkg 3 分黒魔 ^h ^h 演習

ここから、バイナリパッケージと dpkg をネタに、ちょっと困った場面で dpkg を使う練習をしてみましょう。

### 18.3.1 演習：dropbox の deb ファイルをインストールする

※この回の勉強会の直後に、dropbox のパッケージはまともに作り直されて main に入りました。当時の記録ということでそのまま残していますが、今見ても邪悪さは楽しめないかも知れません

`http://www.dropbox.com/downloading?os=lnx` から、ubuntu 専用の邪悪^h ^h 手抜きな deb ファイルをダウンロードしてください。そのままでは debian にインストールしようとするとエラーになってしまうので、ここまでの内容を使ってどうにかしてみてください。

アーキテクチャは各自の環境に合わせて適切な方を選んでください。これを間違えても dpkg コマンドでなんとかすることはできません。

この資料は事前に配布してしまうので、解答はセッションの方で説明します。

一応、解答としては安易な方法を 3 種類ほど想定していますので、余裕でできてしまった人は何種類か方法を考えてみたり、一応ソースもあるので、自分で make してバイナリパッケージ作成にチャレンジしてもいいでしょう。

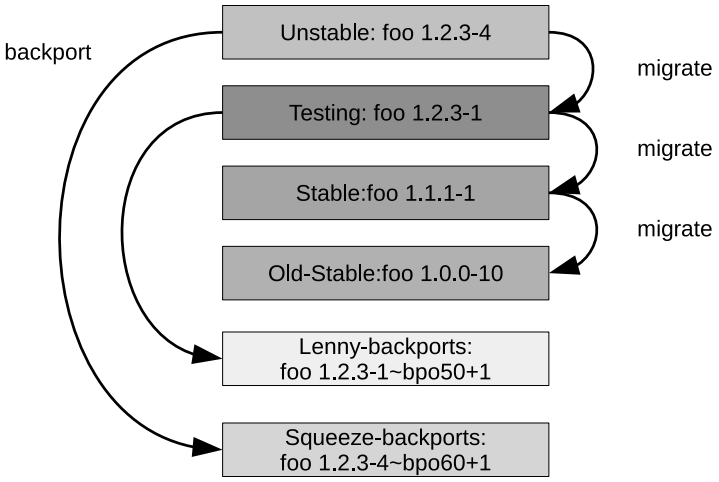
# 19 backports.debian.org

岩松 信洋

2010 の 9 月、正式に debian.org インフラの一部になった backports.debian.org(以下、bpo) ですが、実際にどのように使えばいいのかわからないところがあります。今回はユーザと開発者側からの視点で情報をまとめました。

## 19.1 backports.debian.org とは

backports.debian.org は testing や unstable で提供されているパッケージを既にリリースされた stable（執筆時点では squeeze）および old-stable（執筆時点では lenny） にバックポートしたパッケージを提供するプロジェクトです。バックポートされることによって、stable で提供されているパッケージより新しいバージョンを使うことができるようになります。




アップロードされるパッケージには、元のバージョンに **bpo{Debian リリース番号 }+{ ビルド番号 }** というバージョンが付加されます。これによって、どのバージョンをどの Debian リリースにバックポートしたのかわかります。また、セキュリティアップデートにも対応しています。この場合、DSA（Debian Security Announce）ではなく BSA（Backports Security Announce）となります。また、DSA とは連動していない点に注意が必要です。

## 19.2 backports.debian.org で提供されているパッケージを使う

まずは、backports.debian.org で提供されているパッケージの使い方について説明します。

### 19.2.1 どのようなパッケージが提供されているのか

現在 backports.debian.org で提供されているパッケージは `http://backports.debian.org/changes/squeeze-backports.html` から参照できます。

### 19.2.2 /etc/apat/sources.list に apt-line を追加する

/etc/apat/sources.list に backports.debian.org の apt-line を追加します。squeeze 向けには以下のように設定します。

```
deb http://backports.debian.org/debian-backports squeeze-backports main
```

contrib や non-free も提供しているので、有効にしたい場合には apt-line に追加できます。

追加したらリポジトリ情報を更新します。

```
$ sudo apt-get update
または
$ sudo aptitude update
```

### 19.2.3 パッケージをインストールする

backports.debian.org で提供されているパッケージをインストールするには、apt や aptitude の **-t** オプションを使って ディストリビューションを指定します。例えば、postgresql-9.0 パッケージをインストールするには以下のように実行します。

```
$ sudo apt-get -t squeeze-backports install postgresql-9.0
または
$ sudo aptitude -t squeeze-backports install postgresql-9.0
```

### 19.2.4 パッケージを更新する

backports.debian.org で提供されているパッケージに更新があった場合、**apt-get update ; apt-get upgrade** を実行しても、パッケージは更新されません。これは Pin-Priority の値が 100 （指定すればインストールできるが、アップグレードの対象にはならない）に設定されているためで、更新するには、apt の preferences を使って、パッケージのプライオリティを設定する必要があります。

例えば、 /etc/apt/preferences に以下のような設定をしておくと、backports.debian.org で提供されているパッケージが更新された場合、**apt-get update ; apt-get upgrade** で更新されるようになります。これは、セキュリティアップデートをこのコマンドで行いたい場合に必要な設定でもあります。

```
Package: *
Pin: release a=squeeze-backports
Pin-Priority: 200
```

また、常に backports.debian.org で提供されているパッケージを利用するようには、Pin-Priority の値を 500 に設定しておくとよいでしょう。(表 7[*44])

[*44] http://debian.fam.cx/index.php?AptGet#y193a8b6 から抜粋した

表 7　Pin-Priority の値と意味

| Pin-Priority | 意味 |
|---|---|
| <0 | インストールしない |
| 1 | NotAutomatic アーカイブ (experimental や backports 等) の優先値 |
| 1-100 | 指定すればインストールできるが、アップグレードの対象にはならない |
| 100 | 現在インストールされているパッケージの 優先値 |
| 101-500 | 通常のアーカイブよりも優先度が低いが、指定してインストールしたものはアップグレードの対象になる |
| 500 | **ターゲットリリース**に指定されていない通常のアーカイブの優先値 |
| 501-990 | **ターゲットリリース**指定のアーカイブよりも優先度が低いが、アップグレードの対象になる |
| 990 | **ターゲットリリース**に指定したアーカイブの優先値 |
| 991-1000 | 現在インストールされているパッケージよりも新しければ**ターゲットリリース**指定に関係なくインストールされる |
| >1000 | ダウングレードしてでも、そのパッケージをインストールさせる |

### 19.2.5　セキュリティアップデート

backports.debian.org で提供されているパッケージのセキュリティアップデートは DSA では行われません。BSA として行われ 誰かが 修正してアップロードします。(backports チームがチェック していると思われる) パッケージアップデートのアナウンスは backports-announce メーリングリスト (`http://lists.debian.org/debian-backports-announce`) で行われます。backports.debian.org を使っている人はメーリングリストに登録しましょう。

### 19.2.6　バグレポート

今のところ、backports.debian.org で提供されているパッケージは Debian BTS (`http://bugs.debian.org`) にバグレポートしてはいけないことになっています。なにか問題があった場合には、backports メーリングリスト (`http://lists.debian.org/debian-backports`) に投稿しましょう。backports.debian.org は Debian の正式なインフラなので、Debian BTS に統合される可能性もあります。その場合には、reportbug パッケージからもバグレポートできるようになるかもしれません。

### 19.2.7　欲しいパッケージがない場合

自分の欲しい機能がまだ stable で提供されているパッケージにはなく、unstable にあるパッケージで提供されているといったことはよくあります。このような場合、自分でビルドして使う方法もありますが、backports.debian.org で提供してもらうように依頼する方法もあります。まずはパッケージメンテナに直接依頼するのがよいのですが、パッケージメンテナが乗る気ではない場合、backpoorts メーリングリスト (`http://lists.debian.org/debian-backports/`) で依頼するという方法もあります。

## 19.3　backports.debian.org を使ったパッケージの提供方法

backports.debian.org は誰でも利用可能です。といっても、パッケージをアップロードするには、Debian Developer (以下、DD) である必要があります。Debian Maintainer もパッケージをアップロードおよび更新はできません。よって DD 以外はスポンサーアップロードをしてもらう必要があります。また、自分がメンテナンスしているパッケージ以外でもアップロードできます。ここでは、パッケージアップロードまでの流れと注意すべき点について説明します。

### 19.3.1 backports.debian.org キーリングへの登録

先では DD だとアップロードできると説明しましたが、すぐにアップロードできるわけではありません。Debian Developer キーリングと同じ鍵を backports.debian.org キーリングへの登録してもらうように申請する必要があります。この申請はリクエストトラッカー（`https://rt.debian.org/Ticket/Create.html?Queue=20`）を使います。

申請すると、数日後にキーリングに追加されます（リクエストトラッカーから連絡メールが来ます）。

### 19.3.2 アップロードするパッケージについて

backports.debian.org にアップロードするパッケージは testing や unstable にあるパッケージをリビルドしてアップロードするのではなく、パッケージそのものに手を加える必要があります。また、いくつか注意する点もあります。

- ディストリビューションを コードネーム-backports に変更する。
  debian/changelog では、ディストリビューションに stable や unstable を設定しますが、backports.debian.org にアップロードするパッケージのディストリビューションには **コードネーム-backports** を指定する必要があります。例えば、squeeze へバックポートしたい場合には、**squeeze-backports** とします。
- Debian バージョンに **bpo{Debian リリース番号 }+{ ビルド番号 }** を付加する。
  最初に説明したように、backports.debian.org にアップロードされるパッケージには他のディストリビューションとの違いが分かるように Debian バージョンに **bpo{Debian リリース番号 }+{ ビルド番号 }** というバージョンを付加する必要があります。例えば、unstable にある パッケージ foo の **1.2.3-4** をアップロードし squeeze-backports にアップロードしたい場合には、**1.2.3-4 bpo60+1** とします。60 は squeeze のリリース番号（6.0）です。もし、アップロードしたパッケージ問題があり、再アップロードしたい場合には、**1.2.3-4 bpo60+2** とし、ビルド番号をインクリメントします。
- backports だけの修正を行わない。
  特定のバージョンで発生するのはバグなので、unstable 等で修正して、それを backports.debian.org にバックポートするようにしましょう。
- squeeze および backports の環境でパッケージがビルドできるか確認する。
  backports.debian.org では、buildd と同様に複数のアーキテクチャ向けにパッケージがビルドされます。もし、stable および backports の環境でパッケージがビルドできない場合、FTBFS（Fail To Build From Source）になるため、注意が必要です。また、ABI の問題があるため、動作確認は慎重に行う必要があります。pbuilder[*45] や cowbuilder[*46] を使って、クリーンルームからのビルドチェックを行いましょう。
- インストールとアップデートの確認をする。
  パッケージができても、インストールとアップデートがうまく動作しない場合があります。これは、piuparts[*47] を使って確認できます。

### 19.3.3 パッケージのアップロード先

通常、パッケージは `ftp.upload.debian.org` にアップロードされます。しかし、backports.debian.org の場合には、`backports-master.debian.org` にアップロードする必要があります。通常、アップロードする時には dupload や dput というパッケージをアップロードするツールを使います。各アプリケーションの設定ファイルに以下のような設定を加えておきます。

- dupload の場合

[*45] `http://packages.qa.debian.org/p/pbuilder.html`
[*46] `http://packages.qa.debian.org/c/cowdancer.html`
[*47] `http://packages.qa.debian.org/p/piuparts.html`

```
$cfg{'bpo'} = {
  fqdn => "backports-master.debian.org",
  incoming => "/pub/UploadQueue/",
};
```

- dput の場合

```
[bpo]
fqdn = backports-master.debian.org
incoming = /pub/UploadQueue/
method = ftp
login = anonymous
allow_dcut = 1
```

### 19.3.4 セキュリティアップロードをする

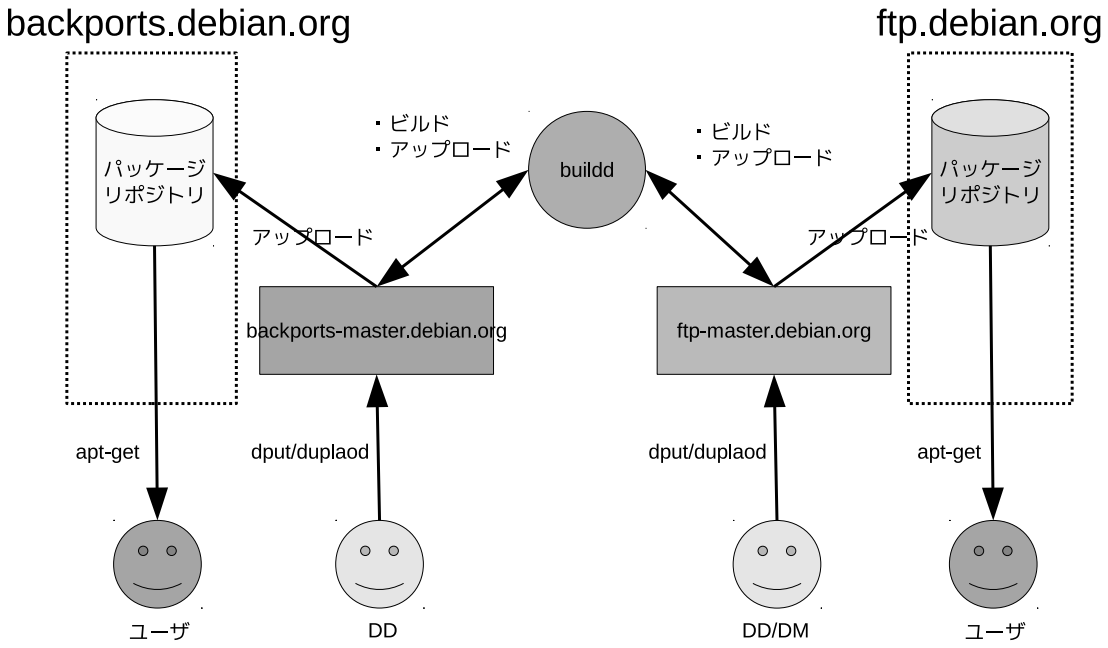
backports-master.debian.org にセキュリティアップロードを行う場合には BSA 番号を割り振ってもらう必要があります。この番号は backports チームで管理しているので、`team@backports.debian.org` に問い合せます。また、セキュリティアップロードの内容を PGP/GPG でサインして、backports-announce メーリングリスト（`http://lists.debian.org/debian-backports-announce`）にアナウンスします。

アナウンスする内容のテンプレートは以下のようになります。

```
Subject: [BSA-XXX] Security Update for <packagename>

<Uploader> uploaded new packages for <packagename> which fixed the
following security problems:

CVE-XXXX or whatever ID if existant
  short description
  ...
CVE-....
  ....

For the squeeze-backports distribution the problems have been fixed in
version <packageversion>.

<other distributions if any>
```

### 19.3.5 backports.debian.org の動作

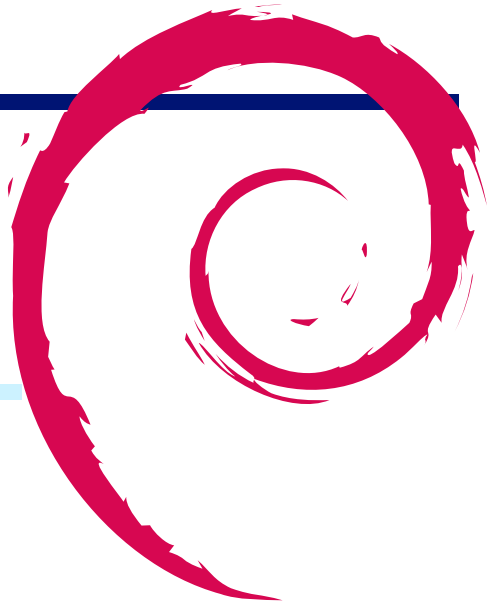
backports.debian.org の動作は buildd network と同じです。アップロード先やパッケージリポジトリが異なるだけで、buildd network を利用したパッケージのビルドを行っています。



## 19.4 まとめ

ユーザが backports.debian.org を使う場合、新しいバージョンを使えるというメリットがあるので使ったほうが良いと思います。先に書いたように BTS 等はうまく連動していないのと、パッケージメンテナではなくてもアップロードできてしまうので、問題があった場合には調整が必要になることがありそうです。このような場合には debian-backports メーリングリストをうまく活用していく必要があると思います。また、apt の pin の設定を正しくしておかないと、うまくアップデートしなかったりするので、注意が必要です。

開発者の場合には、メンテナンスコストが増えるだけであまりメリットがあるようには思えません。しかしユーザからの依頼があった場合には考慮する必要があるので、仕組みだけは知っておくとよいと思います。もう少し使いやすくするには他の Debian インフラとの連携が密になる必要があるでしょう。



# 20 pbuilder を使ってみよう

水野 源

## 20.1 pbuilder とは？

### 20.1.1 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。

### 20.1.2 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
```

### 20.1.3 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

### 20.1.4 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!)。

## 20.2 pbuilder の使い方

### 20.2.1 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。

### 20.2.2 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
```

### 20.2.3 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
```

### 20.2.4 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
```

### 20.2.5 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
```

## 20.3 pbuilder 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。

### 20.3.1 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
```

## 20.4 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。

### 20.4.1 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"
```

### 20.4.2 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"
```

### 20.4.3 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。

## 20.5 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。

# 21 Debian 勉強会予約システムにアンケート追加

上川 純一

## 21.1 アンケートシステムの目的

Debian 勉強会は各自のブログに記述するということが当初事後課題として機能していました。勉強会を開催したあと、参加者は勉強会で得られた内容を整理することができて、開催者は内容がどのようにうけとられたのかをある程度推し量ることができました。

しかし、近年勉強会の感想がブログに記述される率が低下してきています [1]。原因を想像するに、最近はコミュニケーションの道具としての twitter の流行などにともない、ブログに記述するという習慣が減ってきているからなのかもしれません。

時代の変遷によりブログエントリーがなくなるとその機能を代替するものは何でしょう。

ブログの機能としては、フィードバックのシステムと、あとその他のユーザに勉強会の記録を伝えるという面がありました。記録の面は twitter のログをまとめる togetter をなどを使えばよいだろうということが 2010 年 12 月の勉強会で提案され、それで問題は解決しそうです。残る課題はフィードバックシステムとしての機能です。

勉強会へのフィードバックシステムとしてアンケートシステムを提案します。

## 21.2 アンケートシステムの利用方法

### 21.2.1 管理者の事前準備

イベントの管理者はイベントの一覧画面からアンケート作成画面に遷移することができます。そこでは、ユーザに表示するための全体的なメッセージと、各項目を改行区切りで記述する欄があります。

ユーザにメールを送信すると、ユーザにはリンクの書いてあるメールが届きます。そのリンクをクリックすると、ユーザはアンケートに回答することができます。

### 21.2.2 ユーザ

ユーザは各項目については 0 から 5 までの値で回答することができます。そして、全体について自由記述の項目があります。[*48]

二つ目的があります。勉強会の企画の中で、どの部分が評価されているのかというのを出していきたいのと、任意なメッセージを入力できるようにすることで会に対してやりたいことなどの要望が伝えられるようにすることです。

[*48] 勉強会の会場では 5 択だと中心に回答が偏りがちなので、4 択にしたほうがよいのではないかという意見が出ました。Debian 勉強会アンケートシステムでは中心に偏り過ぎているとい傾向はまだ出ていないのでこのまま進めてみようと考えています。

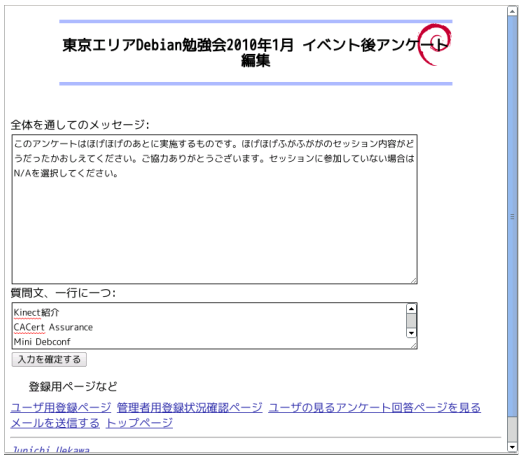



図 10　アンケート編集画面

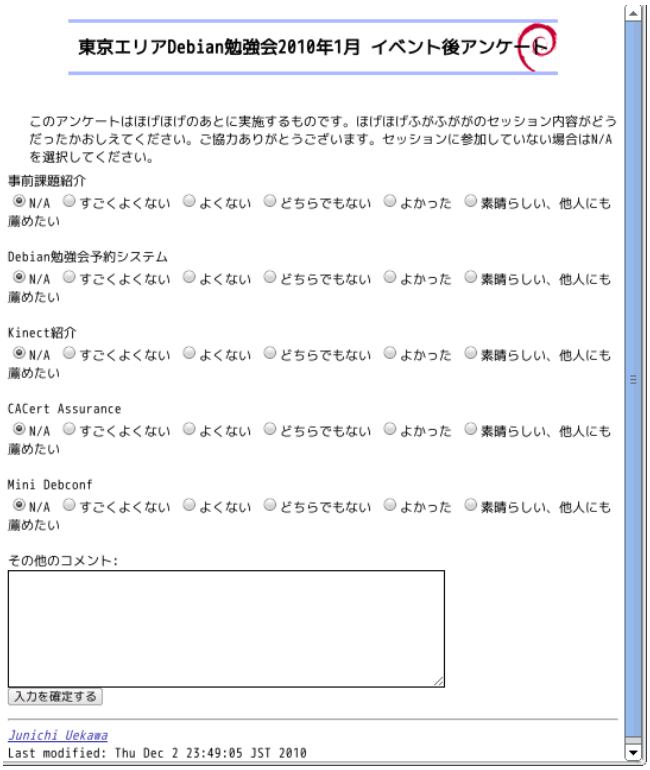



図 11　アンケート回答画面

表 8　アンケート回答結果の内部表現

0. N/A [*49]
1. すごくよくない
2. よくない
3. どちらでもない
4. よかった
5. 素晴らしい、他人にも薦めたい

### 21.2.3　管理者が事後に確認

管理者は、アンケートの [アンケート集計 CSV] リンクをたどることにより、イベント単位のデータの CSV ダンプを見ることができます。

特に集計処理はしていません。ここから先は CSV をとりだして処理することになります。

ここで、アンケートが匿名であるべきか記名であるべきかを最初考えていませんでした。どちらがどういう効果があるのかよくわかっていないので今後の課題としたいと思います。ブログ代わりなのであれば記名でよいと思うのですが、アンケートが記名だとバイアスがかかるというのであれば匿名のほうがよいかもしれません。システム的には今は匿名として扱っています。[*50]

[*50] Debian 勉強会の会場での議論では、回答の匿名性については。参加者のアンケート回答は匿名扱いにしたほうが答えにバイアスがかかりにくいだろうという意見が出ました。

## 21.3 アンケートシステムの設計と実装

アンケートシステムの設計と実装について見てみましょう。アンケートは Debian 勉強会予約システム [2] の一機能として実装しています。

### 21.3.1 データストア

ここの設計で悩んだ点は、EventEnqueteResponse を Attendance と一緒にするべきかどうかです。Datastore が join をサポートしていないためです。今回は分離して必要なときに手動で join[*51]するアプローチにしてみました。

```
class EventEnquete(db.Model):
    """Enquete questions for an event."""
    eventid = db.StringProperty()
    overall_message = db.TextProperty()
    question_text = db.StringListProperty()
    timestamp = db.DateTimeProperty(auto_now_add=True)

class EventEnqueteResponse(db.Model):
    """Enquete respnose for an event by one person."""
    eventid = db.StringProperty()
    # responses for 1-5 questions. 0 is N/A
    question_response = db.ListProperty(long)
    overall_comment = db.TextProperty() # a general comment from user.
    timestamp = db.DateTimeProperty(auto_now_add=True)
    user = db.UserProperty()
```

### 21.3.2 ユーザインタフェース

ユーザインタフェース部分は enquete.py に記述しています。メールや HTML ファイルの表示部分には、Django のテンプレートを利用しています。

- EnqueteAdminEdit.html 管理者用のアンケート編集画面
- EnqueteAdminSendMail.txt 管理者が参加者にアンケートを依頼するメール送信するときのメールのテンプレート
- EnqueteAdminShowEnqueteResult.txt アンケートの結果を CSV 形式で表示する。
- EnqueteRespond.html 参加者がアンケートを返答する際に表示される HTML。
- EnqueteRespondDone.txt 参加者がアンケートを回答したときに送信される確認メール。

### 21.3.3 バックグラウンドタスク

ウェブインタフェースからアンケートの送信リクエストがあった場合、クリックした瞬間にメールの送信処理が発生するわけではなく、メールの送信処理に taskqueue を利用しています。

送信元は noreply@debianmeeting.appspotmail.com を使っています。多分ログインユーザの権限でメールを出すということができないのだと想定してこうなっています[*52]。

## 21.4 先月のアンケート集計結果

それでは、例として 2010 年 12 月のアンケート結果をみてみましょう。0 は欠損値なので、R に処理させる場合は「NA」として処理します。これで R で処理した結果を見てみましょう。まだこの時点ではデータが少ないので、今回のなかでどのセッションが比較的ポジティブな感想を集めたのかということが分かるだけです。

事前準備として、統計解析ツール R[3] をインストールします。ここでは基本パッケージの r-base とドキュメントの r-doc-info[*53]をインストールしています。

[*51] 同じキーで二回別のテーブルを query することになるので必要なときには効率が悪いけど、あまり必要にならないと予想した設計

[*52] 未確認

[*53] R ではなにかわからないことがあればとりあえず info を見るか、help() コマンドをつかえばよいかんじです。

```
# apt-get install r-base r-doc-info
```

R を起動して CSV ファイルをロードして、まず基本的な統計量を調べます。

```
$ R
[中略]
> enquete <- read.csv('201012enquete.csv')
> summary(enquete)
  事前課題紹介   X2010 年の Debian を振り返って.2011 年を企画する
 Min.   :3.00   Min.   :3
 1st Qu.:3.75   1st Qu.:3
 Median :4.00   Median :4
 Mean   :3.75   Mean   :4
 3rd Qu.:4.00   3rd Qu.:5
 Max.   :4.00   Max.   :5
 NA's   :1.00   NA's   :1
 CACert の準備に何が必要か 俺の libsane が火をふくぜ Debian.Miniconf. 企画
 Min.   :4.000            Min.   :4.000           Min.   :1.000
 1st Qu.:4.000            1st Qu.:4.500           1st Qu.:2.000
 Median :5.000            Median :5.000           Median :3.000
 Mean   :4.556            Mean   :4.714           Mean   :2.778
 3rd Qu.:5.000            3rd Qu.:5.000           3rd Qu.:4.000
 Max.   :5.000            Max.   :5.000           Max.   :4.000
                          NA's   :2.000
> quantile(enquete$俺の, na.rm=TRUE)
  0%  25%  50%  75% 100%
 4.0  4.5  5.0  5.0  5.0
> quantile(enquete$Debian)
  0%  25%  50%  75% 100%
   1    2    3    4    4
```

ざっと数字を眺めただけですが、今回特に評価の高かったセッションは sane ネタと、CACert ネタで、評価の低かったセッションは Debian Miniconf 企画ネタでした。今回のアンケートの結果を受けると、Debian miniconf についてはプレゼンテーションの方法を考え直した方がよいだろうと思います。事前課題資料が無かった、プレゼンテーションしようとしたマシンがプロジェクタにうまくつながらなかった、そもそも話の内容もあまり練られていなかったということを考えるとそれがそのままアンケートの統計値にも反映したと考えられます。

一つ目線を変えて相関を調べてみました。しかし、まだこれをどう判断するべきか考えがまとまっていません。

```
> cor(enquete$CAC, enquete$俺の, use="complete.obs")
[1] 0.7302967
> cor(enquete$俺の, enquete$事前, use="complete.obs")
[1] -0.2581989
> cor(enquete$CAC, enquete$事前, use="complete.obs")
[1] 0
```

また、勉強会で議論した結果出てきた懸念点としては、点数のフィードバックは毎回出てこられても講師のやる気がそがれるなどの悪影響がある可能性がある点でした。講師には一年に一回程度まとめて返すのがよいだろうという提案がありました。

# 参考文献


[1] 上川純一, 「2010 年の東京エリア Debian 勉強会をふりかえって」東京エリア Debian 勉強会 Deb 専 2010 年 12 月号

[2] 上川純一, 「東京エリア Debian 勉強会予約システムの構想」東京エリア Debian 勉強会 Deb 専 2010 年 2 月号

[3] R Development Core Team, "R: A Language and Environment for Statistical Computing", R Foundation for Statistical Computing, Vienna, Austria, 2008 http://www.R-project.org

# 22 2010 年の東京エリア Debian 勉強会をふりかえって

上川 純一

今月で 6 年目の Debian 勉強会が終了しました。Debian Developer の数も着々と増えていき、当初の目標が達成されてきていますね。

今年東京エリア Debian 勉強会常連のやまねさんが Debian Developer になりました。苦節 6 年おめでとうございます。

Debian Maintainer 申請も複数通りました。

## 22.1 基本的な数値

Debian 勉強会は毎回事前課題事後課題を設定しており、予習復習を必要だとうたっている勉強会です。実際にどれくらいの人が出席しているのか、またその人たちがどれくらい事前課題・事後課題を提出しているのか、確認してみましょう。図 12 です。値は一年の移動平均です。

結果を見ると事前課題の提出率が 2009 年に低下傾向にあったのが、2010 年に反転上昇しています。これは Debian 勉強会予約システムの導入により事前課題の提出が簡単になったことと関係ありそうです。また、別の傾向として事後課題 (ブログ) の率が低下傾向にあります。もうブログは流行らないのかもしれません。

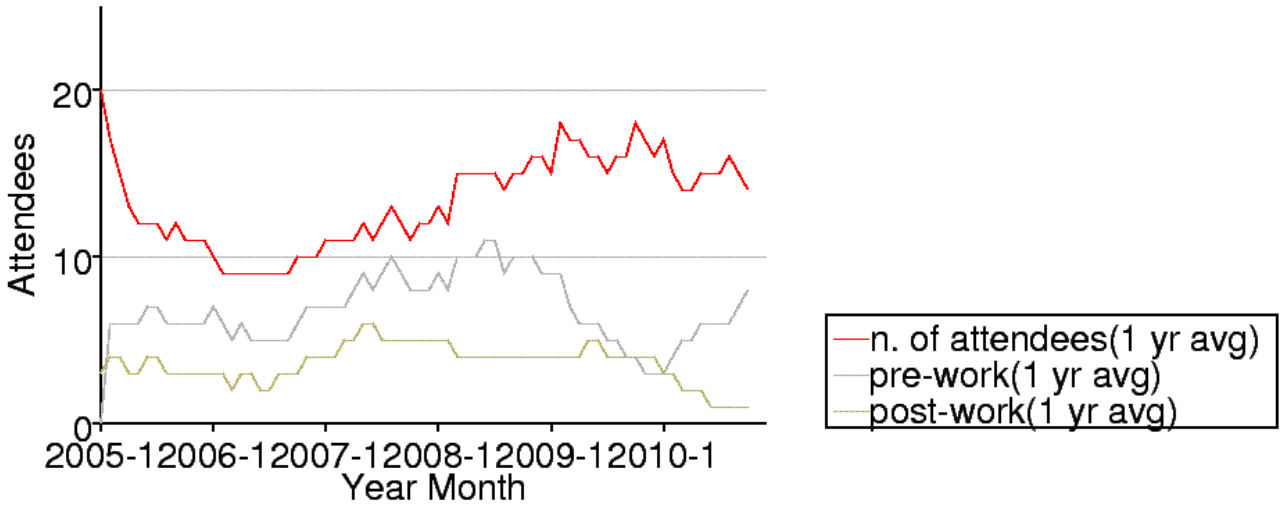



図 12　東京エリア Debian 勉強会事前課題・事後課題提出実績 (12ヶ月移動平均)

毎回の参加者の人数と、その際のトピックを見てみます。今年はとくに目立った感じの回はないですが、参加者についての記録が散逸しているようです。

今年の会場ですが、東京大学、筑波大学、木更津高専とアカデミックな場所の利用が増えました。[*54]。また、NIFTY さんの会場も使い、公営の会議室 (杉並区、オリンピックセンターなど) は数回しか利用していません。

表 9　東京エリア Debian 勉強会参加人数 (2005-2006 年)

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

表 10　東京エリア Debian 勉強会参加人数 (2007-2008 年)

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

[*54] OSC も明星大学で開催されました

表 11　東京エリア Debian 勉強会参加人数 (2009-2010 年)

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

# 23 関西 Debian 勉強会 2010 年度各種イベント開催実績と総括

倉敷・佐々木・野方

## 23.1 運営状況

### 23.1.1 勉強会全体

毎月のセッションについてですが、 勉強会の常連さんやイベント等で交流のある方に講師を依頼したりすることで、 参加者が関心を持っている様々な分野のネタを勉強会で行なう事ができました。その反面 Debian との関連性に薄いネタが増えている、という指摘も受けています[*55]。これは我々運営側が講師に依頼する際にもっと明確にお願いすべき事柄の様な気もします。今後の反省材料としたい所です。また、今回の事前課題で複数の方が提出されている通り、定番のネタ (Debian 入門的なお話やライセンス、パッケージ作成等) を雑誌のように毎年繰り返すことも考えてみたい所です。

講師については、継続して参加者して下さる常連さんへの講師依頼をする中で、その方が勉強会の運営側へ参加して下さったり、という嬉しい事もありました。また、Ubuntu コミュニティとの交流があったり、という感じでした。人の輪がこうして広がっていくのは嬉しいですね。

事前課題については、「Debian 勉強会登録システム」の実装と連動してシステム化されたこともあり、関西でも毎回の流れにのってきました。来年からは東京のように統計をとってもいいかな、と考えています。

イベント参加については、今年は春先に OSC Kobe が開催されたため、年間通して 3 回分をイベントセッションという形で実施しました。また、勉強会から fork (？) する形で GPG キーサインパーティの実施も定例となりつつあります。来年も今年通り 3 回の参加を予定しておりますが、takaya さんより「ユーザ向け」と「開発者 (含むワナビー) 向け」の 2 セッションに分けたらどうか、との提案もあったりしています。当面 4 月の OSC Kobe でのお話になりますが、検討してみたい所です[*56]。

`debian.org` については、今年度は佐々木、倉敷の 2 名がめでたく Debian Maintainer になりました。来年もこの流れを維持できればと思います。

### 23.1.2 イベント関連

例年通り、夏のオープンソースカンファレンス Kansai@Kyoto(OSC 京都) と、秋の関西オープンフォーラム (KOF) の出展にくわえ、3 月に神戸で開催されたオープンソースカンファレンス Kansai@Kobe(OSC 神戸) にも参加しました。

そして OSC 京都と KOF では、佐々木さんが中心となり GPG キーサインパーティを開催しました。

セッションの内容については、OSC 神戸や KOF では最近の Debian 関連情報を伝える軽めのセッション、OSC 京

[*55] ネタ自体は Debian に絡められるのに, あんまり関連性について触れられていない, 等
[*56] 人的リソースやそもそも二枠取れるのか, 等と気になる課題はありますが

都ではパッケージのバックポート講座を開きましたが、情報系より手を動かすセッションのほうが反響があったので、ハンズオン的なことを増やしていければと思います。

イベントについては FLOSS 関連のイベントが年 2 回から 3 回に増えたことと、GPG キーサインパーティの担当、イベント側のスタッフとして参加など、リソースが若干厳しくなってきたのでなにかいいバランスを取ることを考えたほうがいいのかもしれません。

## 23.2 開催実績

関西 Debian 勉強会の出席状況を確認してみましょう。グラフで見ると図 13 になります。表で見ると表 15 です。

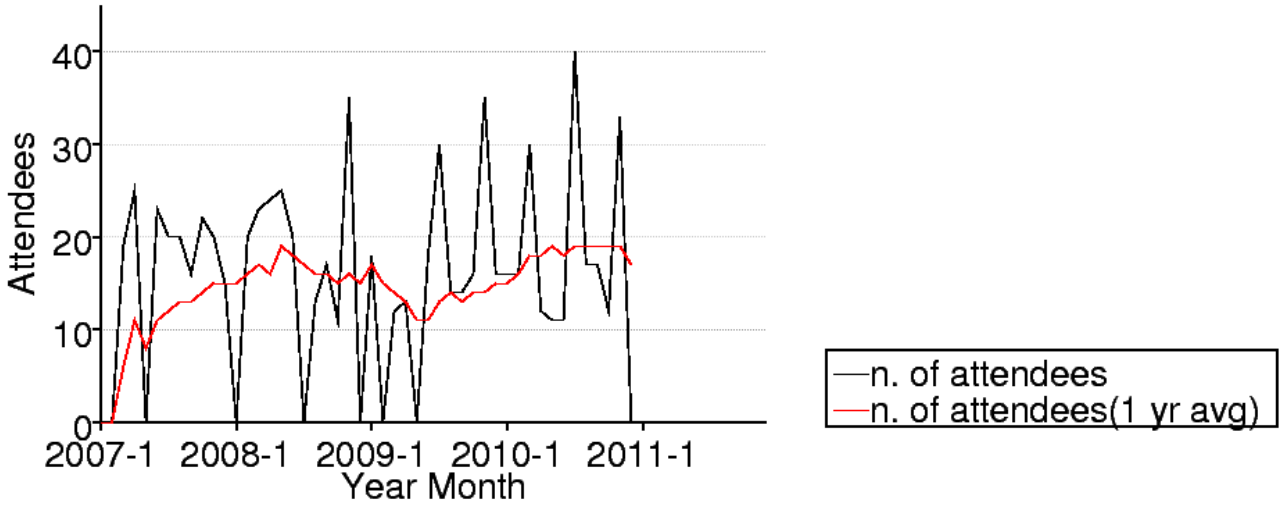



図 13　関西の参加人数推移

表 12　関西 Debian 勉強会参加人数 (2007 年)

| | 参加人数 | 内容 |
|---|---|---|
| 2007 年 3 月 | 19 | 開催にあたり |
| 2007 年 4 月 | 25 | goodbye、youtube、プロジェクトトラッカー |
| 2007 年 6 月 | 23 | 社会契約、テーマ、debian/rules、bugreport |
| 2007 年 7 月 | 20 前後 | OSC-Kansai |
| 2007 年 8 月 | 20 | Inkscape、patch、dpatch |
| 2007 年 9 月 | 16 | ライブラリ、翻訳、debtorrent |
| 2007 年 10 月 | 22 | 日本語入力、SPAM フィルタ |
| 2007 年 11 月 | 20 前後 | KOF |
| 2007 年 12 月 | 15 | 忘年会、iPod touch |

表 13　関西 Debian 勉強会参加人数 (2008 年)

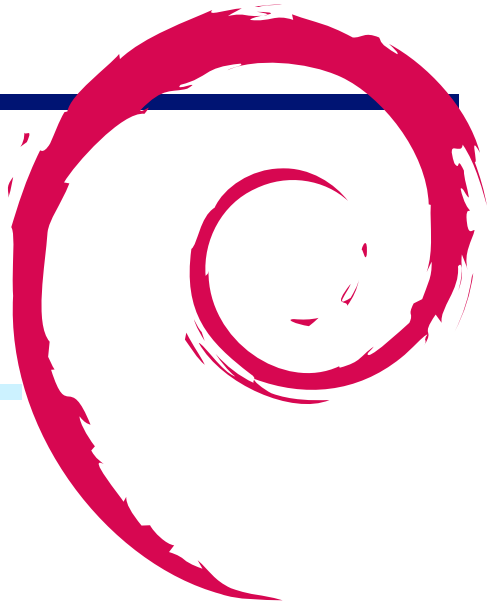
| | 参加人数 | 内容 |
|---|---|---|
| 2008 年 2 月 | 20 | PC Cluster, GIS, TEX |
| 2008 年 3 月 | 23 | bug report, developer corner, GPG |
| 2008 年 4 月 | 24 | coLinux, Debian GNU/kFreeBSD, sid |
| 2008 年 5 月 | 25 | ipv6, emacs, ustream.tv |
| 2008 年 6 月 | 20 | pbuilder, hotplug, ssl |
| 2008 年 8 月 | 13 | coLinux |
| 2008 年 9 月 | 17 | debian mentors, ubiquity, DFSG |
| 2008 年 10 月 | 11 | cdbs,cdn.debian.or.jp |
| 2008 年 11 月 | 35 | KOF |
| 2008 年 12 月 | ? | TeX 資料作成ハンズオン |

表 14　関西 Debian 勉強会参加人数 (2009 年)

| | 参加人数 | 内容 |
|---|---|---|
| 2009 年 1 月 | 18 | DMCK, LT |
| 2009 年 3 月 | 12 | Git |
| 2009 年 4 月 | 13 | Installing sid, Mancoosi, keysign |
| 2009 年 6 月 | 18 | Debian Live, bash |
| 2009 年 7 月 | 30? | OSC2009Kansai |
| 2009 年 8 月 | 14 | DDTSS, lintian |
| 2009 年 9 月 | 14 | reportbug, debian mentors |
| 2009 年 10 月 | 16 | gdb, packaging |
| 2009 年 11 月 | 35 | KOF2009 |
| 2009 年 12 月 | 16 | GPS program, OpenStreetMap |

表 15　関西 Debian 勉強会参加人数 (2010 年)

| | 参加人数 | 内容 |
|---|---|---|
| 2010 年 1 月 | 16 | Xen, 2010 年企画 |
| 2010 年 2 月 | 16 | レンタルサーバでの利用, GAE |
| 2010 年 3 月 | 30? | OSC2010Kobe |
| 2010 年 4 月 | 12 | デスクトップ環境, 正規表現 |
| 2010 年 5 月 | 11 | ubuntu, squeeze |
| 2010 年 6 月 | 11 | debhelper7, cdbs, puppet |
| 2010 年 7 月 | 40? | OSC2010Kyoto |
| 2010 年 8 月 | 17 | emdebian, kFreeBSD |
| 2010 年 9 月 | 17 | タイル WM |
| 2010 年 10 月 | 12 | initramfs, debian live |
| 2010 年 11 月 | 33 | KOF2010 |
| 2010 年 12 月 | 14 | Proxmox, annual review |



# 24 執筆者紹介

本号の執筆に参加したメンバを紹介します。

**本庄 弘典** 必要に迫られ自宅の本を自炊。焼き鳥が好き。

**山田 泰資** Debian JP Project Member。Debian JP 副会長。DD になって自分のソフトをパッケージ化して世界征服するのが野望。これでまた野望に一歩近づいた …?

**野島 貴英** 昔 Debian を使って稼いだ経験から、Contribution 生活に憧れる。しかしその実態は、典型 IT 土方生活との折り合いをなんとか付けようと日々もがくチキン野郎。

**まえだ こうへい** Debian JP Project Member。Debian JP 会長。あまり人の使わないマイノリティな技術を Debian で使って、愛猫こまめとヨメとの生活をいかに楽にできるかに腐心してる。

**岩松 信洋** Debian Project Member (Debian Developer)。東京エリア Debian 勉強会から輩出された Debian Developer 1 号。子供の影響で鉄分（鉄道）を多く摂取するようになった。

**山本 浩之** Debian JP Project Member。PPC64 を Debian にポーティングする日々を過ごす休日プログラマ。

**上川 純一** Debian Project Member (Debian Developer)、 東京エリア Debian 勉強会創始者。印刷物の完成を眺めるのが結構すき。

**森山 京平** NAIST（奈良の森の中の大学院）にいる Ubuntu User。

**杉本 典充** 北海道、関西、東京といろいろな土地で開催する Debian 勉強会に出没する道民。主に Debian GNU/kFreeBSD を使い unstable の醍醐味に浸る日々を過ごしている。

**のがた じゅん** Debian Live をこよなく愛す、38 歳無職ニートヒッキーパンクロッカー。

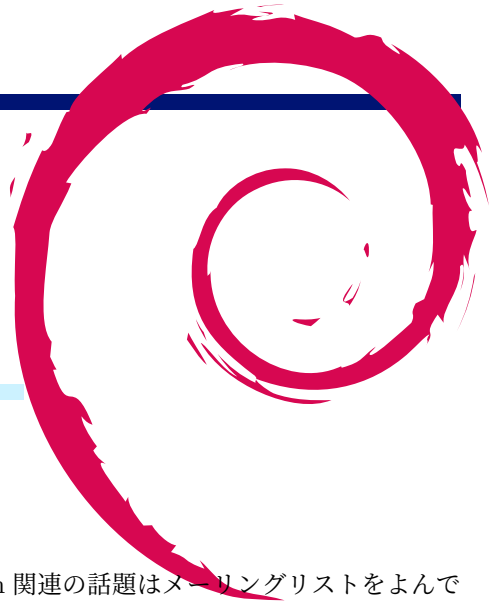
**水野 源** 理想のビールを求めて西へ東へ。最近はサッポロクラシックがフェイバリット。Ubuntu Japanese Team 所属。

**倉敷 悟** Debian Maintainer と翻訳で修行している、Debian Developer ワナビー。puppet 楽しいよ puppet。

**かわだ てつたろう** Debian のドキュメント周りに興味がある関西在住の Debian User。

**木下 達也** Debian Project Member (Debian Developer)。Wanderlust, Mew, w3m のメンテナ。

**山下 康成** LinkStation ／玄箱 を中心にいろいろいじっている人。



# 25 Debian Trivia Quiz

上川 純一

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。

問題 1. DACA ってなんですか?

☐ A Debian Admin Coaching Association

☐ B Debian を使うと (A) あの子と (C) クリスマスに (A) アレができるかもしれない

☐ C Debian's Automated Code Analysi project

問題 2. DebConf11 はいつ開催されるでしょう

☐ A 2011/6/24 - 30

☐ B 2011/7/24 - 30

☐ C 2011/8/24 - 30

問題 3. squeeze の Linux カーネルは一味違います。何が違うでしょう。

☐ A non-free firmware を排除した

☐ B Tux くんを排除した

☐ C 一つのカーネルイメージで kFreeBSD と Linux を提供します

問題 4. 2010/12/17 の時点で RC ハグはいくつあるでしょう。

☐ A 3

☐ B 83

☐ C 183

問題 5. New Maintianer フロントデスクに追加されたメンバーは？

☐ A Xavier Oswald

☐ B Enrico Zini

☐ C Kenshi Muto

問題 6. RC バグの現状ははどこで確認できるか

☐ A `http://bugs.debian.org/release-critical/`

☐ B `http://localhost/`

☐ C `http://debianmeeting.appspot.com/`

問題 7. Debian 勉強会予約システムの URL はどれか

☐ A `http://www.2ch.net/`

☐ B `http://atnd.org/events/`

☐ C `http://debianmeeting.appspot.com/`

問題 8. events@debian.org はどこと統合されたか

☐ A merchants@debian.org

☐ B hoge@debian.org

☐ C fuga@debin.org

問題 9. antiharassment@debian.org のうらにいないのは誰か

☐ A Amaya Rodrigo Sastre

☐ B Patty Langasek

☐ C Kouhei Maeda

問題 10. 現在いくつかの Sprint が開催され、企画されている。現在企画すらされていない sprint はどれか

☐ A -www sprint

☐ B security sprint

☐ C tokyo sprint

問題 11. DACA はどこを見ればよいか
□ A `http://qa.debian.org/daca/`
□ B `http://daca.debian.org/`
□ C `file:/tmp`

問題 12. DEP は何の略か
□ A Debian Enhancement Proposal
□ B Device Enhancement Protocol
□ C でっぷ

問題 13. DEP5 で提案されている debian/copyright の機械可読形式はどういうものか
□ A S 式
□ B RFC822 風
□ C XML

問題 14. Debian 6.0 がリリースされたのはいつか？
□ A 2011/02/05
□ B 2011/02/06
□ C 2011/02/07

問題 15. 今回のリリースでサポートされなくなったアーキテクチャは？
□ A m68k, ppc64
□ B alpha, arm, hppa
□ C blackfin, microblaze

問題 16. www.debian.org のデザインが変更されました。何年同じデザインを使っていたか？
□ A 11 年
□ B 12 年
□ C 13 年

問題 17. 次のリリースコードネームは何か？
□ A rc
□ B spell
□ C wheezy

問題 18. 今度行われる FTP-master 会議で議論されない予定は？
□ A パッケージ自動サイン機構
□ B クロスコンパイルサポート
□ C md5sum と shaXsum をとっぱらう

問題 19. Lars Wirzenius が提案した次のリリースゴール目標は？
□ A UTF-8 で動かないパッケージをなくす
□ B Debian で動いている人工衛星を打ち上げます
□ C 全パッケージを LLVM でビルドします

問題 20. 2011 年度 Debian JP 会長は誰でしょうか？
□ A 前田 耕平
□ B 岩松 信洋
□ C 荒木 靖宏

問題 21. 2011 年度 DPL は誰でしょうか？
□ A Kurt Roeckx
□ B Kenshi Muto
□ C Stefano Zacchiroli

問題 22. Debian Policy 3.9.2 で追加されていない項目はどれか
□ A Debian アカウントをメンテナアドレスに追加する必要があります。
□ B アーキテクチャ依存のライブラリ等は DEB_HOST_MULTIARCH で取得した値を利用する必要があります。
□ C 全てのパッケージは VCS で管理する必要があります。

問題 23. DebConf chairs に指名されたのは誰？
□ A Gunnar Wolf
□ B Maeda Kouhei
□ C Junichi Uekawa

問題 24. ries.debian.org で取得できるようになったデータは？
□ A debian.org の稼働状況データ
□ B debian.org の ML データを gzip で固めた物
□ C dak のデータ

問題 25. /run が消された理由は？
□ A /go のほうがよくね？という人が現れた。
□ B initscripts がまだサポートしてない！
□ C /run を入れたのはパッケージングミスです。

問題 26. HPPA と alpha の移転先はどこでしょうか？
□ A buildd.debian.or.jp
□ B buildd.debian-ports.org
□ C www.buildd.net

問題 27. linux カーネル 2.6.39 が Debian に入ることによって起きる変更は？

☐ A i386-bigmem が i386-pae になった

☐ B amd64 が i386 になった

☐ C i386 は amd64 のマルチバイナリになった

問題 28. Qt3 パッケージが削除されない理由は？

☐ A Qt3 ユーザによる哀願のため

☐ B LSB 4.1 が Qt3 を必要としているため

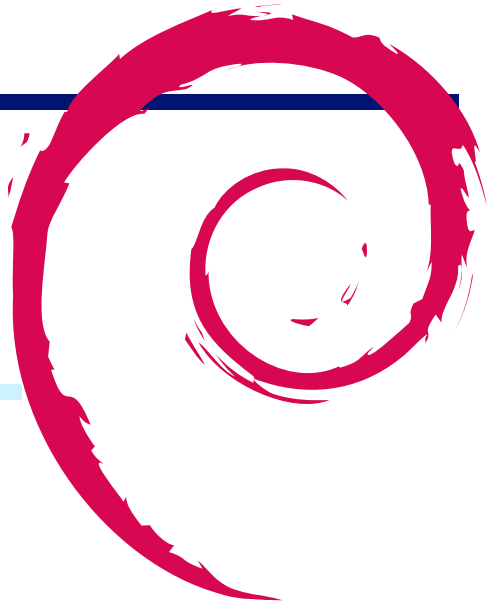
☐ C 削除の仕方がわからない

問題 29. Debian のサーバに追加された機能は？

☐ A ログインしているユーザを IRC に流す機能

☐ B RFC1149 の実装

☐ C DNSSEC



# 26 Debian Trivia Quiz 問題回答

上川 純一

Debian Trivia Quiz の問題回答です。あなたは何問わかりましたか？

1. C
2. B
3. A
4. C
5. A
6. A
7. C
8. A
9. C
10. C
11. A
12. A
13. B
14. A
15. B
16. C
17. C
18. B
19. A
20. A
21. C
22. C
23. A
24. C
25. B
26. B
27. A
28. B
29. C

**『あんどきゅめんてっど でびあん』について**

本書は、東京および関西周辺で毎月行なわれている『東京エリア Debian 勉強会』および『関西エリア Debian 勉強会』で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。収録範囲は東京エリアは 2010 年 12 月勉強会 (第 71 回) から 2011 年 05 月勉強会 (第 77 回)、関西エリアは第 42 回から第 47 回 まで。内容は無保証、つっこみなどがあれば勉強会にて。

**あんどきゅめんてっど でびあん 2011 年夏号**

2011 年 08 月 13 日　　初版第 1 刷発行

東京エリア Debian 勉強会/関西エリア Debian 勉強会 （編集・印刷・発行）