Debian勉強会資料集

2005年夏号

あんどきゅめんてっどでびあん

東京エリアDebian勉強会　著

TODO!このページ修正必要です.

目次

『あんどきゅめんてっど でびあん』について

本書は、東京周辺で毎月行なわれている『東京エリア 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。収録範囲は勉強会第 11 回から第 17 回まで。内容は無保証、つっこみなどがあれば勉強会にて。

1 Debian Policy 入門 第 1 回

岩松

1.1 Debian ポリシーとは

Debian GNU/Linux のポリシーです。Debian GNU/Linux として守るべき方針についてまとめられたものです。Debain パッケージの内部構成やオペレーティングシステムとして必要な設計部について示されており、ドキュメント化されています。

これらのドキュメントには debian-policy マニュアルと他の部分について補足するサブポリシーマニュアル があります。現在の debian-policy マニュアルバージョンは 3.6.2.2 です。毎日議論され、修正が加えられています。

1.2 debian-policy マニュアルの構成

deian-policy マニュアルの構成はどうなっているのか。

主に Debian パッケージの内容になっています。以前は debian-policy マニュアルと Debian パッケージングマニュアルに分かれていたのですが、統合されました (3.2.1.1 で統合) 。

以下に debian-policy の内容をリストにしてみました。

1.2.1 Debain アーカイブ

- Debian Free Software Guidelines (DFSG) とはなにか
- main / contrib / non-free セクションの説明および各セクションに収録されるパッケージの条件について
- Copyright の問題について
- サブセクションについて
- パッケージに対するプライオリティについて

1.2.2 バイナリパッケージについて

- パッケージ名について
- パッケージのバージョンについて日付に基づいたバージョン番号の付け方
- メンテナーのパッケージについて
- パッケージの説明についてパッケージの簡単な説明についてパッケージの詳細な説明について
- パッケージの依存について
- バーチャルパッケージ
- ベースシステムについて
- エッセンシャルなパッケージについて
- メンテナースクリプト

1.2.3 ソースパッケージについて

- 規格への対応
- パッケージ関係
- 上流パッケージソースの変更について
- Debian changelog（debian/changelog）代替の changelog 形式
- Makefile 内でのエラーのトラップについて
- タイムスタンプスタンプ
- ソースパッケージの中の物における制限
- メインビルドスクリプト: (debian/rules)
- Variable substitutions: (debian/substvars)
- 生成されたパッケージリスト: (debian/file)

1.2.4 コントロールファイルについて

- コントロールファイルの構文について
- ソースパッケージ制御ファイル (debian/control)
- バイナリパッケージ制御ファイル (debian/control)
- Debian ソース制御ファイル–.dsc
- Debian Change ファイル–.changes
- コントロールファイルのフィールドリスト
- ユーザによって定義されたフィールド

1.2.5 パッケージメンテナンススクリプトとパッケージがインストールされる手順について

- パッケージメンテナスクリプトの序論
- メンテナスクリプトの再入結果の同一性
- メンテナンススクリプトからのターミナルの制御
- メンテナンススクリプトの呼ばれ方の詳細
- インストールおよびアップグレードのアンパックフェーズの詳細
- 詳細な構成
- パッケージの削除とパッケージ設定の完全削除の詳細

1.2.6 パッケージ同士の関係について

- パッケージ関係フィールドの構文
- バイナリの依存について（Depends, Recommends, Suggests, Enhances, Pre-Depends の説明）
- バイナリパッケージのコンフリクト (Conflicts)
- バーチャルパッケージ (Provides)
- ファイルを上書きし、パッケージを置き換える (Replaces) 他のパッケージの中のファイルを上書きするパッケージの削除を強制して、全体のパッケージを置き換える

1.2.7 共有ライブラリについて

- ldconfig
- ランタイムサポートプログラム
- スタティックライブラリ
- 開発ファイル
- 同じライブラリのパッケージとのの依存関係
- ライブラリと他のパッケージとの依存 (shlibs システム) システム上の現在の shlibs ファイル dpkg-shlibdeps と shlibs ファイルの使い方について shlibs File フォーマット shlibs ファイルの提供する debian/shlibs.local file を書く

1.2.8 オペレーティングシステムについて

- ファイルシステム階層構造 (FHS)
- ユーザとグループ
- システムランレベルと init.d スクリプト
- init.d スクリプトからのコンソールメッセージ
- Cron ジョブ
- メニュー
- Multimedia handler(MIME)
- キーボード構成
- 環境変数
- doc-base パッケージを使ったドキュメントの登録方法

1.2.9 各種ファイルについて

- バイナリファイル
- ライブラリファイル
- 共有ライブラリ
- スクリプト
- シンボリックリンク
- デバイスファイル
- 設定ファイル
- ログファイル
- パーミッションと所有者

1.2.10 アプリケーションの変更について

- アーキテクチャ指定のための文字列
- デーモン
- 仮想 tty の使用、wtmp,utmp,lastlog 等の更新について
- エディタとページャについて
- Web サーバーとアプリケーション
- メール配送、配信、ユーザーエージェント

- ニュースシステムの設定
- X Window System 用のプログラム
- Emacs Lisp プログラム
- ゲーム

1.2.11 ドキュメントについて

- マニュアル (man pages)
- Info フォーマットのドキュメント
- 追加ドキュメント
- ドキュメントの管理
- 推奨されるドキュメント形式
- 著作権関連情報
- 設定例
- Changelog ファイル

1.2.12 付録

- Debian パッケージ パッケージングマニュアル

1.3 サブポリシーマニュアルについて

メインのものは debian-policy として存在し、そのほかに Emacs や Perl に関してのサブポリシーマニュアルというものが存在します。以下にサブポリシーマニュアルの簡単な説明を書きます。

1.3.1 build-essential パッケージの一覧

debian のシステム起動に必要なパッケージをリストにしています。このパッケージが規定されているドキュメントは

/usr/share/build-essential/list

になります。buiid-essential のリストは

/usr/share/doc/build-essential/essential-packages-list

に書かれており、build-essential としてパッケージに収録されています。(アーキテクチャによって内容が異なります。)

1.3.2 メニューシステム

menu システムを使うためのポリシー。引数を持たずに起動可能なアプリケーション (GIMP や xChat など) はメニューを使って起動できるようにするべきであり、どのようなアプリケーションがどのメニュー項目に入れるべきであるか、書かれています。debian-policy パッケージに収録されており、

/usr/share/doc/debian-policy/menu-policy.txt.gz

にインストールされます。

1.3.3 MIME サポート

MIME(Multipurpose Internet Mail Extension RFC1521) をサポートするためのポリシーです。MUA やウェブブラウザで MIME を扱えるようにできる仕組のようです。debian-policy パッケージに収録されており、

/usr/share/doc/debian-policy/mime-policy.txt.gz

にインストールされます。

1.3.4 Emacs ポリシー

Emacs に関連するパッケージは、サブポリシードキュメントに従うことが求められています。それをまとめたものが Emacs ポリシーです。emacsen-common パッケージに収録されており、

/usr/share/doc/emacsen-common/debian-emacs-policy.gz

にインストールされます。

1.3.5 Java ポリシー

Java に関連するパッケージのサブポリシー。ドキュメントは java-common パッケージに収録されており、

/usr/share/doc/java-common/debian-java-policy/index.html

にインストールされます。

1.3.6 Ruby ポリシー

Ruby に関連するパッケージのサブポリシー。ruby パッケージに収録されており、ドキュメントは

/usr/share/doc/ruby/ruby-policy.txt.gz

にあります。

1.3.7 Perl ポリシー

Perl に関連するパッケージのサブポリシー。debian-policy パッケージに収録されており、

/usr/share/doc/debian-policy/perl-policy.txt.gz

にあります。

1.3.8 Python ポリシー

Python に関連するパッケージのサブポリシー。python パッケージに収録されており、

/usr/share/doc/python/python-policy.txt.gz

にインストールされます。

1.3.9 Debconf 仕様書

Debconf のための仕様書。現在プロトコル 2。debian-policy パッケージに収録されており、ドキュメントは

/usr/share/doc/debian-policy/debconf_specification.txt.gz

にあります。

1.3.10 スペル辞書・ツールポリシー

パッケージの中で使う単語や isspell パッケージや myspell パッケージのためのポリシー。2003 年ごろからメンテナンスされてません。ドキュメントは

http://dict-common.alioth.debian.org/

にあります。

1.4 どのようにしてポリシーが決まるのか

1.4.1 policy-process

/usr/share/doc/debian-policy/policy-process.html/

に debian ポリシーの決め方が書かれています。

1.4.2 debian-policy@list.debian.org (ML) があります

ポリシーに関する疑問はこの ML に投げるといいでしょう。policy-process に則って、ここで議論され、承認されたときに debian-policy として反映されます。

1.4.3 debian-policy というパッケージがあります

間違いや提案はこのパッケージに対して BTS を行います. BTS されたメールは debian-policy@list.debian.org に forward されます。

1.4.4 debian-policy のメンテナ

現在の debian-policy のメンテナは以下の 4 人です。

- Julian Gilbey devscripts, tetex のメンテナ
- Branden Robinson X Strike Froce
- Josip Rodin debbugs, lintian のメンテナ
- Manoj Srivastava make, selinux のメンテナ

1.5 次回

次回からは、debian-policy を一つづつチェックして、つっこんだ解説をしていこうと思っています。

2 Debian Policy 入門 第 2 回

岩松

2.1 はじめに

今回から実際に Debian Policy の中身を見ていこうと思います。対象は Debian アーカイブについてと、バイナリファイルのポリシーについてです。

2.2 Debian アーカイブについて

Debian には大量 (1 万パッケージ以上) のパッケージがあります。それらを管理し、フリーなオペレーティングシステムを目指しています。このフリーという言葉はどういうものなのか、Debian ではどのように扱うのかということをガイドラインにしたものが Debian フリーソフトウェアガイドライン (以下、DFSG) です。

2.2.1 Debian フリーソフトウェアガイドライン とは？

Debian GNU/Linux システムのガイドラインである DFSG とはどのような内容なのか、確認してみましょう。

- 自由な再配布
  Debian システムを構成するソフトウェアのライセンスは、そのソフトウェアを複数の異なる提供元から配布されているプログラムを集めたソフトウェアディストリビューションの一部として、誰かが販売したり無料配布したりすることを制限してはいけません。また、ライセンスはそのような販売に対して使用料やその他の手数料を要求してはいけません。

  Debian にインストールされるソフトウェアのライセンスは自由に配布でき、無償またはお金を取って配布することが可能なライセンスでないといけない、ということです。ただし、ディストリビューションに含まれるプログラムのライセンスの内容に配布に対して料金を請求したりするものがあってはいけないということです。

  この項目に合わないライセンスの一つとして aladdin フリー公衆利用許諾契約書 (Aladdin Free Public License) があります。このライセンスは配布において手数料を取るのを禁じています。
- ソースコード
  プログラムにはソースコードが含まれていなければならず、かつ実行形式での配布に加えてソースコードでの配布をも許可していなければなりません。

  ソースコードの配布を許可してないライセンスは Debian にはインストールされることはないということです。

これはそのままです。

- 派生ソフトウェア
  ライセンスは、ソフトウエアの修正や派生ソフトウエアの作成を認めていなければなりません。そして、これらをオリジナルソフトウエアのライセンスと同じ条件の下で配布することが可能でなければなりません。

  あるソフトウェアを改変し、それを配布するときも改変元と同じライセンスで配布できないと Debain にはインストールされないということであり、派生を認めたライセンスでないとだめということです。

  この項目に合わないライセンスの一つとして Qmail のライセンスがあります。改変された場合には配布を認められてないからです。
- 原作者によるソースコードの整合性維持
  ライセンスは、プログラムを構築時に変更する目的で「パッチファイル」をソースコードとともに配布することを容認している場合に限り、ソースコードを修正済の形式で配布することを制限することができます。この場合、そのライセンスは修正済のソースコードから構築されたソフトウエアの配布を明示的に許可していなければなりません。またライセンスは派生ソフトウェアにオリジナルソフトウェアと異なる名前を付けること、あるいは異なるバージョン番号を付けることを要求できます (これは妥協案です。Debian グループは全ての作者に、ファイル、ソース、バイナリについての変更を制限しないよう奨めています)。

  パッチを配布するときに許可が必要とか、ソフトウェアのバージョンを替えてはいけないとかそういうことです。

  この項目に合わないライセンスの一つとして AT&T 公衆利用許諾契約書 (AT&T Public License) があります。このライセンスはパッチを公開するときには連絡しないといけません。
- すべての個人、団体の平等ライセンスは、すべての個人や団体を差別してはなりません。
  Debian には入れるな！とか、そんなライセンスはだめということです。
- 目標分野の平等
  ライセンスは、人々が特定の目標分野でプログラムを利用することを制限してはいけません。たとえば、商用利用や、遺伝学の研究でのプログラムの使用を制限していてはいけません。

  商業のみでしか使えないライセンスや研究目的のみで使用可能なライセンスではだめということです。

  この項目に合わないライセンスとして Jahia コミュニティソースライセンス (Jahia Community Source License) があります、このライセンスは学術のみで使用可能なライセンスになっています。
- ライセンスの配布
  プログラムに付随する権利は、プログラムが再配布されたすべての人々に対して、追加ライ

センスの履行を必要とすることなく、適用されなければなりません。

- ライセンスは Debian に限定されない
  プログラムに付随する権利は、プログラムが Debian システムの一部であるかどうかに左右されてはいけません。プログラムが Debian から取り出され Debian とは別に使用または配布されるとしても、その他の点でそのプログラムのライセンス条項を満たしているならば、プログラムが再配布されたすべての当事者は Debian システムにおいて付与されたのと同じ権利を与えられなければなりません。

  Fedora なら AT&T ライセンスだけど、Debian にインストールされるなら GPL にしていいよ といったライセンスでは不適合ということです。また、Debian 専用のライセンスではいけないということです。
- ライセンスは他のソフトウエアを侵害しない
  ライセンスは、そのソフトウエアとともに配布される他のソフトウエアに制約を加えてはなりません。たとえば、同じ媒体で配布される他のソフトウエアがすべてフリーソフトウエアでなければならないと要求してはいけません。
- フリーなライセンスの例
  ”GPL”、 ”BSD”、および ”Artistic” ライセンスは私たちが「フリー」と判断しているライセンスの例です。

  他のライセンスに関しては `http://www.debian.org/legal/licenses/` に書かれています。

以上の 9 つの項目全てに該当するパッケージが Debian のシステムとしてインストールされています。

2.2.2 セクション

先に書いたように Debian には大量のパッケージがあります。その中には DFSG に沿ったパッケージ以外のものや、輸出に制限があるパッケージも存在します。それらを区別するために Debian ではセクションを用いて分類しています。ここではこのセクションについて示されています。

- main セクション
  main セクションに入るパッケージは以下のことを満たしたパッケージである必要があります。
  - DFSG に準拠したパッケージであること。
  - コンパイル時に main に含まれないパッケージに依存していけない。
  - バグだらけなパッケージであってはいけない。
  - Debian Policy マニュアルに全て適合していないといけない。

  main セクションに入っているパッケージは DFSG に準拠したパッケージであり、それらのパッケージは Debian システムの一部です。
- contrib セクション contrib セクションに入るパッケージは以下のことを満たしたパッケージである必要があります。

– DFSG に準拠したパッケージであること。
– バグだらけなパッケージであってはいけない。
– Debian Policy マニュアルに全て適合していないといけない。

このセクションに入るパッケージの例としてコンパイルや実行するときに Debian に存在しないものを必要とするパッケージ。フリーではないプログラム用のラッパーパッケージや、フリーではないプログラム向けのフリーな付属物などが当てはまります。

contrib セクションに入っているパッケージは Debain システムとして認められていません。
実際のパッケージでは
– atokx
理由：atok for linux をインストールするプログラム
– quake2
理由：ゲームをするために Quake 2 の CD が必要（データがフリーではない。）
があります。

- non-free セクション
DFSG にに準拠していないパッケージや、特許や法律上、問題のあるパッケージが non-free セクションに入ります。
must meet all policy requirements presented in this manual that it is possible for them to meet.
実際のパッケージとして
– fglrx-driver
ATI のデバイスドライバ
– lha
lzh アーカイバー
があります。
non-free セクションに入っているパッケージは Debain システムとして認められていません。
- non-US セクション
sarge から non-US セクションが廃止され、同じアーカイブに収録されることになりました。現在、non-US のパッケージや apt-line は存在しません。

2.2.3 著作権の問題について

Debian にあるパッケージは著作権を示すファイルもポリシーとして決められています。

Debian では全てのパッケージがインストールされたときに、著作権やライセンスが /usr/share/doc/¡package-name¿/copyright として配布されないといけません。しかし、そのようなことができないパッケージは non-free に分類されるべきであると示されています。また、バイナリのみの配布は禁止されており、Debian の FTP にもミラーにも置いてはいけないことが示されています。

国際著作権法も挙げられており、著作権が明記されていないプログラムにも著作権が存在し、このようなプログラムに手を加えることによって著作権侵害で訴えられることもありえるのでこれらのプログラムは注意すべきであるとも書かれています。

2.2.4 サブセクション

パッケージをさらに種類別に分割したものです。main セクションと contrib セクション、non-free セクションにはさらにサブセクションが設けられています。このサブセクションは control ファイルの Section レコードに指定する必要があります。

main セクションの場合、サブセクションが x11 であれば　Section: x11
contrib セクションの場合は　Section: contrib/x11
non-free セクションの場合は　Section: non-free/x11

と指定する必要があります。

現在指定できるサブセクションは以下の通りです。admin, base, comm, contrib, devel, doc, editors, electronics, embedded, games, gnome, graphics, hamradio, interpreters, kde, libs, libdevel, mail, math, misc, net, news, non-free, oldlibs, otherosfs, perl, python, science, shells, sound, tex, text, utils, web, x11.

これらのサブセクションの分類方法は規定されていません。(base サブセクション以外) (ほんとか？)

2.2.5 プライオリティ

それぞれのパッケージにプライオリティが設定されるべきあるとポリシーに示されています。このプライオリティは Debian システムでのパッケージの重用度を定めています。この情報は Debian のパッケージ管理ツールが優先順位の高いパッケージを優先順位の低いパッケージから選択する際に使用します。

required　Debian のシステムで必要なパッケージにあたえられる重要度です。例えば base-passwd とか。アンインストールしようとすると警告メッセージが表示されます。

important　どんな Unix ライクなシステムにおいて存在することを期待されているプログラムはこの重要度を指定すべきです。しかし、X-Window-system や Emacs の大規模なプログラムは含まれません例えば、manpages パッケージがあります。[*1]

standard　スタンダードなアプリケーションがこの重要度を指定すべきです。perl や pyton , Emacs など。

optional　X-window-system などがこの重要度が指定すべきです。とりあえず、入れとくか程度のもののようです。optional なパッケージは互いに conflict しないように設定しないといけないようです。

extra　プライオリティとして required , important ,standard ,optional のいずれかに指定されている他のパッケージと衝突するパッケージはこの重要度が指定されます。しかし、Conflicts で指定している extra のパッケージもあれば、指定していないパッケージもあります。また、パッケージは自分のプライオリティより低いプライオリティがあたえられたパッケージに依存していはいけません。(ビルド時の依存は除きます)。

[*1] こういうのは人によって異なると思うのですが。

2.3 バイナリファイル

Debian では、dpkg と呼ばれるパッケージ管理システムをベースにしています。よって、Debian で配布される全てのパッケージは.deb 形式で提供しなければなりません。

ここではこの deb 形式での配布方法等について示されています。

2.3.1 パッケージ名

全てのパッケージ名は Debian アーカイブ内でにおいて重複しない名前でないといけません。パッケージ名はアルファベット小文字、数字 (0-9) と +,1, ピリオド (.) のみで構成されてないといけません。詳細は Debian ポリシーのセクション 5.6.7 で定義されています。

2.3.2 パッケージのバージョンについて

全てのパッケージはコントロールファイルの Version フィールドに書かれている必要があります。バージョンについては Debian Policy の 5.6.12 節で説明されているので、次回あたりで詳細を説明します。

日付によるバージョン番号のつけ方についても決められています。これはスナップショットでリリースされているアプリケーションにバージョン番号をつけるときに使用します。YYYYMMDD の形式でバージョン番号を付けるようにすべきであるると書かれています。

2.3.3 パッケージのメンテナについて

ここでは、パッケージメンテナについて示されています。

全てのパッケージには必ず一人以上のメンテナを持たないといけなく、連絡可能なメールアドレスを持たなければなりません。グループでメンテナンスすることも可能ですが、この場合でも共通の一つのメールアドレスを持つ必要があります。

メンテナは control ファイルの Maintainer フィールドに正しい名前と連絡可能なメールアドレスを指定します。メンテナが複数のパッケージをメンテナンスしているときは、パッケージ毎に異なった名前やメールアドレスを使うことはやめて、同じものを使うことが推奨されています。

メンテナがパッケージをメンテナンスすることを辞めた場合、他のメンテナがみつかるまで Debian QA グループがメンテナンスを引き継ぎます。このようなパッケージは orphaned （みなしご）パッケージと呼ばれます。

2.3.4 パッケージの説明について

ここではパッケージの説明文について示されています。

全てのパッケージには control ファイルの description フィールドに説明文が記入されている必要があります。簡単なパッケージの説明を記入するラインは半角 80 文字以内である必要があります。詳細な説明文を記入するエリアがあり、ここは上の簡単なパッケージの説明文とわけて書く必要があります。内容はパッケージが何をするか、Debian システムにどのような機能を追加するのかを書くべきであると示されています。

ソフトウェアのオフィシャルサイトに書いてある説明文をそのまま書くと、わかりにくいと BTS が来るときもあるのでよく考えて書きましょう。

2.3.5 依存関係について

ここにはパッケージの依存関係について示されています。

全てのパッケージはパッケージそれぞれが正常に動作するために必要なパッケージパッケージを依存情報として指定されている必要があります。

動作に必要なライブラリなどを依存情報として指定しておかないと、プログラムをインストールしても正常に動作しないからです。例外もあって、Essential が指定されているパッケージは依存情報に指定する必要はありません。(2.8 で説明します。)

また、あるパッケージが、それをインストールする際に別のパッケージがインストールされ、且つ設定されている必要があるときがあります。この場合、そのパッケージには Pre-Depends フィールドに指定する必要があります。

例えば、coreutils パッケージがインストールされる場合に、libacl1 パッケージや libc6 パッケージがインストールされている必要があるので、coreutils の control ファイルの Pre-Depends フィールドには libacl1 や libc6 が指定されています。

この Pre-Depends は勝手に設定していいものではなく、`debian-devel@list.debian.org` でその設定が正しいものなのか、必要なものなのか議論して決めるべきですと書かれています。

2.3.6 バーチャル（仮想）パッケージについて

同じような機能を持つパッケージを仮想パッケージとして定義する方法について示されています。例えば、httpd の機能を持ったパッケージは apatche や boa, lighttpd などがあります。これらのパッケージで提供される機能は同じようなものであり、これらをまとめて、仮想パッケージとして定義することで想定できるパッケージをずらずら書かなくても良くなります。

例えば、これらの機能を必要とするパッケージ、例えば viewcvs などは httpd の機能が必要なのですが、コントロールファイルの Depands フィールドに httpd と書くだけでよくなります。

仮想パッケージは物理的には存在せず、論理的に存在します。例えば、apt-get install httpd と実行すると httpd を仮想パッケージとして指定している (コントロールファイルの Provides フィールドで指定) パッケージがずらずらと表示されます。

仮想パッケージ名は勝手に決めてはいけません。`debian-devel@list.debian.org` で議論する必要があると思います。現在指定可能な仮想パッケージ名は `/usr/share/doc/debian-policy/virtual-pac` に書かれています。

2.3.7 ベースシステムについて

Debian のベースシステムについて示されています。Debian のベースシステムは Debian GNU/Linux システムとして最小のパッケージで構成されています。これらのパッケージのほとんどはプライリティが required か important で、Essential が指定されています。また、Section フィールドに base が指定されています。

勝手に Section フィールドに base を指定してはいけません。`debian-devel@list.debian.org` で議論して同意を得る必要があります。

2.3.8 エッセンシャルパッケージについて

エッセンシャルパッケージのポリシーについて示されています。エッセンシャルパッケージとは Debian GNU/Linux システムとして必要不可欠なパッケージのことを指します。Essential が指定されているパッケージは control フィールドに `Essential: yes` が指定されており、Debian

のシステムとして必ず必要なパッケージであることを示します。

勝手にパッケージ Essential を指定してはいけません。debian-devel@list.debian.org で議論して同意を得る必要があります。

2.3.9 メンテナスクリプトについて

ここでいうメンテナスクリプトとは、パッケージのインストールの際に実行されるスクリプトのことを指します。debconf を使ったり、オリジナルのスクリプトを使ったユーザーへのデータ入力方法や制限が示されています。

インストールする際に毎回設定をユーザーに対して質問するのではなく、設定ファイルをうまく用いて行うよう努力するべきであると書かれています。また、設定ファイルは/etc の適切な場所に置く必要があり、このことについてのドキュメントも書く必要があると書かれています。質問を行うためのプログラムを呼び出すスクリプトも postinst か config にすべきであり、インストールに失敗したときにも適切な処理が行われるように設計されている必要があると書かれてます。

3 Debian Policy 入門 第 3 回

岩松

Debian Policy 第 3 回です。今回は Source package についてです。

3.1 ソースパッケージとは？

ソースパッケージは Debian が配布しているバイナリパッケージの元になっているパッケージのことです。例えば、シェルスクリプトの **bash**[*2] は bash[*3]というソースパッケージからビルドされます。しかし、bash ソースパッケージは bash バイナリパッケージを作成するだけでなく、bash-builtins[*4]パッケージや bash-doc[*5]パッケージもビルドされます。一つのソースパッケージから複数のソースパッケージがビルドされるとがあるということです。

3.2 Standards-Version について

Standards-Version は Debian Policy のバージョンを指します。Debian Policy は常に更新されており、現在、バージョンは 3.6.2.2 です。ソースパッケージは常に最新の Debian-Policy に追従すべきであると書かれています。実際にはパッケージをアップデートしたときに、Debian Policy のバージョンをチェックし上がっていた時、 Standards-Version の追従してバージョンを上げます。

Standards-Version は debian/control ファイルの Standards-Version フィールドに記述します。Standards-Version フィールドのフォーマットもポリシーで決められており、セクション 5.6.11 で説明されています。

3.3 パッケージ関係について

パッケージをビルドする際に必要なパッケージが出てきます。そのビルドに必要なパッケージを指定する必要があると書かれています。

必要なパッケージを全て書くわけではなく、最低限必要なパッケージを書くべきであると書かれており、例えば、bash を例にすると、ビルドの依存関係は以下のようになっています。

```
Build-Depends: autoconf, patch, bison, libncurses5-dev, texinfo, autotools-dev, debhelpe
texi2html, locales
```

libncurses5-dev に注目して、libncurses5-dev[*6]の依存関係を見てみると、

```
Build-Depends: debhelper (>= 3.0.23), libc6-dev-sparc64 [sparc], libc6-dev-s390x [s390]
```

[*2] http://packages.debian.org/unstable/shells/bash
[*3] http://packages.qa.debian.org/b/bash.html
[*4] http://packages.debian.org/unstable/utils/bash-builtins
[*5] http://packages.debian.org/unstable/doc/bash-doc
[*6] ソースパッケージは ncurses

```
libc6-dev-amd64 [i386], libc6-dev-ppc64 [powerpc], lib64gcc1 [i386 powerpc sparc s390],
libgpmg1-dev (>= 1.19.6-20) [!hurd-i386 !kfreebsd-i386], quilt (>= 0.40-1)
```

となっています。依存しているパッケージに依存しているパッケージはもともと依存しているので、書く必要がないということです。

パッケージ間の依存の詳細に関しては セクション 7 で説明されています。

3.4 アップストリームのソース変更について

Debian では Debian social contract に書かれているように、Debian で発生した不具合やパッチをアップストリームに還元するようにしています。アップストリームとは上流開発者のことで、パッケージの開発元を指します。Debian 特有の問題やビルド時における最適化等で修正を入れるときがあります。ビルド前のテストで Debian として追加したい項目があるときは autoconf を使って適切に処理したり、Makefile を修正するときは、Makefile を直接修正せずに、Makefile.in を修正するようにとも書かれています。これは configure を行ったときに Makefile が上書きされてしまうからです。

3.5 Debian changelog について

Debian changelog とは Debian パッケージに関する変更点について書かれたものです。アップストリームの変更とは別書く必要があり、debian/changelog ファイルに記述します。

ポリシーとして、debian/changelog に Debian パッケージによる変更点を簡潔に記述すべきであると書かれています。debian/changelog を修正するときは dch[*7]を使うと便利です。

Debian changelog の役目はこれだけではなく、debian/changelog からパッケージのバージョン情報を取得し、パッケージ構築の際に使用します。形式は以下のようになります。

```
package (version) distribution(s); urgency=urgency
     [optional blank line(s), stripped]
  * change details
    more change details
     [blank line(s), included in output of dpkg-parsechangelog]
  * even more change details
     [optional blank line(s), stripped]
 -- maintainer name <email address>[two spaces]  date
```

- package , version
  ソースパッケージ名とソースパッケージのバージョンを指します。
- distribution
  version で指定されたパッケージがインストールされるディストリビューションを指します。Distribution に関しては Section 5.6.14. で説明されています。
- urgency
  パッケージをアップロードする際の緊急度を指定します。low, medium, high ,emergency を指定することができます。

[*7] http://packages.debian.org/unstable/devel/devscripts

- コメント部
  コメントに関しては先頭は 2 つのスペースが必要です。習慣で各変更内容の先頭はアスタリスクになっています。長い文章は改行するのですが、改行したときは字下げを行います。字下げは上のテキストに沿って行います。空改行は変更内容をわけるために使用したりします。変更内容に不具合の修正内容を書くときがあります。このとき、BTS[*8]に登録されている場合があります。バグの番号をフォーマット通りに changelog に書くことによって、changes ファイルに書き込まれ、パッケージがアップロードされたときに、自動的にバグが close されます。フォーマットは #nnnnnn です。
- maintainer name , email address changelog を書く際にメンテナ名とメールアドレスを記述します。この項目はパッケージがアップロードされた時の承認結果を送る際に使用されます。また、パッケージのキーサインにもこの項目が使われます。
- date 修正した日時を書きます。RFC822 フォーマットに基づいて書く必要があります。
- タイトルタイトル部は左から始まります。メンテナーの前はスペースを入れ、トレーサー (–) を入れる必要があります。また、メンテナと日付の間には 2 つスペースを入れ、分ける必要があります。

changelog がインストールされる場所はセクション 12.7 に説明されています。

また、代替の Changelog フォーマットを使うことができます。実験用ではないパッケージでは、dpkg の最新バージョンでサポートされる debian/changelog のためのフォーマットを使用しなければなりません。自分が使用したいフォーマットがあるなら、パーサーを提供することによって変更することができます。パーサーは dpkg-genchanges および dpkg-gencontrol によって期待された API 互換性を持つ必要があります。

3.6 Error trapping in makefiles

Makefile からシェルスクリプトが呼ばれるときがあります。例えば、dpatch [*9]によって呼ばれる patch ファイルです。Makefile 内でシェルスクリプトファイルがエラーが発生しても、エラーを捉えることができません。そのため、シェルスクリプトファイルは実行の際に -e オプション[*10]を付けなければならないと説明されています。

3.7 タイムスタンプについて

可能な限りアップストリームのソースファイルのタイムスタンプをパッケージ中に変更せず、そのままにしておくことを推奨すると説明されています。

3.8 ソースパッケージの中のオブジェクトファイルにおける制限

ソースパッケージの中にはハードリンク、デバイスファイル、ソケット、setuid や getuid されたファイルを入れてはいけません。

[*8] http://bugs.debian.org
[*9] http://packages.debian.org/unstable/devel/dpatch
[*10] ERR トラップが設定されていればそれを実行して終了します。

3.9 Main building script: debian/rules

deban/rules ファイルは ソースパッケージからバイナリパッケージを作成する方法がスクリプトです。実態は実行可能な (パーミッション:755)makefile です。ファイルの先頭は **#!/usr/bin/make -f** になっています。

スクリプトは非対話式になっています。対話式だと、毎回同じバイナリが生成されるとは限らないので、自動的にバイナリパッケージが生成されるようになっています。スクリプトの内容は dpkg-buildpackage から呼ばれる必要なターゲットとして clean, binary, binary-arch, binary-indep, build があり、これらが最小の構成になっています。

- build
  パッケージの設定、コンパイルを行います。もし、パッケージ構築前に設定作業がある場合は、Debian 化されたソースの設定作業を行った後で行うべきであると書かれています。その理由として設定を再度行わず、パッケージの構築が行えるようにするためです。
  いくつかのパッケージは同じソースパッケージからコンパイルのやり方を変更して異なったバイナリを生成する場合があります。build ターゲットではこのような処理には対応できないので、それぞれの構築方法に従って、それぞれのターゲット（例えば、binary-a と binary-b）を作成して使用するといいと書かれています。この場合は実際は build ターゲットではなにも行わず、binary ターゲットでそれぞれのパッケージをビルドしてそれぞれのバイナリパッケージを作成することになります。
  ルート権限が必要な作業は行ってはいけません。
- build-arch (optional), build-indep (optional)
  build-arch は、提供された場合、アーキテクチャーに依存しているバイナリパッケージ (debian/control ファイルの Architecture フィールドが”all” ではないとき）すべて生成するために必要になった設定やコンパイルをすべて行なうべきです。build-indep は アーキテクチャーから独立しているバイナリパッケージ (debian/control ファイルの Architecture フィールドが”all” のとき）すべて生成するために必要になった設定やコンパイルをすべて行なうべきです。build ターゲットは、rules ファイルの中で提供される build-arch および build-indep に依存するべきです。
- binary, binary-arch, binary-indep
  binary ターゲットはこれだけで、バイナリパッケージを構築できないといけません。binary ターゲットは 2 種類に分けられ、binary-arch は特定のアーキテクチャ用のファイル、binary-indep はそれ以外のファイルを生成します。これらのターゲットは非対話的に動作するものでなければいけません。
- clean
  build ターゲットと binary ターゲットによって生成されたファイルを削除し、元に戻します。例外として、binary ターゲットで出力されたファイルは消さず、残します。このターゲットは非対話的である必要があります。
- get-orig-source (optional)
  このターゲットは主要なアーカイブサイト（例えば、リングサーバー？）から最新のオリジナルソースを HTTP や FTP から取得します。取得したオリジナルソースを tar ファイルに再構成します。

build , binary および clean ターゲットはパッケージのトップディレクトリをカレントディレクトリとして実行されなければなりません。

公開されている、またはいないインターフェイスのためやパッケージ内部で使用するために debian/rules に他のターゲットを置くことは許されます。

パッケージを実際に構築するマシンやインストールの対象となるマシンのアーキテクチャは、dpkg-architecture を使い、変数を指定することによって決定されます。これにより、ホストマシンだけでなくパッケーの構築するマシンの Debian 形式のアーキテクチャーと GNU 形式のアーキテクチャ指定文字列を取得するとができます。

- DEB_BUILD_ARCH
  Debian 形式のパッケージ構築マシンアーキテクチャ
  例：i386
- DEB_HOST_ARCH
  Debian 形式のインストール先アーキテクチャ
  例：i386
- DEB_BUILD_GNU_TYPE
  GNU 形式のパッケージ構築マシンアーキテクチャ指定文字列
  例：i486-linux-gnu
- DEB_HOST_GNU_TYPE
  GNU 形式のインストール先アーキテクチャ指定文字列
  例：i486-linux-gnu
- DEB_BUILD_GNU_CPU
  DEB_BUILD_GNU_TYPE の CPU 部分
  例：i486
- DEB_HOST_GNU_CPU
  DEB_HOST_GNU_TYPE の CPU 部分
  例：i486
- DEB_BUILD_GNU_SYSTEM
  DEB_BUILD_GNU_TPE のシステム部分
  例：linux-gnu
- DEB_HOST_GNU_SYSYTEM
  DEB_HOST_GNU_TYPE のシステム部
  例：linux-gnu

DEB_BUILD_ARCH および DEB_HOST_ARCH は Debian アーキテクチャのみを決定することができます。実際の CPU やシステム情報を取得する際はこれらを使用してはいけません。この場合には GNU 形式の変数を使用しなくてはいけません。

3.10 Variable substitutions: debian/substvars

substvars ファイルはそのパッケージの実行ファイルに関する共有ライブラリの依存関係を計算し、書き出されたものです。bash を例に取ると、内容は以下のようになっています。

```
shlibs:Pre-Depends=libc6 (>= 2.3.5-1), libncurses5 (>= 5.4-5)
```

このファイルは debian/rules によって生成され、動的に変更されます。clean ターゲットで削除されるようにしておく必要があります。実際には dpkg-gencontrol , dpkg-genchanges, dpkg-source が control ファイルを生成するときに substvars を参照してファイルを生成します。substvars を使ったソースの変換方法については、dpkg-source の man に書かれています。

3.11 Generated files list: debian/files

このファイルはソースツリーの常に存在する部分ではありません。これはどのようなパッケージが生成されたのか記録するために用いられます。dpkg-genchanges は、.change ファイルを生成する際に使用します。bash を例に取ると、以下のような内容になっています。

```
bash-doc_3.1-4_all.deb doc optional
bash_3.1-4_i386.deb shells required
bash-builtins_3.1-4_i386.deb utils optional
bash-static_3.1-4_i386.deb shells optional
bash-minimal_3.1-4_i386.deb shells optional
```

また、このファイルはアップロードされるソースパッケージには含めてはならず、debian/rules の clean ターゲットで削除すべきであると書かれています。

4 Debian multimedia project

上川

Debian には Multimedia Project というサブプロジェクトがあります。そこでは、Multimedia 関連のツールについての議論や調整が行われています。そこで議論されている内容は大きく、ビデオとオーディオに分割できます。その内の、上川が担当している、オーディオ関連について現状何がなされていて、何ができるようになっているのかを説明します。

4.1 AGNULA/DeMuDi

Debian Multimedia Distribution というプロジェクトがあります。これは Debian に、リアルタイムカーネルを追加し、いくつかのパッケージをカスタマイズして作成したものです。Debian 用語でいう、「CDD:Custom Debian Distribution」の一つで、インストール直後からオーディオアプリケーションが使える便利なディストリビューションです。

Debian 本体とパッケージ自体はあまりかわりませんが、一部のパッケージが追加されております。

4.2 Debian multimedia policy

multimedia 関連のツールを利用する際には、複数のツールを相互に作用させる必要があり、相互作用のための規格は Debian 外部で活発に議論されていました。たとえば linux-audio-dev メーリングリスト周辺では音楽関連のセッション管理や相互通信のための規格が議論されています。

Debian に関しては過去それぞれのアプリケーションが独立しており、まだポリシーを策定できるような状況ではありませんでした。ただ、最近はいろいろとツールも出そろって来た感があるため、そろそろ相互運用を考えたポリシーの策定が必要になって来ています。

ただ、多くのソフトウェアが準拠できないような高いハードルを設定しても、各パッケージが従わないだけなので、現状準拠することに議論が出なさそうな部分から標準ポリシーとして策定していこうと画策しています。

一番策定したいのは、プラグイン機構、相互接続、デフォルトのオーディオと MIDI デバイスの指定の部分について Debian で統一した操作感を提供する部分です。

4.3 今 Debian でできること

Debian を Digital Audio Workstation (DAW) のプラットフォームとして音楽活動をしようとすると何ができるのか、何ができないのか、追求してみようと思います。

現在存在しているアプリケーションはおおまかに、下記に分類できます。

- フレームワーク系、相互運用のために必要な基本的なドライバなど。ALSA や jack、ladspa など。
- MIDI(音譜) 編集系

- マルチトラック/音声編集系
  音声データを受け取り、録音し、編集し、音声データを出力するもの
- ソフトウェアシンセサイザー
  MIDI を入力として受け取り、音声を出力するもの
- エフェクト
  LADSPA プラグイン。

4.4 MIDI コネクション

音符情報を扱うための統一企画として、MIDI があります。MIDI は、音程と音量を指定して発声を指示するための通信プロトコルで、昔から使われているものです。現在の電子楽器などではほとんど利用できます。

また、ALSA の MIDI シーケンサ機能を利用すると、仮想的にアプリケーションとアプリケーションを接続して、情報の通信プロトコルとして MIDI を利用することができます。

GUI で操作する場合は、qjackctl などで制御します。

4.5 JACK で接続する

発声と録音の経路について、UNIX らしく、各アプリケーションがそれぞれの役割をもつ、という場面を考えてみると、MIDI のような通信プロトコルが必要になります。

jack はそのプロトコルを提供します。

オーディオアプリケーションは jackd というデーモンを介して相互に音声データを送受信します。また、リアルタイム (録音しながらそれを再生しながら処理) に処理をするために工夫がこらされています。

4.6 LADSPA:エフェクトをかける

音声を扱う各アプリケーションはそれぞれ音声に対してエフェクトをかけることができます。一般的にはディレイや、ディストーション、コンプレッサーなどがあります。それらを、各アプリケーション毎に再実装するのは無駄なため、共有化しようということで生まれたのが LADSPA という規格です。

LADSPA という規格にのったプラグインがそれぞれのエフェクト提供し、各アプリケーションがそれを利用してエフェクトをかけます。

現状 LADSPA エフェクトを利用したり提供できるパッケージを確認するには、ladspa-plugin を Provides: していたり、Recommends:しているパッケージを確認すればよいです。[*11]

swh-plugin などはよいエフェクトだと定評があります。

[*11] 今気づいたのですが、現状その依存関係をもっていないパッケージがあるため、現実より少ない数のパッケージしか出て来ません。

```
 > apt-cache showpkg ladspa-plugin
Package: ladspa-plugin
Versions:

Reverse Depends:
  sweep,ladspa-plugin
  ecasound,ladspa-plugin
  xmms-ladspa,ladspa-plugin
  terminatorx,ladspa-plugin
  sweep,ladspa-plugin
  snd,ladspa-plugin
  rosegarden4,ladspa-plugin
  glame,ladspa-plugin
  galan,ladspa-plugin
  ecasound,ladspa-plugin
  audacity,ladspa-plugin
Dependencies:
Provides:
Reverse Provides:
tap-plugins 0.7.0-2
swh-plugins 0.4.14-1
ladspa-sdk 1.1-4
cmt 1.15-3
caps 0.3.0-1
blop 0.2.8-3
```

4.7 Linux オーディオ処理におけるリアルタイムの必要性

オーディオデータを入力したものを加工して再生すると、その間に処理遅延が生じます。処理遅延のバッファとして 1024 フレーム利用するとしてみましょう。通常の 44.100kHz のオーディオデータで 1024 フレームというと、23 ミリ秒程度です。これは、23 ミリ秒分のデータを取得して、jackd に接続している全プロセスがそのバッファに関連した必要な処理をして、23 ミリ秒以内に出力用バッファに配置するということを意味します。23 ミリ秒以内に配置できなかった場合にはその回の音声は途切れ、ユーザはブチという音を効くことになります。この状態を ALSA 用語では「xrun」と呼びます。[*12]

実際問題として、Linux をそのまま利用していると、23 ミリ秒以内に絶対に全プロセスが処理を完了するということは難しいです。

最近の Linux カーネルは下記の追加で改善してきてはいますが、まだまだ進歩が求められています。特に、ライブやレコーディングでは、数時間にわたって jack の処理が時間内に終了しないということが一度でもあってはならないという条件になるため、それなりにチューニングの手腕が求められます。

- カーネルが持っている長時間の処理を分割
- preempt 対応により、カーネル空間の処理でもタイムスライスにより CPU を明け渡すようになった
- リアルタイム対応により、SCHED FIFO などのスケジューリングに対応、オーディオ関連のスレッドの優先度をあげることができるようになった

[*12] 実際は period 数を増やすことでダブルバッファリングみたいなことをしているため、この限りではないようだが、簡単に原理だけは伝わっただろうか。

5 Debian Multimedia Audio application 概観

上川

現在 Debian に存在しているアプリケーションにどのようなものがあるかみてみましょう。ここにあるリストは全てを網羅しているわけではなく、また、途中であきらかに力尽きてます。今後また続きをやります。

5.1 フレームワーク系

Debian では音楽関連のフレームワーク系も独自に管理しています。この関連について議論する場所は debian-multimedia@lists.debian.org メーリングリストです。

5.1.1 ALSA

Linux のオーディオの次期標準といわれつづけて早何年目か。カーネルモジュール (最近は標準) とユーザランドのライブラリ (libasound2) といくつかのツールがあります。

```
$ aplay -l
**** ハードウェアデバイス PLAYBACK のリスト ****
カード 0: IXP [ATI IXP], デバイス 0: ATI IXP AC97 [ATI IXP AC97]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
カード 0: IXP [ATI IXP], デバイス 1: ATI IXP IEC958 [ATI IXP IEC958 (AC97)]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
カード 2: Device [KC USB Audio Device], デバイス 0: USB Audio [USB Audio]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
```

5.1.2 jack-audio-connection-kit

各音楽関連のアプリケーションが利用する音声経路ルーティングプロトコルです。jackd というデーモンが、利用しているユーザの権限で起動し、それを経由して通信します。

コマンドラインで起動する場合は

`jackd -d alsa -d` デバイス名 `-r` サンプルレート

のように指定します。

```
jackd -d alsa -d ixp -r 48000
```

ポートの接続はコマンドラインからでも操作できます。

jackd には複数の ALSA サウンドカードを同時に使えないという制限があります。そういう場合は現状としては、ecasound などの jack と ALSA 対応のアプリケーションをかましてしのいでいます。ALSA 側の機能で対応することもできるようです。

```
$ jack_lsp
 alsa_pcm:capture_1
 alsa_pcm:capture_2
 alsa_pcm:playback_1
 alsa_pcm:playback_2
$ ecasound -i alsaplugin,2,0,0 -o jack_generic,usbaudio &
$ jack_lsp
alsa_pcm:capture_1
alsa_pcm:capture_2
alsa_pcm:playback_1
alsa_pcm:playback_2
alsa_pcm:playback_3
alsa_pcm:playback_4
alsa_pcm:playback_5
alsa_pcm:playback_6
ecasound:usbaudio_1
ecasound:usbaudio_2
$ jack_connect ecasound:usbaudio_1 alsa_pcm:playback_1
$ jack_connect ecasound:usbaudio_2 alsa_pcm:playback_2
```

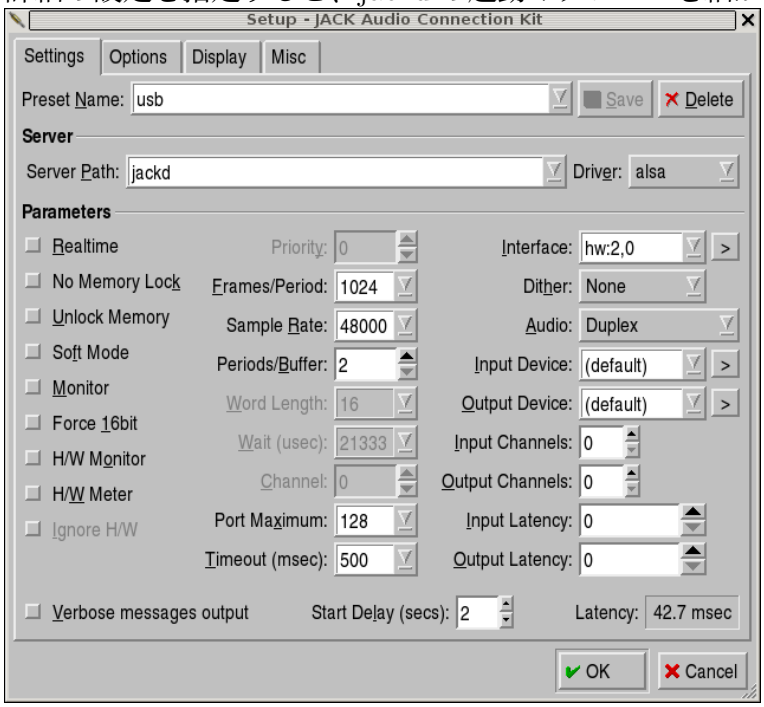
類似したシステムとして esound[*13]や arts[*14]、NAS[*15]がありますが、それらは音をできるだけ切らせないことを主眼としているため、余裕をもったバッファを利用しており、レイテンシの問題でほとんどのオーディオアプリケーションでは利用できないです。

5.1.3 qjackctl

ポートの接続や、jackd の起動/停止は、qjackctl で GUI 経由で操作できます。

まず、起動したら、パネルが起動します。

詳細の設定を指定すると、jackd の起動オプションを細かく指定できます。

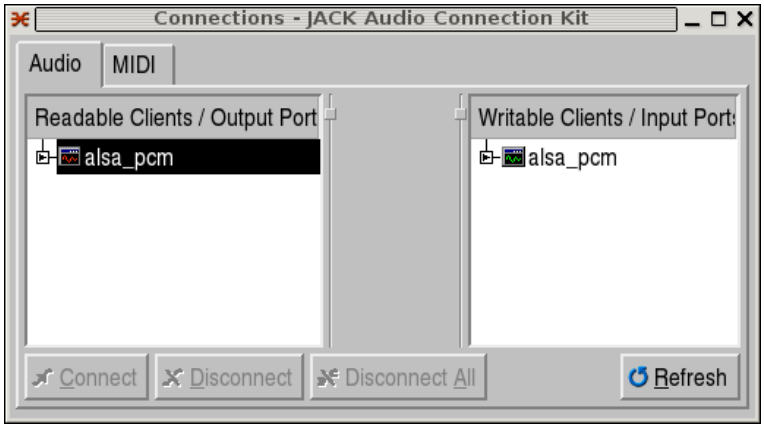


また、コネクションパネルを開くと、jack 接続の管理が出来ます。

[*13] GNOME

[*14] KDE

[*15] ネットワーク経由を主目的としたオーディオプロトコル



5.1.4 ladspa

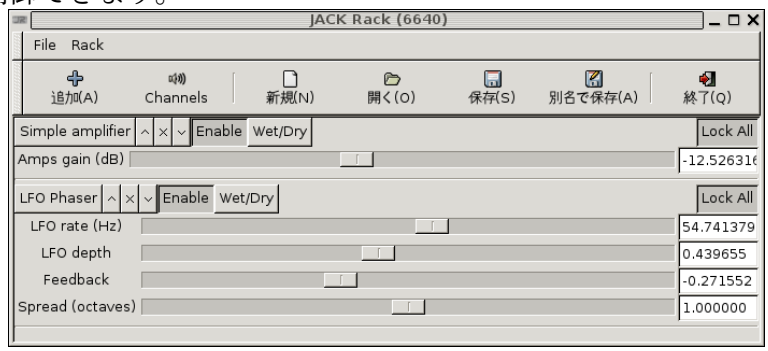
オーディオのエフェクトを処理するための、プラグインインタフェースです。また、ladspa-dev パッケージが存在しており、そのパッケージに含まれている/usr/include/ladspa.h を利用することが推奨されています。LADSPA 自体がポリシーを定義していますが、Debian の ladspa パッケージは、追加で `/usr/share/doc/ladspa-sdk/README.Debian` にて定義されている下記のポリシーにしたがっています。

`/usr/lib/ladspa/` にパッケージが提供する LADSPA プラグインを提供すること。`LADSPA\_PATH` 環境変数が定義されていない場合には、`/usr/local/lib/ladspa:/usr/lib/ladspa` をデフォルトの検索パスとして利用すること。

5.1.5 jack-rack

apt-get install jack-rack でインストールできます。

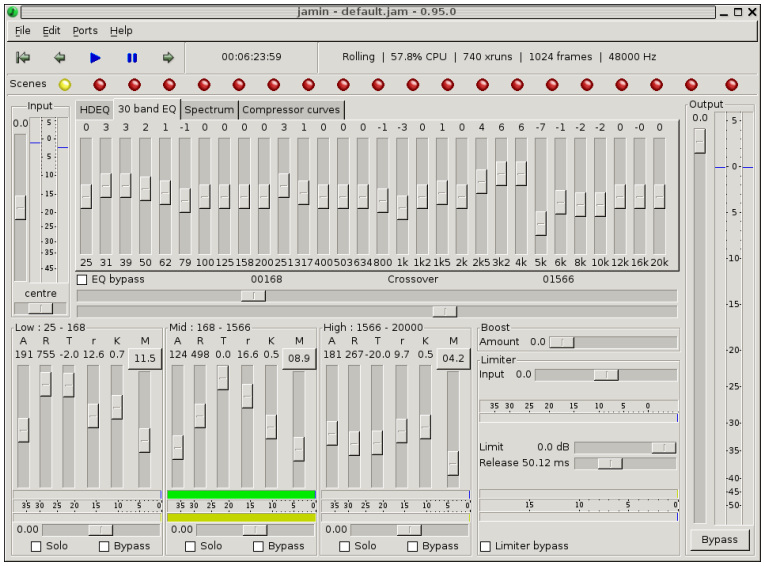
jack 接続経由で LADSPA エフェクトをかけることができ、エフェクトのパラメータを GUI で制御できます。



5.1.6 jamin

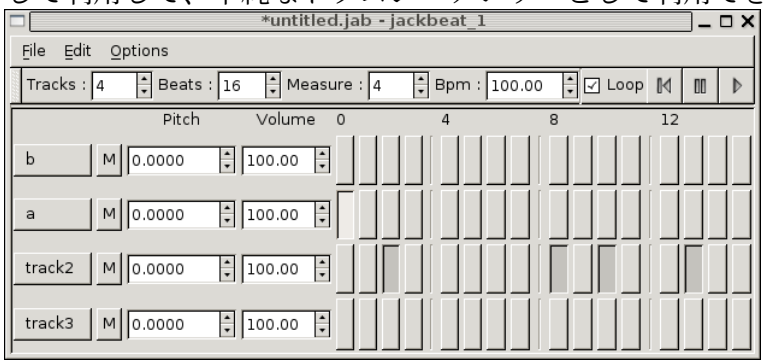
apt-get install jamin でインストールできます。

jack 経由で細かいイコライザーやコンプレッサーの設定が出来るツールです。レコーディングの最終段階のファイナライズに便利そうです。



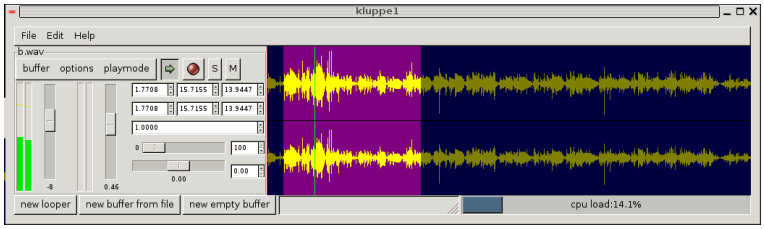
5.1.7 jackbeat

apt-get install jackbeat でインストールできます。wav ファイルをリズムループ用のサンプルとして利用して、単純なドラムシーケンサーとして利用できるようです。



5.1.8 kluppe

apt-get install kluppe でインストールできます。jack 経由で接続し、wav ファイルをループさせることができます。



5.1.9 ladcca

LADCCA というフレームワークが存在しているようです。気づいたら lash `http://www.nongnu.org/lash/` というプロジェクトにかわってしまっているようです。セッション管理のためのフレームワークです。jack の導入にともない、アプリケーションが接続できるのはよいのですが、毎回アプリケーションをユーザが接続する必要があります。その処理を簡便化するためのプロトコルのようです。

5.2 ソフトウェアシンセ

MIDI の接続は ALSA の MIDI 接続が事実上の標準プロトコルとして利用されています。qjackctl の接続画面に MIDI 接続タブがあるので、それを利用して接続してあげればよいです。また、aconnect というコマンドラインインタフェースがあり、それを利用することも可能です。

```
$ aconnect -i -l
クライアント 0: 'System' [タイプ=カーネル]
    0 'Timer           '
    1 'Announce        '
        接続先: 15:0, 128:0
クライアント 14: 'Midi Through' [タイプ=カーネル]
    0 'Midi Through Port-0'
クライアント 130: 'Virtual Keyboard' [タイプ=ユーザ]
    0 'Virtual Keyboard'
        接続先: 129:0
$ aconnect -o -l
クライアント 14: 'Midi Through' [タイプ=カーネル]
    0 'Midi Through Port-0'
クライアント 129: 'FLUID Synth (7910)' [タイプ=ユーザ]
    0 'Synth input port (7910:0)'
        接続元: 130:0
```

5.2.1 TSE3

シーケンサエンジンのようです。KDE 関連のアプリはこれを利用しているような気がしています。どうなのよ。

5.2.2 timidity

ちょっとインタフェースに古めかしい感がありますが、事実上の標準の MIDI シーケンサエンジンです。MIDI データから WAV を生成するためのインタフェースとして利用されています。

5.2.3 freepats

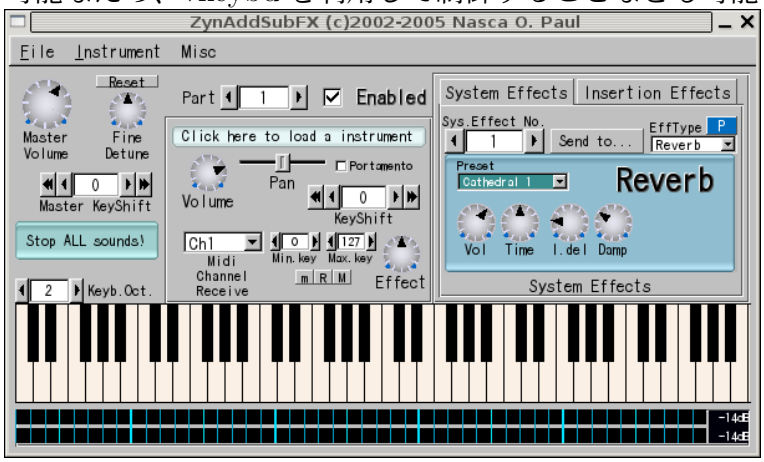
Debian にて、フリーのサウンドフォント集です。形式が pat 形式です。timidity から使える設定になっているようです。

sf2 形式じゃないとほとんどのアプリから利用できないので意味が無い。

5.2.4 zynaddsubfx

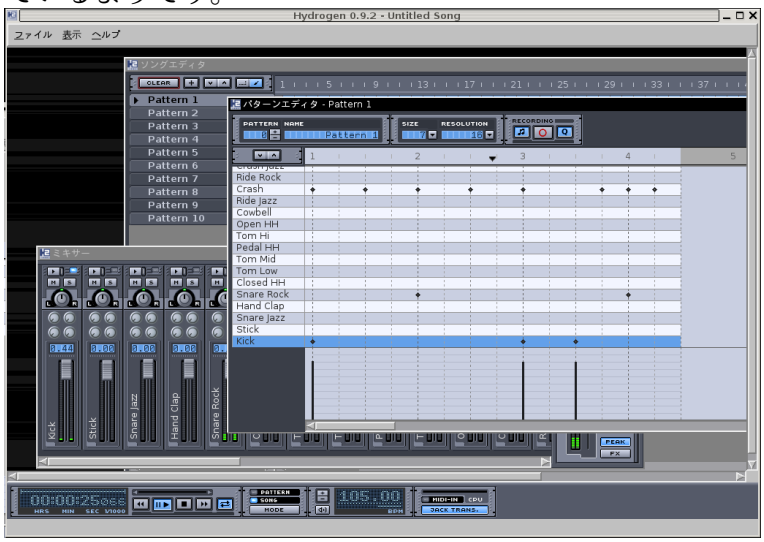
apt-get install zynaddsubfx でインストールできます。

オルガン系の音やパッド系の音が結構使えます。仮想キーボードの UI がお手軽です。MIDI 制御可能なため、vkeybd を利用して制御することなども可能です。jack 対応です。



5.2.5 hydrogen

UI が優秀なのでドラムシーケンサとして活用しています。jack 対応です。MIDI 入力にも対応しているようです。



5.2.6 pd

UI がかなり前時代的ですが、シンセを GUI で編集するという系では元祖みたいな存在です。使い方がわからんです。誰か教えてください。

5.2.7 beast

GTK シンセ。これも結構頑張っています。使い方がわからんです。誰か教えてください。

5.2.8 csound

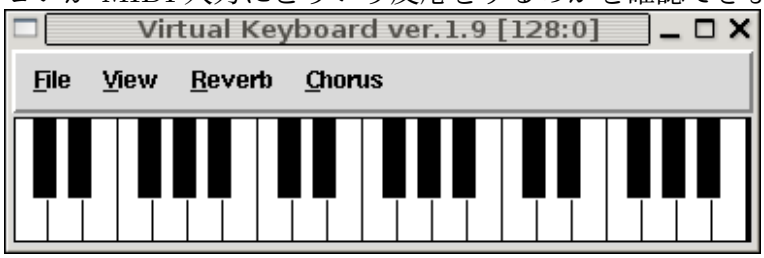
学術系の人々の中で長い間つかわれてきたものらしく、過去の遺産が大量にあります。ちょっと学術的すぎて個人的には使っていません。誰か使い方教えてください。

コマンドラインで操作できる、というよりむしろプログラム言語です。

方程式で波形を設計したいあなたに。

5.2.9 vkeybd

キーボードで操作できる MIDI キーボードです。alsa の MIDI デバイスとして動作します。とりあえず試すのには便利です。こいつを起動して、qjackctl で接続してあげれば他のアプリケーションが MIDI 入力にどういう反応をするのかを確認できます。



5.2.10 fluidsynth

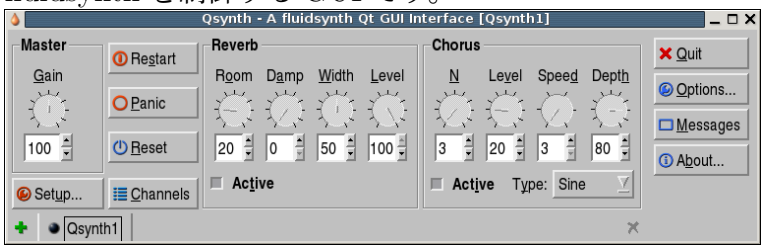
ソフトウェアシンセのようです。sf2 形式のファイルをサポートしているようです。`http://www.hammersound.net/` などに多数のサウンドフォントが存在していて、そのうちの適当な

ファイルを読み込んで利用することが出来ます。jack へ音声を出力することが可能です。

Debian 内で、sf2 のファイルが見付かりません。ウェブを探すとたくさんあるようです。

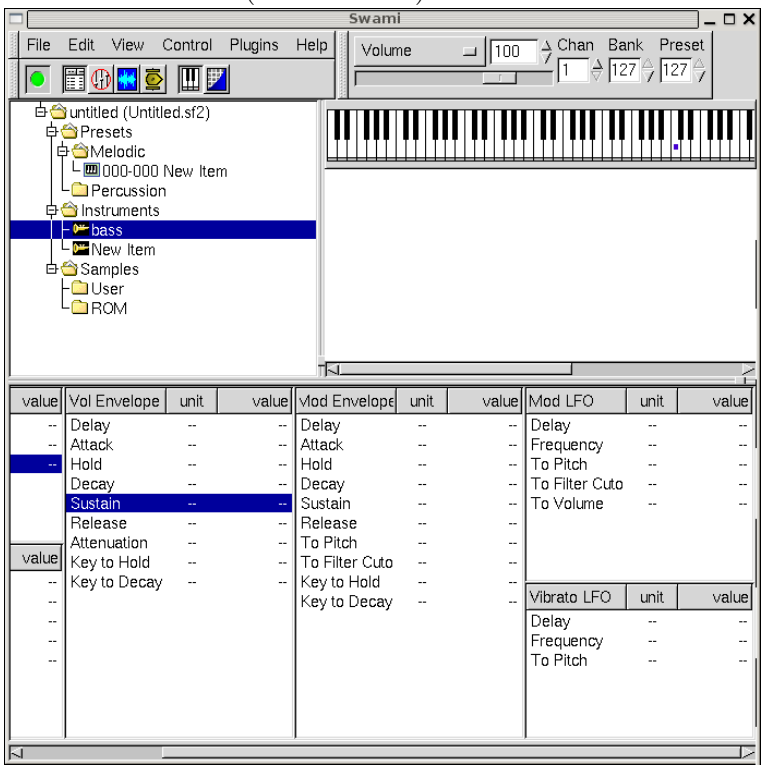
5.2.11 qsynth

fluidsynth を制御する GUI です。



5.2.12 swami

サウンドフォント (sf2 ファイル) を編集するツールです。fluidsynth を内部では利用しています。



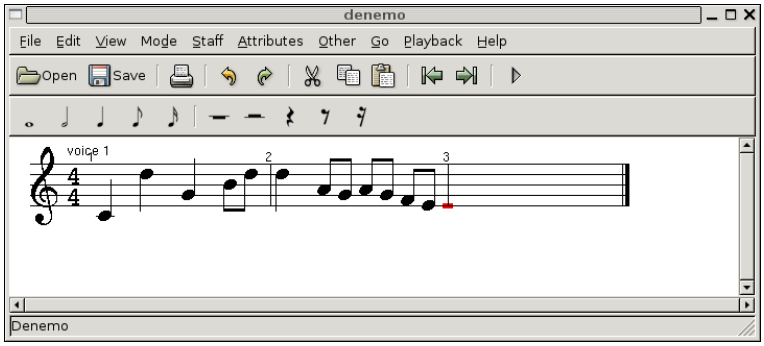
5.3 楽譜編集

5.3.1 lilypond

TEX で楽譜を作成しよう、というパッケージ。まともなクラシックの楽譜を作成するような作業をする際にはこれでやってました。TEX をがんがん使いたいあなたに。

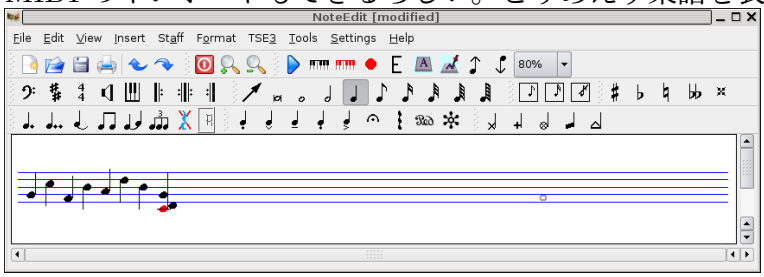
5.3.2 denemo

今までは上川はこれで一小節程度の楽譜ならこちょこちょっと作成して用を足して来ました。キーバインドも数字で音符の長さが決まっていたり、キーの上下で操作できます。久しぶりに見てみるとインタフェースが大幅に改善されているようです。



5.3.3 noteedit

MIDI のインポートもできるらしい。とりあえず楽譜を表示することはできるっぽい。

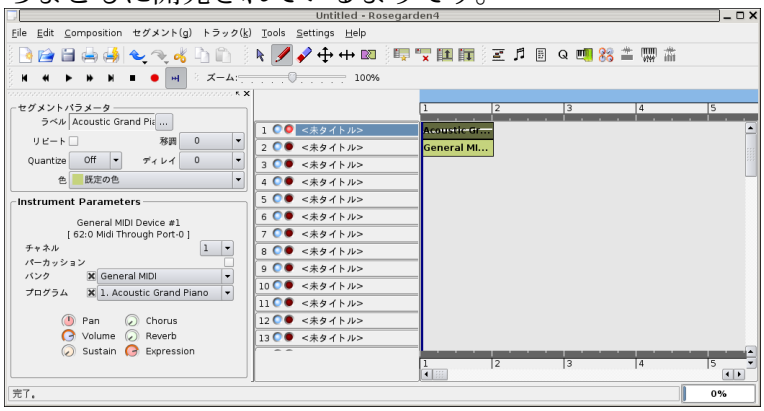


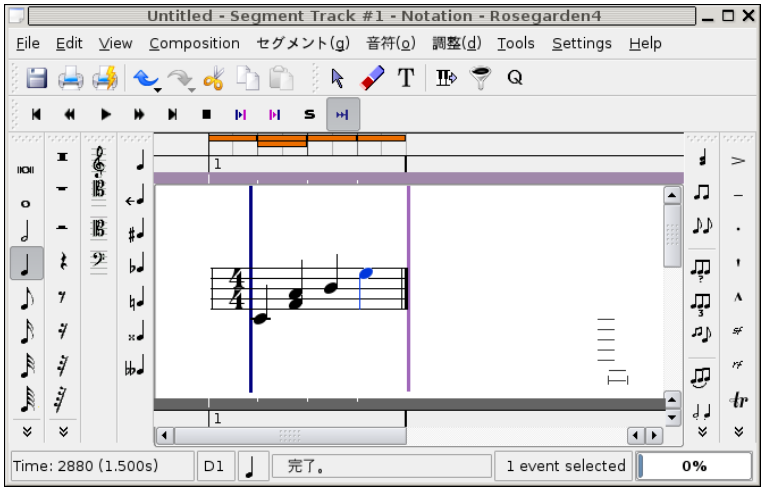
5.3.4 rosegarden4

apt-get install rosegarden4 でインストール。

一応楽譜が編集できます。ステップ録音などもできます。デバッグメッセージが大量に出て来るのとなんだか反応が鈍い感じはします。

rosegarden という古くから何度も書き直されつづけているプログラムの最新版です。いまのところまともに開発されているようです。

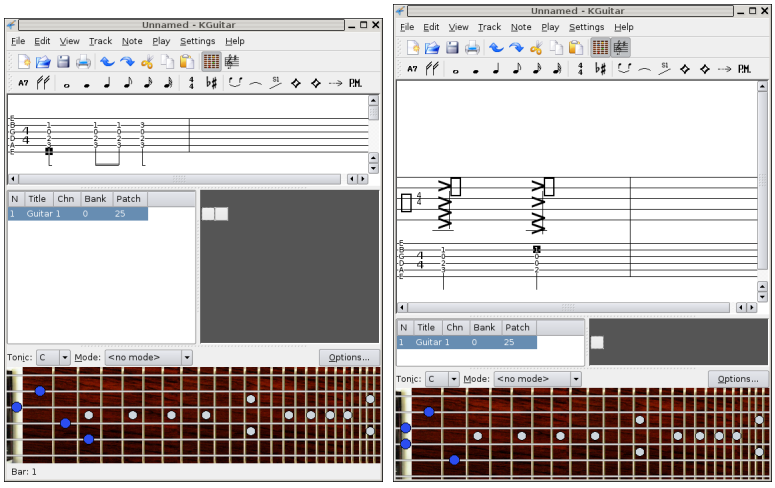




5.3.5 kguitar

apt-get install kguitar でインストール。

ギターのタブ譜を編集できるソフトウェアのようです。使い方が分からないので、困りました。楽譜が出るはずのようですが、出てません。ギターの絵が素敵です。MIDI 入出力ができることになっているようです。



5.4 音声編集系

5.4.1 ecasound

apt-get install ecasound でインストール。

コマンドラインベースで音声加工をするためのツールです。

よく使うコマンドは

音量をノーマライズする。(可能な最大の音量まであげる)

```
$ ecanormalize in.wav
```

in.wav にコンプレッサーエフェクトをかけて、out.wav を生成する。

```
$ ecasound -i in.wav -o out.wav -eca
```

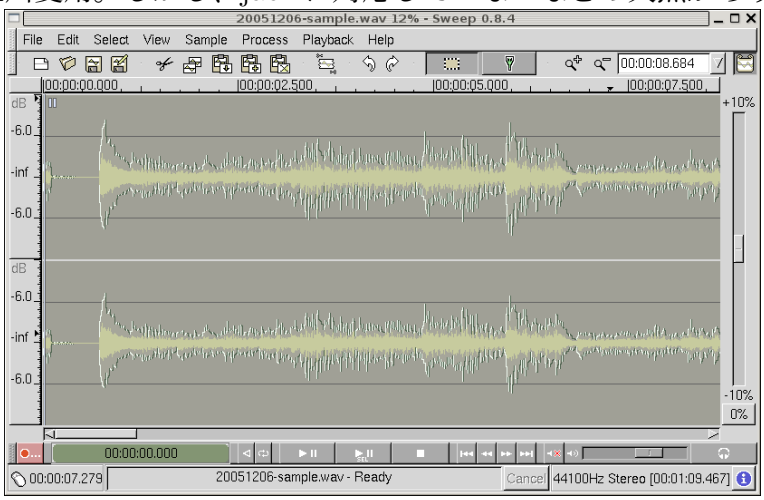
とりあえず録音する

```
$ ecasound -i alsahw,0,0,0 -o /tmp/a.wav
*******************************************************************************
*        ecasound v2.4.3 (C) 1997-2005 Kai Vehmanen and others
*******************************************************************************
- [ Session created ] -------------------------------------------------
- [ Chainsetup created (cmdline) ] ------------------------------------
- [ Connecting chainsetup ] -------------------------------------------
(eca-chainsetup) 'rt' buffering mode selected.
(eca-chainsetup) Audio object "alsahw", mode "read".
(audio-io) Format: s16_le, channels 2, srate 44100, interleaved.
(eca-chainsetup) Audio object "/tmp/a.wav", mode "read/write".
(audio-io) Format: s16_le, channels 2, srate 44100, interleaved.
- [ Chainsetup connected ] --------------------------------------------
(eca-control-objects) Connected chainsetup: "command-line-setup".
- [ Controller/Starting batch processing ] ----------------------------
- [ Engine init - Driver start ] --------------------------------------

ここで ctrl-C で停止

- [ Controller/Processing stopped (cond) ] ----------------------------
- [ Engine exiting ] --------------------------------------------------
(eca-control-objects) Disconnecting chainsetup: "command-line-setup".
- [ Chainsetup disconnected ] -----------------------------------------
- [ Controller/Batch processing finished ] ----------------------------
```

5.4.2 sweep

apt-get install sweep でインストール。

メモリ上に wav ファイルを展開するので、大きい wav ファイルは編集できないです。プリビューが優秀。ダブルクリックしたらそこから再生したりしてくれる。こまかい波形の切りだしなどに上川愛用。しかし、jack に対応していないなどの欠点があります。



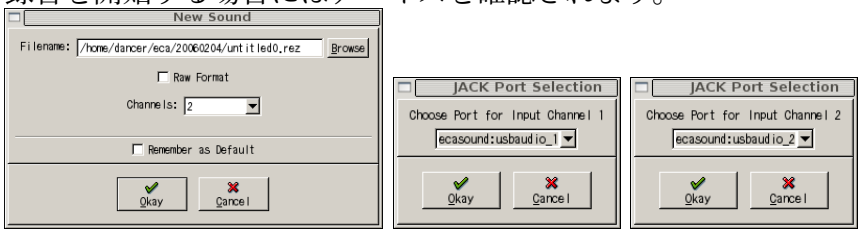
5.4.3 rezound

apt-get install rezound でインストール。

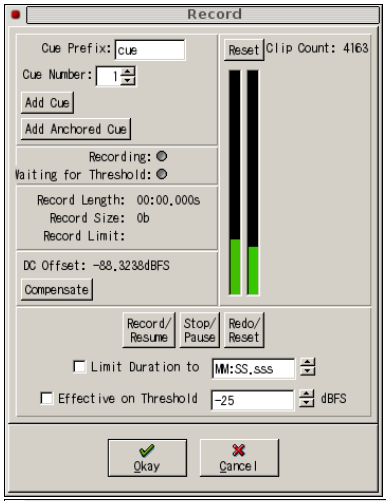
サウンドの編集用のツールです。jack 出力をサポートしています。

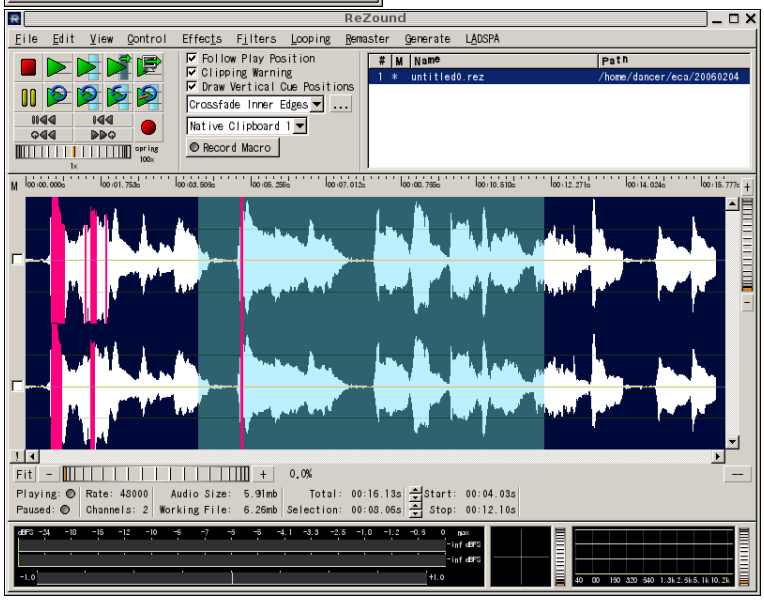
```
$ rezound --audio-method jack
```

録音を開始する場合にはデバイスを確認されます。

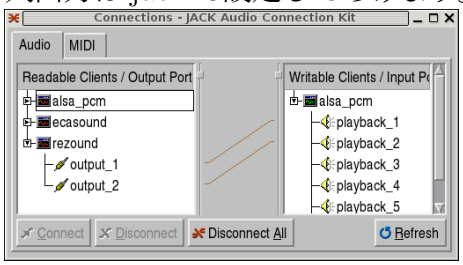


録音する際にはレベルの表示もしてくれます。





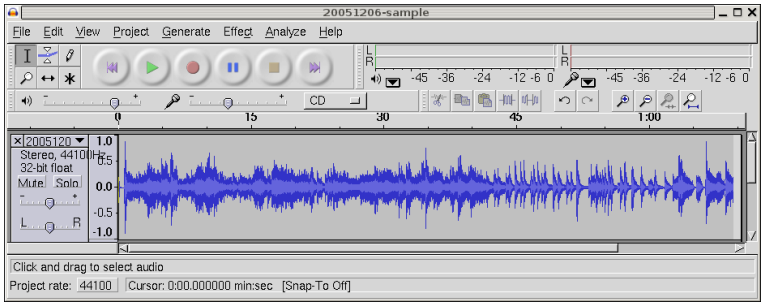
入出力は jack で設定してあげます。



5.4.4 audacity

apt-get install audacity でインストール。

audacity で起動。マルチトラックのオーディオ編集に最適。巨大な波形データもメモリ上に全てをロードしようとはしないので編集できる。巨大なデータの一次処理用には上川愛用。しかし、jack をサポートしていません。



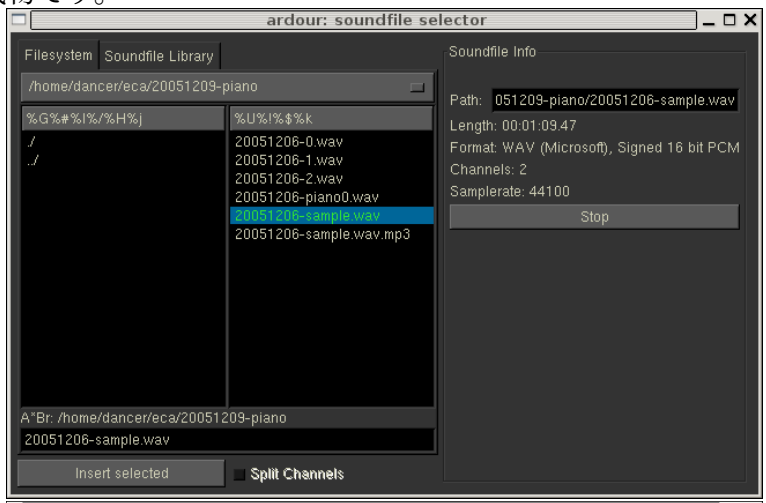
以前は日本語インタフェースを利用しようとすると悲惨でしたが、直っているようです。まだ一部問題が残っているようですが、全く使えないほどではないです。

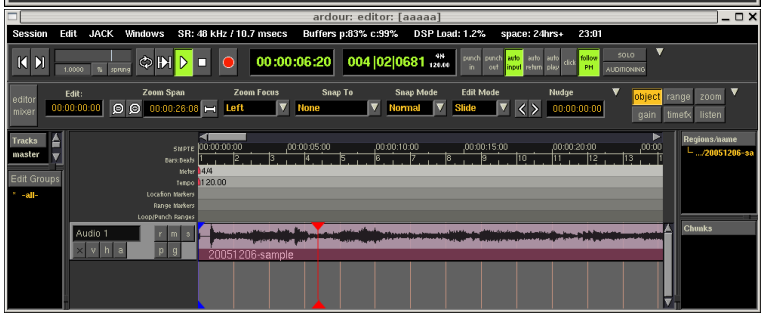
5.4.5 ardour

apt-get install ardour-gtk でインストール。

マルチトラックの音声ファイルは編集できるけど、音符が編集できそうな雰囲気は無いです。

Paul Davis という人が中心に開発していて、彼はこのために jack と LADSPA を実装し、Hammerfall のサウンドカードのドライバを ALSA 用に実装したというよくわからないけど凄い代物です。

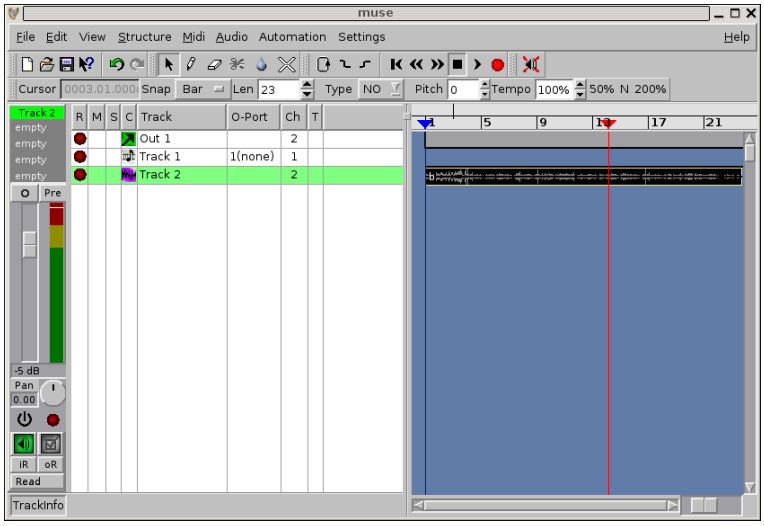




5.4.6 muse

apt-get install muse でインストール。

MIDI トラックと音声トラックが同時に扱えるようです。jack と ALSA MIDI に対応しているようです。使い方がわからず。



5.4.7 snd

音声業界での emacs と呼ばれています。使い方がわからないです。誰か教えて下さい。

6 Debian TEX のファイル構造

上川

TEXpolicy について簡単に解説します.

6.1 文書

この文書は tex-common パッケージに入っている Debian-TEX-Policy ファイル (`file:///usr/share/doc/tex-common/Debian-TeX-Policy.pdf.gz`) を概訳したものです.

6.2 用語

用語が定義してあります.

6.3 ファイル配置

TEX の入力ファイルのみを TEXMF ツリーに配置します. そうでないものは, /usr/share/PACKAGE に配置します. 例外として, 説明のためのテキストファイルは TEXMF ツリーに配置することができます.

6.3.1 パス検索と libkpathsea/libkpse

ファイルフォーマットなどに基づいて, TEXMF ツリーの検索をするためのライブラリです. libkpathsea はあまり考えていませんでしたが, libkpse は, API/ABI を考慮したライブラリです.

スクリプトからは kpsewhich, kpsepath, kpsexpand, kpsestat を利用できます.

6.3.2 ディレクトリツリー

配置は, TEX ディレクトリ構造標準 (TDS) に準拠する. TDS の古いバージョンに準拠するのはバグ. TDS の新しいバージョンに依存しながらこの tex-common パッケージや TEX の基本パッケージの十分新しいバージョンに依存していないのもバグです.

- /usr/share/texmf-tetex/: TEXMFDIST
- /usr/share/texmf-texlive/: TEXMFDIST
- /usr/share/texmf/: TEXMFMAIN
- /var/lib/texmf/: TEXMFSYSVAR
- /etc/texmf/: TEXMFSYSCONFIG
- /usr/share/texmf-site/: TEXMFSITE
- /usr/local/share/texmf/: TEXMFLOCAL
- texmf.cnf に指定してある TEXMFHOME の値, もしくは環境変数としての値.
- 必須ではない : 各ユーザ用の設定ファイルディレクトリ TEXMFCONFIG, 生成されたファイルのディレクトリ TEXMFVAR

検索は下から優先します.

TEXMFMAIN と TEXMFDIST の Debian での使い方は upstream と違います．TEXMFMAIN がバイナリと一致する必要のあるものを配置していあり， TEXMFDIST はあたらしい TEXMF が配布されて上書きされるものを配置していますが，そのような配置はパッケージマネージメントシステムがきちんと動いている環境では必要ありません.

Debian では，TEXMFDIST を basic TEXpackages 用にしており[*16]，TEXMFMAIN は新しいバージョンを提供したりするアドオンパッケージ用にしています.

パッケージはメンテナスクリプトの中で TEXMFHOME を無視するように考慮すべきです.

6.3.3 生成されるファイル

フォントは /var/cache/fonts におきます．その他は /var/lib/texmf 以下，もしくはユーザの指定したディレクトリの下に TDS に準拠したパスに配置します.

/etc/texmf/texmf.cnf は例外です．管理者が編集することは意図していませんが，編集されていた場合には自動生成ツールはその変更を尊重してください．Debian パッケージはそのファイルは変更してはいけません.

6.3.4 ファイル名とファイルの別バージョンのインストール

TEXMF ツリーにすでにあるファイル名と同じ名前ではファイルはインストールしてはいけません．ただし別のアプリケーションからしか見えないサブディレクトリにしかない場合には同じ名前でもかまいません．そういうディレクトリは kpsewhich の –progname や –format にて得られます.

- Basic TEXpackages はそれぞれ同じようなファイルを自分の TEXMFDIST にインストールします.
- より新しいバージョンのファイルが必要な場合，TEXMFDIST にすでにあるファイルを自分のバージョンでおきかえることができ，TEXMFMAIN に配置します.
  ただ，この場合，basic tex packages のメンテナに連絡[*17]してください.
  新しいファイルが後方互換であるように注意してください.
  混乱をまねくため，二種類を越える種類のバージョンが共存することや，dpkg-divert の利用は推奨しません.

6.3.5 ドキュメント

パッケージはドキュメントを texdoc に提供するべきです．/usr/share/doc/texmf 以下のサブディレクトリにファイルをインストールするか，適切なシンボリックリンクを提供することで実現できます.

/usr/share/texmf/doc にはファイルをインストールしないでください．ここは/usr/share/doc/texmf へのシンボリックリンクです.

ドキュメントの代表的なドキュメントはパッケージ名に関連した名前にして，manual.pdf や index.html という名前にしないでください.[*18]

[*16] 一部例外あり

[*17] wishlist バグ

[*18] texdoc packagename と指定できるようにするため，これができないとユーザは texdoc packagename/user.dvi のようなコマンドラインを texdoc -s keyword で検索する必要がでてきます.

6.4 設定

6.4.1 設定ファイル

TEX において，あらゆる TEX の入力ファイルは設定ファイルになりえますが，設定ファイルが多くなりすぎるのを防ぐため， conffile や configuration file としてファイルをインストールしないでください．/etc/texmf/tex には空のディレクトリを作成し，ユーザにどういうファイルを作成するべきかを指定してください．

/etc/texmf/ は TDS のツリーであり，任意の設定ができます．Debian の tex-common が提供する /etc/texmf/texmf.d/ などは検索対象ではないため，TEX の入力ファイルではないものの置場として利用できます．

6.4.2 設定更新プログラム

/etc/texmf/texmf.cnf が中心の設定です．/var/lib/texmf/web2c/updmap.cfg がフォントの設定ファイルです．/var/lib/texmf/tex/generic/config/language.dat がハイフネーションと言語の設定で， /var/lib/texmf/web2c/fmtutil.cnf にてフォーマットの生成が管理されています．

/etc/texmf 以下のツリーからこの四つのファイルは生成されています．

updmap.cfg language.dat fmtutil.cnf についてはこれが唯一の設定方法です．

texmf.cnf は管理者によって編集可能で，変更は ucf で管理されます．

パッケージのメンテナスクリプトは直接編集せず update-texmf を利用します．管理者も update-texmf を利用することを推奨します．

パッケージは設定項目を追加してよいですが，他のパッケージの設定を上書きしようとはしないでください．共有する設定項目は basic TEXpackages でどのパッケージでも利用できるような値を提供させてください．もしデフォルトが不可能であれば，そのことを関係パッケージのメンテナの間で合意をとってください．

メンテナスクリプトは， update-updmap を –quiet オプションをつけて呼び出して下さい．それ以外については，設定アップデートプログラムには特にオプションをつけずに呼び出して下さい．ディレクトリ構造の内部的な変更ができるようにするためです．

updmap.cfg を変更するパッケージは updmap-sys を呼び出す必要があります．language.dat か fmtutil.cnf を変更するプログラムは fmtutil-sys を呼び出す必要があります．その前に mktexlsr を呼び出すのを忘れないで下さい．

6.4.3 フォント設定

PS type1 フォントを提供するパッケージはどんな Basic TEXpackage でも利用できるようにしてください．tex-common で提供されている dh_installtex を利用してください．詳細はマニュアルページ dh_installtex(1) 参照．

- tex-common に depend して，Basic TEXpackage には依存しない
- .map ファイルは TEXMFMAIN/fonts/map にインストールする．
- その他の必要そうなものもインストールする .pfb, .tfm, .enc, .fd, .sty, 文書等
- /etc/texmf/updmap.d/ に 10*name*.cfg という名前法則でファイルを配置する．update-updmap が /var/lib/texmf/web2c/updmap.cfg を生成するのに利用される．`# -_- DebPkgProvidedMaps -_-`を含むファイル形式で詳細は update-updmap(1) 参照．
- /var/lib/tex-common/fontmap-cfg/*package*.list に一行一つさきほどの.cfg を記述．パッ

ケージのファイルとしてインストールすること.

- package.postinst と，postrm(remove/disappear を指定されたばあい) にて， update-updmap –quiet, mktexlsr, updmap-sys をその順番に実行すること.
  (以下 dh_installtex を利用すればよいので詳細な項目略)

6.4.4 言語/ハイフネーション設定

dh_installtex を利用すればよいです.

必要なファイルを TEXMFMAIN に配置し，.cnf ファイルを /etc/texmf/language.d にインストール， update-language を呼び出すと /var/lib/texmf/tex/generic/config/language.dat が生成されます.

ここまでいくと，あとは `fmtutil-sys --byhyphen ‘kpsewhich --progname=latex language.dat‘` を呼び出すと再生成処理が行われ，利用できるようになります．削除するときもこのコマンドです.

現状，update-language は LaTeX 以外の hyphenation 設定ファイルを利用しているものには提供されていません.

6.4.5 フォーマット設定

dh_installtex を利用すればよいです.

fmtutil.cnf(5) に説明してある形式のファイルを/etc/texmf/fmt.d/に配置し，update-fmtutil を実行し，fmtutil-sys –byfmt *format* を実行します．cnf ファイルの最後の行に記述する *format*.ini が見付かった場合だけフォーマットが生成されるので，*format*.ini は conffile であってはいけません.

6.4.6 TeX に Build-Depend する場合のベストプラクティス

Build-Depend するパッケージが変更した設定を必要とする場合，その設定を静的にもつべきではありません．もしパッケージがビルドするのに適切でない設定があるのであれば，それは通常その設定を提供しているパッケージのバグですので，そちらを修正してください．回避策は後々大きな問題となってかえってくる事がおおいです.

もしどうしても必要なら，設定更新プログラムの –outputdir と–add-file を利用して生成してください.

6.4.7 コマンドの実行とフォーマットファイル

TeX のフォーマットが必要な場合は，postinst スクリプトで実施してください．依存するパッケージは Depends: をするだけで十分です．フォーマットファイルの存在を確認などをすると内部構造がかわると壊れやすいのでやめてください.

Debian パッケージは fmtutil か fmtutil-sys を常に利用するべきで，/etc/texmf/fmt.d/ (fmtutil-sys) にファイルを追加するか，fmtutil (–cnffile オプションを指定) で，ローカルの cnf ファイルを更新するようにしてください.

管理者は環境変数を texmf.cnf でオーバライドできます．これが postinst のエラーにつながっている例があります．環境変数はフォーマット生成前などに postinst で unset してください.

6.4.8 Dpkg の Post-Invoke の仕組み

この案は没になりました.

6.5 サンプルコード

(省略)

7 Debian latex の現状調査

上川

まず，Debian の latex で日本語のドキュメントを処理するための手順について確認します．ここでは，例としてドキュメントを準備し，そのドキュメントソースを PDF ファイルにするまでの手順を確認します．

7.1 platex で PDF を作成する方法

platex は ptex-bin パッケージに含まれています．日本語でかかれた tex ファイルから dvi ファイルを生成することができます．

```
$ platex debianmeetingresume200604.tex
```

Debian での platex のデフォルトは，EUC モードです．ソースファイルのエンコーディングは iso-2022-jp か euc-jp にしておくとよいでしょう．SJIS モードでの処理については，Debian パッケージとしてはサポートしていません[*19]

文字コード	可否
EUC-JP	○
SJIS	×
ISO-2022-JP	○
UTF-8	×

dvi ファイルから PDF を作成する方法は，いくつかあります．

- dvipdfmx を利用する
  毎月の Debian 勉強会用の資料を処理するのに利用している方法です．

```
$ dvipdfmx debianmeetingresume200604.dvi
```

- dvips で PS を生成し ps2pdf を利用する

```
$ ps2pdf debianmeetingresume200604.ps
mktexpk: don't know how to create bitmap font for rml.
dvips: Font rml not found, characters will be left blank.
$ ps2pdf debianmeetingresume200604.ps
(結果の PDF ファイルには日本語の文字がまったく表示されない)
```

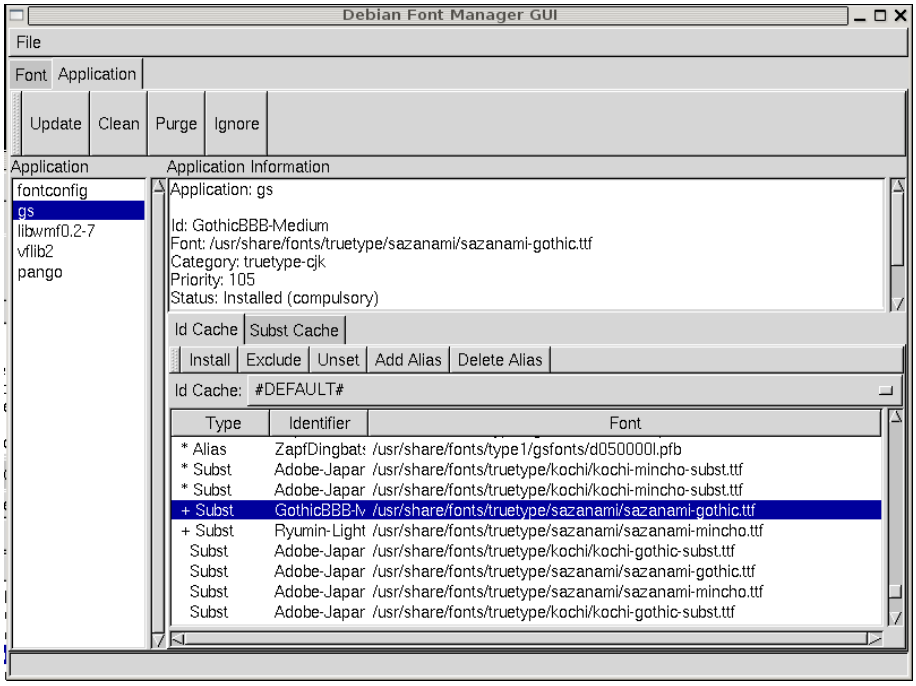
- dvi2ps で PS を生成し，ps2pdf を利用する

```
$ dvi2ps debianmeetingresume200604.dvi > debianmeetingresume200604.ps
$ GS_LIB=/usr/share/fonts ps2pdf debianmeetingresume200604.ps
(Ryumin-Light が見付からない，という gs のエラーが出力され途中で停止する)
```

  現状 GS_LIB 環境変数の指定が必要になっているのと，Kochi フォントを利用しているとエラーを吐いて停止するという問題があります．dfontmgr を利用して，ps2pdf(gs) が利用するフォントとして kochi フォント以外を指定する必要があります．[*20]

[*19] http://bugs.debian.org/234547

[*20] 参　考：http://lists.debian.or.jp/debian-users/200501/msg00008.html, http://kmuto.jp/d/index.cgi/debian/gs-esp-8151.htm



それぞれの方法にハイパーリンクや pstricks の扱いに癖があります．たとえば，dvipdfmx の場合は hyperref パッケージを読み込む際に，dvipdfm オプションを指定してあげる必要があります．

```
\usepackage[dvipdfm]{hyperref}
```

7.2 jlatex

jlatex は，jtex-bin パッケージに入っています．tex ファイルから dvi ファイルを生成することができます．

ただ，platex 向けの既存のドキュメントをコンパイルしようとしてもエラーになります．Debian 勉強会資料で利用している jsarticle.cls や ascmac.sty などが platex 専用だからのようです． j-article などを利用する必要があるようです[*21]．また，このドキュメントに関してはそれ以外にも問題があり，簡単な変更では処理できませんでした．

```
 $ jlatex debianmeetingresume200604.tex
! LaTeX Error: File `jsarticle.cls' not found.
(エラーがでてコンパイルできない)
```

7.3 cjk-latex

babel の CJK パッケージとして実装されており，通常の latex を利用して日本語を処理できるそうです．

そのままでは `/usr/share/doc/cjk-latex/examples` にあるサンプルファイルすらコンパイルできないので，困りものです．

参考：`http://lists.debian.or.jp/debian-devel/200007/msg00150.html`

7.4 pdfelatex

Debian には部品が現状足りないようです．

参考：`http://cise.edu.mie-u.ac.jp/~okumura/texfaq/qa/17780.html`

[*21] jarticle は利用できるようになっている．

7.5 multex

パッケージをインストールしただけでは，サンプルファイルを処理してもフォントが一部足りないようで，表示されない文字があります.

参考：http://lists.debian.or.jp/debian-users/200106/msg00081.htm

7.6 lambda (omega)

http://www.fsci.fuk.kindai.ac.jp/kakuto/soft.html，http://cise.edu.mie-u.ac.jp/~okumura/texfaq/japanese/ などを参考にしてみてください．現状，実用的に既存のドキュメントをそのまま処理できるような形式ではないことがうかがえます.

8　一年間 Debian 勉強会をやってみて

上川

この記事の目的は、終ってからだと忘れてしまいそうだし、最中だといそがしくていっぱいいっぱいなのでどこにも記録されずに忘れ去られてしまいそうな事項についてメモをしています。

希望としては、ここに書いてある内容をみて、今後のミーティングの運営の手伝いを人に頼めるようになればよいなと思っています。

8.1　月例の Debian 勉強会のワークフロー

2005 年、Debian 勉強会を毎回実施する際に利用したワークフローを紹介します。今後の勉強会などの参考にできるかと思い、記録します。

参加者規模、10 名から 20 名程度でした。予算規模は、宴会を含むと一回 5 万円から 10 万円程度です。宴会を含まないのであれば、多くて 1 万円くらいでした。(表 1)

表 1　予算概算

項目	予算
部屋代	1500
コピー代	300 × **人数**
宴会代	5000 × **人数**

8.1.1　1 年前

開催者側のスケジュールの確保。上川は一年前にだいたいその年のスケジュールを決めています。

8.1.2　2ヵ月前

会場の予約確保、開催を決断。

8.1.3　1ヵ月前

この時期にすくなくともテーマの設定をします。講師の確保をしておきます。資料の作成開始をしておかないと間に合わないでしょう。目処がつきそうだったら、開催のスケジュールを対外的に公表します。参加者にスケジュールの調整をお願いします。

8.1.4　1 週間前

宴会の会場選定などを実施します。大体、資料作成のデッドラインです。リマインダーの送付をします。

8.1.5 2 日前

事前課題の文書を事前資料に転記したり、最終的な文書の校正。この時点で資料の印刷用の最終版が作成。

宴会の人数確定。宴会予約。

ただ、二日前に選定するとなると場所が限られる場合が多いので、本当はもっと早い時期がよいです。一般には、確定が早ければ早いほど予約は安くすみます。二日前になっても参加できるかどうかわからないという人がいますが、そういう人の対応は難しいです。店の柔軟な対応に期待するか、コストをかけるしかないです。しかし、12 月 10 日の宴会も前日で予約できたのであれば、実は当日に急に開催決定するとかいうのでさえなければ何とかなる物なのかもしれないです。

8.1.6 1 日前

資料の印刷をします。Kinko's にすべてを依頼する場合は場合にもよりますが、半日くらいは見込む必要があります。自分で全部するとしても量によりますが、一時間は見込む必要があります。

Kinko's にすべてを依頼する場合、部数が少ないとかなり割高になります。[*22]

8.1.7 当日

資料をもっていきます。司会をします。適当にもりあがります。

宴会も実施します。2005 年は、講師は無料で宴会、ということで運営しました。ただ、ときどきそれでは予算が苦しい場合も多々ありました。なぜか宴会に来ているのに現金をもっていない人とかの扱いには苦慮します。

予算は、ほぼ確実になんらかの理由でのキャンセルが発生するため、余裕を 20 %くらい確保できていないと赤字になります。[*23]

8.2 JDMC のような大きなイベントのワークフロー

Japan Debian Miniconf はまだまだこれから育って行くようなイベントです。今回蓄積できたノウハウだけで今後もうまく開催できるとは思っていません。ただ、今回イベントを開催する上で重要でたりなかった点を列記していきます。

- 連絡先を明確にする。
- 緊急時に判断をできる人を明確にする。
- 連絡網を整備する。
- ディスカッションができて、そこで決定した事項が合意したとみなせる環境をきめてしまう。たとえば IRC。

一人ではかぶりきれない責任もあるため、大きなイベントでは、本気で責任をもって開催したい、と思っている人が複数いる必要があります。

会議の内容をログに残して全員に周知させる係の人が必要です。理想としては、実働部隊と分けられればわけたほうがよいです。JDMC では、ほとんど矢吹さんだけに情報が集中していたは

[*22] A3 の紙に A4 を面付けしてもらい、なかとじホッチキス製本にするとホッチキスだけで 150 円/冊になります。コピーが一面 14 円程度になります。結果として、一冊 450 円程度になる。会費を 500 円しか徴収しないことを考えると、会場費用を考えると確実に赤字になってしまうので注意。

[*23] 回避策としては、来ない人から徴収するとかいう案も可能性としてはありますが、来ない人から徴収するということは暗黙に開催者が次回その来ない人から徴収する分について肩代りする、ということを意味するため、オーバヘッドが発生することを忘れてはならない。

ずで、メーリングリスト上ではながれていない情報が多数ありました。もしかすると検討する余裕がなかった項目も多数あったかもしれません。

また、メーリングリストで流れる情報は時系列なので、現在のステータスを一覧で把握できないです。タスクトラッキングが重要になります。

また、全員がどういう方法で情報交換をするのかという点について同意が必要です。メールで主要な情報交換はなされたのだが、一部の主要メンバーの人達がメールをほぼ全く読んでいなかったという問題がありました。

8.2.1 2 年前

参加者が稼動できるように日程を確保します。スポンサーにあたりをつけはじめる。マネージメント層に交渉します。それとなく開催できそうな雰囲気がただよっていることを確認します。

8.2.2 1 年前

一年前か、半年前くらいの時期にスポンサーの予算が大体確定するはずです。講師に関しての予定、参加者の人数、プログラムの大体のイメージが決まっている必要があります。

初の企画でないのなら、前年度のイベントに参加して運営側で何がおきるかを明確にして、会場のサイジングなどをする必要があります。

スポンサーに関しては、通常スポンサーから資金が提供されるのはイベントが終了した後です。そのため、事前に当面必要な運転資金をどう確保するのかというのも検討しておく必要があります。

また、赤字になることが見込まれるのであれば、計画を中止するという選択も必要です。

8.2.3 6 月前

会場を確保します。宴会場を確保します。

予約システムを整備し、広報します。広報は下記を想定しています。

- マスメディアへの広報
- IRC などのくちこみ。`#debian-devel@opn` など
- Blog
- DWN への投稿
- メーリングリスト, debian-devel@debian.or.jp, debian-users@debian.or.jp, debian-devel@lists.debian.org
- Mixi などのソーシャルネットワーク
- Slashdot へ たれこむ

また、GPG サイン会などを実施するのなら事前に充分に準備、広報する必要があります。

8.2.4 1 月前

宴会場の確保、決定が必要です。

参加者の登録が確定しているくらいが本当は好ましいです。人数が足りないのであればがんばってかきあつめるなどのアクションをとります。

ロジスティックの計画があるので、この時点での人数の把握は重要。

8.2.5 7 日前

宴会場に連絡して、大体の人数を調整。

8.2.6 2 日前

宴会場との調整、当日の人数のより確度の高い情報を提供。

8.2.7 当日

参加者の出欠確認
参加費用の集金を実施します。
スポンサー企業からの提供物を提供します。スポンサーのグッズとかです。

8.2.8 事後

スポンサー企業への報告を作成します。結果報告書を書き上げます。
参加者の報告をまとめてもらいます。来年のイベントに繋げるために重要です。
次回への検討をはじめます。

8.2.9 参考文献

いろいろと他のイベントの報告などもあります。参考になりそうなものを列挙します。

- Joey の LinuxTag レポート `http://www.infodrom.org/~joey/Vortraege/2005-06-24/index.html`
- Joey の LinuxTag 感謝状 `http://www.infodrom.org/~joey/log/?200512020951`
- Debconf5 Final Report `http://lists.debian.org/debian-devel-announce/2005/12/msg00001.html`

8.3 やった内容

やった内容はけっこういろいろありました。最初は一般的なうけをねらったものもありましたが、全体的には技術的な内容を主としています。

- 毎月のクイズ
- 最初の数回はグループワーク
- バックアップリストアについて
- ネットワーク監視
- reportbug の使い方
- debhelper
- Social Contract
- po-debconf
- lintian/linda
- dpkg-cross
- dsys/update-alternatives
- debian-installer
- dpatch

- toolchain
- ITP からアップロードまでの流れ
- debconf 2005 参加報告
- Debian JP web の改革
- debconf の使い方
- apt-listbugs
- debbugs
- dpkg-statoverride
- Debian Weekly News 日本語翻訳のフロー

来た人数は表 2 にあるような数字です。正確な記録は実は残っていないような気がしています。議事録をあさればわかるのかもしれません。

表 2　参加人数 (概算)

	人数
2005 年 1 月	21
2005 年 2 月	10
2005 年 3 月 (早朝)	8
2005 年 4 月	6
2005 年 5 月	8
2005 年 6 月	12
2005 年 7 月	12
2005 年 8 月	7
2005 年 9 月	14
2005 年 10 月	9
2005 年 11 月	8
2005 年 12 月	8

8.4 おきたトラブル

勉強会を毎月開催する上で発生したトラブルを紹介します。表 3 です．数字はどれくらいの確率でおきたような気がしているかというのをなんとなく気分的に定量的に書いてみました。

8.5 できた内容

事前課題により事前に awareness を向上しました。いろいろと知らないことを積極的に調べることにより講師がその分野に詳しくなるという副作用があります。調査して文章を書いている過程でバグが気に入らないので、バグが直る、ということを若干期待しています。

勉強会をクイズではじめてみんなで発言することにより場を和ませることができたか？と思っています。クイズは、全員に紙で配布して解いてもらわないと、順番にあてる形でやると、一部の回答している人だけが集中して、その他の人が当事者意識をもたないという問題があります (JDMC での失敗)。紙を毎回印刷するコストは大きいですが、それなりに効果もあります。

終ってからの blog へのリンク、議事録の掲載についてはあまり反響が無いです。事前資料の

表 3　発生トラブル

イベント	発生率
パソコンが盗まれる	10%
家が水没する	10%
病気で倒れる	20%
〆切におくれる	20%
なぜか講師のひとと前日まで音信不通	10%
20 分くらいまえに連絡してきて、来れないという参加予定者がいる。	100%
何も連絡なく来ない人がいる	100%
なぜか赤字	40%

PDF についてはいろいろと blog とかをみているとコメントがあったこともありますが、そちらも反応はあまりないようです。見られているのかどうか不明です。PDF ファイルだからでしょうか？

勉強会の資料を半年分まとめて書籍のような形式にして、Debian 勉強会資料ということで、コミックマーケットにて販売してもらう、という試みをしています。これは、以前 Debian 関係の話題が豊富にはいっていた「Debuan BNU/Linux 不徹底入門」という同人誌があったのですが、それが廃刊になったため、その代替となれることをめざしているためです。

8.6　今後やりたいこと

今後は事前の打合せをもっと密にしたいと考えています。

IRC の debianjp チャンネルで偶然いたメンバーで、なんとなく打合せをする、ということはできていました。しかし、最初のころは事実上打合せは上川が電話で呼び出してどっかの飲み屋でする、という手法をとっていました。後半は時間の都合で、ほとんど打合せができていなくて、前回の勉強会の後の飲み会で決定した内容そのままで次の勉強会にのりこむ感じでした。

事後の処理をなんとかしたい、と考えています。開催した結果をもっと参加していない人にもわかるように効率よくアウトプットできないだろうか、と思っています。

他の人が参加したいと思えるようなアウトプットが出せないだろうか、と考えています。勉強会自体に Debian 関係者が参加したい、と思えるようになることと、Debian にこれから入る人達が参加したい、と思えるようになることが必要だと思います。

来年の提案としてシステムの構築報告、動作検証、というのはどうだろうか。「この組合せはできるだろう」、という組合せに関して、連係はこうやってできる、ということを報告していけば、多くの人がその動作を確認できるようになり、問題も解決していけるでしょう。Debian ユーザの勉強会というのはそういう形になるのではないでしょうか。

Debian 勉強会以外では、おそらく開発に必要な情報についてまとめて情報収集できる場というのが存在しないため、開発に必要な情報については継続してやりたいと考えています。ただ、2 回に一回くらいはそういうユーザよりの情報の検証にあててもよいだろうと考えています。

また、勉強会でいいっぱなしではなく、勉強会の結果何かが起きる、というようにしたい。メンテナがバグトラッキングシステムにバグをファイルします、というように宣言して、毎月その進捗を報告する、という内容にしてみてもよいかな、と思っています。

9 Debian 勉強会の事前資料の作成はどうやってやったか

上川

9.1 作成ツール

作成のデータ共有には alioth の cvs を利用しました.

データの編集は上川は emacs+yatex+whizzytex で実施しました．LaTeX 処理系として platex を利用しました．PDF の作成は，dvipdfmx を利用しました.

プリビューは dvi ファイルは advi，PDF ファイルに関しては，xpdf を利用しました.

上川の編集環境は Debian sid で，常に開発中の環境だったので，その時期において動かないツールというのもたまにあり，それなりに大変でした．原稿の編集中には `apt-get dist-upgrade` しないように自制していました.

9.2 LaTeX ソース

事前資料は LaTeX で作成しました。作業は大きく 3 種類ありました。

- クイズの作成
- 参加事前課題の作成
- 勉強会のネタの作成

9.2.1 クイズ

クイズについては、 LaTeX のマクロでクイズを作成できるようにして、それを利用して本文を作成しました。

LaTeX のソースに下記のように記述すると、

```
\santaku{問題文}{回答 A}{回答 B}{回答 C}{回答}
```

下記のような出力がでるようになりました。

問題 1. 問題文

□ A 回答 A

□ B 回答 B

□ C 回答 C

また、その出力を latex-beamer[*24]で処理をして、プレゼンテーション形式になるようにしました。2005 年 10 月以降、勉強会当日は、それを利用して回答を提示するようにしました。

[*24] LaTeX でプレゼンテーションを作成するためのスタイル

9.2.2 参加事前課題

メールにて参加者から plain text できたものを気合いで LaTeX になおしました。LaTeX で使えない文字というのがあるので、それをエスケープすることと、構造文書については、構造を LaTeX 用に書き直すという手順が必要です。

例えば、下記のような文章は

```
□ これについて
こんなことをしてみた
□ あれについて
あんなことをしてみた
□ それについて
いっぱいしてみた
```

itemize 環境を利用して下記のような文書になります。

```
\begin{itemize}
 \item{これについて} こんなことをしてみた
 \item{あれについて} あんなことをしてみた
 \item{それについて} いっぱいしてみた
\end{itemize}
```

- これについて こんなことをしてみた
- あれについて あんなことをしてみた
- それについて いっぱいしてみた

9.2.3 勉強会のネタ

講師の方に直接 LaTeX で文書を書いてもらいました。CVS レポジトリは alioth.debian.org でホスティングしてもらったので、そこに共同開発者という形で参加してもらいました。

LaTeX のスタイルはほぼそのまま jsarticle を採用しています。ただ、セクションのはじめの部分だけはみかけを派手にしようとして dancersection というマクロを作って独自に定義しています。各筆者は dancersection 以下に適当に subsection を作って文書を作成する、というルールになっています。

```
\dancersection{一年間 Debian 勉強会をやってみて}{上川}
\label{sec:uekawa}
%% 上川の記事はここから
\subsection{セクションの名前 }

文章がだらだらと続く

\subsubsection{セクションの名前 }

.
.
.

\subsection{セクションの名前 }
.
.
.
```

9.2.4 URL やメールアドレスの処理

\url{http://url...} というように表記しています。また、メールアドレスも環境を定義するのが面倒なので、そのまま\url{**メール@アドレス**}という形式にしています。

9.2.5 特殊文字の処理

LaTeX でエスケープが必要な文字については表 4 のように対処しています。

表 4　特殊文字

文字	名称	表記
~	チルダ	\~{ }
_	アンダーライン	\underline{ }
#	ハッシュ	\#
%	パーセント	\%

10　Debian Weekly News trivia quiz

上川純一

ところで、Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、意味がわからないところもあるかも知れません。みんなで DWN を読んでみましょう。

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

10.1　2005 年 46 号

`http://www.debian.org/News/weekly/2005/46/` にある 11 月 15 日版です。

問題 2. Debian armeb の進捗はどうか

□ A やっと gcc/glibc/binutils が移植された
□ B ほとんどのパッケージが移植されている
□ C まだ起動もしていない

問題 3. DevJam で Java の現状について議論があった。その際の認識はどうだったか

□ A まだフリーな java で全てを実装できていないので、動かないものがある
□ B フリーな Java は充分利用できる状況で、それだけで全てが充足できる。
□ C フリーな Java は全く利用出来ない状態

問題 4. Clam Antivirus について Marc Haber が発表したのは

□ A 15 分毎に更新を確認して、あたらしくなっていたら自動で volatile.debian.net にアップロードする
□ B 更新は手動で確認して、メンテナが暇なときにアップデートする。新しいデータを常に欲しい人は、頑張って自分でアップデートすること。
□ C データ量が多いため、更新はしないので、各自がんばって更新してください。

問題 5. debian-installer etch beta が出ました。Joey Hess がこんなに時間がかかったことにつ

いて言明したのは
□ A めんどくさかったので放置していたので、こんなに時間がかかりました
□ B １０位の項目についてそれぞれで３日づつ遅延要因になるため、一月くらいは遅れるはめになる
□ C ちゃんとハックできる人が参加していないので、コードの品質が下がったため、こんなに時間がかかりました。

問題 6. SugarCRM は MPL1.1 をベースとしたライセンスで配布されている。そのライセンスはフリーだろうか
□ A MPL は Mozilla のライセンスなので、その時点でフリーだ
□ B ウェブページにフリーソフトだ、と書いてあるので、フリーだ。
□ C 改変した場合に名前を利用できないことになっているので、名前を変更すればよいだろう

問題 7. Debconf の発表資料を DFSG フリーにしようという提案について Anthony Towns がした反論は
□ A ML でのスレッドなど DFSG フリーでないコンテンツは多数ある。全てがそうである必要はない。
□ B ライセンスなんてつけるだけ無駄なので、つけないほうがよいでしょう。
□ C あらゆるものは DFSG フリーどころか、全部 GPL であるべきなので、GPL 以外のライセンスは考えるのもおこがましい。

問題 8. Gabor Gombas さんが、複数の-dev パッケージが conflict することについて苦情を出した。その対応は
□ A -dev パッケージがインストールできないのは問題なので、上流のやっている内容を改変して共存できるようにするのがよい
□ B openssl と gnutls をまぜるほうがライセンス的に適切なので、両方がリンクされたパッケージを作る
□ C include ファイルのパスなどは開発用の API の一部であり、同じパスを利用する複数の-dev パッケージは conflict して当然だ。

問題 9. ping が Linux 専用である点についての議論で、FreeBSD や Hurd でも動作させるためにパッチを適用することに対してはどういう意見が出たか
□ A 今後の Debian の一貫性を維持するためにはするべきだ
□ B ping なんて BSD 上でははやらないのでなくしてもよい
□ C あきらかに fork しているため、メンテナンスが大変になる

10.2 2005 年 47 号

`http://www.debian.org/News/weekly/2005/47/` にある 11 月 22 日版です。
問題 10. Matthias Klose が g++ について発表したのは何か

□ A g++ は今後 D 言語用のコンパイラによって置き換えられるので、C++ なんて古い言語をつかうのはもうやめろ
□ B g++ のメモリアロケータが変わるため、また g++ で生成されたライブラリの ABI が変更になる
□ C g++ は最適化するために今後はマクロの展開処理を省略する。そのために文法が若干変更になる

問題 11. Anthony Towns が -private メーリングリストについて提案したのは
□ A 3 年たったら一般公開する
□ B 存在自体を抹消する
□ C 即時公開メーリングリストにする

問題 12. Branden Robinson が DPL について何ができるかという説明文を発表した。その条文はいくつあるか
□ A 3
□ B 10
□ C 120

問題 13. Enrico Zini が発表した新しい検索エンジンでは何をもってパッケージを検索できるか
□ A 2ch の過去ログ情報を用いて検索
□ B debtags 情報を使って検索
□ C popcon の利用頻度情報を使って検索

問題 14. Ian Jackson が提案したのは何か
□ A パッケージの自動テストのためのスクリプトインタフェース
□ B パッケージを受け入れるときのための基準
□ C パッケージの品質をあげるための魔法

問題 15. Christopher Berg が発表した、メンテナ向けのパッケージ一覧ページの新機能でないのは
□ A パッケージがどれくらい人気あるのかということを確認できる
□ B パッケージがどれくらいよい品質なのかが確認できる
□ C 一覧で確認できるパッケージを任意に追加できる

問題 16. PHP ライセンスについて Steve Langasek の考えは
□ A PHP を使うこと自体がまず問題だ
□ B PHP 自体については問題ないが、PHP 以外にそのライセンスを適用するのには問題がある
□ C PHP ライセンスは本当に DFSG フリーなのかどうかはグレーだ

10.3 2005 年 48 号

http://www.debian.org/News/weekly/2005/48/ にある 11 月 29 日版です。

問題 17. Freetype に関して何が起きる、と Steve Langasek は宣言したか

□ A 誰も使っていないので、パッケージを削除する

□ B ABI に変更があったので、5 のパッケージが移行する必要がある

□ C ABI に変更があったので、600 のパッケージが移行する必要がある。

問題 18. sbuild の最新版はバージョンが 1.0-1 のパッケージに対しての binary NMU 番号をどうつけてくれるようになったか

□ A 1.0-1+b1

□ B 1.0-1.1

□ C 1.0-1.0.1

問題 19. Frank Küster は、パッケージの conffile への変更の反映を管理者が拒否し、その結果 postinst が失敗になることについて、問題ないだろう、と質問した。それに対しての Petter Reinholdtsen の対応は

□ A そういうエラーは管理者が拒否するのが問題なので、管理者を日勤教育するべきだ

□ B そのような問題は存在しない

□ C そういう場合には、設定ファイルを動作に必須なものとローカルで管理者がオーバライドする部分とに分離することを提案する

問題 20. vserver は何をするものか

□ A chroot などの技術を応用し、複数の仮想サーバコンテキストを作成してくれて、Linux 上で複数のサーバを仮想的に提供できる

□ B サインは

□ C サーバの統合管理のためのツール

10.4 2005 年 49 号

http://www.debian.org/News/weekly/2005/49/ にある 12 月 6 日版です。

問題 21. Manoj Srivastava が GR の議論期間を宣言した。今回の議論は何についてか

□ A -private メーリングリストの一般公開について

□ B -devel メーリングリストの秘密化について

□ C -mentors メーリングリストの会員制化について

問題 22. テンポラリディレクトリについての議論があり、ユーザ毎にテンポラリディレクトリを持つことがよいのではないかという結論が出た。ユーザ毎にテンポラリディレクトリを持つ際にその機能を実装してくれるのは

□ A /etc/profile でテンポラリディレクトリの作成

□ B init スクリプトでのディレクトリの作成

□ C pam-tmpdir という PAM モジュール

問題 23. C++ のメモリアロケータの移行でまだ移行できていない、ということでさらしあげに

なった日本の開発者は
□ A mhatta さんと土屋さん
□ B gniibe さんと鵜飼さん
□ C えとーさんと岩松さん

問題 24. パッケージがどのバージョン (unstable, stable, testing) 用に作成されたのかを確認する簡単な方法がないか、という質問に対しての Marc Brockschmidt の回答は何だったか
□ A パッケージのバージョン番号を見ればわかる
□ B パッケージの changelog を見ると、どのバージョン用にビルドしたのか、ということは確認できる。
□ C Debian のパッケージはほとんど全てが一旦は unstable にあったことがあり、testing と stable に入るため、パッケージがどれ用につくられるというものではない。

10.5 2005 年 50 号

http://www.debian.org/News/weekly/2005/50/ にある 2005 年 12 月 13 日版です。

問題 25. www.skolelinux.org について提案されていないことはどれか
□ A バグトラッキングシステムを共有する
□ B 関係者の blog を planet でアグリゲートしたりする
□ C こっそりと人を誘拐してメンバーを増やす

問題 26. Branden Robinson が Tuxjournal でのインタビューで、Debian の成功に貢献したものとしてあげたのは
□ A 自由なライセンスを強調する人達とソフトウェアの品質を強調する人達がそれぞれ貢献できてきたこと
□ B Ruby をがんがんつかってコードを書いた事
□ C お互いに仲のわるい開発者たちが足をひっぱりあいながらお互いを潰しあっていたこと

問題 27. GPL でリリースされているゲームボーイ用のエミュレータは main にいれてもよいのか
□ A フリーのゲームを開発しているグループがあるため、main にいれてもよい
□ B ゲームは商用のゲームしか存在しないため contrib にいれる必要がある
□ C エミュレーションという不純な動作は non-free にあるべきだ

問題 28. パッケージを実行用のパッケージ *pkg* と データ用のパッケージ *pkg*-data に分割する場合の確認項目について Bill Allombert は投稿した。そこで説明していなかったのは
□ A パッケージ名は pkg と pkg-data にしてほしい
□ B pkg-data は architecture: all にしてほしい
□ C pkg-data のサイズは 5MB を越えていることが望ましい

問題 29. tetex の設定ファイルについて Frank の提案したのは？

□ A /usr/share/texmf にデフォルトがあり、/etc/texmf にアドミニストレータの設定があり、HOME/texmf に各ユーザの設定がある構成
□ B Debian メンテナが一番偉いのでユーザの設定を無視して、世界統一の設定にすること
□ C Debian menu システムの設定システムが優秀なので、それをそのまま採用すること

10.6 2005 年 51 号

http://www.debian.org/News/weekly/2005/50/ にある 2005 年 12 月 20 日版です。

問題 30. debianforum.de は開始何年たったか
□ A 3 年
□ B 4 年
□ C 5 年

問題 31. Jaldhar H. Vyas はインドでは通信コストが高いため、雑誌に付録 DVD として Debian を付けたいと提案しました。ただ、複数枚はコストがかかるので、DVD 1 枚におさめたいと説明しました。Joerg Jaspert の回答は
□ A Cebit などの展示会で利用するためにすでに作成したことがあるので結構簡単だよ、と回答した
□ B そんなものつくることがおこがましい
□ C ソースだけだったら 1 枚でもいけるかも

問題 32. lsb 向けの起動スクリプトの利用方法について検討していた際に、エラーが発生した場合にコマンド自体のエラーが画面に出力されて表示がみだれた．この問題に対して提案された解決策は
□ A エラーは/dev/null へ
□ B エラーなんておきないようにする
□ C エラーなどを syslog に送信してみる

問題 33. dpkg-sig を利用して Debian パッケージに電子署名を追加することができる．最近 dpkg-sig の署名を含む Debian パッケージが Debian archive にアップロードできなくなっていた。その理由は
□ A dpkg にそんな機能拡張はしてはいけないという主義主張の問題
□ B 予期しない原因で jenniffer のチェックが厳しすぎたため
□ C 実は dpkg-sig なんてものはなかった

問題 34. TexLive パッケージのライセンスで Joerg Jaspert がおかしいと指摘したのは
□ A Live という名前がダメだ
□ B tex は時代遅れです
□ C DFSG という謎のライセンスを利用している部分が存在した

10.7　2005 年 52 号

http://www.debian.org/News/weekly/2005/52/ にある 12 月 27 日版です。

問題 35. Norbert Tretkovski は、backports.org で何がおきたと発表したのか。

□ A backports.org が etch に対応した
□ B backports.org のメンテナンスをあきらめた
□ C backports.org が sarge に対応した

問題 36. http://wiki.debian.org/StatusOfUnstable は何を説明してくれるページか

□ A 現状の unstable で何がおきているのかをまとめている wiki ページ
□ B unstable であるということはどういうのかといういことを熱く語るスレッド
□ C 今どういうことが unstable になりえるのかということを解説しているページ

問題 37. Kevin Locke が発表した powermgmt プロジェクトは何をするものか

□ A Debian の中での共通の電源管理用のインフラを提供することを目標とする
□ B ハックに必要な栄養の補給方法について検討することを目標とする
□ C 権力をいかに分配するのかということについて検討することを目標とする

10.8　2006 年 1 号

http://www.debian.org/News/weekly/2006/01/ にある 1 月 3 日版です。

問題 38. Debian パッケージを圧縮しなおすことで一番小さくできたのはどの圧縮ソフトウェアか

□ A gzip
□ B bzip2
□ C 7-zip

問題 39. apt-torrent は何をするものか

□ A apt 風のインタフェースで bittorrent を利用できるツール
□ B apt のパッケージダウンロードを bit-torrent 経由で実行するためのツール
□ C 海流予測用ツール

問題 40. vim-tiny は何をするものか

□ A nvi のかわりにデフォルトにするためにつくられた
□ B ただ vim を小さくしてみました
□ C vim の機能はむだなものが多いので、普通いらないだろうというものだけにしてみた

問題 41. Lars Wirzenius の提案したのは何か

□ A Debian の品質改善のための提案
□ B Debian のパッケージ削減のための提案
□ C Debian の利用方法の改善のための提案

10.9　2006 年 2 号

http://www.debian.org/News/weekly/2006/02/ にある 1 月 10 日版です。

問題 42. Technical Committee に参加した新メンバーは誰か

□ A Steve Langasek, Anthony Towns, と Andreas Barth

□ B Wichert Akkerman, Jason Gunthorpe, と Guy Maor

□ C Branden Robinson, Kenshi Muto, と Goto Masanori

問題 43. カーネルに存在していた non-free firmware blob については現状どうなっているか

□ A 進展がない

□ B ライセンスを変更することで全て対処した

□ C request firmware というフレームワークによりユーザ空間に移動した

問題 44. apt-get update で gpg エラーが発生した、これは何か

□ A Debian のアーカイブに侵入されたため

□ B Debian アーカイブ署名キーが毎年かわるため、2006 年用のものに変更する必要があった

□ C gpg はもうサポートされていない

10.10　2006 年 3 号

http://www.debian.org/News/weekly/2006/03/ にある 1 月 18 日版です。

問題 45. coldfire とはどういう CPU か

□ A ヒートシンクに液体冷却を採用した CPU の総称

□ B 実は CPU ではない

□ C 組み込み用の CPU で，最近は MMU のあるバージョンもある，m68k と一部バイナリ互換

問題 46. Anthony Towns が宣言したアーカイブの分割とはどういうものか

□ A Debian Project は今後 i386 のみを配布することに決定した.

□ B メインアーカイブは i386, x86_64, ppc のみにして，その他は scc.debian.org で提供する

□ C メインアーカイブは i386 のみにして，それ以外は別のミラーサーバ名で提供するようなインフラを準備する

問題 47. 協調メンテナンスをするためのフレームワークとして重要だと Raphael Hertzog が提案したのは何か

□ A SVN で管理している協調管理のソースの進捗状況をトラッキングするためのツール

□ B 仲よくするための定期的な宴会

□ C 説明書をしっかり書く事

10.11　2006 年 4 号

http://www.debian.org/News/weekly/2006/04/ にある 1 月 24 日版です。

問題 48. Debian のフォーラムをつくろうという提案に関して却下した理由は

□ A フォーラムにはフォーラムの主がいついてしまうからダメだ
□ B フォーラムは 2ch 化してしまうから不適切
□ C メーリングリストの参加者とフォーラムの参加者は質的に違う

問題 49. 1 月 1 日に Anthony Towns が発表した Debian のリリース対象のアーキテクチャは

□ A alpha、amd64、hppa、i386、ia64、mips、mipsel、powerpc
□ B i386, powerpc, amd64
□ C i386, m68k

問題 50. 今後の標準で kaffe で利用する java コンパイラはどれか

□ A jikes
□ B gcj
□ C ecj

10.12　2006 年 5 号

http://www.debian.org/News/weekly/2006/05/ にある 1 月 31 日版です。

問題 51. 2006 年の Debian Day が開催されるのはいつか

□ A 5月１３日
□ B 6月１３日
□ C 7月１３日

問題 52. /var/run/ 以下のサブディレクトリをつかう場合はどうするべきか

□ A パッケージに含める
□ B サブディレクトリは使わない
□ C 起動時に毎回作成する

問題 53. launchpad を Debian の開発に使おうという提案に対して出た反論は

□ A ubuntu の成果なんてつかえない
□ B non-free であるため、それに依存するのはよくない
□ C ウェブインタフェースなんて使いたくない

10.13　2006 年 6 号

http://www.debian.org/News/weekly/2006/06/ にある 2 月 7 日版です。

問題 54. Extremadura のハッキングセッションで何ができたか。

□ A 政治的な思想の熟成
□ B D-I の GUI 版
□ C 日焼け

問題 55. 2006 年の Debian Project Leader 選挙に最初に立候補したのは誰か
□ A Branden Robinson
□ B Junichi Uekawa
□ C Lars Wirzenius

問題 56. FLUG は何か
□ A Finland の Linux Users Group
□ B Finland の Lost Users Group
□ C Finland の Legal Users Group

10.14 2006 年 7 号

http://www.debian.org/News/weekly/2006/07/ にある 2 月 14 日版です。
問題 57. iBook G4 の無線がどうなったか
□ A 仕様が公開された
□ B あいかわらず動かない
□ C 動くようになった

問題 58. 商標について Debian はどういう立場をとっているか
□ A 変更と配布の邪魔にならないようであれば問題ない
□ B 商標は自由の思想に反するので許せない
□ C なにそれ、おいしい？

問題 59. http://lists.debian.org/msgid-search/はどうやって使うのか
□ A http://lists.debian.org/msgid-search/メールのメッセージ ID
□ B http://lists.debian.org/msgid-search/送信者の名前
□ C http://lists.debian.org/msgid-search/メールのサブジェクト

10.15 2006 年 8 号

http://www.debian.org/News/weekly/2006/08/ にある 2 月 22 日版です。
問題 60. Debian etch beta1 インストール用メディアにどういう問題があったか
□ A 最新じゃないのでつかってられない
□ B メディアが水に濡れて使えなくなった
□ C Debian アーカイブの変更の影響で動かなくなった

問題 61. Debian Live Initiative は何をしようとするものか

□ A 新しい開発をがんがんする
□ B Debian の Live CD を統合する
□ C リアルタイムハック実況中継のための環境を提供する

10.16 2006 年 9 号

http://www.debian.org/News/weekly/2006/09/ にある 2 月 28 日版です。

問題 62. ミラーシステムについて Anthony Towns が発表したのは何か
□ A i386 と amd64 だけに限定して今後は運用する
□ B 全アーキテクチャを含めた巨大なミラーを継続
□ C アーキテクチャ毎にミラーを分割する

問題 63. NMU を実施する際に，注意するべきことは何か
□ A BTS を通してメンテナに通知すること
□ B NMU なんてしてる暇があったら自分のバグを直す
□ C できるだけメンテナにばれないように実施する

10.17 2006 年 10 号

http://www.debian.org/News/weekly/2006/10/ にある 3 月 7 日版です。

問題 64. AMD64/kFreeBSD について何がおきたか
□ A はじめてパッケージが動いた
□ B glibc/gcc/binutils がポーティングできた
□ C chroot 内部で動作するようになり，buildd が動いている

問題 65. バックポートのサポートが公式になるのか，という質問についての回答は
□ A Utunubu 広報担当によると Debian はもう時代遅れだ
□ B Joseph Smidt によると，Debian はバックポートを主体として今後は活動を続ける
□ C Norbert Tretkowski によると，公式なサポートつきのバックポートを提供することは考えにくい

10.18 2006 年 11 号

http://www.debian.org/News/weekly/2006/11/ にある 3 月 14 日版です。

問題 66. Bastian Blank が発表した，Debian カーネルチームの作業内容は
□ A kernel-image-という名前から linux-image-という名前に変更しました
□ B カーネルは FreeBSD のものに入れ換えました
□ C 今後は SMP 版と Uniprocessor 版というだけでなく，何 CPU の SMP かということで flavor を分けます

問題 67. Martin の後の安定版リリースマネージャは誰にならなかったか

□ A Martin Zobel-Helas
□ B Andreas Barth
□ C Nobuhiro Iwamatsu

10.19 2006 年 12 号

http://www.debian.org/News/weekly/2006/12/ にある 3 月 21 日版です。

問題 68. JBoss4 の Debian パッケージは存在するか
□ A Guido Guenther が作成したものが存在する
□ B non-free なのでそんなものは存在しない
□ C ボスって何？

問題 69. パッケージに含めるドキュメントの形式は PDF だけにしたい，というメールに対しての反応は
□ A HTML のほうが grep しやすいので，HTML をいれてほしい
□ B DVI のほうがファイル構造が安定しているので DVI にしてほしい
□ C プレインテキスト形式のほうが小さいのでプレインテキスト形式にしてほしい

10.20 2006 年 13 号

http://www.debian.org/News/weekly/2006/13/ にある 3 月 28 日版です。

問題 70. David Moreno Garza が作成したのは
□ A DWN の mixi 風インタフェース
□ B DWN の RSS フィード
□ C DWN の 2ch 風インタフェース

問題 71. google groups を利用して Debian バグを検索するにはどのニュースグループをみればよいか
□ A there.is.no.bugs ニュースグループ
□ B bugs.debian.org ニュースグループ
□ C linux.debian.bugs.dist ニュースグループ

10.21 2006 年 14 号

http://www.debian.org/News/weekly/2006/14/ にある 4 月 4 日版です。

問題 72. ndiswrapper が main に入っているのがただしいのかという議論の原因は何か
□ A ndiswrapper のソースコードは実は全部暗号文で構成されているから
□ B ndiswrapper は思想的におかしいから
□ C ndiswrapper はエミュレーションする対象のドライバが無いと実用できないから

問題 73. Debian Project Leader の投票について Clytie が苦情をいったのは何故か

☐ A Debian Developer でないと投票権がないことに気づかずに投票した
☐ B 立候補者がどれもえらびようがないような人達ばっかりだった
☐ C Branden Robinson が立候補していなかった

10.22　2006 年 15 号

`http://www.debian.org/News/weekly/2006/15/` にある 4 月 11 日版です。

問題 74. Xen の Debian パッケージはもともと Julien Danjou たちと Blastian Blank が別個に作業していた．それが統合されたのはいつか
☐ A 4/5
☐ B 3/1
☐ C 1/1

問題 75. sudo のセキュリティーフィックスでどういう変更がなされたか
☐ A アプリケーションを実際に実行するのではなく，実行しているっぽくみせかけるだけになった
☐ B ルート権限でプログラムを実行しなくなる
☐ C 実行プログラムに引き渡される環境変数を制限

11　Debian Weekly News 問題回答

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

あんどきゅめんてっど でびあん 2006 年夏号

2006 年 8 月 14 日　　初版第 1 刷発行
東京エリア Debian 勉強会 （編集・印刷・発行）