第 5 回 東京エリア Debian 勉強会
事前資料*

Debian 勉強会会場係 上川純一†

2005 年 6 月 11 日

* 機密レベル public: 一般開示可能

† Debian Project Official Developer

目次

1 Introduction To Debian 勉強会

上川純一

今月の Debian 勉強会へようこそ．これから Debian のあやしい世界に入るという方も，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？

目的として下記の二つを考えています．

- メールではよみとれない，もしくはよみとってられないような情報を情報共有する場をつくる
- まとまっていない Debian を利用する際の情報をまとめて，ある程度の塊として出してみる

また，東京には Linux の勉強会はたくさんありますので，Debian に限定した勉強会にします．Linux の基本的な利用方法などが知りたい方は，他でがんばってください．Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作りながらスーパーハッカーになれるような姿を妄想しています．

Debian をこれからどうするという能動的な展開への土台としての空間を提供し，情報の共有をしたい，というのが目的です．次回は違うこと言ってるかもしれませんが，御容赦を．

1.1 講師紹介

- kmuto さん debian-installer ハッカーです．
- えとーさん dsys の開発者です．
- 上川純一 宴会の幹事です．Debian Developer です．元超並列計算機やっていて，今は音楽関係とか，気づいたら canna とか．あと，pbuilder や，libpkg-guide などを通して，Debian の品質向上を目指してます．

1.2 事前課題紹介

今回の事前課題は「debian-installer をつかってみて/今後の debian-installer に期待するもの」というタイトルで 200-800 文字程度の文章を書いてください．というものでした．その課題に対して下記の内容を提出いただきました．

1.2.1 みつかさん

少し前に使ったきりなので記憶が曖昧なのですが…。まず驚いたのが国際化が進んだなということです。各国語のメニューがずらっと並んだなかに日本語も含まれているというのはなんだかとても誇らしいような気持ちになりました。日本語を選択して進んでいくと普通にすんなりとインストールできてしまい感動しました。以前のインストーラーと比べて、ステップ数も少なくなったと思います。これも、インストールが簡単になったなという印象に寄与していると思います。

今後への期待としては、日本語を選択してインストールした場合

- jfbterm がデフォルトで起動したり
- ja_JP.eucJP 辺りがデフォルトで設定されてたり

すると嬉しいかなあとか思いました。

1.2.2 kaepapa

debian のインストーラは potato と woody そして sarge で使ってきたが徐々にインストールが容易なものになってきたと思います。個人的には woody の時のインストーラで十分満足できる状態でした。sarge になり cfdisk でなくなりちょっと残念です (経緯は知っている)。partitioner に関してはちょっと間違うと HDD 全域をパーティショニングしてしまうリスクを感じました。この点は十分な情報の提供が必要だと思われます。RC3 当時の partitioner の不具合 (パーティションテーブル不全？) もなくなり (20050419) リリースされたらもう一度ためしてみたいと考えています (VMWare ゲスト)。またパッケージの選択等については以前よりインストーラでは最小構成のみインストールし後から aptitude で必要なものを導入するという癖がついているためあまり記憶に残っていませんが以前とかわらず最小構成でインストールが終れることに好感が持てます。このポリシーはずっと継続して欲しいと願っています。

1.2.3 **やまねさん**

使ってみた感想は昔に web に載せたことがあるので略。その頃からするとぐっとよくなってます :-)

- 改善して欲しいーという点。
  - やはり「X の自動認識」の弱さが気になる。初心者には鬼門。
  - パーティションの分割し辛さと RAID 機能の弱さ
  - パッケージダウンロードをしている間に他の設定をさせるなどは難しいのか? ずーっと待ってるのはちょっと。
  - 特定機種での問題（Dell のとか）は結局解決したんだろうか。
  - 「定期的なアップデート」。たとえ stable といえど、新しい機種に対応していないインストーラーを勧められるのは「？」というところ。ここはインストーラだけでも半年に 1 回ぐらい更新を行うようなポリシーに切り替えて欲しい。
  - iptables でパケットをデフォルト遮断は難しい? WindowsXP SP2 なども基本は『保護』の方向だし、この機能は欲しい
  - tasksel の分け方とか、本当に「最小インストール」するのが簡単になって欲しいところとか
- 思いつき
  - イベントのときに「d-i fest」でひたすらインストールできた・できない機種の情報収集するとか。
    - BTS 報告だと英語がネックで厳しいし
    - こんなイベントなら参加の敷居も低いし
    - レアマシン持ってくる人もいて、結構なデータ取れるかもしれないし
    - エスパー能力なくてもその場で解決できること多そうだし

ということで如何でしょうか :-)

1.2.4 **中島　清貴さん**

Debian 辞典という本がちかくの本屋さんに平積みで売っていたので、さらっと立ち読みしてみたらインストール方法から何からすべて書いてあるではないか。これだ、この本さえあれば今すぐ Debian が使えるはずだ。さっそく買って家で読んでみる。まったく Debian なるものとはかけはなれた業界にいるような私にも分かるような事細かな説明。project の組織構成まで書いてある。これは良書に間違いない。 そこで友人 N を電話でモスバーガによびだして彼にも買わせる。附属の CD に特制インストローラが付いているので、これでインストールはだいじょぶなはずだ。このような本が読めることは Debian を知っている私としての最高の祝福だと思う。

1.2.5 **澤田さん**

Debian-Installer を使ってみて感じたこととして、woody 時代に比べて Linux を知らない人でもインストールが容易になったなぁということがある。大学時代にサーバを作ろうということで Debian をインストールするというアレゲな実験があり TA をやっていたわけだが、多くの人がはまっていたこととして、

- パーティションの設定
- ドライバの組み込み
- パスワードを打っても何も表示されないこと

ってのがあった。Debian-Intaller ではここら辺が改善されているのでいい感じだ。

また、はじめに日本語を選択するとキーボードとかタイムゾーンとかパッケージとかを日本向けにしてくれるのもとってもフレンドリーに感じた。

逆にちょっとあれかな〜と思ったのはネットワーク設定のデフォルトが DHCP な点である。ここは static にするか dhcp にするかの選択肢がほしかったなと思った。

1.2.6 Ryoichi Kato

第 5 回，Debian 勉強会に参加するのにあたり, 実際に使ってみて感想を書いてみようと思い, 久々に真っさらの状態から debian のインストールをしてみました. 手元に潰してもいいマシンがなかったので, CD image (sarge-i386-netinst.iso) から VMWare 上にインストールしました.

普段あまり debian-installer を使う機会もなく, 前回 debian をインストールしたのは 2,3 年前ですが, その頃の印象と比べると非常に使いやすくなっていると感じました. 以前あった「クセ」, 「アク」(コツを知らないと使いこなせない) のようなものがずいぶん薄くなっていて, 次何をすれば良いんだろう」と迷う必要が殆んどなくなったという印象です. また, 特にパーティショニング関連は, 細い設定をするのややり直しをするのがかなり楽でした.

ところで, grub をインストールして再起動するところまではすんなり言ったのですが, その後の再起動でちょっとトラブルに見舞われました.

VMware のディスク領域を 2G しか確保してなかったのですが, base-config で ”desktop environment” を選択しておいたらパッケージをインストールしている最中に disk full になってしまいました. (最初に必要な領域が 600MB + 1700MB 位とか出たのですが, 空き容量チェックはしてないのでしょうか?) そのままにっちもさっちも行かなくなったので, そこで二枚目のコンソールへ移って

```
# rm /var/cache/apt/archives/*.deb!
```

してから手動で apt-get でインストールを続行しようとしたら, /var/cache/debconf/config.dat が lock されてると言われてしまいそのままでは手動でインストールできませんでした.

結局 base-config を kill -STOP (respawn なので) することでなんとか無理矢理 install してみましたが, このような場合はどうやるのが正解だったのでしょうか?

1.2.7 小林儀匡さん

以前の boot-floppies インストーラでも質問に従っていけばインストールは問題なくできましたが、

1. 質問やメッセージなどの数がやたらと多いこと、
2. それらがすべて英語で書かれていること、
3. それらの中には初心者にとってよく分からないものも多いこと

などが理由で、使い勝手はあまりよくなかったように思います。「一応インストールしてみたものの、全てがまともに設定されたかはやや不安だ」というような心地に、(少なくとも以前の自分は) なりました。そのため、新しい debian-installer (d-i) で

1. 多くのプロセスが簡潔化されたこと、
2. 質問やメッセージもすべて翻訳されたこと、
3. 初心者がどうしていいか悩むような設定 (例えばパーティション分け) にいくつかの簡単な選択肢が用意されたこと

を非常に嬉しく思います。

d-i のおかげで、サービスをほとんど入れる必要がない (tasksel で何も選択しなくて済む) 数値計算用のマシンなどは、たった 10 分でインストールが完了しました。また、まだ挑戦したことはありませんが、プレシードを利用して多数のマシンに一度に Debian を入れることが簡単にできるようなので、多数のホストからなる計算機環境を作り上げる機会があったら試してみたいものです。

一度インストールしてしまえば再インストールを必要とせず、ハードウェアが壊れるまで使い続けられるのを売りにする Debian なので、「Debian ユーザ」になった人にとってインストールはあまり重要ではないといっても過言ではないでしょう。しかし一方で、そのインストールの敷居が高いために「Debian ユーザ」になるのを挫折するユーザもいると思います。d-i によってそのような人に Debian の恩恵を与えられることを期待するとともに、コミュニティへの新規参入者のためとも言えるようなインストーラをわざわざ作った開発者の方々に感謝します。

自分にとって d-i は既に十分に使いやすいレベルに達しているため、今後の d-i に欲しい新機能や改善点などは今のところ思い付きません。あえて挙げるとすれば、Windows や Mac の世界から来た初心者の多くには GUI のほうがとっつきやすいと思うので、GUI 化が実現できるとよいかもしれません。もっとも、CUI に慣れた人間からすると GUI はいろいろと面倒なので、たとえ GUI 化されても、CUI と GUI のどちらを使うかを最初に選べるようになっていると嬉しいです。

1.2.8 えとーさん

サーバなどにインストールすることも何度かあったのですが、その際に Software-RAID、LVM の構築があまり好ましくなく、いろいろと不便でした。

よって、dm,md への対応の強化が行なわれることが好ましいと思っています。

他にはカスタムインストーラを手軽に作れるようになると嬉しいなぁと思っています。これは、ノート PC や、サーバ機器などの、ハードウェア構成がある程度固定化しているものについては、一部設定を省けるようなインストーラが欲しいからです、例えば、ハードウェアの設定部分をテキストファイルから読み込めるようにしておき、テンプレートに従って記述したテキストファイルをインストーラに追加しておけば、インストーラの画面から型番等を選択することにより設定終り。

パッケージ選択についても同様でもハードウェアによって必要なパッケージを探すのが面倒ですので環境に合わせたパッケージのタスクみたいのを選択できるようになっていて、そこからも選べるようになるといいなぁと思います。

1.2.9 岩松さん

debian-installer を使ってみての感想 Debian 辞典の CD を使ってみました。

1. インストール文章が全て日本語。すばらしい。
2. ある程度自動的にパーティションを切ってくれるシステム（ガイドによるパーティショニング）が手間を省いてくれてよい。

期待すること

1. マウスを使えるようにして容易にインストール。（初心者向け）
2. XWindows の設定を容易にしてほしい。
3. 日本語でインストールしたときは FB でも日本語表示できるようにしてほしい。(現状で文字化けする。)

1.2.10 上川

debian-installer は実は使ったことがない．debian-installer によってリプレースされた boot-floppies の開発には参加していた．boot-floppies は使っていたが実際 CD から起動してシェルを立ち上げるためのみに利用して，あとは debootstrap とかを利用し，手動でインストールしている．

さて，debian-installer は etch にむけて今後どうなっていくことが期待できるだろうか．

まず，言語の選択のよりよい流れが欲しい．日本語をインストール画面で選択したら，コンソール画面も日本語で進み，インストールし終ったときのデフォルトの画面も日本語になるようにしたい．これを実現するには 下記の二つが実現したら楽になるのではないだろうか．

- `ja_JP.UTF-8` locale であらゆるアプリケーションが標準で動くようになること
- Linux カーネルが utf-8 での日本語の表示対応したフレームバッファをドライブできるようになること

etch としてあと一年くらいハックできる期間があるのだから，これくらい実現できそうな気がするのだが，どうだろう？

2 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．

2.1 2005 年 21 号

問題 1. 12W の電力消費で動くコンパクトな Debian サーバを準備するために，Silas Benett はどうしたか

□ A となりの家の犬を無線 LAN ステーションにした

□ B Mac Mini をかってきて，CDROM ドライブをとりはずしてバッテリーを装着した

□ C Dual Xeon のサーバを購入して Xeon を一つにへらしてみた

問題 2. Michael Banck は Hurd で何をしたか

□ A GNOME と QT を動かした

□ B もうあきらめようと宣言した

□ C tuxracer を動かした

問題 3. Orphan されたパッケージなどの一覧が出て来る WNPP メールは今後どうなるか

□ A 毎週 debian-devel-announce にメールを投げていたのを停止して，debian-wnpp に移動する

□ B もう orphan なんてものは存在しない

□ C orphan したパッケージは今後は無視する

問題 4. Nico Goldeh は unrar パッケージのバージョンがずいぶんさがっていることに気づきました．その理由は

□ A もともとの unrar は non-free で，free な実装ができたのでそちらに移行した

□ B unrar の開発は後向きにすすんでいる

□ C unrar なんて知らないので関係ない

問題 5. Waste パッケージにはどういうライセンス上の問題があるか

□ A ソフトウェアとしてつかいものにならない

□ B 開発元が一旦 GPL でリリースしたが，revoke した

□ C ライセンスが入っていない

問題 6. Debian woody のアップデートが出たが，そのバージョンは
□ A 3.0r6
□ B 3.0.6
□ C 3.0-RC6

2.2 2005 年 22 号

問題 7. Andreas Barth が BTS の LDAP ゲートウェイの高速化のために試したのは
□ A Archived バグが多すぎるので，Archived バグを分離してしまいメモリ負荷を減らす
□ B LDAP をやめて SQL にする
□ C 不要っぽいバグはなかったことにする

問題 8. Philipp Kern は Debian Archive に video セクションを追加しようと提案しました．video セクションが無いため現状は各ビデオ関連のパッケージは
□ A どこのセクションにも属していない
□ B libs セクションに投げ込まれている
□ C 該当するアプリケーションは graphics セクションとデスクトップ環境のセクションに混在している

問題 9. debian-legal のサマリーページについて Frank Lichtenheld はもう削除しようと提案しました．その理由は
□ A 使えないから
□ B 面白くないから
□ C 合意がとれないのであたらしくサマリーが作成できずメンテナンスが不可能

2.3 2005 年 23 号

問題 10. Debian のリリースをおくらせる必要がある，と主張していた KDE のバグは何？
□ A webcollage スクリーンセーバがデフォルトで動作するようになっているので，オンラインのポルノ画像が表示される可能性がある
□ B KDE が動かない
□ C KDE の壁紙が Debian 向けではない

問題 11. Debian GNU/Linux 3.1 がリリースされたのは？
□ A 2005 年 6 月 6 日 (日本時間 6 月 7 日)
□ B 2005 年 7 月 1 日 (日本時間 7 月 2 日
□ C 2005 年 1 月 1 日 (日本時間 1 月 2 日)

問題 12. 3.1r0 の CD セットの問題は

□ A だれも焼けないような枚数になった

□ B デフォルトインストールの sources.list の security.debian.org 向けの行が有効でなかった

□ C 大きすぎて世界中のネットワーク帯域をくいつぶしてしまう

問題 13. Wesley Landaker は GPG キーサインのための確認に実際に会わなくてもできる基準を提案しました．その反論は

□ A GPGって出会い系サイトでしょ？

□ B 写真のみで確認するのであれば，写真は偽造しやすいので，その情報を信頼するのは難しい

□ C 会わなくてよいのだったら GPG は面白くない

問題 14. ライブラリの ABI が変更になった場合に気を付けることは何か

□ A ライブラリパッケージ名を変更しておけば管理者が明示的にアンインストールするまでは古いライブラリバージョンもシステムに残る

□ B 古いライブラリにリンクしてあるプログラムを確実に全部動かなくする

□ C 変更履歴に怒りのコメントを記述する

3 最近の Debian 関連のミーティング報告

上川純一

3.1 東京エリア Debian 勉強会 5 回目報告

前回開催した第 5 回目の勉強会の報告をします.

DWN クイズをしました.

松山さんが DFSG について語りました.

岩松さんが dpkg-cross の使い方について説明しました.

上川が lintian, linda の使い方について説明しました．Debian のパッケージの作成のフローを一度デモできるとよいねぇ，という話しになりました.

グループディスカッションでは，今後のアーキテクチャ対応をどうしましょうか，という話しをしました．現在のアーキテクチャは CPU 毎にわけられており，CPU 以上の部分についてはパッケージでカバーしているのですが，task-ibookg4 とかがあれば，各ハードウェアに必要なパッケージをごっそりいれれてよいよねぇ，という案が出たりしました．全パッケージをスクリプトで再実装してしまい，各アーキテクチャの差分を減らす，という迷案も出ていました.

4 alternatives -選択せよ-

えとー

4.1 alternatives とは？

パッケージ管理というと依存関係の管理についてのみがクローズアップされるが、パッケージ管理としては、それだけでは不足する部分がある、それを補うものの一つが alternatives です。

日本語訳すると「選択肢」、私の定義ですが、「複数のパッケージを特定機能ごとに一つにまとめて扱い機能ベースのパッケージの管理を提供するもの。」「パッケージを機能の視点で管理する。」「プログラムに OS の統一的な API を提供する。」の 3 点です。

機能としては、「類似の機能を持つプログラムの別名を提供する。」で、symlink を使って実現しています。alternatives とは、その symlink を管理するための機構と言えるでしょう。

なぜわざわざ別名が必要？

1. プログラムからの使い勝手の向上万単位のパッケージの存在する Debian では同じような機能を持つ別々のパッケージが複数存在することは珍しくありません。伝統的に Unix 系 OS ではパイプやリダイレクトなどやスクリプト、プログラムなどから他のプログラムの機能を利用することによりコード量の削減と質の向上を図っています。しかし、プログラムから呼び出す際に同じような機能をもつもの全てを把握し条件分岐などで呼んで行くのはコストから言って現実的ではありません。その解決策としてはプログラムへの統一的な API を提供することができるようすることです。awk スクリプトやコンパイラなどを思い浮べるといいと思います。
2. ユーザへの使い勝手の向上ユーザからしても同じ機能なのに別の別の名前であると不便な場合もあります。MUA や IRC クライアント からの Web Browser を呼び出す際を想像してもらえば想像し易いと思います。
3. menu との連携 Debian 独自なものとして menu がありますが、これと alternatives を連携させることによりメニューをいじらずに、違うパッケージを提供することができます。

4.2 update-alternatives の使い方

see man(重要)

alternatives の制御には /usr/sbin/update-alternatives というコマンドを使います。

```
書式一覧
 update-alternatives --install リンク 一般名 パス 優先度  [--slave スレーブリンク スレーブ一般名 スレーブパス]
 update-alternatives --remove 一般名 パス
 update-alternatives --remove-all ディレクトリ
 update-alternatives --all
 update-alternatives --auto 一般名
 update-alternatives --display 一般名
 update-alternatives --list 一般名
 update-alternatives --config 一般名
 update-alternatives --set 一般名 パス
```

4.3 用語の説明

4.3.1 マスター

- マスター alternatives の対象とするもの
- リンク 一般的なリンクです
  例 /usr/bin/awk, /usr/bin/editor, /usr/bin/pager, /usr/bin/x-www-browser
- 一般名 一般的な名前です、
  例 awk, editor, pager, x-www-browser
- パス それぞれの alternatives のリンク先になります
  例 /usr/bin/awk: /usr/bin/nawk, /usr/bin/mawk, /usr/bin/gawk /usr/bin/editor : /bin/ed, /bin/nano, /usr/bin/vim /usr/bin/pager : /bin/more, /usr/bin/less, /usr/bin/w3m, /usr/bin/lv /usr/bin/x-www-browser : /usr/bin/mozilla, /usr/bin/kazehakase, /usr/bin/mozilla-firefox
- 優先度 auto モードで設定される際の選択基準になる整数値です

4.3.2 スレーブ

- スレーブ alternatives のマスターに付随するファイル、man など 複数指定可能

- スレーブリンク 一般的なリンクに付随するリンク
  例 /usr/share/man/man1/awk.1.gz, /usr/share/man/man1/nawk.1.gz, /usr/share/man/man1/editor.1.gz, /usr/share/man/man1/pager.1.gz, /usr/share/man/man1/x-www-browser.1.gz
- スレーブ一般名 一般的な名前に付属する一般名
  例 awk.1.gz, editor.1.gz, pager.1.gz, x-www-browser.1.gz
- スレーブパス それぞれの alternatives に付随するリンク先
  例
  /usr/share/man/man1/awk.1.gz, /usr/share/man/man1/nawk.1.gz : /usr/share/man/man1/mawk.1.gz, /usr/share/man/man1/gawk.1.gz
  /usr/share/man/man1/editor.1.gz : /usr/share/man/man1/ed.1.gz, /usr/share/man/man1/nano.1.gz, /usr/share/man/man1/vim.1.gz
  /usr/share/man/man1/pager.1.gz : /usr/share/man/man1/more.1.gz, /usr/share/man/man1/less.1.gz, /usr/share/man/man1/w3m.1.gz, /usr/share/man/man1/lv.1.gz
  /usr/share/man/man1/x-www-browser.1.gz : /usr/share/man/man1/mozilla.1.gz, /usr/share/man/man1/kazehak
  /usr/share/man/man1/mozilla-firefox.1.gz

4.3.3 automatic と manual

alternatives は優先度によって自動的に選択する automatic モードと優先度を無視しユーザが選択する manual モードがあります。デフォルトは automatic モードで –config や –set 、 –all を使い変更した場合に manual モードに以降します。automatic モードに戻したければ –auto を使います。

4.4 ユーザとして使う場合

ユーザとして主に使用するコマンド

4.4.1 状態表示 (一般ユーザで可)

```
update-alternatives --display 一般名
update-alternatives --list 一般名
```

4.4.2 設定変更 (root のみ)

```
update-alternatives --auto 一般名
update-alternatives --config 一般名
update-alternatives --set 一般名 パス
```

の 4 つです。

4.4.3 使用例と説明

```
# update-alternatives --display editor             <-- editor という一般名の alternatives のステータスを表示
editor - status is auto.                           <-- alternatives の モード
 link currently points to /usr/bin/vim             <-- 現在のリンク先
/bin/ed - priority -100                            <-- マスターリンク - 優先度 (/bin/ed)
 slave editor.1.gz: /usr/share/man/man1/ed.1.gz    <-- スレーブ一般名 : スレーブリンク先 (/bin/ed)
/bin/nano - priority 40                            <-- マスターリンク - 優先度 (/bin/nano)
 slave editor.1.gz: /usr/share/man/man1/nano.1.gz  <-- スレーブ一般名 : スレーブリンク先 (/bin/nano)
/usr/bin/vim - priority 120                        <-- マスターリンク - 優先度 (/usr/bin/vim)
 slave editor.1.gz: /usr/share/man/man1/vim.1.gz   <-- スレーブ一般名 : スレーブリンク先 (/usr/bin/vim)
Current 'best' version is /usr/bin/vim.            <-- 優先度の一番高いもの

# update-alternatives --list editor                <-- editor という一般名の alternatives のマスターリンク先を表示
/bin/ed                                            <-- マスターリンク先
/bin/nano                                          <-- マスターリンク先
/usr/bin/vim                                       <-- マスターリンク先

# update-alternatives --auto editor                <-- 一般名 editor の alternatives を優先度によって変更
# update-alternatives --config editor              <-- 一般名 editor の alternatives を選択肢から手動で変更

There are 3 alternatives which provide 'editor'.   <-- editor という一般名の選択肢が 3 つある

   Selection    Alternative
-----------------------------------------------
     1        /bin/ed                              <-- alternatives
     2        /bin/nano                            <-- alternatives
**   3        /usr/bin/vim                         <-- alternatives 現在選択されている

Press enter to keep the default[*], or type selection number:  <-- ここで番号を選択する
Using '/usr/bin/vim' to provide 'editor'.          <-- editor を /usr/bin/vim で提供する

# update-alternatives --set editor /usr/bin/vim    <-- 一般名 editor の alternatives を指定したリンクに変更する
Using '/usr/bin/vim' to provide 'editor'.          <-- editor を /usr/bin/vim で提供する
```

4.5 パッケージメンテナとして使う場合

パッケージメンテナな人が主に使うコマンドパッケージインストール時に使用 (主に postinst)

```
update-alternatives --install リンク 一般名 パス 優先度  [--slave スレーブリンク スレーブ一般名 スレーブパス]
```

パッケージ削除時に使用 (主に prerm)

```
update-alternatives --remove 一般名 パス
```

4.5.1 使用例

```
update-alternatives --install コマンド

update-alternatives --install リンク 一般名 パス 優先度 [--slave スレーブリンク スレーブ一般名 スレーブパス]
```

この書式ですが、slave は省略可能で、使用する場合には複数のスレーブを指定することもできます。

vim の postinst

リンク元のファイルがインストールされてからリンクを貼るので postinst に書く

```
case "$1" in
  abort-upgrade)
    for i in vi view ex editor ; do
      update-alternatives \
        --install /usr/bin/$i $i /usr/bin/vim 120 \
        --slave /usr/share/man/man1/$i.1.gz $i.1.gz /usr/share/man/man1/vim.1.gz

    done
    ;;
  configure)
    for i in vi view ex editor ; do
      update-alternatives \
        --install /usr/bin/$i $i /usr/bin/vim 120 \
        --slave /usr/share/man/man1/$i.1.gz $i.1.gz /usr/share/man/man1/vim.1.gz

    done
    if [ -L /usr/doc/vim ] ; then
      rm /usr/doc/vim
    fi
    ;;
esac
```

vim の prerm

リンク元のファイルがなくなる前に削除するので prerm に書く

```
case "$1" in
  remove)
    for i in vi view ex editor ; do
      update-alternatives --remove $i /usr/bin/vim
    done
    ;;
esac
```

4.6 マニアックなコマンド

update-alternatives –all

全ての選択肢に –config を使い設定を行なう

update-alternatives –remove-all

ディレクトリにある全ての選択肢を削除する（危険です)

4.7 一般的なオプション

```
--verbose 冗長メッセージ
--quiet メッセージを抑制
--test テスト（未実装）
--help ヘルプ
```

4.8 マニアックなオプション

–altdir

リンクを置くディレクトリ

デフォルトは /etc/alternatives/

–admindir

設定ファイルを置くディレクトリ

デフォルトは /var/lib/dpkg/alternatives/

4.8.1 マニアックなオプションの使用例

/usr/local/以下にソースからインストールしたものや alternatives を提供していないパッケージについて alternatives で管理したい場合などに使う 特権ユーザでなくても使えるのも特徴

使用例: /home/foo/eclipse 以下にある /home/foo/eclipse/eclipse を alternatives で管理したい。

```
$ mkdir /home/foo/altdir/
$ mkdir /home/foo/admindir/
$ mkdir /home/foo/bin/
$  /usr/sbin/update-alternatives --altdir /home/foo/altdir/ --admindir /home/foo/admindir/ \
      --install /home/foo/bin/eclipse eclipse /home/foo/eclipse/eclipse 100
```

で、.bashrc などに /home/foo/bin/ を追加 IM とか、MUA の管理にも向いているのではないだろうか。

4.9 update-alternatives ってどうなってるの?

4.9.1 構造

/usr/sbin/update-alternatives

alternatives の制御コマンド

perl で実装されており dpkg パッケージに含まれています。

/var/lib/dpkg/alternatives

alternatives 設定ファイルディレクトリ

設定ファイル例

```
$ cat /var/lib/dpkg/alternatives/editor
alternatives 設定ファイルディレクトリ
/usr/bin/editor                         <-- マスターリンク先
editor.1.gz                             <-- スレーブ一般名
/usr/share/man/man1/editor.1.gz         <-- スレーブリンク先

/bin/ed                                 <-- マスターリンク元 (/bin/ed)
-100                                    <-- 優先度 (/bin/ed)
/usr/share/man/man1/ed.1.gz             <-- スレーブリンク元 (/bin/ed)
/bin/nano                               <-- マスターリンク元 (/bin/nano)
40                                      <-- 優先度 (/bin/nano)
/usr/share/man/man1/nano.1.gz           <-- スレーブリンク元 (/bin/nano)
/usr/bin/vim                            <-- マスターリンク元 (/usr/bin/vim)
120                                     <-- 優先度 (/usr/bin/vim)
/usr/share/man/man1/vim.1.gz            <-- スレーブリンク元 (/usr/bin/vim)
```

/etc/alternatives/

alternatives のリンクのあるディレクトリです。

リンク例

```
$ ls -l /etc/alternatives/editor
lrwxrwxrwx  1 root root 12 2005-06-01 02:43 /etc/alternatives/editor -> /usr/bin/vi
```

```
/usr/share/man/man8/update-alternatives.8.gz
/usr/share/man/de/man8/update-alternatives.8.gz
/usr/share/man/es/man8/update-alternatives.8.gz
/usr/share/man/fr/man8/update-alternatives.8.gz
/usr/share/man/ja/man8/update-alternatives.8.gz
/usr/share/man/pt_BR/man8/update-alternatives.8.gz
```

alternatives マニュアルです。

4.9.2 動作

update-alternatives –auto

1. 一般名を取得
2. /var/lib/dpkg/alternatives/一般名 のステータス情報が manual の場合は auto に変更

3. /etc/alternatives/一般名 のリンク先を /var/lib/dpkg/alternatives/一般名 の優先度の情報を比較し最大のものへのリンクへと変更

update-alternatives –config 及び update-alternatives –set コマンド

1. 一般名とリンク先のパスを取得
2. /var/lib/dpkg/alternatives/一般名 のステータス情報が auto の場合は manual に変更
3. /etc/alternatives/一般名 のリンク先を指定されたリンク先のパスへと変更

update-alternatives –install コマンド

1. 存在しない場合は指定されたパスのリンクを /etc/alternatives/一般名 へと貼る
2. 存在しない場合は /var/lib/dpkg/alternatives/一般名 ファイルを指定された情報に基ずいて生成
3. /etc/alternatives/一般名 のリンク先を /var/lib/dpkg/alternatives/一般名 の優先度の情報を比較し最大のものへのリンクへと変更

update-alternatives –remove コマンド

1. /var/lib/dpkg/alternatives/一般名 から指定された パスの情報を削除
2. /var/lib/dpkg/alternatives/一般名 で alternatives が提供されない場合は/var/lib/dpkg/alternatives/一般名 及び /etc/alternatives/一般名リンク先を削除
3. /var/lib/dpkg/alternatives/一般名 で まだ alternatives が提供されている場合は/etc/alternatives/一般名 のリンク先を /var/lib/dpkg/alternatives/一般名 の優先度の情報を比較し最大のものへのリンクへと変更

4.10 update-alternatives の改善案

update-alternatives 関連の改善案を自分なりに考えてみた。

1. パッケージのファイル情報への反映
   現在の alternatives はパッケージをインストールされるまではそのパッケージがどんな alternatives を提案するのか という情報が解らない。これだと、インストールしてから、普段使用していた alternatives を使おうとしたら期待してたのと別のものが起動してしまい戸惑うことになるし、無駄なハマりの原因となるし、もし、そのような情報を集積できれば、機能からパッケージをより簡単に検索することができるようになる。
2. より柔軟に
   現在の alternatives では、man を slave にした場合などに英語のみしか表示できなくなってしまう、多言語化が進みつつある昨今これだけでは不足ということになってしまっているので、より柔軟な設定を行なえるのが望ましい。ファイルだけではなくコマンドを登録できるようにするなど。
3. menu 編集インターフェースとの連携
   独自に作成した alternatives などを menu に簡単に登録できるようにするなどの、応用的な使い方を簡便にするインターフェースの拡充

4.11 dsys について

4.11.1 目的

dpkg に含まれる パッケージ依存関係管理を行なう /usr/bin/dpkg 以外の/usr/sbin/update-alternatives, /usr/sbin/dpkg-divert, /usr/sbin/dpkg-statoverride の 3 つのコマンドに対応した GUI フロントエンドを提供することを目指している。

4.11.2 歴史

一番初期は update-alternatives のみに対応したもので、ruby 1.6 と ruby-gtk で書いていたが、やっぱ gtk2 だろう！ ということで、ruby-gtk2 に手を出してのんびりしてるうちに galternatives など競合で出てきてしまった。思い出したら実装したりしているがコーディング能力の低さのために遅々として改善されていない。

4.11.3 機能

/usr/sbin/update-alternatives
一覧の表示、個々のステータスの表示、変更、追加、削除
/usr/sbin/dpkg-divert
一覧の表示
/usr/sbin/dpkg-statoverride
一覧の表示、変更、追加、削除

4.11.4 TODO

まずはツールチップを適切に実装すること綺麗なコードの書き方や、デザインがわからないのでどうにかするせめて man くらい、、alternatives と statoverride のインターフェースを意識的に替えていたがそろそろ統一 treeview で右クリックできるようするショートカットの実装

4.11.5 将来

設定ファイルを作るかもアイコン欲しいよね
協力者の方募集中、、

5 Inside Debian-Installer

武藤 健志

5.1 Debian-Installer とは

Debian-Installer（以下 **d-i**）は、Debian GNU/Linux リリース 3.1、コードネーム Sarge から採用された新しいインストーラシステムです。

これまでのインストーラシステム（Woody 以前）として採用されていた Boot-Floppies には、次のような問題がありました。

- サイズの制限。その名のとおりフロッピーディスクを基盤にしたものなので、システム全体をフロッピー容量の 1.2MB〜1.44MB に圧縮して収まるように悪戦苦闘しなければならなかった。
- 動的ロードの不備。動的にインストーラを拡張する方法がなく、機能向上を行うすべがなかった。
- ハードウェア認識の不備。ハードウェアを自動認識する機構が入っていなかった。ユーザーはすべて手動で設定しなければならなかった。
- インストーラのリリースとテスト。インストーラ一式が一体のため、リリースするためには全体をビルドしなければならず、頻繁にリリースすることができなかった。バグへの対処が遅れがちになった。

2000 夏ごろ	Joey Hess がデザインを始める
2000 秋ごろ	サンプル実装が公開される
2002	Woody が Boot-Floppies に基づいてリリース。Sarge で d-i になるよう本格的に目標が立てられる
2003	開発がかなり進展。開発者も増える
2004	d-i rc2 リリース
2005 春	d-i rc3 リリース
2005 夏	Sarge リリース

表 1　d-i の簡単な歴史

5.2 開発体制

d-i の開発は、ほかのどの Debian のサブプロジェクトよりも大規模です。

- リポジトリのコミッタ権限がある人：153 人
- ローカライズコーディネータのコミッタに代行してもらっている人もいるので、実際はもっと多い？
- オフィシャルな Debian Developer は 60 人

- アカウント管理:Alioth (`http://alioth.debian.org/projects/d-i/`)、メーリングリスト:debian-boot@lists.debian.org、バグ追跡:installation-reports 仮想パッケージ名、IRC:#debian-boot

プロジェクトリーダ	Joey Hess
国際化	Christian Perrier、Dennis Stampfer
フロントエンド	Collin Watson
パーティションマネージャ	Anton Zinobiev
ネットワーク設定	Joshua Kwan
ハードウェア認識	Petter Reinholdtsen
移植	Martin Michlmayr、Steve Langaek、…
ドキュメント	Frans Pop

表 2　主な開発者

当初は CVS、現在は Subversion にて協業を行っています。

- `http://svn.d-i.alioth.debian.org/svn/d-i/trunk`（トランク、unstable 向け）
- `http://svn.d-i.alioth.debian.org/svn/d-i/branches/d-i/sarge`（フリーズされた Sarge のブランチ）

```
d-i/
    installer/build/ ——d-i のビルドディレクトリ（make ... を実行すると d-i イメージができる）
    installer/doc/   ——ドキュメント。manual/に XML 形式のインストールマニュアルが収録されている
    packages/        ——各 udeb のソースコード
    scripts/         ——自動化などに使うスクリプト集
```

図 1　d-i リポジトリの構造

5.3　d-i の構造

d-i は、これまでに蓄積されてきたさまざまな Linux/Debian のフレームワークを応用しています。

udeb　通常の deb から実行に不必要なものを削除してスリムにしたもの
cdebconf　C で記述された debconf フレームワーク (いくらか拡張)
devfs　ハードディスクなどのデバイスファイルの取り扱いを容易にする
フレームバッファと国際化端末　国際化端末を実行する (bogl-term および jfbterm)
discover と hotplug　ハードウェアの自動認識

5.3.1 udeb

udeb は、通常の deb とほぼ同じですが、ドキュメントなどの実行に直接関係ないものを削ぎ落としたバイナリパッケージです。

- インストーラだけで利用する特別なメタ情報として、XB-Installer-Menu-Item が含まれているものがある。これは、インストーラのメニューに表示する際の順番を示す
- Depends や Provides で依存関係を処理したり、debconf 用の国際化済み templates によってローカライズされたメニューを表示できる (メニューに表示されるのはパッケージの short description だが、特別に debconf templates の中でこのローカライズも設定している)
- インストーラのメニューを選択することは、postinst を実行するということになる
- 環境によってロードする順序を変えることができる 。たとえば、ネットワークインストールなら最初にネットワークの設定を済ませてから IDE コントローラを検出、CD インストールなら最初に IDE コントローラの設定を済ませてからネットワークを検出

anna	擬似 APT
udpkg	マイクロ dpkg
languagechooser、countrychooser	言語・国情報の設定
hw-detect、ethdetect、hw-detect-full	ハードウェアの検出
preseed	事前構成のロード
netcfg	ネットワーク設定
network-console	SSH サーバーを起動してリモートインストールを実行する
cdrom-retriever、floppy-retriever、net-retriever	udeb パッケージをダウンロードする
partman	パーティションマネージャ
base-installer	ベースシステムを debootsrap で展開し、CPU に合ったカーネルをインストールする
os-prober	インストールされているほかの OS の検出
prebaseconfig	CD の取り出し、現在の状態の保存など

表 3　主な udeb

リポジトリの packages/だけでなく、libc や discover、console-tools のように、外部の deb パッケージが udeb を提供することもあります。

d-i は udeb の集合体で、最初にロードされるインストーラも udeb を基に構成されています。d-i の make 時にどの udeb から構成しておくかによって、CD 向け、ネットワーク向け、USB メモリ向けといったイメージを作成できます。

5.3.2 cdebconf

cdebconf は、Perl で記述されていた debconf を、C で実装し直したものです。

- Perl の膨大なアーカイブの必要なく動作
- 進捗バーなどの機能拡張

今後は普通の環境も、debconf は捨てて (orphan 済み)、cdebconf に移行しようという計画になっています。d-i では cdebconf が大きな役割を果たしています。

- debconf インターフェイスにのっとったフロントエンドを切り替えることができる (デフォルトは dialog)
- templates からインストールの質問を表示し、答えをデータベースに格納する
- 質問に優先度 (critical、high、medium、low) を設定して、質問の詳細と数を変えられる (デフォルトは high)
- データベースに最初から値を入れるようにして、全自動インストールができる (preseed)

5.3.3 devfs

Linux カーネル 2.4 で導入された devfs では、「動的に」「利用可能な」デバイスファイルが作成されるので、ハードディスクのパーティションやフレームバッファの存在を検出するのに便利です。

ただ、devfs はもう obsolete なので、今後は udev に書き換えていくことになるでしょう。

5.3.4 フレームバッファと国際化端末

Linux のデフォルトの端末画面はラテン文字しか表示できないので、国際化されたインストーラを実行するためにはフレームバッファドライバをロードしたあと、その上で国際化端末を実行する、という手順が必要になります。x86 の場合フレームバッファドライバは、vesafb → vga16fb と試行しています。d-i の国際化端末としては、次の 2 つを採用しています。

- bogl-bterm：UTF-8 対応の端末。
- jfbterm：再起動後、CJK 圏でのみ使われる端末 (表示は EUC-JP)。なぜ bogl-bterm で続けなかったかというと…

5.3.5 discover と hotplug

最初のハードウェア認識には discover が使われます。再起動後、標準で使われるのは hotplug です。

- discover：能動的にハードウェアを認識する。データベースには PCI ID 値などを列挙した discover1-data パッケージの情報を使う
- hotplug：ほぼ受動的にハードウェアを認識する。カーネルから送られた情報を使う

5.4 今後の d-i

Etch に向けての TODO はいろいろあります。

- より良いハードウェア認識：discover2 への移行。既存の discover のデータベースの更新 (volatile 経由？)。

- Woody のときに使えたような「レスキューモード」: rescue.udeb が用意されました
- Gtk+ による GUI フロントエンド：実装が進んでいます。ただ、udeb の持っている情報をもっと増やすか、別の GUI 専用のファイルを用意しないと、GUI のメリットを活かした画面にはならないと思います
- 複数言語の選択：だいぶ実装されてきてはいるようです
- フロッピーの取り扱い：今回は技術上、カーネル 2.4 のみの対応としています (カーネル 2.6 はフロッピーに入らない)。カーネルを分割してロードできないか？

6 個人提案課題

名前 _____

下記の空欄を埋めてください:

Debian installer の (　　) に注目し，今後のネタは (　　) します.

企画案の図：

7 Keysigning Party

上川純一

事前に必要なもの

- 自分の鍵の fingerprint を書いた紙 (`gpg --fingerprint XXXX` の出力．)
- 写真つきの公的機関の発行する身分証明書，fingerprint に書いてある名前が自分のものであると証明するもの

キーサインで確認する内容

- 相手が主張している名前の人物であることを信頼できる身分証明書で証明しているか[*1].
- 相手が fingerprint を自分のものだと主張しているか
- 相手の fingerprint に書いてあるメールアドレスにメールをおくって，その暗号鍵にて復号化することができるか

手順としては

- 相手の証明書を見て，相手だと確認
- fingerprint の書いてある紙をうけとり，これが自分の fingerprint だということを説明してもらう
- (後日) gpg 署名をしたあと，鍵のメールアドレスに対して暗号化して送付，相手が復号化してキーサーバにアップロードする (`gpg --sign-key XXXXX`，`gpg --export --armor XXXX`)

[*1] いままで見た事の無い種類の身分証明書を見せられてもその身分証明書の妥当性は判断しにくいため，学生証明書やなんとか技術者の証明書の利用範囲は制限される．運転免許証明書やパスポートが妥当と上川は判断している

8 次回

次回は 7 月 2 日土曜日の夜を予定しています．内容は本日決定予定です．
参加者募集はまた後程．