2008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
関西エリアDebian勉強会著

デビアン勉強会

目次

1　2008 年度 Debian 勉強会企画

上川 純一

2007 年 12 月に実施した 35 回 Debian 勉強会で、一年分の実施したい内容を出してみました。そこで策定した計画は以下です。

1. 新年会「気合を入れる」
2. Open Source Conference Tokyo (3/1)
3. データだけのパッケージを作成してみる、ライセンスの考え方
4. バイナリ一つのパッケージを作成してみるバージョン管理ツールを使い Debian パッケージを管理する (git) アップストリームの扱い (svn/git/cvs)
5. バイナリの分けたパッケージの作成。バイナリの分け方の考え方、アップグレードなどの運用とか。
6. パッケージ作成 (dpatch/debhelper で作成するパッケージ) man の書き方 (roff or docbook)
7. パッケージ作成 (kernel patch、kernel module) 、Debconf 発表練習
8. Debconf アルゼンチン、共有ライブラリパッケージ作成
9. Open Source Conference Tokyo/Fall、デーモン系のパッケージの作成、latex、 emacs-lisp、フォントパッケージ
10. パッケージの cross-compile の方法、amd64 上で i386 のパッケージとか、OSC-Fall 報告会、Debconf 報告会
11. 国際化 po-debconf / po 化 / DDTP
12. 忘年会

2 Debian パッケージ 管理の流れ

上川 純一

2008 年の Debian 勉強会では対応する種類別に Debian パッケージの作成の個別の内容を検討していくことになります。Debian パッケージの作成の技術的詳細にはいる前に、まず基本に立ち返って全体的な作業の流れを確認してみましょう。

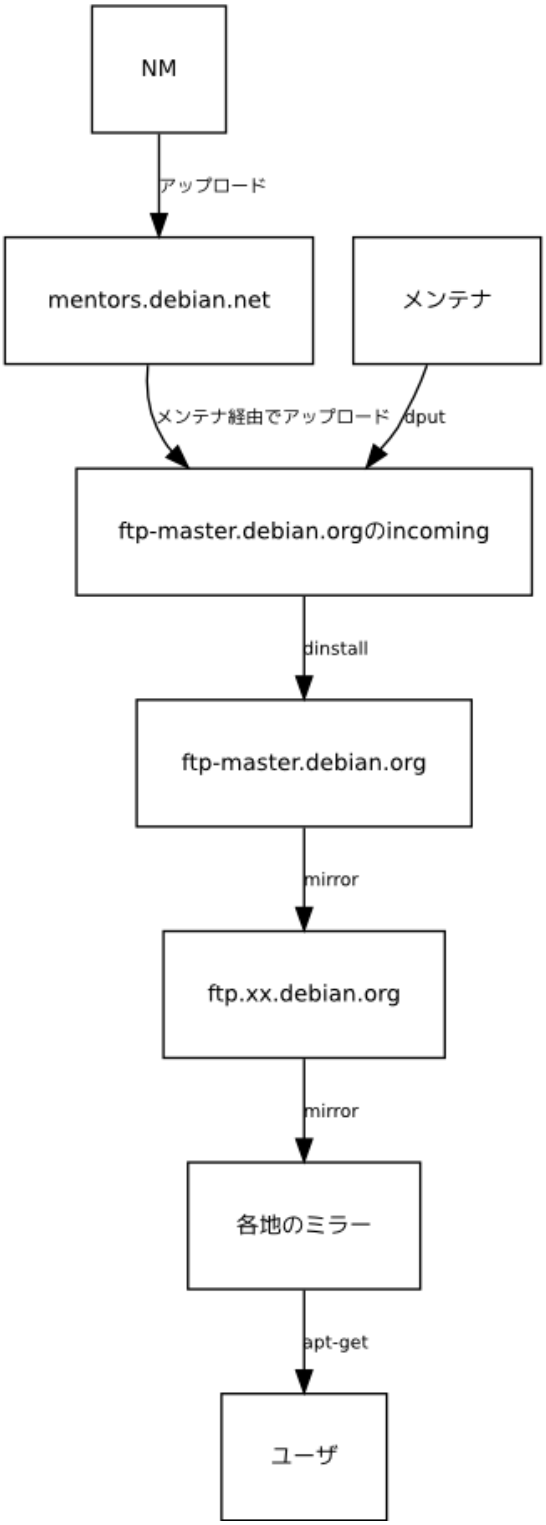
2.1 開発者からユーザにパッケージが到達するまで

Debian Developer はどこからかソースパッケージを取得してきて、パッケージをアップロードします。

公式パッケージの場合は、開発者が dput コマンドで ftp-master.debian.org にパッケージをアップロードすると da-katie が処理し、一日二回のバッチのタイミングで ftp.debian.org ミラーネットワークにパッケージが配信されます。日本のユーザであれば、 cdn.debian.or.jp を利用し、apt-get update / aptitude update をしたタイミングで新しいパッケージが取得できるようになります。

Debian Developer でない場合には、自分で dput を実行して直接 ftp-master.debian.org にアップロードするということはできません。Debian Developer の他の誰かに依頼して実施します。その際にこうしなければならないという定型の方法はありませんが、最近は Debian Mentors という仕組みが普及しています。`http://mentors.debian.net/` にパッケージをアップロードし、Debian Mentors の ML [*1]などで聞いてみるとよいでしょう。

[*1] `http://lists.debian.org/debian-mentors/`



2.2 開発者のパッケージングの契機

開発者がパッケージを作成するタイミングとはどういうものがあるでしょうか。簡単にいうと三つあります:

- Debian に存在しない全く新しいパッケージ (ITP からの一連の流れをふむ)
- Debian にすでに存在しているパッケージの新しいアップストリームバージョンがリリースされた
- Debian にすでに存在しているパッケージにユーザがバグ報告をし、BTS に登録されたバグ報告の内容に対応するにはパッケージの修正が必要になった

- 依存関係が更新されたから更新

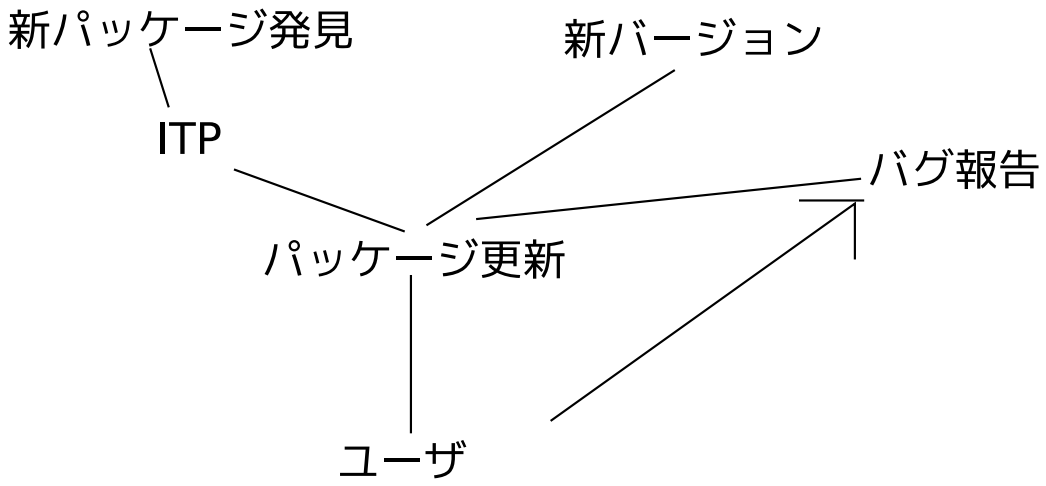


図 1　開発者のロジックの流れ

2.3　作業の流れ

2.3.1　アップストリームの取得

新しいソースパッケージを取得します。tar.gz で公開されていればそのまま使えます。tar.bz2 であったり、よくわからないアーカイブフォーマットだったり、git レポジトリでしか公開されていなかったりするので、それなりに加工します。

最初は適当に取得するわけですが、二回目移行はスクリプトで自動化して取得するようにします。watch という仕組みがあり、 uscan / uupdate コマンドで新しいバージョンをダウンロードしてきて Debian パッケージ用のパッチを適用することができます。

```
version=3
http://ftp.imendio.com/pub/imendio/giggle/src/giggle-(.*)\.tar\.gz
```

図 2　giggle パッケージの debian/watch ファイル

2.3.2　パッケージング

二度めからは差分の適用などで対応できるのですが、最初の新規パッケージの場合は ITP などの処理を必要とします。また、新規にパッケージを作成するので、なんらかのテンプレートパッケージから debian/ ディレクトリを取得してきて修正するか、 dh_make コマンドを利用してテンプレートを作成します。

この段階で debhelper を利用するのか、 cdbs を利用するのかの選択を迫られます。cdbs は debhelper より新しい仕組みですが、2003 年の登場から 5 年もたっているのでそろそろこなれています。簡単なパッケージであれば cdbs を利用すればよいでしょう。現状の cdbs のフレームワークで不具合があるような場合であれば debhelper で詳細に設定するのがよいでしょう。

```
$ dh_make

Type of package: single binary, multiple binary, library, kernel module or cdbs?
 [s/m/l/k/b] b

Maintainer name : Junichi Uekawa
Email-Address   : dancer@debian.org
Date            : Fri, 18 Jan 2008 22:22:51 +0900
Package Name    : tttt
Version         : 2.2
License         : blank
Type of Package : cdbs
Hit <enter> to confirm:
Could not find tttt_2.2.orig.tar.gz
Either specify an alternate file to use with -f,
or add --createorig to create one.
[22:22:55]coreduo:tttt-2.2>
```

2.3.3 BTS との対応

BTS に登録されているバグ報告に対してメールで対応したりします。報告されている不具合をちゃんと修正してみたりします。

BTS は閲覧は Web でできますが、制御は Web ではできません。BTS はメールで命令を直接うって制御できます。メールコマンドはすぐに忘れてしまうので、`/usr/share/doc/debian/bug-*` あたりを参照しながら作業します。BTS の操作に便利なツールがいくつかあるので、そちらを利用することをおすすめします。

- reportbug-ng
- reportbug
- dpkg-dev-el

2.3.4 ChangeLog の管理

パッケージの新しいバージョンを作成する場合には、debian/changelog に対応内容を記述します。ここで活用できるツールには次があります。

- devscripts の dch
- dpkg-dev-el の debian-changelog.el
- vim の debchangelog

バージョン番号を自動で増加させてくれたり、いろいろなエントリーの補完などをしてくれます。BTS を参照して修正されたバグのタイトルから ChangeLog のエントリを生成してくれたりもします。

```
pbuilder (0.177) unstable; urgency=low

  [ Loic Minier ]
  * Run apt-get autoremove after upgrade.

  [ Junichi Uekawa ]
  * python-apt/gdebi based pbuilder-satisfydepends-gdebi (closes:
    #453388)
  * Fix devpts mount permissions (closes: #453862)
  * Document pbuilder-satisfydepends-gebi in manpage

 -- Junichi Uekawa <dancer@debian.org>  Wed, 26 Dec 2007 20:53:24 +0900
```

2.3.5 ビルド・テスト

debian/以下のファイルを微調整して、パッケージをビルドします。dpkg-buildpackage を直接呼び出してもよいですが、devscripts パッケージの debuild コマンドを利用すればよいでしょう。

lintian / linda で生成されたパッケージにあきらかな間違いがないことを確認します。

debc, debdiff で生成されたパッケージにあきらかな間違いや意図しない大きな変化がないことを確認します。

debi でパッケージをインストールして正常に動作することを確認します。

pbuilder でビルドしてみて問題ないことを確認します。

pbuilder のテストスクリプト pbuilder-test を利用してインストール・アンインストール・アップグレードの試験をしてみるとさらによいでしょう。

2.3.6 アップロード

署名してアップロードします。dpkg-buildpackage コマンドを利用してパッケージをビルドすると自動で gpg で署名することになります。あとで署名しなおす場合には、 debsign を利用します。

アップロードは dput コマンドを利用します。

2.4 非公式パッケージレポジトリの作成方法

テスト用に非公式のパッケージレポジトリを準備すると便利です。

Packages.gz ファイルなどを生成して HTTP 経由でアクセスできるようにしておくと、ユーザが /etc/apt/sources.list に追記することで apt を利用してパッケージが取得できるようになります。

- dpkg-scanpackages / dpkg-scansources を直接利用する方法。
- apt-ftparchive を利用する方法。
- mini-dinstall を利用する方法。
- apt-move を利用する方法。
- debarchiver
- reprepro
- dak

2007 年 9 月号で apt-ftparchive が紹介されていたので再掲します。

2.4.1 apt-ftparchive

Sources.gz / Packages.gz などのパッケージ情報用ファイルを作成するためのツール。自分で作ったパッケージを apt-line として公開したいときに使います。

```
% apt-ftparchive packages . | gzip -9 > Packages.gz
% apt-ftparchive sources . | gzip -9 > Sources.gz
% apt-ftparchive release . > Release
```

2.4.2 apt-sortpkgs

Packages ファイル および Sources ファイルをソートします。apt-ftparchive で作成したものはソートされていなかったりするので、アルファベット順にソートするときに使います。

```
% apt-sortpkgs Packages > Packages.sort
```

2.5 参考文献

必読の文献として、以下があります。一通り読んだ後は、パッケージとしてインストールしておき、手元で検索できるようにしておくと便利です。sid を利用しているのであれば、パッケージとしてインストールしておくと最新版に勝手に更新されるため、便利です。内容に誤記などがあれば、ぜひ BTS にバグ報告を登録してください。

- debian-policy: Debian Policy Manual。パッケージの準拠すべきルールが文書化されています。
- developers-reference: Developers Reference。
- doc-debian: Debian Project Documentation。 BTS のメールインタフェースなどの文書がある。
- maint-guide: Debian New Maintainer's guide。New Maintainer として知るべきマナーのようなものが文書化されています。

3 Debian パッケージにおいてのライセンスの取扱い

上川 純一

Debian Project の作成するディストリビューション Debian GNU/Linux はフリーソフトウェアから構成されています。Debian Project においての「フリー」の定義は DFSG というガイドラインで定められています。それでは、それが実際にどういう使い方をされているのかを紹介します。

3.1 なぜ DFSG が必要なのか

フリーソフトウェアにもいくつかの考え方がありえます。どれが正しいというわけではないので、一つに統一することはできないでしょう。具体的には次の三種類くらいを考えればよいでしょう。

- 自由なものを自由でありつづけさせたいライセンス（GPL など）
- 自由なものを不自由にする自由も与えたいライセンス（MIT, BSD など）
- 自分の作ったものは自由につかってくれてよいが、自分の作成したものは完全なため、勝手にいじってほしくない。よって、変更したものについては明示的にしておいてほしいもの（LPPL 1.0 など）

複雑な例を挙げると、昔の LaTeX 関連のライセンス LPPL 1.0 では、変更したものについてはファイル名を変更することを義務としていました。また、現在でも gnuplot ではオリジナルのソースコードの配布を義務とし、オリジナルのソースコードとパッチという形での配布のみ認めています。

もともとの開発者と開発に携わろうとしている人たちにとっては大きな違いがありますが、エンドユーザにとってはフリーソフトウェアとして、利用できることに関してはあまり大きな違いはありません。

Debian はそのどちらの立場でもなく、ディストリビュータとして関わることになります。許諾された範囲で次のことができればよいことになります：

- Debian を有償・無償を問わず配布すること（DVD・公開ミラー・非公開ミラー）
- Debian を改変すること（Debian の改良、Debian 外への流用、Debian からの派生ディストリビューションの開発など）
- Debian をあらゆる用途で利用できること（商用・教育用・宗教用・軍事用・医療用などを問わない）

これらを現実的に阻害しないようなライセンスの条件を取りまとめたのが DFSG です。

3.2 DFSG 全文

DFSG 全文を図 3 に掲載します。

1. Free Redistribution
The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.
2. Source Code
The program must include source code, and must allow distribution in source code as well as compiled form.
3. Derived Works
The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. Integrity of The Author's Source Code
The license may restrict source-code from being distributed in modified form **only** if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.)
5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.
6. No Discrimination Against Fields of Endeavor
The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
7. Distribution of License
The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
8. License Must Not Be Specific to Debian
The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.
9. License Must Not Contaminate Other Software
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.
10. Example Licenses
The "GPL", "BSD", and "Artistic" licenses are examples of licenses that we consider "free".

図 3　The Debian Free Software Guidelines (DFSG)

3.3　ライセンス管理にかかわるプロセス

Debian GNU/Linux で、ライセンス管理に関連するプロセスや仕組みを整理しましょう。

3.3.1　ITP

Debian ディストリビューションにパッケージを追加するのなら、最初に「ITP」を行います。まずここでライセンスも付記することになり、特殊なライセンスであったら、議論になります。

3.3.2　debian-legal

複雑な条件や新規の状況が発生した場合には、debian-legal メーリングリストで議論します。メーリングリストでコンセンサスが醸成されたり、されなかったりします。フレームウォーが展開されたりします。結論はなかなか出ませんが、いろいろな視点が展開されるので見ているとおもしろいです。

3.3.3　NEW キュー

ITP 後パッケージをアップロードすると、メーリングリストの議論とはあまり関係ない方向で、ftp-master が debian/copyright を確認し、パッケージを承認し、新しいパッケージがディストリビューションに入ります。この時点で Debian としてこのパッケージをこのライセンスで配布することは DFSG フリーであるという判断をしたことになります。

この判断をくつがえすには、パッケージに対して、ライセンスが間違っていると serious バグを報告することにな

ります。

3.3.4 再配布

ftp-master に一旦入ると、Debian をミラーしている各サーバで再配布されることになります。また、CDROM イメージなどでも再配布されることになります。

各国の著作権法では著作物の複製は基本的には許諾がなければ許可されないというルールです。そのため、Debian としてはライセンスの条件に同意して複製しているという考え方をとっています。各ミラーが到底許諾できないような条項は受け入れられません。

3.3.5 改変

ユーザはダウンロードした Debian の一部を改変します。ユーザは改変をする際には、各パッケージの許諾にしたがっています。

3.4 パッケージ作成においての debian/copyright 作成の実際

Debian パッケージを作成する際に、ライセンス関連を確認します。著作権は原則として許諾がないと「利用」ができないという類の権利です。Debian パッケージの場合、その情報を debian/copyright ファイルに整理します。

では、ライセンスはどうやって確認するのでしょうか。原作者はフリーソフトウェアとして一旦公開することを決心したのですから、良心的にライセンス情報をある程度整理してくれているはずです。しかしながらその内容は監査の手続きが確立されているわけでもなく、標準が存在するわけでもないので一律ではありません。また、よくあることですが、本人以外でその点を注意して確認しようとしたのは今 Debian パッケージ作業をしようとしているあなたが初めてかもしれません。必要事項が記述されていないこともありうるため、その場合は原作者に確認をとりましょう。

ライセンス関連の情報は、通常はマニュアルやウェブページに記述されています。そこで大まかな状況を確認します。例えば、failmalloc パッケージを例にとってみてみましょう。failmalloc のソースディレクトリを見てみます。

```
$ ls
AUTHORS  ChangeLog  Makefile.am  NEWS         THANKS      configure           debian   failmalloc.c  ltmain.sh  mkinstalldirs
COPYING  INSTALL    Makefile.in  README  aclocal.m4  configure.ac  depcomp  install-sh          missing
```

まず COPYING ファイルがあるのがわかります。これは FSF の出しているライセンスに多く、GPL 全文が入っている場合が多いです。この場合は内容を確認してみると、LGPL v2.1 の全文が入っていました。

```
                  GNU LESSER GENERAL PUBLIC LICENSE
                       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[略]
```

他にも、マニュアルなどにライセンスについて記述されている事が多いです。確認してみましょう。今回は README, AUTHORS, ChangeLog, THANKS を確認しても特にライセンス関連の記述はありませんでした。

あと、最終的にソースコードそれぞれのライセンスを確認しましょう。別のプロジェクトからコピーしてきたファイルなどが含まれることもあり、別のライセンスのものが混じっている可能性もあります。failmalloc.c を確認すると、ライセンスについて明言している文面があるのが分かり、COPYING ファイルと整合性がとれているのが確認できます。

```
/* failmalloc - force to fail in allocating memory sometimes */
/*
 * Copyright (C) 2006 Yoshinori K. Okuji <okuji@enbug.org>
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.

 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.

 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
*/
```

今回は特にありませんが、バイナリデータファイルなども確認しておくとよいでしょう。mp3 ファイルや、画像データなど、本当に作者に権利があるのか念のため確認しておくとよいでしょう。例えば調べた結果音楽データが Creative Commons ライセンスで許諾されているのであれば、それは別のライセンスのファイルであるとして debian/copyright に記述します。

また、インストールされるバイナリがソースコードから生成されていることを確認します。[*2] ソースコードから全部のバイナリが作成できないのであれば、原作者に問い合わせるステップが必要になることもあります。

そうやって調査した一連の内容を debian/copyright に整理して記述します。複数の許諾者や許諾内容がある場合はそれがわかるように整理しましょう。この場合は、LGPL である旨を記録し、あと Debian パッケージ関連のスクリプトについても他の部分と同じライセンスである LGPL2.1 に従うということを明記しています。

```
 This package was debianized by Hideki Yamane (Debian-JP) <henrich@debian.or.jp> on
Sat, 26 Jan 2008 09:28:51 +0900.

It was downloaded from http://www.nongnu.org/failmalloc/

Upstream Author:

    Yoshinori K. Okuji <okuji@enbug.org>

Copyright:

    Copyright 2006 Yoshinori K. Okuji

License:

    This package is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License.

    This package is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this package; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA

On Debian systems, the complete text of the GNU Lesser General
Public License can be found in `/usr/share/common-licenses/LGPL-2.1'.

The Debian packaging is (C) 2008, Hideki Yamane (Debian-JP) <henrich@debian.or.jp> and
is licensed under the LGPL-2.1, see `/usr/share/common-licenses/LGPL-2.1'.
```

なお、`/usr/share/common-licenses` に一般的に使われるライセンス文[*3]が用意されており、一般的な文面の場合はそれを引用することでよいことになっています。そうでない場合はライセンス文を全文 debian/copyright ファイルの中に転記します。

メンテナがこうやって準備した debian/copyright ファイルは、Debian ユーザは `/usr/share/doc/`**パッケージ名**`/copyright` で確認できます。

[*2] ROM から吸い出したバイナリファームウェアが入っている「フリーソフト」もあるので注意しましょう。
[*3] 現在は Artistic, BSD, GFDL, GPL, LGPL

4 Debian パッケージ作成：データだけのパッケージ

David Smith

Debian パッケージ作成にようこそ！dpkg のパッケージ、いわゆる Debian パッケージの作成を分かりやすく、実例の分析に基づいて説明します。[*4]今回データーだけのパッケージについてみてみましょう。

データーだけのパッケージは何かと言えばフォントやドキュメンテーションや設定ファイルなどのパッケージを指します。要するにプログラムとライブラリではないパッケージではないということです。今回、GNOME デスクトップの壁紙ファイルを集めたもの、gnome-backgrounds の構成を元に説明します。Debian Popularity Contest[*5]によると現在 gnome-backgrounds が殆ど全部の GNOME インストールに含まれていて、GNOME を使用していれば gnome-backgrounds も入っているはずです。背景ファイルですが、パッケージの構成に複雑な部分もあると思いますので、細かいところまで見てみようと思います。

4.1 準備

まず、Debian New Maintainer's Guide [*6] のページを開いておいたらいいでしょう。不明な点があった場合、New Maintainer's Guide に分かりやすく書いていますし、質問があった場合は何処まで送れば良いか書いてあります。

次に gnome-backgrounds のソースパッケージをダウンロードしましょう。

```
$ apt-get source gnome-backgrounds
```

準備の最後にパッケージの構成に必要な依存パッケージをインストールしなければビルドが出来ないので以下のコマンドをインストールしましょう。

```
$ apt-get build-dep gnome-backgrounds
```

4.2 基本の Debian パッケージファイル

今のところ、apt-get source がソースパッケージを gnome-backgrounds-2.20.0 というディレクトリに展開されています。そのディレクトリの中には debian/ディレクトリがあり、そこにパッケージ作成の設定ファイルがあります。パッケージによるファイル数が異なるが、gnome-backgrounds は必要最低限のもので構成されています。

[*4] Debian パッケージが作りたいですよね。一応作りたいと前提に思っているからよろしくお願いします。
[*5] Debian Popularity Contest の URL:`http://popcon.debian.org/`
[*6] Debian New Maintainer's Guide の URL: `http://www.debian.org/doc/devel-manuals#maint-guide`

```
piyo:/tmp/gnome-backgrounds-2.20.0> ls -l debian/
合計 36
-rw-r--r-- 1 dds dds 3755 2008-03-08 16:21 changelog
-rw-r--r-- 1 dds dds    2 2008-03-08 16:21 compat
-rw-r--r-- 1 dds dds 1022 2008-03-10 15:11 control
-rw-r--r-- 1 dds dds  771 2008-03-08 16:21 control.in
-rw-r--r-- 1 dds dds 1780 2008-03-08 16:21 copyright
-rwxr-xr-x 1 dds dds  368 2008-03-08 16:21 rules
-rw-r--r-- 1 dds dds  145 2008-03-08 16:21 watch
```

この中で changelog、control、copyright、rules はあらゆるパッケージに必要なファイルです。

4.2.1 debian/copyright

`debian/copyright` は上流ソースコード、データーなどの著作権を認めるためのファイルです。形式は自由ですが、パッケージビルドツールに対して必須なので自分専用でも自分の会社内専用のパッケージでも一応書く必要があります。最低限として「All Rights Reserved」と書いてもよいです。gnome-backgrounds の `debian/copyright` ファイルが長すぎるのでここには示しませんが、気になる方は見てみてください。このファイルはパッケージをインストールすると`/usr/share/doc/`**パッケージ名**`/copyright` にインストールされます。

4.2.2 debian/changelog

`debian/changelog` にはパッケージのバージョンを記録します。必ずしもパッケージのバージョンが上流のバージョンに一致することではないので必須になっています。それにパッケージの最新バージョン番号はこのファイルの最初の 1 行目から直接に抽出されます。

gnome-backgrounds の最新バージョンの changelog を見てみましょう。

```
gnome-backgrounds (2.20.0-1) unstable; urgency=low

  * New upstream release.

 -- Sebastian Dr\"oge <slomo@debian.org>  Sat, 22 Sep 2007 09:49:38 +0200
```

かなりシンプルです。しかし、空白と日付の形式を厳守しなければビルドするときにエラーが発生するから注意しましょう。[*7]

4.2.3 debian/control

`debian/control` にはパッケージ名、Maintainer、依存のパッケージなどのメタ情報が記されています。gnome-backgrounds の場合、独特として `debian/control` と共に `debian/control.in` ファイルもあります。`debian/control.in` は Debian のパッケージツールに対して関係ないから無視しても良いです。Gnome プロジェクトは規模が大きいのでメンテに関わっている人が多いです。`debian/control` にアップロード出来る人達の名前が記載されていますが、人の名前が多すぎて直接管理しにくいので、@GNOME_TEAM@という変数だけを書いています。そして別のファイルにその人達の名前を書いて、発行時にスクリプトによって変数を置換します。

では `debian/control` を見てみましょう。

```
Source: gnome-backgrounds
Section: gnome
Priority: optional
Maintainer: Sebastien Bacher <seb128@debian.org>
Uploaders: @GNOME_TEAM@
Build-Depends: debhelper (>= 5), cdbs, gnome-pkg-tools
Build-Depends-Indep: libxml-parser-perl
Standards-Version: 3.7.2

Package: gnome-backgrounds
Architecture: all
Depends: ${misc:Depends}
Description: a set of backgrounds packaged with the GNOME desktop
 This is a collection of desktop backgrounds created with GNOME users in mind.
 It contains the following backgrounds:
 .
[後略]
```

[*7] debchange(1) というツールで手軽に changelog 編集が出来ます。devscripts というパッケージで提供されています。

第一段落にソースパッケージを記述します。apt-get source と apt-get buildd-dep を実行する時、ソースパッケージの情報を使っています。9 行目以降に作成する gnome-background のバイナリパッケージを定義しています。順番に関してはソースパッケージが必ず最初に記述されます。そして Source 又は Package はそれぞれの段落の最初の 1 行に書きます。それ以外の中身の項目は特に決まった順番がありません。

Section と Priority との項目は少しややこしいです。決まった選択肢の中から 1 つしか選べないので合わない場合があるけど gnome-backgrounds の場合はぴったり、gnome で optional。選択肢の詳しい情報は Debian Policy Manual に書いてあります。[*8] 後は Maintainer と Uploaders はそれぞれ書いた通りです。Uploaders は共同にメンテをしている人になります。つまり、Maintainer の項目は必ず一人しか書けないけど他の人もメンテしていれば Uploaders に書くということになります。

次にビルドのための依存関係パッケージが書いてあります。殆どのパッケージにはビルドするための依存パッケージと、利用するための依存パッケージが異なります。例えば普通のプログラムの場合に C/C++ のヘッダーがビルド時に必要ですが、実行する時には必要ないパッケージがあり、これらを分けて書いています。[*9]gnome-backgrounds がビルド時に使ってる依存パッケージがより典型的なのでそれぞれ少し説明します。

cdbs　**Common Debian Build System** の省略, 共有されているビルドシステムです。多くの人が使っていると思います。

debhelper　いろいろパッケージビルドに役に立つツールです。例えば上流のドキュメンテーションを自動的にインストールするツールなどが提供されています。

gnome-pkg-tools　cdbs に少し似ている感じで GNOME パッケージを作るのに役に立つツール群です。

libxml-parser-perl　Perl の XML::Parser モジュールを提供するパッケージです。

ソースパッケージの段落の最後に Standards-Version という項目は Debian Policy Manual の何バージョンに準拠しているかを説明します。

続いてバイナリパッケージの項目になります。パッケージ名が最初に来ます。それから対応するアーキテクチャーを書きます。データーだけのパッケージの場合にも all にしても良いです。今度のバイナリパッケージ作成発表に all の他を説明します。その他は Depends ラインに runtime の依存関係のパッケージ（この場合は自動的に抽出される）が書いていて、最後にパッケージの説明を記述します。詳しくは Debian Policy Manual を参照してください。

4.2.4　debian/rules

`debian/rules` でパッケージのビルドがどう行われるかを定義します。中身の形式は完全自由ですが、外部のツールが Policy Manual に書いている呼び出し方に依存しています。それで Makefile 形式を用いているパッケージの割合は 100% に近いです。しかし Makefile なのに cdbs を使っているので普通の Makefile にほとんど似ていません。時間と労力を節約したい方は cdbs を是非試してください。

gnome-backgrounds の `debian/rules` を見てみましょう。

```
#!/usr/bin/make -f

include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/gnome.mk
include /usr/share/cdbs/1/rules/simple-patchsys.mk
include /usr/share/gnome-pkg-tools/1/rules/uploaders.mk
-include /usr/share/gnome-pkg-tools/1/rules/gnome-get-source.mk

binary-post-install/gnome-backgrounds::
        rm -rf debian/gnome-backgrounds/usr/share/locale
```

殆ど以前に定義しているルールを再利用しているだけです。1 つだけのこのパッケージに特別に追加している箇所があります。それは binary-post-install/gnome-backgrounds::のところです。gnome-backgrounds というバイナリパッケージを作ってから、必要ないのでそのディレクトリを削除する、という意味です。

[*8] Debian Policy Manual URL:`http://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections`

[*9] Build-Depends と Build-Depends-Indep の違いがあまり分からなくてすみません！

4.2.5 debian/watch

`debian/watch` ファイルは不要ですが、自動的に新しい上流のソフトを取得してビルドする仕組みを利用する場合に使用します。

4.3 パッケージビルド

上記を設定してから、パッケージのビルド自体は dpkg-buildpackage 又は debuild コマンドだけで出来きます。設定以来の変更を `debian/changelog` ファイルに記録するとバージョン管理ができます。dpkg-buildpackage の出力が長いので、ここでは説明しませんが、各自で確認してみてください。

5 バイナリー つだけのパッケージを作成してみる

吉田　俊輔

5.1 前提/対象者

この記事は Debian パッケージを作った経験が無い人むけに一番わかりやすいと思われる dh_make を使ったパッケージの作成について説明しています。

前提知識としてはある程度 make(Makefile) を理解している必要がありますが、多くはありません。若干はこのドキュメント中で説明します。

5.2 deb パッケージを作る/使う理由・メリット

通常のパッケージシステムのメリットとして tar ボール (*.tar.gz などで圧縮されたアーカイブファイル) を使った場合に比較して、複数マシンへのインストール、環境再現が簡単、便利、パッケージのアンインストール、バージョンアップが容易といった点が挙げられます。

deb パッケージを作るメリットとしては、それらに加え、ソースファイルとパッチファイルの管理が簡単、ビルドに必要な環境（のパッケージ）を整備するのが簡単、等のメリットが挙げられるでしょう。

もちろんパッケージを作りたい理由は人それぞれと思いますが、自分の使いたいアプリ/機能がパッケージになっていないという理由が多いでしょう。

Debian 開発者になるためと言う人もいるかもしれません。

5.3 dh_make とは

dh_make コマンドは dh-make パッケージに含まれるコマンドで、パッケージを作る際に一般的な内容のひな形を作ってくれるツールです。

ソフトウェアのソースコードを展開したディレクトリの中で、このコマンドを実行すると、対話的なやりとりを行ったあと、ひな形として debhelper スタイルのディレクトリとファイルを準備します[*10]。

5.4 debhelper スタイル

deb パッケージを作る方法 (スタイル) についてはいくつかあります。今回説明するのは dh_make の debhelper スタイルについてです。そのほかに同様に debhelper スタイルを使用した dpatch,CDBS,dbs,quilt,yada といったスタイルがあるそうです。

その他にもスクリプト言語向け等にもいくつかスタイルがあるようです。debhelper スタイルは debian ではもっと

[*10] CDBS スタイルのひな形も作成可能です。

も一般的でシンプルなスタイルです。現在の debian パッケージで採用しているパッケージの割合がもっとも多いと言われています。基本的に dh_コマンドを rules というファイルからから呼び出す形式となります。

5.5 dh_make での debhelper スタイルのパッケージ作成に最低限必要なファイル

dh_make が作成する、debhelper スタイルのパッケージ作成に最低限必要なファイルとして以下の 2 つのディレクトリと、4つのファイルがあります。

まず、作業に使うディレクトリ名の形式が決まっています。その下に debian ディレクトリが作成され、パッケージングに必要なファイルはここにすべて置かれます。必要なファイルについてはひな形が dh_make で準備されます。

```
softname-version/（ディレクトリ名）
  debian/（Debian ディレクトリの存在=deb パッケージが作成可能）
     control（パッケージ情報：パッケージ名、バージョンの記述）
     changelog（更新履歴）
     copyright（著作権情報の記述）
     rules（rules パッケージの作成方法を記述）
```

5.6 その他のファイル

以下は dh_make が準備するサンプルのファイルです。これらも同様に dh_make で debian ディレクトリの下に準備されます。必須ではありませんので、必要に応じて利用します。

```
debian/
  README.Debian(オリジナルとパッケージ化したときの差分情報）
  conffiles.ex(設定ファイル名のリスト）
  cron.d.ex(cron で実行するファイル名リスト）
  dirs
  docs
  init.d.ex(サービスの起動スクリプト）
  manpage.1.ex, manpage.sgml.ex(man ファイル）
  preinst.ex(インストール前のスクリプト）
  postinst.ex(インストール後のスクリプト）
  prerm.ex(アンインストール前のスクリプト）
  postrm.ex(アンインストール後のスクリプト）
  等
```

5.7 新規作成の流れ

db_make を使ったパッケージの新規作成の流れとしては以下のようになります。

- ソースアーカイブ展開
- テンプレート作成

```
$ dh_make
```

- debian ディレクトリ配下の編集
- changelog 記述

```
$ dch
```

- control ファイル (パッケージ名や依存関係を記述)
- copyright ファイル (著作権情報の記述)
- rules ファイル (ビルド手順の修正、確認、不要なコードの削除)
- ビルド

```
$ dpkg-buildpackage(deb パッケージの作成）
```

5.8 ソースアーカイブ展開

通常、tar ボール等で配布されているファイルを展開し、その展開されたディレクトリ名を修正（リネーム）します。

dh_make でひな形を作るためには packagename-version の形式である必要がありますので、上記の形式になっていない場合は適切にリネームを行ってください。

5.9 dh_make(debian ディレクトリ、テンプレートの作成)

dh_make でのひな形の作成です。

上記で適切な形式にリネームされたディレクトリに cd し、dh_make を実行します。その際、可能ならライセンスの指定を行った方が良いでしょう。後々の手間が省けます。

また、環境変数からメンテナ名とそのメールアドレスを取得しますので、あらかじめ下記の環境変数を設定しておくことをおすすめします。これも後々の手間が省けます。

dh_make を実行すると、パッケージの種類を選択します。今回の例では Single binary (s) を選択します。

その他、許諾条件（ライセンス）の指定 (-c) も可能です。指定できる許諾条件（ライセンス）は gpl、lgpl、artistic、bsd です。

指定できるパッケージ種類は Single binary (s)/Multiple binary (m)/Library (l)/Kernel module (k)/cdbs (b) です。

環境変数は、DEBFULLNAME(メンテナ名)、DEBEMAIL(メールアドレス) で指定します。

5.10 dch(debian/changelog の記述)

次に、debian/changelog の記述の記述について説明します。まず、このファイルはテキストファイルですが、形式が厳密に決まっており、直接エディタで修正するのはおすすめしません。

たとえば、

```
$ dch -i
```

を使えば定型の形式部分は自動的に記入できるのでおすすめです。

次に、debian/changelog の先頭にあるバージョン情報がこれから作成されるパッケージのバージョンとして使用されます。必要なら修正を行ってください。

次がパッケージの状態ですが、通常は unstable で問題ありません。

次は緊急度ですがこれも low で問題ないでしょう。

実際の変更履歴としての内容はアスタリスク (*) の後にコメントとして記述します。その後の定型 dch が作成してくれる内容をそのまま使うのが良いでしょう。

debian/changelog:

```
smp-mgzip (1.2c-1) unstable; urgency=low
 * Initial release (Closes: #nnnn)  <nnnn is the bug number of your ITP>
-- yoshida syunsuke <koedoyoshida@gmail.com>  Sun, 30 Mar 2008 22:21:28 +0900
```

5.11 control の記述

control ファイルにパッケージ名や依存関係を記述します。

ここではソースパッケージとバイナリパッケージで分けて記載します。

Section には main (完全にフリーなソフトウェア)、 non-free (実際の所フリーであるとはいえないソフトウェア)、そして contrib (それ自身はフリーなソフトウェアであるけれども、non-free なソフトウェアが無ければ使えないも

の) があります。 更に、これらの下には各パッケージをおおまかに分類する論理的なサブセクションが用意されており、 そこに含まれるパッケージの種類を簡単に説明するような 名前がつけられています。 管理者専用のプログラムのために「admin」、 基本的なツールのために「base」、 プログラマーのためのツールが含まれる「devel」、 文書の「doc」、ライブラリの「libs」、 電子メールの読み書きに使うリーダや 電子メールサーバを構築するためのデーモンは「mail」、ネットワーク関係のアプリケーションやデーモンの「net」、 他のどんな分類にもあてはまらないような X11 用の プログラムは「x11」など、そしてさらに多くのものが 用意されています。デフォルトは main ですので、ここに記載すると main のサブセクションを指定するということになります。

Priority はこのパッケージをインストールすることが ユーザにとってどれくらい重要なものかを示しています。

新規パッケージの場合、優先度「optional」(選択可能) としておけば、通常は問題無いでしょう。

バイナリパッケージについては CPU 等のアーキティクチャに依存する物 (バイナリの実行ファイル等を含む場合) は any, スクリプトやドキュメントなどアーキティクチャに依存しない場合は all を指定します。その他の項目は必要なら修正してください。

debian/control:

```
Source: smp-mgzip(ソースパッケージ名)
Section: unknown(パッケージのセクション)
Priority: extra(パッケージの重要度)
Maintainer: yoshida syunsuke <koedoyoshida@gmail.com>(メンテナーの名前とメールアドレス)
Build-Depends: debhelper (>= 5), autotools-dev(ビルドに必要なパッケージ)
Standards-Version: 3.7.2(テンプレート作成に参照した Debian ポリシー標準のバージョン)

Package: smp-mgzip(バイナリパッケージ名)
Architecture: any(対応アーキティクチャ)
Depends: ${shlibs:Depends}, ${misc:Depends}(パッケージの依存関係)
Description: <insert up to 60 chars description>(パッケージの説明)
 <insert long description, indented with spaces>
```

5.12 copyright の記述

ファイルの著作権と許諾条件 (ライセンス) についての記述です。

dh_make 実行時にライセンスを指定しておけばひな形が作成されます。ソフトウェアによってはディレクトリやファイルごとに別のライセンスの場合もありますので、注意してください。

debian/copyright:

```
This package was debianized by yoshida syunsuke <koedoyoshida@gmail.com> on
Sun, 30 Mar 2008 22:21:28 +0900.

It was downloaded from <fill in http/ftp site>

Upstream Author: <put author(s) name and email here>

Copyright: <put the year(s) of the copyright, and the names of the
           copyright holder(s) here>

License:
 (中略)
On Debian systems, the complete text of the GNU General
Public License can be found in '/usr/share/common-licenses/GPL'.

The Debian packaging is (C) 2008, yoshida syunsuke <koedoyoshida@gmail.com> and
is licensed under the GPL, see above.
```

5.13 rules の記述

debian/rules にはパッケージングを行うための手順を記述します。Makefile 形式の記述方法ですが、パッケージングの支援をする dh_xxx コマンド群 (debhelper パッケージ) を並べるのがほとんどです。dpkg-buildpackage が期待している必須ターゲットは：

```
build: (ビルド実行)
binary: (バイナリパッケージの生成、通常 binary-indep と binary-arch の実行)
binary-indep: (アーキティクチャ独立のバイナリパッケージ作成)
binary-arch:(アーキティクチャ依存のバイナリパッケージ作成)
```

5.14 rules の記述 build ターゲット

dh_make のデフォルトの場合、build ターゲットは build-stamp に依存している＝ build-stamp ターゲット (make、コンパイル等の実作業) を呼び出します。

同様に build-stamp は config.status(configure の実行) に依存しています。

```
debian/rules
build: build-stamp

build-stamp:  config.status
        dh_testdir

        # Add here commands to compile the package.
        $(MAKE)
        #docbook-to-man debian/smp-mgzip.sgml > smp-mgzip.1

        touch $@
```

5.15 rules の記述 binary ターゲット

```
binary: binary-indep binary-arch

binary-indep: build install

binary-arch: build install
        dh_testdir
        dh_testroot
        dh_installchangelogs ChangeLog
       (中略)
        dh_builddeb
```

5.16 rules の記述 install ターゲット

```
install: build
        dh_testdir
        dh_testroot
        dh_clean -k  --exclude ./mgzip.c.orig
        dh_installdirs

        # Add here commands to install the package into debian/smp-mgzip.
        $(MAKE) prefix=$(CURDIR)/debian/smp-mgzip/usr install
```

5.17 オリジナルの make ファイルの修正

Single binary (s) の場合、Debian のパッケージングツールは$(CURDIR)/debian/(**パッケージ名**)/以下にあるファイルをまとめてパッケージにします。

「make install」実行で$(CURDIR)/debian/(**パッケージ名**)/以下にインストールするようにオリジナルの tarball 等から展開された、Makefile を準備します。

元々の Makefile が提供されていればそれを修正すればよいでしょう。無ければ Makefile を作成してください。詳細な Makefile の作り方については書籍「make 改訂版」を参照してください。

5.18 パッケージ作成

簡易なパッケージ作成のテストとして下記のコマンドを実行するとパッケージ作成が行われます。

```
$ dpkg-buildpackage  -us, -uc  -rfakeroot
```

上記のファイルが適切に作成されていれば、以下のファイルが作成されます。

```
xxx-y.y.y_zzzz.deb(バイナリパッケージ)
ソースパッケージ
xxx-y.y.y.dsc(ソース概要:control より生成)
xxx-y.y.y_zzzz.diff.gz(パッケージ化用の差分)
xxx-y.y.y.orig.tar.gz(オリジナルソースファイル)
xxx-y.y.y_zzzz.changes(パッケージ変更点: control+changelog)

凡例：
        xxx:パッケージ名
        y-y-y:バージョン,
        zzzz: Debian アーキテクチャ(i386,amd64,all, 等))
```

上記の例で dpkg-buildpackage に渡しているオプションの意味は以下の物です。

- -us, -uc(gpg サインをしない)
- -rfakeroot(一般ユーザでの作成)

5.19 dpkg-buildpackage と debuild の違い

dpkg-buildpackage とよく似たプログラムに debuild があります。違いとしては以下の点があります。

- dpkg-buildpackage:
  メリット　dh_make で作ったばかりの設定ファイルでも deb を作成可能
  デメリット 一般ユーザで使用するときは-rfakeroot の指定が必要
- debuild:
  メリット　-rfakeroot 指定が省略可能、ルールに従ったチェックの実行を行ってくれる。
  デメリット 各種ルールに従って設定ファイルを書かないとエラーや警告となる作成されるパッケージに関しては両方とも同じ

5.20 ファイルへの署名

これまでの例で、dpkg-buildpackage や debuild で指定した -us -us のオプションはファイルへの電子署名を省略するためのオプションです。

- -us, -uc Do not sign the source package or the .changes file, respectively.

パッケージを自分のみで使用する場合はともかく、外部へ公開することも視野にいれるのであれば、電子署名を行うのが適切でしょう。

debian パッケージに署名するには GnuPG で鍵を用意し、パッケージ作成時に debuild、dpkg-buildpackage を使用するとデフォルトでパッケージに署名が行われます。

5.21 補足：ファイルの公開方法

パッケージ公開用のファイル (Packages.gz,Sources.gz) を作ります。

```
$ apt-ftparchive packages . | gzip -c9 > Packages.gz
$ apt-ftparchive sources . | gzip -c9 > Sources.gz
```

apt-ftparchive コマンドは apt-utils パッケージに入っています。

5.21.1 ローカルで確認

sources.list(aptline) に以下のような記述を追加し apt-get update;apt-get install でインストールできるかを確認します

```
deb file:/usr/src/ ./
```

5.21.2 ネットワークで確認

ローカルで確認できたディレクトリ構成のまま、そっくり http で公開します。sources.list(aptline) も同様に記述します。

```
deb-src http://XXX ./
```

5.22 補足：deb ソースファイルの展開とリビルド

.dsc,.orig.tar.gz,*.diff.gz を入手して展開し、

```
$ dpkg-source -x XXX.dsc
$ cd 作成されたディレクトリ
```

必要なら修正を実施してリビルドします

```
$ debuild（または dpkg-buildpackage）
```

5.23 参考資料

- 入門 Debian パッケージ (ISBN:4-7741-2768-X)
- Debian 新メンテナガイド[*11]
- Debian GNU/Linux スレッドテンプレ　自作パッケージを作りたい[*12]
- make 改訂版 (ISBN:4-900900-60-5)

[*11] http://www.jp.debian.org/doc/manuals/maint-guide/
[*12] http://debian.fam.cx/index.php?AptGet#n8109a54\

6　バイナリをわけたパッケージの扱い

前田 耕平

6.1　はじめに

今回は Debian パッケージを作成した経験がほぼ無い人を対象とし、先月[*13] の『バイナリ一つだけのパッケージを作成してみる』を予習していることを前提とします。

一応、前回の内容もおさらいしながら、今回も dh_make を使って、debhelper スタイルでの複数のバイナリに分けてパッケージを作る方法を説明しますので、前回参加できなかったとしても、ある程度は問題ないかと思います。

6.2　バイナリの分け方の考え方

通常のパッケージは、一つのバイナリパッケージとして作成・配布されるケースが多いでしょう。ではどのようなケースにおいて、バイナリパッケージは複数に分けられているのでしょうか。大きく分けると次の三パターンくらいになりそうです。

- 開発環境と実行環境
  例）
  - Java の SDK と JRE
- C/S モデルのサーバ側プログラムとクライアント側プログラム
  例）
  - bind と dnsutils
- プログラム本体とアドオンで追加可能なモジュール、ユーティリティ
  例）
  - Apache と Apache module や Apache Bench

これらに共通して言えるのは、単一のソースツリーから複数に分けることにより、導入するユーザの使い勝手が良くなる、ということだと言えると思います。

なお、複数のバイナリパッケージを作成するときも、ソースツリー及びソースパッケージ自体は単一なので、複数のソースツリー・ソースパッケージから生成される事はありません。

6.3　前回のおさらい

前回は、dh_make を使って、debhelper スタイルで単一のバイナリパッケージを作りました。今回は、まず簡単な C のプログラムと Makefile を用意して、それを Debian パッケージにします。そこで単一のバイナリパッケージの場

[*13] 2008 年 4 月

合と複数のバイナリパッケージに分ける場合との違いを見てみましょう。

6.3.1 ソースコードと Makefile の用意

まず、下記のようなディレクトリ、ファイルを用意します。

```
hoge-1.0/
 hoge.c
 Makefile
```

ファイルはそれぞれ下記のようなコードを書きます。

- hoge.c

```
#include <stdio.h>

int main(void)
{
       printf(‘‘hello\n’’);
       return 0;
}
```

- Makefile

```
CC=gcc
CFLAGS=-O
BINDIR=/usr/games

hoge: hoge.c
       $(CC) $(CFLAGS) -o hoge hoge.c

install:
       install -d ${DESTDIR}${BINDIR}
       install -m 755 hoge ${DESTDIR}${BINDIR}

clean:
       rm -f hoge
```

6.3.2 dh_make でテンプレートの作成

それでは、前回と同様に dh_make を使ってシングルパッケージを作ってみます。まず、dh_make でテンプレートを作ります。

```
$ cd hoge-1.0

$ dh_make --single -c gpl --createorig
Maintainer name : Kouhei Maeda
Email-Address   : mkouhei@gmail.com
Date            : Sat, 10 May 2008 15:36:29 +0900
Package Name    : hoge
Version         : 1.0
License         : gpl
Type of Package : Single
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now. You should also
check that the hoge Makefiles install into $DESTDIR and not in / .
```

6.3.3 debian ディレクトリ以下のファイルの編集

debian ディレクトリ以下を編集します。

```
$ cd debian

$ rm *ex *EX (<-不要なサンプルファイルを削除します。)
```

- debian/control

```
Source: hoge
Section: games
Priority: extra
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: hoge
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: sample program
 Hello hoge program
```

- debian/copyright

```
This package was debianized by Kouhei Maeda <mkouhei@gmail.com> on
Sat, 10 May 2008 15:36:29 +0900.

It was downloaded from <http://www.palmtb.net>

Upstream Author(s):

    <Kouhei Maeda mkouhei@gmail.com>

Copyright:

    <Copyright (C) 2008 Kouhei Maeda>

License:
（以下、省略）
```

- debian/dirs[*14]

```
usr/games
```

- debian/changelog

```
$ dch
hoge (1.0-1) unstable: urgency=low

  * Initial release

 -- Kouhei Maeda <mkouhei@gmail.com>  Sat, 10 May 2008 15:47:41 +0900
```

6.3.4 パッケージ作成

それでは、debuild コマンドでパッケージを作成してみます。

```
$ debuild -us -uc
```

deb ファイルができているのを確認します。

```
$ cd ../../
$ ls hoge_1.0*
hoge_1.0-1.diff.gz  hoge_1.0-1_i386.build    hoge_1.0-1_i386.deb
hoge_1.0-1.dsc      hoge_1.0-1_i386.changes  hoge_1.0.orig.tar.gz
```

上記のような簡単なものであれば、rules ファイルを修正しなくても、パッケージを作れる事が分かります。ちゃんとやるには、lintian のエラーや警告を潰し、pbuilder でのチェックやインストール、アンインストールのチェックをする必要がありますが、ここでは省略します。

6.4 複数のバイナリパッケージを作ってみる。

それでは、今度は先ほどの簡単なコードを一部変更して、それぞれ別のパッケージになるようにします。

6.4.1 ソースコード、Makefile の用意

下記のようなファイルを用意します。先ほどの hoge-1.0 をコピーして修正すればよいでしょう。

[*14] hoge のインストール先を指定します。

```
foo-1.0/
 foo.c
 bar.c
 Makefile
```

各ファイルは下記のようにします。

- foo.c

```
#include <stdio.h>

int main(void)
{
  printf(‘‘hello,foo\n’’);
  return 0;
}
```

- bar.c

```
#include <stdio.h>

int main(void)
{
  printf(‘‘hello,bar\n’’);
  return 0;
}
```

- Makefile

```
CC=gcc
CFLAGS=-O
BINDIR=/usr/games

foo:
	$(CC) $(CFLAGS) -o foo foo.c
	$(CC) $(CFLAGS) -o bar bar.c

install:
	install -d ${DESTDIR}${BINDIR}
	install -m 755 foo ${DESTDIR}${BINDIR}
	install -m 755 bar ${DESTDIR}${BINDIR}

clean:
	rm -f foo bar
```

上記を make すると、foo と bar という実行ファイルができます。今回は、これらをそれぞれ foo パッケージと bar パッケージの別々のパッケージにしてみます。

6.4.2 dh_make の実行

単一のバイナリパッケージとは違い、dh_make を実行するときのオプションを”–single” から”–multi” に変更して実行します。

```
$ cd foo-1.0
$ dh_make --multi -c gpl --createorig
Maintainer name : Kouhei Maeda
Email-Address   : mkouhei@gmail.com
Date            : Sat, 10 May 2008 16:32:38 +0900
Package Name    : foo
Version         : 1.0
License         : gpl
Type of Package : Multi-Binary
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now. You should
 also
check that the foo Makefiles install into $DESTDIR and not in / .
```

単一のバイナリパッケージの時と同様に、debian ディレクトリとその下にファイルが作成されます。

単一のバイナリの時に比べ、以下のファイルが増えていることが分かります。このテンプレートと同様なファイルが後ほどビルドする際に必要になります。

- foo-doc.docs
- foo-doc.install

6.4.3 debian ディレクトリ以下のファイルの編集

まず、単一のバイナリパッケージを作成するときと同じように修正して、ビルド出来るのか確認してみます。最初に先ほどと同じように今回は必要のないサンプルファイルを削除します。

```
$ cd debian
$ rm *ex *EX
```

次に、control ファイルを修正します。複数のバイナリパッケージを作成するには、空行で区切られた三つ以上のエントリが必要になります。

一つ目のエントリは、ソースパッケージ自身のことを記述するためのエントリです。二つ目以降のエントリが生成されるバイナリパッケージについての情報を記述するエントリです。

今回は、foo と bar の二つのパッケージを作成するので、dh_make で作成されたテンプレートを以下の様に変更します。

```
Source: foo
Section: games (<- セクションを変更)
Priority: extra
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: foo
Architecture: any
Depends: ${shlibs:epends}, ${misc:Depends}
Description: sample program foo (<- 説明を記載)
 foo foo foo (<- 説明の詳細を記載)

Package: bar
Architecture: any (<- 今回はアーキテクチャ依存なので any にする)
Description: sample program bar (<- 説明を記載)
 bar bar bar (<- 説明の詳細を記載)
```

また、単一のバイナリパッケージの時と同様に、下記ファイルを修正します。

- debian/copyright
- debian/changelog
- debian/dirs

修正後、シングルバイナリの時と同様に debuild を実行してみます。

```
$ debuild -us -uc
 dpkg-buildpackage -rfakeroot -D -us -uc
(中略)
dh_installdirs -s
# Add here commands to install the arch part of the package into
# debian/tmp.
/usr/bin/make DESTDIR=/home/user/foo-1.0/debian/foo
 install
make[1]: ディレクトリ ‘/home/user/foo-1.0’ に入ります
install -d /home/user/tmp/foo-1.0/debian/foo/usr/games
install -m 755 foo
 /home/user/foo-1.0/debian/foo/usr/games
install: cannot stat ‘foo’: そのようなファイルやディレクトリはありません
make[1]: *** [install] エラー 1
make[1]: ディレクトリ ‘/home/user/foo-1.0’ から出ます
make: *** [install-arch] エラー 2
dpkg-buildpackage: failure: fakeroot debian/rules binary gave error exit
 status 2
debuild: fatal error at line 1319:
dpkg-buildpackage -rfakeroot -D -us -uc failed
```

どうやら、単一のバイナリパッケージの時にはちゃんと作成されていた実行ファイルが今回は作成されていない様です。単一のバイナリパッケージを作成する時は、rules ファイルの build-stamp ターゲットの中で、$(MAKE) コマンドが記述されていましたが、複数のバイナリパッケージの場合は、rules ファイルの中で、$(MAKE) がコメントアウトされており、デフォルトでは make が実行されません。そのため、rules ファイルを修正する必要があります。

”–single” の時の rules ファイル

```
build: build-stamp

build-stamp: configure-stamp
 dh_testdir

 # Add here commands to compile the package.
 $(MAKE)
 #docbook-to-man debian/hoge.sgml > hoge.1

 touch $@
```

6.4.4 rules の記述

rules の必須ターゲットは、単一のバイナリパッケージの時と同様に、build, binary, binary-indep, binary-arch の 4 つです。

'–multi' オプションで作成される rules では build ターゲットは下記の様になっています。

```
build: build-arch build-indep

build-arch: build-arch-stamp

build-arch-stamp: configure-stamp

        # Add here commands to compile the arch part of the package.
        #$(MAKE)
        touch $@

build-indep: build-indep-stamp

build-indep-stamp: configure-stamp

        # Add here commands to compile the indep part of the package.
        #$(MAKE) doc
        touch $@
```

今回、foo も bar も、gcc でコンパイルするので、control ファイルでの Architecture セクションではどちらも any を記述しています。そのため、今回は rules ファイルでは、build ターゲットでは、build-arch、build-arch-stamp に依存しますが、build-arch-stamp の中でコメントアウトされている$(MAKE) を有効にする必要があるので、有効にして、ビルドします。

```
$ debuild -us -uc
 dpkg-buildpackage -rfakeroot -D -us -uc
(中略)
dpkg-buildpackage: full upload (original source is included)
Now running lintian...
(中略)
Finished running lintian.
```

今回は、うまくビルドできたように見えます。パッケージのビルド時には、**debian/パッケージ名**ディレクトリが作成され、そこにパッケージ含まれるファイルがコピーされます。ですので、ちゃんと期待するファイルがコピーされているのか確認してみます。

```
$ tree debian/{foo,bar}
debian/foo
|-- DEBIAN
|   |-- control
|   `-- md5sums
`-- usr
    |-- bin
    |-- games
    |   |-- bar
    |   `-- foo
    |-- sbin
    `-- share
        `-- doc
            `-- foo
                |-- README.Debian
                |-- changelog.Debian.gz
                `-- copyright
debian/bar
|-- DEBIAN
|   |-- control
|   `-- md5sums
`-- usr
    `-- share
        `-- doc
            `-- bar
                |-- changelog.Debian.gz
                `-- copyright

13 directories, 11 files
```

実行ファイル foo, bar が今度はきちんと作成されていますが、bar ファイルは期待するディレクトリ bar/usr/games 以下にコピーされていません。bar/usr/games ディレクトリ自体も作成されていない事が分かります。複数のバイナリパッケージを作成するには、rules ファイルを上記の変更だけでなく、さらにもう一ヶ所修正し、さらに以下のファイルを作成する必要があります。

6.4.5 rules ファイルで修正が必要な箇所

debian/rules

```
build: build-arch build-indep

build-arch: build-arch-stamp
build-arch-stamp: configure-stamp

        # Add here commands to compile the arch part of the package.
        $(MAKE) (<-最初の変更箇所。)
        touch $@
(中略)
install-arch:
        dh_testdir
        dh_testroot
        dh_clean -k -s
        dh_installdirs -s

        # Add here commands to install the arch part of the package into
        # debian/tmp.
        $(MAKE) DESTDIR=$(CURDIR)/debian/tmp install (<-debian/foo を
 # debian/tmp に修正します。)

        dh_instll -s
```

6.4.6 新規作成するファイル

- foo.install

```
debian/tmp/usr/games/foo
```

- bar.install

```
debian/tmp/usr/games/bar
```

複数のバイナリパッケージを作成する場合、ビルド時には、通常、ソフトウェアのファイルやディレクトリを、まず **debian/パッケージ名**ディレクトリ以外の一時ディレクトリ[*15] に仮インストールし、そこから **debian/パッケージ名**ディレクトリにコピーします。この時、dh_install コマンドと debian ディレクトリ以下の**パッケージ名.install**

[*15] 通常は debian/tmp ディレクトリなどを指定します。

ファイルが使用されます。

今回の場合、debian/foo.install ファイルと、debian/bar.install ファイルが使われ、ここに記載したディレクトリやファイルが、それぞれのパッケージ用のディレクトリにコピーされます。

つまり、それぞれ以下のようにコピーされます。

- foo.install に記載された”debian/tmp/usr/games/foo” ファイル
  → パッケージ foo 用に debian/foo/usr/games/ディレクトリ以下に
- bar.install に記載された”debian/tmp/usr/games/bar” ファイル
  → パッケージ bar 用に debian/bar/usr/games/ディレクトリ以下に

では、実際にビルドしてみます。

```
$ debuild -us -uc
```

ビルドした結果の **debian/パッケージ**ディレクトリと仮インストール用の **debian/tmp** ディレクトリを見てみましょう。

```
$ tree tmp foo bar
tmp
‘-- usr
    ‘-- games
        |-- bar
        ‘-- foo
foo
|-- DEBIAN
|   |-- control
|   ‘-- md5sums
‘-- usr
    |-- games
    |   ‘-- foo
    ‘-- share
        ‘-- doc
            ‘-- foo
                |-- README.Debian
                |-- changelog.Debian.gz
                ‘-- copyright
bar
|-- DEBIAN
|   |-- control
|   ‘-- md5sums
‘-- usr
    |-- games
    |   ‘-- bar
    ‘-- share
        ‘-- doc
            ‘-- bar
                |-- changelog.Debian.gz
                ‘-- copyright

14 directories, 13 files
```

今度はちゃんとコピーされています。パッケージが作成されているか見てみます。

```
$ cd ../../

$ ls foo_1.0* bar*
bar_1.0-1_i386.deb  foo_1.0-1_i386.build    foo_1.0.orig.tar.gz
foo_1.0-1.diff.gz   foo_1.0-1_i386.changes
foo_1.0-1.dsc       foo_1.0-1_i386.deb
```

pbuilder でチェック後、インストールしてみます。

```
$ sudo dpkg -i foo_1.0-1_i386.deb
 bar_1.0-1_i386.deb
未選択パッケージ foo を選択しています。
（データベースを読み込んでいます ... 現在 222038 個のファイルとディレクト
 リがインストールされています。）
（foo_1.0-1_i386.deb から）foo を展開しています...
未選択パッケージ bar を選択しています。
（bar_1.0-1_i386.deb から）bar を展開しています...
foo (1.0-1) を設定しています ...
bar (1.0-1) を設定しています ...
```

インストールできているか確認してみます。

```
$ dpkg -L foo
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/foo
/usr/share/doc/foo/changelog.Debian.gz
/usr/share/doc/foo/README.Debian
/usr/share/doc/foo/copyright
/usr/games
/usr/games/foo

$ dpkg -L bar
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/bar
/usr/share/doc/bar/changelog.Debian.gz
/usr/share/doc/bar/copyright
/usr/games
/usr/games/bar

$ dpkg -s foo bar
Package: foo
Status: install ok installed
Priority: extra
Section: games
Installed-Size: 44
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Architecture: i386
Version: 1.0-1
Depends: libc6 (>= 2.7-1)
Description: sample program foo
 foo foo foo

Package: bar
Status: install ok installed
Priority: extra
Section: games
Installed-Size: 40
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Architecture: i386
Source: foo
Version: 1.0-1
Description: sample program bar
 bar bar bar
```

6.4.7 補足

ちなみに、dh_make で”–multi” ではなく、”–single” を指定しても、ソースツリーの Makefile と、debian/rules の記述を変更したところ、複数のバイナリパッケージに分ける事ができたので、結局は make 次第でどうにでもできるのではないかと思います。

6.5 まとめ

前回を踏襲して、dh_make を使って、debhelper スタイルで複数のバイナリパッケージを作ってみました。さらに、単一バイナリパッケージとの違いを分かりやすくするため、非常に簡単なプログラムをパッケージにすることで、その違いを見てみました。

単一バイナリパッケージを作成するときと同じような感覚でやると、rules ファイルのところで躓くので、パッケージの作成は makefile を読めることが重要だということがよく実感できました。

6.6 参考書籍

- 入門 Debian パッケージ (ISBN:978-4-7741-2768-X)
- GNU Make 第 3 版 (ISBN:4-87311-269-9)
- 第 39 回 東京エリアデビアン勉強会 (2008 年 4 月) 「バイナリ一つだけのパッケージを作成してみる」
- The Debian Systems その概念と技法 (ISBN:978-4-8399-1897-2)

7 バージョン管理ツールを使い Debian パッケージを管理する Git 編

岩松 信洋

7.1 はじめに

最近の Debian Package は、VCS[*16]を使って管理できるようになっており、実際に利用しているメンテナも多くなりました。VCS と言っても様々ですが、Debian では代表的な VCS である CVS, Subversion から、最近流行りの Git、マイナーな darcs などを使うことができます。今回は、今一番イケていると思われる Git を使い、Debian Package をバージョン管理することによりどのような利点があるのか、どのようなツールを使ってメンテナンスを行えばいいのか、についてを説明します。

7.2 なぜパッケージも VCS で管理するのか

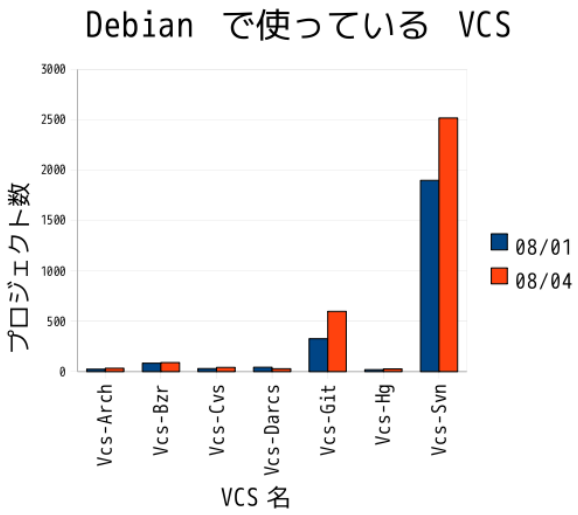
Debian Package を VCS で管理することによる利点は以下が考えれます。

- 異なるバージョンのパッケージを管理することができる。
  Debian パッケージメンテナは、stable, testing, unstable のソースコードを管理している事が多いです。また、多くの Debian バージョンのソースコードを管理することもあります。VCS を使うことによって、各バージョンの状態を管理しやすくなります。
- チームでの開発体制を容易に構築できる。
  Debian Package はチームでメンテナンスすることが多くなっています。VCS を使うことによって、チームでの開発が容易に行うことができるようになります。
- アップストリームとの連携
  アップストリームのソースコードを追従するために、Debian 側で VCS を使い、連携して開発を行うことができるようになります。また、Debian の独自パッチの管理、バグのトラッキングも容易になります。

7.3 Debian で使用可能な VCS

現在、 Debian で利用可能な VCS は Svn、Git、Bzr、Darcs、Arch、Hg、CVS です。プロジェクト管理用サーバである Alioth でもこれらの VCS は利用可能になっています。また、最近では Debian Package のタグに利用している VCS を付けることが可能になっており、各 VCS の利用状況を見ることも可能です。ここ数か月の VCS の利用状況を調べてグラフにしました。Git と Subversion を採用している開発者が増えていることがわかります。

[*16] Version Control System



7.4 ソースパッケージを Git で管理するためのツール git-buildpackage

では、実際に VCS を使って Debian Package を管理するためにはどのようにすればいいのでしょうか。Debian パッケージを Git で管理するためのツールとして、git-buildpackage があります。これを使うことによって、Git を使ってパッケージのソースコードを管理することができるようになります。インストールはいつものとおり、**apt-get** or **aptitude** を使ってインストールすることが可能です。

```
$sudo apt-get install git-buildpackage
```

7.4.1 git-buildpackage で提供されるコマンド

git-buildpackage で提供されるコマンドは表 1 の 4 つしかありません。これらのコマンドと Git コマンド を使って、パッケージのメンテナンスをすることになります。Git の細かい知識は必要ありませんが、基本的な使い方は知っておく必要があります。

提供されるコマンド	機能
/usr/bin/git-buildpackage	パッケージを作成する
/usr/bin/git-dch	Git のコミットログから Debian Changelog を作成する。
/usr/bin/git-import-dsc	既存の Debian Package を Git にインポートする。
/usr/bin/git-import-orig	アップストリームからリリースされたソースコードを Git にインポートする。

表 1 git-buildpackage で提供されるコマンド

7.5 Git の簡単な使い方

表 2 によく使う Git コマンドを紹介します。これだけ知っていれば、Git を使った最低限の開発を行うことが可能です。

7.6 既にパッケージ化されているものを Git で管理する

Debian パッケージには 2 つの状態があると考えられます。一つは既にパッケージ化されているもの、もうひとつは今から Debian パッケージにしようとしているものです。まずは、既に Debian パッケージになっているものを Git

Git コマンド（一部）	機能
git init	ローカルリポジトリを作成する。
git add	ローカルリポジトリのキャッシュ(index) に管理対象のファイルを追加する。
git commit	ローカルリポジトリに変更を反映する。
git rm	ローカルリポジトリから管理対象ファイルを削除する。
git diff	差分を取得する。
git branch	ブランチを作成する。
git checkout	作成したブランチをチェックアウトする。
git format-patch	パッチを作成する。

表 2　よく使う Git のコマンド

で管理する方法を説明します。

まず、git-import-dsc コマンドを使い、Git リポジトリに現在のソースコードの状態を取り込みます。コマンドのオプションに、パッケージの dsc ファイル[*17]を指定します。実行すると、パッケージ名でディレクトリが作成され、Git リポジトリが作成されます。また、ブランチとして、master ブランチと、upstream ブランチが作成されます。Debian 関係のコードは master ブランチ、Upstream のソースコードは upstream ブランチで管理されるようになります[*18]。

```
$ git-import-dsc ../isight-firmware-tools_1.0.2-1.dsc
Upstream version: 1.0.2
Debian version: 1
No git repository found, creating one.
Initialized empty Git repository in .git/
Everything imported under isight-firmware-tools
$ ls
isight-firmware-tools
$ cd isight-firmware-tools
$ git branch
* master
  upstream
```

7.6.1　インポート時のログ

パッケージをインポートしたときに、Git のコミットログに現在のバージョンのコミットログが書き込まれます。また、 Debian Version は Git のタグ機能を使い、タグ名として保存されます。

```
$ git log
commit 9c3669a233afe69d7be2aa8ad1995e6b19c841aa
Author: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
Date:   Sun Apr 6 21:48:40 2008 +0900

    Imported Debian patch 1.0.2-1
$ git tag
debian/1.0.2-1
upstream/1.0.2
```

7.6.2　ソースコードを変更し、修正を管理する

ソースコードを修正し、Debian パッケージ で配布する部分を管理するには、今までどおり、dpatch などのパッチ管理システムを使う必要があります。作成したパッチをリポジトリにコミットする時にに、**git add**, **git commit** コマンドを使い、リポジトリに反映させます。

[*17] Debian パッケージの制御ファイル
[*18] ブランチは git branch コマンドで表示可能

```
$ dpatch-edit-patch 05_change_ift-load_install_dir
... いろいろ修正 ...
$ exit
$ vi debian/patches/00list
$ git add debian/patches/05chage_ift-load_install_dir.dpatch
$ git commit -s debian/patches/00list debian/patches/05_chage_ift-load_install_dir.dpatch
/* エディタが起動するので、コミットログを記述 */

Change ift-load install dir.

Signed-off-by: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>

$ git log
commit c9865153ae1949956fdfe3827c0da9b36c2f0ddb
Author: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
Date:   Sun Apr 6 21:23:20 2008 +0900

    Change ift-load install dir.

    Signed-off-by: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
```

7.6.3 git-buildpackage を使った Debian パッケージの作成

Debian パッケージを作成するには、 git-buildpackage コマンドを使います。–git-ignore-new オプションは Git に反映されていない修正を無視するためのオプションです。

```
$ git-buildpackage --git-ignore-new -us -uc
```

7.6.4 パッケージをリリースする

新しい Debian バージョンのパッケージをリリースする場合は、**git-dch** コマンドに–**release** オプションを付けます。実行することにより、エディタが立ち上がり、Git のコミットログから、Debian Changelog が作成されます。Changelog を作成したら、**git-buildpackage** コマンドに –**git-tag** オプションを付けてパッケージを作成します。–**git-tag** を付けると、リポジトリに Debian バージョン用のタグが Debian changelog より作成され、リリース情報が付加されます。

```
$ git-dch --release
$ git-buildpackage --git-ignore-new --git-tag
$ git tag
debian/1.0.2-1
debian/1.0.2-2
upstream/1.0.2
```

7.6.5 新しいバージョンにする

新しいバージョンにするには、git-import-orig コマンドを使い、リリースされた新しいバージョンの Tar ボールを指定します。指定することにより、ファイル名からバージョンを取得し、新たに Upstream 用のタグが作成されます。また、アップストリームのバージョンが上がるため、自動的に Debian Changelog に 次の Debian パッケージバージョンが追記されます。

```
$ git-import-orig /tmp/isight-firmware-tools-1.2.tar.gz
Upstream version is 1.2.0
Importing '/tmp/isight-firmware-tools-1.2.tar.gz' to branch 'upstream'...
Switched to branch "upstream"
rm 'isight.rules.in'
rm 'po/fr_FR.po'
Created commit f5c85da: Imported Upstream version 1.2.0
 33 files changed, 4434 insertions(+), 1332 deletions(-)

.......<snip>

 src/udev.c                              |  164 +++
 33 files changed, 4434 insertions(+), 1332 deletions(-)
 rename po/{fr_FR.po => fr.po} (66%)
 create mode 100644 src/50-isight-firmware.fdi
 create mode 100644 src/callout.c
 create mode 100644 src/isight-firmware.fdi
 rename isight.rules.in => src/isight.rules.in (100%)
 create mode 100644 src/load.h
 create mode 100644 src/udev.c
Succesfully merged version 1.2 of /home/iwamatsu/Desktop/isight-firmware-tools-1.2.tar.gz into .
$ git branch
debian/1.0.2-1
debian/1.0.2-2
upstream/1.0.2
upstream/1.2
$ cat debian/changelog
isight-firmware-tools (1.2-1) unstable; urgency=low

  * New Upstream Version

 -- Nobuhiro Iwamatsu <iwamatsu@nigauri.org>  Fri, 11 Apr 2008 17:18:23 +0900
```

7.7 新たにパッケージ化する場合

新しくソフトウェアを Debian パッケージ にして、git-buildpackage で管理する場合は 最初に Git の機能が必要です。まず、ローカル Git リポジトリを作成します。次に作成したリポジトリに移動し、**git-import-orig** コマンドにアップストリームのソースコードを指定し、実行します。ソースコードは、gzip、bzip2 などで圧縮されたものと、展開されたソースコードディレクトリを指定することが可能です。また、実行する際に、**-u** オプションで、アップストリームのバージョンを指定する事が可能です。リポジトリを作成した後は upstream ブランチに移動し、**dh_make** 等を使ってパッケージの雛形作成し、上で説明した流れでメンテナンスを行います。

```
$ mkdir isight-firmware-loader-1.2
$ cd isight-firmware-tools-1.2
$ git init /* ローカル Git リポジトリを作成する */
$ git-import-orig -u 1.2 /tmp/isight-firmware-tools-1.2.tar.gz /* ソースコードをコミット */
Upstream version is 1.2
Initial import of '/tmp/isight-firmware-tools-1.2.tar.gz' ...
Succesfully merged version 1.2 of /tmp/isight-firmware-tools-1.2.tar.gz into .
$ git log
commit 9bf014aee2f834576f8f03d67ab66e8c85726832
Author: Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
Date:   Tue Apr 8 21:42:55 2008 +0900

    Imported Upstream version 1.2
$ git branch
* master
  upstream
$ git tag
upstream/1.2
$ git branch upsteam
$ dh_make
$ git branch master
```

8 アップストリームの VCS と付き合う

岩松 信洋

VCS を使ってアップストリームがソフトウェアの開発を行っている事が多くあります。アップストリームでは、Subversion を使っているが、パッケージメンテナは Git を使ってパッケージを行っている場合があったり、同じ VCS を使っている場合もあります。今回は Subversion を例にして、お互い VCS を使っている場合、どのように付き合っていくことができるのか説明します。

8.1 アップストリームが Subversion を使っている場合

Subversion で管理されているソースコードを取得したり、Subversion リポジトリへコミットするツールとして、git-svn パッケージがあります。 git-svn を使うことによって、容易に お互いのリポジトリ間を行き来することができるようになります。

8.1.1 Subversion のリポジトリから Git のリポジトリへ ソースコードを取得する

Subversion のリポジトリから Git のリポジトリへ ソースコードを取得するには適当なディレクトリを作成し、**git svn** の **clone** オプションを使って行います。取得した後は、Git の操作で開発を行うことができます。

```
$ mkdir test
$ git svn clone svn://test/trunk test-0.0.1
```

8.2 取得したコードを元に Debian パッケージ を作成する

取得したコードから新しく Debian パッケージ を作成するためには、自分でタグを付ける必要があります。**git-buildpackage** ではタグと Debian changelog から Upstream の情報を取得し、orig.tar.gz 相当のものを作成するため、タグを付ける必要があります。

```
$ git branch
master
$ git branch upstream
$ git checkout upstream
$ git tag upstream/0.0.1
$ dh_make --createorig
$ git branch master
.... Debian パッケージ 用のファイル作成などを行う ....
$ git-buildpackage -us -uc --git-ignore-new
$ debuild clean
$ git add debian
$ git commit -a
$ git-buildpackage -us -uc --git-ignore-new --git-tag
```

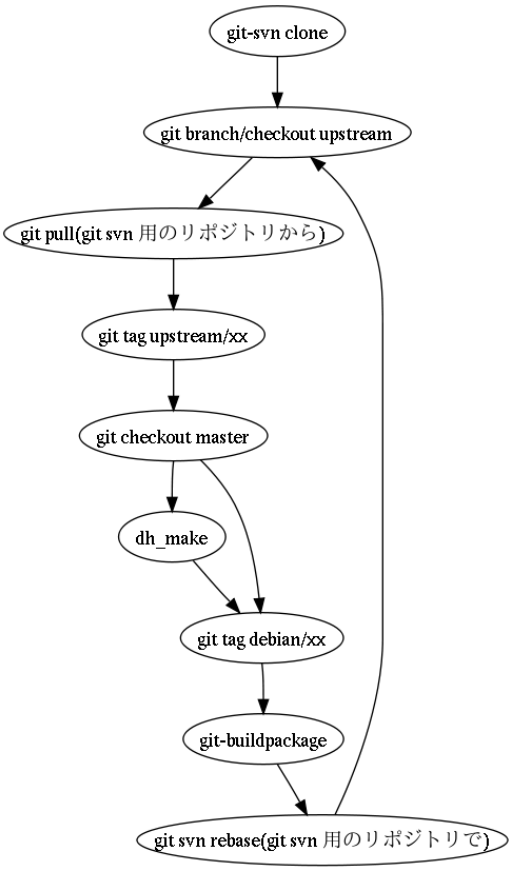
8.2.1 Subversion リポジトリの情報を取得する

Subversion リポジトリの情報を取得するには、 **rebase** オプションを使います。最初から git svn でリポジトリの操作を行っている場合は、rebase を使うことにより、Upstream のコードを パッケージ側に反映させることができます。

```
$ git checkout upstream
$ git svn rebase
```

8.3 すでにあるパッケージと Git リポジトリを元に Debian パッケージ を作成する

git svn で取得した Git リポジトリと 既にある Debian パッケージ を連携させるには操作が少し必要です。まず、**git-import-dsc** で 現在の Debian パッケージ を git-buildpackage で管理できるようにした後、upstream ブランチに git svn で取得したリポジトリから pull をします。pull することにより、コミットログの共有することができます。しかし、Debian Changelog の操作や、**git tag** を使ったタグの操作を手動で行う必要があるのが問題点です。**git-import-orig** を使って、tar.gz や ソースコードを指定して、マージすることも可能ですが、Upstream 側のコミットログが取り込まれないため、 Git を使うメリットがあまり無いと私は考えています。このあたりを改善していく事が今後の課題になりそうです。




```
$ git svn clone svn://svn.berlios.de/linux-uvc/linux-uvc/trunk\
 linux-uvc.git
$ git import-dsc  ../../../debian/linux-uvc_0.1.0.svn193-2.dsc
$ cd linux-uvc
$ git branch
* master
  upstream
$ git tag
debian/0.1.0.svn193-2
upstream/0.1.0.svn193
$ git checkout upstream
$ git pull ../linux-uvc.git/
$ git tag upstream/0.1.0.svn201
$ git checkout master
$ dch -v 0.1.0.svn201
$ git-buildpackage -us -uc --git-ignore-new
$ debuild clean
$ git commit -a
$ git-buildpackage -us -uc --git-ignore-new --git-tag
```

9　東京エリア Debian 勉強会の設計

上川 純一

Debian 勉強会は Debian の日本における発展を支援するために実施しているイベントです。Debian 勉強会を運営している上での仮説を説明します。勉強会はこの仮説に基づいて運営しています。まず前提条件を整理したのち、現状の状況を分析して、Debian 勉強会からどういう成果が期待できるのかを整理します。

9.1　Debian JP とは

Debian Project メンバーの日本在住の有志および Debian Project の Debian Developer になる候補者で構成されているのが Debian 開発者の会（Debian JP）です。Debian JP の会員はどういう人が存在しているのか、という点を議論したところ、下記の分類があぶり出されました。

- Debian JP の会員は開発・翻訳・インフラを担当している人により構成されています。過去の経緯により、既存の会則・方針では、Debian JP は「開発者の会」という位置づけのため、「ユーザ」については Debian JP「会員[*19]」ではありません。
- 開発者とは Debian JP で開発を行っている人、パッケージ・スポンサーされている人、New Maintainer、Debian Developer などをいいます。
- インフラは、Debian JP の運営に必要なインフラで ftp, www, svn, ML, LDAP などをいいます。

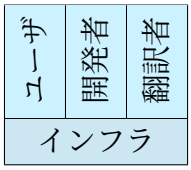


図 4　Debian JP のメンバー要素分類

Debian JP の目的としては、いろいろありますが、Debian の日本の開発者（Debian Developer）のコミュニティーとなることを目指しています。また日本における Debian の代表・とりまとめとしての役割をになっており、Debian が日本で利用しやすくなることの促進を行っており、翻訳作業のとりまとめや、外国から Debian Developer が来訪したときの宴会調整の連絡先として機能していたりします。また、Debian の商標を管理していたりします。

Debian JP が成功すると、Debian の日本語対応がよくなり、日本にいる Debian Developer がふえ、日本にいる Debian ユーザがふえ、日本で Debian を使うことがやりやすくなるはずです。

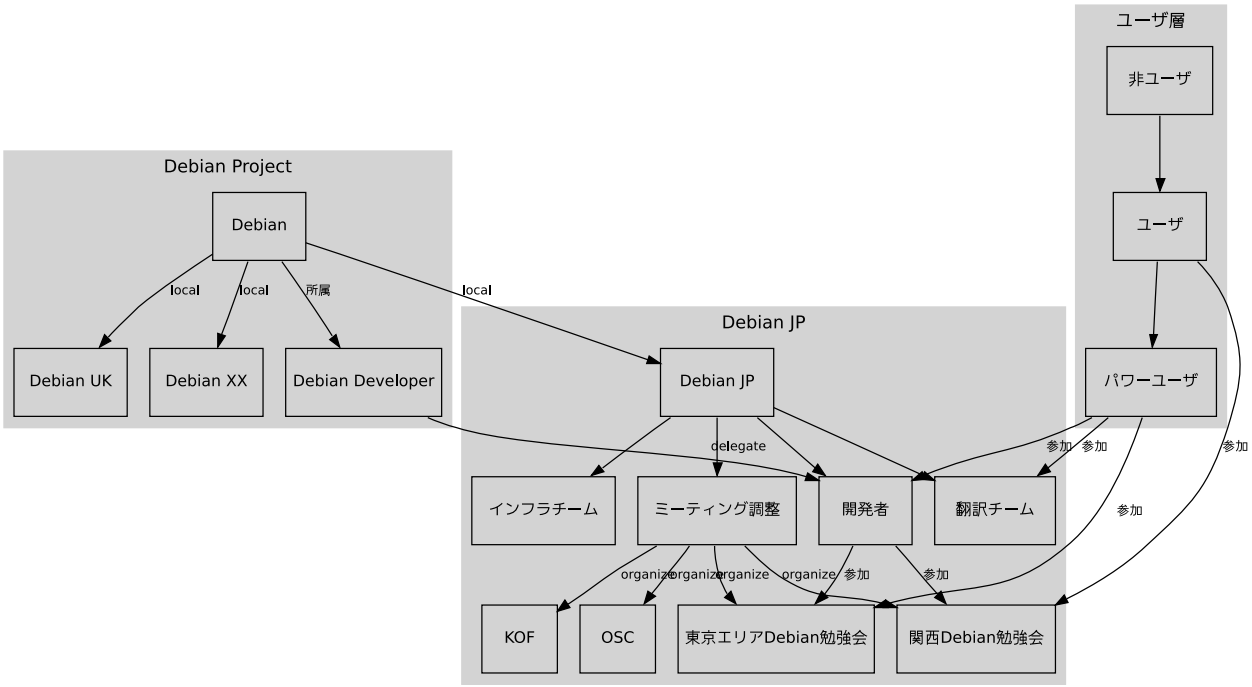
Debian JP の目的から考えると、開発者のコミュニティーとしての機能、およびユーザとの情報交換の機能の部分を保管するのが東京エリア Debian 勉強会でしょう。また、今後の Debian Developer 候補の発掘・育成（・そそのかし）のために活動すればよいでしょう。

[*19] Debian 開発者の会の「会員」は会則に定義されており、例えば投票する義務のある人のことです

9.2 Debian JP の各種会議体の位置付け

Debian Project には各種組織、および会議体が存在します。東京エリア Debian 勉強会を中心に、それらの関係を整理してみます。

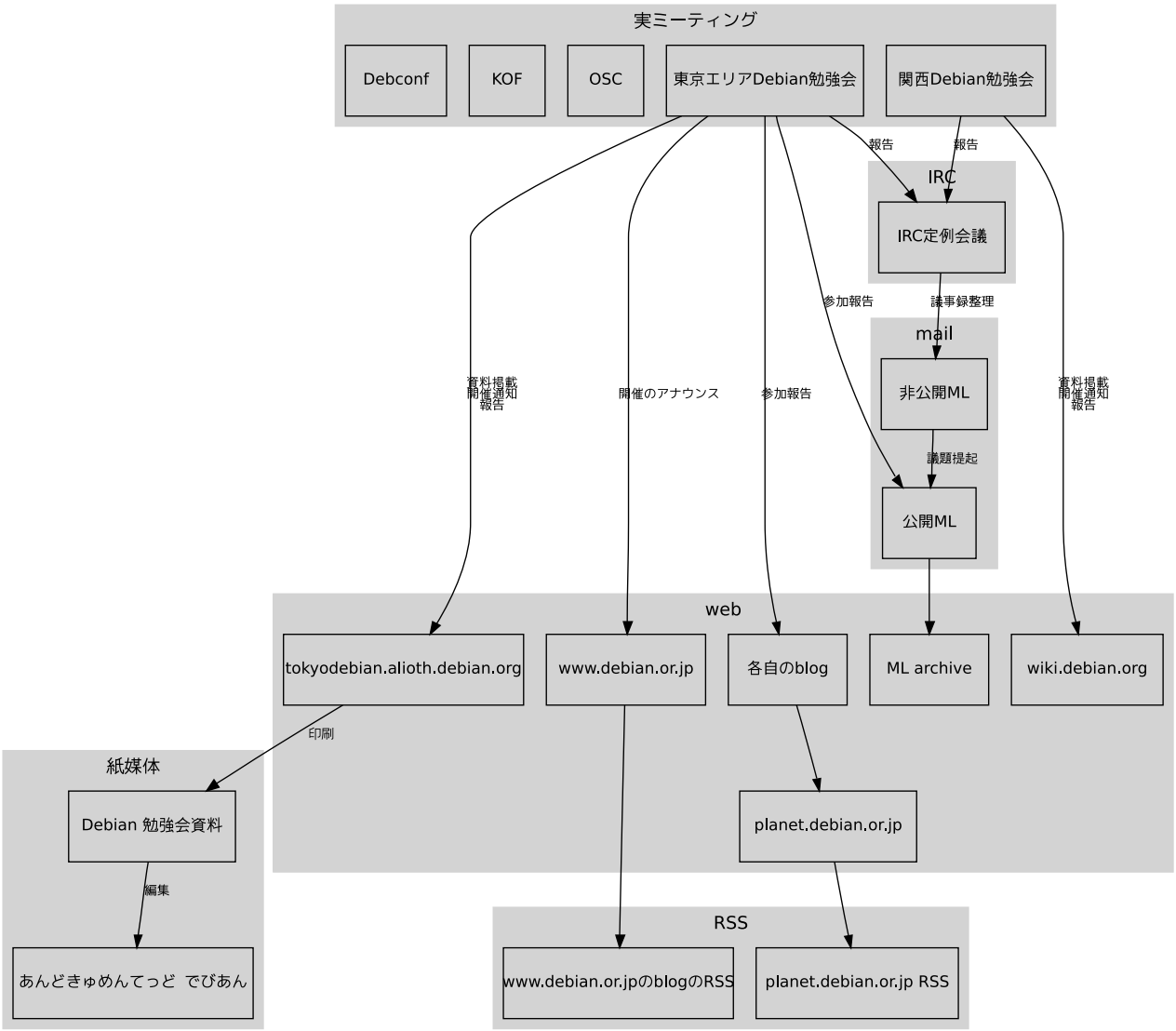
「東京エリア Debian 勉強会」は Debian Project の日本における組織である Debian JP が主体となり、「東京エリア Debian 勉強会幹事」に委任して開催しているイベントです。目的としているのは、Debian のパワーユーザと既存の開発者層に着目し、Debian Developer の育成、および翻訳活動に参加できる人たちへの情報提供、および必要な情報の交換です。



各種定期的に開催している会議体を整理してみます。

会	開催頻度	目的	参加者層
総会・選挙	年一回	Debian JP 内部の運営方針の策定	Debian JP 会員
OSC	年数回	新規ユーザ発掘・公報	OSC の一般参加者
東京エリア Debian 勉強会	月一回	Debian 開発者の新規発掘と支援	Debian の開発者をめざす東京近辺在住のメンバー
関西 Debian 勉強会	月一回	Debian ユーザ・開発者の新規発掘と支援	Debian を使う大阪近辺在住のメンバー
IRC 定例会議	月に二回程度	Debian JP の運営に関する情報共有と意思決定	Debian JP 会員

会議の情報が伝達される経路について整理してみます。Debian JP で利用している主な情報交換の方法を整理してみました。各種の会議が開催され、成果が報告されます。参加したメンバーが情報を得られるのはもちろんですが、参加していないメンバーもなんらかの情報が取得できます。事前資料は公開されるため、ウェブで取得できます。また、その他の情報取得の手段があります。



9.3 東京エリア Debian 勉強会においての事前資料の意義

「事前資料」は記録性、参照性を重視しています。2007 年 12 月の「事前資料」は当日の勉強会でも利用しますが、今年の勉強会の幹事・講師をしてくれる人の説明資料としても随時利用します。「これを読めば何をすればよいかがわかる」という形にしたいと思っています。プレゼンテーションに利用する資料は若干毛色が違い、議論をするためのフレームワークを提供するものだと考えています。

「事前資料」については、勉強会で紙の媒体の資料として利用するのと、コミックマーケットを契機として「あんどきゅめんてっど でびあん」として再度編集し、半年に一度まとめて製本します。そのことで、知識の精査、定着と展開がはかれることを目的としています。

9.4 事前課題の役割

東京エリア Debian 勉強会では、ただのセミナー参加でおわることは期待していません。理想的な最終目標としては、Debian Developer としてばりばりと貢献できるような意識の高い参加者を求めています。そこまでの理想を追い求めるのは現実的ではないにしても、何も準備せずに参加するより、何らかの心構えをして参加してもらった方がよいだろうということで、毎回事前課題を設定しています。いろいろなアイデアが出てきておもしろいです。

参加する際には事前課題は必須ということにしています。今年の参加者の事前課題の提出率を見ると半分以上が提

出していることが分かります。

表 3　2007 年の事前課題

	参加人数	事前課題提出人数	内容
2007 年 1 月	15	6	今後、勉強会につかう施設を提案してください、２００７年の勉強会の各月のアジェンダを提案してください
2007 年 2 月	13	8	apt に足りなさそうな機能, パッケージングをしてみて感じたこと、または何故パッケージングをしないか
2007 年 3 月	80	6	仮想化を実際にこういう利用方法で活用しています
2007 年 4 月	19	14	私はバージョン管理システムをこのようにつかっています
2007 年 5 月	23	14	「エッチになって困った事」
2007 年 6 月	4		
2007 年 7 月	18	12	今後 Debconf を日本で開催するために Debian の認知度を上げる方法、企業が Debconf のスポンサーになるためには
2007 年 8 月	25	18	ここ最近 Debian を使っていて ハマったこと/ちょっと感激したこと、apt の sources.list はこう書く
2007 年 9 月	14	7	「あなたが Debian で使っている MTA のこだわりの設定」もしくは「Debian で利用しているこんな便利な／楽しいメッセージツールあるいは日頃使っていて気にかかるメッセージ関連ソフトのこの部分」
2007 年 10 月	30		
2007 年 11 月	19	10	「Debian の Live CD ってこんなふうに使ってます」もしくは「ノート PC やデスクトップ PC ではなく、サーバ機器での Debian に期待するものって何？」
2007 年 12 月	11	11	「Debian 勉強会の目的と照らし合わせて 2007 年を評価してみた」と「２００８年の Debian 勉強会のために私はこうします」

9.5　12 月の勉強会の役割

毎年の忘年会をかねて東京エリア Debian 勉強会の 12 月開催は、一年間を反省する会を開催しています。12 月号の事前資料は、運営に関連する情報を共有するために準備しています。ターゲットは幹事・講師で、その説明文を見ることで講師や幹事が何をどういう理由で実施する必要があるのかを理解できることを目的としています。内容については定期的に修正が必要なものなので、毎年 12 月の資料として新規に作成しています。勉強会の目的は通常と違い、勉強会の実績の確認と、今後の方向性の策定だけをするという会にしています。

10 東京エリア Debian 勉強会のワークフロー

上川 純一

東京エリア Debian 勉強会は事前に会場も日時も決定してしまっています、そのため資料の準備・告知・予約などができる期限が決まっています。Debian 勉強会では月に二回行われる Debian JP IRC 定例会議と、幹事用のメーリングリストを利用して進捗の管理をしています。開催直前の最後の一週間はタイトになります。ワークフローの内容をみてみましょう。git レポジトリの `planner/200704-tokyo.ods` にファイルが置いてあります。

毎月の作業は次のようなものがあります。

会場確保　勉強会の会場を予約します。2ヶ月前程度に予約しないと大抵の会場はうまってしまいます。日程については年間スケジュールを事前に確定しておき、それから若干の調整をかける形にするのがよいでしょう。

企画立案　企画を立案します。前回の勉強会の宴会、もしくは年初の企画会議などで決定します。

講師確保　前回の勉強会であたりをつけて、講師候補のメールで調整するようにします。

資料作成依頼　資料の作成を講師に依頼します。資料の作成の方法についての資料も必要でしょう。

資料作成　講師が資料を作成します。git レポジトリを活用して共同作業をするのがよいでしょう。

資料編集　資料を集めて資料を編集します。元の資料が LaTeX でない場合には LaTeX 形式にする作業などがあります。ページ数にあてはまるようにする、用語を統一する、誤字脱字を修正する、などの対応を行います。

印刷向け編集（４の倍数）　冊子として印刷に出すときに紙は 4 ページの倍数である必要があります。そのために技を駆使します。内容をけずったり、画像の位置を微調整したり、フォントを小さくしたり、文章が一行におさまるように書き直したり、などの技があるようです。

kinkos **印刷依頼**　kinkos に印刷を依頼します。ウェブ経由でお願いすることができます。`http://www.kinkos.co.jp/` からアクセスして、「オンラインプリント」のメニューを選択します。半日程度（12 時間）は余裕をみてあげるのがよいでしょう。

kinkos **印刷受け取り**　印刷したものを受け取ります。面倒ですが、確認の意味も含めて、直接うけとって支払うのがよいでしょう。

事前課題設定　参加者に事前に考えてきてほしい内容を設定します。勉強会の次回の企画テーマにあうようなものがよいです。またハードルが高すぎると参加者がこまっちゃうので気を付けましょう。

事前課題提示　事前課題をウェブページに提示します。

資料反映　資料を反映します。

DWN Quiz **作成**　クイズを作成します。わからないとどうしようもないので、わかりやすいものをこころがけましょう。

DWN Quiz **景品準備**　何か景品を準備しましょう。

案内文文書案作成　案内文の素案を作成します。見た人が参加したくなるように、また参加して意味のある人が参加できるように、何を期待したらよいかを明確にします。これは幹事 ML でレビューにまわします。

案内文確定　レビューした後、案内文を確定します。各種メディアに展開します。

alioth **ウェブ掲載**　alioth のウェブページに掲載します。

debian.or.jp **掲載**　debian.or.jp のウェブページに掲載します。掲載の方法については `http://www.debian.or.jp/community/translate/webmasters.html` を参照。`www/trunk/src/community/events/index.tt2` のイベントの一覧に追加し、 `www/trunk/blosxom/data/` に blog エントリーを追加します。

mixi **掲載**　mixi の debian コミュニティーに掲載します。`http://mixi.jp/view_community.pl?id=95`

debian-devel **投稿**　Debian JP のメーリングリスト `debian-devel@debian.or.jp`, `debian-users@debian.or.jp` に投稿します。二週間前ごろをメドにします。

debian-devel **再度投稿**　数日前に再度投稿します。

予約システムの作成　宴会くん `http://utage.org/enkai/` などを利用し、予約用のエントリーを作成します。これは 3 週間前くらいにはつくっておきます。

参加者把握　数日前に参加者を把握しておきます。

出席簿印刷　当日のために出席簿を印刷します。

宴会予約　宴会の場所を予約します。場所によりますが前日までに予約したほうがよいです。店に電話で直接連絡します。

宴会費回収　宴会を開催、代金を回収します。5000 円、4000 円などのきりのよい金額が会費として徴収できるように調整します。東京 Debian 勉強会では講師をしてくれた人に関しては徴収しないようにしています。

会場設置　会場を設置します。事前に幹事がはやめに入ることが必要です。一人ではできないので、手伝ってくれる人がきてくれないと困るな。

出席確認　会費を徴収し、資料を配布します。受付担当者としては、参加者の方の名前を覚えるチャンスです。

結果報告書整理　実際に今回どういう会だったのかということを整理し、報告にまとめます。`http://tokyodebian.alioth.debian.org/` ウェブページから反応リンク集という形でまとめ、Debian JP IRC 会議にて内容を報告します。

年間一度している作業内容は次のようなものがあります。

年間スケジュール設定　一年間のスケジュールを決定します。内容についてはあまり重要ではありませんが、会場の予約のためには日程が決まっていることが重要です。

前年度の内容確認　出席者の傾向、開催内容の傾向と参加者数の増減、Debian JP の傾向などを分析し、どういう状況で、何が行われたのか、を確認します。

tokyodebian-XXX ML **作成**　事前課題を投稿してもらっているメーリングリストを再設定します。

11 東京エリア Debian 勉強会資料の準備の方法

上川 純一

11.1 文章ルール

文章は敬体に統一しましょう。

固有名詞は基本としては敬称略、フルネーム、で記述しましょう。日本名称の場合、苗字と名前の間には半角の空白を一文字入れます。

11.2 レポジトリの取得

まず最初に git のレポジトリを取得します[*20] 。読み込み専用であれば、git プロトコル、もしくは、http プロトコルでよいでしょう。書き込み権限を持っているのであれば、ssh プロトコルを利用すれば直接 git push でアクセスすることができます。

```
git clone git://git.debian.org/git/tokyodebian/monthly-report.git
git clone http://git.debian.org/git/tokyodebian/monthly-report.git
git clone ssh://git.debian.org/git/tokyodebian/monthly-report.git
```

この結果、カレントディレクトリに monthly-report というディレクトリができます。monthly-report/.git 以下がレポジトリです。git の使いかたについては 4 月の資料を参照してください。

```
$ ls -la monthly-report/ | head
合計 2500
drwxr-xr-x 28 dancer dancer   1960 2007-04-22 09:46 .
drwxrwxrwt 17 root   root      560 2007-04-22 09:46 ..
-rw-r--r--  1 dancer dancer    168 2007-04-22 09:46 .cvsignore
drwxr-xr-x  8 dancer dancer    240 2007-04-22 09:46 .git
-rw-r--r--  1 dancer dancer     61 2007-04-22 09:46 .gitignore
-rw-r--r--  1 dancer dancer     71 2007-04-22 09:46 .whizzytexrc
-rw-r--r--  1 dancer dancer    145 2007-04-22 09:46 .yatexrc
-rw-r--r--  1 dancer dancer  17989 2007-04-22 09:46 COPYING
-rw-r--r--  1 dancer dancer  25069 2007-04-22 09:46 ChangeLog
```

11.3 コミットの方法

まず、PDF ファイルが生成できることを確認します。Makefile があるので、make コマンドを入力するとビルドしてくれるはずです。文字コードが正しいか、正常にビルドできるか、などのチェックが組み込まれているので、チェックに活用しましょう。

```
make
```

[*20] git の使いかた詳細については、2007 年 4 月の勉強会資料を参照してください。 apt-get install git-core でインストールできます。

その後、git diff でコミットされる内容を確認します。意図している内容が表示され、問題ないようであれば、git commit コマンドでコミットします。手元のレポジトリに反映されます。

```
git diff
git commit -a -m 'revised XXX'
```

問題がないようであれば、git pull / git push でマージします。git pull した後にコンフリクトが発生したら、修正し、git commit でコミットしてから git push します。

```
git pull
git push
```

新規のファイルを追加する場合、ファイルを削除する場合には、 git add / git rm コマンドを利用します。

11.4 ファイルの編集

ドキュメントは pLaTeX で作成しています。ファイル名として下記になっています。(YYYY)(MM) は、年と月で、例えば 2007 年 12 月であれば 200712 です。

debianmeetingresume(YYYY)(MM).tex　事前配布資料
debianmeetingresume(YYYY)(MM)-presentation.tex　プレゼンテーション用 (prosper を利用)
image(YYYY)(MM)/　画像ファイルなどの置き場

作業する前にビルドに必要なパッケージをインストールします。

```
# tex から PDF の生成関連
apt-get install ptex-bin dvipdfmx latex-beamer \
 okumura-clsfiles gs-esp xpdf xpdf-japanese
```

編集に便利なツールもついでにインストールしてみてもよいでしょう。

```
apt-get install whizzytex advi emacs21 yatex gs-cjk-resource gv
```

tex4ht を利用して HTML 出力をさせる場合は下記もインストールしたらよいでしょう。ただし、2007 年 8 月現在、dvi2ps-fontdata-a2n の影響で dvi 出力ができなくなる副作用があります。

```
# tex4ht での HTML 生成関連
apt-get install dvi2ps-fontdata-a2n dvi2dvi dvipng tex4ht
```

文字コードは iso-2022-jp で統一しています[*21]。たとえば、emacs + yatex を使用している場合で iso-2022-jp をデフォルトにするには、下記のような設定を.emacs にかけばよいでしょう。

```
(add-hook 'yatex-mode-hook
          '(lambda ()
             (progn
               (if (string-match "^/home/user/tokyodebian/" default-directory)
                   (progn (set-buffer-file-coding-system 'iso-2022-jp)
                          (set-buffer-modified-p nil))))))
```

emacs での編集で、outline-mode を利用すると、アウトラインをベースに編集することができ、便利です。tex ファイルの最後に以下のようなエントリーを追加しています。M-x outline-minor-mode で有効にできます。

```
;;; Local Variables: ***
;;; outline-regexp: "\\([ <タブ記号>]*\\(\\(documentstyle\\|documentclass\\|<改行しない>
dancersection\\)\\*?[ <タブ記号>]*[[{]\\|[%<^L>]+\\)" ***
;;; End: ***
```

- **<タブ記号>**: タブを入力、

[*21] Windows 版と Linux 版の ptex で共通して扱える文字コードにしたという経緯があります。ただし現状 Windows で全部できる状況ではありません。

- <^L>: ctrl-L を入力、
- **<改行しない>**: ここの改行はみやすいように改行をいれているだけで、実際には改行は入力しない。

また、自動で適切な設定で outline-minor-mode に入るように .emacs に設定してもよいでしょう。

```
(add-hook
 'yatex-mode-hook
 '(lambda ()
    (make-variable-buffer-local 'outline-regexp)
    (setq outline-regexp
          "\\([ \t]*\\\\\\(documentstyle\\|documentclass\\|chapter\\|dancersection\\|
section\\|subsection\\|subsubsection\\|paragraph\\)\\*?[ \t]*[[{]\\|[%\f]+\\)")
    (setq
     outline-level
     (function
      (lambda ()
        (save-excursion
                  (looking-at outline-regexp)
                  (cond
                   ((equal (char-after (match-beginning 0)) 37) (- (match-end 0) (match-beginning 0)))
                   (t (let ((bs (buffer-substring (match-beginning 2) (match-end 2))))
                        (cond ((equal (substring bs 0 2) "do") 15)
                              ((equal (substring bs 0 1) "c") 0)
                              ((equal (substring bs 0 1) "p") 4)
                              ((equal (substring bs 0 2) "da") 1) ; dancersection
                              ((equal (substring bs 0 2) "se") 1) ;section
                              ((equal (substring bs 0 5) "subse") 2) ;subsection
                              ((equal (substring bs 0 8) "subsubse") 3) ;subsubsection
                              (t (length bs)))))))))
    (outline-minor-mode t)))
```

11.4.1 ドキュメントのスタイル

スタイルファイルは monthlyreport.sty パッケージを利用します。過去の資料を参考にしてください。

```
\usepackage{monthlyreport}
```

各担当部分は section として扱います。特別なコマンド dancersection で指定します。形式は \dancersecion{ **タイトル** }{ **作者名** } です。その中で subsection や subsubsection を利用して文書を構成してください。

```
\dancersection{Debian 勉強会資料の準備の方法}{上川 純一}
\label{sec:debmtg2007howtoprepare}
```

11.4.2 目次の処理

目次のエントリは下記の形式で作成します。

```
\index { alphabet もしくは、 ひらがなの読み @ 項目名称 }
```

11.4.3 画像ファイルの処理

画面写真の画像を追加するときは、できるだけサイズの小さい png などを利用してください。グラフなどの線画であれば、eps でかまいません。png であれば、ebb コマンドを利用して bounding box を作成してください。

```
ebb XXX.png
```

ps であれば、 eps2eps でバウンディングボックスを追加してあげるとうまくいきます。sodipodi の出力する ps を eps2eps で処理すれば sodipodi で画像を作成することができます。

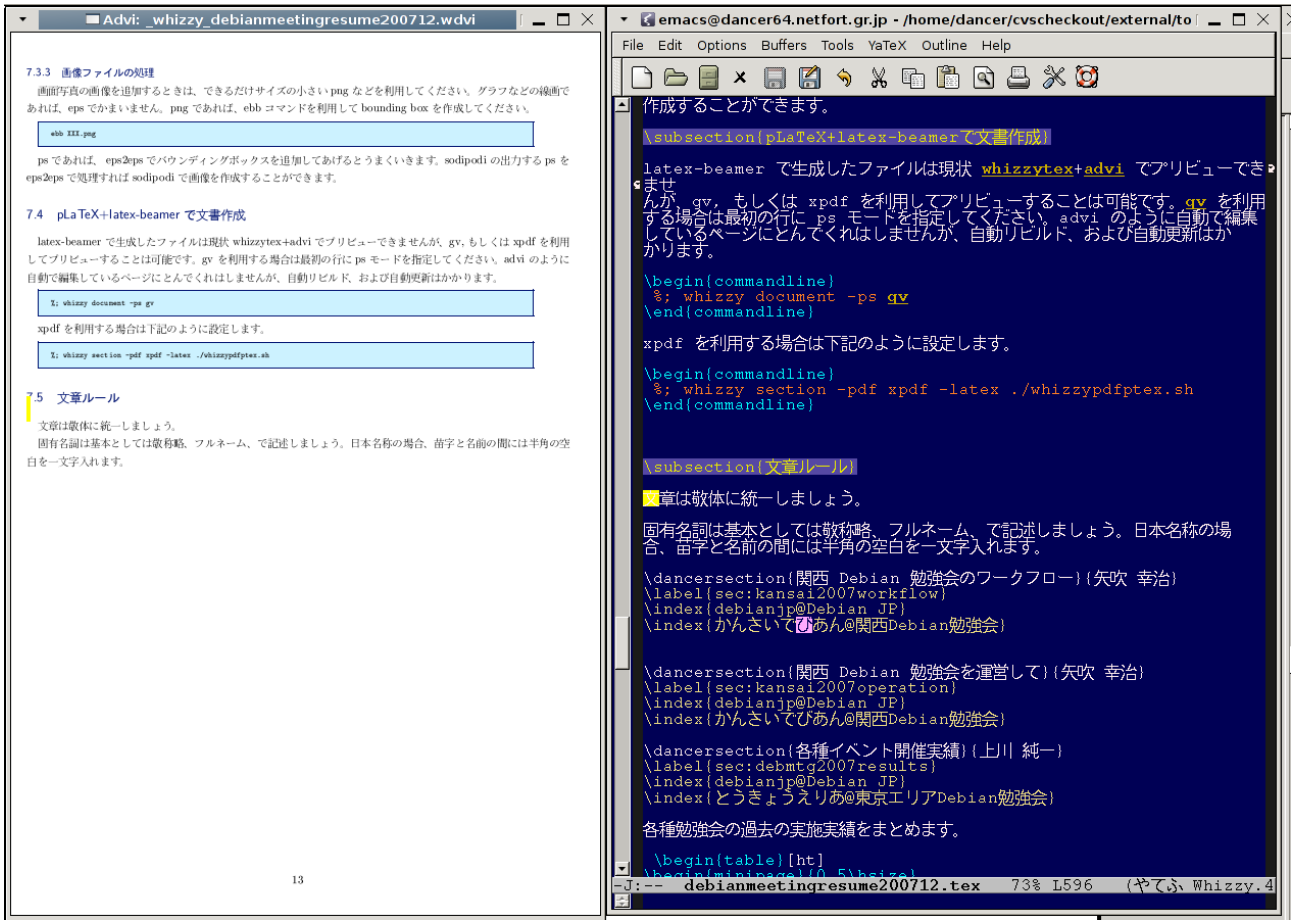
11.5 pLaTeX+latex-beamer で文書作成

latex-beamer で生成したファイルは現状 whizzytex+advi でプリビューできませんが、gv, もしくは xpdf を利用してプリビューすることは可能です。latex ソースファイルの最初の行のコメント部分に特殊な文字列を記述することで whizzytex は設定可能です。そこで、gv を利用する場合は ps モードを指定します。advi を利用しているときのように自動で編集しているページに飛んだりはしませんが、自動リビルド、および自動更新がかかるよういなります。

```
%; whizzy document -ps gv
```

xpdf を利用する場合は次のように設定します。

```
%; whizzy section -pdf xpdf -latex ./whizzypdfptex.sh
```



12 関西 Debian 勉強会

山下 尊也

12.1 関西 Debian 勉強会を運営して

2007 年の 1 年間、いろいろな人の支えがあって勉強会が成立したのだというのが、2007 年 12 月の忘年会が終わった後の私の率直な感想です。9 月に矢吹さんから私に担当が変わり、最初は運営の方でかなり悩んでいました。だって、宴会の幹事とかやった事がない 20 歳になったばかりの担当者ですから。しかし、回を重ね、それらは宴会などで運営についての会話を続ける事により、お互いが理解出来る状態になれたと思います。そのため、来年の目標は、「長続きできる勉強会」と言うテーマで運営していきます。

一人の力で運営を行うと、その一人が忙しくなった場合などに対応できず、バトンパスさえも難しい状況に陥ります。しかし、日頃から分担して作業を行なっていれば、これからも継続出きるでしょう。

また、Debian についてもっと知りたいと言う事で、関西 Debian 勉強会の有志で関西 Debian 勉強会とは独立した形で、週に一度、読書会 (KDR) を開いています。東京と色は違いますが、関西なりの色で私たち関西 Debian 勉強会は、より多く Debian を愛する人で集い、勉強し、お互いが成長出来る勉強会にしたいと考えています。

12.2 関西 Debian 勉強会のワークフロー

では、感想を述べたところで、実際のワークフローについて紹介します。

関西 Debian 勉強会では、東京エリア Debian 勉強会と同様に、Debian JP の IRC 定例会議に参加し、報告はしていますが、IRC 会議に出れる関西 Debian 勉強会のメンバーが少ないため、主に幹事用のメーリングリストを利用して進捗の管理をしています。まだ、確立出来てない部分もあり、最後の一週間はとてもタイトになります。この一年で関西なりに工夫したワークフローです。東京のワークフローを参考に作成しました。関西 Debian 勉強会では、共有作業の利便性の面で Google ドキュメントで管理しています。

会場確保　ほとんどの会場は、3ヶ月前から予約の受付が始まります。関西 Debian 勉強会は土日の午後を予約するため、2ヶ月前にはほとんどの場所が予約で埋まってしまいます。そのため、企画が決定するより前に会場を確保します。会場の料金支払いについては、前日までに入金すれば良いなど比較的融通効くので、とりあえず会場を確保します。また、プロジェクターを利用する場合は会場確保の段階で伝えておかないと、当日借りる事が出来ない場合もあります。気をつけましょう。

企画立案　企画を立案します。前回の勉強会の宴会で大抵決定します。12 月の忘年会で、どこらへんでお願いするかなどをお願いしておきます。

講師確保　前回の勉強会の宴会であたりをつけて、講師候補のメールで調整するようにします。

資料作成依頼　資料の作成を講師に依頼します。

資料作成　講師が資料を作成します。

資料編集　資料を集めて資料を編集します。関西 Debian 勉強会では、講師の負担を軽減するため、LaTeX に限定は

しておりません。そのため、OpenOffice での提出も可能です。

印刷向け編集　利便性の向上のため、PDF で結合します。pdftk[*22]などを使うと良いでしょう。

印刷　kinkos[*23] やカンプリ[*24]などの業者さんに持って行きます。カンプリの場合は、メンバー割引があり、kinkos の場合は、学生割引があります。開催者側の予定が合わない場合は、kinkos に Web から印刷を依頼する事をを検討しても良いかもしれません。

事前課題設定　関西 Debian 勉強会では、基本的に参加者の敷居を下げるために事前課題の設定はしておりません。ただし、希望などがある際は、ML で相談して事前課題を設定します。

事前課題提示　事前課題を Wiki に提示します。

資料反映　資料に反映します。

案内文文章案作成　案内文の作成をします。文面は前回の勉強会の時に使ったものを再利用して加工したものに対して ML で手直しを依頼します。

案内文確定　手直しなどがなくなったら、案内文を確定します。

mixi **掲載**　mixi の debian コミュニティ[*25]と、Debian 初心者！コミュニティ[*26] に投げます。また、女性の Linux 使いコミュニティ[*27]に投げてもらうために、女性の方に依頼します。

イベント管理システムの作成　参加者が参加申し込みのためのフォームを作成します。cotocoto 京都[*28]などを利用し、debian-users,debian-devel に投稿する前には作成しておきます。

debian-users,debian-devel **投稿**　Debian JP のメーリングリスト[*29] に投稿します。二週間前ごろをメドにします。

debian-users,debian-devel **再度投稿**　一週間前ごろをメドにします。ここで変更すべき内容は変更を行い、当日の予定とします。

DebianJP **ページ**　DebianJP 会員が、DebianJP のイベントのページを更新します。

参加者把握　数日前に参加者を把握しておきます。

出席簿印刷　当日か前日に出席簿を印刷します。

宴会予約　宴会の場所を予約します。店に電話で直接連絡します。最近の参加者の意見として、4000 円以内に収まる事が望ましいみたいです。

宴会費回収　宴会の最後に、代金を回収します。関西 Debian 勉強会では、講師の方と学生については 1000 円引きを実施しています。

会場設置　事前に幹事がはやめに入ることが必要です。開始時刻よりも前から会場を取っているので、30 分ほど前には集まり、会場を設置します。

出席確認　出席を確認します。幹事と講師はプロジェクターの設置などで忙しいため、それ以外の方にやっていただきます。

結果報告書整理　Debian wiki[*30]の関西 Debian 勉強会のページに反応集があるので、そこに記述を行います。まだ Debian Wiki のアカウントを持っていない方は、今後のために、アカウントを取っておきましょう。また、結果は DebianJP 定例会議や、次の勉強会で発表します。

資料公開　Debian wiki[*31]の関西 Debian 勉強会のページに資料を公開します。

[*22] http://packages.debian.org/sid/pdftk
[*23] http://www.kinkos.co.jp/
[*24] http://www.kanpuri.co.jp/
[*25] http://mixi.jp/view_community.pl?id=95
[*26] http://mixi.jp/view_community.pl?id=958407
[*27] http://mixi.jp/view_community.pl?id=513105
[*28] http://cotocoto.jp
[*29] debian-users@debian.or.jp,debian-devel@debian.or.jp,
[*30] http://www.debian.org
[*31] http://www.debian.org

13　各種イベント開催実績

上川 純一

各種勉強会の過去の実施実績をまとめます。東京エリア Debian 勉強会と、関西 Debian 勉強会について整理します。グラフにしてみると図 5 になります。

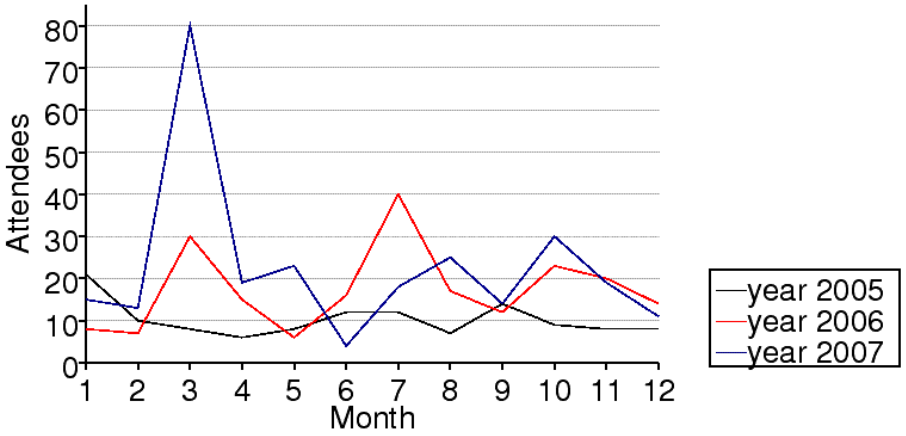


図 5　参加人数推移

表で見てみましょう。

表 4　東京エリア Debian 勉強会参加人数 (2005 年)

	人数	内容
2005 年 1 月	21	秘密
2005 年 2 月	10	debhelper1
2005 年 3 月	8	(早朝) debhelper2、social contract
2005 年 4 月	6	debhelper3
2005 年 5 月	8	DFSG、dpkg-cross、lintian/linda
2005 年 6 月	12	alternatives、d-i
2005 年 7 月	12	toolchain、dpatch
2005 年 8 月	7	Debconf 参加報告、ITP からアップロードまで
2005 年 9 月	14	debconf
2005 年 10 月	9	apt-listbugs、バグレポート、debconf 翻訳、debbugs
2005 年 11 月	8	DWN 翻訳フロー、statoverride
2005 年 12 月	8	忘年会

表 5　東京エリア Debian 勉強会参加人数 (2006 年)

	参加人数	内容
2006 年 1 月	8	policy、Debian 勉強会でやりたいこと
2006 年 2 月	7	policy、multimedia
2006 年 3 月	30	OSC: debian 勉強会、sid
2006 年 4 月	15	policy、latex
2006 年 5 月	6	mexico
2006 年 6 月	16	debconf、cowdancer
2006 年 7 月	40	OSC-Do: MacBook Debian
2006 年 8 月	17	13 執念
2006 年 9 月	12	翻訳、Debian-specific、oprofile
2006 年 10 月	23	network、i18n 会議、Flash、apt
2006 年 11 月	20	関西開催： bug、sid、packaging
2006 年 12 月	14	忘年会

表 6　東京エリア Debian 勉強会参加人数 (2007 年)

	参加人数	内容
2007 年 1 月	15	一年を企画する
2007 年 2 月	13	dbs, dpatch
2007 年 3 月	80	OSC 仮想化
2007 年 4 月	19	quilt, darcs, git
2007 年 5 月	23	etch, pbuilder, superh
2007 年 6 月	4	エジンバラ開催：Debconf7 実況中継
2007 年 7 月	18	Debconf7 参加報告
2007 年 8 月	25	cdn.debian.or.jp
2007 年 9 月	14	exim
2007 年 10 月	30	OSC Tokyo/Fall(CUPS)
2007 年 11 月	19	live-helper, tomoyo linux kernel patch, server
2007 年 12 月	11	忘年会

表 7　関西 Debian 勉強会参加人数 (2007 年)

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

14 東京エリア Debian 勉強会の 3 年間で生まれた Debian Developer は？

上川 純一

Debian 勉強会はなんだかんだといって、3 年間実施してきました。Debian 勉強会の当初の目標はばりばりとした開発者の育成をめざすところにあったはずです。直接的に計測するのは難しいですが、例えば Debian Developer ではなかった人が開発をする上での支援ができれば目標にそった活動ができていたといえるのではないでしょうか。では、確認してみましょう。

残念ながら Debian 勉強会のおかげで Debian Developer になったと言える人はまだいないようです。Debian 勉強会の参加者で、まだ Debian Developer でない人たちで、担当パッケージをもっている人たちを任意に抽出してその状況を確認してみました。

- 山根さん `http://qa.debian.org/developer.php?login=henrich@debian.or.jp`
  eclipse-nls-sdk、jd、ttf-kiloji、ttf-konatu、ttf-vlgothic メンテナンス中
- 岩松さん `http://qa.debian.org/developer.php?login=hemamu@t-base.ne.jp` `http://qa.debian.org/developer.php?login=iwamatsu@nigauri.org`
  [*32]linux-uvc、xfonts-mona、libflash、tinywm メンテナンス中
- 小林さん `http://qa.debian.org/developer.php?login=nori1@dolphin.c.u-tokyo.ac.jp`
  orpheus、serf、skkdic、skksearch メンテナンス中
- 三塚さん `http://qa.debian.org/developer.php?login=mitsuka@misao.gr.jp`
  canna メンテナンス中
- 矢吹さん `http://qa.debian.org/developer.php?login=yabuki@netfort.gr.jp`
  canna-shion、td2planet、yc-el メンテナンス中
- 山本さん `http://qa.debian.org/developer.php?login=yama1066@gmail.com`
  file-kanji メンテナンス中

ちなみに、この中で Debian として一番重要[*33]なパッケージは、 libflash で、その次が canna のようです。

[*32] 岩松さんがなぜ二つもメールアドレスを使い分けてるのかは不明です。
[*33] popcon での投票結果による

15 Open Source Conference 2008 Spring 報告

山本 浩之 / やまね ひでき / 岩松 信洋

2008 年 2 月 29 日と 3 月 1 日の両日に渡って東京新宿で開かれた「Open Source Conference 2008 Tokyo/Spring」に東京エリア Debian 勉強会有志が参加してきました。なお、関係者のスケジュールの都合上、東京エリア Debian 勉強会は 3 月 1 日のみの参加でした。今回は会場 4 階でのブース設営とセミナ、そして初の試みとしてハンズオンを開催しました。

15.1 セッション: Debian Overview

セミナは「Debian Overview」と題して、Debian の始まりの説明から DFSG とオープンソースの関係の紹介、現状の開発の活発さを示す指標として移植／派生ディストリビューション／開発者数と現状／パッケージについてなどを説明し、活況あふれる Debian コミュニティへの参加を促しました。参加者数は 40 名教室で 36,7 名でしたので盛況と言って良いのではないかと思います。

15.2 セッション: Debian Package ハンズオン

今回の OSC では、Debian 勉強会としては初の試みである ハンズオンを行いました。OSC の会場として専門学校を利用させていただいています。専門学校には実習室があります。以前から多くの方から依頼があった Debian Package のハンズオンを行いました。

簡単且つパッケージにする際に難しくないプログラムということで、sl [*34] をベースに、環境の構築、パッケージの作成方法、パッケージテスト方法、問題があったときの対処方法を実際の開発手順に合わせて、参加者に行ってもらいました。結果は岩松の不手際や会場での問題等でスムーズに進行させることができなかったのですが、おおむね好評だったようです。初の試みということと、今後につなげていくために、実際の参加者の方に意見を聞いて回りました。まとめたものを以下に示します。

1. 講師一人ではサポートしきれないので、参加人数にあわせたサポートチームを構成する。
   一人脱落者が出ると、一人では対応できない。人によってレベルが様々なので、対応しきれない。
2. キーボードを打つ人が遅い人のために印刷したものを用意してそれを配布する。ついていけない人はこの紙を読んで進めるようにサポート環境を作る。
3. 環境のレベルを低くする
   vi / emacs なんて使えないぜ！という人がいる。これはこれでいろんな意味で困るが、普段使い慣れたエディタが使えると一番よい。gedit などが使える環境なども必要。今回は手順を省くためにコンソールベースにしていましたが、マウスがないとだめな人もいるので、このあたりも考慮した環境を用意する必要がある。

[*34] http://packages.debian.org/sid/sl

4. 参加者の情報が必要
   入り口を狭めるという意味で情報を集めるわけではなく、チームでやる場合の人の配置などに使うためにスキルの情報があると楽になる。環境変数ってなに？とかいう人を一箇所に集めてフォローしやすくしたり、など。または思い切って参加者スキル制限を行うと良い事もある。
5. プレゼン用の画面とコンソール画面があるとよい
   ２つ同時に出せる環境が理想という意見がありました。会社で行う研修ではこのような方法を使う事が多いらしい。実際にはむずかしいので、紙と連携させてやるのがよい。
6. 45 分では足りない
   簡単な Deban Package でもハンズオンだと 45 分以上かかることが分かった。実際には割り込みが入ってしまい、もっと長くなる可能性もある。円滑に進めるには上に書いたようなことを行う必要がある。
7. ネットワーク制限がないところがいい
   不安要素が減るので。

今回のハンズオンで、いろいろ課題が分かりました。課題の解決と今後につなげていくために、近い将来にハンズオンのリベンジを行いたいと考えています。ハンズオンするにはそれなりの設備が必要なので、良い場所とかあれば教えてください。

15.3 セッション: 展示ブース

ブースでの展示では Debian JP への寄付金、計 9,520 円が集まりました。リアル Debian 掲示板には次のようなことが寄せられました：

- ラヴ・Etch ちなみに alpha ユーザです
- 関西から来ました！ Debian 最高 Death
- ３月の勉強会行きます！
- Air を Sid にしたよ
- Debian の Asterisk パッケージのバージョンを最新に！（誰か、よろ）
- 人脈作成中
- パッケージ作成中
- Debi りましょう !!

展示ブースではお金の管理の問題があり、寄付金と物販の売上の違いが分からなくなり、結局、一部寄付を強要するような形となりました。今後は、寄付金集めと物販は同時には行わないなどの対策が必要でしょう。

15.4 まとめ

次回に向けて、ブースの準備（配布用 LiveDVD 準備が間に合わなかった、寄付金管理）、セミナの時間配分（15 分オーバー）、ハンズオンの準備（段取りなど諸々）など課題も見えたイベントでしたが、検索エンジンなどでの Blog 投稿や mixi の日記などを見る限りでは、参加頂いた方々に楽しんでいただけたのではないかと思われます。

参加者の皆様、お疲れさまでした。

16 Nexenta Core Platform を使ってみる

上川 純一

16.1 はじめに

Nexenta[*35]とは Debian GNU/Linux のパッケージングシステムを OpenSolaris[*36]に移植したもののようです。OpenSolaris とは Solaris のオープンソース版のようです。Nexenta Core Platform が 2008 年 2 月にリリースされました。今回はそれを試してみます。

16.2 ダウンロード

http://www.nexenta.org/os/DownloadMirrors からリンクをたどりダウンロードします。今回は http://mirror.stanford.edu/nexenta/isos/nexenta-core-platform_1.0-b82_x86.iso.zip を利用しました。

unzip コマンドで展開すると iso イメージが作成されます。

```
[21:54:42]dancer64:nexenta> unzip nexenta-core-platform_1.0-b82_x86.iso.zip
Archive:  nexenta-core-platform_1.0-b82_x86.iso.zip
  inflating: nexenta-core-platform_1.0-b82_x86.iso
```

16.3 Qemu 環境でのインストール

まず、qemu 用のディスクイメージを作成します。

```
$ qemu-img create -f qcow2 nexenta.cow 3GB
Formatting 'nexenta.cow', fmt=qcow2, size=3145728 kB
```

qemu を起動します。[*37]

```
$ qemu-system-x86_64 -hda nexenta.cow \
 -cdrom  nexenta-core-platform_1.0-b82_x86.iso \
 -boot d \
 -m 512
```

メニューを選択し順番にインストール作業を進めます。

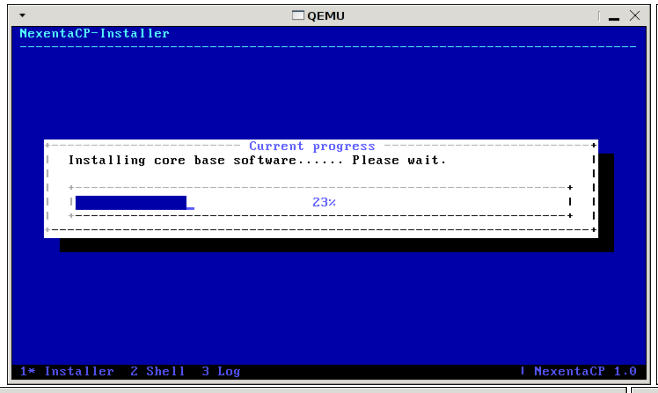
[*35] http://www.nexenta.org/os

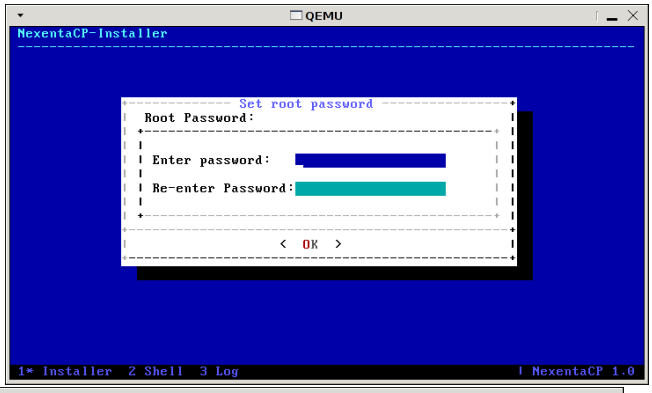
[*36] http://www.opensolaris.org/os/

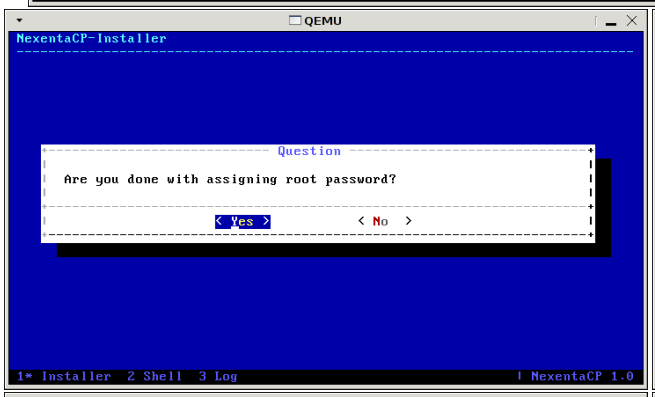
[*37] 時間の設定についてはただしい組み合わせを見つけられませんでした。BIOS 時間をローカルタイムとして扱うわけでもなく、UTC として扱うわけでもないように見えます。正しい設定をご存知でしたらご一報ください。

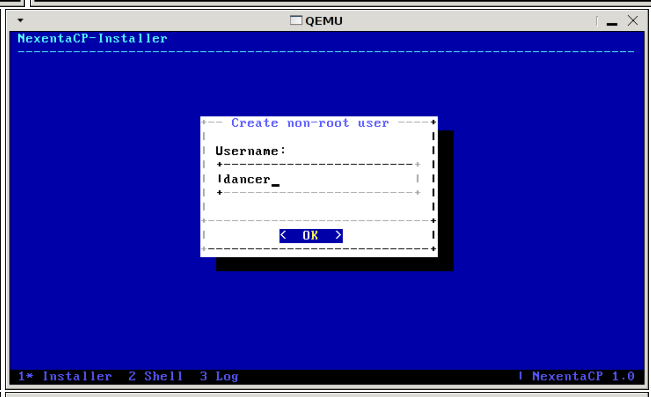
インストールが開始したらしばし待ちます。[*38]

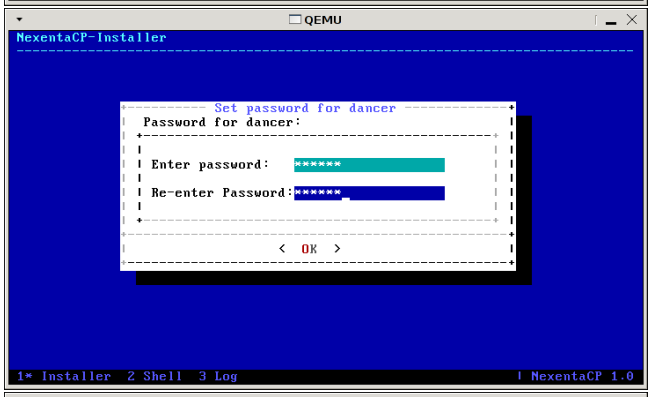
[*38] AMD Athlon 64 3500+ のマシン (64bit) で kqemu を利用して 2 時間かかりました

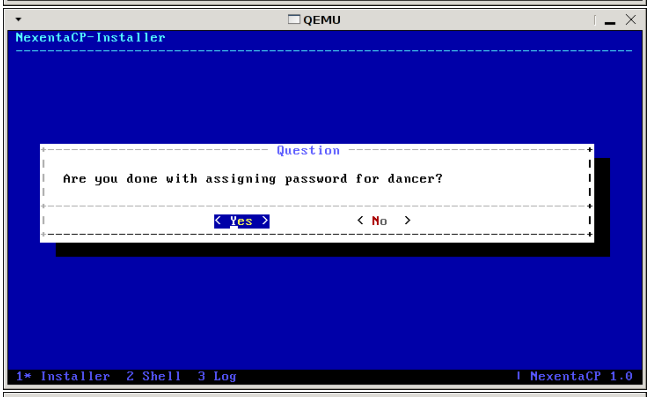

QEMU
NexentaCP-Installer
Current progress
Installing core base software..... Please wait.
23%
1* Installer 2 Shell 3 Log
NexentaCP 1.0

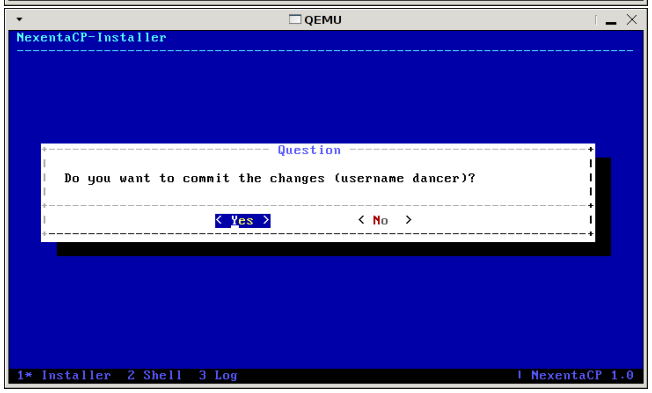

QEMU
NexentaCP-Installer
Set root password
Root Password:
Enter password:
Re-enter Password:
< OK >
1* Installer 2 Shell 3 Log
NexentaCP 1.0

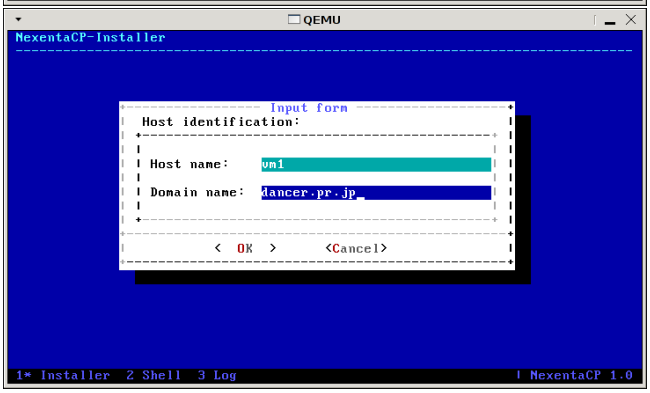

QEMU
NexentaCP-Installer
Question
Are you done with assigning root password?
< Yes >
< No >
1* Installer 2 Shell 3 Log
NexentaCP 1.0

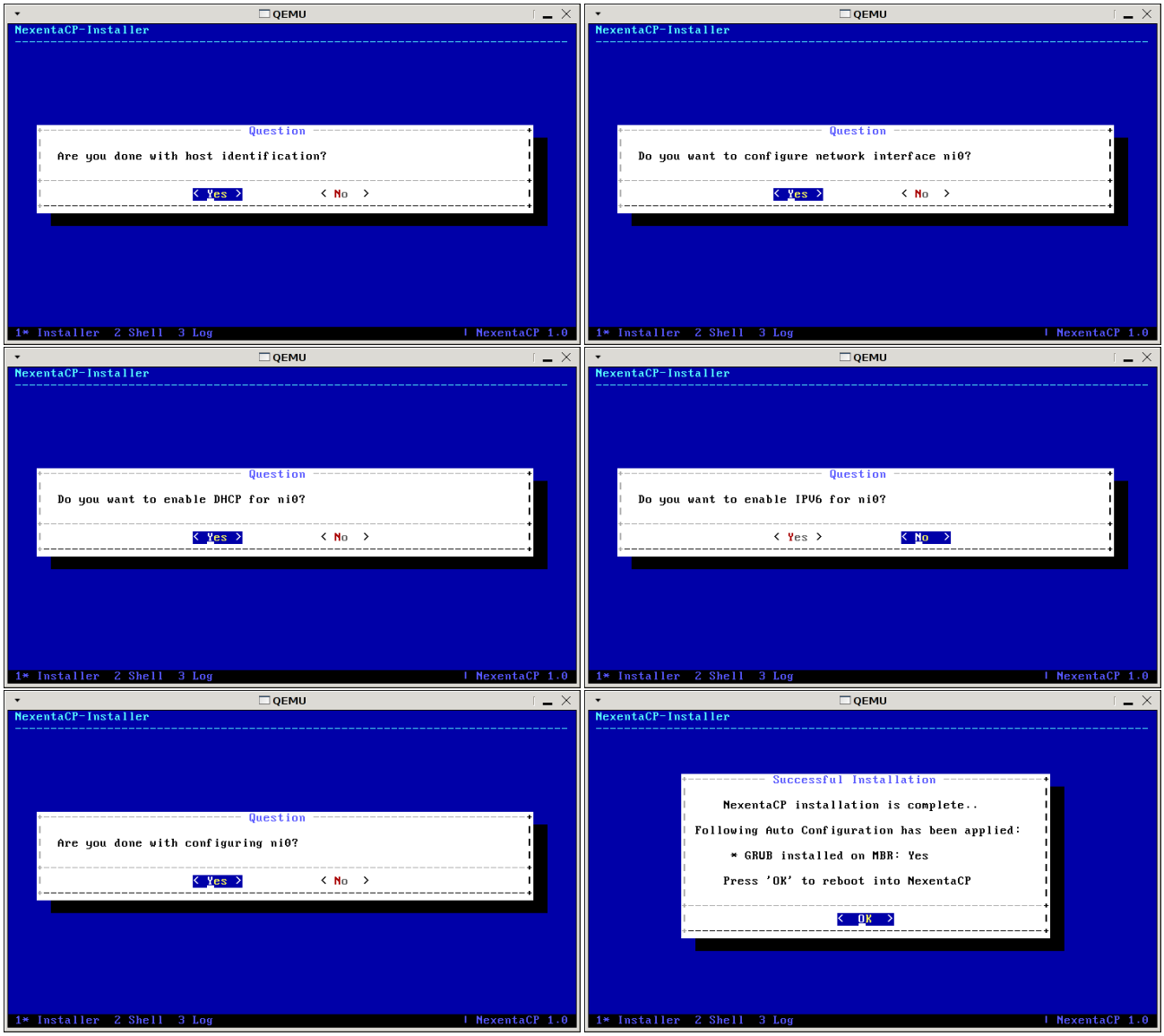

QEMU
NexentaCP-Installer
Create non-root user
Username:
!dancer
< OK >
1* Installer 2 Shell 3 Log
NexentaCP 1.0


QEMU
NexentaCP-Installer
Set password for dancer
Password for dancer:
Enter password: ******
Re-enter Password: ******
< OK >
1* Installer 2 Shell 3 Log
NexentaCP 1.0


QEMU
NexentaCP-Installer
Question
Are you done with assigning password for dancer?
< Yes >
< No >
1* Installer 2 Shell 3 Log
NexentaCP 1.0


QEMU
NexentaCP-Installer
Question
Do you want to commit the changes (username dancer)?
< Yes >
< No >
1* Installer 2 Shell 3 Log
NexentaCP 1.0


QEMU
NexentaCP-Installer
Input form
Host identification:
Host name: vm1
Domain name: dancer.pr.jp
< OK > <Cancel>
1* Installer 2 Shell 3 Log
NexentaCP 1.0

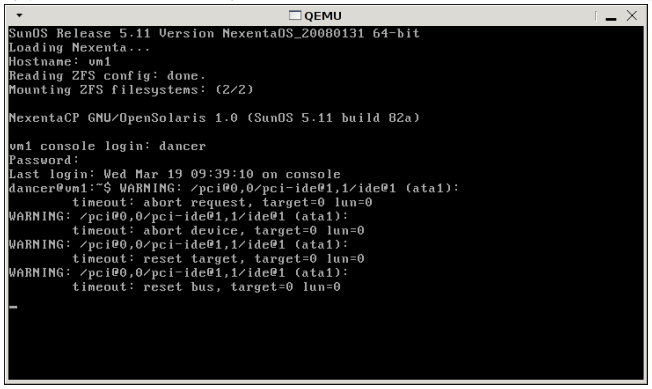


16.4 実行してみる

それでは、インストールした OS を起動してみましょう。メモリは 512MB 程度割り当てないと起動しないようです。

```
$ qemu-system-x86_64 -hda nexenta.cow \
 -m 512 -kernel-kqemu \
 -redir tcp:2222::22
```

手元のシステムでは、コマンドラインでログインができる状態まで 3 分程度かかりました。



通常のコンソールのログイン画面が立ち上がり、ログイン可能になります。

デフォルトで sshd [*39] などもインストールされているようです。netstat で確認すると listen しているようですので、ssh でログインして作業することが可能なようです。ここでは、 qemu で -redir オプションを使い、 SSH の 22 番ポートをホスト OS の 2222 ポートからアクセスできるようにしています。

-nographic -serial stdio オプションも追加してヘッドレスで稼働させることも可能です。ただし、デフォルトではシリアルコンソールには何も出力されないようです。

ネットワークデバイスは ni0 になっているようです。IP アドレスは qemu の機能で DHCP で提供されているアドレス（10.0.2.15）になっています。

16.5 Debian との互換性

初期インストールのパッケージ数が非常に多いです。OpenSolaris のデフォルトインストールに相当するものをそのままパッケージにしているのでしょうか？SUNW ではじまるパッケージ名などがそれっぽさを物語ります。

```
dancer@vm1:~$ dpkg -l | wc -l
464
dancer@vm1:~$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  adduser        3.80nexenta3   Add and remove users and groups
ii  alien          8.73.4         install non-native packages with dpkg
ii  apt            0.6.46.4nexent Advanced front-end for dpkg
ii  apt-utils      0.6.46.4nexent APT utility programs
ii  aptitude       0.4.4-1nexenta terminal-based apt frontend
[中略]
ii  sunw1394       5.11.82-1      Sun IEEE1394 Framework
ii  sunwaac        5.11.82-1      Adaptec AdvanceRaid Controller SCSI HBA Driv
ii  sunwad810      5.11.82-1      SUNW W1100z & W2100z Audio Drivers
ii  sunwadixp      5.11.82-1      SUNW Audio Driver for ATI IXP
ii  sunwadpu320    5.11.82-1      Adaptec Ultra320 Driver
ii  sunwafe        5.11.82-1      ADMtek Ethernet Driver
ii  sunwagp        5.11.82-1      AGP GART Driver
ii  sunwahci       5.11.82-1      Advanced Host Controller Interface (AHCI) SA
ii  sunwamd8111s   5.11.82-1      AMD8111 FAST Ethernet Network Adapter Driver
ii  sunwamr        5.11.82-1      LSI MegaRAID SCSI HBA Driver
[中略]
ii  sunwsshcu      5.11.82-1      SSH Common, (Usr)
ii  sunwsshdr      5.11.82-1      SSH Server, (Root)
ii  sunwsshdu      5.11.82-1      SSH Server, (Usr)
ii  sunwsshr       5.11.82-1      SSH Client and utilities, (Root)
ii  sunwsshu       5.11.82-1      SSH Client and utilities, (Usr)
ii  sunwtavor      5.11.82-1      Sun Tavor HCA driver
[略]
```

16.6 ないパッケージをビルドしてみた

/etc/hosts をいじってみようとしたら、シンボリックリンクになってました。しかも root 権限でも書き込み不可になっています。

```
root@vm1:/export/home/dancer# ls -l /etc/hosts
lrwxrwxrwx 1 root root 12 Mar 19 08:15 /etc/hosts -> ./inet/hosts
root@vm1:/export/home/dancer# ls -l /etc/inet/hosts -l
-r--r--r-- 1 root sys 1078 Apr  6 10:28 /etc/inet/hosts
```

気をとりなおして、 /etc/apt/sources.list に適当にソースを追加します。

```
deb-src http://cdn.debian.or.jp/debian unstable main contrib non-free
```

apt-listbugs が存在しないみたいなので、ビルドしてみようと思います。

まず、なぜだか racc, rdtool が足りないのでビルドしてインストールします。無事に apt-listbugs 自体もビルドできたので、意気揚々とインストールしてみます。

[*39] Debian のパッケージとは異なり、SUNWsshdu, SUNWsshdr などのパッケージになっています。見た感じは alien で作成されているパッケージのようです。

```
root@vm1:/tmp/apt-listbugs-0.0.88# debi
Selecting previously deselected package apt-listbugs.
(Reading database ... 27441 files and directories currently installed.)
Unpacking apt-listbugs (from apt-listbugs_0.0.88_all.deb) ...
dpkg: dependency problems prevent configuration of apt-listbugs:
 apt-listbugs depends on ruby (>= 1.8); however:
  Package ruby is not installed.
 apt-listbugs depends on libruby1.8 (>= 1.8.5); however:
  Version of libruby1.8 on system is 1.8.4-1nexenta1.3.
 apt-listbugs depends on libdpkg-ruby1.8 (>= 0.3.2); however:
  Package libdpkg-ruby1.8 is not installed.
 apt-listbugs depends on libgettext-ruby1.8; however:
  Package libgettext-ruby1.8 is not installed.
 apt-listbugs depends on libxml-parser-ruby1.8; however:
  Package libxml-parser-ruby1.8 is not installed.
 apt-listbugs depends on libhttp-access2-ruby1.8 (>= 2.0.6); however:
  Package libhttp-access2-ruby1.8 is not installed.
dpkg: error processing apt-listbugs (--install):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 apt-listbugs
debi: debpkg -i failed
```

残念、いくつかエラーがあるようです。ruby 自体が古いというエラーや、いくつかのライブラリが足りないというエラーがでています。

解決できそうな問題も、解決できなさそうな問題もあります。先は長いか？

16.7 まとめ

今回は Nexenta Core Platform をとりあえず仮想マシンにインストールしてみました。いろいろと Debian GNU/Linux と違う部分があったので、今後どのように開発がすすむのか、気になります。

17 GIS on Debian GNU/Linux!

清野 陽一

17.1 はじめに

17.1.1 社会的整備の進展

- 国際情勢
  - 地図情報を電子化して扱おうという試みは 1960 年代ごろから始まっています。当初は研究目的でした。
  - メインフレーム上ではなく、ワークステーションやパーソナルコンピュータの発達に伴い、1980 年代頃からこれらのコンピュータ上で動かせるシステムが普及し始めました。
  - 後述する地上観測衛星（ランドサット衛星 1 号機の打ち上げは 1972 年）やスペースシャトルの運用により、リモートセンシングなどの活用が 1970 年代より行われるようになりました。
- 日本国内
  - 行政サイド

    日本においては、国土に関する数値情報の電子化

    「平成 7 年 1 月の阪神・淡路大震災の反省等をきっかけに、政府において、GIS に関する本格的な取組が始まった。」（国土地理院 GIS[*40]による）

    →国を挙げての政策実施。2002 年の小泉内閣における「e-Japan 重点計画-2002」の重点政策のうち、4 番目の「行政の情報化及び公共分野における情報通信技術の活用の推進」において GIS の推進が盛り込まれる。ex) 国土地理院の数値地図 (CD-ROM 版) は平成 9(1997) 年頃から整備されるようになってきています。
  - 建設サイド
    - CAD(Computer Aided Design) の普及
    - 電子納品・標準化
    - 建設省・国土交通省の後押し

17.1.2 個人レベル

- Google Earth、NASA World Wind などの無料かつ高性能なデスクトップ地図ビューワーの登場

  →”Google Earth ショック”
- Google Map などの Web マッピングサイト

  → MashUp が流行

  →空間情報を個人が扱う時代になった

[*40] http://www.gsi.go.jp/GIS/whatisgis.html

17.2 GIS とは

17.2.1 定義

- 国土地理院[*41]の説明

  地理情報システム（GIS：Geographic Information System）は、地理的位置を手がかりに、位置に関する情報を持ったデータ（空間データ）を総合的に管理・加工し、視覚的に表示し、高度な分析や迅速な判断を可能にする技術です。

- GIS ポータルサイト[*42]の説明

  GIS（Geographic Information System：地理情報システム）とは、位置や空間に関する様々な情報を、コンピュータを用いて重ね合わせ、情報の分析・解析をおこなったり、情報を視覚的に表示させるシステムです。元々は専門的な分野での利用が一般的でしたが、最近では、私たちの生活の中での身近な利用へと、その活用範囲が広がってきています。

- 国土交通省国土計画局[*43] の説明

  位置や空間に関する情報をもったデータ（空間データ）を総合的に管理・加工し、視覚的に表示できる高度な分析や迅速な判断を可能にする技術です。

- ESRI ジャパン社[*44]の説明

  GIS とは、Geographic Information System の略で、広義には「実世界を空間的に管理することにより、より合理的な意思決定を行おうとするアプローチ全般」を意味しますが、狭義には、「空間情報を作成、加工、管理、分析、表現、共有するための情報テクノロジ」を意味します。

- インフォマティクス社[*45]の説明

  GIS とは、Geographical Information Systems（地理情報システム）の略で地図上に様々な情報を重ね合わせて表示・編集したり、分析するシステムのことをいいます。

”GIS”と”GPS”の違いってわかる？

混同している人を良く見かけます。GPS とは ”Global Positioning System”の略です。
全地球測位システム、汎地球測位システムの事です。アメリカの衛星システムや元軍事用として使われています。
カーナビとか携帯電話に入っているものです。

17.2.2 大別して 2 タイプ

- 管理系 GIS
  - 統合型・全庁型

    商用製品が主流。ラージスケール指向。分析もできます。

    → ESRI 社 ArcGIS、インフォマティクス社の SIS、MapInfo 社の MapInfo などです。

    →実際、助成金などでお金がある人は、個人研究者でも導入する人は多いようです。

    →その場合、操作方法を知っていることで就職時のアドバンテージにもなるようです。
  - 表示系：Google Earth、Google Map みたいなものです。
    - mapserver:WebGIS サーバ[*46]
    - ka-Map[*47]

[*41] http://www.gsi.go.jp/GIS/whatisgis.html
[*42] http://www.gis.go.jp/contents/about/whatis/index.html
[*43] http://www.mlit.go.jp/kokudokeikaku/gis/aboutgis/index.html
[*44] http://www.esrij.com/whatisgis/gis/index.shtml
[*45] http://www.informatix.co.jp/sis/aboutgis/aboutgis.html
[*46] http://mapserver.gis.umn.edu/
[*47] http://ka-map.maptools.org/

  - CAD との関係
    設計・測量・建設・土木・防災と密接に連携します。
    →境目が曖昧になってきています
    → Autodesk(AutoCAD) 社のオープンソースへの参入
- 解析系 GIS
  - GRASS GIS[*48]
    1980 年代半ばから開発開始しました。
  - Quantum GIS[*49]
    高機能なビューワ。簡単な解析機能も持ちます。
  - SAGA GIS[*50]
    Release 2.0.1 は Gentoo Linux の Portage 用が用意されてるようです。
  - Mandara[*51]
    Windows 用の GIS ソフトです。初心者向けです。GIS がどういうものかをてっとり早く知りたい人にはお手軽かもしれません。MS Excel との連携もできます。

17.3 最近の話題

- OSGeo 財団[*52]日本支部[*53]
  大阪市立大学や株式会社オークニーなどが中心
- 関西オープンソース 2007 におけるカンファレンス[*54]
  「特別企画：OSGeo.JP　創造都市を支えるオープンソース GIS の最前線」
- 書籍の整備
  -『入門 Web マッピング（*Web Mapping Illustrated*)』(2006 年 5 月、オライリー)[*55]
  - *Open Source Gis: A Grass Gis Approach*(2007 年末に 3rd Edition が出た、Springer)[*56]

17.4 なぜ Debian GNU/Linux か？

17.4.1 Debian GNU/Linux を選んだ動機 (個人的経験)

★ オープンソースなソフトウェアが使いたかった。
→高機能なプロプライエタリなソフトウェアは沢山あるが、高価！組織や補助金で導入した場合はライセンスの問題が . . .。
→プログラマブルなのもいいかも (勉強になるかも)

★ DebianGIS のようなプロジェクトがある（最近は停滞気味？）

○ 初心者的に、パッケージの管理が楽そうだった (APT 万歳!!)。

● でも、使いこなせるなら他のディストリビューションでも OK？
→ソースコードは公開されてることが多いし、Debian GNU/Linux じゃなきゃダメなバイナリというのは無いと思う。
→ Windows 上で使える環境も着々と進んできている。まだ Unix/Linux にアドバンテージがありますが。

[*48] http://grass.itc.it/
[*49] http://www.qgis.org/
[*50] http://www.saga-gis.uni-goettingen.de/html/index.php
[*51] http://www5c.biglobe.ne.jp/~mandara/
[*52] http://www.osgeo.org/node/271
[*53] http://www.osgeo.jp/
[*54] http://www.osgeo.jp/?page_id=8
[*55] http://www.oreilly.co.jp/books/4873112826/
[*56] http://www.grassbook.org/

→ MacOS は GIS はあんまり得意じゃないかも . . .。

- 海外では Mandriva とか (かつての ArcheOS とか)Gentoo Linux(Gentoo-GIS overlay Project [*57]なんてのがある。) などもよく利用されているようです。
- 最近は人に勧めるとしたら Ubuntu か . . .。(個人的に紹介実績有り)

17.4.2 Debian GNU/Linux におけるパッケージ、ライブラリなど

DebianGIS[*58]に紹介されている Debian GNU/Linux におけるパッケージ、ライブラリを以下に示します。

```
Geospatial packages of core concern
    Meta-packages
        *education-geography task from debian-edu: DebianGIS recommends these additions: qgis, gmt, gdal, proj.
 (education-geography already depends on grass):
    Binary packages
        *PostGIS
            RDBMS の PostgreSQL の地理情報拡張
        *GDAL(Geospatial Data Abstraction Library 地理空間データ抽象化ライブラリ？)
            座標系（投影）変換ライブラリ
        *GRASS
        *QuantumGIS. Includes qgis-plugin-grass
        *Mapserver
        *GEOS
        *PROJ: proj-doc: there are PDF files available, perhaps those would be better?
        *Earth3D
        *gpsd
        *gpsdrive
        *gpx2shp
        *gpsman
        *gpsmanshp
        *gpstrans
        *gpsbabel
        *thuban
        *gmt
        *OpenSceneGraph
        *OpenThreads
        *Avce00 and E00compr
        *OGDI
        *OpenJUMP: in debian/contrib, because it depend on batik and sun jre. Work is being done to get batik into
debian/main, and openjump to work with GNU Classpath. Packaged instead of JUMP
        *JTS
        *TerraLib
        *mapnik
        *marble
        *Viking: a GPS track editor and analyzer
        Useful packages to be packaged
        Sorted by approximate priority.
            Libraries and bindings
            *Python Shapelib binding
            *Python Cartographic Library (PCL)
            *Tcl Shapelib binding
            *Ruby Shapelib binding
        Desktop/Analysis/Database
            *Ossim: currently being worked on (Francesco Lovergine)
            *PostLBS: powerful routing solution for PostgreSQL/PostGIS
            *TerraView: powerful GIS based on TerraLib (already in Debian, but obsolete)
            *OpenModeller: species occurrence modelling software
            *r-spatial: R/GRASS interface for GRASS 6 and other spatial software for R
                R 言語:統計処理。空間統計ライブラリ
            *garmin-utils: similar in scope and function to gpstrans, but works with modern serial Garmins.
See also the NetBSD port
            *e-foto: aerial photogrammetry. See this unanswered post
        Data
        OpenStreetMap
            *OpenStreetmap: Useful for upcoming release of gpsdrive
            *josm: Java Open Street Map Editor
            *gosmore: viewer of OSM XML data such as the planet.osm
        Sample datasets
            * GRASS's Spearfish dataset
            * OSGeo's NC sample dataset
```

[*57] http://gentoo-gis.sourceforge.net/

[*58] http://wiki.debian.org/DebianGis

```
    Java
        *gvSIG (top priority)
        *uDig: User-friendly Desktop Internet GIS
        *GeoServer
        *Postgisdriver-JUMP: currently being worked on. Contrib - depends on jump
        *GeoTools: Java GIS Toolkit
        *Deegree2 and iGeoPortal: currently being worked on. Binary package structure not yet clear.
        *NASA World Wind (Java version): Needs JOGL and perhaps SUN Java
        *JOGL: Java
        *Kosmos Desktop GIS derived from OpenJUMP (an unofficial package available)
            For more info on the Java state in Debian, see the moving java to Debian/main page.
    Live Web Apps
        *PyWPS: Python Web Processing Service
        *OpenLayers, TileCache, FeatureServer (all from MetaCarta)
        *p.mapper (unofficial packages available
    3D Visualization
        *ParaView: Very useful for 3D rapresentation of GRASS rasters and vectors
        *Visual Terrain Project: proposed
        *VisIt: 3D visualization
        *X3D: modern version of VRML, various libs & apps
        *OpenDX: the open source version of IBM's Visualization Data Explorer. Uses the IBM Public License
        *MINI: proposed, needed for VTerrain
        *OpenProducer: proposed, useful for OpenSceneGraph
    Lower Priority
        *Shape file utilities, including ShapeChecker
        *http://www.primagis.fi/ map extension for Plone; it builds on top of Mapserver,
Python Cartographic Library (PCL) and Cartographic Objects for Zope (ZCO)
        *OpenEV: currently being worked on (Alex Bodnaru); old version, based on gtk1; the new one, based on
gtk2, is still unsuitable to packaging
        *MB-System: multibeam, interferometry, and sidescan sonar data processing (GPL)
        *PhpPgGIS: ITP #381974. Project apparently inactive
        *Open3D GIS: Requires FreeWRL, see http://sourceforge.net/projects/freewrl/.
 Project apparently dead? https://sourceforge.net/projects/open3dgis/
        *dxf2svg: also the reciprocal svg2dxf (depends on pstoedit)
        *JGrass: currently being worked on. Contrib - needs jdk1.4, several contrib packages. Currently being
 fused with uDig - better postpone packaging until settled down
```

この他に FOSS4G Toolkit CD や ArcheOS(Mandriva だったが最近 Ubuntu 化した) などの Live CD もあります。[*59]

17.5 空間データ

以前は必要とする者が自ら取得することが一般的であったが、最近では各種空間データが整備されてきている。

17.5.1 国内

- 国土地理院発行の各種数値地図（有償）[*60]
  ただし、近年、情報公開により閲覧を目的としてデータの無償公開が進んでいる。
  ex) 国土地理院数値地図 (空間データ基盤) の閲覧（試験公開）[*61]
- 電子国土ポータル[*62]
  各種地図の閲覧を主眼。精細なベクトルデータを提供。専用プラグインを用いて各自が Web 地図作成を行える。
- 陸域観測技術衛星「だいち」(ALOS)
  宇宙航空研究開発機構（JAXA[*63]）が 2006 年 1 月 24 日に打ち上げた衛星の観測データ。有料？

「高精度で標高抽出を行うためのパンクロマチック立体視センサ (PRISM)、および土地被覆の観測を高精度に行うための高性能可視近赤外放射計 2 型 (AVNIR-2)、昼夜の別なく、また天候によらず陸域の観測が可能なフェーズドアレイ方式 L バンド合成開口レーダ (PALSAR) の 3 つの地球観測センサを搭載」

※ 2008 年 1 月 8 日、予定した精度が取得できないのではないかとの報道[*64] がなされたが、その後の調査により、同年同月 16 日、新たな方法を用いることで懸念されていた問題は改善できるとの発表がなされた[*65]。

[*59] FOSS4G Toolkit CD は Mandriva 向けの RPM セットとして配布するのみとなったようです。
[*60] http://www.gsi.go.jp/
[*61] http://sdf.gsi.go.jp/
[*62] http://portal.cyberjapan.jp/index.html
[*63] http://www.jaxa.jp/index_j.html
[*64] http://www.asahi.com/special/space/TKY200801090162.html
[*65] http://www.jaxa.jp/press/2008/01/20080116_sac_daichi_j.html

17.5.2 地球規模

- スペースシャトル地形データ Shuttle Radar Topography Mission (SRTM)
  1994 年より実験開始。現在主に利用されているのは 2000 年に行われた第 3 回実験によって取得されたデータ。
  11 日間の飛行で両極を除く地上の陸地の約 80 %、全人口密集地の約 95 %をカバーするデータを取得した。
  NASA の FTP サイト[*66]よりデータをダウンロード可能。一部有料。
  DEM(Digital Elevation Model) の作成用
- ランドサット衛星画像 Landsat Imagery (TM, ETM+)[*67]
  メリーランド大学[*68] からデータをダウンロード可能。様々な波長の光の反射データ。リモートセンシング用。
- 商用衛星画像データ
  Google Earth や Google Map などでは高精細な商用衛星画像データが用いられている。Earthsat 社や Digital Globe 社、Bluesky 社など。
  確かに解像度が高く利用価値は高いが、非常に高価なので、費用対効果を考えて利用されている。

でも結局研究目的の場合などは、今でも自分達でデータを取得しなければならないことの方が多いです。なので、いかに効率的に、必要とされ、正確・詳細な空間データを取得するかが問題となります。

17.6 おわりに

17.6.1 まとめ

以前に比べて環境は格段に良くなっています。特に、こちらから能動的に各種データの整備を要求・もしくは自ら多大なコストをかけて取得しなくても、次々と空間基盤データの整備が行われてきています。表示系の地図サイトなどは個人レベルで管理・運用できる環境が整っています。
皆さんもちょっと触ってみませんか？

17.6.2 反省点

今回は GIS の概要を説明することで大半を消費してしまいました。より具体的な話が出来なかったのが悔やまれます。しかし、実際に使ってみないと実感が湧かないのも事実です。また、具体的な課題がないと、なかなか触るきっかけもないかもしれません。

[*66] ftp://e0srp01u.ecs.nasa.gov/srtm/
[*67] http://www.eorc.jaxa.jp/hatoyama/satellite/satdata/landsat_j.html
[*68] http://glcfapp.umiacs.umd.edu:8080/esdi/index.jsp

18 Debian で PC クラスタを作ってみよう

中尾 昌広

18.1 はじめに

安価で高速な並列計算機システムである PC クラスタを、Debian を使ってセットアップする方法について説明します。

18.1.1 PC クラスタとは何か？

クラスタ (cluster) とは英語で「ブドウの房、同じものが群らがっている様」みたいな意味です。つまり、PC クラスタとは、複数台の PC をブドウのように（LAN ケーブルで）相互接続し、それらを協調動作させるシステムのことです。

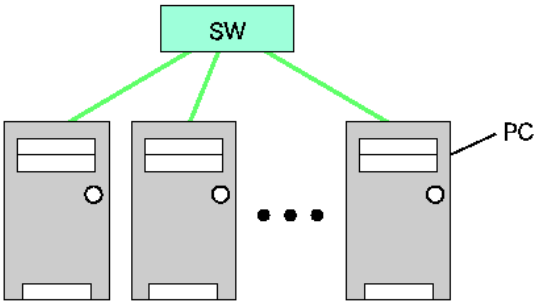


図 6　クラスタの概念図

18.1.2 PC クラスタをつくる理由

高速な計算機環境が欲しいけど、スーパーコンピュータみたいな高価な専用並列計算機はとても買えません。そこで、普通の電気屋で売っているような PC をクラスタ化して、安価に並列計算環境を作ろう、ということが考えられました。

PC クラスタには大きく分けて、HA（High Availability）クラスタと HPC（High Performance Computing）クラスタがあります。HA クラスタとは、高可用性を持つシステム、つまりサービスを止まりにくくすることを目的としたクラスタシステムです。最近は Web 系のサーバでこのシステムがよく利用されています。HPC クラスタとは、科学計算分野などでおいて、ものすごく時間のかかる処理を、複数台の PC に分散させて、高速に結果を得ることを目的としたクラスタシステムです。企業のメーカでも、車体や航空機を設計する際のシミュレーションなどに用いられています。

本資料とプレゼンでは基本的に HPC クラスタについての話ですが、HA クラスタに共通している点も多いはず

です。

18.1.3 Debian で PC クラスタを作る理由

- Debian には数多くの PC クラスタ用ツールがある。
- PC クラスタの OS として普通の Debian を利用するので、PC クラスタツール以外の有用なツールも数多く利用できる。
- コミュニティが活発（debian-users で質問に答えてくれる人達は素晴らしい）。
- apt が楽。

18.2 PC クラスタの作り方

並列計算ができるまでを目標として、PC クラスタシステムをセットアップする手順を説明します。

1. PC クラスタを構成する部品（PC、LAN ケーブル、スイッチングハブ）を用意し、それぞれ接続する。
2. 用意した PC に Debian GNU/Linux をインストールする。
3. 自分が必要とするコンパイラと並列計算ライブラリをインストールする。並列計算ライブラリは mpich と PVM がよく用いられる。今回は mpich を使う。

   ```
   # aptitude install mpich-bin libmpich1.0-dev gcc g77 g++
   ```

   インストール後に並列計算に用いるホスト名（もしくは IP アドレス）を/etc/mpich/machines.LINUX 記述する。
4. クラスタ内ネットワークでは、速度重視のため暗号化なしで通信を行いたいので、rsh-server と rsh-client をインストールする。

   ```
   # aptitude install rsh-server rsh-client
   ```

   インストール後に rsh を許可するホスト名（もしくは IP アドレス）を/etc/hosts.equiv に記述する。
5. NFS を使って/home 以下の共有を行うと便利なため、PC クラスタを構成する PC の 1 台に nfs-kernel-server をインストールする。

   ```
   # aptitude install nfs-kernel-server
   ```

   そして、nfs-kernel-server をインストールした PC の/etc/exports に、どの PC に NFS のサービスを許可するかの情報を記述する。

   ```
   【/etc/exports の例】
   /home   192.168.1.0/255.255.255.0(rw,async,no_subtree_check)
   ```

   /etc/exports の変更後に NFS のサービスを再起動させる。

   ```
   # /etc/init.d/nfs-kernel-server restart
   ```

   そして、nfs-kernel-server をインストールした PC 以外は、mount コマンドを用いて NFS サーバにマウントする。

   ```
   # mount -t nfs （NFS サーバ）:/home /home
   ```

   以上で、PC クラスタの完成です。簡単です。意外かもしれませんが、並列計算ライブラリ以外は特殊なソフトウェアを用いていません。このように既存のソフトウェアを有効活用できる所が、PC クラスタの大きな特徴です。
6. 並列計算用サンプルファイルが、/usr/share/doc/libmpich1.0-dev/examples/cpi.c にあるので、実行してみましょう。

```
$ mpicc cpi.c -o pi 【コンパイル】
$ mpirun -np 5 pi 【実行】
```

cpi.c は π の値を計算するプログラムです。mpicc というコマンドで C 言語で書かれた並列計算プログラムをコンパイルします。Fortran の場合は mpif77、C++ の場合は mpicxx などを用います。
そして、mpirun というコマンドで、並列計算プログラムを実行します。引数の-np 以降はプロセス数を指定しています。また、並列計算に使用したいマシンを指定する際は、-machinefile というオプションを利用します。

```
$ mpirun -np 5 -machinefile hoge.txt pi
```

この場合、hoge.txt に並列計算に使用したいホスト名を書くと、その PC で並列計算されます。

18.3 クラスタで使うと便利なソフトウェア

Ganglia という各ノードの負荷、生死を Web ブラウザから確認できるソフトウェアです。日、週、月、年といった単位でクラスタ全体のロードアベレージの傾向も見ることができます。

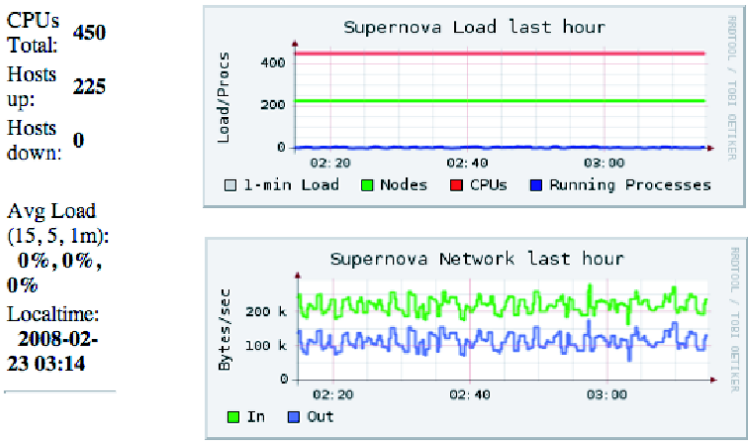


図 7　Ganglia

18.4 おわりに

駆け足でクラスタの概要とセットアップ手順について説明しました。Debian でクラスタを作るのは簡単だというのが伝われば幸いです。

18.5 参考文献

- 同志社大学知的システムデザイン研究室（http://mikilab.doshisha.ac.jp/）
- 超並列計算研究会（http://www.is.doshisha.ac.jp/SMPP/）

19 資料作成基礎 (TeX)

山下 尊也

19.1 インストール

今回、対象とするものは 2008 年 2 月 21 日現在の stable である etch を対象とします。私のセッションでは最小限の事しか述べないため、YaTeX などを活用したい方は、関連 URL や東京エリア Debian 勉強会 2007 年 12 月の資料[*69]をご覧下さい。

19.1.1 /etc/apt/sources.list の確認

non-free に含まれているものも利用するため、/etc/apt/sources.list は以下のようにしておいて下さい。

```
deb http://ftp.debian.or.jp/debian/ etch main contrib non-free
```

19.1.2 teTeX と pTeX 一式のインストール

```
$ sudo aptitude install ptex-bin
```

19.1.3 奥村さんの新クラスファイルのインストール

```
$ sudo aptitude install okumura-clsfiles
```

19.1.4 dvipdfmx のインストール

```
$ sudo aptitude install dvipdfmx
```

19.1.5 Adobe Reader および CMAP のインストール

Etch の Evince には日本語のフォントを埋め込んでない場合に文字が化けると言うバグ[*70] があるので、PDF を見るために Adobe Reader をインストールします。

Adobe Reader の配布ページ[*71] にアクセスし、Adobe Reader の deb パッケージを取ってきます。

[*69] whizzytex などを用いてプレビューさせるなど色々と技が載っています。

[*70] いくやさんが修正したものを配布しています。http://ikuya.info/wiki/index.php?etchpackages ただ、2008 年 2 月 22 日現在の unstable でも、Evince には、poppler の問題があり、http://lists.debian.or.jp/debian-devel/200712/msg00000.html これは、Adobe との CMAP 問題に繋がるので、Adobe Reader で見る事をお勧めします。

[*71] http://www.adobe.com/jp/products/acrobat/readstep2.html

```
$ sudo dpkg -i AdobeReader_jpn-8.1.2-1.i386.deb
$ sudo aptitude install cmap-adobe-japan1 cmap-adobe-japan2
```

19.1.6 YaTeX(野鳥) のインストール

エディタとして Emacs を使っている人は、YaTeX と呼ばれる LaTeX 入力支援環境があるので、それを利用すれば良いでしょう。[*72]

```
$ sudo aptitude install yatex emacs21
```

文字コードは iso-2022-jp で統一しています[*73]。たとえば、emacs + yatex を使用している場合で iso-2022-jp をデフォルトにするには、下記のような設定を.emacs にかけばよいでしょう。

```
(add-hook 'yatex-mode-hook
          '(lambda ()
             (progn
               (if (string-match "^/home/user/monthly-report/" default-directory)
                   (progn (set-buffer-file-coding-system 'iso-2022-jp)
                          (set-buffer-modified-p nil))))))
```

.emacs の例です。

```
(setq auto-mode-alist
      (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)
(defvar YaTeX-dvi2-command-ext-alist
  '(("xdvi" . ".dvi")
    ("ghostview\\|gv" . ".ps")
    ("acroread" . ".pdf")))
 (setq dvi2-command "acroread")
(setq dviprint-command-format "dvipdfmx `basename %s pdf`dvi")
;;; 色付け
(setq YaTeX-use-font-lock t)
(add-hook 'yatex-mode-hook
          '(lambda ()
             (progn
               (if (string-match "^/home/user/monthly-report/" default-directory)
                   (progn (set-buffer-file-coding-system 'iso-2022-jp)
                          (set-buffer-modified-p nil))))))
```

19.2 文章の編集

LaTeX は最初は難しいと感じるかもしれませんが、基本は HTML と似たようなものなので、まずはソースを読む事から始めてみると良いかもしれません。LaTeX についてさらに知りたい方は、「[改訂第 4 版] LaTeX2 美文書作成入門 (大型本)　奥村 晴彦 (著) 」などの本を読む事をお勧めします。

19.2.1 リポジトリからデータを取ってくる

現在、関西 Debian 勉強会では、リポジトリの用意が出来ていないため、東京エリア Debian 勉強会のリポジトリを借りている状態です。

```
$ sudo aptitude install git-core
$ git clone git://git.debian.org/git/tokyodebian/monthly-report.git
```

ドキュメントは pLaTeX で作成しています。ファイル名として下記になっています。(YYYY)(MM) は、年と月で、例えば 2008 年 02 月であれば 200802 です。

- debianmeetingresume(YYYY)(MM)-kansai.tex
  事前配布資料
- debianmeetingresume(YYYY)(MM)-kansai-presentation.tex

[*72] Emacs を使っていない方でも、VIM-LaTeX(http://vim-latex.sourceforge.net/) や KDE 環境に Kile(http://kile.sourceforge.net/) という LaTeX を編集する環境があるようです。

[*73] Windows 版と Linux 版の ptex で共通して扱える文字コードにしたという経緯があります。ただし現状 Windows で全部できる状況ではありません。

プレゼンテーション用 (prosper を利用)

- image(YYYY)(MM)
  画像ファイルなどの置き場

19.2.2 基本部分の説明

スタイルファイルは kansaimonthlyreport.sty パッケージを利用します。このスタイルファイルは、東京エリア Debian 勉強会のスタイルファイルがカラーなので、モノクロでも見やすいように編集しました。今後、いろいろと手を加えていくと思います。

```
\usepackage{kansaimonthlyreport}
```

各担当部分は section として扱います。特別なコマンド dancersection で指定します。形式は dancersecion{ **タイトル** }{ **作者名** } です。その中で subsection や subsubsection を利用して文書を構成してください。

```
\dancersection{Debian 勉強会資料の準備の方法}{上川 純一}
\label{sec:debmtg2007howtoprepare}
```

各担当部分は section として扱います。特別なコマンド dancersection で指定します。形式は dancersecion{ **タイトル** }{ **作者名** } です。その中で subsection や subsubsection を利用して文書を構成してください。

```
\dancersection{Debian 勉強会資料の準備の方法}{上川 純一}
\label{sec:debmtg2007howtoprepare}
```

19.2.3 画像ファイルの処理

画面写真の画像を追加するときは、できるだけサイズの小さい png などを利用してください。グラフなどの線画であれば、eps でかまいません。png であれば、ebb コマンドを利用して bounding box を作成してください。

```
ebb XXX.png
```

そして次のようにして文章に埋め込みます。

```
\begin{figure}[!htbp]
\begin{center}
 \includegraphics[width=120mm]{image200802/latex.png}
 \caption{\LaTeX で変換するイメージ}
 \label{fig:latex}
\end{center}
\end{figure}
```

19.3 PDF への変換

変換の過程を簡単な図で示すと、図 8 のようになります。

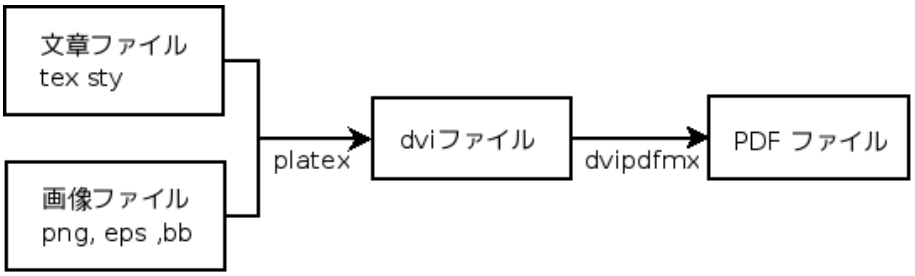


図 8 LaTeX で変換するイメージ

コマンドで一つずつ変換をするならば、

```
$ platex debianmeetingresume200802-kansai.tex
$ dvipdfmx debianmeetingresume200802-kansai.dvi
```

ですが、リポジトリから入手したものであれば、

```
$ make
```

make コマンドだけでファイルの更新があったファイルの変換などを行なってくれます。

20 バグレポートから参加する Debian パッケージ開発

木下 達也

20.1 Debian 開発への参加

Debian は、どんな用途に使ってもよい、ソースを変更できる、そのままでも変更版でもコピーして配布できる、自由 (フリー) なソフトウェアであり続ける、ユーザーの、フリーソフトウェアコミュニティのためのオペレーティングシステムです。

Debian Project は有志によって構成されています。皆さんのできる範囲での参加・手助けを歓迎します。まずはバグレポートから……

- Debian プロジェクトホームページ
  `http://www.debian.org/`
- Debian 社会契約 / Debian フリーソフトウェアガイドライン (DFSG)
  `http://www.debian.org/social_contract`
- How You Can Join
  `http://www.debian.org/devel/join/`

20.2 Debian パッケージを使っていて問題に遭遇!

バグ? 設定誤り? その他、環境依存の問題など?
さて、どうする?

20.3 問題解決へ

- 現状は? 望ましい姿は? 再現できる?
- 問題の切り分け、仮説、実行、検証
  - 設定をデフォルト値 (または最小の設定) にしてみると?
  - その他、条件を変えてみると?
- ソースを見てみる?

```
# apt-get update
(/etc/apt/sources.list に deb-src 行が必要)
$ apt-get source パッケージ名
```

20.4 パッケージ情報の調べ方

- ファイルがどのパッケージに含まれているか検索
  - インストール済のパッケージから検索

```
$ dpkg -S /bin/ls
$ dpkg -S bin/apt
```

  - Debian Packages ページで「パッケージの内容」を検索

    `http://packages.debian.org/`

  - apt-file

```
# apt-file update
$ apt-file -F search bin/ls
  （キーワードを末尾に持つパス）
$ apt-file search bin/apt
  （キーワードを含む名前のファイルを含むパッケージ）
```

- Debian Bug Tracking System (BTS)

  バグレポートを記録・追跡

  `http://bugs.debian.org/`

  `http://bugs.debian.org/`**パッケージ名**

  `http://bugs.debian.org/src:`**ソースパッケージ名**

20.5 メーリングリスト

- Debian Mailing Lists

  `http://lists.debian.org/`

  (バグレポートやアップデート情報はメーリングリストにも流れています)
- 日本語で相談するには、Debian JP Project のメーリングリスト

  `http://www.debian.or.jp/community/ml/`

  `http://www.debian.or.jp/community/ml/openml.html`
- 検索
  - Google グループ: `http://groups.google.com/`
  - Gmane Search: `http://search.gmane.org/` (日本語検索は難有り)

20.6 バグレポートの書き方

- メールクライアントを起動
- メール作成
  - Text/Plain で
  - 添付や gpg 署名は可
- 宛先、題目を記入

```
To: submit@bugs.debian.org
Subject: yaskkserv: funny dictionary order when skkdic-extra is installed
```

- 本文の 1 行目からパッケージ名、バージョン、重要度を記入

```
Package: yaskkserv
Version: 0.3.8-1
Severity: wishlist
```

- Severity(重要度)
  - critical (致命的) 他のパッケージやシステムに影響する重大なバグ
  - grave (重大) そのパッケージ自体の重大なバグ
  - serious (深刻) ポリシー違反など、リリース品質でないと判断されるバグ
  - important (重要)
  - normal (通常)
  - minor (軽度)
  - wishlist (要望)
- 1 行空けて、本文を記入、送信
  (例: `http://bugs.debian.org/464812`)

```
Date: Sat, 09 Feb 2008 12:45:58 +0900 (JST)
From: Tatsuya Kinoshita <tats@vega.ocn.ne.jp>
To: submit@bugs.debian.org
Subject: yaskkserv: funny dictionary order when skkdic-extra is installed

Package: yaskkserv
Version: 0.3.8-1
Severity: wishlist

When the skkdic-extra package is installed, SKK-JISYO.JIS* in
skkdic-extra seem to be prefered over SKK-JISYO.L* in skkdic.
(e.g. when converting "あし", the default setting of yaskkserv
prefers "趾", "蹇", "躄", ... over "足")

Please prefer SKK-JISYO.L* over SKK-JISYO.JIS* by default.

Thanks,
--
Tatsuya Kinoshita
```

- Emacs ユーザーなら 'M-x debian-bug RET'
  - 要 debian-el パッケージ
  - Mew や Wanderlust をデフォルトの Emacs メールクライアントに設定
    (/usr/share/doc/パッケージ名/README.Debian を参照)

20.7 Debian 開発者との協同

- Debian Package Tracking System (PTS)
  ソースパッケージ毎のまとまった情報ページ
  `http://packages.qa.debian.org/` ソースパッケージ名
  (パッケージ名でも OK。ソースパッケージのページへ転送されます)
- PTS subscribe, unsubscribe
  - バグレポート、アップロード情報などを購読できる
  - どの情報を購読するかは、メールで pts@qa.debian.org へコマンドを送って変更
    `http://www.debian.org/doc/manuals/developers-reference/ch-resources.en.html`
- メールアドレス
  - バグレポートへの返信: バグ番号@bugs.debian.org
  - バグレポートの制御: control@bugs.debian.org
  - パッケージメンテナ宛: パッケージ名@packages.debian.org
  - Debian 開発者宛、セキュリティ情報、非公開: security@debian.org
  - Debian セキュリティチーム宛、非公開: team@security.debian.org

21 GPG 最初の一歩

倉敷 悟

GnuPG というと、「ああ、なんか apt-get の時にゴチャゴチャ文句言ってくるアレか」といった印象をお持ちの方もおられるのではないかと思います。実は Debian には様々な形で GPG が組みこまれていて、ssh 等にならぶマストアイテムといっても過言ではありません。このセッションでは、GPG の用途と役割について簡単にご紹介した後、典型的な使用方法についてデモを行います。

21.1 GPG の概要

端的に GPG とは何か、というと：

- PGP 規格のフリーな実装 (特許の観点で、あるいは処理系として)
  - 公開鍵暗号と共通鍵暗号のハイブリッド方式
- メールやファイルの暗号化をするソフト
  - 暗号化で盗聴を防ぐ (機密性/Confidentiality)
  - 署名で改竄を防ぐ (完全性/Integrity)
  - 信頼の輪で本人同定 (認証/Authentication)

より詳しい解説は、下記 URL で読むことができます。

- The GNU Privacy Guard `http://gnupg.org/`
  GnuPG 本家。ドキュメントは豊富ですが全部英語です。
- GNU Privacy Guard 講座 `http://hp.vector.co.jp/authors/VA019487/`
  日本語のポータル的なサイト。使い方を調べたい時に最適かと思います。
- GnuPG (The GNU Privacy Guard) `http://www.math.s.chiba-u.ac.jp/~matsu/gpg/index.html`
  少し古い記事ですが、暗号機能の一般的な背景がわかりやすく解説されています。

21.2 Debian における GPG

21.2.1 パッケージアーカイブ

etch 以降、配布しているパッケージアーカイブの改竄防止のため、Release ファイルに GPG でデジタル署名を施しています。APT でパッケージを取得する際に、この署名を確認して、検証できない場合は警告が表示されるようになっています。

公式アーカイブの公開鍵はパッケージとして配布されているので、これをインストールしておけば、後は APT がよしなに面倒をみてくれます。

```
kura@acacia:~$ dpkg -l | grep keyring
ii  debian-archive-keyring              2007.07.31                GnuPG archive keys of the Debian archive
ii  debian-multimedia-keyring           2007.02.14                GnuPG archive key of the debian-multimedia r
```

上記の例では debian-multimedia の鍵も入っていますが、公式でないアーカイブでも、鍵が配布されていれば同じように検証をしてくれます。

- SecureApt `http://wiki.debian.org/SecureApt`

21.2.2 開発者の認証

Debian では、GnuPG が本人の身分証明手段として広く浸透しており、特にパッケージまわりで作業しようと思うと、なんだかんだで必要になります。

私に見えてる範囲では：

- keysign に必要 (本末転倒ですが……)
- パッケージのビルドに必要 (必須ではない)
- mentors.debian.net への登録に必要
- Debian JP への参加に必要
- Debian Maintainer—Developer への応募に必要

といった場面で自分の GPG 鍵が必要です。また、ただ作るだけではなくて、Debian Developer と keysign を行っておく必要もあります。

21.3 名刺としての GPG 鍵

現時点では、「Debian は使うだけだし、面倒だから別に要らないや」と思うかも知れません。

ただ、他にもメリットがありますし、これを機会に是非 GnuPG を使ってみてください。最初に必要なのは、たった一つ、「これから作る鍵をちゃんと維持するぞ」という意気込みだけです。

- デジタルな identity として
- こういったコミュニティでの名刺交換に
- 公開鍵基盤の身近な活用例として
- Debian Developer はレア
- keysign は結構おもしろい

21.4 はじめてみよう

21.4.1 GPG 鍵を作る

というわけで、自分の鍵を作ってみましょう。

```
kura@acacia:~$ gpg --gen-key
gpg (GnuPG) 1.4.6; Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: directory ‘/home/kura/.gnupg’ created
gpg: can’t open ‘/gnupg/options.skel’: そのようなファイルやディレクトリはありません
gpg: keyring ‘/home/kura/.gnupg/secring.gpg’ created
gpg: keyring ‘/home/kura/.gnupg/pubring.gpg’ created
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?
```

まず、鍵の種類を聞かれます。ここはデフォルトのままで構いませんので、何も入力せずに enter を押します。

```
DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

次に、鍵の流さを聞かれます。選択の目安としては、1024 は短かすぎるので NG、デフォルトの 2048 はまぁ OK、計算機資源が余っているなら 4096 もアリ、といったところでしょうか。

この例では、そのまま enter を押して、デフォルトの 2048 としています。

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y
```

次は、鍵の有効期限を選びます。私の場合、最初ここをどうするべきか困ったところですが、これまで keysign してもらった方々のを見る限り、期限なしで問題ないようです。

というわけで、そのまま enter を押して、確認にも y と答えます。

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: KURASHIKI Satoru
Email address: lurdan@gmail.com
Comment:
```

ここでは、自分の個人情報を設定します。keysign などでは、公的な証明書とのつきあわせを行いますので、その時に確認できる本名を入力する必要があります。このあたり、日本のネットワーク文化とはいまひとつ馴染みがよくないかも知れませんね。

GPG 鍵では、一つの ID に対して複数のメールアドレスを設定することができるので、Comment でそれを明示することができます。

```
You selected this USER-ID:
    "KURASHIKI Satoru <lurdan@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

最終的に、ユーザ ID がこうなりますよ、という確認です。問題ないので O と入力します。

```
You need a Passphrase to protect your secret key.

Enter passphrase:
Repeat passphrase:
```

ここでパスフレーズを確認含めて 2 回入力します。パスフレーズには流さの制限がないので、覚えやすい長さの英文を使うといいのではないでしょうか。

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.++++++++++++++..++++++++++.+++++++++++++++++++++++++++++++++++++++.+++++++++++++++.+++++++++++++++.+++++.++
++++++++++++++++++.++++++++++>.++++++++++.............+++++

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 230 more bytes)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
++++++++++.....+++++..+++++.+++++.+++++.+++++..++++++++++.+++++.++++++++++++++..+++++.++++++++++....+++++.++
++++++++++++.++++++++++.++++++++++.+++++..++++++++++++++..+++++>.++++++++++...>.+++++.....................
.....................................................................................................
..............+++++^^^^^
```

パスフレーズを入力した後は、乱数の生成のために、ブラウザでも開いて適当に別の作業をします。そろそろいいかな、と元の画面に戻ると……

```
gpg: /home/kura/.gnupg/trustdb.gpg: trustdb created
gpg: key 9751B54D marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   1024D/9751B54D 2008-03-21
      Key fingerprint = 3AAA 2ED1 ACDA 49AE 4C68  66B9 CE7D E89C 9751 B54D
uid                  KURASHIKI Satoru <lurdan@gmail.com>
sub   2048g/B79D76D0 2008-03-21
```

このように完成しています。最後に表示された「key fingerprint」のうち、最後の 2 ブロックが自分の鍵の ID となります。

最後に表示されている部分で、「pub」の行が公開鍵、「sub」の行が秘密鍵を表しています。

```
kura@acacia:~$ ls -l .gnupg
合計 20
-rw------- 1 kura kura 1167 2008-03-21 16:06 pubring.gpg
-rw------- 1 kura kura 1167 2008-03-21 16:06 pubring.gpg~
-rw------- 1 kura kura  600 2008-03-21 16:06 random_seed
-rw------- 1 kura kura 1316 2008-03-21 16:06 secring.gpg
-rw------- 1 kura kura 1280 2008-03-21 16:06 trustdb.gpg
```

生成された鍵は、このように $HOME/.gnupg/ に保管されています。これらのファイルは超大事なので、大切に取り扱うようにしましょう。

ちなみに、上の例で使っている鍵は、この記事のために作ったもので、もはや存在しません。

21.4.2 基本的な操作

通常は、自分の鍵束 ($HOME/.gnupg/*) には、

- 自分の秘密鍵
- 自分の公開鍵
- 他の人の公開鍵

が入っています。これらの情報を確認する基本的なコマンドをご紹介します。なお、鍵束の中で特定の鍵を指定するには、次のうちどれかが使えます。

- 鍵 ID (ex. 0x9751B54D)
- 鍵 ID の 0x を省略 (ex. 9751B54D)
- 自分の UID の一部 (ex. KURA)

自分の鍵束にある公開鍵を一覧表示：

```
$ gpg --list-keys
```

そのうち特定の公開鍵を表示：

```
$ gpg --list-keys KURA
```

鍵指紋 (fingerprint) を確認：

```
$ gpg --fingerprint KURA
```

鍵署名を確認：

```
$ gpg --list-sigs KURA
```

21.4.3 メールで使ってみる

私が GPG を使うのは、主に gmail 上のメールアドレスなので、firefox のプラグイン「firegpg」を使っています。対象となる本文を選択して増えたボタンを押すだけなので簡単に使うことができます。

21.5 keysign の流れ

21.5.1 準備

keysign のためには、次のものが必要です。

- 公的な証明書 (鍵の持ち主の個人情報を証明できる、写真つきの書類)
  - 運転免許
  - 住民基本台帳カード
  - パスポート
  - 学生証
- 署名してもらう鍵の指紋 (fingerprint) を印刷したもの

私の場合、鍵の指紋は手で書いたり、鍵表示をそのままコピー&ペーストして印刷したりしていました。

```
pub   1024D/9751B54D 2008-03-21
      Key fingerprint = 3AAA 2ED1 ACDA 49AE 4C68  66B9 CE7D E89C 9751 B54D
uid                  KURASHIKI Satoru <lurdan@gmail.com>
sub   2048g/B79D76D0 2008-03-21
```

21.5.2 当日

公的な証明書を使って、keysign の相手とお互いに相手が相手であることを確認し、用意した鍵指紋 (fingerprint) の印刷を交換します。

これだけです。簡単ですね。

21.5.3 その後

端末の前に戻ったら、まずは相手の公開鍵を手に入れましょう。

```
$ gpg --keyserver subkeys.pgp.net --recv-keys 0x????????
```

keyserver は、相手が Debian Developer であれば keyring.debian.org も使えます。また、? の部分には、もらった鍵指紋 (fingerprint) の最後の 8 文字を入力します。

無事に公開鍵が入手できたら、それに署名をします。

```
$ gpg --edit-key 0x????????
```

gpg のコマンドモードに入るので、「uid (数字)」で署名をする uid を選び (*マークがつきます)、「sign」と入力します (fingerprint とメールアドレスを確認して署名)。

なお、相手が複数の uid を持っている場合、私は一通り全てに署名することにしています。

終わったら、「quit」と入力し、変更を保存してください。

次のように署名の結果を表示して、自分の名前が表示されていることを確認してください。

```
$ gpg --list-sigs 0x????????
```

ここまでで問題がなければ、相手の公開鍵をテキストファイルに export します。

```
$ gpg --export -a  0x???????? > someones.key
```

後は、これを相手に送るだけです。添付するなり、本文に貼るなりして、メールで送信します。相手の公開鍵で暗

号化しておくとよいでしょう。

さて、相手も同じように署名をしてくれているはずですので、しばらくするとメールで自分の公開鍵を送ってもらえるはずです。

届いたら、テキストファイルに export された鍵を自分の鍵束にとりこみ、ついでなので公開鍵サーバにも送っておきます。

```
$ gpg --import yours.key
$ gpg --keyserver subkeys.debian.org --send-keys 0x????????
```

? には、自分の公開鍵 ID を入力します。

- 鍵署名 (Keysigning) `http://www.debian.org/events/keysigning`

22 Debian 開発者のコーナーの歩き方

大浦 真

Debian プロジェクトの Web ページ内の「Debian 開発者のコーナー」(`http://www.debian.org/devel/`) には、Debian の開発や改良を行っていくために必要な情報が集まっています。ただ、いろいろあり過ぎて、どこに何があるか分かりにくいです。以下に有用なものをいくつか紹介します。

22.1 基本

プロジェクトに関する基本的な情報です。

- Debian の組織構成
- Debian に参加する
- 開発者データベース — Debian 開発者の情報やプロジェクトのマシンを検索できます。
- 投票情報 — プロジェクトリーダの選挙結果などを参照できます。
- さまざまなアーキテクチャ — Debian は i386 以外にも多くのアーキテクチャに移植されています。Linux 以外への移植もあります。

22.2 パッケージ開発

パッケージを作成、メンテナンスするために必要な情報です。メンテナが読むべき文書が集まっています。

- Debian ポリシーマニュアル — Debian のシステムが守るべき技術的事項。いくつかサブポリシーもあります。
- デベロッパーズリファレンス — Debian のメンテナが守るべき手順と利用できるリソースの情報です。
- 新規メンテナのためのガイド

22.3 進行中の仕事

パッケージのメンテナンスを進めていくために必要な情報です。

- バグ追跡システム
- パッケージ追跡システム
- Lintian レポート — パッケージがポリシーを満たしているかどうかチェックすることができます。

22.4 プロジェクト

Debian 内のプロジェクトに関するリンクです。

- 品質保証 (Quality Assurance) グループ — Debian 全体の品質を向上させるためのプロジェクト
- 自動構築ネットワーク — ソースパッケージを各アーキテクチャ用に自動的に構築します。
- いろいろなカスタム Debian ディストリビューション

22.5 その他

その他の情報。

- Debian の ロゴとバナー

23 Windows な PC でも Debian を楽しもう

名村 知弘

23.1 はじめに

Windows 上でしか動かないソフトがあったり、会社の PC が Windows だったりと、Windows を完全に手放すことができない環境はまだまだ多いと思います。そんな Windows な環境にあっても、coLinux があれば Windows を起動しながらにして、お気軽に Debian を楽しむことができます。

coLinux の設定から、Debian を楽しむまでの流れをご紹介します。

23.1.1 coLinux とは何か？

coLinux(COoperative LINUX) とは、Windows 上で動作する Linux カーネルです。coLinux 上で Debian を動作させる場合、coLinux パッチの当てられた専用の Linux カーネル上で Debian が動作することになります。Debian が Windows 上の 1 つのアプリケーションとして動作するため、Windows を使いながら必要になった時に Debian を起動し使用することができます。

23.1.2 なぜ coLinux か？

Windows 上で Debian を使う環境としては、VMWare などの PC エミュレータが考えられますが、PC エミュレータはオーバヘッドやメモリ使用量が大きく、全体的に動作が重くなってしまします。coLinux は Windows アプリケーションとして Linux が起動するため、PC エミュレータに比べ動作が軽いのが特徴となっています。ただし、ディスプレイデバイスやサウンドデバイスが実装されていないため、単体では音のでない CUI 環境となります。[*74]

23.2 coLinux インストール

coLinux で Debian を使用するには、

1. coLinux のインストール
2. network の設定
3. coLinux の設定

の 3 ステップを実行します。

以下では、Windows Vista[*75]での、TAP-Win32 を使用したインストールのステップを簡単に説明します。

[*74] 別途 VNC や X サーバ、サウンドサーバを使用することで、これら機能を使用することができます。
[*75] WindowsXP でも同様の手順で作業できると思います。

23.2.1 coLinux のインストール

coLinux の配布サイト[*76]から、coLinux のインストーラ[*77]をダウンロードします。

ダウンロードしたインストーラを起動し、デフォルトのままインストールを進めていくと、図 9 のような、インストールモジュールを選択する画面が表示されます。ここではネットワークアダプタとして「TAP-Win32」を使用しますので、「SLiRP」「WinPcap」のチェックを外してください。

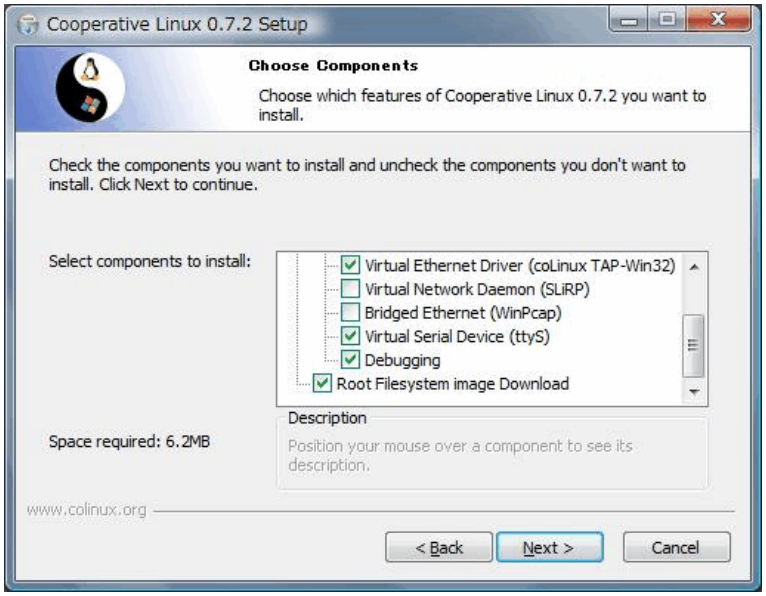


図 9　インストールコンポーネントの選択

引き続きデフォルトのままインストールを進めていくと、図 10 のような root ファイルシステムを選択する画面が表示されるので、間違えずに「Debian」を選択してください。

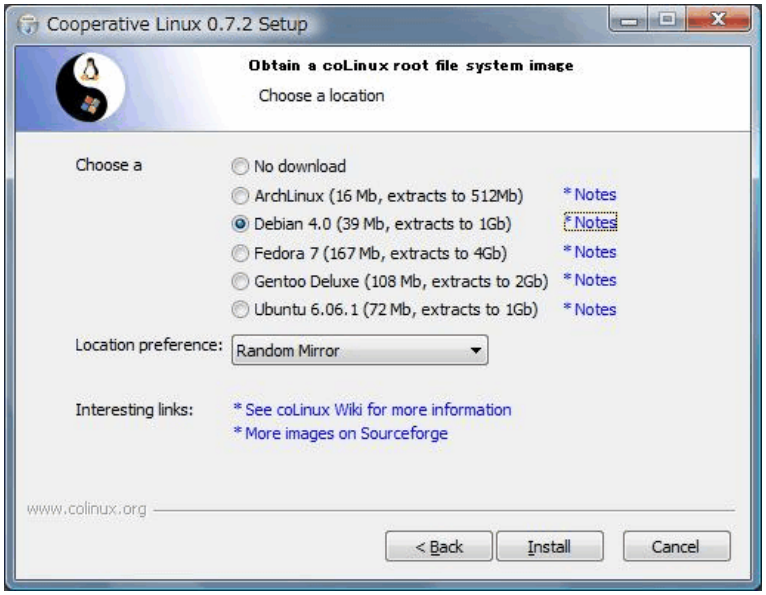


図 10　ルートファイルシステム選択

[*76] http://sourceforge.net/projects/colinux/files

[*77] 執筆時では kernel2.6.22 をベースとした coLinux0.7.2 が最新安定版となっています

あとは再びデフォルトのままインストールを進めると、インストール完了です。ここで、「コントロールパネル」-「ネットワーク接続」を表示し、「ローカルエリア接続 2」が作成されていると思いますので、右クリックし名前の変更を行い、「TAP」に変更しておきます。

「ローカル エリア接続」 が 2 つ以上ある場合はプロパティを表示し、「接続の方法」が「TAP-Win32 Adapter V8 (coLinux)」となっているものの名前を変更します。

23.2.2 network 設定

coLinux の使用するネットワーク接続は、TAP-Win32(仮想ネットワークアダプタ) を使用します。この仮想ネットワークアダプタは物理的に LAN などのネットワークに接続していないため、ホスト PC の Real NIC を通じて物理的なネットワークに接続する必要があります。物理的なネットワークに接続する方法として、ここでは以下の 2 つの方法をご紹介します。

Bridge 接続を用いた方法

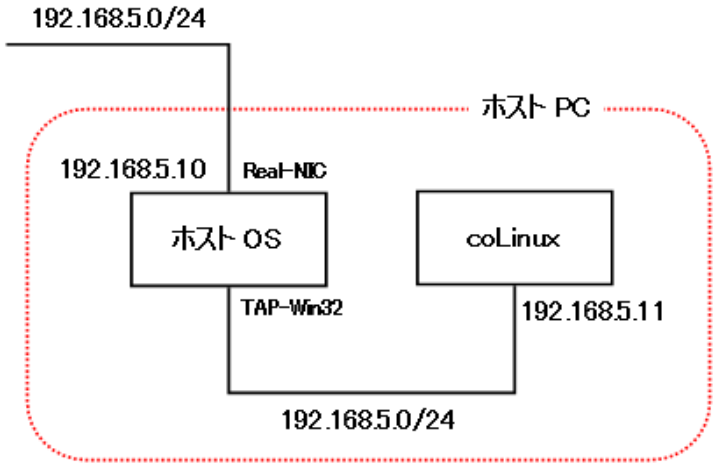


図 11　Bridge 接続を使用したネットワーク構成

Bridge 接続では図 11 のように、TAP-Win32 を通じて coLinux が Real NIC と同じサブネットに接続されるため、他の PC からも coLinux にアクセスすることができます。Real NIC のネットワーク環境 (固定 IP や DHCP など) に合わせて、coLinux のネットワーク設定を行う必要があります。長所としては、ネットワーク上の他の PC から coLinux に直接アクセスすることができます。短所としては、ネットワークに接続されていない環境では、Windows からも coLinux にアクセスすることができません。

設定は以下のようになります。

1. 「コントロールパネル」-「ネットワーク接続」を開きます。
2. 「ローカル エリア接続」と「TAP」を選択し、右クリックから「ブリッジ接続」を選択します。
3. 「ネットワークブリッジ」という接続設定が表示されたら完了です。

ネットワーク接続の共有を用いた方法

ネットワーク接続の共有では図 12 のように、TAP-Win32 は Real NIC とは異なるサブネットに参加し、ホスト OS が NAT になることで中継します。TAP-Win32 の IP アドレスは「192.168.0.1」固定となり、coLinux は「192.168.0.1/24」の IP を固定で割り当てることになります。Real NIC の接続しているサブネットが「192.168.0.0/24」の場合、IP アドレスがバッティングしてしまうため、使用することができません。長所としては、ネットワークに接続されていない環境でも、Windows から常に coLinux に対して接続することができます。短所としては、ネットワーク上の他の PC からは coLinux にアクセスすることができません。

設定は以下のようになります。

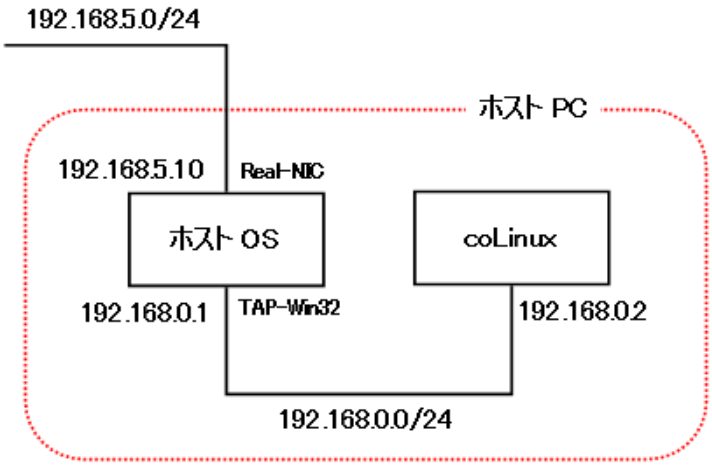


図 12　ネットワーク接続の共有を使用したネットワーク構成

1. 「コントロールパネル」-「ネットワーク接続」を開きます。
2. 「ローカル エリア接続」を右クリックし、プロパティを表示します。
3. 「共有タブ」を開き、「ネットワークのほかのユーザーに、このコンピュータのインターネット接続をとおしての接続を許可する」にチェックを入れます。
4. 「ホーム ネットワーク接続」は「TAP」を選択し、「OK」ボタンで閉じたら完了です。

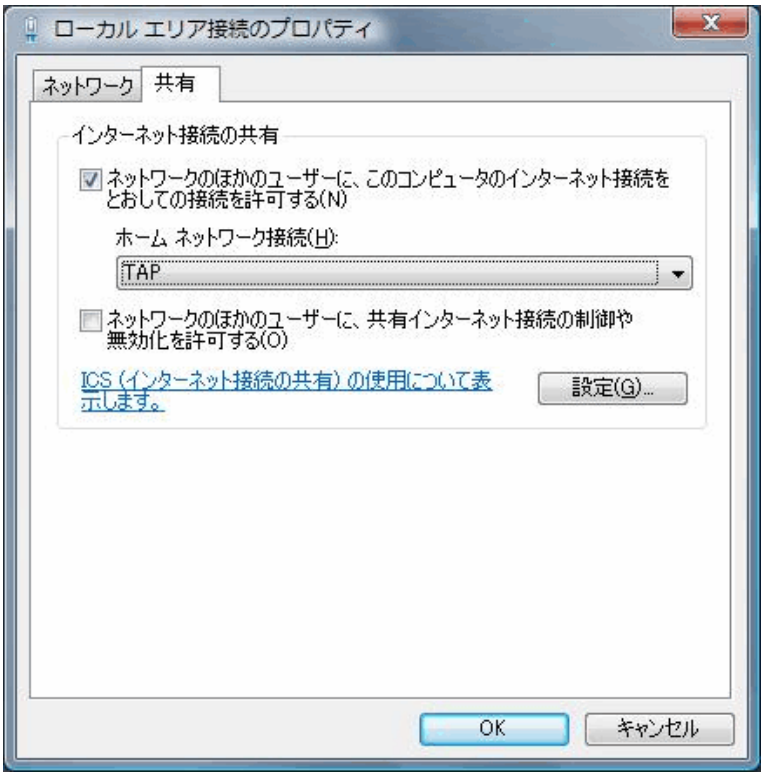


図 13　ネットワーク接続の共有の設定

23.3 coLinux の設定

coLinux を起動するまでには、次の 3 つのステップが必要です。

- Debian のルートイメージの準備
- swap ディスクの準備
- 設定ファイルの作成
- coLinux の起動

23.3.1 Debian のルートイメージの準備

coLinux のインストール時にダウンロードされた Debian のルートイメージファイルが、coLinux インストールディレクトリに「Debian-4.0r0-etch.ext3.1gb.bz2」というファイル名で作成されていますので、ファイルを展開[*78]します。

展開すると「Debian-4.0r0-etch.ext3.1gb」というファイルが生成されるため、分かりやすいように「fs-root-etch.img」などと名前を変更しておきます。

23.3.2 swap ディスクの準備

coLinux のインストールでは、swap ファイルは用意されていないため、各自で用意する必要があります。ここでは既に作成された swap ファイルをダウンロードできるサイトを利用してダウンロードします。

`http://gniarf.nerim.net/colinux/swap/` にアクセスし、「swap_512Mb.bz2」をダウンロードします。

「swap_512Mb.bz2」を展開すると、「swap_512Mb」というファイルが生成されるため、分かりやすいように「fs-swap-512.img」などと名前を変更しておきます。

23.3.3 設定ファイルの作成

これで必要なファイルは全てそろったので、coLinux で Debian を起動するための設定を行います。

設定ファイルのフォーマットには、xml を使用した方法と、独自の conf ファイルの 2 種類がありますが、ここでは conf ファイルを使用した方法を記載します。

coLinux インストールディレクトリに「etch.conf」というファイルを作成し、以下の内容を記載します。

```
kernel=vmlinux
initrd=initrd.gz
mem=512
#「c:\coLinux」は適宜 coLinux のインストールディレクトリで読み替えてください。
cobd0="c:\coLinux\fs-root-etch.img"
cobd1="c:\coLinux\fs-swap-512.img"
eth1=tuntap
root=/dev/cobd0
ro
```

coLinux をインストールしたディレクトリに、「example.conf」というファイルがありますので、必要であればこのファイルを参考に設定を変更してください。

23.3.4 coLinux の起動

coLinux には 2 通りの起動方法が用意されています。

- コンソールアプリケーションとして起動
- サービスアプリケーションとして起動

以下にそれぞれの起動方法を記載します。

[*78] 展開には、bz2 の展開ができる Lhaca などのツールが別途必要になります。

コンソールアプリケーションとして起動する方法

coLinux のインストールディレクトリに、「colinux-daemon.exe」のショートカットを作成し、プロパティから「リンク先」の末尾に「-t nt @etch.conf」を追加します。

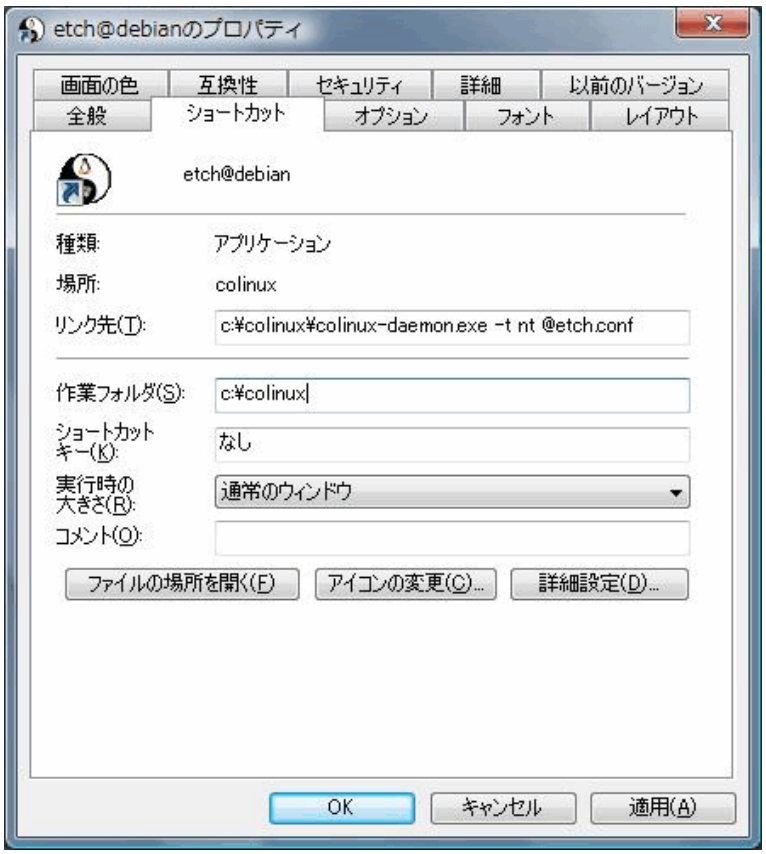


図 14　coLinux 起動ショートカット

このショートカットを起動すると、coLinux がコンソールアプリケーションとして起動します。コンソールアプリケーションとして起動しているため、コンソールを終了すると当然 coLinux も終了してしまいます。

うっかり終了してしまうことを防ぐため、coLinux をサービスアプリケーションとして起動する方法が用意されています。

サービスアプリケーションとして起動する方法

コマンドプロンプトを起動し、coLinux のインストールディレクトリに移動し、以下のコマンドを実行します。「c:\coLinux」は適宜 coLinux のインストールディレクトリで読み替えてください。

```
colinux-daemon.exe "@c:\coLinux\etch.conf" --install-service
```

これで coLinux がサービスとして登録されました。ファイル名を指定して実行より以下のコマンドを実行し、「Cooperative Linux」が登録されていることを確認してください。

```
%SystemRoot%\system32\services.msc
```

「スタートアップの種類」を「手動」に設定し、以下のコマンドでそれぞれ起動、停止することができます。

```
net start "Cooperative Linux"
net stop "Cooperative Linux"
```

24 Debian GNU/kFreeBSD

大浦 真

24.1 Debian GNU/kFreeBSD とは

Debian は、カーネルとして Linux だけをターゲットとしているのではない。Debian GNU/kFreeBSD は、カーネルとして FreeBSD のカーネルを利用し、その上に GNU C library と Debian のパッケージセットを移植したものである。(カーネルだけを使っているので、“**k**FreeBSD”)

正式なリリースはまだだが、今回はそのインストール方法を紹介する。

24.2 Debian GNU/kFreeBSD のインストール

QEMU 上の仮想マシンにインストールする。

1. `http://glibc-bsd.alioth.debian.org/install-cd/` よりインストール CD のダウンロード。現時点での最新版は、20080218 の日付のもの。`debian-20080218-kfreebsd-i386-install.iso`
   - インストーラは、FreeBSD のインストーラを改造したものである。他のアーキテクチャで使われている Debian Installer は移植されていない。
2. qemu 用のイメージの作成

   ```
   $ qemu-img create -f qcow2 kfreebsd.img 10G
   ```
3. qemu 起動

   ```
   $ qemu -hda kfreebsd.img \
   -cdrom debian-20080218-kfreebsd-i386-install.iso -m 512 -boot d
   ```
4. FreeBSD のインストーラが起動したら Express を選択。
5. ハードディスクの設定。
   - FDISK Partition Editor で slice の設定。Create Slice でハードディスク全体を選択。
   - Boot Manager の設定。GRUB も使えるようだが、手動で設定する必要があるので、FreeBSD の Boot Magager を使う。
   - FreeBSD Disklabel Editor。Partition の設定。ひとまず Swap Partition と / を作成。
6. ベースシステムのインストール
   - Choose Distribution で Minimal を選択。
   - Choose Installation Media で CD/DVD を選択。
   - CD から base system のインストールが始まる。
   - Alt-F3 でコンソールに移る。
   - tzdata の設定。
   - popularity-contest の設定。

7. インストールの終了と再起動。
   - User Configuration Requested で No を選択。
   - sysinstall Main menu に戻ったら Exit Install でインストーラを終了。
   - 再起動。
8. 再起動後の設定
   - ユーザとしては、root のみが作成されている。パスワードなし。
   - ネットワークの設定はなされていない。DHCP 環境の場合、dhclient を実行するとネットワークにつながる。
   - ftp.debian-ports.org のアーカイブキーを取得。

     ```
     $ wget -O - http://ftp.debian-ports.org/archive/archive_2008.key \
     | apt-key add -
     ```

24.3 問題点と今後

- インストーラが行う設定が必要最低限のみ。一般ユーザの作成やネットワークの設定は自分で行う必要がある。
- 基本的なパッケージでも移植できていないパッケージやきちんと動作しないパッケージが結構多い。(Debian GNU/Linux であることを前提して作られたパッケージが、GNU/kFreeBSD に持ってくるとビルドできないことがある。)
  - console-tools、sysklogd、emacs22 など
  - 現時点で、Architecuture: any のパッケージの 80% ほどしかビルドできていない。
    `http://buildd.debian-ports.org/stats/`
- ハックのしがいあり。
  - `http://glibc-bsd.alioth.debian.org/TODO`
  - `http://glibc-bsd.alioth.debian.org/porting/PORTING`

24.4 関連リンク

- `http://www.debian.org/ports/kfreebsd-gnu/`
- `http://wiki.debian.org/Debian_GNU/kFreeBSD`
- `http://glibc-bsd.alioth.debian.org/doc/`
  「Installing Debian GNU/kFreeBSD」
- `http://tokyodebian.alioth.debian.org/html/debianmeetingresume200708se7.html`
  「Debian GNU/kFreeBSD のインストール」(第 31 回東京エリア Debian 勉強会資料)

25 chroot からはじめる sid

山下 尊也

sid って危険って言うけど、本当に危険なんですか？とか、unstableって名前が怖かったりとか、はたまた、本日締切りの原稿とかあるのに、新しいバージョンなものを触ってみたいとか、いろいろあると思います。

今回は、初めて sid の環境を触ろうと考えていらっしゃる方にお勧めなものを紹介します。

debootstrap を用いれば、chroot した Debian 環境を安易に構築する事が可能です。

今回は、/home/tommy/chroot/sid の下に chroot 環境な sid を構築します。

```
 $ sudo mkdir -p /home/tommy/chroot/sid
 $ sudo debootstrap sid /home/tommy/chroot/sid http://ftp.jp.debian.org/debian
I: Retrieving Release
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
 ...
I: Configuring klogd...
I: Configuring tasksel-data...
I: Configuring tasksel...
I: Base system installed successfully.
```

さっそく、作った環境にログインしてみましょう。

```
 $ sudo chroot /home/tommy/chroot/sid /bin/bash
root@hoge:/#
```

これでログイン出来ました。

基本的なパッケージしか入っていないため、/etc/apt/souces.list を更新し、自分に必要なパッケージなどをいれましょう。

```
root@hoge:/# vi /etc/apt/sources.list
```

/home などを共有したい場合は、以下のように親の/etc/fstab に書き加えます。

```
 $ sudo vi /etc/fstab
/home           /home/tommy/chroot/sid/home       none    bind          0       0
/tmp            /home/tommy/chroot/sid/tmp        none    bind          0       0
proc-chroot     /home/tommy/chroot/sid/proc       proc    defaults      0       0
devpts-chroot   /home/tommy/chroot/sid/dev/pts    devpts  defaults      0       0
```

ユーザ情報をコピーしましょう。

```
 $ sudo cp /etc/passwd /home/tommy/chroot/sid/etc/
 $ sudo sed 's/\([^:]*\):[^:]*:/\1:*:/' /etc/shadow | sudo tee /home/tommy/chroot/sid/etc/shadow
 $ sudo cp /etc/group /home/tommy/chroot/sid/etc/
 $ sudo cp /etc/hosts /home/tommy/chroot/sid/etc/
```

chroot は、管理者権限がなければ使う事が出来ません。dchroot は、chroot の環境のコマンドを一般ユーザの権限で実行する事を可能にします。アプリケーションを用いたい場合などは、一般ユーザ権限で実行したい場合が多いと思います。

```
$ echo ‘‘mychroot /home/tommy/chroot/sid’’ | sudo tee /etc/dchroot.conf
$ dchroot -c mychroot （ログイン）
```

現在、experimental な環境に iceweasel のバージョン 3 ベータがきています。これを実行する事を考えてみましょう。まず、experimental な環境をインストール出来るように/etc/apt/sources.list を編集します。

```
$ dchroot -c mychroot
$ su -
root@hoge:/# vi /etc/apt/sources.list
deb http://cdn.debian.or.jp/debian/ sid main contrib non-free
deb http://cdn.debian.or.jp/debian/ experimental main contrib non-free
```

そして、experimental な Iceweasel をインストールします。

```
root@hoge:/# aptitude install iceweasel/experimental
```

dchroot は、chroot 環境以下に対して、一般ユーザの権限でコマンドを用いる事も可能です。

```
$ dchroot -c mychroot -d iceweasel
```

これで、chroot 環境内に構築しました experimental の Iceweasel を使う事が出来ます。

chroot は現在動いているカーネル上に仮想 OS を作成することから、オーバーヘッドが比較的小さい事が大きな利点であると思います。

すでに安定版の Debian を使っているのであれば、chroot を用いれば、簡単に不安定版の Debian のアプリケーションを利用する事が出きるので、ちょっと触れてみたい方にはお勧めな方法の一つだと思うので是非試してみて下さい。

25.1 参考文献

- https://wiki.ubuntulinux.jp/UbuntuPackagingGuideJa/appendix-chroot
  The Ubuntu Packaging Guide 日本語版/chroot 環境
- Debian 辞典　著者　武藤健志
- http://kmuto.jp/d/index.cgi/debian/debootstrap.htm
  KeN’s GNU/Linux Diary Etch/Sid の上に Sarge 環境を作る方法

26 ustream.tv を使った関西 Debian 勉強会中継の裏側

野方 純

26.1 はじめに

4 月 29 日の第 12 回関西 Debian 勉強会は、姫路での開催ということもあり、ustream.tv のサービスを利用したインターネットビデオ中継を行いました。

ustream.tv は Windows や Mac の環境からは比較的簡単に利用できるようになっていますが、Debian では少しつまづくところがあったので、その辺りを中心に Tips などを交えながら述べたいと思います。

26.2 ustream.tv について

26.2.1 ustream.tv とは

ustream.tv(`http://www.ustream.tv/`) とは Web ブラウザと Adobe Flash Player を使いインターネットビデオ中継を行うことができる web サービスです。特徴としては、以下のようなものがあります。

- Web ブラウザと Flash プラグインとカメラがあればどこでもストリーミング中継ができる。
- ストリーミング中継は同時に録画でき、すぐに Web 上で公開できる。
- チャットがあるので視聴者も参加でき講演者は反応をダイレクトに知ることができる。

手軽にストリーミング中継ができることから、オープンソース系の勉強会では最近よく使われるサービスです。

26.2.2 ustream.tv を使って中継をするまでの流れ

中継をするまでの流れを `http://www.ustream.tv/get-started` から引用すると、このような感じになります。

1. ustream アカウントを作る (アカウントのある人はログインする)
2. カメラを PC に接続する。
3. ログインして”My Show”をクリック。
4. ”Name Your Show”にストリーミングの番組名を入力して”BROADCAST NOW” ボタンをクリック。
5. Flash プラグインに「www.ustream.tv のカメラおよびマイクへのアクセスを許可しますか?」と尋ねられたら「許可」をする。
6. ”START BROADCAST”ボタンをクリックして放送開始。

26.2.3 中継をするために用意するもの

ustream.tv の中継に必要なものをまとめました。必須なものについては、Debian をデスクトップ環境にて利用されている方は普段使っているものばかりだと思います。

- 必須
  - インターネット接続環境
  - Debian がインストールされたマシン;)
  - Adobe Flash プラグイン
  - Web ブラウザ
  - カメラ
  - マイク
- あれば便利
  - オーディオミキサー
  - IRC クライアント

26.3 実際に中継をする

以上で中継に必要なものと手順がわかったので実際に中継してみました。

26.3.1 中継ができた人とできない人がいた

中継の実験を T 氏と行っていましたが、筆者はほとんどトラブルもなく中継を行うことができたのに対し、T 氏は中継をすることができませんでした。その理由を探っていくとどうやら Flash Player に原因があることがわかりました。

26.3.2 Flash Player について調べる

■Debian GNU/Linux(i386) で使える Flash Player について Debian で使える Flash Player は 3 つ存在しますが、ustream.tv のサービスを利用するには SWF V9 が必要なので、フリーソフトウェアの Gnash と Swfdec を使うことができません。ですので Adobe Flash Player を使用することになります。

ソフト名 (SWF バージョン)	ライセンス	インストール方法
Adobe Flash Player(SWF V9)	プロプライエタリ	non-free セクションを追加して flashplugin-nonfree をインストールするほかに、iceweasel2 を使っていればユーザーディレクトリに自動インストールも可。
Gnash(SWF V7)	GPL V3	aptitude install mozilla-plugin-gnash でインストール可
Swfdec(SWF V7)	GPL V2	aptitude install swfdec-mozilla でインストール可

■Adobe Flash Player の制限 Adobe Flash Player は他の OS とほぼ同じバージョンなので、同じように扱えるように見えますが Linux 版には制限があります。

1. 対応しているビデオ API が Video4Linux(V4L) のみ。(Kernel 2.6 以降では V4L2 が標準。V4L は互換レイヤーで対応) [*79]
2. サウンドの入出力に ALSA を利用しているがサウンドデバイスを直接扱う。

[*79] Adobe Flash Player 10 では V4L2 に対応予定。

3. インプットメソッド (IM) の扱いが GTK の IM 周りを考慮していない。

1 と 2 より使用するデバイスで中継ができた人とできない人に分かれました。3 は直接中継に関係ありませんが、Flash で作られたチャットクライアントを使う際に支障が出ます。

26.4 Adobe Flash で利用できるデバイスを考える

使用できるデバイスに制限があるということが分かったので、Adobe Flash で使えるデバイスについて考えてみます。

26.4.1 カメラ

PC に直接接続できるカメラは大きくわけて、このようなものがあります。

- Web カメラ
  - USB Video Class 対応カメラ (linux-uvc)
  - USB Video Class 非対応カメラ (gspca, ov511, qc-usb)
- DV(デジタルビデオ) カメラ
  - IEEE1394 接続
  - USB 接続

USB 接続のカメラで Web カメラと呼ばれるものは、USB Video Class(UVC) 対応のものと非対応なものでわけることができます。まず USB Video Class 対応カメラはドライバの linux-uvc が V4L2 しか対応しておらず V4L 互換レイヤーをサポートしていないので V4L しか使えない Flash では使うことができません。[*80]

USB Video Class 対応カメラかそうでないかの見分け方は、

1. 解像度が 130 万画素以上。
2. Windows ドライバ不要とうたっている。

どちらかに当てはまれば、UVC 対応と見ればほぼ間違いないでしょう。

USB Video Class 非対応のカメラについては Microsoft VX-1000 と Logitech Quickcam Express の 2 種類使用しましたが、ドライバが V4L に対応していたので楽に使うことができました。ただ UVC 非対応のカメラはどのドライバに対応しているのか自分で見分けなければいけないことと、画素数が 30 万〜50 万クラスのものしかないので画質に不満が出るかもしれません。

デジタルビデオ (DV) カメラについてですが、IEEE1394 接続のカメラは転送されるデータ形式が DV 形式で V4L は直接扱えないので DV4Linux を経由して使う必要があります。映像についてはカメラとしての機能がしっかりしているので不満はないと思います。

USB 接続の DV カメラについては他の OS では使用したとの記事を見かけたのですが、まだ使用したことがないので debian 上で使えるかどうかわかりません。

Linux で使えるカメラについてまとめましたが、現時点では一長一短がありコレといっておすすめできるものがありません。なので、まだまだ試行錯誤は続くと思われます。

26.4.2 カメラの設定について

使用したカメラの設定について述べます。

Microsoft VX-1000/Logitech Quickcam Express の設定 それぞれ UVC 非対応のカメラなので、VX-1000 は gspca、Quickcam Express は qc-usb のカーネルモジュールを使います。標準のカーネルを使っている人は自分の使っ

[*80] The Flashcam Project (`http://www.swift-tools.net/Flashcam/`) の flashcam を利用すれば Flash でも linux-uvc カメラが使えるようです (未確認)

ているカーネル用のカーネルモジュールが用意されるので、それをインストールしてください。
自分で作ったカーネルを使っている人は module-assitant を使ってカーネルモジュールパッケージをインストールします。

```
# aptitude install module-assistant
# m-a a-i gspca (VX-1000 の場合)
# m-a a-i qc-usb (Quickcam Express の場合)
```

あとはカメラを接続してブラウザで ustream.tv にアクセスするだけで使うことができました。

IEEE1394 接続の DV カメラの設定 IEEE1394 接続の DV カメラは、DV4Linux を経由してブラウザを起動する必要があります。

DV4Linux のインストール DV4Linux は debian パッケージになっていないので配布元 (`http://dv4l.berlios.de/`) からソースをダウンロードしてインストールする必要があります。
パッケージ作成練習と自分で使うために作った sid 用 debian パッケージを作りましたが、きちんと作っていないのでリスクを覚悟できる人だけ使ってください。
`http://regret.nofuture.tv/packages/dv4l_1.0-1_i386.deb`

```
$ tar xvfz dv4l-1.0.tar.gz
$ cd dv4l-1.0/
$ ./configure
$ make
$ sudo make install
```

DV4Linux をインストールすると vloopback [*81] を経由して DV カメラとデータをやりとりをする dv4l と、直接 DV カメラとやりとりする dv4lstart の 2 種類がインストールされますが、dv4l は使うと画像が途切れ途切れに送られてきて不安定なので dv4lstart を使用します。
dv4lstart を使ってブラウザを起動するには、このようにします。

```
$ dv4lstart iceweasel
```

統合デスクトップ環境を使っている方はショートカットアイコンを作っておくと便利です。

26.4.3 参考

- DV4Linux
  (`http://dv4l.berlios.de/`)
- ビデオキャプチャ:デバイス編 - Oss4art
  (`http://megaui.net/oss4art/wiki/%E3%83%93%E3%83%87%E3%82%AA%E3%82%AD%E3%83%A3%E3%83%97%E3%83%81%E3%83%A3:%E3%83%87%E3%83%90%E3%82%A4%E3%82%B9%E7%B7%A8`>)
- Debian の最新 Linux kernel で DV 関係が全滅の件 - ほげめも
  (`http://www.keshi.org/blog/2007/08/debian_linux_kernel_disaster.html`)

26.5 サウンド

基本的には ALSA で録音再生できるものであれば種類を問いませんが、Adobe Flash は ALSA の標準的な PCM デバイス名の default を使わず、最初に認識されたサウンドのハードウェアにつけられるデバイス名 hw:0,0 を利用して録音再生しようとします。最近のカメラは USB サウンドを使ったマイクを内蔵している場合があり、音を拾わない場合はこの辺りをチェックするとよいでしょう。[*82]

筆者の場合、USB サウンドユニットを ALSA の設定ファイルの .asoundrc で default に定義して使っていました

[*81] 仮想ビデオデバイスを作るカーネルモジュール。DV4Linux とは別途インストールする必要があります。Vloopback - Oss4art (`http://megaui.net/oss4art/wiki/Vloopback`)
[*82] サウンドデバイスは cat /proc/asound/cards で確認することができます。

が、これではまったく Flash は録音再生をしてくれないので、サウンドカード認識の順番を変更することで対処しました。

26.5.1 サウンドカードの認識順を変更する

サウンドカードの認識順を変更方法は、/etc/modprobe.d/sound を変更する事によって行います。

サウンドカードの設定はこのような感じで指定するので

```
alias snd-card-(番号) snd-(ドライバ名)
options snd-card-(番号) index=(番号)
```

実際にはこうなります。

```
# alias snd-card-0 snd-hda-intel           元々の設定をコメントアウト
# options snd-hda-intel index=0 model=will 元々の設定をコメントアウト
alias snd-card-0 snd-usb-audio
options snd-usb-audio index=0
```

詳しい設定方法についてはカーネルソース付属のドキュメント、Documentation/sound/alsa/ALSA-Configuration.txt を参照してください。

Flash のサウンド入出力について、contrib セクションにある flashplugin-nonfree-extrasound パッケージを使えば、サウンドサーバーの PulseAudio や Esound 経由で変更できるようですが、筆者はデスクトップ環境に KDE を使用しているので未確認です。

26.5.2 参考

- Flash Player:Additional Interface Support for Linux - Adobe Labs (`http://labs.adobe.com/wiki/index.php/Flash_Player:Additional_Interface_Support_for_Linux`)
- PulseAudio - Revolution Linux OpenSource projects (`http://pulseaudio.revolutionlinux.com/PulseAudio`)

26.6 本番で中継をするにあたっての Tips

ustream.tv で中継をするにあたり、障害になるデバイスの問題も解決できたので基本的にはこれで中継を行うことができます。しかし、よりよい中継を行うために少しこだわってみます。

26.6.1 マイクにこだわる

ustream.tv での中継を見ていると、マイクの音が小さすぎて会場で何を言っているのかさっぱりわからない事があります。そうすると見ている方としては興ざめしてしまうので、会場の音をきちんと拾うためにマイクにこだわってみましょう。

姫路からの中継では手持ちのサウンドミキサーとマイクを使用しましたが、サウンドミキサーは 6000〜8000 円前後、マイクは一本 3000 円程度のものでも十分なので用意しておくとよいでしょう。

BEHRINGER というメーカーの ULTRAVOICE XM1800S というマイクは 3 本セットで 4000 円前後と非常にお買い得なのですが、マイクが複数本あると講演者と会場の音というように分けて音を拾うことができるのでおすすめです。

26.6.2 IRC クライアントでチャットに接続する

ustream.tv はビデオストリーミングだけでなく IRC によるチャットも用意されています。しかし用意されているチャットクライアントは Flash で作られており日本語がうまく扱えません。

ストレスをためながらチャットをするよりも、使い慣れた IRC クライアントを使ったほうがストレスもたまらず、

ログなども取れたり便利なので、ustream.tv の IRC サーバーに接続する設定を書いておきます。

項目	設定内容
サーバー名	chat1.ustream.tv
ポート	6667
文字コード	UTF-8
ユーザー名	アカウントがある人は ustream アカウント名。なければ空にしておきます。
ニックネーム	適当なニックネーム。
サーバーパスワード	ustream アカウントのパスワード。アカウントのない人は必要ありません。
チャンネル名	中継アドレスの channel 以下に#をつけたもの。

会場でチャットに参加している人も中継役　ustream のチャットは中継を見ながらチャットしているので当然ながら講演に対しての質問など意見が出てきます。ですが講演者が必ずしもチャットを見ているわけではないので、会場でチャットに参加している人がいるなら、適当なころ合いを見計らってチャットで出た質問などを視聴者に代わってしてあげてください。

そうすると会場はもちろん、インターネットを通じて見ている人も盛り上がることは間違いないありません。

26.6.3 画面を中継するには?

今までカメラを使っての中継を書いてきましたが、実は/dev/video に出力できるものなら、なんでも中継することができます。Screencast4Linux を使えば PC の画面をキャプチャして/dev/video に流すことができるので、画面を中継することもできます。

ちなみに Screencast4Linux はまだ debian パッケージにはなっていません。筆者はカーネルモジュールをパッケージにする方法をまだわかっていないので誰か挑戦してみて!

26.6.4 参考

- Screencast4Linux: Linux の X Window 上で手軽にスクリーンキャスト
  (`http://yanbe.org/screencast4linux/`)

26.7 最後に

本来ならばここまで大きくなることもなかったのですが、Adobe Flash Player の奇妙な仕様によりこんなに長くなってしまいました。

Adobe も「Open Screen Project」の一環として仕様や API の利用制限を撤廃したので、自由なソフトウェアの Flash Player が発展し、早くこのようなバッドノウハウに頼らなくなればと思っています。

26.8 資料

26.8.1 ビデオデバイス周り

- V4LWiki
  Video4Linux の Wiki。Linux のビデオデバイス周りの情報は一通り揃っています。
  `http://linuxtv.org/v4lwiki/index.php/Main_Page`
- Welcome to QuickCam Team! - QuickCam Team
  Logitech Quickcam Team のまとめサイト。ドライバの情報がまとまっています。
  `http://www.quickcamteam.net/`
- VideoFourLinuxLoopbackDevice
  仮想ビデオデバイス vloopback のサイト

http://www.lavrsen.dk/twiki/bin/view/Motion/VideoFourLinuxLoopbackDevice

- AVLD - Another Video Loopback Device
  もう一つの仮想ビデオデバイスの実装。
  http://allonlinux.free.fr/Projets/AVLD/
- gstfakevideo
  gstreamer に流すための仮想ビデオデバイス。gstreamer は鼻血が出るほどむずかしい。
  http://code.google.com/p/gstfakevideo/
- Screencast4Linux: Linux の X Window 上で手軽にスクリーンキャスト
  PC の画面をスクリーンキャストする。
  http://yanbe.org/screencast4linux/
- DV4Linux
  DV カメラと V4L とやりとりするラッパーソフト。
  http://dv4l.berlios.de/
- The Flashcam Project
  UVC カメラと V4L をやりとりするラッパーソフト。
  http://www.swift-tools.net/Flashcam/

26.8.2 その他

- Ustream まとめ Wiki
  ustream.tv で中継をするためのノウハウを集めた Wiki。
  http://usy.jp/ustream/index.php?FrontPage
- ustream.tv から smilevideo にアップロードするための講座
  ドワンゴの溝口氏による ustream.tv で録画した flv を ffmpeg を使ってエンコードしてニコニコ動画の smilevideo にアップロードするまでの講座。
  http://www.ustream.tv/recorded/379002
- 表現のためのオープンソースソフトウェア Oss4art
  EffecTV 開発者のふくち氏による Linux マルチメディア系の情報を集めた Wiki。
  http://megaui.net/oss4art/wiki/
- HowtoCast - dcastkit
  八重樫氏製作による Debian Live ベースのライブストリーミングに特化した LiveCD dcastkit を使ったストリーミング方法。
  https://ssl.keshi.org/projects/dcastkit/trac.fcgi/wiki/HowtoCast
- Penguin.SWF
  Adobe の Linux 用 Flash 開発者の blog
  http://blogs.adobe.com/penguin.swf/
- Adobe Bug and Issue Management System
  Flash Player のバグトラック。まだできて間もないのであまりバグが登録されていません。
  https://bugs.adobe.com/flashplayer/

27 Debian で始める Emacs エディタ

木下 達也

27.1 GNU Emacs とは

GNU Emacs は、拡張・カスタマイズが可能なテキストエディタです。

GNU システムの構成要素として 1984 年から開発が始まり、現在もなお愛され続けています。

テキストエディタとしての基本的な機能に加えて、複数の「バッファ」それぞれに応じた「モード」、キー割り当てのカスタマイズ、Emacs Lisp によるプログラミングが可能、といった特徴があります。

その柔軟性は、文書作成、コーディングという用途に留まらず、メール送受信、ウェブブラウズなどもこなせるほどです。

27.2 Emacs のインストール

27.2.1 Debian パッケージ

Debian では emacs パッケージをインストールすれば、Debian 標準の Emacs パッケージ (etch では emacs21、lenny では emacs22) がインストールされます。

```
# apt-get install emacs
```

ウィンドウアプリケーションとして動かすのではなく、端末でのみ利用したいのであれば、代わりに-nox パッケージ (etch では emacs21-nox、lenny では emacs22-nox) を選べば、依存関係が少なく、インストールサイズが小さくて済みます。

27.2.2 Unicode 日本語対応

emacs21 では、Unicode (UTF-8) 日本語対応のためには、別途 mule-ucs パッケージが必要です。(emacs22 では mule-ucs 無しでも対応)

```
# apt-get install mule-ucs
```

mule-ucs を有効にするには、設定ファイルのサンプル`/usr/share/doc/mule-ucs/examples/dot.emacs.ja` の内容を、`~/.emacs` の先頭の方に貼り付けておくとよいでしょう。(emacs22 でも有用な設定が含まれています)

注: mule-ucs を有効にする際には、やや時間がかかります。また、`~/.emacs` 自身は UTF-8 にしないように。`~/.emacs` 読み込み前に環境変数で mule-ucs を有効にする方法もあります。`/usr/share/doc/mule-ucs/README.Debian` を参照。

27.3 Emacs の利用法

起動方法:

```
emacs                          通常起動
emacs -nw                      端末内で起動
emacs -q                       ~/.emacs を読まずに起動
emacs -q -no-site-file         startup ファイルを読まずに起動 (emacs22 なら-Q でも可)
```

キーボード操作に慣れることで、Emacs をより快適に使えるようになります。

```
C-x     Ctrl キーを押しながら x
M-x     ESC を押してから x  (または Alt キーを押しながら x)

ESC     Esc キーまたは C-[
TAB     Tab キーまたは C-i (字下げや補完に使う)
RET     Enter キーまたは C-m (改行と同時に字下げする場合には C-j)
DEL     Back space キー
SPC     スペースキー

C-x C-c         Emacs を終了

C-p  上へ       C-n  下へ
C-b  左へ       C-f  右へ
C-a  行頭へ     C-e  行末へ

DEL     手前 1 文字を削除 (C-h はデフォルトでは help-command)
C-d     1 文字を削除
C-k     行末まで削除
C-o     行末までを次の行へ (改行を挿入、位置はそのまま)

C-x C-f         ファイルを読み込む
C-x C-s         ファイルを保存
C-x i           現在位置にファイルの内容を挿入
C-x C-w         別ファイルへ保存

C-x RET f       現在のバッファの coding system(文字コード) を設定
C-x RET c       次に実行するコマンドの coding system を設定

C-v 次ページへ  M-v 前ページへ
M-< 先頭へ      M-> 末尾へ

C-s  前方検索   C-r  後方検索  (M-付きで正規表現)
M-%  置換  (C-付きで正規表現)

C-SPC マーク
C-w カット   M-w コピー   C-y ペースト

C-g     処理を中断
C-x u   Undo
C-l     現在行を中央にして再表示

C-x b バッファ移動   C-x k バッファ削除

C-x (   キーボードマクロ記録開始
C-x )   キーボードマクロ記録終了
C-x e   キーボードマクロ実行

C-x 2   ウィンドウ分割
C-x 1   他のウィンドウを削除
C-x 0   現在のウィンドウを削除

M-x     コマンド実行 (Lisp 関数を呼び出し)
M-:     ミニバッファで入力して Lisp 式を評価
C-x C-e 手前の Lisp 式を評価 (*scratch*では C-j で評価・表示)
M-!     外部コマンド実行 (C-u 付きで出力を現在位置へ挿入)

M-x help RET    ヘルプ
M-x info RET    マニュアル
M-x apropos RET 関数、変数等の名前を検索
```

カスタマイズ例:

```
(global-set-key "\C-h" 'delete-backward-char)
(global-set-key "\C-ch" 'help-command)
(global-set-key "\C-z" 'scroll-down) ;; C-x C-z to use suspend/iconify
(global-set-key "\C-cg" 'goto-line)
(if (locate-library "iswitchb") (require 'iswitchb))
(if (fboundp 'iswitchb-default-keybindings) (iswitchb-default-keybindings))
(global-set-key "\C-cb" 'switch-to-buffer)
(setq visible-bell t)
(setq inhibit-startup-message t)
(if (fboundp 'menu-bar-mode) (menu-bar-mode -1))
(if (fboundp 'tool-bar-mode) (tool-bar-mode -1))
(if (fboundp 'auto-compression-mode) (auto-compression-mode 1))
```

27.4 add-on パッケージ

Emacs 標準の機能だけでなく、さらに追加できる Emacs Lisp パッケージがたくさんあります。
2008-05-13 時点の Debian sid で、usr/share/emacs を含むパッケージ:

```
# apt-file update
$ apt-file search usr/share/emacs | awk '{print $1}' | sort -u | cat -n
     1  a2ps:
     2  acl2-emacs:
     3  ada-mode:
     4  afnix:
     5  anthy-el:
     6  apel:
[...]
   256  yatex:
   257  yc-el:
   258  yorick-data:
   259  zenirc:
```

個人的なお勧め・注目パッケージ:

- メールクライアント: mew, mew-beta, wl, wl-beta
- 引用ツール: mu-cite
- スケジュール管理: mhc, planner-el
- ウェブブラウザ: w3m-el, w3m-el-snapshot
- 日本語入力: ddskk, prime-el
- 辞書検索: lookup-el
- ローマ字での日本語検索: migemo
- 編集支援: muse-el, auctex, yatex

27.5 emacsen パッケージ

Debian sid に現時点で存在する emacsen パッケージ:

- emacs22 (emacs22, emacs22-gtk, emacs22-nox)
- emacs21 (emacs21, emacs21-nox)
- xemacs21 (xemacs21-mule, xemacs21-mule-canna-wnn, xemacs21-gnome-mule, xemacs21-gnome-mule-canna-wnn, xemacs21-nomule, xemacs21-gnome-nomule)

非公式パッケージとして、Romain Francoise さんによる emacs-snapshot パッケージ (CVS 版 Emacs 23.0.x) もあります。(`http://emacs.orebokech.com/`)

Emacs Lisp add-on パッケージでは、emacsen それぞれに応じた処理 (バイトコンパイル等) はインストール時に実施されるので、パッケージはほぼ共通のものが使えます。

月例 Debian GNU/kFreeBSD

大浦 真

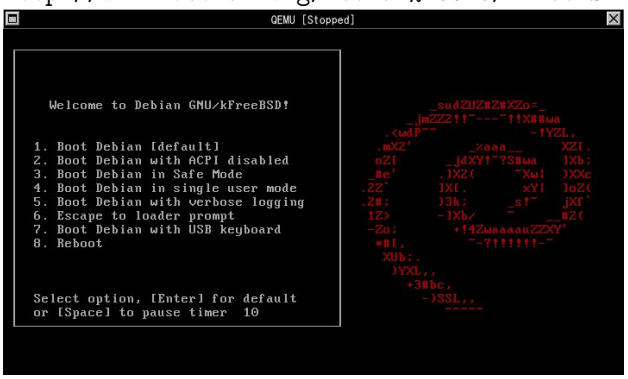
Debian Project の大浦です。(現在、京都在住) 今月からこの場をお借りして Debian GNU/kFreeBSD の状況をお伝えすることになりました。よろしくお願いします。

Debian は、現在、その OS の中心として Linux カーネルを利用していますが、Debian GNU/kFreeBSD とは、カーネルとして FreeBSD のカーネルを利用し、その上に GNU C library と GNU のユーザランド、および、Debian のパッケージセットを移植したものです。FreeBSD のカーネルだけを使っているので、“**k**FreeBSD” と呼んでいます。

そんな Debian GNU/kFreeBSD は、開発途上にあり、Debian の一部として、まだ、正式にはリリースされていません。しかし、FreeBSD のインストーラを改造した kFreeBSD 用のインストーラも用意されていますし、kFreeBSD 用の buildd も設置されていますので、比較的簡単に実機やエミュレータ上で試すことができます。また、X.Org も使えますので、それなりに使うことができるようになっています。ただ、重要なパッケージで、buildd でビルドできず、バイナリが用意されていないものがあったり、用意されていてもまともに動作しないものがあったりもするので、まだまだ、いろいろとハックのしがいがあると思います。

Debian GNU/kFreeBSD に関する情報は、以下の Debian Wiki のページにまとまっていますので、ぜひご覧下さい。

http://wiki.debian.org/Debian%20GNU/kFreeBSD



月例 Debian GNU/Linux SuperH port

岩松 信洋

今月の Debian GNU/Linux SuperH 移植版の状況をお伝えします。

Debian/SH port は先月から加速しはじめました。その理由として、フランス人の方が協力してくれることになったためです。いまのところ、パッケージ用リポジトリを共同にしようということになっており、新しい SH4 用のリポジトリがフランスと日本にできました。

現在のステータスは、gcc-4.3 ベースでパッケージを作っています。マシンが複数台あるのですが、buildd をまだ 1 台しかセットアップしていません。今月中には複数台セットアップしたいと考えています。

現在抱えている問題として、mesa のコンパイルに失敗するというものがあります。internal compiler error なので、調べるのは難しいですが、マクロ展開でこけているところまで追いました。絶賛調査中です。

```
gcc -c -I../../include -I../../src/mesa \
-I../../src/mesa/main \-I../../src/mesa/glapi \
-I../../src/mesa/math -I../../src/mesa/tnl
-I../../src/mesa/shader \
-I../../src/mesa/shader/grammar \
-I../../src/mesa/shader/slang -I../../src/mesa/swrast \
-I../../src/mesa/swrast_setup -Wall -Wmissing-prototypes \
-O2 -g   -D_POSIX_SOURCE -D_POSIX_C_SOURCE=199309L \
-D_SVID_SOURCE -D_BSD_SOURCE -D_GNU_SOURCE -DPTHREADS\
 -DUSE_XSHM -DHAVE_POSIX_MEMALIGN  -I/usr/X11R6/include \
-std=c99 -ffast-math  -fno-strict-aliasing \
vbo/vbo_exec_api.c -o vbo/vbo_exec_api.o
vbo/vbo_exec_api.c: In function 'vbo_exec_fixup_vertex':
 vbo/vbo_exec_api.c:340: internal compiler error: Segmentation fault
 Please submit a full bug report,
with preprocessed source if appropriate.
See  for instructions.
 For Debian GNU/Linux specific bug reporting instructions,
 see .
```

今月のパッチ

- util-linux
  sh4 アーキテクチャで mount パッケージが作成されない問題を修正。

月例 Debian GNU/Hurd

山本 浩之

今月の Debian GNU/Hurd の状況をお伝えします。Debian GNU/Hurd は sid に入っていながら、未だ testing にすら入れてもらえず、冷や飯を食わされ続けていますが、それでもなお、 開発者たちは地味に頑張りつづけ、現在では K16CD インストールイメージをリリースするに至っております。まずはこの K16CD-mini インストールイメージを VMware ディスクイメージにインストールしてみて問題点を調べてみました。

ユーザランドは基本的には Debian GNU/Linux と同じですが、いくつか Hurd 特有の呪文があります。

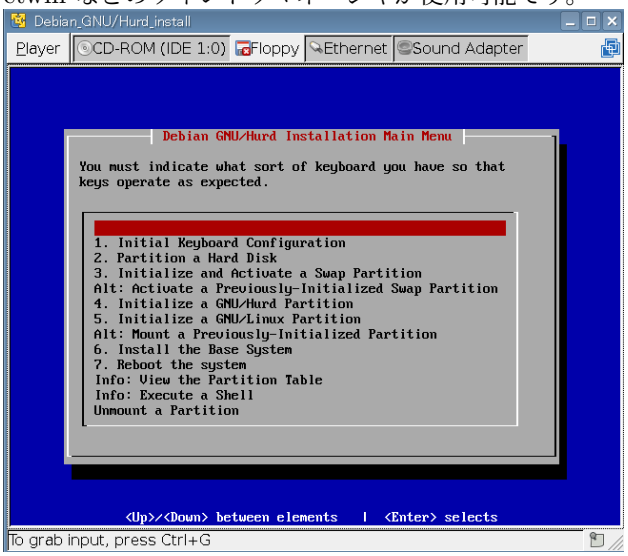
まず、ネットワークの設定ですが、/sbin/ifconfig にあたるものがなく、

```
# settrans -fgap /servers/socket/2 /hurd/pfinet \
 -i eth0 -a 192.168.1.3 -g 192.168.1.1 -m 255.255.255.0
```

というように、settrans コマンドを使用します。また X を起動するにはかなり謎 (?) な呪文を唱えないといけません：

```
# console -d vga -d pc_mouse --repeat=mouse -d pc_kbd \
 --repeat=kbd -d generic_speaker -c /dev/vcs
```

現在のところ、Gnome や KDE のような統合デスクトップ環境は、build-dep なパッケージが FTBFS バグのため揃っておらず、提供されていませんが、Window Maker、Afterstep、IceWM、qvwm、ratposion、fvwm、ctwm などのウィンドウマネージャが使用可能です。



月例 Nexenta Operating System

上川 純一

2008 年 4 月号でインストールしてみた Nexenta Core Platform 1.0 で今月も何か操作してみます。まず、ホスト OS とのデータの交換の方法を確立してみます。

仮想マシン内部を快適に利用するために、まず SSH 接続経由でのファイルシステムを共有をしてみましょう。

まず、ゲスト OS の 22 番ポートをホスト OS の 2299 番ポートとしてゲスト OS を起動してみました。こうすると ssh で localhost の 2299 番ポートにアクセスしてゲスト OS にログインできます。

```
dancer@host$ qemu -hda nexenta.cow -m 512 -redir tcp:2299::22 &
dancer@host$ ssh localhost -p 2299
Password:
Last login: Wed Apr 30 05:02:35 2008 from 10.0.2.2
dancer@vm1:~$
```

この状態で sshfs を利用するとゲスト OS のパスをマウントして見ることができます。ホスト OS からゲスト OS の/export/home/dancer にシームレスにアクセスすることができます。

```
dancer@host$ sshfs localhost:  ./nexenta/  -p2299
Password:
dancer@host$ ls -la nexenta/
合計 36
drwxr-xr-x 1 dancer uucp      9 2008-04-15 23:53 .
drwxr-xr-x 5 dancer dancer 4096 2008-04-30 14:05 ..
-rw------- 1 dancer uucp   1130 2008-04-23 13:05 .bash_history
-rw-r--r-- 1 dancer uucp    220 2008-03-19 18:07 .bash_logout
[以下略]
```

nexenta では各種ファイルを staff グループ (gid=10) が所有し、それがちょうど Debian では uucp グループ (gid=10) が所有しているように見えています。

```
dancer@vm1:~$ id dancer
uid=1000(dancer) gid=10(staff) groups=10(staff)
```

28 Debian Trivia Quiz

上川 純一

ところで、みなさん Debian 関連の話題においついていますか？Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。漫然と読むだけではおもしろくないので、Debian 関連の話題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。

28.1 問題

出題範囲は `debian-devel-announce@lists.debian.org` に投稿された内容からです。

問題 1. Debian Developer になる前の人たちのための制度で最近 Open Beta を開始したのは何か

☐ A Debian Maintainers
☐ B Ubuntu
☐ C Debian Account Manager

問題 2. Debian Auditor は誰か

☐ A Kalle Kivimaa
☐ B Anthony Towns
☐ C Sam Hocevar

問題 3. Anibal Monsalve Salazar が参加したのはどのチームか

☐ A Debian System Administrator
☐ B Debian Cabal
☐ C Debian Maintainer Keyring Team

問題 4. debian/control に追加された Vcs-* フィールドでないのはどれか

☐ A Vcs-git
☐ B Vcs-rcs
☐ C Vcs-Mtn

問題 5. debian/control でパッケージのアップストリームプロジェクトの URL を記述するためのフィールドは

☐ A Homepage:
☐ B URL:
☐ C Upstream:

問題 6. dpkg-buildpackage で並列ビルドをするためのコマンドラインオプションは

☐ A -rfakeroot
☐ B -j
☐ C –nori1

問題 7. 2009 年開催予定の Debconf9 の場所が発表された、その場所は

☐ A Osaka
☐ B Tokyo
☐ C Extremadura

問題 8. Policy 3.7.3.0 で追加されなかった変更は

☐ A バージョン番号にチルドが利用できるようになった
☐ B 自由でないものはパッケージではない
☐ C Debconf を利用する場合は国際化対応必須

問題 9. gnuab にかわるリリースされていない Debian 移植版のホスティングの場所はどこか

☐ A `http://www.debian-doorstop.org/`
☐ B `http://www.debian-ports.org/`
☐ C `http://www.ubuntu.org/`

28.2 問題

debian-devel-announce@lists.debian.org に投稿された内容からです。

問題 10. Debian Miniconf 7 が開催される、Linux Conf Au はいつ開始か

☐ A 1 月 28 日

☐ B 1 月 29 日

☐ C 1 月 19 日

問題 11. lintian は Debian パッケージのよくある間違いを検出してくれるツールである。 lintian.debian.org は全アーカイブに対して実行した lintian の実行結果を報告してくれているページだが、メンテナ単位のウェブページの URL が変更になった。どうかわったか？

☐ A http://www.youtube.com/email.html

☐ B http://people.ubuntu.com/~liw/lintian/gutsy-i386-main/reports/maintainer/email.html

☐ C http://lintian.debian.org/reports/maintainer/email.html

問題 12. dpkg のシンボルファイルで何が実現できるか？

☐ A 共有ライブラリなんてつかってられないので全部 DLL にしてみた依存関係

☐ B 共有ライブラリはどんなバージョンでもよくなるような依存関係

☐ C 共有ライブラリのバージョン付きシンボルをベースにした依存関係

問題 13. http://wiki.debian.org/HelpDebian/Start には何がかかれているか

☐ A なぜ Debian に貢献するべきか

☐ B Debian はなぜ存在するのか

☐ C GNU の存在意義

問題 14. d-i での翻訳作業の省力化のための工夫を Christian Perrier が実施した。それはメッセージを使われかたの優先度によって分割するものだったが、何分割にしたか

☐ A 5

☐ B 4

☐ C 3

問題 15. Fonts task force (http://wiki.debian.org/Fonts) は週次で何を確認するスクリプトを用意したか？

☐ A Debian パッケージとしてはたしてどれだけのフォントファイルが存在するのかを棚卸する

☐ B いかがわしい形のフォントを検出する

☐ C うまく表示できないフォントを検出する

問題 16. Lenny リリースに標準として含まれる予定の KDE はどれか

☐ A KDE3.0

☐ B KDE4.0

☐ C KDE5.0

問題 17. Debian-i18n meeting で main,contrib,non-free/i18n/Translation-* に対して何がおきたか

☐ A Grisu が悟りを開いた

☐ B AUTOBYHAND の仕組みを利用するようになった

☐ C あきらめて変更しないことにした

問題 18. FOSS.in の開催期間中に DD になったインド人は何人目のインド人 DD か？

☐ A 1

☐ B 2

☐ C 4

問題 19. insserv では何が実現できるか？

☐ A 自動でデータベースに insert してくれる

☐ B 依存関係から init スクリプトの実行順序を計算して実行

☐ C ネームサービスの提供

28.3 問題

debian-devel-announce@lists.debian.org に投稿された内容からです。

問題 20. lenny でのリリースゴールの 1 つに GCC 4.3 でのビルドが通ることが挙げられているが、それに伴って変更する必要が最も多く生じるのは、どの言語で書かれたプログラムか?

☐ A C

☐ B C++

☐ C Ruby

問題 21. 今のところ lenny で完全に削除される予定なのは次のうちどれか?
☐ A GNOME 1.x
☐ B KDE 3.x
☐ C Linux 2.x

問題 22. Debian に宣伝力がないと感じた Debian プロジェクトリーダー (DPL) の Sam Hocevar が提案したのは?
☐ A イベント会場で Debian ロゴの入ったシャツを着て練り歩く Debian Tank Top & Brief Team
☐ B ロゴや T シャツのデザインなどマーケティング面の任務を一手に引き受ける Debian Marketing Team
☐ C Debian のマスコットキャラクター「でびにゃん」

問題 23. 3 月頭の時点で lenny のリリースにとって致命的なバグ (RC バグ) は 460 ほどあるが、それを減らす方法として推奨されているのは?
☐ A debian/changelog に「closes: #xxxxxx」と書いて片っ端からパッケージをアップロードする
☐ B バグ退治パーティー (BSP) を開いて片っ端からパッケージを非メンテナアップロード (NMU) する
☐ C RC バグを放置しているパッケージメンテナを片っ端から呼び出して説教する

問題 24. lenny はリリースされると何になる?
☐ A Debian 4.1
☐ B Debian 5.0
☐ C Ubuntu 8.0

問題 25. lists.debian.org の新しい検索エンジンには何が用いられているか?
☐ A Google Search
☐ B Hyper Estraier
☐ C Xapian Omega

問題 26. 新しい armel アーキテクチャに関する現況として適切でないものは?
☐ A lenny のリリースアーキテクチャに含まれている
☐ B 全パッケージの 90% がビルドされている
☐ C パッケージを ftp.debian.org からダウンロードできるようになっている

問題 27. 次の開発者のうち、今年の DPL 選挙に立候補したのは?
☐ A Lucas Nussbaum
☐ B Junichi Uekawa
☐ C Steve McIntyre

28.4 問題

`debian-devel-announce@lists.debian.org` に投稿された内容からです。

問題 28. パッケージの品質確認に有用な次のツールのうち、lenny および sid から削除されたものは?
☐ A lintian
☐ B linda
☐ C piuparts

問題 29. init スクリプトの実行順序を依存関係から自動的に決定できるようにすることが lenny のリリースゴールの一つになっているが、2008 年 4 月 16 日現在、/etc/init.d に init スクリプトを持つパッケージの数は?
☐ A 873
☐ B 857
☐ C 98

問題 30. 次のうち、Debian Installer Lenny Beta 1 における改良点でないものは?
☐ A NTP を用いた時刻合わせ
☐ B volatile のサポートの追加
☐ C Mac OS X からのインストーラ起動のサポート

問題 31. 新たな Debian プロジェクトリーダー (DPL) に選ばれた Steve McIntyre が目標の一つに掲げているものは?
☐ A Communications within the project
☐ B Make Debian sexy again
☐ C Welcoming people to the project

問題 32. Debian Weekly News を復活させようというメールを debian-devel-announce に流したのは?
☐ A Martin Schulze
☐ B Joey Hess
☐ C Alexander Schmehl

問題 33. lenny のリリースに関する現況として正しくないものは?
☐ A GCC 4.3 がデフォルトコンパイラとなり、それでコンパイルできないことが RC バグとなった
☐ B lenny のリリースに向けて既に必須パッケージがフリーズされている
☐ C デフォルトの syslog デーモンを rsyslog に変更することが決定した

問題 34. dpkg 1.14.18 で新たにいくつかのソースパッケージ形式がサポートされるようになったが、それらのうち次期デフォルトとされる「3.0 (quilt)」が持っていない機能は?
☐ A ソースパッケージの upstream tarball として複数のファイルを使用できる
☐ B ソースパッケージの diff.gz ファイルを、適切に複数のパッチに分けてくれる
☐ C ソースパッケージに Debian 固有のバイナリファイルを挿入できる

問題 35. 今年の DPL 選挙の投票率は?
☐ A 62.248%
☐ B 53.084%
☐ C 37.302%

問題 36. etch の次期ポイントリリースは「etch and a half」になるとされているが、そこに含まれる予定の Linux カーネルのバージョンは?
☐ A 2.6.18
☐ B 2.6.24
☐ C 2.6.25

問題 37. 次の開発者のうち、新たに FTP マスターに任命されたのは?
☐ A Sam Hocevar
☐ B Joerg Jaspert
☐ C James Troup

問題 38. 次の開発者のうち、Debian Marketing Team のメンバーに任命されていないのは?
☐ A Sam Hocevar
☐ B Andreas Schuldei
☐ C Moritz Muehlenhoff

問題 39. Joerg Jaspert が DAM に任命されたのは Sam Hocevar の任期の切れる何時間前か？
☐ A 24 時間
☐ B 2 時間
☐ C -1 時間

問題 40. Software Design 2008 年 4 月号 155 ページの git-import-dsc の事例にのっているパッケージ名は？
☐ A ecasound
☐ B pbuilder
☐ C Windows Media Player

問題 41. 昨日 Debian 勉強会の参加メンバーで Debian Developer になったのはだれか？
☐ A 岩松
☐ B 小林
☐ C Charles Plessy

28.5 問題

`debian-devel-announce@lists.debian.org` への投稿内容からです。
問題 42. Joerg Jaspert が dak のソースコードに取り込んだ、Thomas Viehmann からのパッチとは?
☐ A パッケージを受け取ったときにメンテナだけでなくスポンサーにもメールを送るようにするパッチ
☐ B パッケージを受け取ったときに現在のカルマをメンテナに通知するパッチ
☐ C パッケージを受け取ったときに現在のバグの一覧をメンテナに通知するパッチ

問題 43. debhelper バージョン 7 では、何から学んだことを取り込んでいるか
☐ A MSDN
☐ B CDDB
☐ C CDBS

問題 44. 新たにリリースマネージャになったのは?
☐ A Andreas Barth
☐ B Luk Claes
☐ C Marc 'HE' Brockschmidt

問題 45. Debian パッケージメンテナンスに関する自動通知メールにおいて、メールを生成したソフトウェアを示すのに使うよう提案されたメールヘッダは?

☐ A X-Debian:

☐ B X-Debian-Package:

☐ C User-Agent:

28.6 問題

`http://www.debian.org/News/weekly/2008/01/` にある 4 月 21 日版です。

問題 46. Siobhán O'Mahony と Fabrizio Ferraro の研究では Debian の統治システムの歴史を 4 つの時期に分けているが、そのうち 1999 年から 2003 年に当たるものは?

☐ A Stabilizing Governance

☐ B Implementing Governance

☐ C Designing Governance

問題 47. Institute for Advanced Professional Studies Yankee group の研究結果として適切でないものは?

☐ A 2006 年と比べて Debian サーバのダウンタイムが 41% 減った

☐ B 2006 年と比べてネットワークに 1 台以上の Debian を入れているユーザが 15% から 24% に上昇した

☐ C 2006 年と比べて Debian ユーザが 41% 減った

問題 48. 開発者のブログ記事を収集して表示する Planet Debian に関して、Martin Joey Schulze が始めたサービスとは?

☐ A 記事をメーリングリストとして購読できるようにするサービス

☐ B すべての記事に対してコメントを一斉に送れるようにするサービス

☐ C 記事を Slashdot や各種ソーシャルブックマークサービスに容易に投稿できるようにするサービス

問題 49. Christian Perrier によると、最近正式な Debian 開発者として登録されたメンバーの女性率は?

☐ A 5%

☐ B 10%

☐ C 50%

28.7 問題

`http://www.debian.org/News/weekly/2008/02/` にある 5 月 9 日版です。

問題 50. WWW の生みの親として有名な Tim Berners-Lee が WWW2008 カンファレンスの基調演説で Debian について述べたこととは?

☐ A Debian のウェブページのように古臭いページは消えていく運命にある

☐ B Debian のウェブページは様々な言語に地域化されており素晴らしい

☐ C Debian のパッケージングシステムは素晴らしい

問題 51. 今年の Google Summer of Code プログラムで Debian Project のタスクを得た学生の人数は?

☐ A 6

☐ B 12

☐ C 24

問題 52. 5 月 9 日現在、lenny のリリースに向けて進められている移行の状況として適切でないものは?

☐ A Perl のバージョン 5.10 を lenny でデフォルトとするための移行作業が行われている

☐ B 既に lenny において Python のデフォルトがバージョン 2.5 になっている

☐ C Ruby のバージョン 1.9 を lenny でデフォルトとするための移行作業が行われている

問題 53. 4 月 18 日に何人の Debian 正式開発者が追加されたか?

☐ A 9

☐ B 19

☐ C 29

29 Debian Trivia Quiz 問題回答

上川 純一

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

1. A
2. A
3. C
4. B
5. A
6. B
7. C
8. B
9. B
10. A
11. C
12. C
13. A
14. A
15. A
16. A
17. B
18. C
19. B
20. B
21. A
22. B
23. B
24. B
25. C
26. A
27. C
28. B
29. A
30. C
31. A
32. C
33. C
34. B
35. C
36. B
37. B
38. A
39. B
40. A
41. C
42. A
43. C
44. C
45. A
46. B
47. C
48. A
49. B
50. C
51. B
52. C
53. B

30 索引

『あんどきゅめんてっど でびあん』について

本書は、東京および関西周辺で毎月行なわれている『東京エリア Debian 勉強会』および『関西エリア Debian 勉強会』で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。収録範囲は東京エリアは勉強会第 35 回から第 40 回、関西エリアは第 10 回から第 13 回まで。内容は無保証、つっこみなどがあれば勉強会にて。

あんどきゅめんてっど でびあん 2008 年夏号

2008 年 8 月 16 日　　初版第 1 刷発行
東京エリア Debian 勉強会/関西エリア Debian 勉強会 （編集・印刷・発行）