東京エリア Debian 勉強会/つくらぐ勉強会

資料

岩松 信洋 iwamatsu@debian.org
IRC nick: iwamatsu

2010 年 05 月 15 日

設営準備にご協力ください

勉強会の連絡事項

- 注意事項
  - 飲食?

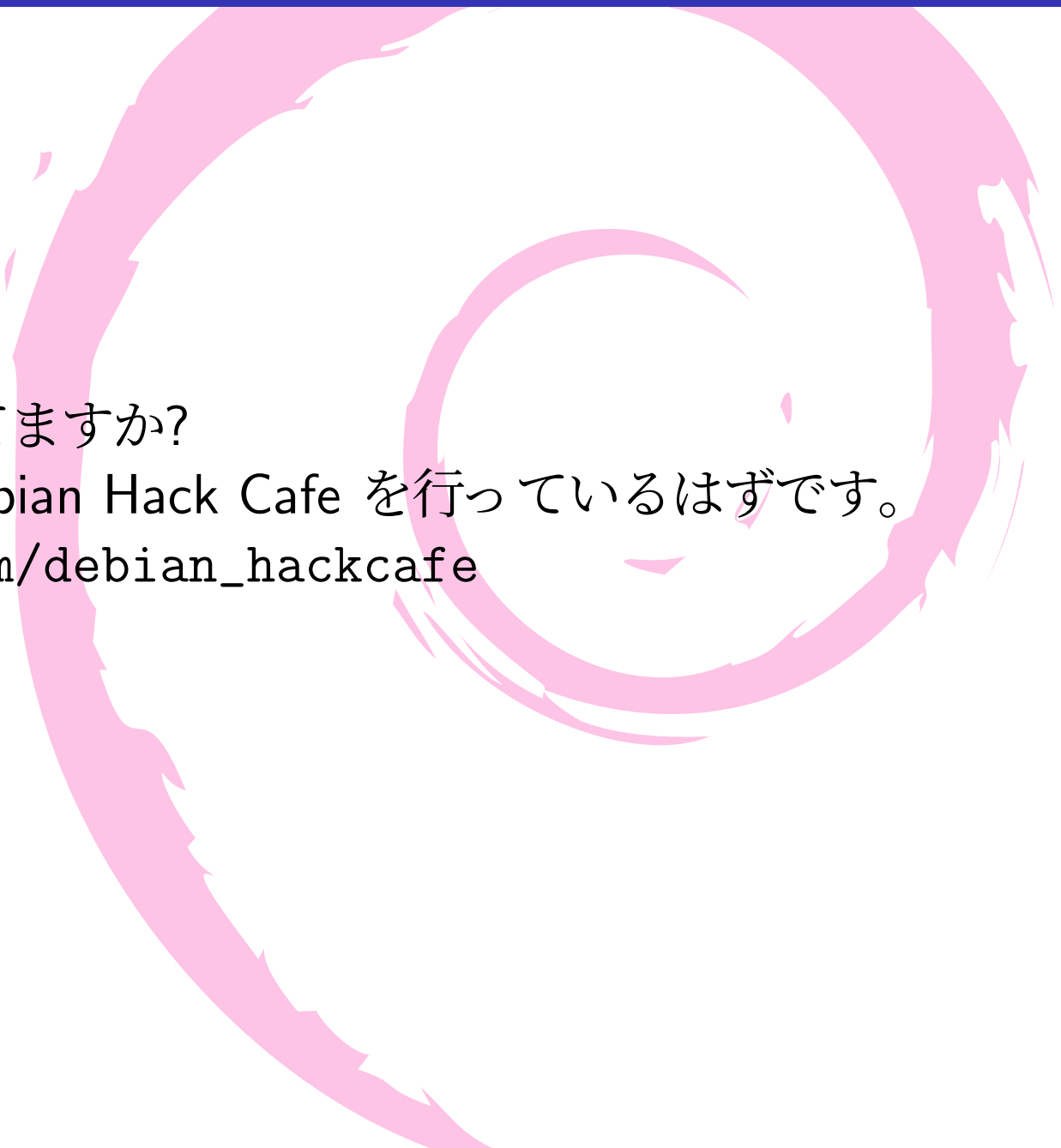
- GNU/Linux とは
- ディストリビューション LT 大会
- 未定
- Debian での Linux カーネルとの付き合い方
- Debian / mozc を Debian パッケージにした

前回の Debian 勉強会

- 西東京/稲城市で開催
- 注意事項
  - 飲食?

- piuparts の使い方
- upstart 再入門
- debtags 入門
- Debian JP Project Leader からのありがたいお言葉

Hack Cafe



最近ハックカフェしてますか?
毎週、新宿近辺で Debian Hack Cafe を行っているはずです。
`http://twitter.com/debian_hackcafe`

事前課題

事前課題

1. 普段使っている Linux ディストリビューションと使ってみたい Linux ディストリビューションは何ですか?
2. 使っている理由、魅力、不満点、使ってみたい理由を教えてください。

和田健

普段使っている Linux ディストリビューション：Debian /GNU Linux 使ってみたい Linux ディストリビューション：特になし使っている理由：パッケージ管理が楽魅力：パッケージ管理が楽不満：特になし

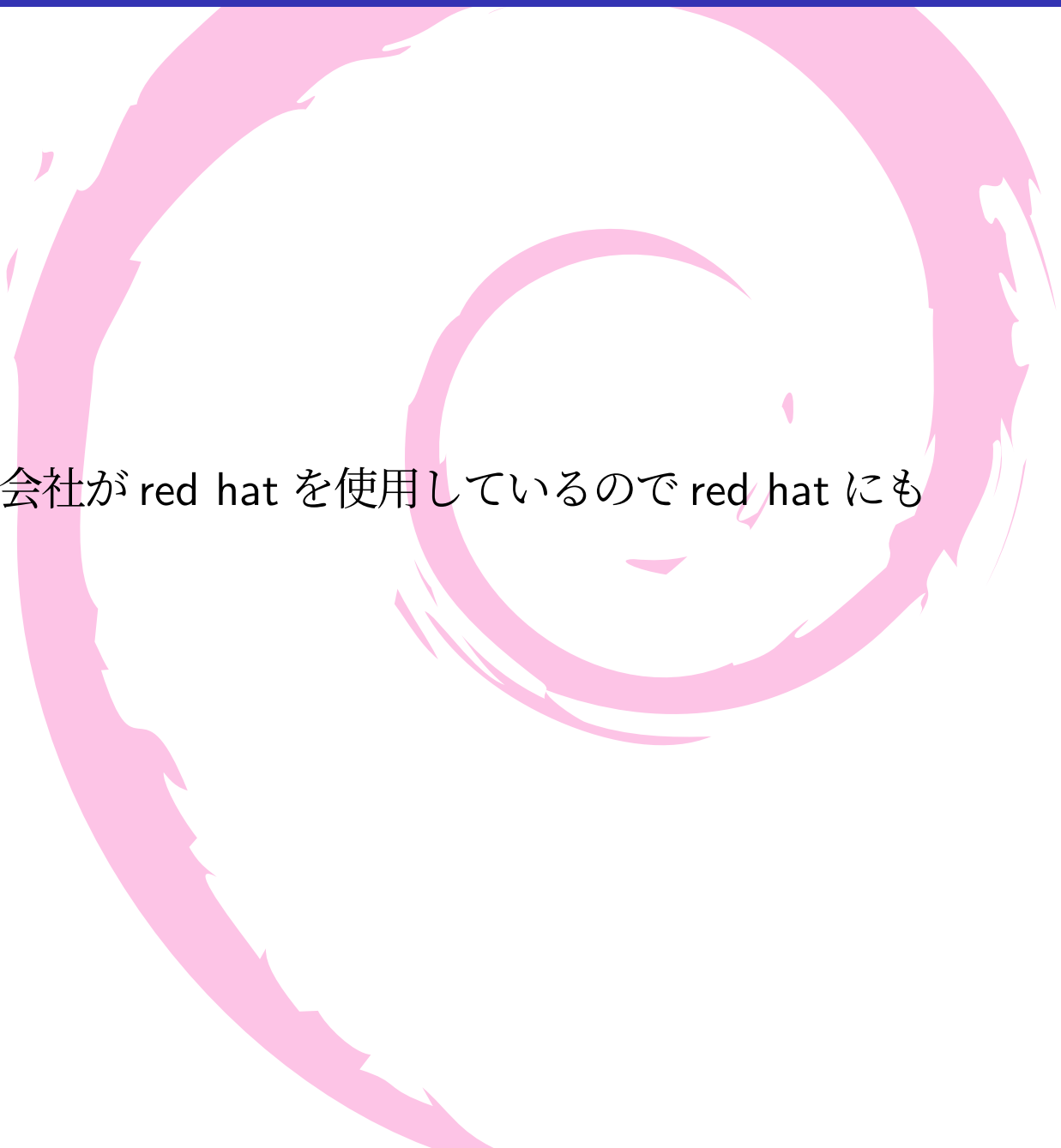
藤沢理聡(risou)

普段使っているディストリビューションは Debian と RedHat。 RedHat は仕事でしか使ってません。 Debian を使ってる理由は大学時代に所属していた研究室のサーバが Debian だったから。他のディストリビューションに触れる機会もあったけれど、結局一番使いなれたものをずっと使っている感じです。使ってみたいディストリビューションは、 Ubuntu や Gentoo など。今まで使う機会がなかったので、一度使ってみたい、という簡単な動機です。

吉野(yy_y_ja_jp)

普段 Debian を使っています．色々な意味で自由だからです．不満はあまり感じてません． Ubuntu さんはより global な Debian にもっと成果を還元していただけるとうれしく思います．

原口秀康



debian 軽い
個人的には ubuntu ですが会社が red hat を使用しているので red hat にも興味があります。

hard2259

普段使っている Linux ディストリビューション

- VineLinux
- Ubuntu
- CentOS

使っている理由

- Vine
  学校の授業で指定されてるから。
  いい意味で枯れた技術といわれたものを多く使っているため、Linux 系を一から勉強するにはもってこいのモノらしいので。
- Ubuntu
  ゼミ室の先輩から勧められた　Windows に近いモノがあるから
- CentOS
  ゼミ室のサーバの OS が CentOS だから使わざるを得ない。
  RedHat 系の構造と似ているから、会社はいったとき役立つかも。※不満を言えるほど使い込んでませんごめんなさい。

使ってみたい Linux ディストリビューション

キタハラ

普段使っている Linux ディストリビューション:debian, Ubuntu　　無料で使える、ライセンス管理で悩まなくて良い、何か使いたいソフトがあると「 apt-get」ですぐ試せる。　でも、環境によって音が出なかったり、グラフィックやプリンタのドライバで悩んだり、 Windows 依存の Web ページにアクセスできなかったり、人に勧めるには少々悩ましい所がある。
使ってみたい Linux ディストリビューション:Ubuntu(ARM), Android(?)
　　モバイル用途で利用している「 Zaurus」の製品寿命がつきているので、その後継として「 NetWalker」か「 JN-DK01」とか「 IS01」の Android 機が欲しいなぁーと、思ってまして・・・。

KIM_TPDN

デスクトップ用途で openSUSE、 VPS で CentOS を使っています。ちなみに自宅鯖は FreeBSD を使用しています。
使っている理由 openSUSE・ YaST がかなり便利。・ KDE が好き。
CentOS・ 初めて使ったディストロだから。一通りのことをこれで学習した。・ VPS や専鯖の OS 選択肢にはたいてい入ってる。
使ってみたいディストリビューション・ Arudius・ Ubuntu Studio

上川純一

いまふと振り返ると普段もっとも使っている Linux ディストリビューションは Android です。携帯電話にプリインストールされているので使い倒しています。不満点はカーネルをいじれないことと、 emacs が動かないことです。

compozz

・普段使っているディストリビューション　Ubuntu・使ってみたいディストリビューション　Fedora
・(Ubuntu を) 使っている理由　ユーザーの多さや安定性がある(と思っている）からです。
・魅力　KNOOPIX の LiveCD を使おうとした時、起動時からディスプレイドライバではまったけれど、Ubuntu は設定変更しなくてもインストール画面から GUI で感動しました。　とにかく Linux の中では導入や運用が楽だと思うので、普段使いにとても役立っています。
・不満点　ディストリビューションの不満はほとんど無いですが、詰まったときに Ubuntu のバグだったりしてがっかりしたことがあります。
・(Fedora を) 使ってみたい理由　使ってみたことがないからです。Gentoo は時間があれば。Debian 勉強会なのにすみません。。。

mnakao

普段使っている Linux ディストリビューション：それは Debian でしょう!
使ってみたい Linux ディストリビューション： openSUSE を仕事で少し使ってみたけど、結構便利そうなので気になってます。
（ Debian を）使っている理由、魅力： apt が便利なのと、一度インストールしたら、再インストールする必要がないこと（ 私はヘタレなので、何度もクリーンインストールしてますが:p）
（ Debian の）不満点：入門者はその文化に慣れるのが大変な所。
（ Suse を）使ってみたい理由： YaST があるから。 LDAP の設定も YaST で出来るので便利そう。

koedoyoshida

普段使っている Linux ディストリビューション
Debian,Ubuntu,Asianux,RHEL 使ってみたい Linux ディストリビューションは何ですか? Gentoo 使っている理由 Debian:パッケージが多い。 stable のバージョンアップが遅いので、サーバ向けに安定している。インストール後の手間が掛からない。 Ubuntu:パッケージが多い。カーネルが新しく、 LiveCD が標準なので新型機で各種デバイスの動作を確認するのに便利。インストールの手間が掛からない。 Asianux:飯の種 RHEL:良くも悪くも標準で参考になる魅力、不満点、使ってみたい理由 Gentoo:パッケージの多さ、手間がかかりそうなところ、マニア向け

moli3

普段使っているディストリビューション CentOS 両方初心者にはなかなか難しいよく躓くことがあります。サーバに向いていると聞いてサーバとして使っていました。 Debian 最近サーバとディスクトップをサーバにしました。 CentOS と違う点で戸惑うときがあります。使ってみたいディストリビューション gentoo よくわからないけど話題なので入れてみたいです。 Debian から chroot でやろうかと思ってます。

村田信人

使っている: Ubuntu + 半年ごとに完成度の高いバージョンがリリースされるというサイクル+ Humanity アイコンテーマ+ Mark Shuttleworth の行動力- Mark Shuttleworth の行動力
使ってみたい: Debian stable, testing, unstable を並列で進めているところ。

akedon

普段使っている Linux ディストリビューションは Debian,CentOS,MIKO GNYO/Linux (Ubuntu Desktop) です。使ってみたい Linux ディストリビューションは SuSE ですね。前記の Linux を使っている理由は Debian は使い始めた当時は圧倒的に apt ツールが他の管理ツール rpm より使いやすかったので、そこから使い続けて慣れているからです。 CentOS は Xen を使い始めた時に一番手軽で確実に動いたからです。 MIKO GNYO/Linux は壁紙が素敵ですよね、落ち着いて作業するには最適です。魅力は Linux 全体に言えることですが、不要な機能を止めたり削除したり、必要最低限に絞り込みやすく、使いやすい様にカスタマイズしやすいという点にあります。不満点は、デバイスドライバ等、ベンダに依存しているものが Windows や MacOS に見劣りするものがあったりする点になります。
SuSE を使ってみたい理由は単純で、未だ試していないからです。

yama1066

普段使っている Linux ディストリビューション： Debian GNU/Linux (sid, squeeze on lenny) 使っている理由：なんとなく。魅力：なにかあったかな? (お不満点：特に無し。
普段使ってはいないが、 apt-get upgrade だけしている Linux ディストリビューション： Ubuntu Linux (maverick) 使っている理由：使ってない。魅力： Debian 上でも chroot で共存可能。不満点：特に無し。
使ってみたい Linux ディストリビューション： Fedora (Rawhide) 使ってみたい理由：人柱は重要だよね。

岩松 信洋

普段使っている Linux ディストリビューション Debian, Gentoo, buildroot, openembedded, オレオレ busybox ベースディストリ
使ってみたい Linux ディストリビューション Arch Linux とか。なんか人気があるようです。使っている理由、魅力、不満点、使ってみたい理由
Debian: 仕事と開発用。 Gentoo: 開発用で主に GCC の HEAD を追っかけるのに使っています。 buildroot: 仕事で。クロスコンパイラとか一式をとりあえず作る場合に利用しています。ディストリというのかは不明。 openembedded: 最近いじりはじめました。 sh サポートとか。オレオレ busybox ベースディストリ:

monoqlo

普段使っているLnuxディストリビューション：Ubuntu 使ってみたいLinuxディストリビューション：しばらくUbuntuでいいです。使っている理由：日本語の情報がたくさんあることと、初めてLinuxを使おうと思った時にWindowsからの移行が楽だと感じたため。魅力：初心者にも勧めやすい不満点：とくになし

なかおけいすけ

使っているディストリビューションは Debian で、長期的に使えそうなのが魅力です。使ってみたいディストリビューションは、 Scientific Linux です。

henrich

普段は debian、その上の KVM で fedora と ubuntu がたまに動きます。使ってみたいのは…CentOS かなぁ。何が良いのか魅力がよく分からんので。後大元の RHEL。 2.1 ぐらいの時にインストールした記憶しかないので。

まえだこうへい

普段使っているディストロは Debian です。他に使ってみたいのは無いです。複数種類のアーキテクチャで同じシステムを使える (サポートしているアーキテクチャが多い) のが一番の理由です。 (いつも言っている理由と同じで代わり映えしませんが)

倉世古 恭平

普段: Fedora 11。適当に動くので。情報が多くて詰まって死ぬことが少ない。変にこだわりをもって使ってる人が多いのでたまに議論がかみ合わなくて困る使ってみたい: OpenSUSE。結構敷居が高いと思っていたら、最近周囲で使っている人が増えたので。

堀本 貴幸 (opentaka)

普段使っている Linux ディストリビューションと使ってみたい Linux ディストリビューションは何ですか? — Gentoo Linux をメインに、 Debian を eee pc で使っている。 LFS(Linux From Scratch) を使いこなしてみたい。—
使っている理由、魅力、不満点、使ってみたい理由を教えてください。—
Gentoo Linux: Portage に多くのパッケージがある。
不満点: bleeding edge すぎて、本当に bleeding する時がある。 (壊れる) —
— Debian(debian eee project): eeepc 向けのディストリビューションで一番簡単にインストールできそうだったから。—

自己紹介

岩松です

- 岩松 信洋 (いわまつ のぶひろ)
- Debian Officiall Developer
- Debian でやっていること
  SH4 アーキテクチャメンテナ / SH4 カーネル 担当 / Buildd / Bluetooth, OpenCV, slim, USB カメラ, gcc, eglibc, etc...
- その他
  Linux Kernel 開発者, ドライバメンテナ、 U-Boot メンテナ、コミッタ

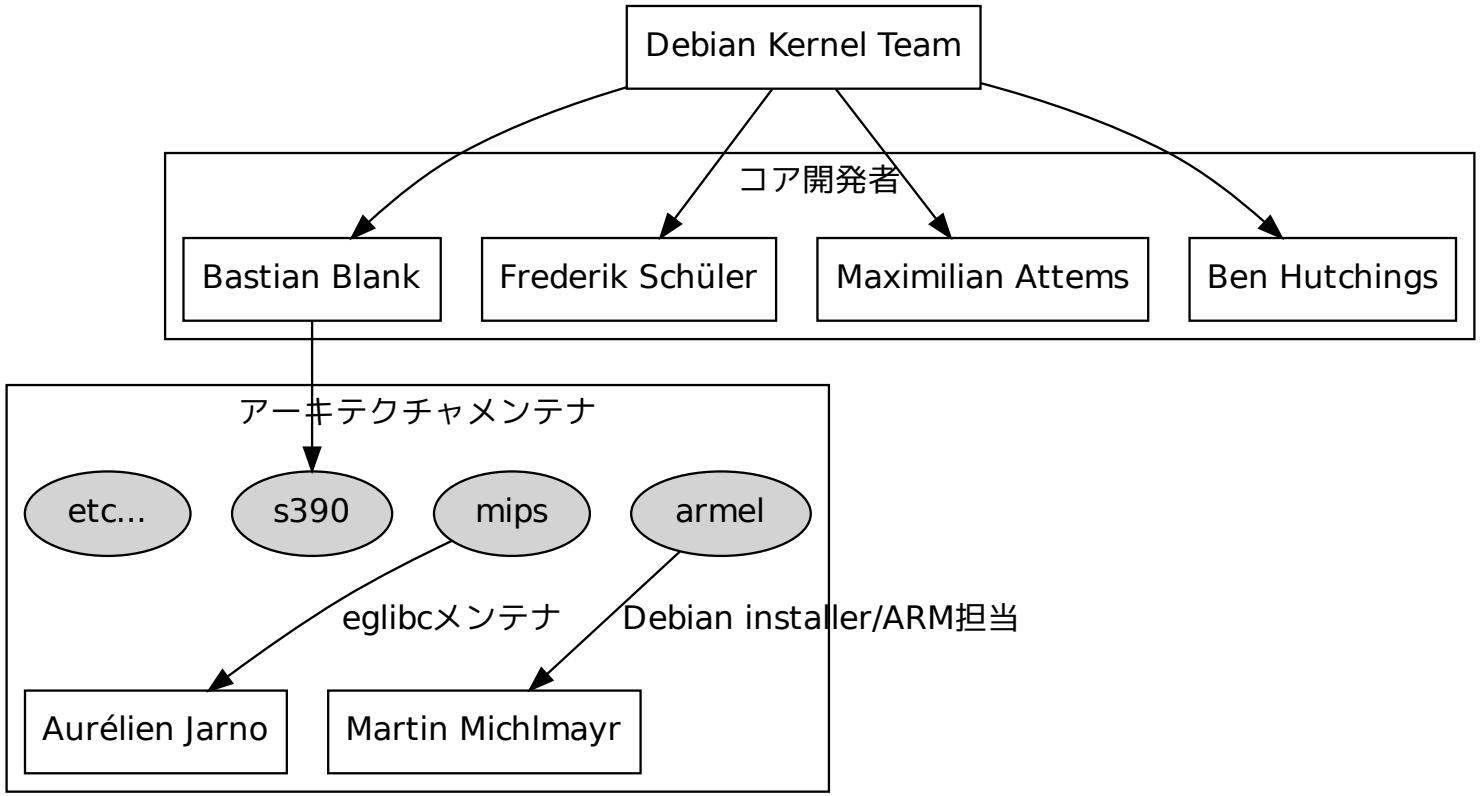
Debian/Linux カーネル

アジェンダ

- Debian Linux カーネルの開発体制、開発状況
- 各カーネルに関するパッケージの説明
- カーネルパッケージの作り方
- よくある質問

Debian Linux カーネルの開発状況

- Debian の Linux カーネルは Debian Kernel Team によって開発およびメンテナンス。
- チームコアメンバは Bastian Blank, Frederik Schul, Maximilian Attems, Ben Hutchings
- 各アーキテクチャメンテナがいる。
  アーキテクチャメンテナは Buildd メンテナや、
  Debian-instller チーム、 eglibc メンテナなど。


Debian Kernel Team
コア開発者
Bastian Blank
Frederik Schüler
Maximilian Attems
Ben Hutchings
アーキテクチャメンテナ
etc...
s390
mips
armel
eglibcメンテナ
Debian installer/ARM担当
Aurélien Jarno
Martin Michlmayr

DebianでのLinuxカーネル開発プロセス

Debian カーネル用語

- Debian カーネル
  Debian からパッケージとしてリリースされているパッケージ。
- LTS
  Long-term Support の略。現在は 2.6.32 が対象。以前は 2.6.27。
- stable カーネル
  The Linux Kernel Archives からダウンロードできるカーネル。現在、 2.6.33.3、 2.6.32.12、 2.6.31.13、 2.6.30.10、 2.6.27.46 の 5 つが存在します。
- Linus/HEAD
  Linus git リポジトリの HEAD。 HEAD はその時の最新を意味します。

カーネルパッケージのバージョン関係

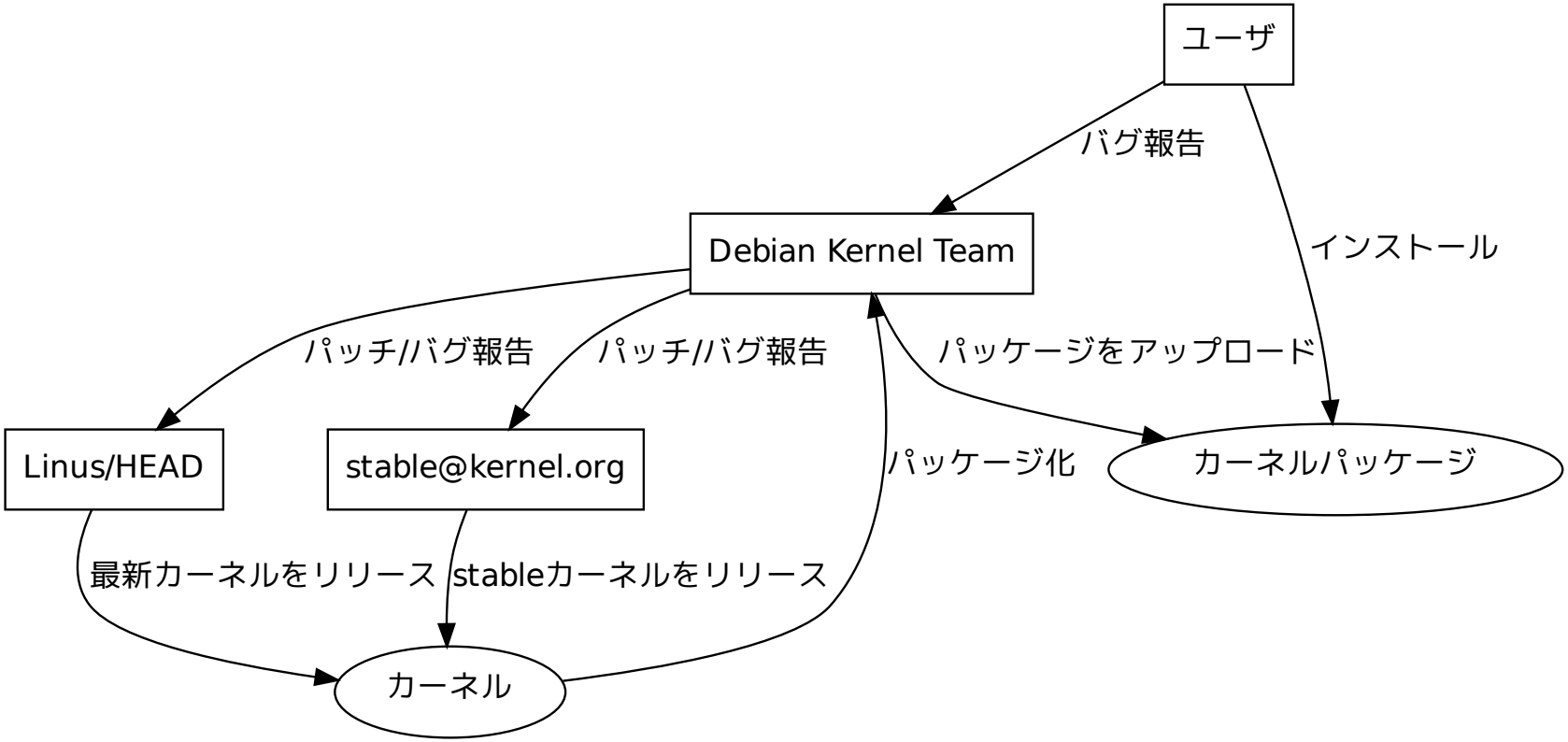
- 開発体制プロセスの変更により、カーネルパッケージのバージョンやアップロードのタイミングが変更された。
- Debian カーネルは stable カーネルリリースベース。最新版は 2.6.32.12。このカーネルをベースに Debian パッケージにした場合、パッケージバージョンは linux-2.6_2.6.32-12 。 stable リリースバージョンを Debian バージョンに置き換えており、 Debian バージョン = Linux カーネルの stable リリースバージョンとなる。
- 新しい stable リリースが出ない限り、 Debian パッケージもアップロードされない(現時点では。)

パッチのバックポート

- バックポートパッチは Debian では直接取り込まない。
- stable カーネル経由。 (stable@kernel.org)

バグレポートとパッチ

- バグがあった場合には、 Debian の BTS を利用できる。
- Upstream(stabel カーネル、場合によってはLinus/HEAD) に転送してくれるかも。
- Debian Kernel Team で作成されたパッチは積極的に Linus カーネルに取り込む。
- 取り込まれない場合、 Debian specific パッチとして管理される。

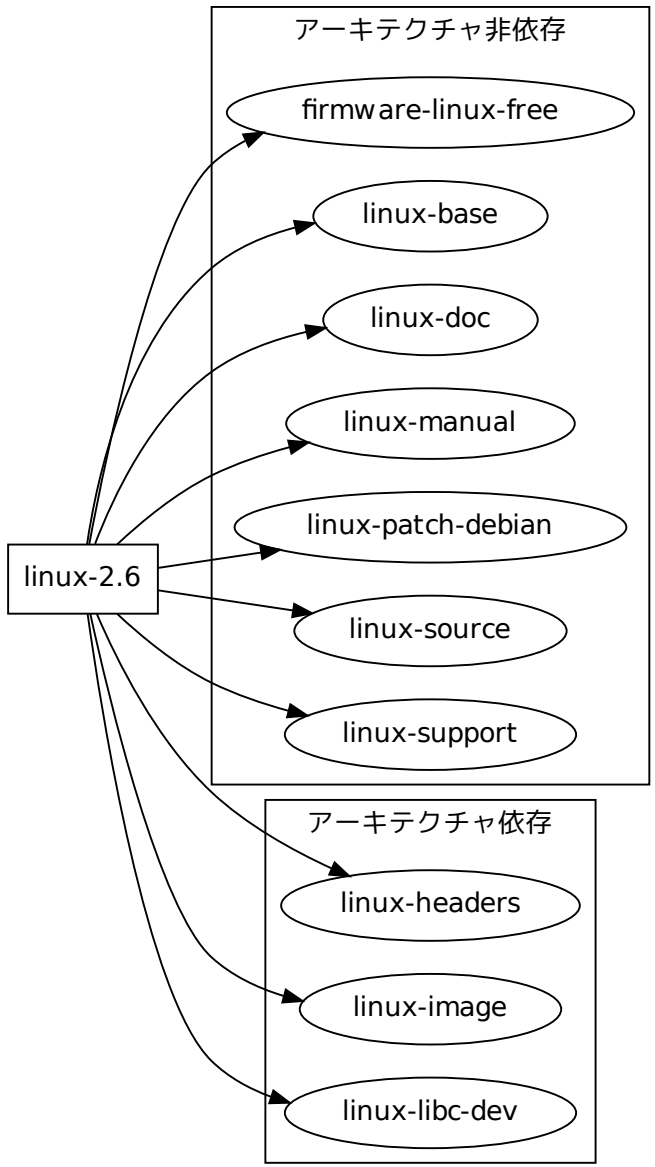

ユーザ
バグ報告
インストール
Debian Kernel Team
パッチ/バグ報告
パッチ/バグ報告
パッケージをアップロード
Linus/HEAD
stable@kernel.org
パッケージ化
カーネルパッケージ
最新カーネルをリリース
stableカーネルをリリース
カーネル

Debian Kernel Team によってメンテナンスされている主要なパッケージ

Debian Kernel Team では Linux カーネルに関するいくつかのパッケージをメンテナンスしています。ここでは、主要なパッケージと関係について説明します。

linux-2.6 パッケージ

- Linux カーネルの主要なパッケージ
- 各アーキテクチャ向けのカーネル、ヘッダファイル、libc 向けヘッダファイル、ドキュメント等のパッケージが生成される。

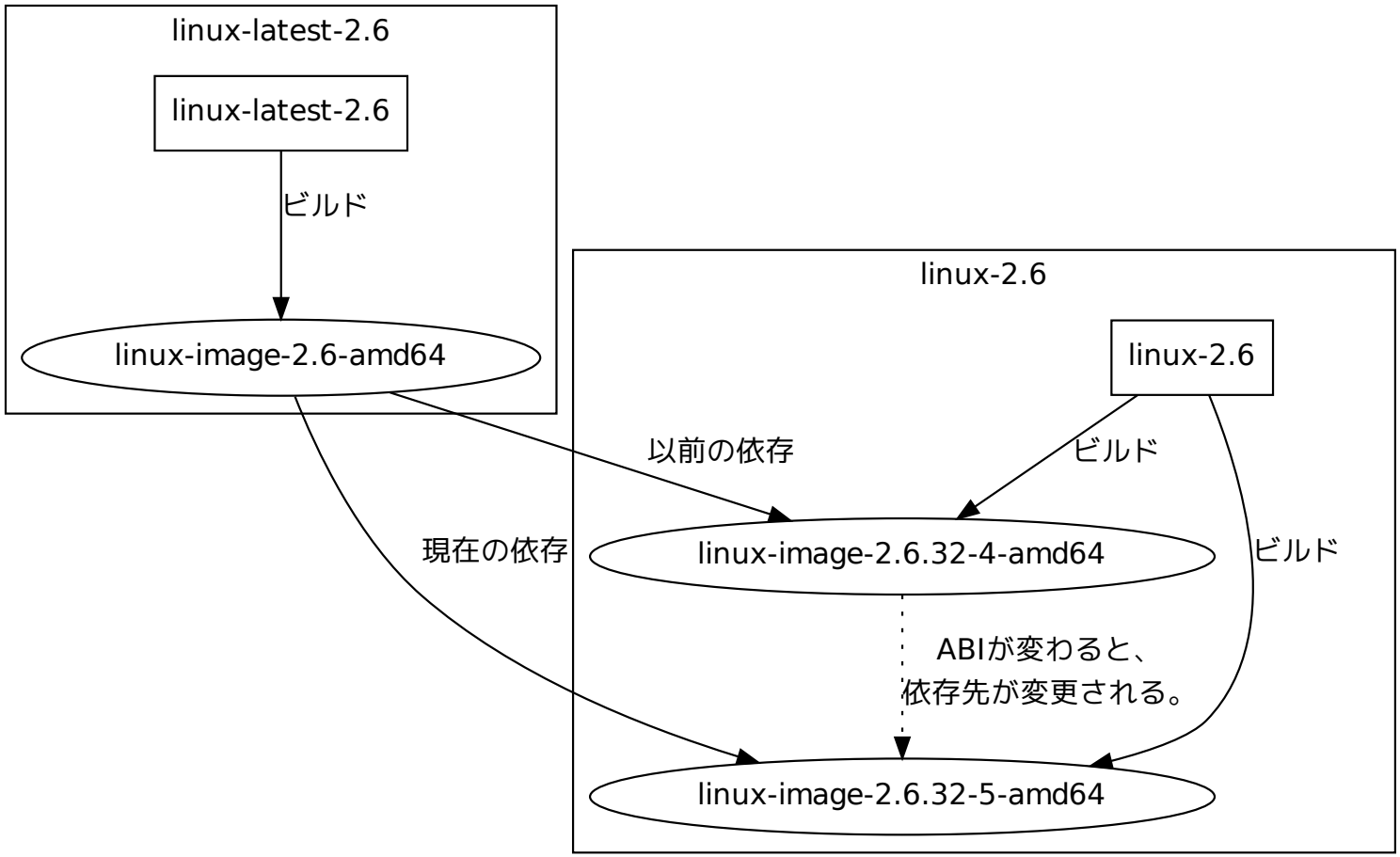


カーネルコンフィグ

- 基本 config，アーキテクチャ用 config，flavour 用 config で構成されている。
- これらがパッケージビルド時にひとつになって、カーネルコンフィグが行われる。
- 優先順位は 基本 config ＜アーキテクチャ用 config ＜flavour 用 config
- コンフィグを各ファイルに自動的で分割するプログラムは存在しないので手作業で更新する。

linux-latest-2.6 パッケージ

Debian カーネルの最新 ABI を追従するためのメタパッケージを提供する。



amd64の場合

ABI のチェック

- syscall などの ABI が変更された場合のチェックなど。
- linux-2.6/debian/bin/buildcheck.py カーネルパッケージビルド時に実行。
- 大きな変更がない場合は更新しない場合もある。

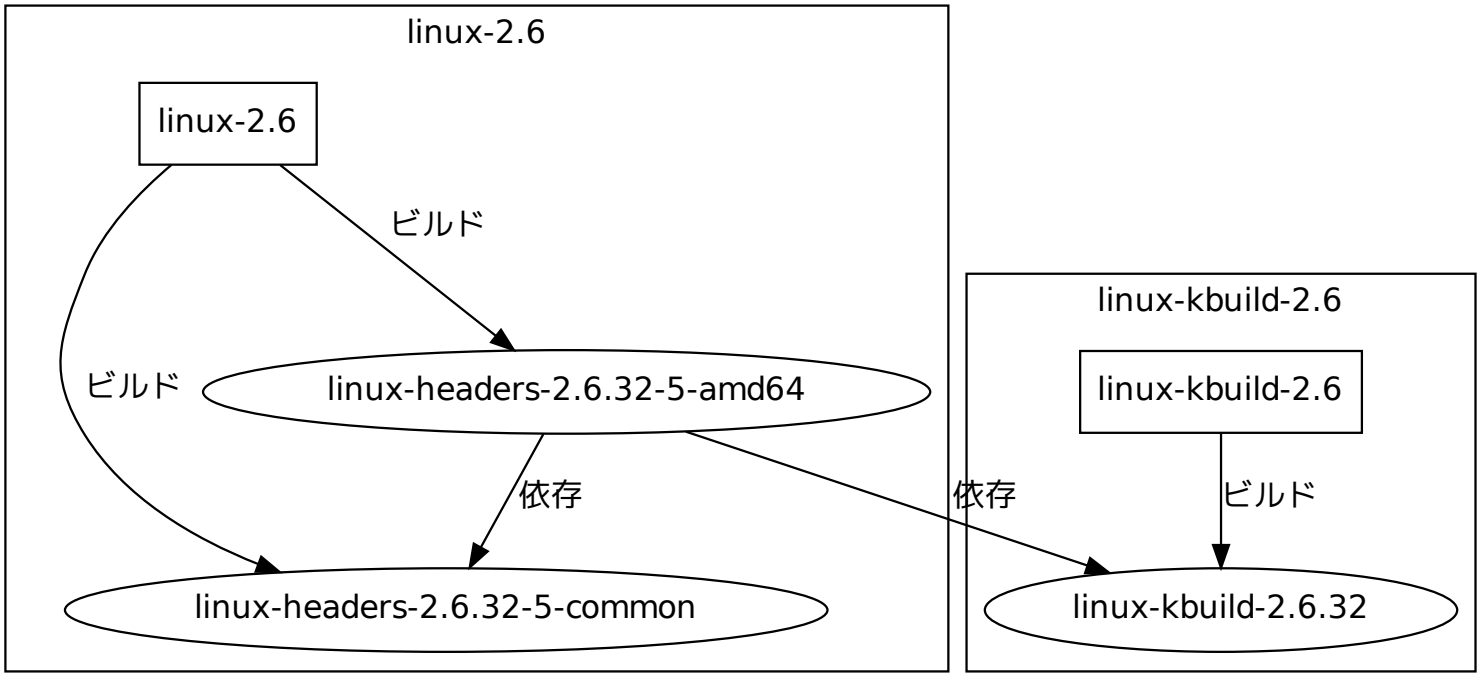
```
--省略--
make[3]: Leaving directory `/home/mattems/src/linux-2.6-2.6.32
python debian/bin/buildcheck.py debian/build/build_amd64_none_a
ABI has changed!  Refusing to continue.

Added symbols:
dev_attr_usbip_debug    module: drivers/staging/usbip/usbip_cor
getboottime             module: vmlinux, version: 0x0619ca8a, e
monotonic_to_bootbased  module: vmlinux, version: 0xdb274e52, e
--省略--
```

linux-kbuild-2.6

- `linux-kbuild-2.6` はカーネルドライバ構築をサポートするためのスクリプトを持つ。
- stable カーネルのソースコードから kbuild を行うために必要な部分を抽出して作られる。

linux-kbuild-2.6



amd64の場合

まとめ

- Debian のカーネルメンテナンスはチーム制。
- カーネルに関係するパッケージメンテナやアーキテクチャメンテナによってメンテナンスされている。
- パッケージのアップデートは stabel リリースベース。
- 更新を用意にするためにメタパッケージを使っている。
- ABI のチェックなどもしてけっこう真面目。

今時の Debian カーネルのビルド方法

みなさん、カーネルのコンパイルしていますか?

今時の Debian カーネルのビルド方法

- 大抵のユーザは Debian で提供されているカーネルを使う。
- たまにビルドしたくなるときがある。
  - 何か修正するためのパッチを適用したい。
  - オレオレパッチを適用したカーネルを使いたい。
  - プリエンプションモデルが気に入らない。
    `CONFIG_PREEMPT_XXX` の変更
  - Timer frequency を変更したい。
    `CONFIG_HZ_XXX` の変更
  - 毎朝、自分のマシンで使うカーネルをビルドしないと気が済まない。

日頃からの鍛錬がものを言う

このような事から日頃からカーネルのビルドを行って置くことが重要です。しかし、 Debian ではいくつかのカーネル構築方法があります。これらを一つずつみてみましょう。

Debian カーネルビルド方法

- Debian オフィシャルカーネルをリビルドする
- Debian カーネルにパッチを適用してビルドする。
- Debian オフィシャルカーネルをリビルドする (その 2)。
- リリースされたカーネルを Debian パッケージにする。
- git/HEAD を Debian パッケージにする。

Debian オフィシャルカーネルをリビルドする

- カーネルソースパッケージを持ってきて debuild
  これではすべての flavour をビルドしてしまう。
- flavour： amd64, vserver , xen, openvz, etc..
- 指定した flavour だけをビルドする方法。

- Linux-2.6 ソースコードをダウンロードする。
  まず、 linux-2.6 ソースパッケージをダウンロードします。ダウンロードできたら、展開されたディレクトリに移動します。

```
$ apt-get source linux-2.6
$ cd linux-2.6-2.6.32
```

- linux-2.6 パッケージのビルドに必要なパッケージをインストールする。
  パッケージのビルドに必要なパッケージをインストールするには build-dep オプションを使います。

```
$ sudo apt-get build-dep linux-2.6
```

- Debian カーネル向けのパッチを適用する。

```
$ make -f debian/rules clean
$ make -f debian/rules source-all
```

debian/rules source-all では、全てのアーキテクチャ向けにパッチを適用してしまうので、特定のアーキテクチャのパッチを適用したい場合には以下のように実行します。

```
$ make -f debian/rules.gen source_amd64
```

- 利用したい flavour で初期化する。
amd64 アーキテクチャの amd64 flavour で初期化したい場合には以下のように実行します。

```
$ fakeroot make -f debian/rules.gen setup_amd64_none_amd64
```

- カーネルコンフィグを変更する。
  カーネルコンフィグを変更したい場合には、
  debian/build/build_amd64_none_amd64 ディレクトリ
  移動して、カーネルコンフィグを行います。コンフィグ終
  了後は元のディレクトリに戻る必要があります。

```
$ cd debian/build/build_amd64_none_amd64
$ make menuconfig
$ cd ../../..
```

- パッケージをビルドする。
  debuild / dpkg-buildpackage コマンドは利用せず、
  debian/rules のターゲットを指定してパッケージをビルド
  します。

```
$ fakeroot make -f debian/rules.gen binary-arch_amd64_none
```

Debian カーネルにパッチを適用して利用する

- Debian カーネルをベースに自分が作ったパッチを当てて利用する。
- 自分のパッチを Debian のカーネルパッチ機構に取り込む必要がある。

Debian カーネルパッチ機構

- debian/patches/bugfix
  重要なバグ修正用パッチを格納する。
- debian/patches/debian
  Debian 専用パッチを格納する。
- debian/patches/features
  まだ upstream にマージされていないパッチを格納する。
- debian/patches/series
  パッチを管理するファイルを格納しているディレクトリ。
  Debian バージョン毎にファイルがあります。

さらにアーキテクチャ毎にパッチが分かれる。

自分が作成したパッチを適用する例

自分が作成したパッチ (oreore.patch) を適用する例。

- カーネルソースコードを取得する。
  展開後に、ディレクトリに移動します。

```
$ apt-get source linux-2.6
$ cd linux-2.6-2.6.32
```

- Changelog を更新する。
  Debian パッケージの Changelog を更新する場合、 dch コマンドを使う。

```
$ dch --local +test -D UNRELEASED
```

  Liux カーネルパッケージのバージョンが 2.6.32-12 の場合には 2.6.32-12+test1 になる。
- パッチをディレクトリにコピーする。
  パッチを debian/patches ディレクトリ以下にコピーします。

```
$ cp ~/oreore.patch debian/patches/bugfix/
```

- コピーしたパッチを有効にする。
  コピーしたパッチを有効にするには、debian/patches/series/ディレクトリにパッチを適用したいDebianバージョンのファイルを作成し、パッチのパスを指定します。

```
$ echo ‘‘+ bugfix/oreore.patch’’ >> debian/patches/series/
```

- ./debian/bin/gencontrol.pyを実行する。
  ./debian/bin/gencontrol.pyを実行し、ビルド用のスクリプトや設定ファイルを新しいDebianバージョン向けに更新します。

```
$ ./debian/bin/gencontrol.py
```

- 一度初期化し、パッチを適用する。

```
$ make -f debian/rules clean
$ make -f debian/rules.gen source_amd64
```

- パッケージをビルドする。
  amd64 の amd64 flavour をビルドする場合には以下のように実行します。

```
$ fakeroot make -f debian/rules.gen binary-arch_amd64_none
```

  エラーがなければ、パッチが有効になったカーネルパッケージがビルドされます。

Debian オフィシャルカーネルをリビルドする

- Debian カーネルを再ビルドする方法はもうひとつの方法。
- `linux-source-2.6.XX` パッケージを利用する。

linux-source-2.6.XX パッケージから再ビルドする。

- Debian が提供しているカーネルのビルドに必要なパッケージをインストールする。

```
$ sudo apt-get build-dep linux-source-2.6.32
```

- Debian のカーネルソースをインストールする。

```
$ sudo apt-get install linux-source-2.6.32
```

- make-kpkg コマンドを使ってカーネルパッケージをビルドする。

```
$ fakeroot make-kpkg --revision=test00 kernel_image kernel
```

- これでは Debian パッケージにはパッチが適用されていない状態。。
- linux-patch-debian-XXXX パッケージからパッチを適用する必要がある。
- パッチを当てた後に再度ビルド。

```
$ sudo apt-get install linux-patch-debian-2.6.32
$ /usr/src/kernel-patches/all/2.6.32/apply/debian -a x86_64 -f
```

-a でアーキテクチャ、-f で flavour を指定します。

リリースされたカーネルを debian パッケージにする

- Linus や stable チームによってリリースされた Linux カーネルを Debian パッケージにする。
- `kernel-package` パッケージを使うのが Debian 流です。

- kernel-package パッケージと fakeroot パッケージをインストールします。

```
$ sudo apt-get install kernel-package fakeroot
```

- ソースをダウンロードし、展開します。
- カーネルコンフィグを実行します。

```
$ make menuconfig
```

- make-kpkg コマンドを使ってカーネルパッケージを構築する。
  make-kpkg コマンドにはいくつかのオプションがありますが、よく利用するオプションについて説明します。
  - kernel_image
    カーネルイメージパッケージビルドを指定します。
  - kernel_headers
    カーネルヘッダビルドを指定します。
  - –revision
    リビジョンを指定します。これは Debian バージョンに付加されます。
  - –append_to_version
    カーネルバージョンを追加します。これはパッケージ名に付加されます。

- –added_modules
  Debian パッケージになっているカーネルモジュールをビルドします。
- –added_patches
  Debian パッケージなっているカーネルパッチを有効にしてビルドします。
- –initrd
  initrd イメージをビルドする際に必要です。 initrd イメージはパッケージインストール時に作成するように仕様が変わっています。

例えば、リビジョンを test12345、バージョンに append67890 指定し、カーネルパッケージとカーネルヘッダパッケージをビルドする場合には以下のように実行します。リビジョンの前に.(ピリオド) をつけているのは、2.6.33.3 の場合には 2.6.33.3.append67890 となるようにするため。

```
$ fakeroot make-kpkg --revision=.test12345 \
  --append-to-version=append67890 kernel_image
```

作成されるパッケージ名は以下のようになり、--append-to-version と--revision は以下のように配置されます。

```
linux-image-(kernel-version)(--append-to-version)_(--revision)_
```

- ビルドが終わるとパッケージがビルドされているので、インストールする。

```
$ sudo dpkg -i \
  ../linux-image-2.6.33.3.append67890_testrev12345_amd64.d
```

git/HEAD を Debian パッケージにする

- リリースされる度にソースダウンロードするの?
  そんなことが許されるのは小学生まで。
- いまどきは git リポジトリを利用する。みんな持ってるよね?
- kernel-package がサポート。

- Linux git リポジトリをコピーする。
  Linux カーネルの git リポジトリがない場合には git clone コマンドで取得します。

```
$ git clone \
  git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/l
```

  linux-2.6 ディレクトリができるので移動します。

```
$ cd linux-2.6
```

- リポジトリのアップデートを行う。
  普段から git リポジトリを使っていじっている人はアップデートしましょう。

```
$ git pull
```

- `make-kpkg clean` を実行する。
  `make-kpkg clean` を実行し、一度初期化をします。

```
$ make-kpkg clean
```

- カーネルパッケージをビルドする。
  make-kpkg を使ってカーネルパッケージをビルドします。
  ビルドすると、 Makefile からバージョンを抽出し、パッケージバージョンをつけてくれます。git log --pretty=format:%h -1 はチェックアウトしている HEAD の短縮されたハッシュ値を取得し、--revision オプションに渡しています。これにより、どのコミットから作成したカーネルイメージなのかわかるようになります。

```
$ fakeroot make-kpkg --revision=1+‘git log --pretty=format
   --initrd kernel_image
-- 省略 --
$ ls ../
linux-2.6  linux-image-2.6.34-rc7_1+be83567_amd64.deb
```

よくある質問

最新カーネル向けパッケージを lenny 上で作れません

最新カーネル向けパッケージを lenny 上で作れません

kernel-package パッケージが古いので lenny ではビルドできません。

最新カーネル向けパッケージを lenny 上で作れません

testing/unstable にある kenrel-package パッケージを lenny にインストールすることによって対応できます。 kernel-package に依存しているパッケージも lenny 内から持ってくることができるので、特にシステムが壊れるということはないと思われます。

initrd イメージ が作られません。

initrd イメージ が作られません。

grub のメニューを変更して、 initrd を使わないようにしましょう。

initrd イメージ が作られません。

というのは半分冗談で、 kenrel-pakcage 12.012 以降から initrd を作らない仕様に変更されました。 make-kpkg コマンドを使って initrd を含めたカーネルイメージを作成するには、以下を実行する必要があります。

```
$ sudo mkdir -p /etc/kernel/postinst.d/
$ sudo cp
 /usr/share/doc/kernel-package/examples/etc/kernel/postinst.d/i
 /etc/kernel/postinst.d/
$ fakeroot make-kpkg --revision=1 --initrd kernel_image
```

実行した後に再度カーネルパッケージを作ると、インストール時に initrd イメージを構築します。

-j オプションを使ってカーネルパッケージをビルドしたいのですが

-j オプションを使ってカーネルパッケージをビルドしたいのですが

make-kpkg コマンドの DEBIAN_KERNEL_JOBS 変数を使いましょう。例えば、-j8 相当は以下のように実行します。

```
$ make-kpkg --revision=test00 kernel_image DEBIAN_KERNEL_JOBS=8
```

最新カーネル向けのlinux-kbuild-2.6を作りたいのですが

最新カーネル向けの linux-kbuild-2.6 を作りたいのですが

- linux-kbuild-2.6 パッケージがない場合がある。experimental にある場合のみ。
- ぶっちゃけ、メンテナの手抜き
- 自分で用意する必要がある
  `http://wiki.debian.org/HowToRebuildAnOfficialDebianKernelPackage#Thestoryoflinux-kbuild-2.6` ちなみに 2.6.33 以降のパッケージを作成する場合には、 バグ (#573176) があります。

最新のカーネルを使いたいのだけど、パッケージにするのがめんどいです。

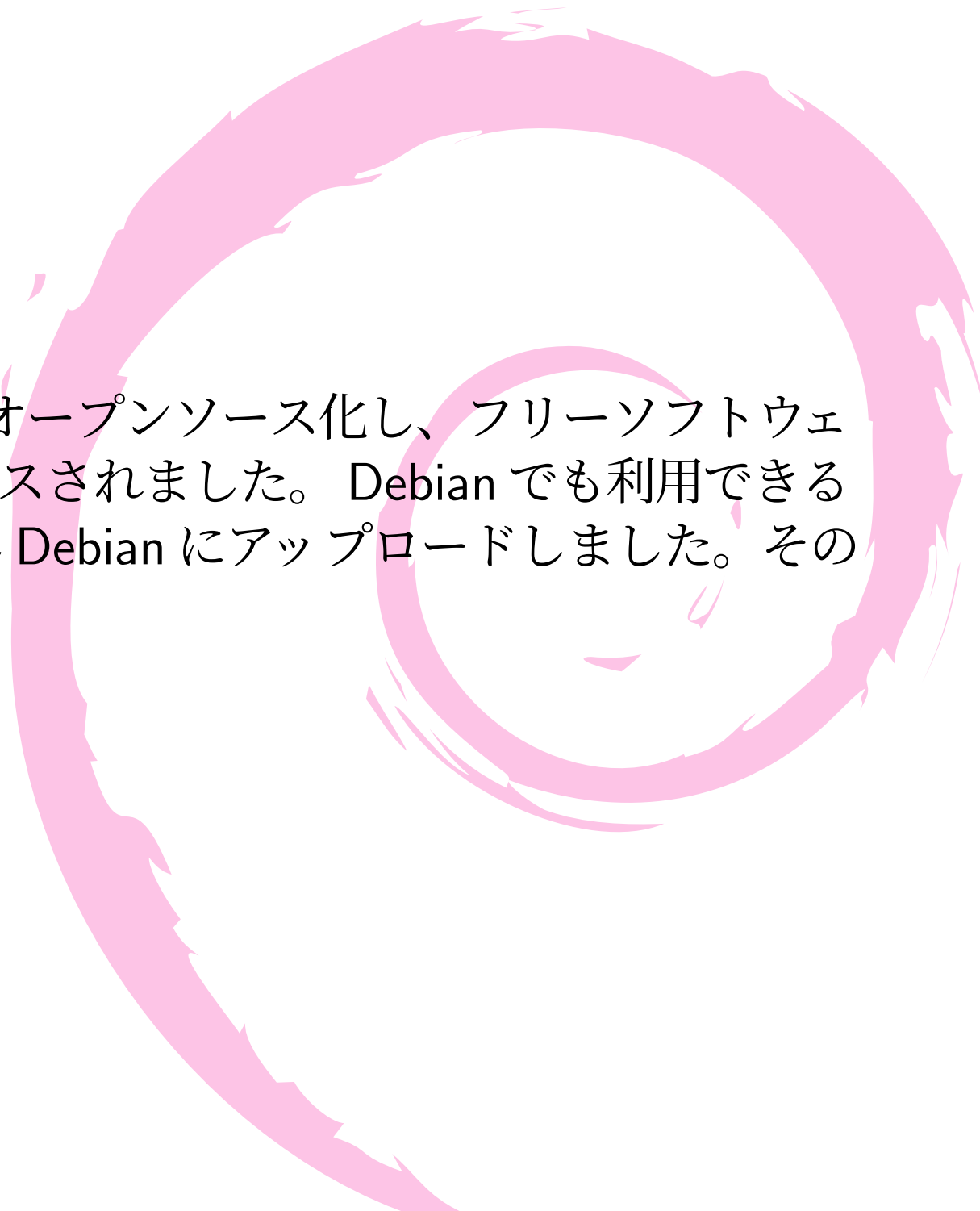
最新のカーネルを使いたいのだけど、パッケージにするのがめんどいです。

http://kernel-archive.buildserver.net で提供されていますが、現在サーバダウン中です。

まとめ

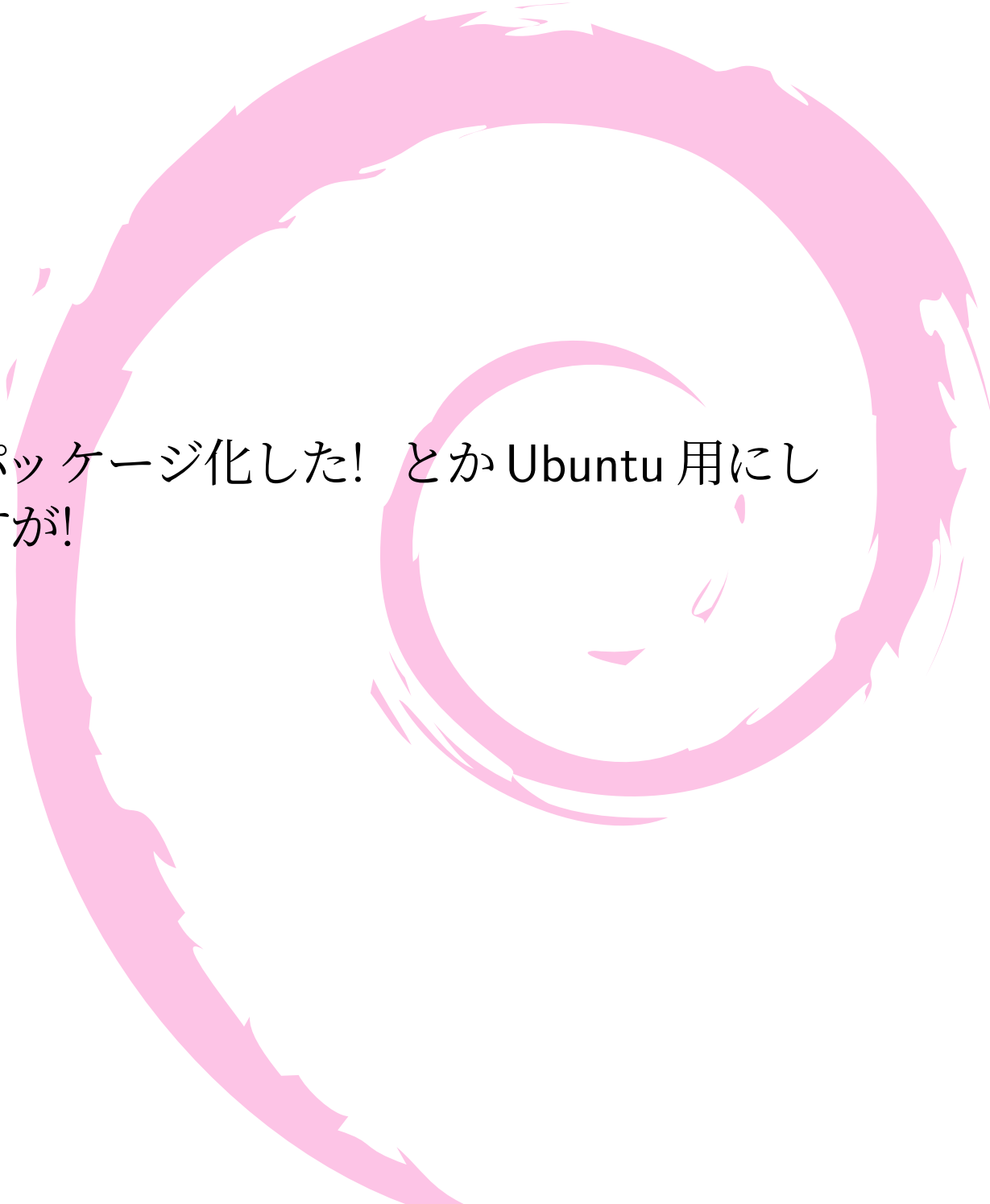
- Debian のカーネル構築方法はいくつかある。
- 自分のライフスタイルの合わせたカーネルコンパイルを。
- 知らないとハメられるところが多い。
- 技術が分散しているので、ハックできる余地あり。
- 自分でカーネルぐらいコンパイルしましょう。
- git いいよ、 git。

mozc を Debian パッケージ化してみた



Google 日本語入力がオープンソース化し、フリーソフトウェア mozc としてリリースされました。 Debian でも利用できるよう、パッケージ化し Debian にアップロードしました。その流れを説明します。

所々に煽り文句が入っていますが、スルーカを使いましょう。
ネタなので過剰反応しないように!



twitter では Debian パッケージ化した! とか Ubuntu 用にした! とか流れていますが!

- debian/rules ファイルが既にあるので、パッケージ化は簡単。
  ほとんどはこれを使って debuild しただけのもの。
- 使えればいいとかじゃなくて!
- なんで ITP しないかね。
- 中途半端な Debian パッケージを駆逐するために立ち上がった! <− いまここ。

簡単なパッケージアップロードまでの流れ

1. 気になるソフトウェアを見つけました!
2. ITP / RFP / パッケージ有無のチェック
3. ソースコードをダウンロード
4. ライセンスをチェック
5. ソースコードをチェック
6. ITP する。
7. パッケージ化
   デバッグ、 piuparts, pbuilder, lintian, debhelper etc...
8. Debian へのアップロードアップロード。
9. FTP master によるチェック。
10. Debian へのインストール。

修正した内容

- まず、バージョンが違うよ。
  debian/changelog そのままつかわないで。
  今のバージョンは 0.11 だよ。ソース見ましょう。

- パッケージの説明が適当すぎるよ。
  修正した。

- ビルド時に gyp をダウンロードするよ!
  gyp を自動ダウンロードしないように修正。
  Debian パッケージを使うように修正。
  gyp Debian パッケージちょっと古いので、更新メールをしたら、早速アップロードされた。

- protobuf もビルドするよ!
  protobuf をビルドしないように修正。
  ソースコードからばっさり削除。 Debian パッケージによってインストールされた protobuf を使うように修正。

- rx のライセンスちゃんと書着ましょう。
  rx は apache license だよ。 copyright ファイルのアップデート。
  rx は Debian パッケージになってないので、後日 パッケージアップロード予定。

- gtest も Debian パッケージがあるよ!
  ソースコードから削除。インストールされた gtest を使うように修正。

- ibus 向けのアイコンがない!
  そのまま使うと mozc のアイコンでないよ!
  バグ報告した。 mozc BTS 1 ゲットずさー。

- ibus の ヘッダファイル、ライブラリファイルのパスがハードコーディングだよ。
  pkg-config を使うように修正。 mozc BTS にパッチを送った。

- base/scoped_ptr.h のライセンスは 修正 BSD ライセンスじゃ ないよ。
  License Boost Software License - Version 1.0 です。
  debian/copyright に書いた。

というわけで、さっきアップロードしました。

今回得た知識

- protobuf の使い方
- gtest の使い方
- gyp の使い方
- iBus の簡単な動き

Debian パッケージ化する場合には、 Debian に持っていくよう心がけてください。

次回の勉強会

- 2010 年 6 月 19 日: 未定
- 2010 年 6 月 26 日: OSC 2010 北海道