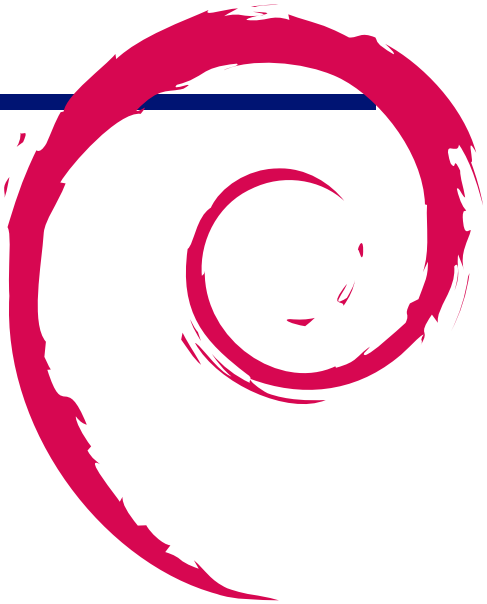
第 64 回 東京エリア Debian 勉強会資料

月刊 Deb専

日本唯一のDebian専門月刊誌

2010 年 5 月 17 日

特集1: DebianでのLinuxカーネルとの付き合



1 Introduction

上川 純一

今月の Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか?

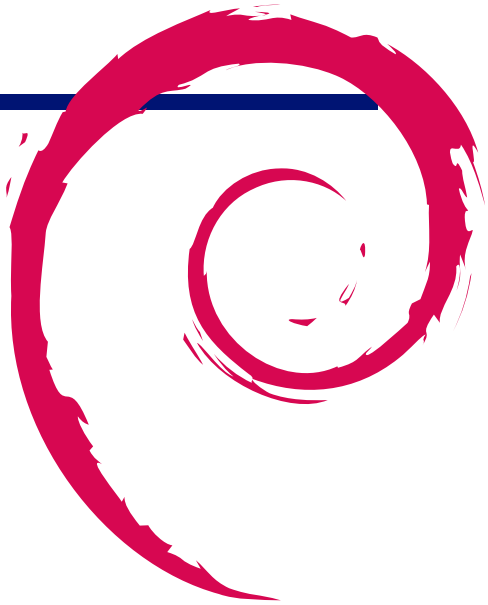
Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での「開発に関する情報」を整理してまとめ、アップデートする。
- 場の提供。
  - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
  - Debian のためになることを語る場を提供する。
  - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、「場」としての空間を提供するのが目的です。

デビアン勉強会

目次



2 事前課題

上川 純一

今回の事前課題は以下です:

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? 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。

この課題に対して提出いただいた内容は以下です。

2.1 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

2.2 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。

2.3 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．

2.4 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。

2.5 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
・ Gentoo
使ってみたい理由
・ Gentoo
色々難しいといわれているから挑戦してみたい。

2.6 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・・・。

2.7 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

2.8 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。

2.9 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。。。

2.10 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。

2.11 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

2.12 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。

2.13 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。

2.14 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。

2.15 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。

2.16 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:

2.17 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

2.18 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。

2.19 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。

2.20 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)

2.21 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。

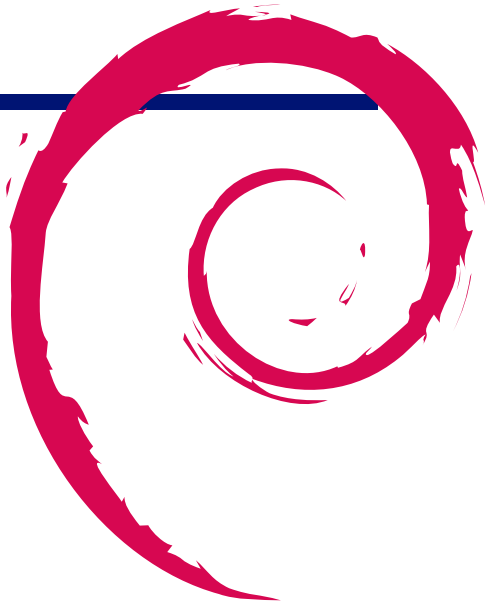
2.22 堀本 貴幸 (opentaka)

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。—



3 最近の Debian 関連のミーティング報告

岩松信洋

3.1 東京エリア Debian 勉強会 63 回目報告

第 63 回東京エリア Debian 勉強会 を行いました。場所は稲城。

稲城といっても最寄り駅は京王稲城ではなく、若葉台というところ。

参加者は原口秀康さん、キタハラさん、やまねさん、山本さん、 koedoyoshida さん、鈴木崇文さん、藤沢理聡さん、村田信人さん、あけどさん、 Hirotaka Kawata さん、 opentaka さん、松澤二郎さん、前田さん、荒木さん、岩松 の 16 名でした。

まず私は piuparts の使い方と内部の動きについて話しました。 piuparts は開発者寄りのツールです。使うとテストが楽ちんになるので、みなさん使いましょう。

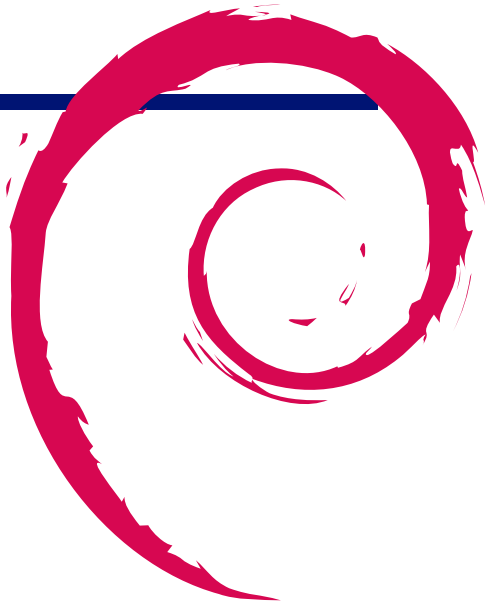
次に前田さんが upstart について話しました。現状では upstart に移行するメリットがないことがわかりました。upstart-native さくっとしたいものですね。

そして、山根さんが debtags について話ました。 debtags はあまり知られてないので、好評だったようです。現状ではタグの付け方などに問題があるのでこれらを解決する必要がありそうです。

これらが終わったあとに我らがが荒木新会長からありがたいお言葉をいただきました。自分も前会長としてやりたかったことを話したのですが、よくよく考えてみるとなんか違っていて、自分は Debian JP に入ったのなら Debian JP でできることを目標にして活動してほしいというのがあったのでした。

案の定時間が余ったので、ネタ帳にに残っていた Debian/SH4 進捗報告会と、 Emdebian の現状と Emdebian/Grip について話しました。

勉強会後の宴会は牛角で。 1 時間半ほど待たされるプレイをしていました。肉はうまかったです。



4 Debian での Linux カーネルとの付き合い方

岩松

4.1 はじめに

ここ数年で Debian の Linux カーネル開発体制や Linux カーネルに関するツールの使い方が変わってきました。Debian JP の ML でもたまに Linux カーネルに関してハマっている方おられるようです。昔と違って最近のユーザは Debian から提供されているカーネルを使う人が多いようで、ネット上でも情報がまとまっていません。今回は、 Debian Linux カーネル開発の背景と、今時のカーネルコンパイル方法について簡単にまとめました。

4.2 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されています。もちろん Debian では Linux カーネルも Debian パッケージとして提供されており、容易に利用可能です。チームコアメンバは Bastian Blank, Frederik Schul, Maximilian Attems, Ben Hutchings で、彼らがが中心になってメンテナンスしています。各々はもちろんカーネル開発者です。彼らだけでは全アーキテクチャの面倒を見きれないので、各アーキテクチャメンテナとともにカーネルパッケージをメンテナンスしています。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、カーネルに関係するパッケージをメンテナンスしている約 20 名の開発者によって構成されます。

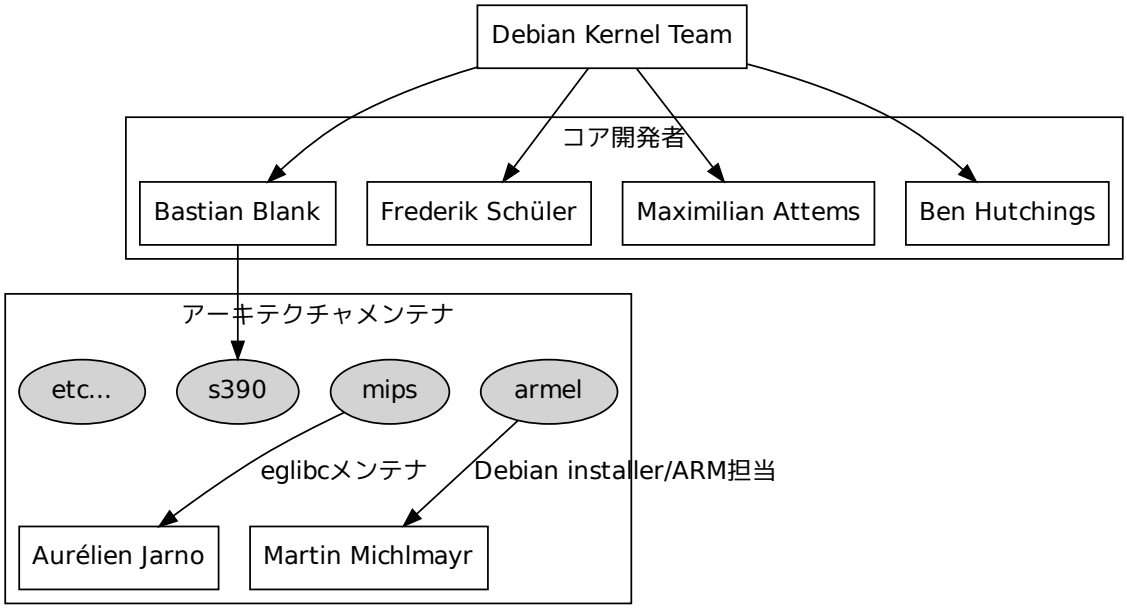


図 1　Debian Linux カーネルチーム構成図

4.3　Debian での Linux カーネル開発プロセス

去年の 6 月頃までは Debian でベースとする Linux カーネルバージョンは決まっていましたが、パッケージのリリースサイクルはあまり決まっていませんでした。去年の Debconf では Linux カーネルの stable リリースに合わせて開発を行うことが決まり、パッケージのバージョニングと開発/メンテナンススタイルもリリースサイクルに合わせて変更されました。

4.3.1　Debian カーネル用語

Debian カーネルを扱うにあたり、カーネルという言葉はいろんな意味を持ちます。よく使われる用語をまとめてみました。

- 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。

4.3.2 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わりまました。2010 年 5 月現在、 Linux の LTS サポートカーネルバージョンは 2.6.32、最新版は 2.6.32.12 です。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 になります。 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になります。これにより新しい 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ません。

4.3.3 パッチのバックポート

Linus カーネルからのバックポート(linux-2.6.34-rc7 で取り込まれたパッチを linux-2.6.32 に取り入れてもらうなど）は、 Debian パッケージに直接取り込まれることはなく、 stable カーネルからのみ受け付けます。 stable カーネルで採用されない限りは Debian でも使えないということです。取り込んでもらうには、 stable@kernel.org にメールし、stable カーネルに取り込んでもらうように交渉する必要があります。

4.3.4 バグレポートとパッチ

バグがあった場合には、 Debian の BTS を利用できます。パッチが用意できる場合には、添付しましょう。メンテナが Upstream(stabel カーネル、場合によっては Linus/HEAD) に転送してくれます。

Debian Kernel Team で作成されたパッチは積極的に Linus カーネルに取り込まれるように働きかけています。これらが、 Liuns/HEAD または stable カーネルに取り込まれない場合、 Debian specific パッチとして管理されます。例えば、ドライバの non-free ファームウェアのパッチなどはまだ全て取り込まれておらず、一部は Debian specifc パッチとして残っています。

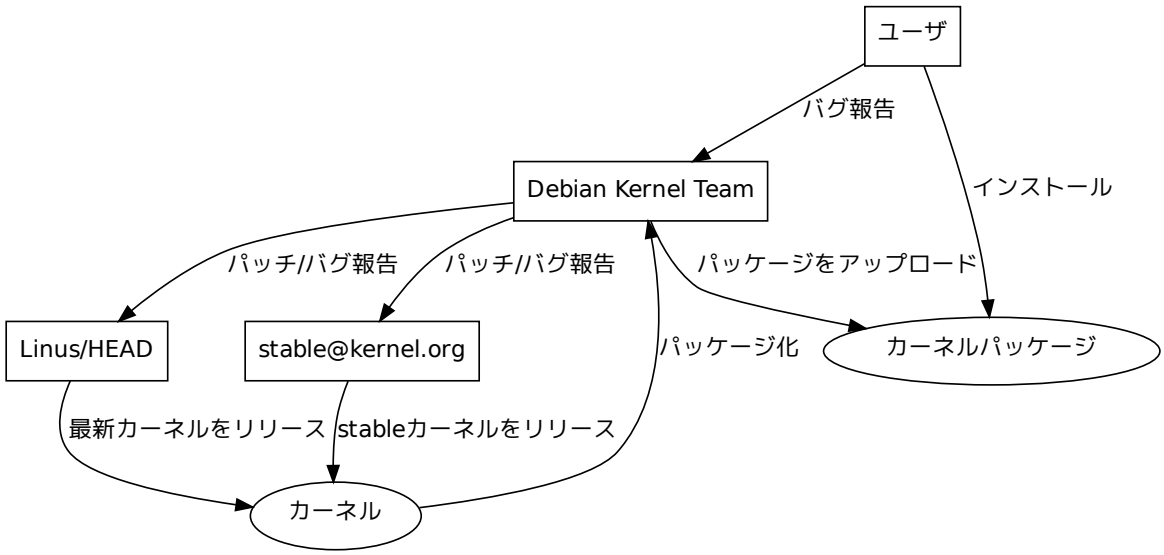


図 2　Debian Linux カーネル開発プロセス

4.4 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。

4.4.1 linux-2.6 パッケージ

Debian Kernel Team がメンテナンスしているパッケージの一つに linux-2.6 があります。これは Linux カーネルの主要なパッケージであり、このパッケージから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ます。

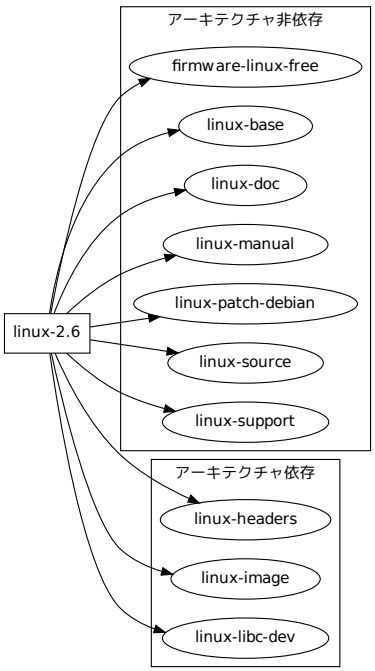


図 3　linux-2.6 パッケージから生成されるパッケージ

カーネルコンフィグ

linux-2.6 ソースパッケージで持っているカーネルコンフィグは基本 config , アーキテクチャ用 config , flavour 用 config 3 つに分けられており、 debian/config ファイルに格納されています。これらのファイルはバイナリパッケージビルド時に一つにまとめられ、まとめられた config を使ってカーネルコンフィグが行われます。ファイルで設定されているコンフィグは基本的に重複しません。しかし、組み込みで使われるボードでは、ドライバモジュールを組み込みにしないと動作しないものもあるため、 flavour 用の config ファイルによってオーバライドされます。よって、各ファイルの優先順位としては基本 config <アーキテクチャ用 config <flavour 用 config となります。また、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せず、アーキテクチャメンテナがちまちまファイルを修正し、アップデートします。

4.4.2 linux-latest-2.6 パッケージ

`linux-latest-2.6` パッケージは 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します。例えば、 amd64 アーキテクチャ向けの基本 Linux カーネルイメージパッケージは `linux-image-2.6.32-5-amd64` になりますが、 `linux-latest-2.6` ソースパッケージからビルドされる `linux-image-2.6-amd64` は `linux-image-2.6.32-5-amd64` に依存します。パッケージ名に ABI のバージョン(上の例だと 5）を含めてるので、 ABI が変更された場合にカーネルアップデートがされません。新規にパッケージをインストールする必要があるわけです。例えば、 `linux-image-2.6.32-4-amd64` と `linux-image-2.6.32-5-amd64` では ABI が異なるので別パッケージ扱いになり

ます。このときに、 linux-image-2.6-amd64 をインストールしておくと、 ABI が 4 から 5 にアップーデートされた場合に、 linux-image-2.6-amd64 もアップデートされ、 linux-image-2.6.32-5-amd64 が自動的にインストールされます。

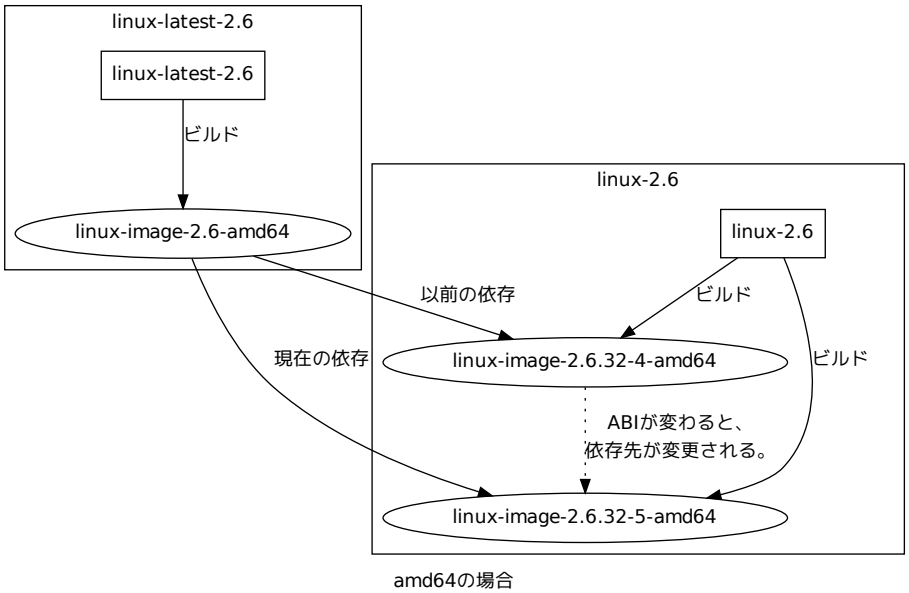


図 4　linux-2.6 と linux-latest-2.6 の関係

ABI のチェック

ABI のチェックは linux-2.6 パッケージ内にある debian/bin/buildcheck.py を使って、カーネルパッケージビルド時に実行されます。カーネルパッケージメンテナは手元でビルドしたときに、 ABI のアップデートをチェックし、 ABI を更新して Debian にアップロードします。といっても、大幅な変更、例えば sycall が追加/変更されるなどの変更がない場合には ABI を変更しない場合もあるようです。

```
--省略--
make[3]: Leaving directory `/home/mattems/src/linux-2.6-2.6.32/debian/build/build_amd64_none_amd64'
python debian/bin/buildcheck.py debian/build/build_amd64_none_amd64 amd64 none amd64
ABI has changed!  Refusing to continue.

Added symbols:
dev_attr_usbip_debug    module: drivers/staging/usbip/usbip_common_mod, version: 0x79bd9084, export: EXPORT_SYMBOL_GPL
getboottime             module: vmlinux, version: 0x0619ca8a, export: EXPORT_SYMBOL_GPL
monotonic_to_bootbased  module: vmlinux, version: 0xdb274e52, export: EXPORT_SYMBOL_GPL
--省略--
```

4.4.3　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っています。よって、 linux-headers パッケージに依存しています。このパッケージのソースコードは linux-2.6 パッケージから作られず、別途の 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ます。これは、 kbuild システムが stable リリース毎に更新する必要がないためです。

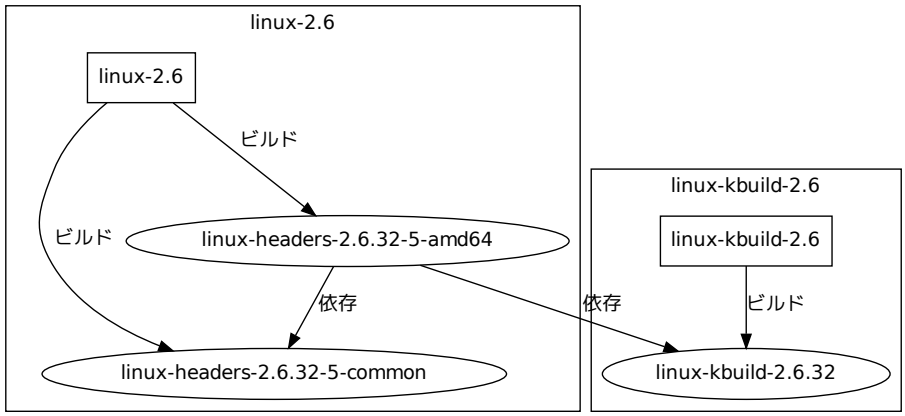


図 5　linux-2.6 と linux-kbuild-2.6 の関係

4.4.4　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。

4.5　今時の Debian カーネルのビルド方法

大抵のユーザは Debian で提供されているバイナリパッケージを使います。しかし、たまにビルドしたい人がいるわけです。理由としては以下のものが考えられます。

- 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。

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。

4.5.1　Debian オフィシャルカーネルをリビルドする

まずは基本の Debian カーネルのリビルド方法を説明します。ソースパッケージをダウンロードし、debuild を実行すれば linux カーネルパッケージがビルドされますが、この方法では自分の必要のない flavour までビルドします。ここでは、指定した flavour のみをビルドする方法を説明します。

1. 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
```

2. 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
```

3. 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
```

4. 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
```

5. 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 ../../..
```

6. 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_amd64
```

4.5.2 Debian カーネルにパッチを適用して利用する。

よく行うと思われるのが、 Debian カーネルをベースに自分が作ったパッチを当てて管理するというものです。これを行うには、 Debian のカーネルパッチ機構を知る必要があります。

Debian カーネルパッチ機構

基本的に通常のパッケージのパッチシステムと変わりません。 debian/patches にパッチが格納されています。四つのディレクトリがあり、さらにアーキテクチャ毎にパッチが分かれています。

- bugfix
  重要なバグ修正用パッチを格納します。
- debian
  Debian 専用パッチを格納します。
- features
  まだ upstream にマージされていないパッチを格納します。
- 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。

自分のパッチを適用したカーネルをビルドする方法

1. 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
```

2. チェンジログを更新する。
   dch コマンドを使って、新しい Debian バージョンで Changelog を作成します。このときに、-D オプションを使って、ディストリビューション名に UNRELEASED を指定しない場合、カーネルパッケージビルドのチェックに引っかかります。以下の例では、サフィックスにローカルバージョンとして +text を指定し、 Changelog ファイルを更新しています。この場合、 Liux カーネルパッケージのバージョンが 2.6.32-12 の場合、 2.6.32-12+test1 というバージョンになります。

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

3. 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/
```

4. 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/12+test1
```

5. ./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
```

6. 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
```

7. パッケージをビルドする。
   パッチが適用できたら以下のコマンドを実行し、パッケージをビルドします。

```
$ fakeroot make -f debian/rules.gen binary-arch_amd64_none_amd64
```

   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。

4.5.3 Debian カーネルの Linux カーネルソースコード (linux-source-2.6.XX パッケージ) から再ビルドする。

Debian カーネルを再ビルドする方法はもうひとつあり、 linux-source-2.6.XX パッケージを利用する方法です。このパッケージにはアップストリームのソースコードのみを提供しています。このパッケージからカーネルパッケージをビルドする方法を説明します。

1. 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
```

2. 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
```

3. 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=test00debian kernel_image kernel_headers
```

しかし、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です。 `linux-patch-debian-XXXX` パッケージをインストールし、パッチを適用する必要があります。-a でアーキテクチャ、-f で flavour を指定します。

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

4.5.4 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 パッケージを作成するには `kernel-package` パッケージを使うのが Debian 流です。

1. `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
```

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

```
$ make menuconfig
```

4. `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です。

```
$ 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)_(architecture).deb
```

5. ビルドが終わるとパッケージがビルドされているので、インストールします。

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

4.6 git/HEAD を Debian パッケージにする

今時はカーネルがリリースされる度にカーネルのソースコードをダウンロードするのではなく、常に git リポジトリを更新し、 git リポジトリからビルドするのが通でしょう。たぶん。 kernel-package パッケージでは、 git リポジトリからビルドできる機能があるのでこれを利用すると容易にカーネルパッケージを作成することができます。

1. 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/linux-2.6.git
```

   linux-2.6 ディレクトリができるので移動します。

```
cd linux-2.6
```

2. 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
```

3. 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
```

4. 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:\%h -1‘ --initrd kernel_image
-- 省略 --
$ ls ../
linux-2.6  linux-image-2.6.34-rc7_1+be83567_amd64.deb
```

4.7 ドライバモジュールの取扱い

Debian カーネルパッケージでは、多くのカーネルドライバモジュールが用意されています。最近では module-init-tools の仕様変更があり、設定ファイルのサフィックスが変更されています。ドライバモジュールの取り扱いについて一度簡単におさらいしてみましょう。

4.7.1 ロードしたいカーネルを指定する。

起動時に自動的にモジュールをロードする設定は、/etc/modprobe.d/に新しいファイルを作成して、そこに記述するします。

たとえば、 eth0 として eepro100 をロードさせたい場合、`/etc/modprobe.d/local.conf` ファイルを用意して、設定を記述します。

```
echo 'alias eth0 e100' >> /etc/modprobe.d/local.conf
```

とします。

4.7.2 カーネルモジュールパラメータの設定

パラメータの設定も/etc/modprobe.d/local.conf に記述します。カーネルモジュールのパラメータを調べるには、`modinfo` コマンドを利用します。

```
$ modinfo e100
filename:       /lib/modules/2.6.31-1-686/kernel/drivers/net/e100.ko
firmware:       e100/d102e_ucode.bin
firmware:       e100/d101s_ucode.bin
firmware:       e100/d101m_ucode.bin
version:        3.5.24-k2-NAPI
--省略--
vermagic:       2.6.31-1-686 SMP mod_unload modversions 686
parm:           debug:Debug level (0=none,...,16=all) (int)
parm:           eeprom_bad_csum_allow:Allow bad eeprom checksums (int)
parm:           use_io:Force use of i/o access mode (int)
```

設定する場合には以下のように記述します。

```
$ echo 'options e100 debug=16' >> /etc/modprobe.d/local.conf
```

設定したら、再起動を行うかカーネルモジュールを再読込します。

4.8 よくある質問

4.8.1 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。 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。

4.8.2 initrd 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。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/initramfs \
 /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。

4.8.3 -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 kernel_headers DEBIAN_KERNEL_JOBS=8
```

4.8.4 最新カーネル向けの linux-kbuild-2.6 を作りたいのですが

最新リリースカーネルの Debian パッケージは experimental ディストリビューションにアップロードされます。2010 年 05 月の時点では、バージョン 2.6.33 で、 `linux-2.6_2.6.33-1 experimental.5` としてアップロードされています。カーネルはアップロードされているのですが、 linux-kbuild-2.6 パッケージが最新カーネル向けにアップロードされない事があるので、使いたいユーザは自分で用意する必要がある場合があります。作り方は `http://wiki.debian.org/HowToRebuildAnOfficialDebianKernelPackage#Thestoryoflinux-kbuild-2.6` を参照してください。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。パッチを当てて、ビルドしましょう。

4.8.5 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。

Debian 勉強会資料

2010 年 5 月 17 日　　初版第 1 刷発行
東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）