

東京エリア デビアン 勉強会



Debian勉強会幹事 上川純一

2009年1月17日

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。

2009 年の計画は仮です。

1. 新年の企画 (アンサンブル荻窪開催)
2. OSC
3. (東京大学?)
4. (千代田区都立図書館?*¹)
5. (東京大学?)
- 6.
7. スペインにて開催
8. Debconf 報告会
- 9.
- 10.
- 11.
12. 忘年会

会場候補としては下記があります:

- 大学
- 恵比寿 SGI ホール
- Google オフィス
- 公民館 (あんさんぶる荻窪等)
- 都立会議室 (無線 LAN)
- 健保の施設

*¹ <http://www.library.chiyoda.tokyo.jp/>

会 強 勉 ア ビ ト

目次

1	Introduction	1
2	事前課題	3
3	最近の Debian 関連のミーティング報告	5
4	Git+ メールでの事前課題提出にまつわる課題	6
5	Namazu みたいに Google AJAX Search API	9
6	AspireOne で Debian sid サーバー	12
7	Debian GNU/Linux 2ch ブラウザ普及計画	14
8	Linux カーネルコンフィグ変換ツールを作ってみた	16
9	黒 MacBook の Lenny/Sid 64bit 化でのハマリポイント	21

2 事前課題

上川 純一



2012 年の正月、僕は実家に帰って旧友を深めるべく喫茶店にいた。最近こころへんには立ち寄っていないなあ。そういうことをつらつらとっているとふと人影が目に入る。

「おお、待たせたな」

「ああ。久しぶり」

「どうした」

「いや」

そして僕は手元の Debian デバイスから目をはなす。

「お、Debian の調子はどうよ、最近はどんな使い方しているの?」

この質問をいま僕にするなんて。血圧がすこし上がるのを感じながら僕は饒舌になりすぎないように説明する。

「(A)」

問題

1. Debian に関しての、2009 年の抱負を述べよ
2. 「2012 の Debian」と題して (A) を埋めよ

この課題に対して提出いただいた内容は以下です。

2.1 上川純一

2.1.1 Debian に関しての、2009 年の抱負

- 使っているパッケージをメンテナンス、使っていないパッケージを捨てる
- 今止まっている pbuilder / cowdancer / qemubuilder のリグレッションテストサーバを復活させる

2.1.2 2012 の Debian

この時計なんだけどさ、タッチスクリーンとプロジェクトがついていて、それで動くんだぜ。おっと、あまり頭をうごかさなでくれ、こいつが敵として認識して攻撃してしまう。

2.2 山本 浩之

2.2.1 Debian に関しての、2009 年の抱負

メンテナンスしているパッケージを地味に増やしていく。

2.2.2 2012 の Debian

順調に squeeze のリリースも遅れてますな。

2.3 岩松 信洋

2.3.1 Debian に関しての、2009 年の抱負

- DD になる。
- 他のサブプロジェクトにも積極的に参加する。
- SH の開発を継続する。

2.3.2 2012 の Debian

まだ目で操作するのは慣れないな。メガネで操作するのはつらいぜ。

2.4 小林 儀匡

2.4.1 Debian に関しての、2009 年の抱負

- DD になる。
- upstream へのパッチの積極的な投稿を続ける。

2.4.2 2012 の Debian

おまえみたいに久し振りに会う奴の名前や好きな話題がとっさに出てこないから、その辺りの情報を瞬時にこの超小型イヤホンに送ってもらって*2.....いや何でもない。いたって普通に、生活で使うデバイス全般で動かし

ているよ。
それよりのび太、おまえ、噂によれば最近、Debian の乗った、猫耳のついたロボットを開発しているそうじゃないか。

2.5 キタハラ

2.5.1 Debian に関しての、2009 年の抱負

会社で消滅した debian で動作しているサーバを復活させる。

2.5.2 2012 の Debian

相変わらず調子がいいよ。でも、これ Android だよ。
(画面は自宅サーバのリモートコンソールだったのだ! ちゃんちゃん!!)

2.6 まえだこうへい

2.6.1 Debian に関しての、2009 年の抱負

- ヨメを連れてスペインに行く。(行きたい。)

- スペイン前までにヨメを洗脳する。(わら

2.6.2 2012 年の Debian

時計なんだけどさ、電波時計のボタンと見せかけて、半径 10m 以内に Debian 信者がいるか分かるんだぜ。

2.7 あけど

2.7.1 Debian に関しての、2009 年の抱負

- (挫折中の) 翻訳の査読を進める
- 管理している Debian な公開サーバがあるのですが、これを仮想サーバにサービス無停止で移行しようかと思っています。また、この移行に関する資料を作成・公開したいと思います。

2.7.2 2012 年の Debian

これ、ぱっと見ると電子辞書なんだけどころすると動画とか見れるんだな、実は携帯端末で自宅サーバと VPN 接続してこれだけで大抵の仕事ができたりするんだ。そんでこの端末とか自宅のルータ&サーバは全部 Debian なのよ

2.8 じつかた

2.8.1 2009 年の Debian 抱負

- パッケージ作成の勉強を進める
- Debian サーバ導入 (仕事)

2.8.2 2012 年の Debian

「この携帯、Debian で動いてるんだよ。」

*2 元ネタ: 星新一『ささやき』。



3 最近の Debian 関連のミーティング報告

上川 純一

3.1 東京エリア Debian 勉強会 47 回目報告

東京エリア Debian 勉強会報告。2008 年 12 月 20 日土曜日に第 47 回東京エリア Debian 勉強会を実施しました。今回の参加者は前田耕平さん、一井崇さん、やまだたくまさん、小室文さん、平澤俊雄さん、藤沢理聡さん、じつかたさん、岩松信洋さん、小林儀匡さん、山本浩之さん、キタハラさん、日比野啓さん、id774 さん、やまねひできさん、でんさん、でんさんの友人 A さん、上川 x2 の 18 名でした。

まず、2008 年の勉強会の実績について上川が話をしました。今回まとめてみた情報は、参加人数と事前課題の提出実績と事後のブログ掲載の率です。事後のブログ掲載は開催している側としては励みになるのでうれしいというような話をした気がします。

次に上川が、2009 年にむけて最近のトレンドについて整理して、SWOT の図を作成するという企画をしました。議論はいろいろと発散しましたが、アクションアイテムはたくさんできたのでこれが次につながるとよいですね。

最後にライトニングトークを実施しました。小林さんは apt-cache dotty の話。id774 さんは Debian ユーザとしての告白。上川は sqlite3 使って見ましたネタ。やまだたくまさんは資料の翻訳のやりかたについて、機械翻訳をベースにどう労力を節約するかというネタ。岩松さんは日本 . ユーザ会設立について。前田さんはヨメを Debian ユーザにする方法。日比野さんは ocaml ユーザ会設立宣言。

3.2 Debian JP 定例会議 on IAX

Debian JP の定例会議は通常 IRC で行っているのですが、2009 年 1 月 8 日に執り行われた会議は実験的に音声通話も交えて行いました。利用したプロトコルは IAX プロトコルで、iaxcomm をクライアントとして利用し、asterisk サーバに全員接続しました。

MacBook のマイクまわりのハックが十分でなく、iaxcomm 自体の安定度合いもよくなかったので音質が悪いという問題がありましたが通信状態は概ね良好でした。今後さらに試験を重ねて実用的に使って行きたいものです。

4 Git+メールでの事前課題提出にまつわる課題

上川純一



2008 年 11 月、12 月、および 2009 年 1 月には事前課題を Git を利用して Git の生成するパッチをメールで提出してもらおうという形式にしていました。その場合には git pull するたびにコンフリクトが発生します。その原因となる仕組みと対策方法について説明します。

4.1 コンフリクトが発生しやすい理由

それでは、Debian 勉強会の 2008 年 11,12 月の資料でコンフリクトが発生しやすかった理由を分析してみましょう。

4.1.1 元のファイルの形式

```
-\subsection{}  
+\subsection{XXXX}  
+ 内容  
+
```

同じファイルについて作業し、しかも同じ場所にすこしづつ異なる変更を別の人が行うという仕組みになっていました。この場合、二つのパッチはかならずコンフリクトし、手動で解消する必要があります。今回パッチが同じ場所に挿入している場合に、順序は問わないので適当でよいのですが、git apply にはその知識はないため、毎回コンフリクトの解消を行います。

4.1.2 メールを利用したサブミット

締切り直前に作業するのが習慣のため、複数の人がある時点の同じコミットに対してパッチを作成、それを上川がある時点でマージしました (図 1)。

毎回コンフリクトが発生するのですが、それを上川が解消してマージとして記録されています。そのため、各自が提出したバージョンとは若干違うパッチとなって alioth.debian.org にある Git ツリーにマージされています。

4.2 マージ・コンフリクトの発生のしかた

Git はデフォルトでは master ブランチで作業します。git pull コマンドは Alioth にあるリモートのツリーの情報を origin ブランチにとってきて、master ブランチにマージします。マージする内容がなければ、Fast-forward されます。

まず最初に各自が自分の Git ツリーの master ブランチでパッチを作成します。そして、git format-patch の結果をパッチファイルとして送付します。上川が git am でそのパッチを適用し、Alioth の Git ツリーに公開します。複数人が同時期に事前課題を提出しているため、git am でパッチを適用した場合にそのままでは適用できない状況がおき、「コンフリクト」が発生します。「コンフリクト」が発生した場合には上川が手動でその部分を修正します。

Graph	Short Log	Author	Author Date
●	finalize for printin...	Junichi ...	2008年12月19日 09時28分05秒
■	Merge branch '20...	Junichi ...	2008年12月19日 08時31分45秒
●	Add by Yamamoto	Hiroyuki...	2008年12月19日 07時05分21秒
■	Merge branch '20...	Junichi ...	2008年12月19日 08時29分55秒
●	キタハラ 事前課題	KITAHARA...	2008年12月19日 02時04分27秒
●	home work added...	Kazuyu...	2008年12月19日 06時21分06秒
■	use clearpage ins...	Junichi ...	2008年12月19日 00時41分40秒
●	tweak presentatio...	Junichi ...	2008年12月19日 00時05分40秒
■	Merge branch 'ma...	Junichi ...	2008年12月19日 00時01分43秒
●	Add my own answe...	Noritada...	2008年12月18日 23時24分46秒
●	initial presentation...	Junichi ...	2008年12月18日 23時59分04秒
●	add references to ...	Junichi ...	2008年12月18日 23時58分58秒
●	added by Aya Ko...	Aya Ko...	2008年12月18日 22時21分12秒
●	add comment by ...	Takuma...	2008年12月18日 08時54分42秒

図1 qgit でみたツリーの状況、手動でマージが複数発生している

各自が Alioth の Git ツリーの最新バージョンを `git pull` で取得してくると `origin` ブランチには上川の作成した変更を伴ったパッチが入ります。 `master` ブランチで自分の作成したものからは若干変更が加わっています。そのため、内容に整合性がとれるとはかぎらないため、コンフリクトが発生します。コンフリクトがおきなかったとしても履歴にマージが残ります。

4.3 マージ・コンフリクトの解消方法

4.3.1 rebase して履歴をきれいにする

マージなどを解消するためには、 `git rebase -i origin` で不要なパッチを目視して消す作業をすればよいです。そもそも `git pull --rebase` すると、おそらくマージが残らないが、コンフリクトの解消は必要になります。これは面倒です。

4.3.2 作業の仕方を変える

解決策としては、各自が提出用のブランチを作成してそこで作業し、そこで提出用のデータを作成してしまえばよいというのがあります。

```
$ git checkout -b preworkXXXX origin
$ # ... このブランチで作業
$ git format-patch ...

# 提出してしまったら master ブランチにもどる
$ git checkout master
$ git pull # 上川の適用した版がとりこまれる

# さらに新しいブランチを作成して次の月の作業を行う
$ git checkout -b preworkYYYY origin
```

preworkXXXX ブランチを毎回捨てる。

4.3.3 format-patch ではない方法を使う

もう一つの選択肢としては、 `format-patch` ではない方法でデータの送受信を行うことがあります。 `format-patch` と `git am` の送受信の過程でコミットのハッシュが異なっていることから各位のコンフリクトが発生しているのだろ

うということです。

git bundle でバイナリ形式でデータを送付することができます。これを使うとハッシュは保存されるので上川のコンフリクトは解消されませんが、コンフリクトを解消したログがそのまま残るので、各位のコンフリクトは発生しないでしょう。

4.4 リポジトリ管理者側の課題

視点をすこし変えてリポジトリの管理者として上川の作業を見てみましょう。git am でパッチを適用して、マージするという作業を行うのですが、そのワークフローは実は面倒です。

特にメールを受けて、git am がコンフリクトを起こす確率が高く、複数のパッチがあるとほぼ確実にコンフリクトを解消することになります。

現状は複数の一時的に利用するブランチに git am で適用したものをあとでマージする、という方法をとっています。

```
git checkout -b マージ用のブランチ A master
git am -3 パッチ
git checkout -b マージ用のブランチ B master
git am -3 パッチ
git checkout master
git merge マージ用のブランチ A マージ用のブランチ B
コンフリクトの解消
git commit -a
```

ただ、10人以上の事前課題のハンドリングには時間もかかり、毎回したい作業ではないです。

自動でコンフリクトするかしないかについては判定できるし、その結果がビルドするかはチェックできるのであれば、自動処理でコミットを管理できないか?と考えています。

最低限のチェックだけしてパッチを master ブランチに自動でとりこんでくれるような仕組み、ないもんではないか?かええ?

5 Namazu みたいに Google AJAX Search API

小室 文

自分のウェブサイトの検索用にわざわざ Namazu を導入しなくても Google の検索エンジンを使ってお手軽にサイト内全文検索する方法を説明します。

5.1 Google の検索エンジンをカスタマイズして使うには

利用するパターンがいくつかあります。それぞれのパターンについて設定方法を紹介します。

どれを選んでも基本は HTML ファイル(もしくは PHP など)をウェブサーバに置くことで動作します。JavaScript 版はブラウザさえあればどこでも動きます。

1. JavaScript あり
 - (a) 自分で JavaScript を組む
 - (b) ウィザードを使う
2. JavaScript なし

5.1.1 JavaScript あり・なし両方

Google AJAX Search API の KEY を事前に取得しておきます。

5.1.2 JavaScript あり

HTML ファイルの中で Google AJAX Search API JavaScript ライブラリをロードします。

```
<script src="http://www.google.com/uds/api?file=uds.js&v=1.0" type="text/javascript"></script>
```

```

//名前空間の為に追加
google.load("search", "1.0");

//検索をするオブジェクトを作成する
var searchControl = new google.search.SearchControl();

//検索準備
//子オブジェクトのサーチャーメソッドを検索コントロールに追加。必要であればオプションも。
var options = new GsearcherOptions(); //表示オプション指定
options.setExpandMode(GSearchControl.EXPAND_MODE_OPEN); //結果一覧を開いた状態で出力する
options.setExpandMode(GSearchControl.EXPAND_MODE_CLOSED); //結果を閉じた状態のまま結果出力をする
options.setExpandMode(GSearchControl.EXPAND_MODE_PARTIAL); //結果がオープン拡張モード
options.setRoot(document.getElementById("resultsComeHere")); //結果を指定する場所に表示するオプション

//サイト制限の設定
var siteSearch = new GwebSearch();
siteSearch.setUserDefinedLabel("Aya's Site");
siteSearch.setSiteRestriction("popowa.com"); //直接 URL を指定
siteSearch.setSiteRestriction("000455696194071821846:reviews"); //カ
スタムエンジンの Key を指定\footnote{カスタム検索エンジンとはウィザードで
検索窓のある程度のデザインや制限を付けて作ることが出来るものです。\\
\url{http://www.google.com/coop/cse/}}

//準備終了、セットする
searchControl.addSearcher(siteSearch, options); //オプションがあれば
searchControl.addSearcher(new GwebSearch()); //オプションがなければ
//出力準備
var result = document.getElementById("googleResults");
var drawOptions = new GdrawOptions(); //オプション用に drawOptions オブジェクトを作成
drawOptions.setDrawMode(GSearchControl.DRAW_MODE_LINEAR); //1) 一列に出力
drawOptions.setDrawMode(GSearchControl.DRAW_MODE_TABBED); //2) タブに出力
drawOptions.setSearchFormRoot(document.getElementById("searchForm")); //検索結果と検索フォームを切り離す機能
searchControl.draw(result, drawOptions);
//検索結果の保持 (prototype)
searchControl.setOnKeepCallback(this, MyKeepHandler); //この (this) 検索状況を MyKeepHandler に持たせておく。
searchControl.setSearchCompleteCallback(this, OnSearchComplete); //検索の実行後
searchControl.setSearchStartingCallback(this, OnSearchStarting); //検索の完了前
//検索する！
searchControl.execute(keyword);

```

そして、body 内に検索窓、検索結果を出力したい場所に div を入れます。

```

<div id="searchForm" class="searchForm">form</div>
<div id="googleResults" class="googleResults">googleResults</div>

```

5.1.3 JavaScript なし

http://ajax.googleapis.com/ajax/services/search/web に引数を渡してリクエストすると JSON 形式でレスポンスが得られます。

パラメータ	項目	注意事項
q?	検索したいキーワード、検索式	日本語だった場合は urlencode() しないとうまく動かない
v=1.0	プロトコル番号の指定	現在は 1.0 しかない
key?	Google AJAX Search API の Key	申請した FQDN=実際に使っている FQDN でなくてもよい
start?	検索結果の開始インデックス	ドメイン指定をすると 160 件から持って来てくれなくなる
cx?	カスタム検索エンジンの ID	特定のドメインからのみ検索する事が出来る
lr?	特定の言語のドキュメントを検索対象とする	

レスポンス形式

```

{"responseData": {
  "results": [
    {
      "GsearchResultClass": "GwebSearch",
      "unescapeUrl": "http://en.wikipedia.org/wiki/Paris_Hilton",
      "url": "http://en.wikipedia.org/wiki/Paris_Hilton",
      "visibleUrl": "en.wikipedia.org",
      "cacheUrl": "http://www.google.com/search?q\u003dcache:TwrPfh22hYJ:en.wikipedia.org",
      "title": "\u003cb\u003eParis Hilton\u003c/b\u003e - Wikipedia, the free encyclopedia",
      "titleNoFormatting": "Paris Hilton - Wikipedia, the free encyclopedia",
      "content": "\[1\] In 2006, she released her debut album..."
    },
    ...
  ],
  "cursor": {
    "pages": [
      { "start": "0", "label": "1" },
      { "start": "4", "label": "2" },
      { "start": "8", "label": "3" },
      { "start": "12", "label": "4" }
    ],
    "estimatedResultCount": "59600000",
    "currentPageIndex": 0,
    "moreResultsUrl": "http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8..."
  }
}, "responseDetails": null, "responseStatus": 200}

```

レスポンスを個々のプログラムで処理する

```

<?php
function search($keyword, $page, $key){
    $uri = 'http://ajax.googleapis.com/ajax/services/search/web';
    $uri .= '?q=' . urlencode($keyword);
    $uri .= '&key='.$key;
    $uri .= '&v=1.0&rsz=large&hl=ja&start=' . $page;
    return json_decode(file_get_contents($uri));
}
$keyword = "東京エリア Debian 勉強会";
$key = 'アプリケーションのキー';
$data = search($keyword, 1, $key);
var_dump($data);
?>

```

5.1.4 現状の問題点

Google AJAX Search API のかえしてくる「検索結果」の総数には問題があります。

JavaScript あり: setSiteRestriction でドメインを指定して検索をしようとした場合、検索結果の総数が検索するたびに変わります。

JavaScript なし: ?q=DMC%\%20site:http://www.debian.or.jp/とするとドメイン検索は出来るが、start?で引数を渡すたびに同じように結果総数が変わります。

5.1.5 まとめ

Namazu みたいにページ分けされた検索の仕組みは構築出来ませんでした (2009/1 現時点)。

Google AJAX Search API の総数が変わる件については、estimatedResultCount というくらいなので、estimate なのでしょう。^{*3}ちなみに Yahoo Web 検索でも同じような問題が発生します。Yahoo!は検索リクエストがある度に検索結果を積算しているのだから違いますが、と予め免責してあります。^{*4}

^{*3} <http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=70920> 'How does Google calculate the number of results?' に記述があります

^{*4} 表示件数について: <http://help.yahoo.co.jp/help/jp/search/web/web-14.html>

6 AspireOne で Debian sid サーバー

id774

この冬休みを利用して AspireOne に Debian sid の環境を構築しました。AspireOne と言えば今流行りの NetBook としてモバイル用途で知られています。今回はその省電力性に注目し、モバイル以外にもポータブルで場所を取らない簡易サーバーとして利用可能なのではないかと考えトライしました。

6.1 AspireOne で sid を使ってみる

6.1.1 AspireOne の電力消費効率

AspireOne は省電力性に優れた Intel Atom CPU を搭載しており、アイドル状態で約 10W、高負荷状態でも 15W 程度で動作します。高性能を要するエンコードやリッピングではなく、ちょっとした Web サーバー等の用途であれば安価でエコなサーバーとして利用できそうです。

6.1.2 sid を選択する理由

通常、サーバーであれば安定版を選択します。しかし AspireOne ならいつでも携帯して持ち運び、管理者が対話的に操作してメンテナンスすることができるので、これなら最新のパッケージを利用できる sid を使っても、何かトラブルがあればすぐ対応できるでしょう。

6.2 Debian をインストール

Debian Wiki ^{*5} に情報がまとまっているので、基本的にこれを参考に作業しました。

6.2.1 名刺サイズのイメージからのインストール

sid のインストールには主に 2 種類の方法^{*6} があります。

- テスト版の Debian の sources.list を sid に書き換えて aptitude full-upgrade する。
- 最小サイズの最新ビルドイメージを利用してエキスパートモードでネットワークインストールをおこなう。

今回は後者の方法を採用しました。インストールには Daily ビルド最新版の Debian イメージ^{*7} を利用します。名刺サイズのインストールイメージを利用して、エキスパートモードでインストールを開始すると、不安定版を選択することができます。

^{*5} <http://wiki.debian.org/DebianAcerOne>

^{*6} <http://www.debian.org/CD/faq/#unstable-images>

^{*7} <http://cdimage.debian.org/cdimage/daily-builds/daily/arch-latest/i386/iso-cd/>

6.2.2 暗号化 LVM の設定

モバイルで外に持ち出すことを考えると、HDD や PC 本体の盗難によるリスクを考慮しなければなりません。Debian は etch から暗号化 LVM を標準で利用出来るようになりました。これで /boot 以外の領域を暗号化することができますのでセキュリティとしては非常に強力になるでしょう。

この際、元の Windows 領域を全て上書き末梢するので 4~5 時間程度かかります。そこで一晩放置して翌日作業をするようにしました。

6.2.3 無線 LAN の設定

AspireOne には Atheros AR5007 チップが搭載されています。そこで module-assistant を利用して MadWifi をインストールしました。

```
sudo aptitude install build-essential module-assistant madwifi-source
sudo m-a prepare
sudo m-a auto-install madwifi
```

6.2.4 イー・モバイルの設定

現時点で lenny/sid に搭載されているカーネル 2.6.26 ならイー・モバイルによる通信が可能で、さらに USB の着脱によるプラグアンドプレイにも対応しています。

pppconfig パッケージをインストールし /etc/ppp/peers/em ファイルに以下のように記述します。文字列はイー・モバイル固有ですのでそのまま指定します。dip グループに所属したアカウントなら pon em コマンドを発行すればイー・モバイルによるネットワーク接続ができるようになります。

```
user "em@em"
connect "/usr/sbin/chat -v -f /etc/chatscripts/pap -T *99***1#"
/dev/ttyUSB0
115200
noipdefault
usepeerdns
defaultroute
persist
noauth
```

6.3 sid をメンテナンスする

6.3.1 アップグレード

私は以下のコマンドの実行結果を cron で定期的を取得しています。

```
aptitude update && aptitude -s -v -y full-upgrade
```

実際にアップグレードはしませんが更新予定のパッケージの一覧を見ることができます。これに加え必要に応じて apt-listchanges や apt-listbugs 等を使い、既存パッケージの依存関係に影響を及ぼさないことを確認の上でアップグレードすることになっています。

6.3.2 仮想環境での事前検証

さらに確実性を高めるための方法として VMware など仮想環境を利用する方法があります。あらかじめゲスト OS として実機と同じパッケージ構成の sid を用意し、先にそちらを変更して問題が無いことを確認してから実機のパッケージを変更すればトラブルを避けることができます。

6.4 まとめ

Debian sid なら最新のドライバや機能が使えるので、流行の NetBook にインストールしてモバイルなサーバーとして活用するのも良いですね。ぜひみなさんもトライしてみてください。



7 Debian GNU/Linux 2ch ブラウザ普及計画

山本 浩之

結構前から色々な 2ch ブラウザを stable 用にバックポートして、2ちゃんねるスレッドテンプレなどで非公式に upload していました。しかし、最近 JD が公式パッケージ入りしましたが、それ以外はなかなか公式には入ってき
てはいません。そこでいくつかの 2ch ブラウザについて、公式パッケージ入りを検討してみました。

今回検討対象とした 2ch ブラウザは Navi2ch for Emacs、おちゅ～しゃ、Kita、Kita2、Chalice for Vim、w3m-2ch、Gnview です、それではそれぞれみてみましょう。

7.1 Navi2ch for Emacs

Emacs 上で動く 2ch ブラウザ。	ライセンス:GPL
----------------------	-----------

昔から Debian Developer の野首さんが、Navi2ch の本家で最新の CVS 版のパッケージを配っていることで有名
です。ライセンス的にも問題はありますが、どうやら野首さんのポリシーとして、公式パッケージに入れるのは控
えているそうです。現に、やっと Lenny 入りした JD も 2ch の仕様変更のため、Lenny 版は掲示板に書き込めない、
ということになってしまいました。今後も続きそうな、いきなりの仕様変更にも対応できる配りかた (backports と
か volatile とか) を検討すれば、きっと野首さんも公式パッケージに入れてくれるものと信じています。

7.2 おちゅ～しゃ

GTK+ 上で動く、C++ で書かれた 2ch ブラウザ。	ライセンス:2 項目 BSD (LGPL ライブラリ を含む)
-------------------------------	------------------------------------

これはかなり前から検討していましたが、upstream の開発周期が不安定で、時々完全に止まってしまうのが難
でした。最近、また開発が再開し、久々のアップデートが出ていたので、思わず ITP してしまいました。パッケージは
CDBS を用いて、問題なくでき、起動も問題ないようです。man と README.Debian を書けばアップロード出来
る所までできています。

7.3 Kita

KDE/QT 上で動く、C++ で書かれた 2ch ブラウザ。	ライセンス:GPL
---------------------------------	-----------

これも前から非公式で配られていますが (<http://tossi.orz.hm>)、tossi さんにコンタクトを取った所、公式パッ
ケージにするつもりはないそうです。私も検討しましたが、upstream が KDE4 への対応を、今のところする気が無
いということなので、かなり速くて高機能なのですが、現在保留中です。

7.4 Kita2

KDE/QT 上で動く、Ruby で書かれた 2ch ブラウザ。	ライセンス:MIT/X
----------------------------------	-------------

Kita の後継として出てきたブラウザです。まだまだバグもあり、機能も少ないのですが、ITP してみました。パッケージ的にも一応一通り揃え (CDBS)、スポンサー探しをしていたのですが、実は EUC-JP only な環境では警告もなく文字化けすることを上川さんに指摘され、upstream に連絡を入れている所です。

7.5 Chalice for Vim

Vim 上で動く 2ch ブラウザ。	ライセンス：俺俺ライセンス
--------------------	---------------

開発も順調ですが、元々 Vim を Windows に移植している人が upstream なためか、以下のような'俺俺'ライセンスです。

利用許諾

以下の諸条件に同意された方へ本ソフトウェアの利用が許諾されます。

本ソフトウェアを利用することで、ソフトウェア利用者にはソフトウェア作成者へ対価を支払う義務が生じません。

本ソフトウェアの不具合がソフトウェア作成者へ報告された場合には、ソフトウェア作成者は期間を限定せずに本ソフトウェアを修正しますが、修正前後を問わず本ソフトウェアの利用に際して生じた損害をソフトウェア作成者は補償しません。

ソフトウェア利用者は本ソフトウェアを、商用・非商用を問わず、使用・再配布することができます。

ソフトウェア利用者へは本ソフトウェアを改変する権利がソフトウェア作成者より与えられます。但し本ソフトウェアへ改変を施したバージョンを再配布する場合には、改変内容及びその実装方法をソフトウェア作成者へ無条件で開示する義務が生じます。

以上の諸条件に同意できない場合は本ソフトウェアの利用を中止してください。

このうち、特に問題になるのは、「但し本ソフトウェアへ改変を施したバージョンを再配布する場合には、改変内容及びその実装方法をソフトウェア作成者へ無条件で開示する義務が生じます。」の部分だと思います。これを「ソフトウェア作成者」から、「公衆」に変えると多分 GPL が近くなるのではないかと考えていますが、その変更でありうる損害が良く分からず、まだライセンス変更交渉もしていません。

7.6 w3m-2ch

w3m 上で動く 2ch ブラウザ。	ライセンス：記述無し
--------------------	------------

半角カナ以外の書き込みはできることは確認しましたが、残念ながらライセンスも、upstream への連絡先も記述が無く、断念しました。

7.7 Gnview

GTK+ 上で動く perl で書かれた 2ch ブラウザ。	ライセンス:GPL
--------------------------------	-----------

既にやまねさんにより ITP が出されていますが、日本語限定なこともあり、なかなかスポンサーが付かないようです。

7.7.1 現状の問題点

- 2ch ブラウザは日本語限定ソフトウェアで、日本語が分かるスポンサーを探さねばならない。
- 2ch は予告無く仕様変更をするため、新しいバージョンを即座に配布できる仕組みが必要。
- 2ch 自体の評判も芳しくなく、名前を出して関わることを嫌う人も多い。

などなど。

8 Linux カーネルコンフィグ変換ツールを作ってみた

岩松 信洋



8.1 はじめに

2008 年末に 2.6.28 が出ましたが、みなさんコンパイルしていますか？2、3 名ほどは毎朝昼晩 Linus ツリーから git pull してコンパイルされていると思いますが、大抵の方は Debian が提供しているカーネルを使っていると思います。使っている理由は様々ですが、コンフィグレーションがめんどくさいとか、どこを変えていいのかわからない、などが理由だと思います。^{*8}

また、最近で Linux カーネルも賢くなり、ドライバをモジュールにしておくと、ある程度自動的に必要なモジュールをロードしてくれるようになったのも理由の一つかもしれません。今回はユーザの立場からカーネルに触れるようになる方法の一つとして、Debian が提供しているカーネルからモジュールを組み込みにするためのスクリプトを作りました。これによって、どこを有効にすればいいのかわからないなどの問題がすべて解決します。

8.2 なぜ彼らはカーネルをリコンパイルするのか？

普通の Debian ユーザはカーネルをリコンパイルしないようです。毎日狂ったようにリコンパイルしている人はカーネルハッカーか、変態さんぐらいでしょう。彼らがカーネルコンパイル！コンパイル！と言っているからには何か理由があると思います。カーネルをリコンパイルする理由として以下の事が考えられます。

- カーネルハックのため。
- カーネル BTS の深追い。
- 最新のカーネルは新しいドライバや機能が使えるから。
- ドライバを組み込みにして、起動の高速化。
- ドキドキ感を味わうことができる。
- 無駄に CPU を使いたい。(反エコ)

カーネルをリコンパイルすることはメリットだけではなく、デメリットもあります。

- 失敗したら動かなくなるかもしれない。
- コンパイルに CPU リソースを食いすぎる (CPU が遅いため)。

たぶん、この文章を読んでいる人たちは前者の予備軍なので、デメリットは気にせずに突き進んでいけると思います。

^{*8} i386 で約 4000 の設定項目がある

8.3 今回作ったプログラム

今回作ったプログラムは、システム情報を元にシステムに必要なカーネルモジュールが組み込み指定に変換されたカーネルコンフィグファイルを出力するというものです。

- 使うカーネル
 - Debian で提供しているカーネル (lenny では 2.6.26)
- 入力するデータ
 1. カーネルソースコード
 2. 動いているカーネルのコンフィグファイル
 - /boot/config-2.6.26-1-xxx
 3. システム情報
- 出力されるデータ
 - システム情報を元にシステムに必要なカーネルモジュールが組み込み状態になっているカーネルコンフィグファイル

8.3.1 システム情報の取得方法

今回のキモはどのようにして、システム情報を取得するか、にかかっています。今動いているカーネルから得られる情報は以下のものが考えられます。

コマンド	内容
dmidecode	BIOS からシステム情報を出力する
lspci	PCI の情報を出力する
lsusb	USB の情報を出力する
dmesg	カーネルデバッグメッセージを出力する
lsmod	ロードしてるモジュールを出力する

表 1 起動しているカーネルから得られる情報

これらの中で信用できて簡単に扱えるものは `lsmod` でしょう。理由はロードしているモジュール = 現在の Linux システムに必要なものなので、わかりやすいからです。よって、今回は `lsmod` の出力を利用することにしました。^{*9}

8.3.2 簡単な流れ

以下に簡単な処理の流れを説明します。

1. データとして、カーネルソースコードへのパス、動作しているカーネルコンフィグファイル、`lsmod` の出力結果を指定する。
2. `lsmod` からロードしているドライバモジュール一覧を取得する
`lsmod` を実行すると、以下のような内容が出力されます。

```
$ lsmod
Module          Size Used by
i915             25280 2
drm             65256 3 i915
ipv6            235300 10
rfcomm          28272 2
l2cap           17248 9 rfcomm
.....
```

^{*9} カーネルの自動認識がどこまで信用できるかはこの際無視します。

3. ドライバモジュール名 と modinfo コマンドから ドライバモジュールのパスを取得する。

modinfo コマンドを使うと、指定したドライバモジュール名の情報を取得することができます。-n オプションを使うと、ドライバオブジェクトファイルのパスが出力されます。

```
$ modinfo -n i915
/lib/modules/2.6.26-1-amd64/kernel/drivers/char/drm/i915.ko
```

上の結果を例にすると、/lib/modules/2.6.26-1-amd64/kernel/ 以下と カーネルソースコードのパス構造は同じなため、ドライバモジュール名 i915 の Makefile のあるパスは drivers/char/drm/Makefile になります。また、ドライバオブジェクトファイルは i915.ko であることが分かります。

4. 上で取得した Makefile へのパスとドライバオブジェクトファイル名より、対象になるドライバコンフィグ名を取得する。

ドライバオブジェクトファイル (i915.ko) とドライバモジュール名 (i915) は必ずしも一致するとは限らないので、ドライバモジュール名を使って、ドライバコンフィグ名を検索します。^{*10}

```
.....
obj-$(CONFIG_DRM_I830) += i830.o
obj-$(CONFIG_DRM_I915) += i915.o <- これ
obj-$(CONFIG_DRM_SIS) += sis.o
.....
```

5. 取得したドライバコンフィグ名を保存する。
6. 動作しているカーネルコンフィグファイルのドライバコンフィグを書き換える。

正規表現を使って書き換えると、以下のようになります。m はモジュールを意味し、y は組み込みを意味します。

変更前

```
.....
CONFIG_DRM_I830=m
CONFIG_DRM_I915=m
CONFIG_DRM_MGA=m
.....
```

変更後

```
.....
CONFIG_DRM_I830=m
CONFIG_DRM_I915=y <- 書き換え
CONFIG_DRM_MGA=m
.....
```

7. 変更したものをファイルに出力する。

8.3.3 実際に使ってみる

今回作成したソフトウェアは以下のように使います。ちなみに、lsmod の出力ファイルを指定しない場合は、プログラムの中で自動的に取得します。

```
$ moge -h
moge - Script to Kernel Module Enabler from lsmod command output

Copyright (C) 2008,2009 Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
Usage: moge [options]
  -c, --configfile <file>   Kernel config file name
  -k, --kernel               Kernel source path
  -o, --output <file>       outout file
  -l, --lsmod                lsmod command output file
  -h, --help                 display this help screen and exit
  -v, --version              show the version and exit

By Nobuhiro Iwamatsu <iwamatsu@nigauri.org>
$ moge -o sage -c config-2.6.26-1-686 -l lsmod.list -k /usr/src/linux-2.6-2.6.26/
```

*11

^{*10} 例えば snd_hda_intel

^{*11} プログラム名が mogeなのはまだ決めていないためです。

8.4 作成されたコンフィグファイルを使ってカーネルをコンパイルする

さっそく、作成されたコンフィグファイルを使ってカーネルをコンパイルしてみましょう。カーネルコンパイルの方法は以下の通りです。

```
$ sudo apt-get update
$ sudo apt-get install linux-source-2.6.26 kernel-package
$ cd /usr/src/linux-source-2.6.26
$ make oldconfig
$ fakeroot make-kpkg --revision=yourpc00 kernel_image kernel_header
$ ls ../
linux-image-2.6.26_yourpc00_i386.deb
linux-headers-2.6.26_yourpc00_i386.deb
.....
```

8.5 変更結果

8.5.1 lsmod の結果

eeePC で試してみても、どれくらい変わったのか調べました。

変更前 61 モジュール

Module	Size	Used by
i915	25280	2
drm	65256	3 i915
ipv6	235300	10
rfcomm	28272	2
l2cap	17248	9 rfcomm
bluetooth	44900	4 rfcomm,l2cap
psmouse	32336	0
uvcvideo	45704	0
serio_raw	4740	0
compat_ioctl32	1312	1 uvcvideo
videodev	27520	1 uvcvideo
v4l1_compat	12260	2 uvcvideo,videodev
i2c_i801	7920	0
i2c_core	19828	1 i2c_i801
pcspkr	2432	0
itCO_wdt	9508	0
snd_hda_intel	324248	0
rng_core	3940	0
snd_pcm_oss	32800	0
snd_pcm	62596	2 snd_hda_intel,snd_pcm_oss
snd_mixer_oss	12320	1 snd_pcm_oss
video	16432	0
output	2912	1 video
battery	10180	0
ac	4196	0
snd_seq_dummy	2660	0
eeepc_laptop	7440	0
button	6096	0
at12	22872	0
snd_seq_oss	24992	0
snd_seq_midi_event	6432	1 snd_seq_oss
snd_seq	41456	5
snd_seq_dummy,snd_seq_oss,snd_seq_midi_event		
snd_timer	17800	2 snd_pcm,snd_seq
snd_seq_device	6380	3 snd_seq_dummy,snd_seq_oss,snd_seq
snd	45604	8 snd_hda_intel,snd_pcm_oss,snd_pcm,snd_mixer_oss,snd_seq_oss,snd_seq,snd_timer,snd_seq_device
soundcore	6368	1 snd
snd_page_alloc	7816	2 snd_hda_intel,snd_pcm
intel_agp	22332	1
agpgart	28776	3 drm,intel_agp
evdev	8000	6
ext3	105512	1
jbd	39444	1 ext3
mbcache	7108	1 ext3
usb_storage	75936	0
sd_mod	22200	3
ata_piix	14180	2
ahci	23176	0
ata_generic	4676	0
libata	140384	3 ata_piix,ahci,ata_generic
scsi_mod	129356	3 usb_storage,sd_mod,libata
dock	8304	1 libata
ide_pci_generic	3908	0 [permanent]
ide_core	96168	1 ide_pci_generic
ehci_hcd	28428	0
uhci_hcd	18672	0
usbcore	118160	5 uvcvideo,usb_storage,ehci_hcd,uhci_hcd
thermal	15228	0
processor	32576	2 thermal
fan	4164	0
thermal_sys	10856	4 video,thermal,processor,fan

変更後 0 モジュール

Module	Size	Used by
--------	------	---------

8.5.2 bootchart

bootchart でどれくらい起動が速くなったか、調べてみました。

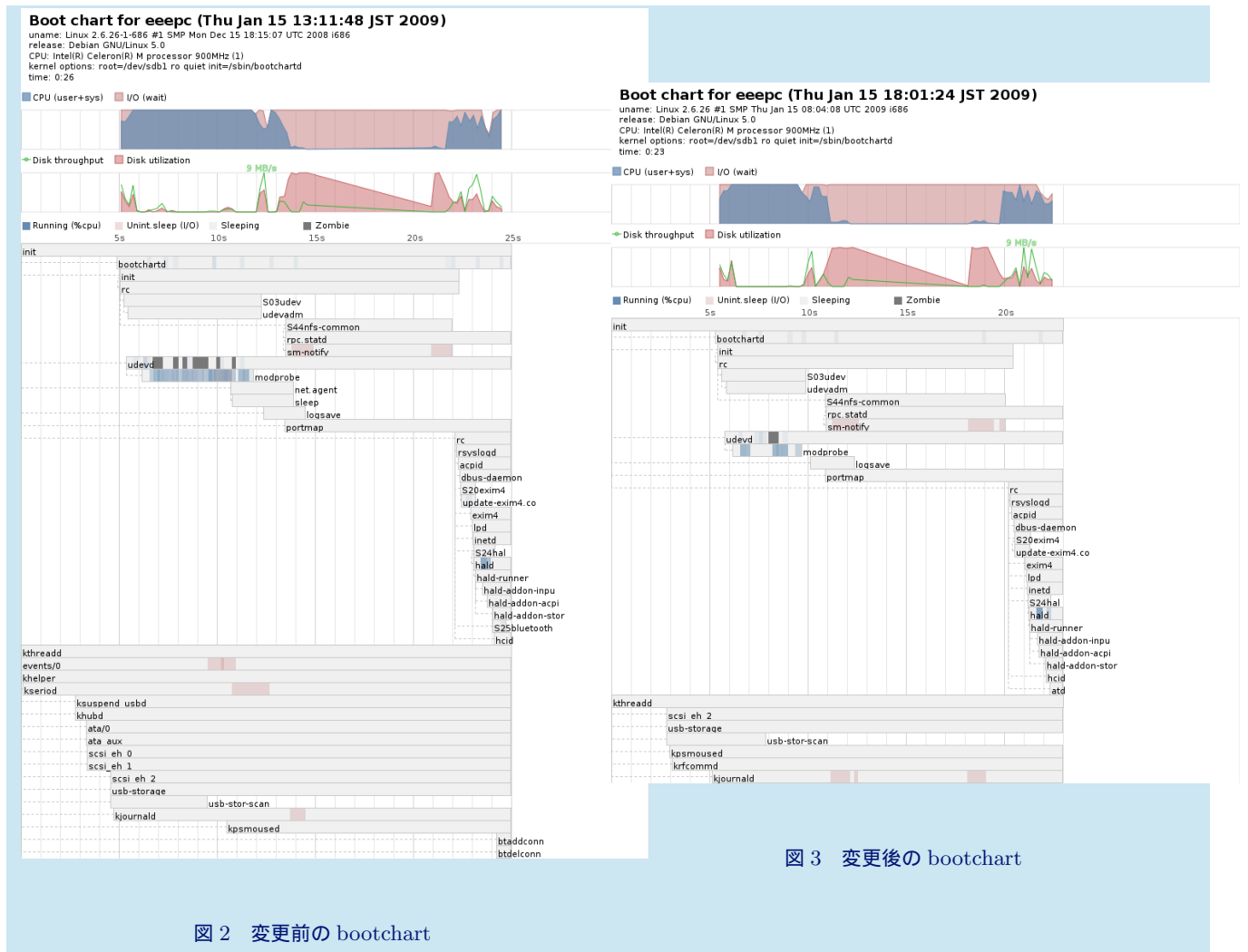


図 2 変更前の bootchart

図 3 変更後の bootchart

上の起動チャート図より、起動時のモジュールの読み込みがなくなり、三秒ほど速くなっていることが分かります。

8.6 今後の予定

このソフトウェアは Perl の勉強のために作っていましたが、今後の展開としては以下のことを考えています。

1. make-kpkg に入れる？
make-kpkg は Perl で作られているので入れやすいかもしれません。
2. パッケージ化？
Debian パッケージ化するとユーザは喜ぶかもしれません。
3. カーネルコンパイル Web サービスの提供？
lsmold の出力が分かれば Debian カーネルはコンパイルが容易なので、サービス化するとユーザは喜ぶかもしれません。が、それぐらい自分でやれという感じです。
4. ドライバオブジェクトファイルとドライバモジュール名が一致しないやつを直す。
知らないとハマるので、一致しておいた方がいいと思っています。

9 黒 MacBook の Lenny/Sid 64bit 化でのハマりポイント

まえだこうへい



年末のネタ^{*12}を実現するために、冬休みを利用して 32bit Sid 環境だった黒 MacBook を 64bit Sid で再構築しました。インストール自体はいつものとおりの作業なので、皆さんもいつもやっていらっしゃることで面白みもないのですが、年明けにリリースされていた Lenny の Snapshot イメージでブートローダに LILO を選択し、インストール後に起動させると Kernel Panic になってしまい、いきなり起動できない問題に遭遇します。今回はその回避方法について^{*13}説明します。

9.1 用意したインストールイメージ

ご存知のとおり、MacBook を 64bit でインストールするには、amd64 版のインストールイメージが必要です。今回は、Lenny のスナップショット^{*14}を使用しました。

9.2 簡単なインストール手順

MacBook への Debian のインストールに関する情報は、過去の Debian 勉強会の資料や、Debian Wiki に公開されています。今回もそれらと同様の手順です。

1. expert install での起動。
2. 日本語ロケール、米国キーボードを指定。
3. パーティションの指定し直し。^{*15}
4. 基本パッケージのインストール。
5. シェルモードへ移行。
6. gptsync, refit パッケージのインストール、gptsync の実行。
7. apt-line の変更、apt-get {update,upgrade,dist-upgrade} の実行。
8. 再起動。

9.3 起動させると...

次のようなメッセージを吐いて、Kernel Panic になります。

^{*12} 2008 年 12 月の勉強会で発表した、ヨメを Debian GNU/Linux Sid ユーザにする。

^{*13} バグの修正ではありません。新婚旅行前日の数時間しか冬休みは時間を割けませんでした…。orz

^{*14} Debian GNU/Linux testing "Lenny" - Official Snapshot amd64 NETINST Binary-1 20090104-09:09

^{*15} LVM でボリュームを作っているパーティションはパーティションを切り直せない問題があります。

```
RAMDISK: Couldn't find valid RAM disk image starting at 0.
List of all partitions:
No filesystem could mount root, tried:
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(8,4)
```

実は、インストール直後に Kernel Panic となるのは今回が初めてではありません。MacBook Air を 64bit 化した際にも同じ現象が発生しました。^{*16}しかし、MacBook Air の時は、レスキューモードで起動し、Kernel を再構築してインストールしたら再発しなくなりました。ところが今回は Kernel を再構築しただけでは解決できません。困ったものです。

9.4 回避方法

困ったので、ググってみたところ、同じ現象に対しての回避策も公開されていました。^{*17}簡単に回避策をまとめると次の手順です。

1. レスキューモードで起動します。
2. シェルモードになり、/target へ chroot します。
3. カーネルソースを展開します。
4. /boot 以下の該当の kernel config をカーネルソースツリーにコピーします。
5. kernel を構築し、インストールします。

```
REVISION=$(date +%Y%m%d.%H%M)
make-kpkg --initrd --revision $REVISION kernel_image
dpkg -i ../kernel-image-2.6.26_20090105.2330_amd64.deb
```

6. /ramdisk ディレクトリを作り^{*18}、initrd を展開します。

```
mkdir /ramdisk
cd /ramdisk
zcat /boot/initrd-2.6.26 | cpio -i
```

7. 先ほどインストールした kernel をアンインストールします。

```
dpkg --purge kernel-image-2.6.26
```

8. .config の "INITRAMFS_SOURCE" を書き換え、-initrd オプションなしで kernel をリビルドします。

```
sed -i 's:INITRAMFS_SOURCE="":INITRAMFS_SOURCE="/ramdisk:'.config
make-kpkg --revision $REVISION kernel_image
```

9. できた kernel パッケージをインストールします。
10. lilo.conf の "initrd=/initrd.img" をコメントアウトし、lilo を書き込みます。

これで、kernel panic を起こさず、ちゃんと起動できるようになりました。ただし、今のところ、新たなバージョンの kernel を構築する度、同じ手順を踏まなければなりません。

9.5 もっと簡単な回避方法。

lilo なんかをやめて、grub2 にしてしましましょう。おあとがよろしいようで。

^{*16} <http://d.hatena.ne.jp/mkouhei/20080713/1215913997>

^{*17} <http://linux.derkeiler.com/Mailing-Lists/Debian/2008-11/msg01918.html>

^{*18} ディレクトリ名は任意。/ 直下に所有者、グループを root:root で作ります。



Debian 勉強会資料

2009年1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）
