第 87 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

2012 年 4 月 21 日

特集1: Debianでのnode入門

特集2: Android機でDebian

特集3: 月刊Debhelper

# 1 はじめに

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?

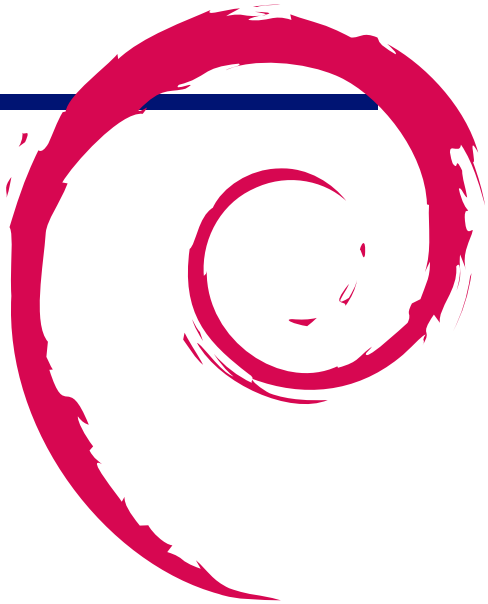
Debian 勉強会の目的は下記です。

- 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

野島 貴英

今回の事前課題は以下です:

1. Debian で触ったことのあるウェブアプリケーションフレームワーク一つを簡潔に教えてください。
2. 近々作成してみたいと思っているウェブアプリケーションを教えてください。

この課題に対して提出いただいた内容は以下です。

## 2.1 koedoyoshida

1. Debian で触ったことのあるウェブアプリケーションフレームワーク一つを簡潔に紹介してください。
   Greasemonkey
2. 近々作成してみたいと思っているウェブアプリケーションを教えてください。
   なし

## 2.2 noritadak

1. Ruby on Rails。でもかなり昔です。
2. Web アプリケーションと言えるのか微妙ですが、ブラウザ上で使えるシェルを個人的に作りたいです。

## 2.3 dictoss(**杉本　典充**)

1. python の Django はなかなかいい感じですが python3 に現状非対応というところ。 cakephp もほんの少し試しましたがよくわらかず挫折。
2. 自宅の本を管理するシステムがほしい。また、ほしいジャンルの本を自動で探してくれるボットつきがいいな。

## 2.4 henrich

1. ウェブアプリケーションフレームワーク、一つも触ったことがございません…
2. 作成してみたいというのは特には無いですが、将来的には Django 辺りを触ってみたいとは思っています。

## 2.5 ikeomasa

- django

## 2.6 MATOHARA

1. Debian で触ったことのあるウェブアプリケーションフレームワーク一つを簡潔に紹介してください。
   フレームワークは利用したことがありません． Perl の Catalyst を使ってみたいなと思っています．
2. 近々作成してみたいと思っているウェブアプリケーションを教えてください。
   最近写真の exif 情報を編集するものを作りたいなと思っています．
   #アップロードした写真の exif 情報を表示してカメラのシリアル，撮影者，位置情報などを選択して削除してダウンロード．

## 2.7 beatenavenue

1. pukiwiki も web アプリに入りますでしょうか・・・。グループウェアみたいな使い方ができないかなと思って試していましたが、仲間うちで wiki 文法の評判が非常に悪く今は触っていません。
2. 作成したい web アプリは今のところありません。

## 2.8 **本庄**

1. CakePHP を使ったことがあります。 pear に依存しないということで選択しました。
2. 最近、テレビ番組をチェックする自分用の適当なスクリプ

トを作成しました。

### 2.9 yamamoto

1. 全く触ったことがありません。
2. ウェブアプリケーションって、なんすかね? おいしいの?

### 2.10 まえだこうへい

1. Sinatra 使ってます。 Debian じゃなくて Ubuntu ですが…。
2. 近々、というか 1 で OS インストール用ツールの開発やってます。

### 2.11 鈴木崇文

1. 書籍にのっていたサンプルを動作させるために pylons を少しだけ使ったことがあります。現在は pyramid を代わりに使用することが推奨されているみたいです。ところで、フレームワークを使って web 開発する場合はどうやって debian パッケージを利用すべきなんでしょうか。いままではパッケージを使用せずに直接ダウンロードして使っていました。
2. 位置情報を使った Web アプリを作ってみたいです。

### 2.12 emasaka

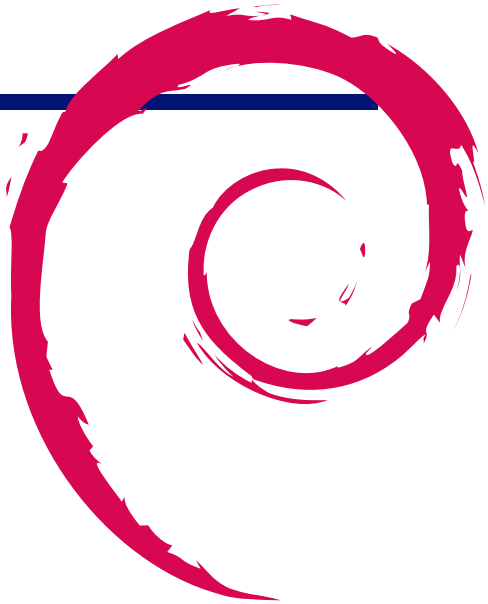
1. Bash on Rails 作ったわー 3 年ぐらい前に作ったわー（ ミサワ）
   （ と課題に書くために参加します）
2. 仕事でちょっとした Web アプリを提案中

### 2.13 大森　俊秀

1. django
2. かんたん週報アプリ

### 2.14 野島　貴英

1. 古い話ですが、 Perl の Catalyst ぐらいでしょうか?(今だと mojiolicious なんだろうか...。) 紹介については... べ、勉強してきますー。
2. サイト巡回して何かデータ取ってきて自動的に分類／似ているものを探す個人利用専用のサイトを作ってみたいなぁ ... あ、これだとウェブフレームワークよりも、機械学習とかのテーマ... ごふっ ...



# 3 最近の Debian 関連のミーティング報告

野島 貴英

## 3.1 東京エリア Debian 勉強会 85 回目報告

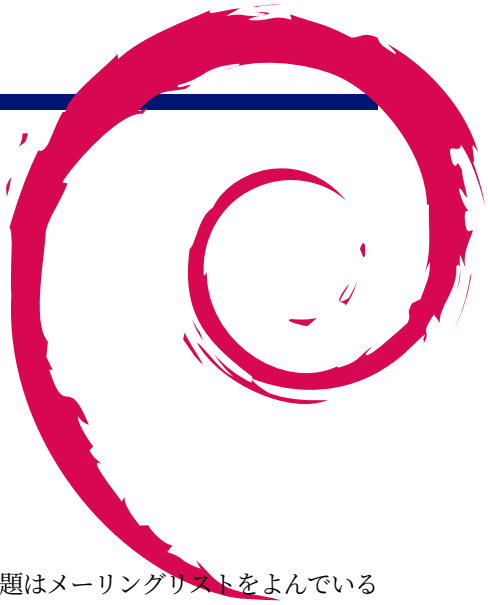
2 月の東京エリア Debian 勉強会は、三栄町生涯学習館 視聴覚室で行われました。

当日、福岡 Debian 勉強会が開かれ、そちらに出席者された方が多かった為か、出席者 5 人での開催となりました。発表内容としては、 debian で KDE4.7.4 用の開発環境を作り簡単なパッケージを作るまでの話、月刊 Debhelper では dh_dpatch_patch コマンド、 dh_autotools-dev_updateconfig コマンド、最後に KDE 開発に使われる cmake についてあれこれの話が行われました。参加者にソフトウェア開発関係者の方が多かった為か、 autotools の使い方をはじめ、ソフトウェア開発について話がいろいろ盛り上がったのが印象的でした。

## 3.2 東京エリア Debian 勉強会 86 回目報告

3 月の東京エリア Debian 勉強会は OSC 2012 Tokyo/Spring の会場で岩松さんにて行われました。本来東京エリア Debian 勉強会は開発者寄りの話題を扱うのですが、今回はユーザ向け企画として Debian の Apache サーバーの構成と Debian ならではの使い方に関する発表が行われました。最近の W3Techs の発表によると、世界で利用されている Linux ベースの Web サーバーでは 1 位,2 位( 2012/1 時点では 1 位）を僅差で争う状況との事[*1]です。今後も益々数が増えているとの事ですので、 Debian ユーザの期待に答えるべく開発の方も頑張りましょう。

[*1] http://w3techs.com/blog/entry/debian_is_now_the_most_popular_linux_distribution_on_web_servers



# 4 Debian Trivia Quiz

野島 貴英

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。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容と Debian Project News からです。

問題 1. 今年度の Debian JP 会長は誰か?

☐ A Kouhei Maeda
☐ B Nobuhiro Iwamatsu
☐ C Junichi Uekawa

問題 2. Debian.org DPL 選挙は誰が立候補したか

☐ A Stefano Zacchiroli
☐ B Nobuhiro Iwamatsu
☐ C Kouhei Maeda

問題 3. Debconf12 の suponsord な参加の締切りはいつ

☐ A 4 月末
☐ B 5 月 15 日
☐ C 5 月末

問題 4. 大統一 Debian 勉強会での発表の公募 (CFP) はいつが締切り?

☐ A もう過ぎた
☐ B 4 月末
☐ C 4 月 22 日

問題 5. experimental 版の apache のパッケージのバージョンはいくつ?

☐ A 2.3
☐ B 2.4.0
☐ C 2.4.2

問題 6. Debian Edu はまたの名を何というでしょう?

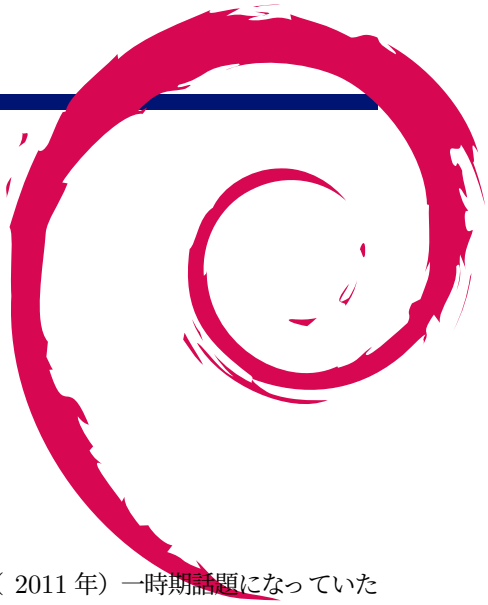
☐ A emdebian
☐ B Scientific Linux
☐ C Skolelinux

問題 7. 3/30 に Debian Project が加盟した団体の名前は?

☐ A OSC
☐ B OSI
☐ C ETF

問題 8. Debian Project の gobby サーバーとして gobby.debian.org がアナウンスされました。ところで gobby って何?

☐ A 旧ソ連で開発された諜報活動用ソフトウェア
☐ B churro の代替サーバ
☐ C エディタの名前



# 5 Debian での node 入門

上川純一

JavaScript でプログラムをガリガリ書きたいとおもったことはありませんか? 昨年( 2011 年) 一時期話題になっていた node を Debian で試してみましょう。 node[*2]はイベントベースのサーバサイド JavaScript エンジンとフレームワークです。何に使えるかというと、 JavaScript でウェブサーバが書きやすくなっています。シェルスクリプトでコードを書く代わりに node を使って JavaScript を使うこともまぁ出来ますがウェブサーバが便利になるのが一番大きいと思われます。特徴はシングルスレッドなんだけどほとんどすべての I/O 処理を非同期イベント処理によって実現することによって、たくさんのリクエストを効率よく処理するあたりでしょうか。

node のパッケージは squeeze にはないですが、 wheezy にはすでに入っています。インストールは簡単:

```
# apt-get install nodejs
```

Node 本体は nodejs というパッケージ名で入っています[*3]が、関連するモジュールパッケージの名前は node-ではじまるようになっています。

node コマンドを実行すると JavaScript インタープリタの REPL インタフェースが起動します。ここでコマンドラインから JavaScript を適当に実行できるようです。この時点では単なる V8 のコマンドラインインタフェースですね。

```
$ node -v
v0.6.12
$ node
> console.log('hello world')
hello world
```

node のバージョン番号ですが、 node の 2012 年 4 月 1 日時点の安定版の最新版は 0.6 系列で、 0.7 系列は開発版という位置づけのようです。 0.8 系列がリリースされたらまた Debian の node も更新されるでしょう。

まとめると次のようになっています。

- 0.4: 2012 年 1 月まで Debian sid に入っていた安定版
- 0.6.12: 2012 年 4 月時点での最新安定版 (stable)
- 0.7.x: 開発版 (unstable)
- 0.8.x: 多分近い将来リリースされるだろう安定版

## 5.1 Debian 流儀での Node モジュールパッケージインストール

Debian パッケージで提供されている node のモジュールパッケージを使ってみましょう。

[*2] マニュアルなどには node と記述されていますが、通称は node.js のようです

[*3] すでに node というパッケージが存在するから node という名前がつけられなかったのだと思われます。

### 5.1.1 Node で CLI ツールを作ってみる

とりあえず、 node でコマンドラインインタフェース( CLI) のツールをつくってみたい、ので node cli をインストールして適当にコードを書いてみるという場面を想定してみます。 cli モジュールは Debian パッケージになっているので以下でインストールできます。

```
# apt-get install node-cli
```

とりあえず無駄に平均を計算してみるコードを書いてみました。

```
// Command-line tool to sum the stdin items.
var cli = require('cli');

cli.withStdinLines(function(lines, newline) {
    var sum = 0;
    var count = 0;

    for (var i = 0; i < lines.length; ++i) {
        console.log(lines[i]);
        if (lines[i] != '') {
            sum += parseInt(lines[i]);
            count ++;
        }
    }
    console.log('sum: ' + sum + ' avg: ' + sum / count);
});
```

コマンドラインで適当に実行してみたところ、結果が表示されました。

```
$ node sum.js < testdata.txt
10
15
200
8

sum: 233 avg: 58.25
```

## 5.2 npm – Node のパッケージ管理システム

Node のモジュールは npm で管理されています。 Debian パッケージになっていないパッケージなどは、 npm コマンドを利用して直接インストールすることも可能です。 Perl でいう CPAN、 TeX でいう CTAN のようなもののようです。 Debian パッケージを使うべきか npm を使って導入するべきか悩ましいところですが、 Debian パッケージになるまでにはどうしてもタイムラグがあるので、最新のコードをつかって開発する場合には npm を利用することになると思われます。ある程度こなれてきたらないパッケージは ITP するのがよいでしょう。

Debian の提供するモジュールパッケージは/usr/lib/nodejs 以下にインストールされますが、 Debian パッケージの npm を利用する場合は /usr/local/lib/nodejs 以下に入ります。

で、喜び勇んで手元で実行したところ Exception をはいて終了しました。どうやら node 0.6.2 に対応していない古いバージョンの npm (Node 0.4 系列ではうごいたはず) が現在パッケージされており、動かなくなっている模様です。 Bug#622628[*4]

## 5.3 npm のアップストリーム版をインストールしてみる

npm が使えないのは不便なので、 Debian パッケージになっていない node モジュールをインストールする方法として Debian パッケージではない npm を利用する方法を紹介します。

node 0.6 系列に対応している npm は 1.1 です。 npmjs[3] サイトからインストーラをダウンロードして実行します。 npm がインストールできたらモジュールは npm install コマンドでインストールできるようになります。

npm は sudo 前提で設計されているようです。 sudo で root 権限に昇格するのはビルド時に nobody 権限に切り替えるのに使うようです。

[*4] http://bugs.debian.org/622628

npm パッケージはシステムグローバルにもパッケージローカルにもインストールできますが、一般ユーザ権限でプロジェクトローカルに利用するためにパッケージをインストールすることを基本として設計されているようです。

`sudo npm install` **パッケージ名**だとカレントディレクトリ以下に sudo を発行したユーザ権限でインストールするようです。[*5]`sudo npm install -g` **パッケージ名** だと `/usr/lib/node_modules` 以下にインストールするようですが、権限は nobody のままです[*6]。

npm は開発者視点で便利なように設計されているようで、個人的におもしろいなと思ったのは -g をつけずに `sudo npm install` だけすると現在のディレクトリにあるプロジェクトの依存しているパッケージをカレントディレクトリにインストールするという挙動になることです。依存しているパッケージがどんどん変更されている熱いプロジェクトである node っぽい感じがします。 npm の作者のウェブサイトを読んでいると、システムワイドでインストールするのは CLI ツールなどに必要な時に限って、ウェブサイトにデプロイする用のコードではモジュールはプロジェクトのディレクトリにインストールすることを推奨すると説明しています。

```
# apt-get install nodejs nodejs-dev
# curl http://npmjs.org/install.sh | sh
$ sudo npm install -g express ejs socket.io
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/ejs
npm http GET https://registry.npmjs.org/socket.io
npm http 304 https://registry.npmjs.org/socket.io
npm http 200 https://registry.npmjs.org/ejs
npm http GET https://registry.npmjs.org/ejs/-/ejs-0.6.1.tgz
npm http 200 https://registry.npmjs.org/express
  .
  .
/usr/bin/express -> /usr/lib/node_modules/express/bin/express
ejs@0.6.1 /usr/lib/node_modules/ejs
express@2.5.8 /usr/lib/node_modules/express
├── qs@0.4.2
├── mkdirp@0.3.0
├── mime@1.2.4
└── connect@1.8.6
socket.io@0.9.2 /usr/lib/node_modules/socket.io
├── policyfile@0.0.4
├── redis@0.6.7
└── socket.io-client@0.9.2
```

ディレクトリ構成をまとめました。

表 1　Debian パッケージおよび npm でインストールされる場所

| | ディレクトリ |
|---|---|
| Debian パッケージ | `/usr/lib/nodejs` |
| Debian の npm | `/usr/local/lib/nodejs` |
| npm install -g | `/usr/lib/node_modules` |
| npm install | `./node_modules` |

マニュアルなどは npm コマンドを実行すると表示されるヘルプが充実しています。

```
$ npm help
$ npm help npm
$ npm help install
```

### 5.3.1　npm を使っている場合のプログラム実行

カレントディレクトリにインストールしたときはよいのですが、そうではない場合は、`/usr/lib/node_modules` 以下にインストールされても Debian の nodejs の標準のモジュールパスに含まれていないためモジュールがロードされません。

ひとつの回避策としてはモジュールを探しに行く PATH を追加するという方法があります。 `NODE_PATH` 環境変数を指定すれば追加でロードされるようになります。

[*5] ビルド自体は nobody 権限で行うようです
[*6] 挙動としてはおかしいので操作を間違っているかバグのように思われる

```
$ NODE_PATH=/usr/lib/node_modules node ./program.js
```

### 5.3.2 パッケージのメタデータ: package.json

推奨されているのはパッケージに必要なモジュールを package.json に記述して、 npm link コマンドを実行すればパスの追加は必要なくなります。

```
$ npm link
```

とすると必要なモジュールをプロジェクトのディレクトリの./node_modules にリンクしてくれるということでした。[*7][*8]

npm のメタデータは ./package.json です。 npm help json (man npm-json.1) に詳しく説明されています。す。とりあえずは name/version フィールドさえあればよくて、 dependencies を追加すると npm install や npm link コマンドで利用してくれるようです。

インストールに必要なファイルの一覧や、 node-waf の実行方法などが記載されているのでこれさえあれば Debian パッケージを自動で生成することも可能な気がします。

手元で作成してみた npm link のためだけの最低限な内容の package.json を紹介します

```
{
    "name": "aptserver",
    "version": "v0.0.1",
    "dependencies": {
        "cli": "",
        "express": "",
        "ejs": ""
    }
}
```

## 5.4 とりあえず簡単なサーバを書いてみた

apt-cache search をして出力を返すだけの簡単なサーバを書いてみました。イベントドリブンな部分としては、 HTTP request のハンドラーを app.get() で登録している部分と、 apt cache の出力を aptCache.stdout.on で登録したハンドラーで取得して aptCache.on exit ハンドラーで終了したら HTTP レスポンスを返すようになっている部分でしょうか。

書いてみて気づきましたが、いまいち node を使うメリットが出てないきもします。

[*7] マニュアルにはシンボリックリンクをすると書いているけど、モジュールはハードリンクしてました。

[*8] 逆方向に現在作業中のパッケージを/usr/lib/node_modules/以下にシンボリックリンクしてくれます。作業した内容がすぐに反映するので便利といえば便利。

```
/*
 * A simple server which serves apt cache search results.
 */
var child_process = require('child_process');
var cli = require('cli');
var url = require('url');
var ejs = require('ejs');

cli.parse({
    port: ['p', 'HTTP server will listen on this port.', 'number', 8088]
});

cli.main(function cliMain(args, options) {
    var express = require('express');
    var app = express.createServer();
    // set view options.
    app.set('view engine', 'ejs');
    app.set('view options', { layout: false });
    app.set('views', __dirname + '/views');

    // Set up routes.
    app.get('/', getSlash);
    app.get('/search', getSearch);

    console.log('Start listening on http://localhost:' + options.port + '/');
    app.listen(options.port);
});

/** handler for '/' request */
function getSlash(request, response) {
    response.render('index.ejs');
}

/** handler for '/search?' request */
function getSearch(request, response) {
    var query = request.query.q;
    var aptCache = child_process.spawn('apt-cache',
                                       [ 'search', query ]);
    /** Output of apt-cache search, to be used for response. */
    var responseString = '';
    aptCache.stdout.on('data', function handleAptCacheStdout(data) {
        responseString += data;
    });
    aptCache.stderr.on('data', function handleAptCacheStderr(data) {
        responseString += data;
    });
    aptCache.on('exit', function handleAptCacheExit(exitCode) {
        // apt-cache finished executing, send repsponse back to the server.
        response.render('search.ejs',
                        {locals:{query: query,
                                 response: responseString}});
    });
}
```

ejs HTML テンプレートファイル `views/search.ejs` はこんな内容になりました。

```
<!-- -*- html; -*- -->
<html>
  <body>
    <form
      method=GET
      action='/search'>
      <input type=text name=q></input>
      <button>search</button>
    </form>
    <h3>Search for: <%= query %></h3>
    <pre>
<%= response %>
    </pre>
  </body>
</html>
```

追加モジュールは express, ejs, cli を使いました。それぞれを簡単に紹介すると以下です

- express: ウェブアプリケーションフレームワークとして node で最もポピュラーなモジュール、リクエストベースのルーティングなどを担当。
- ejs: HTML テンプレートエンジンとしておそらく最もポピュラーなモジュール。
- cli: コマンドラインオプションをパースしてくれるモジュール。

### 5.5 最後に

今回は node を Debian で使う方法を紹介してみました。モジュールのパッケージングが更新においついていない感じなのでもうしばらくしたらまた良くなっているかもしれませんが、そうこうしているうちに node 0.8 がリリースされてしまいそうです。 node の頻繁な更新とそれによってモジュールが複数バージョン必要になってくる現状から、 nave / nvm などの環境管理ツールで複数バージョンをインストールして利用するということもよくやられているようです。この資料が Debian で node を活用するきっかけ、およびパッケージ化活動の一助になれば幸いです。

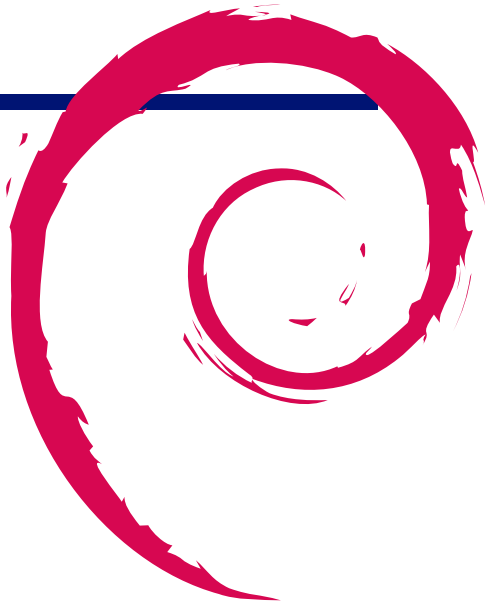
## 参考文献


[1] Node.js v0.6.12 Manual & Documentation `/usr/share/doc/nodejs/api/index.html`

[2] Alioth の pkg-javascript-devel メーリングリスト `http://lists.alioth.debian.org/pipermail/pkg-javascript-devel/`

[3] NPM ホームページ `http://npmjs.org/`

[4] nodejs for Debian `/usr/share/doc/nodejs-dev/README.Debian`



# 6 Android 機で Debian

野島 貴英

## 6.1 はじめに

携帯電話、タブレット型 PC など、高性能の情報端末を持ち歩くのが一般的になってきました。ここでは、これら情報端末のうち、 Android OS を搭載した情報端末に Debian を入れ、 Debian 開発者の環境を築いてみようとした事について述べます。

## 6.2 材料: Android 端末 Barnes & Noble Nook Color について

手頃な Android 端末として、たまたま手元に Barnes & Noble 社 (以下 B&N) の Nook Color という製品[*9]があります[*10]。値段も日本で輸入した場合、 2 万円前半〜後半ぐらいで安く、電子書籍ビューアということもありデザインも小型でそれなりに薄く軽量です。これは元々、 PDF にしたカラーの本( 漫画も) を持ち歩きながら読めればと思って買っていたものでした。

これが Debian の開発環境としても動いたら素敵と思い、こちらを早速利用する事にします。

※実は、自分はフィーチャーフォン( ガラケーともいう) しか持っておらず、 Android 端末はこれしか持ってなかったことは秘密です。

## 6.3 Android 端末の Debian 動作の方針

Android 端末を利用して Debian 関係のディストリビューションの OS を動作させる場合、後に述べる 2 つの方針が取れます。

### 6.3.1 方針 1: Android 端末の機能を最大限活用する

本方針は、 Android 端末はベースの OS が基本的に Linux であることを最大限利用します。

ここで、

1. Android 端末の CPU にあわせた Debian 関係の Linux のファイルシステムを/( ルート) ファイルシステムから SD カードや、内臓メモリへ用意します。
2. Android 端末に Android アプリの VNC Viewer[*11]を搭載しておきます。
3. 何らかの方法で Android 端末上の管理権限を奪取した状態で、先ほど用意した/ファイルシステムをマウントし、chroot します。これで Debian のバイナリが動作できるようになります。

[*9] http://www.barnesandnoble.com/p/nook-color-barnes-noble/1100437663

[*10] 製品の写真は権利関係がよく判らないので割愛させてください。詳しくは先の URL 参照。

[*11] http://code.google.com/p/android-vnc-viewer/

4. 最後に Debian のバイナリを利用して vncserver を立ち上げ、 Android アプリの VNC Viewer で 127.0.0.1:5901 に接続し、 Debian を利用します。

という方法がよく利用されます。

この方法は、 Android 端末で、管理権限さえ奪取できていれば Debian 関係の Linux ディストリ意ビューションを動作させる事ができます。さらに良いことに、本体の Android 端末が最初から搭載している Wi-Fi ネットワークもそのまま利用できるるため、非常に都合が良いです。実際、自分は未評価ですが、これらの諸々の手続きを全部アプリに詰め込んでしまったものに、

- ubuntu を動作させる Android アプリ： ubuntu instller free `https://play.google.com/store/apps/details?id=com.zpwebsites.ubuntuinstall&hl=ja`
- Debian を動作させる Android アプリ: Lil' Debi `https://github.com/guardianproject/lildebi/wiki`

などがある模様です。

### 6.3.2 方針 2: Android OS を使わずそのまま Debian をブートする

Android 端末は大抵 ARM ベースの CPU で動いてます。 Debian も ARM ベースのバイナリを用意しています。これはつまり、うまくカーネル/アプリケーションを Android 端末のハードの仕様に合わせることができ、ブートさせる事ができれば、 Android OS の代わりに Debian をそのままブートさせて使えるはずです。

ただ、 Android 端末は ARM ベースの CPU は使っているものの、グラフィックス（ 液晶画面出力）、タッチパネル、サウンド、 Wi-Fi 等は各端末のハード独自のものでかつ、仕様/設計は未公開であったり、 Linux カーネル本体の機能だけでは対応できなかったりする為、これらの周辺デバイスを利用できるようにする為のハードルは高い状況です。特に、 Wi-Fi はおろか、通常のデスクトップ PC を使っているときにはあまりに基本的な機能で気にもとめなかったような機能（ キー入力、マウス入力、画面に文字を描画）すらも最初から未対応がほとんどですので、本方針を取るにはそれ相応のスキルが必要です。（ VGA BIOS とか、 IBM PC AT の BIOS の規格が大変ありがたく見えてきます）

但し、本方針でもし Debian をそのまま利用できれば、より自由なソフトウェア開発環境を搭載した携帯端末を手にできる事になる為、非常に魅力的ではあります（ よね? ）

## 6.4 数々の障害と方針変更

先に述べた方針 1 が手軽なはずなので、こちらの方針をとって作業を進めていました。が、実は以下に列挙する問題にあたってしまい、解決出来なかった為、不完全ながらも方針 2 を取らざるを得ない状況となってしまいました。

1. B&N の Nook Color は、 Android 2.1〜2.2 OS を改造して電子書籍端末にした製品のため、通常の Android OS とは異なります。そのため、 Android 端末でそのまま動作させることができるようなアプリをどうやってもインストールできませんでした。つまり頼みの VNC Viewer を動作させる事が出来ませんでした。また、こちらの原因を調べようにも、ソース公開義務の発生しない肝心のライブラリ、アプリケーションフレームワーク、アプリケーション[*12]が全くの非公開であるため原因追求が困難です。
2. 管理権限を奪取して adb が使えるようにしたのですが、 USB 経由で利用する Nook Color 側の adbd の動作が非常に不安定で、一度でも USB を外す/PC の電源を落とす/何もせず時間を空けると次からどうやっても端末側 adbd に接続できなくなる現象に悩まされました（ 管理権限奪取の方法を最初からやり直すと復活できる事がたまにありましたが、なぜか失敗する事が多いです。）さらに悪い事に、端末側ファームを 1.2.2 にアップデートすると、もはやどうやっても adbd に接続できなくなってしまいました[*13]。

[*12] Android OS の用語となります。基本的な Android OS の構造: `http://developer.android.com/images/system-architecture.jpg`

[*13] 後の調査でわかったことですが、 Wi-Fi 経由で端末側 adbd に接続すると安定するかも? との情報もあります。が、 Wi-Fi 環境を自分は持っていない為、こちらも未評価です

そこで、方針 1 をあきらめ、方針 2 を取る事にしました。

## 6.5 再考: Nook Color

Debian を Android 端末の機能を最大限活用して動作させるのにいろいろてこずる Nook Color ですが、このやり方を諦めれば、唯一 Debian を動作させるのに救いのある機能があります。 Nook Color は miniSD を挿入して利用できるのですが、ここに OMAP 用ブート形式[*14]のブートイメージを書きこんでおくと、こちらからそのままブートしてしまう機能( 脆弱性? ) があります。

そこで、ここにブートイメージを書き込んで、ブートすれば Debian を利用できるのでは? という事を試しました。

## 6.6 Debian の Nook Color 用ブートイメージを作る

### 6.6.1 方針

ここでは、以下の方針にそってブートイメージを作成します。

1. qemu を使って Debian の ARM 用ディスクイメージを構築します。
2. Nook Color の root 奪取に使う nooter0.2.zip のカーネル/initrd イメージは Nook Color の USB を RNDIS 用のネットワーク I/F にセットアップする能力を持ちます。これを利用すると、コミュニケーションポートとして USB が非常に便利になるのでこれらを拝借します。
3. 以上 1,2 を合体した OMAP 用ブート形式の miniSD カードを作成します。

具体的なやり方を次に述べます。

### 6.6.2 やり方

用意するもの: Debian sid の動く PC, グローバル回線, miniSD カード 1 枚、必要なら miniSD カードライター( USB メモリに変換するタイプのコネクタでも可)

1. 諸々 PC 側に取り揃えます。

```
$ sudo aptitude install debian-installer-6.0-netboot-armel qemu-system util-linux kpartx dump unzip gvncviewer
$ wget http://cgit.openembedded.org/openembedded/plain/contrib/angstrom/omap3-mkcard.sh; chmod 755 omap3-mkcard.sh
```

2. イメージディスクの準備をします。

```
$ qemu-img create -f raw arm-versatile.img 5G
```

3. インストール作業を行う為、以下のコマンドを実行します[*15]。

```
$ qemu-system-arm -M versatilepb -kernel /usr/lib/debian-installer/images/armel/versatile/vmlinuz-2.6.32-5-versatile \
 -initrd /usr/lib/debian-installer/images/armel/versatile/initrd.gz -m 256 -drive file=./arm-versatile.img,if=scsi,\
 bus=0,unit=0,cache=writeback -append "root=/dev/ram desktop=lxde priority=medium"
```

4. Debian インストーラがウインドウに立ち上がり、通常のインストーラメニュー形式の Debian のインストールを進める事ができるようになります。ネットワークも自動的に認識 (IP アドレスなども) され、 qemu の持つネットワークの仕組みで PC に接続されたネットワークがそのままネットワーク経由のインストールに利用されます。表 2 を選択しつつインストールを行います。
5. インストールが完了し、最後にシャットダウン画面になります。最後 Reboot system の指示が出たのを確認してそのまま qemu を Ctrl+C で停止させます。
6. インストール完了後のディスクイメージに内臓されているカーネルイメージ、 initrd イメージを取り出します。

[*14] `http://processors.wiki.ti.com/index.php/SD/MMC_format_for_OMAP3_boot`。但しこのページのコマンド通りにそのまま fdisk 実行しても、 Debian ではブートイメージは作れないので注意。

[*15] LXDE デスクトップにする理由は、 GNOME だとメモリの必要量が多すぎて swap が発生してしまい、快適に動作しない為です

| 項番 | 設定項目 | 指定内容 | 備考 |
|---|---|---|---|
| 1 | Choose language | Japanese 日本語 | |
| 2 | ネットワークの設定 | DHCP でネットワークを自動で設定 | |
| 3 | インストールする Debian バージョン | sid | |
| 4 | ロードするインストーラコンポーネント | 何も選ばない | 何も選ばないが、「 続ける」だけは選択 |
| 5 | ディスクのパーティショニング | ディスク全体を使う, すべてのファイルを 1 つのパーティションに | |
| 6 | ソフトウェアの選択 | デスクトップ環境、ラップトップ、標準システムの 3 つ | お好みで |
| 7 | インストールする Linux カーネルのバージョン | 3.2.0-2 | |

表 2　インストーラで選択する項目

```
$ mkdir debian
$ sudo kpartx -a ./arm-versatile.img
$ sudo mount -t ext3 /dev/mappoer/loop0p1 ./debian
$ cp debian/boot/initrd.img-3.2.0-2-versatile .
$ cp debian/boot/vmlinuz-3.2.0-2-versatile .
$ sudo umount ./debian
$ sudo kpartx -d ./arm-versatile.img
loop deleted : /dev/loop0
$
```

7. 以下のコマンドで取り出したカーネルイメージ、 initrd イメージで再度 qemu を立ち上げます。 xdm の画面が出るので、そのままログインします。すると LXDE のデスクトップ画面が現れます。

```
$ qemu-system-arm -M versatilepb -kernel ./vmlinuz-3.2.0-2-versatile \
 -initrd ./initrd.img-3.2.0-2-versatile -m 256 -drive file=./arm-versatile.img,if=scsi,\
 bus=0,unit=0,cache=writeback -append "root=/dev/sda1 "
```

8. LXDE の画面から、 lxterminal を開き、パッケージをアップデートします。さらに、 vnc サーバーの導入を行っておきます。

```
$ su root
Passwd: インストール時の root パスワード
# aptitude update;aptitude safe-upgrade;aptitude install tightvncserver
```

9. vnc サーバーのインストールが完了したら、 LXDE 端末画面でシャットダウンします。

```
$ sudo shutdown -h now
```

10. Power off のメッセージが出力されたら、 qemu を Ctrl+C で停止します。
11. miniSD を PC に差し込みます。 mount されてしまっているようであれば、**必ず umount** しておきます。( 注: ここでは/dev/sdb1 としてマウントされてしまった事を過程します)

```
$ sudo umount /dev/sdb1
```

Tips:
なお、 PC の miniSD のリーダ/ライターを搭載している場合、 miniSD アクセス用のチップが Richo R5C822 を搭載している場合で( よくある)、 miniSD を挿入すると”mmc0: error -110 whilst initialising SD card” や、”mmc0: Reset 0x1 never completed” などが大量に dmesg に記録されて全くアクセスできない場合があります。この場合は以下の定義を/etc/modules に記載してリブートすると、安定して miniSD へアクセスできるよう

になります。

```
$ cat /etc/modules
... 中略...
# for R5C822
mmc_block
tifm_sd
$
```

12. miniSD に OMAP 用ブート形式のパーティションを作成します。
    注: **必ず miniSD の認識されているデバイスファイルを omap-mkcard.sh に指定してください!** ここでは/dev/sdb と仮定しています。間違ったデバイスファイルを指定すると該当ディスクの中身が消えてしまいます!

```
$ sudo ./omap3-mkcard.sh /dev/sdb
```

13. nooter0.2 配布先 (`http://www.mediafire.com/?cfddu9wt9d8dunl`) を epiphany などの WEB ブラウザでアクセスして、 nooter0.2.zip を入手します。
14. 解凍して、ディレクトリ nooter/にマウントします。

```
$ unzip nooter0.2.zip
$ mkdir nooter
$ sudo kpartx -a ./nooter_sdcard_40mb.img
$ sudo mount /dev/mapper/loop0p1 ./nooter
```

15. miniSD カードの 1 つ目のパーティション (/dev/sdb1) をマウントします。

```
$ mkdir miniSD
$ sudo mount -t vfat -o rw,uid=your-uid,gid=your-gid /dev/sdb1 ./miniSD
```

16. nooter の u-boot イメージ、カーネルイメージ、 initrd イメージをコピーして、 miniSD,nooter をアンマウントします。

```
$ cp nooter/* ./miniSD/
$ sudo umount ./nooter
$ sudo kpartx -d ./nooter_sdcard_40mb.img
loop deleted : /dev/loop0
$ sudo umount ./miniSD
```

17. miniSD の 2 つ目のパーティション (/dev/sdb2) をマウントして、 Debian イメージをまるごとコピーします。終わったらアンマウントします。

```
$ su root
Passwd:
# kpartx -a ./arm-versatile.img
# mount /dev/mapper/loop0p1 ./debian
# mount /dev/sdb2 ./miniSD
# cd debian
# dump -0 -f- ./ | (cd ../miniSD && restore -rvf- )
# cd ..
# umount ./debian
# umount ./miniSD
```

以上となります。

## 6.7 動作させる

早速動作させてみましょう。

1. Nook Color の電源ボタンを長押しして電源を OFF にします。
2. Nook Color の miniSD スロットに作った miniSD を挿入します。
3. Nook Color に製品付属の usb ケーブルを差し込み、 PC の USB に差し込みます。
4. usb ケーブルが橙→緑に変わる。同時に PC 側の dmesg を観察すると、 cdc_ether モジュールがロードされ、 usb ポートが NIC に変化するのが観察できます。
5. 以下のコマンドを打ち込んで、 Nook Color へ root でログインします。なお、 PC 側を 192.168.2.10 と仮決めし

ます。( Nook Color 側は nooter 由来の initrd ですと 192.168.2.2 に決め打ちでセットアップされます。)

```
$ sudo /sbin/ifconfig usb0 inet 192.168.2.10 netmask 255.255.255.0 up
$ ssh root@192.168.2.2
#
```

6. miniSD の Debian を動かし、 vncserver を立ち上げます。

```
# mount -t ext3 /dev/mmcblk0p2 /mnt
# mount -t devpts devpts /mnt/dev/pts
# mount -t proc proc /mnt/proc
# mount -t sysfs sysfs /mnt/sys
# chroot /mnt /bin/bash
root@buildroot:/# cd
root@buildroot:~# vncserver -geometry 800x600
Passwd: <適当にパスワード入れる>
Retry : <上と同じパスワード入れる>
... 多少エラーメッセージが出て vncserver が立ち上がる気にしない...
root@buildroot:~#
```

7. PC 側にて gvncviewer を起動すると、 Nook Color 上で Debian にログインした状態になる。

```
$ vncviewer 192.168.2.2:1
```

8. lxterminal を vncviewer 上で開き、適当に/etc/resolv.conf し、 route add default gw 192.168.2.10 等して、PC 側を NAT 設定にすると、 Nook Color は USB 越しで PC 側の持つグローバル側ネットワークを利用できるようになります。 aptitude などもも全く問題なく出来ます。

## 6.8 電源の切り方

いろいろ試したら、電源を OFF にしたくなるかと思います。この場合の手続きは以下の通りです。

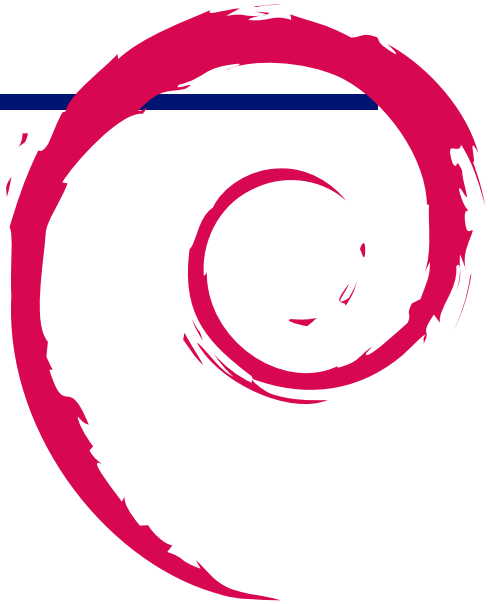
1. Nook Color に ssh ログインします。
2. sync;sync;halt と打ち込みます。
3. Nook Color との ssh が切れます。
4. Nook Color から miniSD を抜きます。
5. Nook Color の電源ボタンを 15 秒以上押します。
6. もう一度 Nook Color の電源ボタンを 15 秒以上押すと、 Nook Color 本体がブートします。あとはいつもの Nook Color ですので、そのまま電源ボタンを軽く押すなり放置するなりするといつも通りの電源 OFF ( Sleep?) 状態になります。

## 6.9 終わりに

今回手動で chroot することにより、 Nook Color で Debian を動かしてみました。これで電池動作で小型軽量ということもあり、 Debian を入れたノートパソコンと組み合わせると、 ARM CPU のタブレット開発環境を持ち歩く事ができます。実際、本記事を書くにあたり、平日は通勤電車の中を主に活用して試行錯誤していました。

今回 Nook Color が持っている LCD/タッチパネル/Audio/Wi-Fi について一切 Debian 側から制御までは到達できませんでした。 100 % Debian で動く安価で軽量なタブレット端末の開発の道は険しそうです。

次は OMAP3 用の CPU 向けに kernel のクロスコンパイルに挑戦してみようかと思うこの頃でした。



# 7 月刊 Debhelper

杉本 典充

## 7.1 今月のコマンド: dh_md5sums

dh_md5sums コマンドは「 DEBIAN/md5sums ファイルを生成する」コマンドです。

### 7.1.1 DEBIAN/md5sums ファイルについて

「 $ ar x debian-package.deb」を実行し現れる control.tar.xx ファイルを展開すると md5sums ファイルが出てきます。この md5sums ファイルは data.tar.xx ファイルに含むファイルそれぞれから取得した md5sum を記述しています。

```
$ apt-get download hello-debhelper
$ ar x hello-debhelper_2.7-3_i386.deb
$ ls
control.tar.gz  data.tar.gz  debian-binary  hello-debhelper_2.7-3_i386.deb
$ tar xf control.tar.gz
$ ls
control         data.tar.gz    hello-debhelper_2.7-3_i386.deb
control.tar.gz  debian-binary  md5sums
$ head -n 1 md5sums
098518cc321f0467dc0e7c67f65e2cc1  usr/bin/hello
```

### 7.1.2 パッケージのビルド処理における dh_md5sums の実行

dh_md5sums コマンドはインストールするファイルの md5sum を取得する処理のため、ビルド処理の終盤で実行されます。

```
$ apt-get source hello-debhelper
$ cd hello-debhelper-2.7
$ debuild -uc -us
  (省略)
  dh\_gencontrol -a
  dh\_md5sums -a
  dh\_builddeb -a
  (省略)
$ ls debian/hello-debhelper
DEBIAN   usr
$ head -n 1 debian/hello-debhelper/DEBIAN/md5sums
098518cc321f0467dc0e7c67f65e2cc1  usr/bin/hello
```

### 7.1.3 コマンドのオプション

表 3　dh_md5sums のコマンドラインオプション一覧

| オプション | 説明 |
|---|---|
| -x, –include-conffiles | DEBIAN/conffiles ファイルに記述した設定ファイルの md5 も生成します。 |
| -Xitem, –exclude=item | md5sum の生成を除外するファイル名を指定します。ただしディレクトリが別でもファイル名が一致すればどちらも除外されます。 |

## 7.2 今月のコマンド: dh_strip

dh_strip コマンドは「実行ファイル、共有ライブラリ、スタティックライブラリを strip する」コマンドです。

### 7.2.1 デバッグシンボルの扱い方を制御する

オプションなしや環境変数を指定せずに dh_strip コマンドを実行するとコンパイルしたオブジェクトファイルのデバッグシンボルを strip するのが通常の処理です。しかし、デバッグを目的とする場合はデバッグシンボルを strip してしまうとデバッガが十分に機能しないため困ります。

dh_strip コマンドではデバッグシンボルを以下のように扱うことができます。[1]

- オプションなしで実行すると、strip する。
- 環境変数 DEB_BUILD_OPTIONS=nostrip を指定して実行すると、strip しない。(処理的には dh_strip が即座に終了する)
- –dbg-package オプションを指定すると、/usr/lib/debug 配下にデバッグシンボルを分離して残すパッケージ(=デバッグパッケージ)を作成する。

### 7.2.2 デバッグパッケージを作成するための条件

debhelper の機能を利用すると strip 済みのバイナリパッケージに加えて簡単にデバッグパッケージを作成できます。デバッグパッケージを追加で作成したい場合は以下の処理を記述してパッケージをビルドすればよいです。

- CFLAGS などのコンパイルオプションに"-g' を付与しビルド時にデバッグシンボルを生成するようにする。
- debian/rules で override_dh_strip を定義し、dh_strip –dbg-package=package-dbg を処理させる。
- debian/control にパッケージ「package-dbg」の定義を記述する。このとき、パッケージ「package-dbg」はパッケージ「package」にバージョン指定をして depend すること。

### 7.2.3 コマンドのオプション

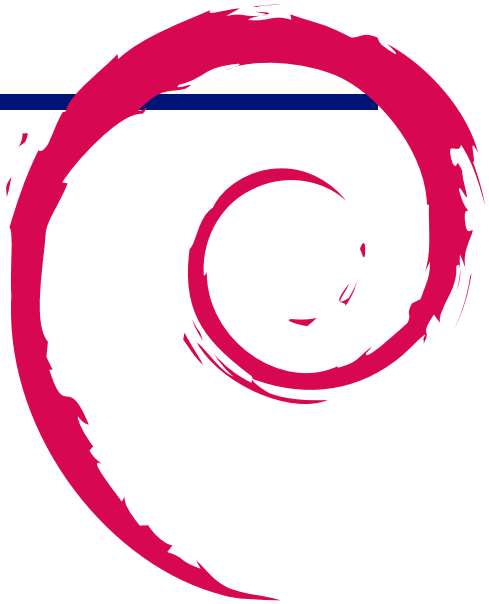
表 4 dh_strip のコマンドラインオプション一覧

| オプション | 説明 |
|---|---|
| -Xitem, –exclude=item | 指定した文字列を含むファイルを strip 処理の対象から除外する。複数のファイルを指定したい場合はオプションを複数回指定することも可能。 |
| –dbg-package=package | デバッグシンボルを含むパッケージ「package-dbg」を作成する。 |
| -k, –keep-debug | パッケージをビルドした作業ディレクトリ内の usr/lib/debug に strip 後のデバッグシンボルファイルを残す。–dbg-package オプションの指定で事足りる場合は多いが、より細かくデバッグシンボルを扱いたい場合を想定して用意されている。 |

# 参考文献


[1] Debian Wiki - DebugPackage http://wiki.debian.org/DebugPackage

[2] Debian.org 第 6 章パッケージ化のベストプラクティス http://www.debian.org/doc/manuals/developers-reference/best-pkging-practices.html



# 8 索引

**Debian 勉強会資料**

2012 年 4 月 21 日　　初版第 1 刷発行

東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）