第 4 回 東京エリア Debian 勉強会
事前資料*

Debian 勉強会会場係 上川純一†

2005 年 5 月 21 日

* 機密レベル public: 一般開示可能

† Debian Project Official Developer

目次

1 Introduction To Debian 勉強会 3
1.1 講師紹介 . . . 3
1.2 事前課題紹介 . . . 3

2 Debian Weekly News trivia quiz 7
2.1 2005 年 15 号 . . . 7
2.2 2005 年 16 号 . . . 8
2.3 2005 年 17 号 . . . 9
2.4 2005 年 18 号 . . . 9
2.5 2005 年 19 号 . . . 10
2.6 2005 年 20 号 . . . 10

3 最近の Debian 関連のミーティング報告 12
3.1 東京エリア Debian 勉強会 4 回目報告 . . . 12

4 DFSG 教 13
4.1 DFSGって何だ? . . . 13
4.2 DFSGってどんなんだ? . . . 13
4.3 フリーなライセンスの例 . . . 13
4.4 FAQ . . . 14
4.5 おわりに . . . 14

5 dpkg-cross 15
5.1 はじめに . . . 15
5.2 やってみる . . . 15
5.3 おわりに . . . 19

6 lintian と linda を使っての Debian パッケージの作成精度向上 20
6.1 Debian パッケージに関する問題のパターンマッチング . . . 20
6.2 実行してみる . . . 20
6.3 どうなっているのか . . . 21
6.4 どう応用するか . . . 22

7 個人提案課題 23

8 グループ提案課題 24

9 Keysigning Party 25

10 次回 26

1 Introduction To Debian 勉強会

上川純一

今月の Debian 勉強会へようこそ．これから Debian のあやしい世界に入るという方も，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？

目的として下記の二つを考えています．

- メールではよみとれない，もしくはよみとってられないような情報を情報共有する場をつくる
- まとまっていない Debian を利用する際の情報をまとめて，ある程度の塊として出してみる

また，東京には Linux の勉強会はたくさんありますので，Debian に限定した勉強会にします．Linux の基本的な利用方法などが知りたい方は，他でがんばってください．Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作りながらスーパーハッカーになれるような姿を妄想しています．

Debian をこれからどうするという能動的な展開への土台としての空間を提供し，情報の共有をしたい，というのが目的です．次回は違うこと言ってるかもしれませんが，御容赦を．

1.1 講師紹介

- 松山さん Debian ユーザです．普段は java いじってはります．
- 岩松さん 普段はすーぱーえっちはっかーだそうです．
- 上川純一 宴会の幹事です．Debian Developer です．元超並列計算機やっていて，今は音楽関係とか，気づいたら canna とか．あと，pbuilder や，libpkg-guide などを通して，Debian の品質向上を目指してます．

1.2 事前課題紹介

今回の事前課題は「さわってみた/さわってみたい Debian のこんなアーキテクチャ移植版」というタイトルで 200-800 文字程度の文章を書いてください．というものでした．その課題に対して下記の内容を提出いただきました．

1.2.1 田中さん

Debian の「移植版」という単語に引っかかってしまいました．kernel の i386 以外への移植版はあるけど，Debian GNU/Linux system は i386 で開発されたものを移植してるわけではないのでは? と思い込んでました．http://www.debian.org/ports/ を読んでみると，(おおらかに言えば) 全ての architecture でのものが移植版という感じもします．

話は飛びますが，僕が Debian を使っている 90% の理由が仕事上のものです．職場の，およそ 10 台ほどの

計算機で linux を稼働させています. i386 系のものは最初から Debian で運用してましたが, やむを得ない (?) 理由で SuSE が走ってる alpha 機や, 昔々の iBook (Mac OS 8 で動いてました) とかもありました.

最近, これら非 Debian な system を”粛清”し, 全て Debian にしてしまいました. ということで, alpha と powerpc をさわりました. 現在も運用中です. 他に hppa な WS が数台あって, 彼らの余生は Debian で過ごさせようかとも思いましたが, なんせ発熱と騒音がひどいので, 何もせずに引退してもらいました.

非 i386 系でも, 一旦 boot してしまえば i386 系との違いはあまり感じません (感じないような使いかたをしてるとも言えますが). ただ, 恐らくは user が少ないせいで, 特定の architecture でだけ bug が取れてない package がいくつかあります. tripwire は ppc では (alpha でも) 動いてませんでしたが, こないだ fix していただけました (BTS #240982). ちょっと困ってるのは, acct が alpha ではまともに動いてないことです (BTS #291154).

特に他の architecture でさわりたいのはありませんが, これから購入する計算機は powerpc 系 (powerbook や mac mini) にして, そこで Debian を使う可能性は考えてます. Debian GNU/Mac OS X があればいいなとは思ってます (Fink の packaging system は Debian のとは随分違うもののような印象があるので).

あと, Debian-Plan9 が本当にあれば遊びとしていじってみたい気がします. plan9 自体はいじったことはないんですけど.

1.2.2 澤田さん

さわってみたいアーキテクチャというどう字数を稼げばいいのか難しいお題を突きつけられたのでいろいろ考えてみた。まず、JavaVM や RubyVM などの VM 上で動く Debian。これはまず Linux を動かさないといけないので Debian の話じゃないなあということになり廃案。というわけで Linux がすでにあるものの上で Debian が動くことを考える。そういえば最近 IPod Linuxってどうなってんだっけ？とか CE Linux 上で Debian が動くと萌えだよなとか。ここで問題となるのは何ができたら Debian なのかということである。やはり apt-get できないと駄目だろうか。そうすると IPod は辛い気がする。無線が搭載されることを期待しよう (そういう問題か？) 一応最後に本気でさわってみたいアーキテクチャを書いておくと arm(SL Zaurus) かな。Pocket Workstation とかいうのもあるようだし。

1.2.3 きんねこさん

さわってみた Debian の SH アーキテクチャ移植版

近年、どうにも盛り上がりに欠ける SH アーキテクチャなのだけど、kernel や gcc の対応は地道に続けらている。　一時は Debian の正式対応アーキテクチャになれるようにという動きもあったようだが、現在はそういった動きもなく低調。数年前に sid をベースに dodes プロジェクトがリリースした環境が最もまとまったディストリビューションであったが、それすら簡単にインストールを行うことも難しい状態であった。それでも国産 CPU である SH シリーズは国内では人気があり、いくつかの組み込み系の Linux 実装が販売されていて、細々とメンテナンスされている。中には、株式会社シリコンリナックスのように自社のボード向けに Debian をベースにしたディストリビューションを保守しているところもある。

SH の使われる組み込み環境では、商用の組み込みディストリビューション実装があっても、それらは NDA や利用契約で縛られていて、なかなか一般の人は入手がしがたいし、自由に再利用できる環境としてソフトウエアが提供されることは稀である。さらにハードウエアに至っては趣味や学習向けとしても５〜１０万円の投資が必要であり、使いたいと思ってもなかなか手の出るものではない。以前のようにドリームキャストや PDA 類のような SH をベースにしたリテール品があり、その移植がある程度進んでいれば、まだ利用者、

開発者ともに増えてゆく可能性があるが、このまま状況が変化しなれば、SH での Linux 実装は一般の人の目に見える場所から消えてゆき、インハウスにだけ存在することになりかねない。　国産で、海外ではあまり使われていない超マイナーな CPU でありバリエーションも少なく単価も高いとなれば、国内での対応を進めなければこのまま火は消えてしまうかもしれない。そのような状況はあまり面白くないので、昨年から iohack project を土台に再度の盛り上がりを仕掛けようといろいろな取り組みを行っているところだ。

幸い、その一部が実を結んで、I-O DATA の HDL シリーズのハードウエア上で動作する Sarge 相当のディストリビューションの利用ができるようになってきた。これは細渕氏によるハードウエア・ソフトウエアの解析情報の提供や lilo の改造、iWA 氏による base と pool の構築と提供作業のおかげだ。最近では、iohack 以外のところでも kogiidena 氏による kernel2.6 のテスト実装も出ており、それをベースにした pbuilder でのパッケージ自動作成が海老原氏によって進められている。すばらしいことだ。

昨年、UNIX USER 誌に HDL シリーズの Debian 化カスタム方法を記載した記事を書かせていただいたが、その反応として、興味はあって雑誌も HDL も購入したのだが、ハードウエアの改造という点で、普通の方には大きな抵抗があることがわかってきた。近年の PC の高発熱と高騒音化によって、PC アーキテクチャが家庭でのサーバー運用に適さなくなってきており、ローパワーで静かなサーバーが望まれており、ニッチではあるが市場が存在するのも見えてきた。そこで、業務の一環として、Debian が動作する SH 環境のハードウエアと、IA-32 環境で動作し SH の起動 HDD を作成する LiveCD のインストーラーを企画作成してみた。一部のドライバを除いて、オープンソースの部品ばかりで構成されているので、カスタムも自在にできる。LiveCD の作成には bootcd パッケージを利用してみた。比較的簡単に LiveCD が作成できるものの、FD が必須であるなどのいくつか問題を持っているように思われた。　製品としてはノンサポートにはなるが、発売をきっかけに SH-Linux に触れる人材が増える事を期待したい。組み込みに Linux を利用しているメーカーに勤務するものとして、会社レベルでのコミュニティへのフィードバックはなかなか難しいものがあるが、コードや資金の提供ではなく、このような形での貢献というのもありうるのではないかという１つの可能性を示す事ができたのではないかと思っている。

1.2.4　**中島さん**

MSX ワールドに行って 1 チップ MSX を予約してしまった。たいしたことないと思っていたけれどなかなかちゃんと作ってあるもんだった。ということで MSX に移植してほしい。予約だけしたけれどゲームやらないから、たぶん使わない。使わないのに限定販売だったので即予約してしまった。　　このまま使わないと、もったいないので、とにかく使いたい。使ってる証を誰かに見せたい。ユーザである証拠がほしい。この MSX に移植してくれれば人に自慢できる。１チップだからどこにでも持っていける。このように目的を決めなければやる気を起こさないので、とにかく使うために MSX の移植版がいい。

1.2.5　**岩松さん**

いままで触ったことのある Debian パッケージのあるアーキテクチャー

- i386
- powerpc（ powerbook / openblockS ）
- arm（ psion ）
- sh3 /sh4（ オフィシャルではありませんが ）

です。一番おもしろかったのは psion にいれたときです。こんなものでも動くのか！とびっくりしましたが 1 週間ほどで飽きました。

触ってみたいアーキテクチャー

- s390 を触ってみたい（といいますか、実機を見た事がない）のでどんなものなのか非常に興味があります。

1.2.6 上川

なんとなく，これからは 64bit の時代だ，ということで，とりあえず手が届きそうな 64bit のアーキテクチャを全部さわっておきたいところです．ppc64, ia64, amd64 を全部そろえたいのですが，どうでしょう？

物欲としては，Zaurus とか欲しいですが，電車でハックしようにも通勤時間が短いのが難点．通勤時間のかかる遠くに引っ越すしか．

2　Debian Weekly News trivia quiz

上川純一

ところで，Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが，一人で読んでいても，解説が少ないので，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 DWN を読んでみましょう．

漫然と読むだけではおもしろくないので，DWN の記事から出題した以下の質問にこたえてみてください．後で内容は解説します．

2.1　2005 年 15 号

4 月 12 日時点の情報です．

問題 1. Debian Project Leader 選挙の結果 DPL になったのは

□ A Anthony Towns

□ B　Branden Robinson

□ C　Michael Jackson

問題 2. Evan Prodomou が Creative Commons のライセンスに関して任じられたのは

□ A Debian 側の Creative Commons の Committee として動くこと

□ B 文書の翻訳

□ C 文書の精査

問題 3. tigon II チップのファームウェアが問題なのであれば，仕様を見て，フリーで実装しなおせばよいのではないか，といったのは

□ A Branden Robinson

□ B　Peter De Schrijver

□ C　通りがかりの人

問題 4.　パッケージの自動テストについて Petter Reinholdtsen がプロトタイプを作成したというのは

□ A アップグレードの自動テストのスクリプト

□ B リリースクリティカルバグの原因となった人に自動で罰ゲームを選択してくれるシステム

□ C パッケージの使いやすさに関して定量的に計測して，使いにくいパッケージは自動で警告を出してリジェクトできるようなシステム．

問題 5. selinux について Manoj Srivastava が開始したのは

□ A etch での selinux の導入に向けてのプロジェクト
□ B selinux はなかったことにするための隠蔽プロジェクト
□ C selinux に対抗する実装を開始

2.2 2005 年 16 号

問題 6. ミュンヘン市がデスクトップ PC の OS として選択した OS は
□ A Windows
□ B Debian
□ C Solaris

問題 7. Debian 3.0 のアップデートが出たが，それはどのバージョンか
□ A 3.0r5
□ B 3.0rX
□ C 3.0TNG

問題 8. Adrian Bunk が GPL がフリーであるか，ということについて議論した内容は何か
□ A ライセンス文章については変更できないので，フリーではないのではないか
□ B GPL のイデオロギーが気に入らない
□ C RMS が挙動不審だ

問題 9. カーネルチームの IRC ミーティングで決定した内容はなにか
□ A etch 以降は Hurd カーネルを採用する
□ B etch は FreeBSD カーネルと Linux カーネルのソースツリーをマージする方向で検討する
□ C testing に現在はいっているカーネルで基本はフリーズする.

問題 10. ライセンスの文書で GPL に追加で制限を加える場合にどうなるか，という議論でどういう問題点が指摘されたか
□ A Debian は作者の意向を尊重するため，GPL ライセンスで配布されているプログラムに追加で制限を加える場合には，Debian として配布できない可能性がある
□ B ライセンスはいくらでも変更してよいので問題無い.
□ C 作者は神様です

2.3　2005 年 17 号

問題 11. GNOME2.10 はどうなったか
□ A unstable にアップロードされた
□ B experimental にアップロードされた.
□ C sarge に含まれる予定

問題 12. 遅々として Debian に入って来ない mplayer だが，MJRay は mplayer について FAQ を作成した. それによると
□ A ftpmaster を人質にとったので，もうすぐアーカイブにインストールされるだろう.
□ B ftpmaster に賄賂をおくったので，もうすぐアーカイブにインストールされるだろう.
□ C 問題となっているコードは削除しており，アップロードは ftpmaster の承認待ち.

問題 13. snapshot.debian.net に関して提案されなかったことは
□ A debian.org に昇格しよう
□ B バックアップをとりましょう
□ C サーバに DDoS をかけましょう

2.4　2005 年 18 号

問題 14. Leadership meeting にてなされた金銭的な議論は
□ A aKademy への出席に関する参加補助
□ B Leader への給与について
□ C Debian Developer への報酬について

問題 15. PHP アプリケーションにおいてよくある問題で，Martin Schulze がセキュリティーの観点からもっとも問題なので注意してほしいと指摘したのは
□ A 設定ファイルが http で取得できる場所に存在する設計になっている場合
□ B PHP という言語処理系をつかっていること自体
□ C PHP の文法でプログラムを作成しようとする発想

問題 16. Andreas Barth によるとリリースの状況は
□ A ARM の buildd も追加されたので testing-security の準備がほぼ完了した.
□ B 地下組織の暗躍により進展が阻害されている
□ C リリースはそろそろあきらめようかと考えている.

問題 17. Debian Conference で今年は新しいイベントとして何をする予定か

□ A 全員参加のジャンケンゲーム
□ B 一般参加者をつのり，その参加者のためのイベントを最初の日に開催する
□ C Branden Robinson は誰だゲーム

問題 18. Jorgen Schäfer の提案したのは
□ A scheme パッケージのポリシを作成し update-alternatives で管理する
□ B scheme は廃棄して common lisp に移行する
□ C ruby で書かれているスクリプトを全て scheme で実装しなおす

2.5 2005 年 19 号

問題 19. sarge のバージョン番号はどれか
□ A 3.2
□ B 3.1
□ C 4

問題 20. sarge のフリーズが開始して，新しいパッケージのバージョンはリリースマネージャの承認がないと追加できないようになりました．さて，フリーズの開始したのは？
□ A 2005 年 5 月 3 日
□ B 2005 年 5 月 30 日
□ C 2005 年 4 月 1 日

問題 21. amd64 ポートは alioth から移行した．その移行先のサーバは？
□ A ftp.debian.org
□ B amd64.debian.net
□ C amd64.org

問題 22. apt の新しいバージョンを experimental にアップロードする際の問題は何か
□ A libapt-pkg に依存するパッケージを NMU で experimental 向けにアップロードしても新しいバージョンが unstable にアップロードされるたびに experimental にアップロードしたバージョンが削除されてしまう．
□ B apt の最新版は不安定すぎる
□ C ubuntu との利権の衝突

2.6 2005 年 20 号

問題 23. adduser のオプションの話しに端を発して disabled login と disabled password に関して，ssh のパスワードの認証の話しで何が問題となっていたか

□ A UsePam を使った場合のパスワードの扱い
□ B パスワードをつかってログインができない
□ C パスワードを入力しなくてもログインができる.

問題 24. GPL と FDL のドキュメントを一つとして混ぜることはできるか
□ A 両方とも RMS が作成したライセンスなので問題ない
□ B そういうことは気にしなくてよい
□ C 互換性がないので無理だろう，と Anthony DeRobertis は説明した

問題 25. alioth のサーバはどうなるか
□ A amd64 が別のサーバに移動してディスクスペースの問題が解消したので，いままで複数のサーバで構成していたのを一つのサーバに移行する.
□ B 使い勝手が悪いので停止
□ C アカウントが増えすぎなので募集を停止する

問題 26. Lars Wirzenius はパッケージのテストが必要だ，と説明した．その主張した内容で，現状足りないものは何だといっているか
□ A インストールと削除が無事に動作することを確認するには，新しいツールが必要だ.
□ B 暇な人材が必要
□ C 開発者がもっと必要

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 4 回目報告

前回開催した第 4 回目の勉強会の報告をします.

DWN クイズをしました．一月分の DWN についての質問が 30 問ほど.

上川が debhelper のファイル処理の仕組みについて語りました．どういう順番でどういうファイルがコピーされていくのか，置換されていくのか，などの点を説明しました.

やまねさんが po-debconf の翻訳をどのように実施しているのか，利用するツールはなにがあるのかということを説明しました．現状としては，日本語の debconf の翻訳について武藤さんの Japanese debconf-po translation status 頁を参照していじるのがよい，とのことでした．po ファイルの翻訳には特にツールは利用していないということでした．template.pot を ja.po にコピーして，そのまま翻訳し，reportbug で添付して l10n タグをつけてバグ報告する，というフローだそうです.

グループディスカッションでは，日本人のためになにができるか，という話しをしました．Debian としてのイベントへの露出や，ドキュメントの整備，出版，雑誌への露出ができるとよいのでは，という提案や，日本で普及している掲示版を利用するための Debian のインフラが足りないのでは，という提案などがありました．コンソール画面の漢字の表示の扱いも，今後なんとかしたいね，ということでした.

その後ははなの舞にて宴会

その後は朝までデニーズ．task-2ch は必要なインフラなのでは，という議論をしてから解散.

4 DFSG 教

松山

4.1 DFSGって何だ?

Debian Free Software Guidline の略. Debian Social Contract の一部を成し, またその第一条「Debian は 100% フリーソフトウェアであり続けます」のよりどころとなっています. したがって,DFSG にそったライセンスのもとにないソフトウェアは,(正式の)Debian には含まれません (non-free に入る). また,DFSG は OSI の The Open Source Definition の元となっています.

4.2 DFSGってどんなんだ?

- 自由な再配布
- ソースコードの配布
- 派生ソフトウェアの作成と, オリジナルと同条件下での配布の許可
- 原作者によるソースコードの整合性維持
- すべての個人, 団体の平等
- 目標分野の平等
- ライセンスの配布
- ライセンスは Debian に限定されない
- ライセンスは他のソフトウエアを侵害しない

4.3 フリーなライセンスの例

DFSG-free とみなさえれているライセンスには以下などがあるようです.

- GPL
- BSD
- Artistic

ただ, これらは一般的な話であって, 厳密には個々のパッケージ毎に DFSG-free か否かが判断されるそうです.

4.4 FAQ

DFSG の FAQ からいくつかピックアップしてみました[*1].

[*1] http://people.debian.org/ bap/dfsg-faq.html

どうやって DFSG-free かどうかを判定するのか　人が判定し,debian-legal という ML で議論されるそうです. ただ,DFSG は文字どおりガイドラインであって法ではないので,「DFSG-free だから自由ソフトウェアだ」と断言はできません.

DFSG フリーだと法的リスクがないのか　否. 各自弁護士を雇ってください.

ドキュメントには?　ドキュメントも DFSG-free である必要があります. DFSG は Debian のあらゆる部分に適用されます. また, プログラムのドキュメントに関してはプログラムとの間の往来を考えると, プログラムと同じ方が好ましいが...?

4.5 おわりに

かつて RMS は自由なソフトウェアとは以下のような自由のあるソフトウェアだと説きました.

- 目的を問わず実行する自由
- ソースを調べ修正する自由
- 仲間のために配布する自由
- コミュニティ全体のために修正したものを配布する自由

今の Debian と FSF の関係がどうであれ, 基本的な「思い」というのは同じだと思います. DFSG はそういう「思い」のもとで, より具体的にどうしたらよいのかを示すガイドラインとして有意義だと思います.

5 dpkg-cross

岩松

5.1 はじめに

今回、dpkg-cross について調べてみました。

5.1.1 クロス開発ってなんですか

dpkg-cross についてお話する前にクロス開発というものを理解する必要があります。i386 などの高速な CPU を扱っていると、ソフトウェアをコンパイルするマシンと実行するマシンは一緒だったりしますが、非力な CPU（ ARM とか SupserH など）でコンパイルしようとすると非常に時間がかかります。また、プログラムを動かしたいが、対象のマシンには開発環境がないというときもあります。このような時にコンパイルは高速な CPU で行うと非常に速くコンパイルできたり、別のマシンでコンパイルしたソフトウェアを対象のマシンに転送し、動かすことができます。

このように、クロス開発とはプログラムを開発する環境と実行する環境が異なる開発方法をクロス開発と言います。

5.1.2 dpkg-cross で何をするの？

dpkg-cross はクロス環境用パッケージを生成するときに使用します。クロスで開発するときには対象アーキテクチャのバイナリパッケージが必要になります。

例えば、i386 上で PowerPC で動く libncurses を使ったアプリケーションをクロスコンパイルしたいときには libncurses5_*.*.*_powerpc.deb と libncurses5-dev_*.*.*_powerpc.deb が必要になります。これらのパッケージは PowerPC 用なので 通常 i386 上ではインストールできません。 PowerPC 用のパッケージを持ってたとしても、クロスコンパイル用にディレクトリを作成してそこにヘッダファイルをコピーして..... と面倒くさいです。そこで dpkg-cross の登場になります。

5.2 やってみる

5.2.1 インストール

apt-get install dpkg-cross

以上。

5.2.2 設定

使うアーキテクチャによって設定を行う必要があります。設定ファイルは /etc/dpkg-cross/cross-compile です。以下が内容です。

```
#
# /etc/dpkg-cross/cross-compile: configuration for dpkg-cross & Co.
#

# default architecture for dpkg-cross (to avoid always typing the -a option
# if you do cross installations only for one architecture)
#default_arch = m68k

#
# general section: paths of cross compiling environment
#
# you can set the following variables here:
#  crossprefix: prefix for cross compiling binaries; default: $(ARCH)-linux-
#  crossbase  : base prefix for the following; default: /usr
#  crossdir   : base directory for architecture; default:
#               $(CROSSBASE)/$(ARCH)-linux
#  crossbin   : dir for binaries; default: $(CROSSDIR)/bin
#  crosslib   : dir for libraries; default: $(CROSSDIR)/lib
#  crossinc   : dir for headers; default: $(CROSSDIR)/include
#  crossinfo  : dir dpkg-cross' package info files; default:
#               \$(CROSSLIB)/dpkg-cross-info
#  maintainer : maintainer name to pass to original dpkg-buildpackage
#               in -m option. If not set at all, don't pass a -m, thus
#               dpkg-buildpackage will use the name from the changelog
#               file. If set to the special string CURRENTUSER,
#               dpkg-buildpackage will use the name from the
#               changelog, too, but signing the .changes will be done
#               as the current user (default key).
#  removedeps : comma-separated list of package names that should be removed
#               from depends/conflicts/etc fields
#  keepdeps   : comma-separated list of package names thet should be kept
#               in depends/conflicts/etc fields as is, without adding
#               -arch-cross.
#
# Usually, you need only set crossbase, or maybe also crossdir
#
crossbase = /usr

# A crossroot definition is for the complete-Debian-system-mounted-somewhere
# approach, mainly used for Hurd.
#crossroot-hurd-i386 = /gnu

#
# This setting for maintainer is usually right:
#
maintainer = CURRENTUSER
#
# This list is far from being complete ...
# Please send additions to Nikita Youshchenko <youshcs.msu.su>
```

```
#
removedeps = gcc, binutils, gpm, cpp, debianutils, xfree86-common, libpam-runtime,
xlibs-data, debconf

#:w

# per-package sections: additional environment variables to set
#
# Please send additions to Nikita Youshchenko <youshc@cs.msu.su>

package e2fsprogs:
    unset LD

# package gs-aladdin:
# # must be a native gcc
#   CCAUX = gcc
#
# -------------------------------
# This should fit EmDebian needs:
#
# mode emdebian:
# package all:
#   scope environment:
#       emdebian = true
#   scope makeflags:
#       CROSSPREFIX = $(crossprefix)
#       EXTRA_CFLAGS = ...
#       LIBC = ...
#       CONFIG = ...
```

- default_arch
  対象のアーキテクチャを指定します。PowerPC なら **default_arch = powerpc** と指定します。ひとつのアーキテクチャだけクロスコンパイルを行うときはここに書いておくと、アーキテクチャを指定せずに済みます。
- crossprefix クロスコンパイル用のツールを呼び出すために使用します。クロスコンパイル用の gcc などは **powerpc-linux-gcc** などになっているのでこの形式を指定します。デフォルトでは $(ARCH)-linux- が指定されています。
- crossbase
  クロス開発環境をインストールするベースとなるディレクトリを指定します。**crossbase = /home** と指定すると CROSSBASE に /home が指定されます。デフォルトでは /usr 設定されています。
- crossdir
  クロス開発環境インストール先を指定します。**crossdir = /usr/cross** と指定しますと /usr/cross 以下にインストールされます。デフォルトでは $(CROSSBASE)/$(ARCH)-linux 設定されています。
- crossbin

実行バイナリをインストールするディレクトリを指定します。デフォルトでは $(CROSSDIR)/bin が設定されています。

- crosslib
  ライブラリをインストールするディレクトリを指定します。デフォルトでは $(CROSSDIR)/lib が設定されています。
- crossinc
  ヘッダファイルをインストールするディレクトリを指定します。デフォルトでは $(CROSSDIR)/inc が設定されています。
- crossinfo
  info ファイルをインストールするディレクトリを指定します。デフォルトでは $(CROSSDIR)/info が設定されています。
- removedeps 設定されているパッケージを depends/conflicts/etc から削除します。クロスコンパイル用のパッケージをインストールしやすくするためのものだと思います。
- keepdeps 設定されているパッケージを depends/conflicts/etc から削除しないようにします。

これらの中で重要なのは

- defautl_arch
- crossbase

です。あとは特に気にしなくてよいと思います。

5.2.3 パッケージを作ってみる

クロス開発用のパッケージを作成するには dpkg-cross コマンドを使います。

```
$ ls
libncurses5_5.4-4_powerpc.deb
libncurses5-dev_5.4-4_powerpc.deb
$ dpkg-cross -b -apowerpc libncurses5_5.4-4_powerpc.deb
$ dpkg-cross -b -apowerpc libncurses5-dev_5.4-4_powerpc.deb
```

-a でアーキテクチャを指定します。-b は build を行うという意味です。行うと以下のような名前のパッケージが作成されます。

```
$ ls
libncurses5_5.4-4_powerpc.deb
libncurses5-dev_5.4-4_powerpc.deb
libncurses5-powerpc-cross_5.4-4_all.deb
libncurses5-dev-powerpc-cross_5.4-4_all.deb
```

5.2.4 インストールしてソフトウェアを作ってみる

最初はクロスコンパイル用のパッケージを入れないままコンパイルしてみます。

```
iwamatsu@soki:~/dev/study/src$ ls
sample.c
iwamatsu@soki:~/dev/study/src$ powerpc-linux-gcc -o sample sample.c -lncurses
sample.c:10:20: curses.h: そのようなファイルやディレクトリはありません
sample.c: In function ‘main’:
sample.c:24: error: ‘WINDOW’ undeclared (first use in this function)
sample.c:24: error: (Each undeclared identifier is reported only once
sample.c:24: error: for each function it appears in.)
sample.c:24: error: ‘w_pRootwin’ undeclared (first use in this function)
sample.c:24: error: ‘w_pWin’ undeclared (first use in this function)
```

/usr/power-pc/include に curses.h がないのでエラーになっています。クロスコンパイル用のパッケージをインストールして再度コンパイルしてみます。

```
iwamatsu@soki:~/dev/study/src$ sudo dpkg -i libncurses5-powerpc-cross_5.4-4_all.deb
iwamatsu@soki:~/dev/study/src$ sudo dpkg -i libncurses5-dev-powerpc-cross_5.4-4_all.deb
iwamatsu@soki:~/dev/study/src$ powerpc-linux-gcc -o sample sample.c -lncurses
iwamatsu@soki:~/dev/study/src$ ls
sample.c
sample
iwamatsu@soki:~/dev/study/src$ file sample
sample: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV),
for GNU/Linux 2.2.0, dynamically linked (uses shared libs), not stripped
```

PowerPC 用のプログラムとしてコンパイルができました。PowerPC なマシンに持っていって動作確認してみて、正常に動作すれば OK です。

5.3 おわりに

今回 dpkg-cross について説明してみました。debian は Redhat 系とくらべてクロス開発環境の構築方法が容易だなと感じました。Redhat だとクロス開発環境用に再コンパイルしなおすという方法しかないようです(私が調べた限りでは)。やっぱ Debian はすばらしいなぁと思いました。

また、今回の事前課題が「さわってみた/さわってみたい Debian のこんなアーキテクチャ移植版」でもありますし、x86 以外のアーキテクチャ(最近なら PowerPC?) を触ってみていろいろコンパイルしてみるのもいいかもしれません。

6 lintian と linda を使っての Debian パッケージの作成精度向上

上川

6.1 Debian パッケージに関する問題のパターンマッチング

Debian パッケージを作成する際に，このファイルにこういう内容を書いているのであれば一般的に問題だろう，とか，このファイルがこの内容になっているのであればテンプレートファイルのままになっているのではないか，ということを指摘するのは実はそんなに難しいことではありません．Debian パッケージのファイルの中を正規表現で検索してしまえば見付かります．そういう一般的な問題の解決方法として，lintian や linda は利用されています．

例えば，間違ったタイプのファイルが間違ったディレクトリに配置されている，というような問題を検出するのに優れています．

6.2 実行してみる

では，実行してみましょう．

lintian を実行してみます．deb パッケージを指定すると，バリナリパッケージを確認します．dsc ファイルを指定すると，ソースパッケージを確認します．changes ファイルを指定すると，そのアップロードに含まれる予定のファイルを全て確認してくれます．何かメッセージが出ると，-i オプションをつけると，詳細なメッセージを表示してくれます．

```
# lintian wysihtml-el_0.11.cvs.1-1_powerpc.deb
# lintian wysihtml*dsc
# lintian wysihtml*changes
E: wysihtml_0.11.cvs.1-1_powerpc.changes: bad-distribution-in-changes-file UNRELEASED
# lintian -i wysihtml*changes
E: wysihtml_0.11.cvs.1-1_powerpc.changes: bad-distribution-in-changes-file UNRELEASED
N:
N:   You've specified an unknown 'target distribution' for your upload in
N:   the debian/changelog file.
N:
N:   Note, that the distributions non-free and contrib are no longer valid.
N:   You'll have to use distribution 'unstable' and 'Section: non-free/xxx'
N:   or 'Section: contrib/xxx' instead.
N:
```

出力が E:ではじまる場合は，ポリシーに反するエラーです．W:は特にポリシーに反するというわけではないですが，直したほうが良いのではないか，と推測される「よくある問題」に対しての警告です．

同様に linda を実行してみます．

```
# linda wysihtml-el_0.11.cvs.1-1_powerpc.deb
# linda wysihtml*dsc
# linda wysihtml*changes
```

この二つのツールの出力は全く同じではないので，とりあえず両方実行しておきましょう．

6.2.1 警告メッセージの対応方法

Debian Policy に基づいてどの部分が問題なのか，を指摘してくれます．修正の方法も書いてある場合もあるので，適切に対応しましょう．

/usr/share/lintian/overrides/, /usr/share/linda/overrides/ にファイルを追加すると，メッセージを無視するように設定することができます．これらのファイルのことを lintian overrides, linda overrides とよびます．

適切でない警告[*2]を出す場合もあるので，そういう場合にはここにファイルを追加するようにパッケージを作成します．個人的には，ユーザに必要でないファイルをシステムに追加することになるため，好きではありません．できることであれば，lintian や linda に適切な修正パッチを投げましょう．

6.3 どうなっているのか

lintian と linda が内部構成としてどうなっているのか解説します．

6.3.1 lintian の構成

lintian は perl script です．Debian Package を一時ディレクトリに展開して，それに対して grep などを活用し，構成を分析します．正規表現でそれっぽい間違いを発見したら，報告します．

/usr/share/lintian/checks/以下に説明文と perl のスクリプトが大量に入っています．これらが順番に実行されます．

追加するためには，そのディレクトリにある.desc ファイルを編集して，perl のスクリプトを作成すればよいようです．

lintian は実行時にまずパッケージを lintian lab と呼ぶ場所に展開してくれるので，その展開されたファイルを解析すれば良いです．

6.3.2 linda の構成

linda は lintian を python をつかって再実装したものです．

/usr/share/linda/checks/ 以下に python のスクリプトが大量に入っています．/usr/share/linda/data/以下にチェックのメッセージの対応表があります．各メッセージは短いのですが，それぞれの文章は gettext 経由で取得するように実装されているようです．

```
msgid "versioned-provides_1"
msgstr ""
"The package shown above is trying to do versioned provides, but they aren't "
"supported by dpkg, and therefore, should not be used."

msgid "versioned-provides_s"
msgstr "Package contains a versioned Provides with package %s."
```

参考として，lintian, linda 各パッケージのアップロードの履歴の状況を調べてみました．lintian が活発でなかった 2002,2003 年の時期に linda が活発に開発されていたような雰囲気がうかがえなくもないです．

```
 zgrep '^ -- ' /usr/share/doc/lintian/changelog.Debian.gz | cut -d, -f2 | awk '{print $3}' | uniq -c
      2 2005
     10 2004
      6 2003
     14 2002
     18 2001
     13 2000
     12 1999
     44 1998


 zgrep '^ -- ' /usr/share/doc/linda/changelog.gz | cut -d, -f2 | awk '{print $3}' | uniq -c
      2 2005
      9 2004
     20 2003
     36 2002
```

[*2] false positive という

6.4 どう応用するか

devscripts に入っている debuild コマンドでパッケージをビルドすると，lintian を自動的に実行してくれます[*3]．debuild コマンドを活用しましょう．

lintian linda で確認できる問題もありますが，他にも問題は発生します．パッケージの問題は他に何があるか，確認してみます．段階がいくつかありますが，簡単に考えてみても，それぞれ別の段階で下記のような不具合が発生するでしょう．

- ソースコードがコンパイルできない．
- deb パッケージがインストールできない．
- アプリケーションが実行できない．
- 操作に対する動作がおかしい．

この問題については lintian と linda の確認はほぼ無力です．

実際は pbuilder などの他のツールを併用します．debdiff で前のバージョンとの差を確認したり，debit コマンド[*4]を利用して，自動テストツールで動作を確認したりします．

あとは，Debian のユーザがパッケージをインストールして，利用してくれて，問題を発見して BTS で報告してくれるのを待つばかりです．

[*3] /etc/devscripts.conf 参照

[*4] 現在は利用できません

7　個人提案課題

名前 _____

下記の空欄を埋めてください:

Debian の（　　）に注目し，アーキテクチャ対応は（　　）します.

企画案の図：

8 グループ提案課題

名前 _____ 名前 _____
名前 _____ 名前 _____
名前 _____ 名前 _____

下記の空欄を埋めてください:

Debian の (　　)
に注目しアーキテクチャ対応は (　　)
します.

企画案の図：

9 Keysigning Party

上川純一

事前に必要なもの

- 自分の鍵の fingerprint を書いた紙 (`gpg --fingerprint XXXX` の出力．)
- 写真つきの公的機関の発行する身分証明書，fingerprint に書いてある名前が自分のものであると証明するもの

キーサインで確認する内容

- 相手が主張している名前の人物であることを信頼できる身分証明書で証明しているか[*5].
- 相手が fingerprint を自分のものだと主張しているか
- 相手の fingerprint に書いてあるメールアドレスにメールをおくって，その暗号鍵にて復号化することができるか

手順としては

- 相手の証明書を見て，相手だと確認
- fingerprint の書いてある紙をうけとり，これが自分の fingerprint だということを説明してもらう
- (後日) gpg 署名をしたあと，鍵のメールアドレスに対して暗号化して送付，相手が復号化してキーサーバにアップロードする (`gpg --sign-key XXXXX`，`gpg --export --armor XXXX`)

[*5] いままで見た事の無い種類の身分証明書を見せられてもその身分証明書の妥当性は判断しにくいため，学生証明書やなんとか技術者の証明書の利用範囲は制限される．運転免許証明書やパスポートが妥当と上川は判断している

10 次回

次回は 6 月 11 日土曜日の夜を予定しています．内容は本日決定予定です．
参加者募集はまた後程．