

東京エリア デビアン 勉強会



Debian勉強会幹事 上川純一

2008年5月15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 をがりがりとす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。

以上を目的とした、2008 年アジェンダです：

1. 新年会「気合を入れる」
2. Open Source Conference Tokyo (3/1)
3. データだけのパッケージを作成してみる、ライセンスの考え方 (David Smith)
4. バイナリーつのパッケージを作成してみる (吉田@板橋)
バージョン管理ツールを使い Debian パッケージを管理する (git)
アップストリームの扱い (svn/git/cvs)(岩松 信洋さん)
5. バイナリーの分けたパッケージの作成。(前田さん)
バイナリーの分け方の考え方、アップグレードなどの運用とか。
6. パッケージ作成 (dpatch/debhelper で作成するパッケージ)(小林儀匡さん)
man の書き方 (roff or docbook)(でんさん)
7. パッケージ作成 (kernel patch、kernel module)、Debconf 発表練習
8. Debconf アルゼンチン、共有ライブラリパッケージ作成
9. Open Source Conference Tokyo/Fall、デーモン系のパッケージの作成、latex、 emacs-lisp、フォントパッケージ
10. パッケージの cross-compile の方法、amd64 上で i386 のパッケージとか、OSC-Fall 報告会、Debconf 報告会
11. 国際化 po-debconf / po 化 / DDTP
12. 忘年会

目次

1	Introduction	1
2	事前課題	3
2.1	前田 耕平	3
2.2	岩松 信洋	3
2.3	堀内寛己	3
2.4	やまねひでき	4
2.5	吉田@板橋	4
2.6	あけど	4
2.7	福永	4
2.8	奥野	5
2.9	沖中 研心	5
2.10	日比野 啓	5
2.11	山本 浩之	5
2.12	野村 健太郎	5
2.13	藤沢 理聡	5
2.14	荒木 靖宏	6
3	Debian Trivia Quiz	7
3.1	debian-devel-announce	7
3.2	2008 年 01 号	7
3.3	2008 年 02 号	8
4	最近の Debian 関連のミーティング報告	9
4.1	東京エリア Debian 勉強会 39 回目報告	9
5	バイナリをわけたパッケージの扱い	10
5.1	はじめに	10
5.2	バイナリの分け方の考え方	10
5.3	前回のおさらい	11
5.4	複数のバイナリパッケージを作ってみる。	12
5.5	まとめ	18
5.6	参考書籍	18

2 事前課題

上川 純一



今回の事前課題は以下です。

1. 「フリーになって嬉しい / となると嬉しいソフトウェア」
2. 「パッケージになって嬉しい / となると嬉しい / するのは難しそうなおソフトウェア」
3. 「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

この課題に対して提出いただいた内容は以下です。

2.1 前田 耕平

「フリーになって嬉しい / となると嬉しいソフトウェア」
djbdns です。qmail ははっきり言ってどうでも良いのですが、djbdns は重宝してます。設定が簡単なのと、OpenBlockS で BIND を動かすのは重すぎるので軽快な djbdns と dnscache は無いと困るので、今までは OBS 用のコンパイル環境として古い iBook でビルドしてから OBS に流し込んでました。サーバにコンパイラを入れるのはナンセンスなので。

「パッケージになって嬉しい / となると嬉しい / するのは難しそうなおソフトウェア」

Hobbit。これもこれで OBS 用にビルドしないで済むようになったので。

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

GW は子供の日が誕生日の嫁への奉仕が 70 %、家事が 25 %、今回の発表用の資料の下書き作成に 5 % でした。

2.2 岩松 信洋

「フリーになって嬉しい / となると嬉しいソフトウェア」

- Flash plugin
先日 Flash に関するライセンスの変更がありま

したけど、開発が少しは楽になるのかな、と思います。

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

- Macbook iSight のファームウェアハック
- Backlight をコントロールするための Xfce4 プラグインの開発
- Xradnr を設定するための GUI 開発 および Xfce4 プラグインの開発
- Xfce の翻訳
- U-boot for SuperH の開発
- Debian kernel module auto builder の妄想
- Debian/SH の開発

2.3 堀内寛己

僕は無職の精神病患者で、障害年金で暮らしています。それで、毎日日曜プログラミングをしています。自由なソフトウェアの作者として、ある意味究極の境遇にあるのかもしれませんが。

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

ですから、ゴールデンウィークだからといって普段と差があるわけではありません。たまたまこの時期、そし

て今も取り組んでいるのは、レスキュー OS 作成用の、live-helper より気の利いたスクリプトの開発です。

このレスキュー OS のファイルシステムは、USB フラッシュか NFS に置き、必要なとき「だけ」CD に、ブートローダー「だけ」を置きます。

どうも live-helper は、こういう用途に向いていないように思います。live-helper の構成パラメータは山ほどありますが、これは山ほど構成して 1 枚に焼きこんで、試してダメなら構成しなおし焼き直し、という使い方を想定しているからに思えます。僕が目指しているのはそういうものではありません。

live-helper の開発者は、LiveCD 作りの発想から抜け出せないようです。誤解があったら指摘してください。

2.4 やまねひでき

「フリーになって嬉しい / となると嬉しいソフトウェア」

- adobe-cmap - これでハマることが多すぎです。

「パッケージになって嬉しい / となると嬉しい / するのは難しそうなソフトウェア」 パッケージにした / しようとして止まっているのとかはあります。

- mirmon - ミラーサーバの状態チェックができます。スポンサー待ちで半年以上...
- gnview - gtk2-perl の 2ch ブラウザ。スポンサー待ちです。
- FailSafeC - 安全な C 言語コンパイラ。upstream の man 作成やライセンスの確認待ち
- naist-jdic - non-free な ipadic の置き換え。upstream の作業待ち。

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

ゴールデンウィークって何? だったので...まっ、いつも通りにパッケージのアップデートとかでした。あ、そうそうミラーサーバの復旧作業してきました。機材は Ubuntu Japanese Team の方から提供いただき、増設 HDD は自費購入。

2.5 吉田@板橋

僕がゴールデンウィークに自由なソフトウェアのために行ったこと

ゴールデンウィーク (GW) には家のサーバのファイル名等のエンコーディングを eucJP から UTF-8 に変更する作業を行っていました。そろそろ変えておいた方が、

他の環境 (今時のディストリビューションや Windows 等) との相互運用性が良いと判断し、時間がとれる GW に作業しました。基本的にサービスを絞ったサーバ運用だったので、ファイル名変更のほかは samba や apache、bash 等の設定変更のみでいけました。その準備 (バックアップ) として、漢字ファイル名を使用しているファイルに対して kakasi およびハードリンクを使用してローマ字にリネームするスクリプトを (似たようなことをしている web の記述を参考に) 作成し、適用しました。そのログやスクリプトは web にある自分の備忘録に上げておきました、ご参考まで。

2.6 あけど

お題: パッケージになって嬉しい / となると嬉しい / するのは難しそうなソフトウェア

実は「フリーになって嬉しい / となると嬉しいソフトウェア」ってのがあるのですがたぶん他の方が触れられるかと思しますので、一つだけ挙げておきます。

adobe-cmap

これって元ネタは RFC らしいのですが、どうなったんでしょう?

では本題、パッケージになって嬉しいというか「こんなあったらなあ」というのを挙げてみます。テキストコンソールで日本語を扱っていると文字化けする事がよくあるのですが、ターミナルの設定に合わせてコンソールのロケール情報を自動設定してくれる仕組みって欲しかったりします。(実はあったりします?) termcap/terminfo の端末特性みたいな感じでフィルタを設定する仕組みとかどうかなあ? と思ってみたりしてます。

もう一つ、以前挙げた iptables のフィルタですが、だれかパッケージにしたりしませんか? というか、自分で作って見たら良いのかな?

2.7 福永

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

現在、フリーの Common Lisp で並列計算を支援するライブラリは少ないので、Common Lisp で手軽に並列計算を出来る為に MPI 並列プログラミングライブラリの Foreign Function Interface Library (言語バインディング) を作り始めました。基本的な通信 Primitive(blocking/nonblocking, system/self buffered) 等の Wrapper 実装を終了し、任意の Lisp コードを Send/Receive により他 CPU で実行出来るように

なりました。UFFI というポータブルなバインディングジェネレーターを用いたので、どの Common Lisp 実装でも使えるはずですが、今夏中、パッケージ化してリリースしたいと思います。

2.8 奥野

「フリーになると嬉しいソフトウェア」

Adobe のプラットフォーム (AIR とか Flash とか Flex とか) です。Flash がないと見られないサイトが増えましたし、AIR とか Flex とか一部ですげーとか言われていますが、プロプライエタリなので普及すればするほど Debian のフリー的には面白くないのでは。

2.9 沖中 研心

「パッケージになると嬉しいソフトウェア」

商用のソフトウェア全般ですね。最近 Atok の deb パッケージが用意されていたりして、うれしく思うことがあります。まだまだ Debian サポートしている商用ソフトウェアはほとんどないので、増えてくれるとうれしいですね。個人的には、トレンドマイクロの Internet Message Security Suite は、RedHat には対応していますが、Debian には対応していないので、何とかなればいいのになぁと思っています。

2.10 日比野 啓

表題: パッケージになって嬉しい / なる嬉しい / するのは難しそうなソフトウェア

Objective Caml を改造したもので MetaOCaml という実装 (<http://www.metaocaml.org/>) があるのですが、これを使うと、Lisp のようなメタプログラミングを OCaml の型推論システムの上で行なうことができるようになります。すると、メタプログラミングで生成した OCaml プログラムを型推論で型付けすることができて、型安全なメタプログラミングが行なえます。パッケージにも是非してみたいのですが、すでにある OCaml 処理系とのインストール位置の棲み分けやライブラリのパッケージングが面倒そうです。

2.11 山本 浩之

「フリーになって嬉しい / なる嬉しいソフトウェア」

VMware が DFSG フリーになってくれるとかなり嬉しいです。あと悪名高き Flash Player とか。CMap は早くフリーになってくれると良いのですが…。フリー

になったといえば djv ソフトウェアがありますが、私自身はあまり恩恵を受けていません。

「パッケージになって嬉しい / なる嬉しい / するのは難しそうなソフトウェア」

パッケージにするのは難しいというより、安定版としてリリースされるものに取り込むのが難しそうなソフトウェアなら、youtube-dl とか nicovideo-dl とか、あと 2ch ブラウザ類とかですかね？度々の仕様変更で全く使えなくなることもありますし。

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

ITP をゴールデンウィーク中にできれば良かったのですが、できませんでした。すみません。結局、今日のために、Debian GNU/Hurd いじりと、それを動かす環境作りで時間がなくなりました。

2.12 野村 健太郎

フリーになると嬉しいソフトウェア

Microsoft Office。金銭的・仕様の面からフリーになって欲しいという意味で。Office 文書がプラットフォームに関係なく使用可能になれば、PC から Windows 完全消去の実現に一步近づく。XP までは仕方なく使っていたが、Vista で我慢の限界。ノート PC にプリインストールされている Vista(最低 20G は必要)のおかげで、デュアルブートの Debian 用に思いっきりディスクを割けない。先月、Office 2007 の文書フォーマットである Open XML が ISO の標準として承認されたそうだが、GPL とは相容れないという情報もあり、どうなるのやら。

2.13 藤沢 理聡

「フリーになって嬉しい / なる嬉しいソフトウェア」

「自由」という意味のフリーだとちょっと思いつかないので、「無償」の意味の方であげるなら、Opera ブラウザです。様々な OS に対応している Web ブラウザの中で私の周囲で最も利用されているのは Firefox ですが、私自身は Opera 派なので、Opera が無償になったときは喜びました。Debian しか使わないなら特に Opera にこだわりますが、他の OS も頻繁に利用する環境では、できれば利用する全ての OS に対応していることが望ましいと思っています。

2.14 荒木 靖宏

「僕がゴールデンウィークに自由なソフトウェアのために行ったこと」

ちょうどGWは出張だったのですが、出張で成田に行く途中でメンテをしている sip-tester に strcat つかってんぞ、strncat のつかいかたが間違っているぞ、という debian security team からの BTS がやってきました。そ

のおかげでしばしたのしい思いをしながらヒコーキにのりこみ、出張先についてこちらからメールを出したら即バッチがやってきました。というわけで自分がやったことといえば、それを upstream に伝えるだけ。upstream に mail をしたら upstream は休暇中だよーというメールが戻ってきたのでした。

「フリーになって嬉しい/なると嬉しいソフトウェア」
自分が仕事で書いたコードですね。rails で書いているだけに公開したいしたい。

3 Debian Trivia Quiz

小林 儀匡



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` に投稿された内容と Debian Project News からです。

3.1 debian-devel-announce

`debian-devel-announce@lists.debian.org` への投稿内容からです。

問題 1. Joerg Jaspert が dak のソースコードに取り込んだ、Thomas Viehmann からのパッチとは？

A パッケージを受け取ったときにメンテナだけでなくスポンサーにもメールを送るようにするパッチ

B パッケージを受け取ったときに現在のカルマをメンテナに通知するパッチ

C パッケージを受け取ったときに現在のバグの一覧をメンテナに通知するパッチ

問題 2. debhelper バージョン 7 では、何から学んだことを取り込んでいるか

A MSDN

B CDDb

C CDbs

問題 3. 新たにリリースマネージャになったのは？

A Andreas Barth

B Luk Claes

C Marc 'HE' Brockschmidt

問題 4. Debian パッケージメンテナに関する自動通知メールにおいて、メールを生成したソフトウェアを示すのに使うよう提案されたメールヘッダは？

A X-Debian:

B X-Debian-Package:

C User-Agent:

3.2 2008 年 01 号

<http://www.debian.org/News/weekly/2008/01/>

にある 4 月 21 日版です。

問題 5. Siobhán O'Mahony と Fabrizio Ferraro の研究では Debian の統治システムの歴史を 4 つの時期に分けているが、そのうち 1999 年から 2003 年に当たるものは？

A Stabilizing Governance

B Implementing Governance

C Designing Governance

問題 6. Institute for Advanced Professional Studies Yankee group の研究結果として適切でないものは？

A 2006 年と比べて Debian サーバのダウンタイムが 41% 減った

B 2006 年と比べてネットワークに 1 台以上の Debian を入れているユーザが 15% から 24% に上昇した

C 2006 年と比べて Debian ユーザが 41% 減った

問題 7. 開発者のブログ記事を収集して表示する Planet Debian に関して、Martin Joey Schulze が始めたサービスとは?

A 記事をメーリングリストとして購読できるようにするサービス

B すべての記事に対してコメントを一斉に送れるようにするサービス

C 記事を Slashdot や各種ソーシャルブックマークサービスに容易に投稿できるようにするサービス

問題 8. Christian Perrier によると、最近正式な Debian 開発者として登録されたメンバーの女性率は?

A 5%

B 10%

C 50%

3.3 2008 年 02 号

<http://www.debian.org/News/weekly/2008/02/>

にある 5 月 9 日版です。

問題 9. WWW の生みの親として有名な Tim Berners-Lee が WWW2008 カンファレンスの基調演説で Debian について述べたこととは?

A Debian のウェブページのように古臭いページは消えていく運命にある

B Debian のウェブページは様々な言語に地域化されており素晴らしい

C Debian のパッケージングシステムは素晴らしい

問題 10. 今年の Google Summer of Code プログラムで Debian Project のタスクを得た学生の人数は?

A 6

B 12

C 24

問題 11. 5 月 9 日現在、lenny のリリースに向けて進められている移行の状況として適切でないものは?

A Perl のバージョン 5.10 を lenny でデフォルトとするための移行作業が行われている

B 既に lenny において Python のデフォルトがバージョン 2.5 になっている

C Ruby のバージョン 1.9 を lenny でデフォルトとするための移行作業が行われている

問題 12. 4 月 18 日に何人の Debian 正式開発者が追加されたか?

A 9

B 19

C 29

4 最近の Debian 関連のミーティング報告

上川 純一



4.1 東京エリア Debian 勉強会 39 回目報告

東京エリア Debian 勉強会報告。4 月の第 39 回東京エリア Debian 勉強会を実施しました。

今回の参加者はあけどさん、やまねさん、山本浩之さん、前田さん、沖中さん、岩松さん、吉田さん、山本琢さん、本庄さん、濱野さん、日比野啓さん、jitsukata さん、キタハラさん、藤沢理聡さん、でんさん、吉藤さん、上川の 17 人でした。

まず、クイズを今回も実施しました。今回も、debian-devel-announce の内容から出題しました。今回の景品はピンク色の「あれ」でした。

2008 年のテーマは DEB パッケージの開発・管理に関連した内容ですが、二回目のテーマとしてバイナリパッケージ一つの場合の話をしました。吉田さんが debhelper を使った流れを紹介しました。

岩松さんが Git を使った場合の Debian パッケージの管理方法について紹介しました。

上川が Nexenta Core Platform をインストールしてみても試した内容について報告しました。

今回は宴会は駒忠にて開催しました。

5 バイナリをわけたパッケージの扱い

前田 耕平



5.1 はじめに

今回は Debian パッケージを作成した経験がほぼ無い人を対象とし、先月^{*1} の『バイナリーつだけのパッケージを作成してみる』を予習していることを前提とします。

一応、前回の内容もおさらいしながら、今回も `dh_make` を使って、`debhelper` スタイルでの複数のバイナリに分けてパッケージを作る方法を説明しますので、前回参加できなかったとしても、ある程度は問題ないかと思います。

5.2 バイナリの分け方の考え方

通常のパッケージは、一つのバイナリパッケージとして作成・配布されるケースが多いでしょう。ではどのようなケースにおいて、バイナリパッケージは複数に分けられているのでしょうか。大きく分けると次の三パターンくらいになりそうです。

- 開発環境と実行環境
例)
 - Java の SDK と JRE
- C/S モデルのサーバ側プログラムとクライアント側プログラム
例)
 - `bind` と `dnstools`
- プログラム本体とアドオンで追加可能なモジュール、ユーティリティ
例)
 - Apache と Apache module や Apache Bench

これらに共通して言えるのは、単一のソースツリーから複数に分けることにより、導入するユーザの使い勝手が良くなる、ということだと言えます。

なお、複数のバイナリパッケージを作成するときも、ソースツリー及びソースパッケージ自体は単一なので、複数のソースツリー・ソースパッケージから生成される事はありません。

^{*1} 2008 年 4 月

5.3 前回のおさらい

今回は、dh_make を使って、debhelper スタイルで単一のバイナリパッケージを作りました。今回は、まず簡単な C のプログラムと Makefile を用意して、それを Debian パッケージにします。そこで単一のバイナリパッケージの場合と複数のバイナリパッケージに分ける場合との違いを見てみましょう。

5.3.1 ソースコードと Makefile の用意

まず、下記のようなディレクトリ、ファイルを用意します。

```
hoge-1.0/  
hoge.c  
Makefile
```

ファイルはそれぞれ下記のようなコードを書きます。

- hoge.c

```
#include <stdio.h>  
  
int main(void)  
{  
    printf('hello\n');  
    return 0;  
}
```

- Makefile

```
CC=gcc  
CFLAGS=-O  
BINDIR=/usr/games  
  
hoge: hoge.c  
    $(CC) $(CFLAGS) -o hoge hoge.c  
  
install:  
    install -d ${DESTDIR}${BINDIR}  
    install -m 755 hoge ${DESTDIR}${BINDIR}  
  
clean:  
    rm -f hoge
```

5.3.2 dh_make でテンプレートの作成

それでは、前回と同様に dh_make を使ってシングルパッケージを作ってみます。まず、dh_make でテンプレートを作ります。

```
$ cd hoge-1.0  
  
$ dh_make --single -c gpl --createorig  
Maintainer name : Kouhei Maeda  
Email-Address   : mkouhei@gmail.com  
Date            : Sat, 10 May 2008 15:36:29 +0900  
Package Name    : hoge  
Version        : 1.0  
License        : gpl  
Type of Package : Single  
Hit <enter> to confirm:  
Done. Please edit the files in the debian/ subdirectory now. You should also  
check that the hoge Makefiles install into $DESTDIR and not in / .
```

5.3.3 debian ディレクトリ以下のファイルの編集

debian ディレクトリ以下を編集します。

```
$ cd debian  
  
$ rm *ex *EX (<-不要なサンプルファイルを削除します。)
```

- debian/control

```
Source: hoge
Section: games
Priority: extra
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: hoge
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: sample program
 Hello hoge program
```

- debian/copyright

```
This package was debianized by Kouhei Maeda <mkouhei@gmail.com> on
Sat, 10 May 2008 15:36:29 +0900.

It was downloaded from <http://www.palmtb.net>

Upstream Author(s):

 <Kouhei Maeda mkouhei@gmail.com>

Copyright:

 <Copyright (C) 2008 Kouhei Maeda>

License:
(以下、省略)
```

- debian/dirs*²

```
usr/games
```

- debian/changelog

```
$ dch
hoge (1.0-1) unstable: urgency=low

 * Initial release

-- Kouhei Maeda <mkouhei@gmail.com> Sat, 10 May 2008 15:47:41 +0900
```

5.3.4 パッケージ作成

それでは、debuild コマンドでパッケージを作成してみます。

```
$ debuild -us -uc
```

deb ファイルができているのを確認します。

```
$ cd ../../
$ ls hoge_1.0*
hoge_1.0-1.diff.gz  hoge_1.0-1_i386.build  hoge_1.0-1_i386.deb
hoge_1.0-1.dsc     hoge_1.0-1_i386.changes  hoge_1.0.orig.tar.gz
```

上記のような簡単なものであれば、rules ファイルを修正しなくても、パッケージを作れる事が分かります。ちゃんとやるには、lintian のエラーや警告を潰し、pbuilder でのチェックやインストール、アンインストールのチェックをする必要がありますが、ここでは省略します。

5.4 複数のバイナリパッケージを作ってみる。

それでは、今度は先ほどの簡単なコードを一部変更して、それぞれ別のパッケージになるようにします。

5.4.1 ソースコード、Makefile の用意

下記のようなファイルを用意します。先ほどの hoge-1.0 をコピーして修正すればよいでしょう。

*² hoge のインストール先を指定します。

```
foo-1.0/  
foo.c  
bar.c  
Makefile
```

各ファイルは下記のようにします。

- foo.c

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("hello,foo\n");  
    return 0;  
}
```

- bar.c

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("hello,bar\n");  
    return 0;  
}
```

- Makefile

```
CC=gcc  
CFLAGS=-O  
BINDIR=/usr/games  
  
foo:  
    $(CC) $(CFLAGS) -o foo foo.c  
    $(CC) $(CFLAGS) -o bar bar.c  
  
install:  
    install -d ${DESTDIR}${BINDIR}  
    install -m 755 foo ${DESTDIR}${BINDIR}  
    install -m 755 bar ${DESTDIR}${BINDIR}  
  
clean:  
    rm -f foo bar
```

上記を make すると、foo と bar という実行ファイルができます。今回は、これらをそれぞれ foo パッケージと bar パッケージの別々のパッケージにしてみます。

5.4.2 dh_make の実行

単一のバイナリパッケージとは違い、dh_make を実行するときのオプションを”-single” から”-multi” に変更して実行します。

```
$ cd foo-1.0  
$ dh_make --multi -c gpl --createorig  
Maintainer name : Kouhei Maeda  
Email-Address   : mkouhei@gmail.com  
Date            : Sat, 10 May 2008 16:32:38 +0900  
Package Name    : foo  
Version         : 1.0  
License         : gpl  
Type of Package : Multi-Binary  
Hit <enter> to confirm:  
Done. Please edit the files in the debian/ subdirectory now. You should  
also  
check that the foo Makefiles install into $DESTDIR and not in / .
```

単一のバイナリパッケージの時と同様に、debian ディレクトリとその下にファイルが作成されます。

単一のバイナリの方に比べ、以下のファイルが増えていることが分かります。このテンプレートと同様なファイルが後ほどビルドする際に必要になります。

- foo-doc.docs
- foo-doc.install

5.4.3 debian ディレクトリ以下のファイルの編集

まず、単一のバイナリパッケージを作成するときと同じように修正して、ビルド出来るのか確認してみます。最初に先ほどと同じように今回は必要のないサンプルファイルを削除します。

```
$ cd debian
$ rm *ex *EX
```

次に、control ファイルを修正します。複数のバイナリパッケージを作成するには、空行で区切られた三つ以上のエントリが必要になります。

一つ目のエントリは、ソースパッケージ自身のことを記述するためのエントリです。二つ目以降のエントリが生成されるバイナリパッケージについての情報を記述するエントリです。

今回は、foo と bar の二つのパッケージを作成するので、dh_make で作成されたテンプレートを以下の様に変更します。

```
Source: foo
Section: games (<- セクションを変更)
Priority: extra
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: foo
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: sample program foo (<- 説明を記載)
foo foo foo (<- 説明の詳細を記載)

Package: bar
Architecture: any (<- 今回はアーキテクチャ依存なので any にする)
Description: sample program bar (<- 説明を記載)
bar bar bar (<- 説明の詳細を記載)
```

また、単一のバイナリパッケージの時と同様に、下記ファイルを修正します。

- debian/copyright
- debian/changelog
- debian/dirs

修正後、シングルバイナリの時と同様に debuild を実行してみます。

```
$ debuild -us -uc
dpkg-buildpackage -rfakeroot -D -us -uc
(中略)
dh_installdirs -s
# Add here commands to install the arch part of the package into
# debian/tmp.
/usr/bin/make DESTDIR=/home/user/foo-1.0/debian/foo
install
make[1]: ディレクトリ '/home/user/foo-1.0' に入ります
install -d /home/user/tmp/foo-1.0/debian/foo/usr/games
install -m 755 foo
/home/user/foo-1.0/debian/foo/usr/games
install: cannot stat 'foo': そのようなファイルやディレクトリはありません
make[1]: *** [install] エラー 1
make[1]: ディレクトリ '/home/user/foo-1.0' から出ます
make: *** [install-arch] エラー 2
dpkg-buildpackage: failure: fakeroot debian/rules binary gave error exit
status 2
debuild: fatal error at line 1319:
dpkg-buildpackage -rfakeroot -D -us -uc failed
```

どうやら、単一のバイナリパッケージの時にはちゃんと作成されていた実行ファイルが今回は作成されていない様です。単一のバイナリパッケージを作成する時は、rules ファイルの build-stamp ターゲットの中で、\$(MAKE) コマンドが記述されていましたが、複数のバイナリパッケージの場合は、rules ファイルの中で、\$(MAKE) がコメントアウトされており、デフォルトでは make が実行されません。そのため、rules ファイルを修正する必要があります。

”-single” の時の rules ファイル

```

build: build-stamp

build-stamp: configure-stamp
dh_testdir

# Add here commands to compile the package.
$(MAKE)
#docbook-to-man debian/hoge.sgml > hoge.1

touch $@

```

5.4.4 rules の記述

rules の必須ターゲットは、単一のバイナリパッケージの時と同様に、build, binary, binary-indep, binary-arch の 4 つです。

'-multi' オプションで作成される rules では build ターゲットは下記の様になっています。

```

build: build-arch build-indep

build-arch: build-arch-stamp

build-arch-stamp: configure-stamp

    # Add here commands to compile the arch part of the package.
    #$(MAKE)
    touch $@

build-indep: build-indep-stamp

build-indep-stamp: configure-stamp

    # Add here commands to compile the indep part of the package.
    #$(MAKE) doc
    touch $@

```

今回、foo も bar も、gcc でコンパイルするので、control ファイルでの Architecture セクションではどちらも any を記述しています。そのため、今回は rules ファイルでは、build ターゲットでは、build-arch、build-arch-stamp に依存しますが、build-arch-stamp の中でコメントアウトされている\$(MAKE) を有効にする必要があるので、有効にして、ビルドします。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -D -us -uc
(中略)
dpkg-buildpackage: full upload (original source is included)
Now running lintian...
(中略)
Finished running lintian.

```

今回は、うまくビルドできたように見えます。パッケージのビルド時には、debian/パッケージ名ディレクトリが作成され、そこにパッケージ含まれるファイルがコピーされます。ですので、ちゃんと期待するファイルがコピーされているのか確認してみます。

```

$ tree debian/{foo,bar}
debian/foo
|-- DEBIAN
|   |-- control
|   |-- md5sums
|-- usr
|   |-- bin
|   |-- games
|   |-- bar
|   |-- foo
|   |-- sbin
|   |-- share
|   |-- doc
|       |-- foo
|       |-- README.Debian
|       |-- changelog.Debian.gz
|       |-- copyright
debian/bar
|-- DEBIAN
|   |-- control
|   |-- md5sums
|-- usr
|   |-- share
|   |-- doc
|       |-- bar
|       |-- changelog.Debian.gz
|       |-- copyright
13 directories, 11 files

```

実行ファイル `foo`, `bar` が今度はきちんと作成されていますが、`bar` ファイルは期待するディレクトリ `bar/usr/games` 以下にコピーされていません。`bar/usr/games` ディレクトリ自体も作成されていない事が分かります。複数のバイナリパッケージを作成するには、`rules` ファイルを上記の変更だけでなく、さらにもう一ヶ所修正し、さらに以下のファイルを作成する必要があります。

5.4.5 rules ファイルで修正が必要な箇所

`debian/rules`

```

build: build-arch build-indep

build-arch: build-arch-stamp
build-arch-stamp: configure-stamp

    # Add here commands to compile the arch part of the package.
    $(MAKE) (<-最初の変更箇所。)
    touch $@
(中略)
install-arch:
    dh_testdir
    dh_testroot
    dh_clean -k -s
    dh_installdirs -s

    # Add here commands to install the arch part of the package into
    # debian/tmp.
    $(MAKE) DESTDIR=$(CURDIR)/debian/tmp install (<-debian/foo を
# debian/tmp に修正します。)

    dh_instll -s

```

5.4.6 新規作成するファイル

- `foo.install`

```
debian/tmp/usr/games/foo
```

- `bar.install`

```
debian/tmp/usr/games/bar
```

複数のバイナリパッケージを作成する場合、ビルド時には、通常、ソフトウェアのファイルやディレクトリを、まず `debian/パッケージ名ディレクトリ` 以外の一時ディレクトリ^{*3} に仮インストールし、そこから `debian/パッケージ名ディレクトリ` にコピーします。この時、`dh_install` コマンドと `debian` ディレクトリ以下のパッケージ名 `.install`

*3 通常は `debian/tmp` ディレクトリなどを指定します。

ファイルが使用されます。

今回の場合、debian/foo.install ファイルと、debian/bar.install ファイルが使われ、ここに記載したディレクトリやファイルが、それぞれのパッケージ用のディレクトリにコピーされます。

つまり、それぞれ以下のようにコピーされます。

- foo.install に記載された”debian/tmp/usr/games/foo” ファイル
パッケージ foo 用に debian/foo/usr/games/ディレクトリ以下に
- bar.install に記載された”debian/tmp/usr/games/bar” ファイル
パッケージ bar 用に debian/bar/usr/games/ディレクトリ以下に

では、実際にビルドしてみます。

```
$ debuild -us -uc
```

ビルドした結果の debian/パッケージディレクトリと仮インストール用の debian/tmp ディレクトリを見てみましょう。

```
$ tree tmp foo bar
tmp
|-- usr
    |-- games
    |   |-- bar
    |   |-- foo
foo
|-- DEBIAN
|   |-- control
|   |-- md5sums
|-- usr
    |-- games
    |   |-- foo
    |   |-- share
    |-- doc
        |-- foo
            |-- README.Debian
            |-- changelog.Debian.gz
            |-- copyright
bar
|-- DEBIAN
|   |-- control
|   |-- md5sums
|-- usr
    |-- games
    |   |-- bar
    |   |-- share
    |-- doc
        |-- bar
            |-- changelog.Debian.gz
            |-- copyright

14 directories, 13 files
```

今度はちゃんとコピーされています。パッケージが作成されているか見てみます。

```
$ cd ../../

$ ls foo_1.0* bar*
bar_1.0-1_i386.deb  foo_1.0-1_i386.build  foo_1.0.orig.tar.gz
foo_1.0-1.diff.gz  foo_1.0-1_i386.changes
foo_1.0-1.dsc      foo_1.0-1_i386.deb
```

pbuilder でチェック後、インストールしてみます。

```
$ sudo dpkg -i foo_1.0-1_i386.deb
bar_1.0-1_i386.deb
未選択パッケージ foo を選択しています。
(データベースを読み込んでいます ... 現在 222038 個のファイルとディレクトリがインストールされています。)
(foo_1.0-1_i386.deb から) foo を展開しています...
未選択パッケージ bar を選択しています。
(bar_1.0-1_i386.deb から) bar を展開しています...
foo (1.0-1) を設定しています ...
bar (1.0-1) を設定しています ...
```

インストールできているか確認してみます。

```

$ dpkg -L foo
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/foo
/usr/share/doc/foo/changelog.Debian.gz
/usr/share/doc/foo/README.Debian
/usr/share/doc/foo/copyright
/usr/games
/usr/games/foo

$ dpkg -L bar
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/bar
/usr/share/doc/bar/changelog.Debian.gz
/usr/share/doc/bar/copyright
/usr/games
/usr/games/bar

$ dpkg -s foo bar
Package: foo
Status: install ok installed
Priority: extra
Section: games
Installed-Size: 44
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Architecture: i386
Version: 1.0-1
Depends: libc6 (>= 2.7-1)
Description: sample program foo
foo foo foo

Package: bar
Status: install ok installed
Priority: extra
Section: games
Installed-Size: 40
Maintainer: Kouhei Maeda <mkouhei@gmail.com>
Architecture: i386
Source: foo
Version: 1.0-1
Description: sample program bar
bar bar bar

```

5.4.7 補足

ちなみに、`dh_make` で”-multi”ではなく、”-single”を指定しても、ソースツリーの Makefile と、`debian/rules` の記述を変更したところ、複数のバイナリパッケージに分ける事ができたので、結局は `make` 次第でどうにでもできるのではないかと思います。

5.5 まとめ

前回は踏襲して、`dh_make` を使って、`debhelper` スタイルで複数のバイナリパッケージを作ってみました。さらに、単一バイナリパッケージとの違いを分かりやすくするため、非常に簡単なプログラムをパッケージにすることで、その違いを見てみました。

単一バイナリパッケージを作成するときと同じような感覚でやると、`rules` ファイルのところで躓くので、パッケージの作成は `makefile` を読むことが重要だということがよく実感できました。

5.6 参考書籍

- 入門 Debian パッケージ (ISBN:978-4-7741-2768-X)
- GNU Make 第 3 版 (ISBN:4-87311-269-9)
- 第 39 会 東京エリアデビアン勉強会 (2008 年 4 月) 「バイナリーだけのパッケージを作成してみる」
- The Debian Systems その概念と技法 (ISBN:978-4-8399-1897-2)

月例 Debian GNU/kFreeBSD 大浦 真

Debian Project の大浦です。(現在、京都在住) 今月からこの場をお借りして Debian GNU/kFreeBSD の状況をお伝えることになりました。よろしくお願ひします。

Debian は、現在、その OS の中心として Linux カーネルを利用していますが、Debian GNU/kFreeBSD とは、カーネルとして FreeBSD のカーネルを利用し、その上に GNU C library と GNU のユーザランド、および、Debian のパッケージセットを移植したものです。FreeBSD のカーネルだけを使っているのので、“kFreeBSD”と呼んでいます。

そんな Debian GNU/kFreeBSD は、開発途上にあり、Debian の一部として、まだ、正式にはリリースされていません。しかし、FreeBSD のインストーラを改造した kFreeBSD 用のインストーラも用意されていますし、kFreeBSD 用の builddd も設置されていますので、比較的簡単に実機やエミュレータ上で試すことができます。また、X.Org も使えますので、それなりに使うことがで

きるようになってます。ただ、重要なパッケージで、builddd でビルドできず、バイナリが用意されていないものがあつたり、用意されていてもまともに動作しないものがあつたりもするので、まだまだ、いろいろとハックのしがいがあると思います。

Debian GNU/kFreeBSD に関する情報は、以下の Debian Wiki のページにまとまっていますので、ぜひご覧下さい。

<http://wiki.debian.org/Debian%20GNU/kFreeBSD>



月例 Debian GNU/Linux SuperH port 岩松 信洋

今月の Debian GNU/Linux SuperH 移植版の状況をお伝えします。

Debian/SH port は先月から加速しはじめました。その理由として、フランス人の方が協力してくれることになったためです。いまのところ、パッケージ用リポジトリを共同にしようということになっており、新しい SH4 用のリポジトリがフランスと日本にできました。

現在のステータスは、gcc-4.3 ベースでパッケージを作っています。マシンが複数台あるのですが、builddd をまだ 1 台しかセットアップしていません。今月中には複数台セットアップしたいと考えています。

現在抱えている問題として、mesa のコンパイルに失敗するというものがあります。internal compiler error なので、調べるのは難しいですが、マクロ展開でこけていくところまで追いました。絶賛調査中です。

```
gcc -c -I../include -I../src/mesa \
-I../src/mesa/main -I../src/mesa/glapi \
-I../src/mesa/math -I../src/mesa/tnl \
-I../src/mesa/shader \
-I../src/mesa/shader/grammar \
-I../src/mesa/shader/slang -I../src/mesa/swrast \
-I../src/mesa/swrast_setup -Wall -Wmissing-prototypes \
-O2 -g -D_POSIX_SOURCE -D_POSIX_C_SOURCE=199309L \
-D_SVID_SOURCE -D_BSD_SOURCE -D_GNU_SOURCE -DPTHREADS \
-DUSE_XSHM -DHAVE_POSIX_MEMALIGN -I/usr/X11R6/include \
-std=c99 -ffast-math -fno-strict-aliasing \
vbo/vbo_exec_api.c -o vbo/vbo_exec_api.o
vbo/vbo_exec_api.c: In function 'vbo_exec_fixup_vertex':
vbo/vbo_exec_api.c:340: internal compiler error: Segmentation fault
Please submit a full bug report,
with preprocessed source if appropriate.
See for instructions.
For Debian GNU/Linux specific bug reporting instructions,
see .
```

今月のパッチ

- util-linux
sh4 アーキテクチャで mount パッケージが作成されない問題を修正。

月例 Debian GNU/Hurd 山本 浩之

今月の Debian GNU/Hurd の状況をお伝えします。Debian GNU/Hurd は sid に入っていないながら、未だ testing にすら入れてもらえず、冷や飯を食わされ続けていますが、それでもなお、開発者たちは地味に頑張りつづけ、現在では K16CD インストールイメージをリリースするに至っております。まずはこの K16CD-mini インストールイメージを VMware ディスクイメージにインストールしてみて問題点を調べてみました。

ユーザランドは基本的には Debian GNU/Linux と同じですが、いくつか Hurd 特有の呪文があります。

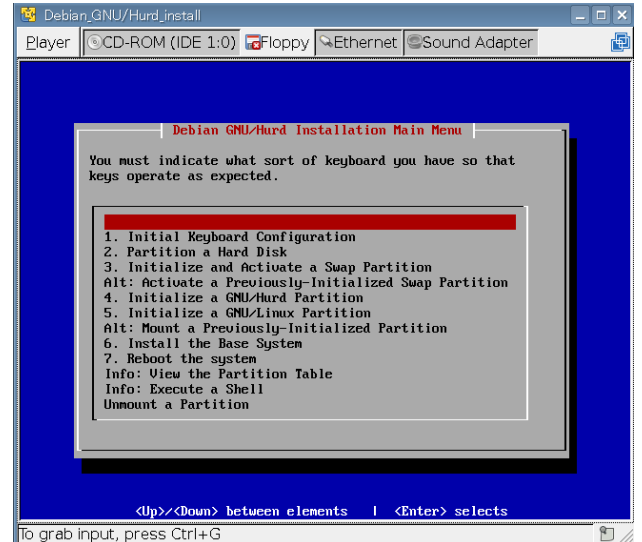
まず、ネットワークの設定ですが、`/sbin/ifconfig` にあたるものがなく、

```
# settrans -fgap /servers/socket/2 /hurd/pfinet \  
-i eth0 -a 192.168.1.3 -g 192.168.1.1 -m 255.255.255.0
```

というように、`settrans` コマンドを使用します。また X を起動するにはかなり謎 (?) な呪文を唱えないといけません :

```
# console -d vga -d pc_mouse --repeat=mouse -d pc_kbd \  
--repeat=kbd -d generic_speaker -c /dev/vcs
```

現在のところ、Gnome や KDE のような統合デスクトップ環境は、build-dep なパッケージが FTBFS バグのため揃っておらず、提供されていませんが、Window Maker、Afterstep、IceWM、qvwmm、ratposion、fvwm、ctwm などのウィンドウマネージャが使用可能です。



月例 Nexenta Operating System 上川 純一

2008 年 4 月号でインストールしてみた Nexenta Core Platform 1.0 で今月も何か操作してみます。まず、ホスト OS とのデータの交換の方法を確立してみます。

仮想マシン内部を快適に利用するために、まず SSH 接続経由でのファイルシステムを共有をしてみましょう。まず、ゲスト OS の 22 番ポートをホスト OS の 2299 番ポートとしてゲスト OS を起動してみました。こうすると ssh で localhost の 2299 番ポートにアクセスしてゲスト OS にログインできます。

```
dancer@host$ qemu -hda nexenta.cow -m 512 -redir tcp:2299::22  
dancer@host$ ssh localhost -p 2299  
Password:  
Last login: Wed Apr 30 05:02:35 2008 from 10.0.2.2  
dancer@vm1:~$
```

この状態で `sshfs` を利用するとゲスト OS のパスをマウ

ントして見ることができます。ホスト OS からゲスト OS の `/export/home/dancer` にシームレスにアクセスすることができます。

```
dancer@host$ sshfs localhost: ./nexenta/ -p2299  
Password:  
dancer@host$ ls -la nexenta/  
合計 36  
drwxr-xr-x 1 dancer uucp 9 2008-04-15 23:53 .  
drwxr-xr-x 5 dancer dancer 4096 2008-04-30 14:05 ..  
-rw----- 1 dancer uucp 1130 2008-04-23 13:05 .bash_history  
-rw-r--r-- 1 dancer uucp 220 2008-03-19 18:07 .bash_logout  
[以下略]
```

nexenta では各種ファイルを `staff` グループ (`gid=10`) が所有し、それがちょうど Debian では `uucp` グループ (`gid=10`) が所有しているように見えています。

```
dancer@vm1:~$ id dancer  
uid=1000(dancer) gid=10(staff) groups=10(staff)
```




Debian 勉強会資料

2008年5月15日 初版第1刷発行

東京エリア Debian 勉強会（編集・印刷・発行）
