

東京エリア デビアン 勉強会



Debian勉強会幹事 上川純一

2009年5月16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。

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 Tokyo
3. VAIO P インストール記録、カーネル読書会 ディストリビューション大集合 (小林さん)(東京大学?)
4. Git Handson (岩松)(あんさんぶる荻窪?)
5. 家 Debian サーバ vs 職場のネットワーク (千代田区都立図書館?*¹)
6. Asterisk (東京大学?)
7. スペインにて開催
8. Debconf 報告会
9. OSC Fall?
10. udev + HAL(岩松さん)
11. 3D graphics 開発 (藤沢さん)
12. Debian サーバ + VMware + 各種 OS、他の仮想化ツール (vserver etc.)、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/>

今更な勉強会 レポート

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	6
4	Debian Trivia Quiz	7
5	Erlang コードを Debian パッケージにしてみた	8
6	MC-MPI/GXP 公式パッケージへの道	17
7	Debian GNU/Linux 上で Android SDK を使ってみた	24
8	DDTSS 活用	30

2 事前課題

上川 純一

事前課題は:

DDTSS でいくつか (2 個以上) Debian パッケージの説明文を翻訳してみて、2 個レビューしてみてください。その上で次の内容について説明してください:

1. 適用した主要な方法
2. 発見した課題
3. 提案する理想像 (ツールとか)、共有したい情報

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 適用した主要な方法

DDTSS をブラウザで眺めて、翻訳をレビューしてみました。パッケージの description だけで分からない部分についてはバグレポートをしました。不明点は `debian-doc@jp` メーリングリストに質問として投稿しました。

2.1.2 発見した課題

レビューをして分かったことですが、翻訳の作業で単語の翻訳まではできているのですが、文章全体として係り受けがおかしく、意味が通っていないものがありました。また、英語の句読点をそのまま日本語の句読点に置き換えており、そのままでは日本語として文章が長すぎるものもありました。翻訳作業を直訳作業とすると読みにくい Description ができあがってしまうと思われます。

2.1.3 提案する理想像 (ツールとか)、共有したい情報

- Description に対してすでにバグレポートが投稿されているかどうかのチェックするツール。
- 用語集にすでに掲載されている用語を簡単にウェブブラウザからチェックできる `greasemonkey` 。
- DDTSS で行ったレビュー・コメントなどが

`debian-doc@jp` メーリングリストに反映すること。

2.2 まえだこうへい

2.2.1 適用した主要な方法

事前課題まで手が回らずじまい。ただだと芸がないので、現在友人と進めている CouchDB の Web サイトの翻訳について話します。

1. CouchDB の Web サイトを Google Docs でまるごと取り込む。1 ページにつき、翻訳者は限定。他は査読し、一文ずつ翻訳する。直訳ではなく、日本語としてわかりやすい文章を重視。
2. Web サイト、Wiki をある程度の比率まで翻訳を進める。
3. CouchDB の開発者向けの ML に公開したい旨を相談する。
4. Debian の CouchDB のパッケージメンテナでもある、開発者から `po4a` を使うこと、他にもいろいろ課題 (システム管理、アップロードの方法、翻訳の陳腐化など) があるけどそれを考慮しないといけないよと、アドバイスをもらう。
5. PO 形式でパッチ投げて反映自体はお願いし、更

新はなんらかの手段で確認して随時翻訳していく旨を伝える。

6. 今後の方針をどうするかを友人と相談して、今ココ。

2.2.2 発見した課題

Web サイトは XHTML になっていないので、まず変換するところから始めないといけないことと、Wiki は更新が頻繁にありすぎだから、対象から外した方が良いよね、と友人と相談したところ。翻訳がそもそもやりたいいことではなかった。

2.2.3 提案する理想像 (ツールとか)、共有したい情報

最初の翻訳だけ行ったら、後は手離れする (誰かに引き継ぐ) のが理想ではあるものの、協力者を増やさないといけない。そもそも Debian パッケージになっているので、ソースパッケージに含まれるドキュメントなら、Debian-doc で翻訳して、そっち経由でアップストリームに反映してもらえ、というのも手だなということを検討中。

2.3 明渡忠郎

2.3.1 適用した主要な方法

手入力とネット上の英和辞典、必要に応じて Google、wikipedia、その他専門サイトを参照

2.3.2 発見した課題

DNA 解析ツールが多くあり、専門用語がまちまちでその統一若しくは共通化が必要だと思いました。

2.3.3 提案する理想像 (ツールとか)

専門用語については辞書ページを用意するか、基準にする専門用語のサイトを定めるのが良いかと思います。

共有したい情報 Debian パッケージには数多くの DNA 解析ツールがあって、専門用語の翻訳にあたって注意深く作業する必要があると思います。DNA 解析について予備知識として下記サイトが参考になるかと思います。

タンパク質 / 核酸のアライメント解析^{*2}

Debian JP 文書作成の指針 について DDTSS for ja にリンクがあった方が良いかと思います。

^{*2} http://www.icot.or.jp/ARCHIVE/Museum/SOFTWARE/GIP/gene_alignment.html 文字コードが ISO-2022-JP なのにエンコード情報は Shift-JIS になっています。

2.4 日比野 啓

DDTSS で ocaml と libperl4caml-ocaml の Description を翻訳してみました。レビューは nasm と libobjc2 をやってみました。

2.4.1 適用した主要な方法

DDTSS の Web インターフェースから翻訳とレビューを行ってみました。

2.4.2 発見した課題

自分が翻訳を行なった内容が更新されたときやコメントが付いたときに、再び見に行かないと更新を知ることができないのが気になりました。

2.4.3 提案する理想像 (ツールとか)、共有したい情報

翻訳した人やレビューした人に更新を通知する機能とそれをコントロールする機能が欲しいです。

2.5 小川 伸一郎

以下、やってみた感想など。

2.5.1 登録

まず登録時の Alias と言うのが直感でわかりませんでした。

2.5.2 翻訳

翻訳は以下の 2 つのパッケージについてやってみました。

- wgerman-medical
- libbio-ruby

思ったのは、いきなり全てを訳さなければならないのは辛いんじゃないかということ。少しずつ翻訳できたり、もしくは途中結果を保存できる方がいいんじゃないかと思いました。

あとは、最初にどう何をすればいいのかがわかりにくということ。わかった当たり前なのかも知れませんが、幅広く協力者を求めるのなら、何かしら情報 (HowTo とか?) あった方がい気がします。

2.5.3 レビュー

いきなり 'Accept as is' と書かれるのが怖かった。ちゃんと, "Accept as is' means you agree with this translation." と書かれてはいるんですが, どうにも「えっ, いいのか?」と言う思いが抜けきれませんでした。ので, レビューは見ただけです。

で, multimap を見たときの感想ですが, diff をもう少しわかりやすく表示できないかと思いました。さすがに前後にあると, 頭の中でいろいろ組み立てないと読めないの, 上下に異なる部分だけ文字長を併せて表示できたりすると, わかりやすいかなと。

「思った」ばかりの感想になってしまいましたが, こんな感じでした。

2.6 やまだたくま

2.6.1 適用した主要な方法

DDTSS より原文と訳文を入手して、翻訳支援ツール OmegaT またはテキストエディタで翻訳しました。複数のオンライン版の辞書、ペーパー版の辞書、翻訳ソフト、対象ソフトウェア、Web 検索、書籍の調査を併用しました。レビュー依頼を debian-doc メーリングリストに投稿しました。情報提供依頼を YLUG メーリングリストに投稿しました。

2.6.2 発見した課題

Deiban JP 関連の翻訳のシステムと運用についての情報収集の難易度が高い状態でした。ドキュメントが整備されていない情報や、数年間アップデートされていないページがありました。

要求される訳の品質が不明なため、作業時間や作業手順の見積りが困難でした。

訳の品質安定に必要な執筆基準の整備状況が不十分でした。

2.6.3 提案する理想像 (ツールとか)、共有したい情報

- 作業手順ページのアップデート。
- 執筆基準と用語集のアップデート。
- 執筆基準適合テストツールや翻訳支援ツール用ライブラリの提供ができるとうれしいでしょう。

2.7 やまねひでき

2.7.1 適用した主要な方法

DDTSS をブラウザで眺めて、翻訳&レビュー...したような記憶が微かにあります。

2.7.2 発見した課題

一人だと寂しすぎる。というのはさておき、全体像と現状の把握が難しく、一体どこからこの山を登ればいいのかよく分からない状態だった記憶が。

2.7.3 提案する理想像 (ツールとか)、共有したい情報

- 見やすいページ構成が欲しい。素朴過ぎて何が何やら分からんのは勘弁。せめて po-debconf のページ (<http://www.debian.org/international/l10n/po-debconf/ja>) レベルのは欲しい。
- 「一ユーザが読んで理解できる」文章を心がけること。当たり前なんですけど、難しいので改めて(逆に言うと、査読はユーザにってもらう仕組みが欲しいところです)。

2.8 山本浩之

2.8.1 適用した主要な方法

DDTSS をブラウザで眺めて、Cross translation の翻訳結果を参考にしながら、翻訳とレビューをしました。

2.8.2 発見した課題

翻訳に時間がかかりすぎたせいか、なぜかロックがはずれて他の人と重なったみたいです。やっと翻訳したものが消えてしまった時は泣けた。(T.T)

2.8.3 提案する理想像 (ツールとか)、共有したい情報

- なんか使い方が良く分からなかったの、使い方のドキュメントがあると良いです。
- コメント欄になにを書けば良いのかも良く分からなかったです。みんなは何を書いているんだろう?

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 51 回目報告

東京エリア Debian 勉強会報告。2009 年 4 月 18 日土曜日に東京エリア Debian 勉強会の第 51 回をあんさんぶる荻窪にて開催しました。

今回の参加者は中尾、北原、明渡、眞戸江、小川、山本、山田たくま、高橋 (masaka)、前田耕平、吉田@板橋、日比野、吉藤、でんすけ藤澤りそう、上川 x2 の 16 名でした。

まず、クイズ。

前田さんが java policy を読んでみたので話をしました。まだパッケージ作成には着手していないようなので今後パッケージ作成レポートが出てくるかな？

上川が ocaml をつかってみたので報告。Debian 上で ocaml をつかったパッケージが意外とたくさんあるということで盛り上がりました。しかしまだ十分入門できていないのでこれからかな。

その後、Debian 開発のワークフローについてディスカッション。事前課題を紹介していたら時間が尽きたので深い議論はなかった感じで。いろいろと紹介されていたので面白かったのではないのでしょうか。

宴会ははなの舞西荻窪店で。

4 Debian Trivia Quiz

上川 純一



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。

問題 1. Debian Project Leader からアシスタントに任命されたのは

- A Nobuhiro Iwamatsu
- B Hiroyuki Yamamoto
- C Luk Claes

問題 2. Debian Groupware Meeting が開催されたのはどこだったか

- A New York
- B Ogikubo, Suginami, Tokyo
- C LinuxHotel, Essen, Germany

問題 3. Debconf で 249 ユーロで配布される予定の携帯電話は？

- A Openmoko Neo Freerunner
- B iPhone
- C Android

問題 4. Debian Secretary Team にはいないのは？

- A Kurt Roeckx
- B Neil McGovern
- C Kohei Maeda

問題 5. Andrew McMillan, Bill Allombert とともに Policy Team に追加されたのは

- A Colin Watson
- B Junichi Uekawa
- C Takuma Yamada

問題 6. DDTSS の URL は

- A `http://www.2ch.net/`
- B `https://launchpad.net/rosetta`
- C `http://dntp.debian.net/ddtss/index.cgi/`

xx

5 Erlang コードを Debian パッケージにしてみた

前田 耕平



5.1 入門のきっかけ

Erlang というプログラミング言語があります。今回、これを勉強してみようと思ったのは、Erlang そのものがきっかけではなく、いつものように別のものがきっかけです。今回は、CouchDB という Apache Incubator^{*3}にあった^{*4}、ドキュメント指向のデータベースを友人から教えてもらったのがきっかけでした。CouchDB が Erlang と JavaScript で実装されている、というので、「Erlangって何だろう？」と興味を持ちました。

5.1.1 ちょっと脱線 : CouchDB とは

CouchDB の特徴には以下のものがあります。

- ドキュメント指向
- スキーマレス
- RESTful
- JSON 形式でデータをやりとり

RDB のように事前にしっかりとスキーマの定義を行っておく必要はなく、HTTP PUT でデータ (=ドキュメント) を作成・更新し、GET でデータを取得し、DELETE でデータを削除する、ちょっと変わったデータベースです。RDB を置き換えるものではないので、摘要範囲は正直まだよく分かりません。ですが、面白そうなので友人とこれを広めるべく、目下水面下で活動中です。

CouchDB のアーキテクチャについては、CouchDB Project のホームページから転載した、下記のシステム概念図をご覧ください。これらの実装は主に Erlang によるものです。

また、CouchDB については、水面下の活動の一環で翻訳なども行っているのですが、また別の機会にお話できればと思います。

5.1.2 閑話休題 : Erlang の概要

本題の Erlang ですが、CouchDB に興味を持ち、さらにその中身をどうなっているのを知りたいと思ったのですが、Erlang という言語を知りません。関数型プログラミングで、並列処理、分散処理、耐障害性を兼ね備えた、並行指向プログラミングというのですが、そもそも文法がまったく分からないので、ソースコードだけを眺めて理解でき

^{*3} <http://incubator.apache.org/>

^{*4} 現在は、Apache Incubator から卒業して Apache Project の一つになっています。 <http://couchdb.apache.org/>

るものではないな、ということから勉強を始めてみました。また、並列処理、分散処理に興味を持っていたので、興味があまい具合に重なったのも、きっかけとなりました。

Erlang の歴史

Erlang は、Ericsson Computer Science Laboratory で、分散化された環境、耐障害、ある程度のリアルタイム性、無停止で稼働する、といった条件のシステムを構築できるように設計されたプログラミング言語です。もともとは、エリクソンの社内だけで使われていましたが、1998 年にオープンソースとして公開されました。ライセンスは、Erlang Public License^{*5}で、Mozilla Public License の派生ライセンスです。関数型で並列指向のプログラミング言語ということですが、20 年を超える歴史と、高度な信頼性が要求される通信機器の分野でも実績があり、例えばエリクソンの AXD301^{*6}などがあるそうです。

仕様

プログラミング Erlang という書籍を一通り読んで、今 2 周目で実際に手を動かしている段階ですが、その中で興味を持った特徴を挙げます。

- 評価できるものは全て式
- 変数の書き換えはできない
- 関数型プログラミング言語
- 並行処理 (並行プログラミング)

自分には馴染みが薄いものばかりです。特に、「変数の書き換えはできない」なんて、定数じゃないの? と思いました。詳しくは後述。

Erlang による最近の実装例

Erlang で実装されているソフトウェアの例です。これらは deb パッケージになっており、Lenny にも含まれています。

- CouchDB
- ejabberd^{*7} twitter で使われているという IM サーバ
- YAWS(Yet another web server)^{*8} Erlang で実装された Web サーバ

Erlang の検証事例

数年前から、一部では流行りらしいので、検証事例もちらほら出ています。

- KLab 株式会社による検証
 - Erlang Performance
http://lab.klab.org/wiki/Erlang_Performance
 - Erlang Process
http://lab.klab.org/wiki/Erlang_Process
- Erlang vs. Stackless python: a first benchmark
<http://muharem.wordpress.com/2007/07/31/erlang-vs-stackless-python-a-first-benchmark/>

一般的に普及していくのはまだこれからみたいな感じですが、あまり使われていないうちに遊んでおこうと思います。

^{*5} <http://erlang.org/EPLICENSE>

^{*6} ATM スイッチ

^{*7} <http://ejabberd.jabber.ru/>

^{*8} <http://yaws.hyber.org/>

5.2 環境を整える

Debian では数はまだ少ないものの Erlang と Erlang で書かれたソフトウェアのパッケージがあります。

```
$ apt-cache search erlang | wc -l
63
```

ちなみに、例として挙げた CouchDB や YAWS も deb パッケージになっています。

Erlang を使うために最低限必要なパッケージは次の erlang-base パッケージです。対話型ではなく、ソースコードを書くのには emacs の Erlang 用のモードもあるので、それも一緒にインストールしておくといいでしょう。

```
$ sudo apt-get install erlang-base erlang-mode
```

ところで、erlang-base パッケージではなく、erlang パッケージをインストールすれば、erlang ソースパッケージから分割されている関連パッケージも軒並みインストールされるので便利かもしれませんが、この場合は依存関係も含めて関連するパッケージが 90 以上導入されます。状況に応じて適宜選択してください。

```
$ sudo apt-get install erlang
```

5.3 対話型で試してみる

対話型で Erlang を使うには、erl コマンドを実行します。すると、Erlang shell(Eshell) が起動します。

```
$ erl
Erlang (BEAM) emulator version 5.6.5 [source] [64-bit] [smp:2] [async-threads:0] [kernel-poll:false]

Eshell V5.6.5 (abort with ^G)
1>
```

“1>” はプロンプトです。Eshell を終了するには、q(). と入力するか、

```
1> q().
ok
2> $
```

または、ctrl+ c, a と入力します。

```
1>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
a
$
```

では、変数を書き換えられない、という件の特徴をみてみます。

```
1> A=10.
10
2> A=20.
** exception error: no match of right hand side value 20
```

見ての通り、エラーになってしまいました。Erlang の変数の書き換えはできない、単一代入変数なのです。もう一度代入の仕方を変えて見てみましょう。

```

1> A1=1.
1
2> A2=2.
2
3> A3=A1+A2.
3
4> A3=3.
3
5> A3=A1+2.
3
6> A3=A1+4.
** exception error: no match of right hand side value 5
7> A1=A2+A3.
** exception error: no match of right hand side value 5
8> A1=A3-A2.
1

```

1,2,3のように未代入の変数に対しては値を再代入できますが、6,7のように一度代入された変数に対しては違う値を代入しなおすことができないことが分かります。しかし、4,5,8のように同じ値であれば再代入できるように見えます。実は、4,5,8は変数を再代入しているのではなく、代入済みの変数に対し、パターン照合を行っているのです。

代入済みの変数のことを束縛済み変数、と言いますが、未束縛状態の変数に対しては、イコールは値の代入を行いますが、束縛済みの変数に対してはパターン照合を行う、という訳です。

今までこんな仕様のプログラミング言語は見たことなかったのですが、プログラミング言語ではなく、実は数学の代数であれば同じような考え方ができることが分かります。例えば、 $x + y = 6$, $x - y = 2$ という関数を考えてみます。この状態では、 x , y の代数は $x=4$, $y=2$ 以外の値にはなりません。単一代入変数は、数学の代数と同じようなものと考えれば、良いのかもしれませんが。

5.4 コンパイルしてみる

次に、ソースコードからコンパイルして実行する方法について説明します。プログラムを実行する方法は、次の3種類があります。

- Eshell からコンパイルして実行する。
- シェルからコンパイルして実行する。
- escript として実行する。

例としてはやはり、Hello World を書いてみます。これは引数なしのプログラムです。引数ありとなしではプログラム実行時に若干の違いが出てきます。hello.erl というファイル名で次のプログラムを用意して下さい。

```

-module(hello).
-export([start/0]).

start() ->
  io:format("Hello world on Erlang~n").

```

Erlang のプログラムはモジュールを基本単位とし、どの関数もモジュールに属する構造になっています。コードを実行するには、まずモジュールをコンパイルしなければなりません。コンパイルしたモジュールは、拡張子 beam^{*9} という名前のファイルが生成されます。コンパイルには2つのやり方があります。

5.4.1 Eshell からコンパイルする方法

先ほどの hello.erl を Eshell からコンパイルし、実行するには、次の手順です。

```

1> c(hello).
{ok,hello}
2> he
heart    hello
2> hello:start().
Hello world on Erlang
ok
3>

```

^{*9} Bogdan's Erlang Abstract Machine の略。

c(hello). でコンパイルの実行を行います。c() の引数が module(hello) および、プログラムファイル名 hello.erl と一致することに注意してください。コンパイルすると、hello.beam ファイルが生成されます。これは Erlang VM 向けのバイナリコードです。

```
$ ls -lrt hello.*
-rwx----- 1 user user   90 2009-05-14 01:35 hello.erl
-rw----- 1 user user  552 2009-05-14 01:35 hello.beam
$ file hello.beam
hello.beam: Erlang BEAM file
```

5.4.2 シェルからコンパイルする

シェルからコンパイルする場合は、erlc コマンドを使います。

```
$ erlc hello.erl
$ ls -lrt hello.*
-rwx----- 1 user user   90 2009-05-14 01:35 hello.erl
-rw----- 1 user user  708 2009-05-14 01:53 hello.beam
```

実行もシェルから行ってみます。

```
$ erl -noshell -s hello start -s init stop
Hello world on Erlang
```

無事、実行できました。なお、各引数は次の意味があります。

- -noshell
対話型シェルなしで Erlang を起動する。
- -hello start
関数 hello:start() を実行する。
- -init stop
apply(hello, start, []) が完了すると^{*10}、システムは関数 init:stop() を評価する。

5.4.3 escript で実行する

Erlang では escript という仕組みもあり、これを使うとプログラムをコンパイルせずにスクリプトとして直接実行できます。

関数 hello:start を escript として実行するには、次のようにファイルを書き換えます。ファイル名は今回、hello.esc としましたが、コンパイルするときと違い、任意の拡張子で問題ありません。

```
#!/usr/bin/env escript
main(_) ->
  io:format("Hello world on Erlang\n").
```

これを実行権限をつけて実行してみます。

```
$ chmod 700 hello.esc
$ ./hello.esc
Hello world on Erlang
```

5.4.4 引数ありの関数をコンパイル、実行する

今回の例は、引数なしでしたが、シェルで実行する場合のプログラムはどうやって引数を渡せば良いのでしょうか。今度は引数ありのプログラムを試してみます。

フィボナッチ数列を計算する関数を書いてみました。まず、Eshell で引数を渡す場合です。

^{*10} -s ... コマンドは apply 文で評価される。1 つのコマンドが完了すると次の -s ... コマンドが評価される。

```
-module(fib).
-export([fib/1]).

fib(0) -> 1;
fib(1) -> 1;
fib(N) ->
    fib(N-2) + fib(N-1).
```

実行するとこんな感じ。

```
1> c(fib)
1> .
{ok,fib}
2> fib:fib(10).
89
3> fib:fib(20).
10946
4> fib:fib(30).
1346269
```

シェルから実行する場合は、書き方を変える必要があります。

```
-module(fib1).
-export([main/1]).

main([A]) ->
    I = list_to_integer(atom_to_list(A)),
    F = fib(I),
    io:format("fibonacci ~w = ~w~n", [I, F]),
    init:stop().

fib(0) -> 1;
fib(1) -> 1;
fib(N) ->
    fib(N-2) + fib(N-1).
```

これをシェルでコンパイル、実行してみます。

```
$ erlc fib1.erl
$ erl -noshell -s fib1 main 10 -s init stop
fibonacci 10 = 89
$ erl -noshell -s fib1 main 20 -s init stop
fibonacci 20 = 10946
$ erl -noshell -s fib1 main 30 -s init stop
fibonacci 30 = 1346269
```

5.5 deb パッケージにしてみる

前述のとおり、Erlang は beam ファイルからコードをロードするか、`escript` で実行する必要があります。Erlang のランタイムシステムは、コード自動ロード機構を利用しています。現在のロードパスは Eshell から `code:get_path()` 確認することができます。

```
1> code:get_path().
[".", "/usr/lib/erlang/lib/kernel-2.12.5/ebin",
"/usr/lib/erlang/lib/stdlib-1.15.5/ebin",
"/usr/lib/erlang/lib/xmerl-1.1.10/ebin",
"/usr/lib/erlang/lib/webtool-0.8.3.2/ebin",
"/usr/lib/erlang/lib/typer-0.1.5/ebin",
"/usr/lib/erlang/lib/tv-2.1.4.2/ebin",
"/usr/lib/erlang/lib/tools-2.6.2/ebin",
"/usr/lib/erlang/lib/toolbar-1.3.0.1/ebin",
"/usr/lib/erlang/lib/test_server-3.2.4/ebin",
"/usr/lib/erlang/lib/syntax_tools-1.5.6/ebin",
"/usr/lib/erlang/lib/ssl-3.10/ebin",
"/usr/lib/erlang/lib/ssh-1.0.2/ebin",
"/usr/lib/erlang/lib/snmp-4.12/ebin",
"/usr/lib/erlang/lib/sasl-2.1.5.4/ebin",
"/usr/lib/erlang/lib/runtime_tools-1.7.3/ebin",
"/usr/lib/erlang/lib/public_key-0.1/ebin",
"/usr/lib/erlang/lib/pman-2.6/ebin",
"/usr/lib/erlang/lib/percept-0.7.3/ebin",
"/usr/lib/erlang/lib/parsetools-1.4.5/ebin",
"/usr/lib/erlang/lib/otp_mibs-1.0.4.1/ebin",
"/usr/lib/erlang/lib/os_mon-2.1.8/ebin",
"/usr/lib/erlang/lib/orber-3.6.10/ebin",
"/usr/lib/erlang/lib/odbc-2.10.3/ebin",
"/usr/lib/erlang/lib/observer-0.9.7.4/ebin",
"/usr/lib/erlang/lib/mnesia-4.4.7/ebin",
"/usr/lib/erlang/lib/megaco-3.9.1.1/ebin",
"/usr/lib/erlang/lib/inviso-0.6/ebin",
[...]|...]
```

先頭に、“.”が入っているため、カレントディレクトリにある、beam ファイルは自動的にコードがロードされるため、実行できたわけです。コードパスは /usr/lib/erlang/lib が共通しています。これは、Erlang のデフォルトのライブラリディレクトリで、code:lib_dir() で確認できます。

```
1> code:lib_dir().
"/usr/lib/erlang/lib"
```

先ほど作成した hello モジュールもこのライブラリディレクトリ配下に配置します。しかし、ここに配置すると自動的にロードパスに入るわけではないので、実際に必要な場合は、ちゃんと指定する必要があります。

さて、hello-erlang-1.0 というディレクトリを作り、hello モジュール関係は全てこのディレクトリに移します。

```
$ mkdir hello-erlang-1.0
$ mv hello.erl hello.esc
```

Makefile を用意します。

```
.SUFFIXES: .erl .beam .yrl

.erl.beam:
    erlc -W $<

BINDIR?=/usr/lib/erlang/lib/hello-1.0/ebin

ERL = erl -boot start_clean

# compile erlang module list
MODS = hello

all: compile
    ${ERL} -noshell -s hello start -s init stop

compile: ${MODS:%=%.beam}

install:
    install -d ${DESTDIR}${BINDIR}
    install -m 644 hello.beam ${DESTDIR}${BINDIR}
    install -m 755 hello.esc ${DESTDIR}${usr/bin}/

clean:
    rm -rf *.beam erl_crash.dump
```

dh_make を実行します。今回は CDBS 形式にしてみます。

```
$ dh_make -c gpl -b --createorig
Maintainer name : Kouhei Maeda
Email-Address   : mkouhei@palmtb.net
Date            : Fri, 15 May 2009 00:36:24 +0900
Package Name    : hello-erlang
Version        : 1.0
License         : gpl3
Using dpatch    : no
Using quilt     : no
Type of Package : cdb
Hit <enter> to confirm:
Skipping creating ../hello-erlang_1.0.orig.tar.gz because it already exists
Done. Please edit the files in the debian/ subdirectory now. You should also
check that the hello-erlang Makefiles install into $DESTDIR and not in / .
```

debian ディレクトリ以下を編集します。まず、不要なサンプルファイルを削除します。

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

changelog を編集します。

```
$ dch
hello-erlang (1.0-1) unstable; urgency=low

* Initial releas

-- Kouhei Maeda <mkouhei@palmtb.net> Thu, 14 May 2009 22:56:51 +0900
```

control を編集します。erlang-base パッケージが必要なので追記します。

```
Source: hello-erlang
Section: game
Priority: extra
Maintainer: Kouhei Maeda <mkouhei@palmtb.net>
Build-Depends: cdb, debhelper (>= 7), erlang-dev
Standards-Version: 3.8.1
Homepage: http://www.palmtb.net/

Package: hello-erlang
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, ${erlang-base:Depends}
Description: Hello World for Erlang.
 Hello World Erlang version.
```

copyright を編集します。

```
This package was debianized by:

    Kouhei Maeda <mkouhei@palmtb.net> on Thu, 14 May 2009 22:54:34 +0900

Upstream Author(s):

    Kouhei Maeda <mkouhei@palmtb.net>

Copyright:

    <Copyright (C) 2009 Kouhei Maeda>

License:
(以下略)
```

rules を編集します。dh.make で生成された時は、include 文しかないなので、他を追加します。特に、DEB_FIXPERMS_EXCLUDE で、hello.beam を指定しておかないと、インストール時に実行権限が付与されてしまいます。^{*11}beam ファイルは Erlang VM が読み込むだけですので、実行権は必要ありません。

```
#!/usr/bin/make -f

DEB_MAKE_CHECK_TARGET :=
DEB_FIXPERMS_EXCLUDE := hello.beam

include /usr/share/cdb/1/rules/debhelper.mk
include /usr/share/cdb/1/class/makefile.mk

binary-arch binary-indep: build

DEB_MAKE_INSTALL_TARGET := install DESTDIR=$(DEB_DESTDIR)$(cdb_curpkg)
# Add here any variable or target overrides you need.
```

以上が終わったら、debuild を実行します。

```
$ debuild -us -uc
\end{commandline}

pbuilder を実行します。
\begin{commandline}
$ sudo pbuilder create --distribution sid
$ sudo pbuilder build hello-erlang_1.0-1.dsc
```

問題なければ、最後にインストール、アンインストールできるか確認しておきます。

^{*11} 勉強会当日に分かったことですが、実はインストール先のパスに”bin” という文字列があると実行権限を付与してしまう、CDBS のバグが原因のようです。

```

$ sudo dpkg -i hello-erlang_1.0-1_amd64.deb
未選択パッケージ hello-erlang 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 193995 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。)
(hello-erlang_1.0-1_amd64.deb から) hello-erlang を展開しています...
hello-erlang (1.0-1) を設定しています ...

$ dpkg -L hello-erlang
/.
/usr
/usr/bin
/usr/bin/hello.esc
/usr/share
/usr/share/doc
/usr/share/doc/hello-erlang
/usr/share/doc/hello-erlang/changelog.Debian.gz
/usr/share/doc/hello-erlang/README.Debian
/usr/share/doc/hello-erlang/copyright
/usr/sbin
/usr/lib
/usr/lib/erlang
/usr/lib/erlang/lib
/usr/lib/erlang/lib/hello-1.0
/usr/lib/erlang/lib/hello-1.0/ebin
/usr/lib/erlang/lib/hello-1.0/ebin/hello.beam
$ hello.esc
Hello world on Erlang
$ erl -noshell -pa /usr/lib/erlang/lib/hello-1.0/ebin -s hello start -s init stop
Hello world on Erlang
$ sudo apt-get remove --purge hello-erlang
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
以下のパッケージが自動でインストールされましたが、もう必要とされていません:
(snip)
これらを削除するには 'apt-get autoremove' を利用してください。
以下のパッケージは「削除」されます:
  hello-erlang*
アップグレード: 0 個、新規インストール: 0 個、削除: 1 個、保留: 17 個。
この操作後に 73.7kB のディスク容量が解放されます。
続行しますか [Y/n]?
(データベースを読み込んでいます ... 現在 194002 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。)
hello-erlang を削除しています ...

```

5.6 まとめ

今回、Erlang の入り口を少しだけ眺めてみました。まだまだ分からないことが多いのでさらに勉強していきます。また、最後に記述したパッケージの作成については、Erlang のコードパスは自動的にロードされないので、パッケージにする = パッケージにされたコードが便利に使える訳ではありません。先月の勉強会で紹介された OCaml では、dh-ocaml というパッケージ作成支援ツールがありますが、Erlang にはありませんでした。Erlang 独自に必要な設定を支援する、dh-erlang のようなツールがあると、Debian でも Erlang はさらに便利になるなと思いました。

5.7 参考

- プログラミング Erlang
ISBN : 978-4-274-06714-3
- Erlang
<http://erlang.org/>
- Erlang - Wikipedia
<http://ja.wikipedia.org/wiki/Erlang>
- Apache CouchDB : The CouchDB Project
<http://couchdb.apache.org/>

6 MC-MPI/GXP 公式パッケージへの道

藤澤 徹



6.1 前回のおさらい

東京大学 近山・田浦研究室で開発されているグリッド用の MPI ライブラリである MC-MPI, および MC-MPI を動作させるのに必要となる, 同じく近山・田浦研究室で開発されている並列分散シェルの GXP の 2 つを Debian パッケージにしてみました.

6.2 今回やること その 1

前回つくったパッケージは割と適当なので, 公式にするには問題が残っています. まずはこれをちょくちょく解決しておきたいと思います.

といっても, とりあえず解決できたのは 1 つだけです.

6.3 MC-MPI で FORTRAN 用ライブラリが無効

これは, 元々のコードが g77 用に使われていたところを前回出した gFortran でコンパイルしようとしたため, 両者の互換性の無い部分をモロの踏んでしまった事が原因でした.

6.3.1 gfortran_getarg, gfortran_iargc が無い?

```
fortran/.libs/libfortran.a(farg.o): In function 'mpigxp_getarg':  
/home/arai/projects/deb/deb_mcmpi2/mcmpi-0.21.0/src/fortran/farg.f:9:  
undefined reference to '_gfortran_getarg_i4'  
fortran/.libs/libfortran.a(farg.o): In function 'mpigxp_iargc':  
/home/arai/projects/deb/deb_mcmpi2/mcmpi-0.21.0/src/fortran/farg.f:2:  
undefined reference to '_gfortran_iargc'
```

前回の発表では gFortran に getarg, iargc が無い, という風に言ってしまったのですが実際には gFortran に無いわけではなく, gFortran で単体でコンパイルした場合にオブジェクトファイルに含まれない事が原因でした.

というわけでライブラリを作る時に libfortran をリンクする事で解決します.

```
LIBS = @LIBS@ -lgfortran
```

6.3.2 mpigxp_getarg, mpigxp_iargc が無い?

```
fortran/.libs/libfortran.a(initf.o): In function 'mpi_init_':  
/home/arai/projects/deb/deb_mcmpi2/mcmpi-0.21.0/src/fortran/initf.c:16:  
undefined reference to 'mpigxp_iargc_'  
/home/arai/projects/deb/deb_mcmpi2/mcmpi-0.21.0/src/fortran/initf.c:20:  
undefined reference to 'mpigxp_getarg_'
```

これも g77 と gFortran の互換性の問題で、g77 でコンパイルすると元の関数名にアンダースコアが 2 つ追加されるのに対し、gFortran でコンパイルするとアンダースコアが 1 つしか追加されない事が原因です。

というわけでこちらは呼び出し側のコードでアンダースコアの数を調整して解決します。

6.4 今回やること その 2

せっかくパッケージにしたので、これを公式パッケージにして、まっさらな Debian に apt-get でインストールできるようにしてみましょう。

6.5 ところで本当にまだ公式パッケージになってない?

もし他の人の手によって公式パッケージになってたり、途中まで進んでたりしたらそれを知らないまま作業を進めるのは無駄です。

以下のページで作業中のパッケージ等の一覧を見る事ができます。

Work-Needing and Prospective Packages; WNPP

<http://www.debian.org/devel/wnpp/>

とりあえずひととおり検索して、今から作るパッケージがここに無い事を確認しましょう。

6.6 作業開始を宣言するぞ

さきほどのページの下の方に新規項目追加のどうのこうのという節があります。

作業中のものが無いかどうかをこのページで見て確認したわけですから、いざ作業を開始するならばこのページで他の人が見られるようにしなければなりません。

6.7 作業中のパッケージの管理体制

Debian では、作業中のパッケージは Debian バグ追跡システムの上で WNPP という仮想パッケージのバグとして登録され、パッケージ化が完了した時点でこのバグが閉じられます。

つまり、さきほどみたページはこのシステムに登録されている WNPP のバグのリストを表示していた事になります。

6.8 パッケージの登録方法

WNPP にパッケージを登録するには 2 種類の方法が提供されています。

- reportbug を使う
- 自分でメールを出す

6.9 reportbug

reportbug はバグレポートの作成を支援するツールです。

一般的な環境には無いのでインストールしましょう。

```
$ sudo apt-get install reportbug
```

6.10 初期設定

とりあえず起動してみましょう

```
$ reportbug

Welcome to reportbug! Since it looks like this is the first time you have used
reportbug, we are configuring its behavior. These settings will be saved to the
file "/home/arai/.reportbugrc", which you will be free to edit further.
Please choose the default operating mode for reportbug.

1 novice    Offer simple prompts, bypassing technical questions.

2 standard  Offer more extensive prompts, including asking about things that a
            moderately sophisticated user would be expected to know about
            Debian.

3 advanced  Like standard, but assumes you know a bit more about Debian,
            including "incoming".

4 expert    Bypass most handholding measures and preliminary triage routines.
            This mode should not be used by people unfamiliar with Debian's
            policies and operating procedures.

Select mode: [novice]
```

起動すると初期設定の方法を聞かれます。正直に novice と答えましょう。

```
> novice

Please choose the default interface for reportbug.

1 gtk2      A graphical (GTK+) user interface

2 text      A text-oriented console user interface

Select interface:
```

使用するインターフェースは当然 text ですな。

```
> 2

Will reportbug often have direct Internet access? (You should answer yes to
this question unless you know what you are doing and plan to check whether
duplicate reports have been filed via some other channel.) [Y|n|q|?]?

> Y
```

回線が無いと生きていけません。

```
What real name should be used for sending bug reports?
[arai]>

> Tooru Fujisawa

Which of your email addresses should be used when sending bug reports? (Note
that this address will be visible in the bug tracking system, so you may want
to use a webmail address or another address with good spam filtering
capabilities.)
[arai@halko]>

> arai_a@mac.com
```

適当に個人情報を入力します。

```
Do you have a "mail transport agent" (MTA) like Exim, Postfix or SSMTP
configured on this computer to send mail to the Internet? [Y|n|q|?]?
```

このマシンはメールサーバじゃないので n を選択します。

```
> n
Traceback (most recent call last):
  File "/usr/bin/reportbug", line 1841, in <module>
    main()
  File "/usr/bin/reportbug", line 861, in main
    return iface.user_interface()
  File "/usr/bin/reportbug", line 965, in user_interface
    offer_configuration(self.options)
  File "/usr/bin/reportbug", line 496, in offer_configuration
    options=opts, empty_ok=True, force_prompt=True)
TypeError: get_string() got an unexpected keyword argument 'empty_ok'
```

落ちます。

6.11 何が起きたのだろう

...と思ったらアップデートがありました。

```
apt-get install python-reportbug
```

reportbug が 4.2 なのに python-reportbug が 4.0 というよく分からない状態になっていたのが原因のようです。

6.12 初期設定ふたたび

もういちどやりなおしましょう。

```
$ reportbug
...
Do you have a "mail transport agent" (MTA) like Exim, Postfix or SSMTP
configured on this computer to send mail to the Internet? [Y|N|q|?]?
> n
Please enter the name of your SMTP host. Usually it's called something like
"mail.example.org" or "smtp.example.org". If you need to use a different port
than default, use the <host>:<port> alternative format. Just press ENTER if you
don't have one or don't know.
```

MTA の設定は無事通過し、SMTP サーバを聞かれます。reportbug は自動でメールを送信してくれるのですが、今回はこれを使わずにメール送信は自分でやる事にしましょう。

```
> [RET]
Default preferences file written. To reconfigure, re-run reportbug with the "--
configure" option.
```

6.13 パッケージを登録する

さて、初期設定ができたのでパッケージを登録する作業に移ります。

```
Please enter the name of the package in which you have found a problem, or type
'other' to report a more general problem.
```

バグの対象のパッケージを聞かれるので、さきほどの話にあったとおり wnpp を指定します。

```
> wnpp
Are you sure you want to file a WNPP report? [y|N|q|?]?
```

確認されるので Y と入力します。

```

> Y

*** Welcome to reportbug. Use ? for help at prompts. ***
Detected character set: UTF-8
Please change your locale if this is incorrect.

Using 'Tooru Fujisawa <arai_a@mac.com>' as your from address.
Getting status for wnpp...
Will send report to Debian (per lsb_release).
What sort of request is this? (If none of these things mean anything to you, or
you are trying to report a bug in an existing package, please press Enter to
exit reportbug.)

1 ITP This is an 'Intent To Package'. Please submit a package description
along with copyright and URL in such a report.
2 O The package has been 'Orphaned'. It needs a new maintainer as soon as
possible.
3 RFA This is a 'Request for Adoption'. Due to lack of time, resources,
interest or something similar, the current maintainer is asking for
someone else to maintain this package. They will maintain it in the
meantime, but perhaps not in the best possible way. In short: the
package needs a new maintainer.
4 RFH This is a 'Request For Help'. The current maintainer wants to continue
to maintain this package, but they needs some help to do this, because
their time is limited or the package is quite big and needs several
maintainers.
5 RFP This is a 'Request For Package'. You have found an interesting piece of
software and would like someone else to maintain it for Debian. Please
submit a package description along with copyright and URL in such a
report.

Choose the request type:

```

まずバグの種類を聞かれます。wnpp に登録するバグには上記のように 5 種類あります。簡単に説明すると以下のとおりです。

ITP これからあるソフトウェアのパッケージを作成するぞ、という宣言
O 自分のパッケージを手放すという宣言
RFA 自分のパッケージの里親になってくれという要求
RFH 誰か手伝ってくれという要求
RFP 誰かパッケージにしてくれという要求

というわけで ITP を選びます

```

> 1
Please enter the proposed package name:

```

パッケージ名を要求されますまずは mcmpi のパッケージを作る事にしましょう。

```

> mcmpi
Checking status database...
Please briefly describe this package; this should be an appropriate short
description for the eventual package:

```

説明文はパッケージを作った時に決めたのでそのままです。

```

> Grid-enabled implementation of MPI

Your report will be carbon-copied to debian-devel, per Debian policy.
Querying Debian BTS for reports on wnpp (source)...
3082 bug reports found:

Outstanding bugs -- Important bugs; Unclassified (1 bug)
  1) #502809 0: potracegui -- KDE frontend for potrace, severity it normal
(1-1/3082) Is the bug you found listed above [y|N|m|r|q|s|f|?]?

```

登録済みの 3082 個のバグに重複が無いかどうかチェックしろという大変素晴らしい要請を受けました。回避する策を探しましょう。

```
> ?
y - Problem already reported; optionally add extra information.
N - (default) Problem not listed above; possibly check more (skip to Next
page).
m - Get more information about a bug (you can also enter a number without
selecting "m" first).
r - Redisplay the last bugs shown.
q - I'm bored; quit please.
s - Skip remaining problems; file a new report immediately.
f - Filter bug list using a pattern.
? - Display this help.
(1-1/3082) Is the bug you found listed above [y|N|m|r|q|s|f|?]?
```

ほらあるじゃないですか。

```
> s
Spawning emacs...
```

ほら回避できた。まあさきほどウェブ上でもチェックしたし、とりあえずは良しとしましょう。

さて、Emacs が起動されてバグレポートの雛形が表示されました。バージョン、上流作者、URL、ライセンス、使用言語、長い説明を書きましょう。どれもパッケージを作る時にもう書いたものなのでコピペで OK です。

```
Subject: ITP: mcmpi -- Grid-enabled implementation of MPI
Package: wnpp
Owner: Tooru Fujisawa <arai_a@mac.com>
Severity: wishlist

*** Please type your report below this line ***

* Package name      : mcmpi
  Version           : 0.21.0
  Upstream Author   : Hideo Saito <h_saito@logos.ic.i.u-tokyo.ac.jp>
* URL               : http://www.logos.ic.i.u-tokyo.ac.jp/~h_saito/mcmpi/
* License           : GPL
  Programming Lang: C, FORTRAN
  Description       : Grid-enabled implementation of MPI

MC-MPI is a Grid-enabled implementation of MPI, developed by Hideo
Saito at the University of Tokyo. Its main features include the
following:
- [Firewall and NAT traversal]: MC-MPI constructs an overlay
network, allowing nodes behind firewalls and nodes without global
IP addresses to participate in computations. There is no need to
perform manual configuration; MC-MPI automatically probes
connectivity, selects which connections to establish, and performs
routing.
- [Locality-aware connection management]: Establishing too many
connections, especially wide-area connections, results in many
problems, including but not limited to the following: exhaustion of
system resources (e.g., file descriptors, memory), high message
reception overhead, and congestion between clusters during
all-to-all communication. Therefore, MC-MPI limits the number of
connections that are established. If we assume, for simplicity,
that n processes are distributed equally among c clusters, then at
most O(log n) connections are established by each process and at
most O(n log c) connections are established between clusters. As
MC-MPI uses a lazy connect strategy, fewer connections are
established for applications in which few process pairs
communicate. The maximum number of connections allowed can be
controlled by passing the -beta option to mpirun (see Subsection 3).
- [Locality-aware rank assignment]: Temporarily disabled in this
version.
```

こんな感じで保存しましょう。

```
Report will be sent to "Debian Bug Tracking System" <submit@bugs.debian.org>
Submit this report on wnpp (e to edit) [Y|n|a|c|e|i|l|m|p|q|?]?
```

メールサーバは持っていないので Y は選べません。

```
> ?
Y - (default) Submit the bug report via email.
n - Don't submit the bug report; instead, save it in a temporary file (exits
  reportbug).
a - Attach a file.
c - Change editor and re-edit.
e - Re-edit the bug report.
i - Include a text file.
l - Pipe the message through the pager.
m - Choose a mailer to edit the report.
p - Print message to stdout.
q - Save it in a temporary file and quit.
? - Display this help.
Submit this report on wnpp (e to edit) [Y|n|a|c|e|i|l|m|p|q|?]?
```

ファイルに保存してくれれば自分で送れるので, q を選びましょう.

```
> q
reportbug stopped; your incomplete report is stored as "/tmp/reportbug-
wnpp-20090516-28182-HoQ43f". This file may be located in a temporary directory;
if so, it might disappear without any further notice. To recover this file to
use it as bug report body, please take a look at the "-i FILE, --include=FILE"
option.
```

というわけでさっきのバグレポートを submit@bugs.debian.org に送って完了です.

reportbug を使わなくても自分で全部書くという手もあります.

メールを送るとバグに ID が振られ, そのスレッドのトップとしてさっきのメールが返ってきます.

しばらくするとウェブ上の WNPP からも参照可能になります.

さて, 表明もしてしまったところで, これからパッケージの手直しを頑張ろうと決意して今回はおしまいです.



7 Debian GNU/Linux 上で Android SDK を使ってみた

上川純一

7.1 はじめに

Android は Open Handset Alliance の提供する携帯プラットフォームです。携帯電話ではめずらしくオープンソースの集大成であるということで、Debian 開発者としては試してみずにはいられません。Debian だからどうということはありませんが、とりあえず、SDK 1.5r1 がリリースされたのを機にそろそろ試してみることにします。

7.2 利用機材

- Android 開発用の携帯電話, Android 1.5 を実行
- Debian GNU/Linux lenny + squeeze on MacBook

7.3 ダウンロード・インストール

Debian 上で開発するために、各種必要なパッケージなどをインストールします。

7.3.1 必要な Debian パッケージのインストール

Java や 32bit エミュレーション環境のパッケージなどをまずインストールしておきます。

```
# apt-get install ia32-libs sun-java6-bin sun-java6-jdk
```

x86_64 の 64bit 環境で 32bit のアプリケーションを利用できるように、ia32-libs をインストールします。また、java もインストールしておきます。

7.3.2 SDK ダウンロード

Android の SDK をダウンロードしてきます。Debian 用は Linux を選択します。i386 版しかないので、注意。amd64 でも i386 エミュレーション上で稼働するので利用できます。

http://developer.android.com/sdk/1.5_r1/index.html

```

$ unzip android-sdk-linux_x86-1.5_r1.zip
$ ls -l android-sdk-linux_x86-1.5_r1
合計 24
-rw-rw---- 1 dancer dancer 193 2009-04-22 13:15 RELEASE_NOTES.html
drwxrwx--- 3 dancer dancer 4096 2009-04-23 10:50 add-ons
drwxrwx--- 12 dancer dancer 4096 2009-05-01 16:13 docs
-rw-rw---- 1 dancer dancer 176 2009-04-22 13:15 documentation.html
drwxrwx--- 4 dancer dancer 4096 2009-04-23 10:50 platforms
drwxrwx--- 3 dancer dancer 4096 2009-04-22 13:14 tools

```

tools ディレクトリにパスを通しておきます。 .bashrc 辺りに下記を記述しておきます。

```
export PATH="${PATH}":path/to/android/android-sdk-linux_x86-1.5_r1/tools/
```

7.3.3 eclipse ダウンロード

Debian パッケージになっているバージョンは古いため、eclipse のサイトからダウンロードしてきます。

<http://www.eclipse.org/> でバージョン Ganymede の x86-64 アーキテクチャ向けのバイナリをダウンロードしてきました。

展開すればそこで ./eclipse を実行すればエディタが起動します。

```

$ tar xzf eclipse-java-ganymede-SR1-linux-gtk-x86_64.tar.gz
$ ls eclipse
about.html      configuration  eclipse.ini    libcairo-swt.so  plugins
about_files     dropins       epl-v10.html  notice.html      readme
artifacts.xml  eclipse       features      p2
$ cd eclipse/
$ ./eclipse

```

7.3.4 Eclipse に Android 用プラグインの追加

Eclipse のプラグインをインストールします。 http://developer.android.com/sdk/1.5_r1/installing.html を参考に設定します。うまく動かない部分はメーリングリスト検索を駆使し、正しい方法を探します。

プラグインは <http://dl-ssl.google.com/android/eclipse/site.xml> から Android DDMS と Android Development Tools をインストールすればよいようです。

あと、設定する内容としては、Window の Properties の Android でさきほど展開した SDK のパスを指定します。

7.3.5 Android デバイス接続

udev の設定の修正が必要です。 /etc/udev/rules.d/z60_android.rules に下記記述します。

```
SUBSYSTEM=="usb_device",
SYSFS{idVendor}=="0bb4",MODE="0666",
SYMLINK+="%k"
```

接続できていれば、adb コマンドで接続を確認できます。

```

$ adb devices
List of devices attached
HT----- device

```

adb shell コマンドでシェルが起動します。

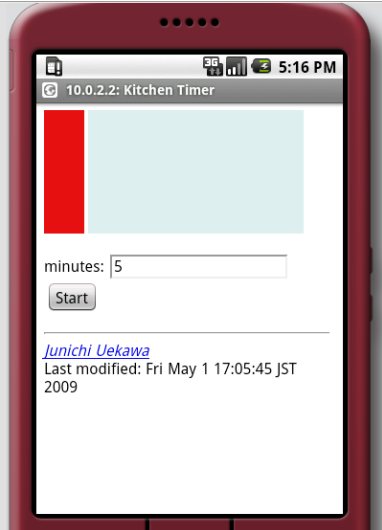
```
$ adb shell
```

7.4 emulator を試してみる

まず、コマンドラインから emulator を起動するのを試してみました。 emulator の起動には AVD (Android Virtual Device) の作成がまず必要なようです。 とりあえず適当に作成してみました。

```
$ android list targets
[選択可能なターゲットの一覧]
$ android create avd -n uekawadroid -t 2
Android 1.5 is a basic Android platform.
Do you wish to create a custom hardware profile [no]
Created AVD 'uekawadroid' based on Android 1.5
$ emulator -avd uekawadroid
```

エミュレータは Android の実機をあつかうのと同じような感覚で操作できます。Android のブラウザを起動したりできます。エミュレータ内部から見ると、10.0.2.2 がホスト OS のアドレスになります。HTML+javascript で作成したキッチンタイマーを動作させて見たところ、動作しました。



7.5 Android の簡単なアプリケーションを作成して起動してみる

それでは、Android 用の簡単なアプリケーションを書いてみましょう。

<http://developer.android.com/guide/developing/eclipse-adt.html> にしたがって淡々と作成します。まず、File の New Project で Android のをプロジェクト選択して作成しました。ターゲットビルドは 1.1 にとりあえずしると書いてありますが、よくわからないので、手元のファームウェアのバージョン 1.5 にあわせておきました。

Run の Run(Ctrl-F11) で Android Application を選択するとエミュレータを起動してアプリケーションを実行することができます。最初に生成されたソースコードのままの状態だと Hello World アプリケーションが起動します。

7.6 Android の簡単なアプリケーションを書いてみる

アプリケーションの作成やエディタの起動の部分はなんとかなるようになったので、次はチュートリアルを眺めてみます。Hello World あたりがよさげです。

<http://developer.android.com/guide/tutorials/hello-world.html>

一通り読んで、とりあえずサンプルを眺めてどうやれば文字列を表示できるのかを確認したので、好きな文字列を表示するアプリケーションを作ってみます。

Android の top コマンドでプロセスの一覧を出力できるので、コマンドを実行してその出力を表示するだけのアプリケーションを作成してみました。中核をなす `src/jp.gr.netfort.dancer/TopView.java` は以下になりました。

```

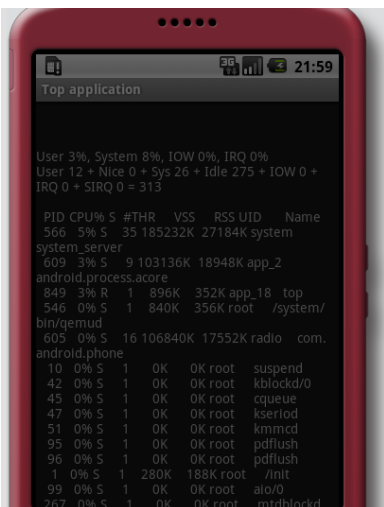
package jp.gr.netfort.dancer;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import java.io.*;

public class TopView extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String [] command = { "top", "-n", "1"};
        String output = "";

        Runtime runtime = Runtime.getRuntime();
        Process process = null;
        try {
            process = runtime.exec(command);
        } catch (Exception exception){
            System.exit(1);
        }
        BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String line;
        try {
            while((line = reader.readLine()) != null) {
                output = output + line + "\n";
            }
        } catch (Exception exception) {
            System.exit(1);
        } finally {
            try {
                reader.close();
            } catch (Exception exception) {
                System.exit(1);
            }
        }
        TextView tv = new TextView(this);
        tv.setText(output);
        setContentView(tv);
    }
}

```



7.7 Android の簡単なアプリケーションを実機で稼働させてみる

実際に使う便利なアプリケーションを作ってみましょう。ライトニングトークを実施する際にどのプレゼンテーションが気に入ったかを表明する、投票用に利用するボタンでも作成しましょうか。

7.7.1 エフェクト用のデータ

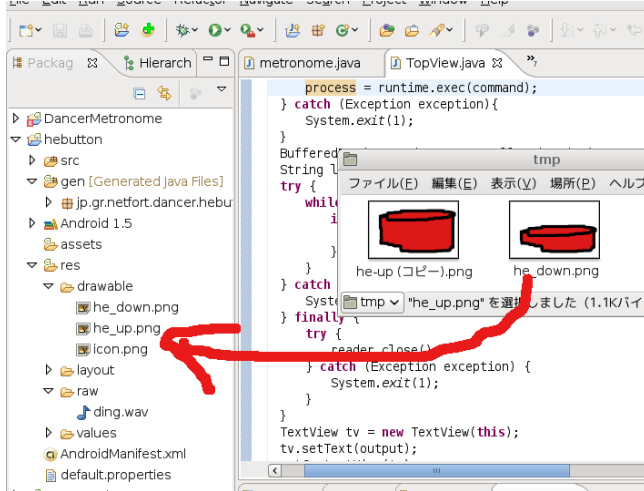
まず、なんらかの効果音が必要です。以前作成した ding.wav をひろってきます。res/raw 以下に nautilus からドラッグアンドドロップでコピーします。

あと、ボタン風味の画像が必要です。適当に gimp で絵をかいて png ファイルを作成しました。

nautilus から res/drawable 以下に png ファイルをドラッグアンドドロップでコピーします。

ファイル名からそのままリソースの名前を作成するため、-は利用できないようです、_ は利用できるようです。最

初 he-up.png を作成しようとして、エラーが出たので he_up.png に変更しました。



7.7.2 レイアウトの作成

おもむろに ImageButton を作成して、src に画像リソースを追加しておきましょう。

7.7.3 クリックアクションの作成

R.id.ImageButton01 に対して onClickListener を作成し、クリックされた瞬間の反応を作成しましょう。

画像をクリックされた状態に切り替えるには ImageButton の setImageResource() を呼び出せばよいようです。

音を出すには、MediaPlayer のインスタンスを作成し、start() を呼び出せばよいようです。

一定時間経過してから元の画像に戻すには、アラームを設定してコールバックしてもらえるようにして、コールバックを呼び出されたときに処理をするようにすればよいようです。

7.7.4 サーバ側のアクション

それでは、クリックされたときにどういふことを行うのかのアクションを作成してみましょう。HTTP サーバを準備します。こういうイロモノ系のサーバを作成するときには upaccho2 サーバを使うことが上川家の常識なので、それを踏襲します。(予定)

7.8 おまけ:開発環境に必要なメモリ消費は節減できるのか?

手元の MacBook には 1GB しかメモリを搭載していません。eclipse はデフォルトの設定だと起動した瞬間に 1GB 近くの仮想メモリを消費し、即 Swap を使用してしまうという問題がありました。

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17124 dancer 20 0 972m 302m 20m S 1 31.2 0:39.85 java
```

とりあえず、ヒントを求めて maps あたりを眺めてみます。それなりにたくさん mmap していますが、それだけでは説明できないようなメモリ領域を確保しています。

```
$ cat /proc/17124/status | grep Vm
VmPeak:      1056748 kB
VmSize:      1003316 kB
VmLck:        0 kB
VmHWM:       310740 kB
VmRSS:       221616 kB
VmData:      788504 kB
VmStk:        84 kB
VmExe:        36 kB
VmLib:       38812 kB
VmPTE:       1108 kB
$ cat /proc/17124/maps | grep rw | sort -rn -k5
```

とりあえず Java のアプリケーションの Heap の使い方がよくわからないので、jconsole で接続して問題を解析し

ます。

メモリプールがいくつかあることがわかります。

- Tenured Gen が 60MB
- Survivor 1MB
- Code Cache が 4MB
- Perm Gen が 70-90MB
- Heap は 100MB(GC したら 50MB 程度)

eclipse.ini が現状こうなっているのですが、これをおそろおそろチューニングしてみます。

```
-showsplash
org.eclipse.platform
-framework
plugins/org.eclipse.osgi_3.4.2.R34x_v20080826-1230.jar
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms40m
-Xmx256m
-XX:MaxPermSize=256m
```

起動時に確保したヒープメモリの領域を開放していない部分については、GC を頻繁に行えば起動は若干遅くなりますがうまくいけるような気がします。Xmx 128m に変更してみると、若干メモリの消費が下がりました。しかしそこまで劇的ではありませんね。MaxPermSize を 64MB にするとエラー

```
Exception in thread "RMI TCP Connection(idle)" java.lang.OutOfMemoryError: PermGen space
```

が発生しました。96MB だとエラーでおちないようです。

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	コメント
17124	dancer	20	0	972m	302m	20m	S	1	31.2	0:39.85	java
19663	dancer	20	0	795m	296m	23m	S	1	30.6	0:36.69	Heap 128MB 版
20266	dancer	20	0	661m	292m	22m	S	1	30.1	0:41.62	MaxPermSize 96MB

がんばってみた割にはメモリが 10MB 程度しか節約できませんでした。エミュレータも起動していると 500MB くらい swap にいってしまうので、つらいです。しかも、ギリギリのメモリ設定にしていたところ、いろいろと操作しているとメモリが足りないというエラーで落ちました。こりゃダメだ。メモリ買いにいくかな。

```
-showsplash
org.eclipse.platform
-framework
plugins/org.eclipse.osgi_3.4.2.R34x_v20080826-1230.jar
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms40m
-Xmx128m
-XX:MaxPermSize=96m
```

7.9 参考文献

本記事の作成に参考にした文献です。

- <http://www.eclipse.org/>: eclipse
- <http://developer.android.com/>: Android のページ、開発情報がまとまっている。特に http://developer.android.com/sdk/1.5_r1/index.html: Android SDK 1.5 のダウンロードページからたどるページが重要。
- <http://developer.android.com/guide/basics/what-is-android.html>: SDK の開発マニュアル。SDK の `./android-sdk-linux_x86-1.5_r1/docs/` に同じものがあるのでオフラインでも安心。ただしオンラインのドキュメントは微妙な訂正がアップデートされ続けているようなのでネットワーク接続が利用できるオンラインの場合はそちらを参照するほうがよいかもしれません。

8 DDTSS 活用

上川 純一



8.1 はじめに

最近 apt の国際化も進み、Debian archive の Description の翻訳もパッケージの配布の仕組みと一緒に配布されるようになってきました。一方、翻訳を用意するプロジェクト (DDTP:Debian Description Translation Project) はまだ普及していません。今回はその活用を事前課題として、今後どうやってうまく活用していくのかを考えます。

8.2 Debian のパッケージ説明文の翻訳の目的とは

Debian Distribution には数万のパッケージがあります。パッケージの説明文はインストールするソフトウェアを探すのに寄与する内容である必要があります。

現在、その説明文はすべて英語で提供されています。しかし、英語で記述されているとそもそも英語を理解していないとわからない部分があり、しかも技術的な専門用語をまじえた説明文章は日本語を主言語とするユーザにとって理解しにくい場合があります。

そもそもの Description が翻訳に適するレベルに至っていない、日本語に訳しにくい文章であれば、もとの文書を訂正するように依頼しましょう。

また、翻訳した Description が日本語として理解しにくい、技術的な表現が不適切な表現であれば、その Description は適切ではないでしょう。

また、Debian 全体で表記をできるだけ統一しましょう。同じ内容を説明するのに表記のゆれがあるとパッケージの説明文をみているときにその差異が目がいてしまいます。

その他に考慮する点は?

8.3 DDTP

DDTP の説明は Debian.org のページにあります: <http://www.debian.org/intl/l10n/ddtp.ja.html>。 <http://www.debian.or.jp/community/translate/description-ja.html> などとあわせて眺めてください。

メールフロントエンドや、ウェブフロントエンドなどがあります。

8.4 DDTSS とは

DDTP のフロントエンドの一つが DDTSS です。 <http://ddtp.debian.net/ddtss/index.cgi/xx> に提供されています。

8.5 DDTSS 活用の手順

8.5.1 アカウントの登録

ログイン画面^{*12}に飛ぶと、アカウントを作成するためのリンク (Create Login^{*13}) があるので、そこからアカウント登録します。

メールが届くので、そこからリンクをクリックしてアカウントをアクティベートすると、ログイン画面からログインできるようになります。

8.5.2 パッケージ説明文の翻訳作成

<http://ddtp.debian.net/ddtss/index.cgi/ja> 画面で、Pending Translation の欄にあるパッケージを適切にクリックすると、パッケージ説明文の編集画面が登場します。一部の文書が翻訳されておらず<trans>となっていたり、まったく翻訳されていなかったり、状態はまちまちです。

この画面で翻訳を作成します。

8.5.3 レビューの実施

<http://ddtp.debian.net/ddtss/index.cgi/ja> 画面で、Pending Review の欄にあるパッケージを適切にクリックすると、パッケージ説明文のレビュー画面が登場します。

レビューして変な部分があれば訂正しましょう。

8.5.4 debian-doc メーリングリストでのレビュー実施

debian-doc というメーリングリストで議論とかレビューはおこなうみたいです。

8.6 参考文献

以下のページも参考にしてください。

- Debian パッケージの説明文を日本語で読みたい! ~DDTP へのお誘い~<http://www.debian.or.jp/community/translate/description-ja.html>
- 武藤健志さんの blog の『Debian ドキュメント翻訳手続き』: <http://kmuto.jp/d/index.cgi/debian/debian-doc-procedure.htm>
- 小林儀匡さんの Debian 勉強会 2006 年 9 月資料「翻訳への誘い」: tokyodebian.alioth.debian.org/pdf/debianmeetingresume200609.pdf
- debian-doc メーリングリスト: 主要な議論が行われています。質問なども、こちらで。

^{*12} <https://ddtp.debian.net/ddtss/index.cgi/login>

^{*13} <https://ddtp.debian.net/ddtss/index.cgi/createlogin>



Debian 勉強会資料

2009年5月16日 初版第1刷発行

東京エリア Debian 勉強会（編集・印刷・発行）
