第 76 回 東京エリア Debian 勉強会資料

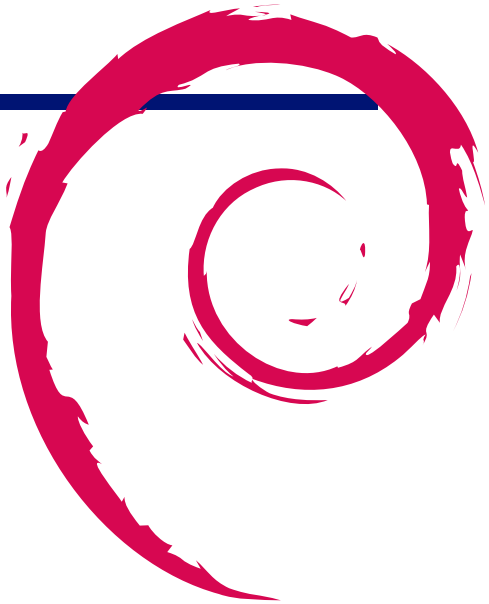
月刊 Deb専

日本唯一のDebian専門月刊誌

2011 年 5 月 21 日

特集1: Apache2 モジュールを作ってみた

特集2: Debian/m68k 開発



1 Introduction

上川 純一

今月の Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか?

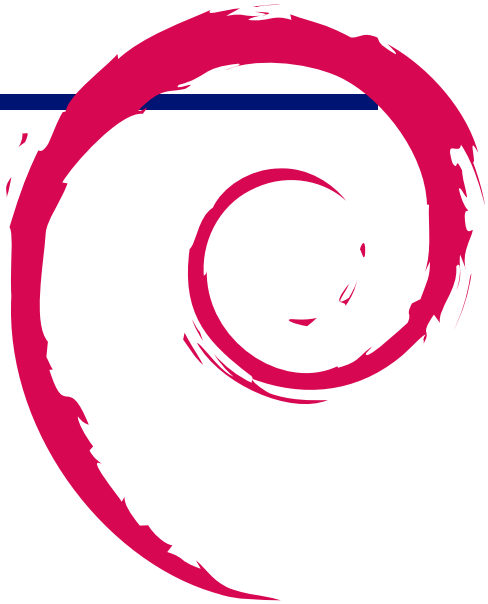
Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での「開発に関する情報」を整理してまとめ、アップデートする。
- 場の提供。
  - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
  - Debian のためになることを語る場を提供する。
  - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、「場」としての空間を提供するのが目的です。

デビアン勉強会

目次



2 事前課題

岩松 信洋

今回の事前課題は以下です:

1. Debian 使いとしてウェブサービスに期待することは何ですか?

この課題に対して提出いただいた内容は以下です。

2.1 キタハラ

Debian 限定だと思いつかない・・・。(お題の意図を読み違えているのかも) apt-get を http で実行するとウェブサービスと言える?

2.2 MATOHARA

Debian 使いとしてウェブサービスに期待すること．最近は少なくなりましたが， IE 必須のサービス等の環境依存のサービスをやめて欲しいです．最近だと Silverlight 必須のサービスで Moonlight で動きそうで動かないといったことがありました．
`http://live6.channel.ne.jp/world_ipv6/`

2.3 taitioooo

情報に対する課金がなくなること。

2.4 野島 貴英

- jslinux という強力なエミュレータも出たので、ブラウザで動く Debian experimental 環境とかブラウザで動く Gnome のお試し環境とかを提供するウェブサービスとか素敵かも。こもきっとウェブサービス!（ なんか空気読めてない回答な気もするけど...)
- USB に書き込めば debian 環境がそのままブートできるようなイメージをつくってくれるウェブサービスが良さそうな気も... 例えば、パッケージ一覧にチェック入れて、sid とかにチェック入れると、 USB メモリにそのまま書き込めばその仕様で debian sid がブートできるようなカスタムイメージを作ってくれるとか。
- チェックボックスとセレクタだけで、 preceed ファイル生成してくれるウェブサービスもいいかも... 大量のインストール時とかよさそう。(もう言いたい放題ですね...)

2.5 岩松 信洋

- 全世界の Web サーバを提供する OS が Debian になること。
- 分散コンパイルサーバとか欲しい。

2.6 日比野 啓

Web サービスもできれば機械処理しやすいものが良い。あと、クラウド上での API を提供しているようなサービスに、関数型言語に対するサポートが増えてほしい。

2.7 dictoss(杉本 典充)

CPU とある deb パッケージを選択すると、その CPU 向けに最大限の最適化したパッケージと依存するパッケージを再ビルドしてくれるサービス。

2.8 kazken3

翻訳をたまにしているので、ディストリビューション間横どおしでの翻訳関連情報を提供するサイトがあればいいなと思うことがあります。
＃課題とは少しズレているかも知れませんが、＃個人向けのウェブサービスには食傷気味というところもあるので。

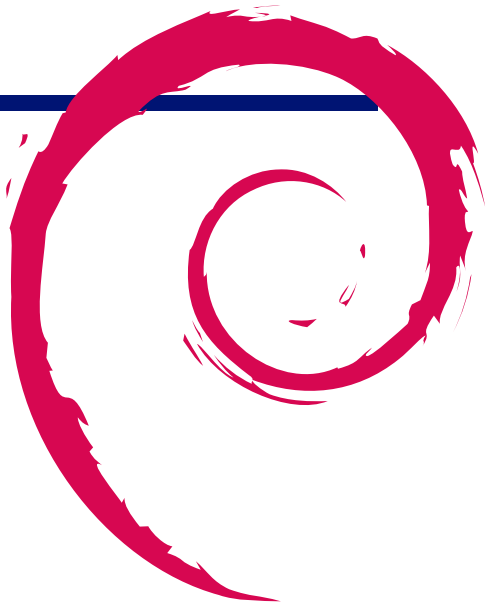
2.9 まえだこうへい

Debian システムで作った環境との相互互換性。例えば、最近 GAE/Python をよく使うので、作ったシステムを

GAE/Python ¡-¿ → Debian システムのどちらでも (ほとんど変更なしで) 動かせると便利ですね。すぐ始めるのにクラウドサービスを利用して作ったけど将来は Debian で動かしたい、逆に今は政治的な理由で外に出せない Debian システムを将来は自分の管理から外れるので手離れをよくするためにクラウドサービスに簡単に移行できる、など。

2.10 yamamoto

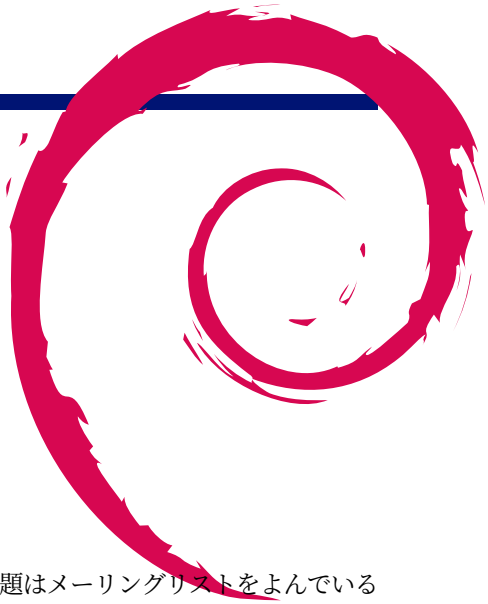
そうですね。今の所導入を検討しているのは、パーソナルストレージサービスぐらいですかね。あらゆる所で自分のデータが自分で共有できれば、それで十分な感じです。



3 最近の Debian 関連のミーティング報告

岩松 信洋

3.1 東京エリア Debian 勉強会 75 回目報告



4 Debian Trivia Quiz

岩松 信洋

ところで、みなさん Debian 関連の話題においついていますか? Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

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

5 Apache2 のモジュールをつくってみた

上川純一

Apache httpd というウェブサーバがあります。おそらく定番と言われるウェブサーバで、多くの人が利用していると思います。ウェブサーバとして多くの機能を提供している Apache ですが、モジュールという仕組みで機能拡張可能です。今回は Debian 上で Apache モジュールを作る方法をさぐってみることにします。

5.1 Apache モジュールをつくりたいとき

Apache モジュールが作りたいときはどういうときでしょうか。作りたいと思ったときがつくりどき。 cgi で fork する手間が惜しい、インタプリタ言語でインタプリタが走っている時間がもったいない、 GC のある言語でときどき反応が悪くなるのが気にくわない。いろんな理由があるとおもいますが、 C 言語でがりがりとウェブリクエストを処理したいとかそういう要望があるのではないでしょうか。

Apache モジュールでできることはどういうことでしょうか。 Apache の受け取る入力 (HTTP Request) から出力 (HTTP Response) のほぼ任意の場所にフックとして割り込むことができて、入出力のデータを加工することが可能です。

5.2 Debian の Apache パッケージの構成

ところで、 httpd と Debian apache2 はどれくらい違うかご存知でしょうか。
Debian パッケージの README.Debian を見てみましょう。 [1]
ざっと眺めただけでも以下の点で特徴があります。

- apache2 コマンドが環境変数に依存している。
- apache2ctl / /etc/init.d/apache2 を利用しないと起動とかできない。
- 設定ファイルの配置が全然違う。
- a2enmod / a2dismod コマンドが追加されている。
- apxs コマンドが apxs2 という名前になっている。

5.3 Apache モジュールの作成

普通の Apache モジュールの作成の流れをみてみましょう。

5.3.1 パッケージの準備

とりあえず開発環境をインストールしましょう。

```
# apt-get install apache2-threaded-dev
```

5.3.2 テンプレート生成

apxs2 コマンドを利用してテンプレートを作成します。モジュール名でディレクトリが作成されます。

```
$ apxs2 -g -n dancerqps
$ cd dancerqps
$ ls
$ ls
Makefile  mod_dancerqps.c  modules.mk
```

5.3.3 ソースコード編集

ソースコードを編集しましょう。この場合、自動で生成された mod_dancerqps.c がメインのモジュールソースコードです。

まず一番下に一番重要な構造が定義されています。ここで重要なのは dancerqps_register_hooks を登録しているところでしょうか。

```
module AP_MODULE_DECLARE_DATA dancerqps_module = {
    STANDARD20_MODULE_STUFF,
    NULL,                  /* create per-dir    config structures */
    NULL,                  /* merge  per-dir    config structures */
    NULL,                  /* create per-server config structures */
    NULL,                  /* merge  per-server config structures */
    NULL,                  /* table of config file commands       */
    dancerqps_register_hooks  /* register hooks                      */
};
```

dancerqps_register_hooks で実際にフックの登録を行います。ここでは、アウトプットフィルタを登録しています。

```
static void dancerqps_register_hooks(apr_pool_t *p) {
  ap_register_output_filter(dancerqps_name, dancerqps_filter,
                            NULL, AP_FTYPE_RESOURCE);
}
```

そして、実際に各リクエストのアウトプットに対して実効するフィルタのコードを書きます。

```
static apr_status_t dancerqps_filter(ap_filter_t *f, apr_bucket_brigade *bb) {
  /* ここでなにか処理をする */
  ap_log_rerror(APLOG_MARK, APLOG_ERR, 0, f->r,
                "dancerqps processing was here!");
  return ap_pass_brigade(f->next, bb);
}
```

5.3.4 モジュールのファイルインストール

モジュールをビルドしてインストールする一連の手順をしてくれるコマンドがあります。

```
$ sudo apxs2 -c -i mod_dancerqps.c
```

5.3.5 モジュールを有効にする

apxs2 コマンドでモジュールを有効にする方法は Debian 独自の a2enmod などのコマンドの存在を無視しているようなので a2enmod などと整合性のとれるような方法をとるのがよいでしょう。で、それでは Debian 的な方法とはなんでしょうか。

Debian way 1

めんどくさい方法だけどシステム運用者としては便利かもしれない方法。 a2enmod とかができます。

/etc/apache2/mods-available に hoge.conf, hoge.load ファイルを作成します。 hoge.conf は設定が不要なら不要ですが、 httpd.conf に書くような内容を記述します。 hoge.load には LoadModule 行を記述します。

この方法でファイルを作成しておけば、/etc/apache2/mods-enabled ディレクトリにシンボリックリンクが作成されます。

Debian way 2

デフォルトでは空のファイルだけど、/etc/apache2/httpd.conf に書き込むとその設定が有効になります。

```
LoadModule test_module /usr/lib/apache2/modules/mod_test.so
<Location "/hoge/">
   SetHandler test
</Location>
```

Debian 的な方法から乖離しているっぽい方法

apache をコマンドラインから設定ファイルを指定して起動することができます。 Debian の Apache は環境変数を設定しないと直接は起動できなくなっているので、面倒な手続きが必要になります。

まず、適当にテスト用の http.conf をでっち上げます。

```
Listen 8080

LockFile /home/test/tmp/apache.1.lock
PidFile /home/test/tmp/apache.1.pid

# log configuration.
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "/home/test/log/access_log" common
ErrorLog "/home/test/log/error_log"

# Order, Allow.
LoadModule authz_host_module /usr/lib/apache2/modules/mod_authz_host.so
# map from / -> /index.html
LoadModule dir_module /usr/lib/apache2/modules/mod_dir.so
DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
# .html -> content-type: text/html
LoadModule mime_module /usr/lib/apache2/modules/mod_mime.so
TypesConfig /etc/mime.types

# Document root
DocumentRoot "/home/test/hoge"
<Directory "/home/test/hoge">
    Options Indexes FollowSymLinks

    AllowOverride None

    Order allow,deny
    Allow from all

</Directory>

# Load my custom filter.
LoadModule dancerqps_module /usr/lib/apache2/modules/mod_dancerqps.so
SetOutputFilter DANCERQPS
```

適当な設定ファイルを指定して Apache を起動してみます。

```
APACHE_RUN_USER=dancer  \
 APACHE_RUN_GROUP=dancer \
 /usr/sbin/apache2 -f $(readlink -f ./httpd.conf) -k restart
```

5.3.6 負荷テスト

ab (apachebench) ツールが標準で入っているのでそれで計測します。適当に確認しましょう。

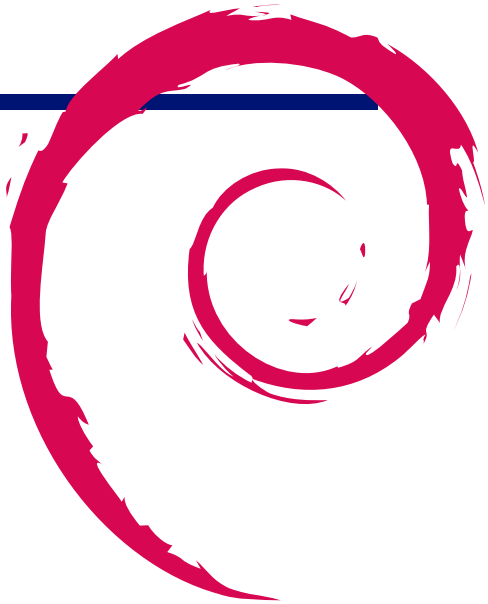
```
$ /usr/sbin/ab -c 100 -n 100 http://localhost:8080/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:        Apache/2.2.9
Server Hostname:        localhost
Server Port:            8080

Document Path:          /
Document Length:        44 bytes

Concurrency Level:      100
Time taken for tests:   0.056 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      29600 bytes
HTML transferred:       4400 bytes
Requests per second:    1796.17 [#/sec] (mean)
Time per request:       55.674 [ms] (mean)
Time per request:       0.557 [ms] (mean, across all concurrent requests)
Transfer rate:          519.21 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        7    9   0.5      9      10
Processing:     9   26   8.9     27      40
Waiting:        6   26   9.3     27      40
Total:         16   36   8.9     37      49

Percentage of the requests served within a certain time (ms)
  50%     37
  66%     41
  75%     43
  80%     45
  90%     47
  95%     48
  98%     49
  99%     49
 100%     49 (longest request)
```

5.3.7 まとめ

はじめて Apache モジュールを作成して、インストールして動かすところまでやってみました。

参考文献

[1] /usr/share/doc/apache2.2-common/README.Debian.gz



6 Debian/m68k 開発

岩松 信洋

Debian/m68k 開発環境を構築する機会があったのでまとめてみました。

6.1 m68k とは?

m68k とはなにか? Motorola 680x0/m68000/68000 の事。省略して m68k。 32bit で CISC。エンディアンはビッグ。世界中で未だに人気にある CPU の一つ。今はフリースケール・セミコンダクタによって、 Coldfire という名前で製造および販売されています。なので今は 68k と言うことが多いです(モトローラじゃないから)。 Apple 社の Macintosh SE やシャープの X68000、 Palm Pilot の CPU として活躍していました。もちろん Linux でもサポートされており、 Debian では hamm から正式にサポートアーキテクチャとして採用されましたが、 etch から脱落しました。 Debian に最初にポーティングされ、最初に脱落したアーキテクチャということで覚えておくと良いでしょう。

表 1 m68k を使った主な機器

メーカ	ハードウェア
ATARI	Atari Falcon
HP	HP 9000 Series 200
SUN	Sun-1
DEC	VAXstation 100
SGI	RIS 1000
SEGA	メガドライブ
SNK	ネオジオ

6.2 Debian/m68k の現状

etch から脱落した後も開発は続けられており、今は debian-ports.org 上で開発しています。 1 年前に開発が停滞しましたが、 Thorsten Glaser 氏[*1]が拾い上げ、数名の開発者と共に楽しく開発を続けています。

ポーティング開始当時は Macintosh や ATARI 社の Amiga 上で開発していましたが、既にこれらのハードウェアは入手が難しくなっているので主にエミュレータを使って開発しています。 Debian の bootstrap が行える程度のパッケージは常に最新に近い状態が維持されているので、 X や GUI を使わない環境程度ならすぐに構築可能です。

ちなみに、 Debian に再度取り込むことは目標にしておらず、 linux/m68k(68k) の開発用として生きる道を選んだようです。もし Debian で クロスコンパイルやエミュレータによる開発が許可されるようになったら、復活するかもしれま

[*1] MirOS の開発者。 OpenWRT の開発者でもある。

せん。

開発議論は ML（ http://lists.debian.org/debian-68k/）と IRC（ debian-68k@oftc）で行われています。

6.3 なぜ m68k に手を出してしまったのか?

先月、 Ruby のコミッタになってしまったので、他になにかか自分でもできることないかなと思ってバグを見ていたら、 Ruby1.9.1 パッケージのバグ #611691（ m68k が FTBFS）を見つけたのが事の始まりです。

6.4 開発環境設定方法

先にも説明したように、実機での開発は行われておらずエミュレータを使って開発が行われています。エミュレータといえば、 ARM や SH などが使える qemu が有名ですが、 qemu の 68k は不具合が多いので、 Debian では ARAnyM [*2] という 68k エミュレータを使って開発しています。ここでは ARAnyM を使った開発環境の構築方法を説明します。

6.4.1 ARAnyM とは

ARAnyM（ Atari Running on Any Machine） は 68040 + MMU + FPU(68882) を実装したエミュレータです。全ての 68k をサポートしているわけではなく、人気のあった CPU、特に Atari のハードウェアがサポートしていた CPU をサポートしています。グラフィックス、ディスクドライブ、 CDROM、ネットワークもサポートしており、特徴として、 OpenGL を使った高速なグラフィックと 4GB のメモリを扱えることがあります。

6.4.2 ホスト側の設定

まず、 ARAnyM をインストールします。 Debian パッケージになっているので apt で簡単にインストールできます。また、後で必要になるパッケージもインストールしておきます。

```
$ sudo apt-get install aranym p7zip
```

Debian m68k の開発に必要なカーネル、ユーザランドイメージのダウンロードします。

カーネルは linux-image-2.6.38-2-atari カーネルパッケージ[*3] を展開した vmlinuz を使います。

```
$ wget http://debian.nctu.edu.tw/debian-ports/pool-m68k/main/l/linux-2.6/linux-image-2.6.38-2-atari_2.6.38-5_m68k.deb
$ ar -x linux-image-2.6.38-2-atari_2.6.38-5_m68k.deb
$ tar -xzf data.tar.gz
$ ls boot/vmlinuz-2.6.38-2-atari
-rw-r--r-- 1 iwamatsu iwamatsu 1767311 2011-05-12 00:48 boot/vmlinuz-2.6.38-2-atari
```

次のユーザランドイメージをダウンロードします。 build-essentail がインストールされたイメージが既にあるので、これを活用します。

```
$ wget http://people.debian.org/~smarenka/aranym/sid/disk.tar.7z
$ 7zr x -so disk.tar.7z | tar xvf -
$ ls -l disk.img
-rw-r--r-- 1 iwamatsu iwamatsu 10737377280 2011-05-18 00:37 disk.img
```

次にネットワークを設定します。以下で説明するネットワークは図 1 のようなネットワーク構成になるようにしています。

今回の ARAnyM 環境では tun を使うので uml-utilities パッケージをインストールします。

```
$ sudo apt-get install uml-utilities
```

そして、 tun および ARAnyM を使うユーザを uml-net に追加します。

```
$ sudo gpasswd -a iwamatsu uml-net
```

[*2] http://aranym.org/

[*3] http://packages.debian.org/search?keywords=linux-image-2.6.38-2-atari

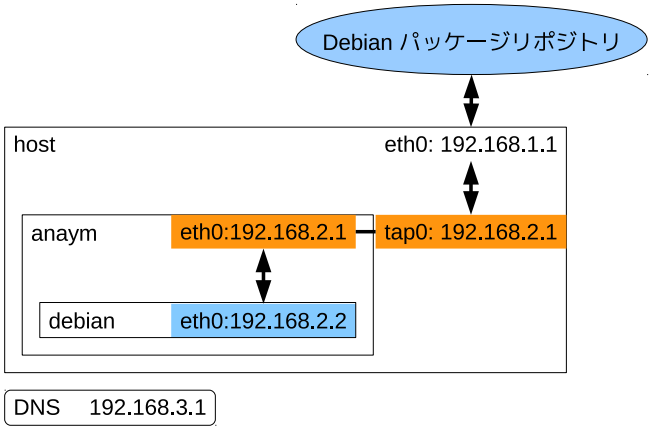


図 1　ARAnyM のネットワーク構成図

ホスト側の ネットワークを以下のように設定します。

```
$ cat /etc/network/interfaces
auto tap0
iface tap0 inet static
address 192.168.2.1
pointopoint 192.168.2.2
netmask 255.255.255.255
tunctl_user iwamatsu
up iptables -t nat -A POSTROUTING -s 192.168.2.2 -j MASQUERADE
down iptables -t nat -D POSTROUTING -s 192.168.2.2 -j MASQUERADE
```

フォ ワーディ ングを有効にして、 tap0 ネットワークデバイスを上げます。

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
$ sudo ifup tap0
```

次に ARAnyM の設定を行います。 ARAnyM は 特に指定しない場合には ~/.aranym/config を読み込みます。

```
$ cat aranym.config
[GLOBAL]
FastRAM = 768 # メモリサイズ。単位は MB。
Floppy =
TOS =
EmuTOS =
AutoGrabMouse = No
GMTime = Yes

[LILO]
# Linux カーネルイメージ
Kernel = vmlinuz-2.6.38-2-atari
# these Args for normal X operation
# カーネルコマンドライン
Args = root=/dev/hda1 console=tty debug=par

# these Args for headless
#Args = root=/dev/hda1 console=nfcon

# ネットワーク設定
[ETH0]
Type = bridge
Tunnel = tap0
# エミュレータで使う仮想ネットワークデバイスの Mac アドレス
Mac = XX:XX:XX:XX:XX:XX

[STARTUP]
GrabMouse = No
Debugger = No

[IDE0]
Present = Yes
IsCDROM = No
ByteSwap = No
ReadOnly = No
# ディスクイメージ
Path = disk.img
Cylinders = 20805
Heads = 16
SectorsPerTrack = 63
ModelName = Master

[VIDEO]
FullScreen = No
BootColorDepth = 8
VidelRefresh = 1
```

以上で設定は終わりなので、 ARAnyM を使って、 Debian OS を立ち上げます。

```
$ aranym-mmu -l -c aranym.config
```

```
$ uname -a
Linux aranym 2.6.38-2-atari #1 Mon May 9 16:39:31 UTC
 2011 m68k GNU/Linux
$ cat /proc/cpuinfo
 CPU:68040
 MMU:68040
 FPU:68040
 Clocking:73.5MHz
 BogoMips:49.04
 Calibration:245248 loops
```

6.4.3 ターゲットでの設定

Debian OS が立ち上がったら、 root ユーザでログイン(パスワードは無し) し、ネットワーク設定を行います。起動時に ARAnyM の仮想ネットワークデバイス(nfeth:nat-feature) を eth0 として認識します。認識されている場合には ARAnyM の設定した MAC アドレスで eth0 が認識されています。

```
# dmesg  | grep eth0
eth0: nfeth addr:192.168.0.1 (192.168.0.2) HWaddr:XX:XX:XX:XX:XX:XX
```

もしホスト側の設定が間違っている場合、 eth0 が存在しない状態になります。このような場合には、ホスト側の設定を見直してください。

eth0 が認識されているのなら、/etc/network/interfaces と /etc/resolv.conf を以下のように変更します。

```
# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.2.2
netmask 255.255.255.0
gateway 192.168.2.1
# cat /etc/resolv.conf
nameserver 192.168.3.1
```

設定が終わったら、各デバイスを上げネットワークがつながることを確認できれば、 apt を使って最新の環境にアップデートし開発環境環境は完成です。

```
# ifup lo
# ifup eth0
# ping 192.168.2.1 # gateway へのチェック
# ping 192.168.3.1 # DNS へのチェック
# apt-get update   # apt-get update
# apt-get install debian-ports-archive-keyring
# apt-get update
# apt-get dist-upgrade
```

6.4.4 その他開発環境

エミュレータを使って開発できるのはすごく良いことなのですが、エミュレータだけでは遅いのでクロスツールチェインが欲しくなります。 Debian でのクロス toolchain は emdebian プロジェクトが提供していますが、 m68k のものは提供されていません。しかし、 amd64 バイナリは Thorsten Glaser 氏が以下の apt-line で提供しています。

```
deb http://www.freewrt.org/~tg/debs68k/ cross main
```

6.5 ARAnyM 上での開発

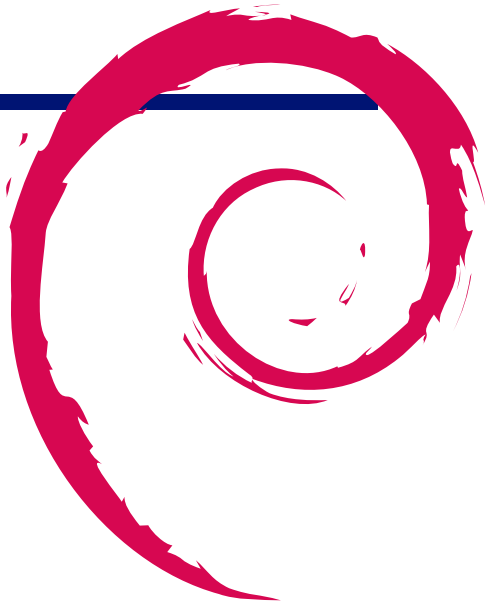
動作しているのが エミュレータ上というだけで通常の開発と変わりません。 cowbuilder も使えるので、遅いという以外には問題はないでしょう。開発速度を上げたい場合には、 distcc/icecc/ccache などを駆使すれば快適な開発ができるようになります。このあたりの話はまた今度。

6.6 Ruby の FTBFS バグはどうなったのか?

Debian/m68k の開発環境は構築できましたが、 Ruby のバグはどうなったのかというと、 `http://redmine.ruby-lang.org/issues/4745` としてバグレポートし、 r31646 でコミットしておきました。

参考文献

[1] `https://wiki.debian.org/M68k`



7 月刊 PPC64 ポーティング

山本浩之

週末ハッカーなおいらですが、今月の進捗状況を報告します。

今月は perl 2.12 投入、 eglibc 2.13 投入、 gcc-4.6 のデフォルト化などと、公式でも FTBFS 続出な月でした。しかしそれにもめげず、 debian-ports への応募の前段階として、 buildd 環境に最低限必要なパッケージ群を公開し始めました。

```
http://yama.fam.cx/debian/dists/unreleased/main/binary-ppc64
```

でも、まだ全ては揃っていません。足りないのは

```
aptitude    build-deps なパッケージが FTBFS でビルドできていない
coreutils   どうやら公式でも FTBFS らしいです。パッチが一ヶ月以上放置されています。
debconf     all なパッケージなんですが、 ppc64 では FTBFS でした。
pam         なんのパッケージが悪さをしているのかはっきりとしていませんが、ある時点から Segment fault するようなパッケージしかビルドできなくなりました。
```

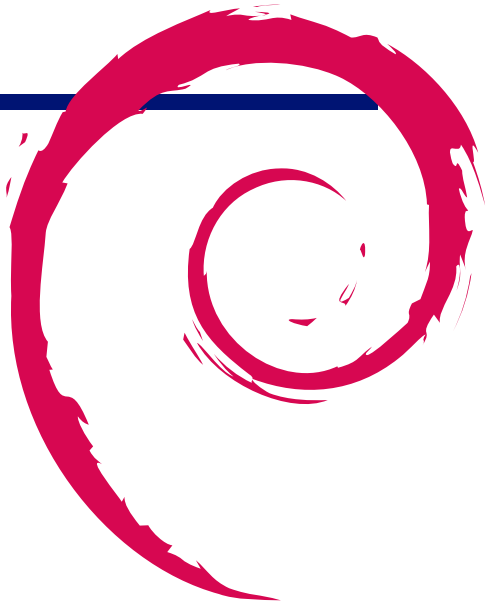
と言った所です。

また、先月の宿題、「 nbench で、 ppc64 port を powerpc port と比較してみよ」ですが、以下の結果になりました。

```
ppc64 port:
-----
BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          762.72  :      19.56  :       6.42
STRING SORT         :          162.35  :      72.54  :      11.23
BITFIELD            :      1.4628e+08  :      25.09  :       5.24
FP EMULATION        :          165.36  :      79.35  :      18.31
FOURIER             :           12004  :      13.65  :       7.67
ASSIGNMENT          :          16.388  :      62.36  :      16.17
IDEA                :            2479  :      37.92  :      11.26
HUFFMAN             :          1095.6  :      30.38  :       9.70
NEURAL NET          :          25.628  :      41.17  :      17.32
LU DECOMPOSITION    :          806.48  :      41.78  :      30.17
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 41.242
FLOATING-POINT INDEX: 28.635
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
============================LINUX DATA BELOW============================
CPU                 : Dual
L2 Cache            :
OS                  : Linux 2.6.38-2-powerpc64
C compiler          : gcc version 4.6.1 20110428 (prerelease) (Debian 4.6.0-6)
libc                : libc-2.13.so
MEMORY INDEX        : 9.837
INTEGER INDEX       : 10.646
FLOATING-POINT INDEX: 15.882
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
-----

powerpc port:
-----

BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          781.12  :      20.03  :       6.58
STRING SORT         :           99.52  :      44.47  :       6.88
BITFIELD            :      1.8306e+08  :      31.40  :       6.56
FP EMULATION        :          168.36  :      80.79  :      18.64
FOURIER             :           11881  :      13.51  :       7.59
ASSIGNMENT          :          17.011  :      64.73  :      16.79
IDEA                :          3354.6  :      51.31  :      15.23
HUFFMAN             :            1149  :      31.86  :      10.17
NEURAL NET          :          23.981  :      38.52  :      16.20
LU DECOMPOSITION    :             731  :      37.87  :      27.35
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 42.220
FLOATING-POINT INDEX: 27.013
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
============================LINUX DATA BELOW============================
CPU                 : Dual
L2 Cache            :
OS                  : Linux 2.6.38-1-powerpc64
C compiler          : gcc version 4.5.2 (Debian 4.5.2-7)
libc                : libc-2.11.2.so
MEMORY INDEX        : 9.118
INTEGER INDEX       : 11.742
FLOATING-POINT INDEX: 14.982
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
-----
```

概要を言うと、 ppc64 port は powerpc port と比べ、誤差範囲程度で同じです。ただし、「 STRING SORT」で飛躍的に良くなり、「 IDEA」で若干悪く、「 LU DECOMPOSITION」で若干良くなっている結果となっています。

また、 VMX 命令サポートの効果ですが、どうやら nbench は CPU 性能を主に計測するベンチマークソフトらしく、各アプリケーション自体のベンチマークは測定できませんでした。



8 索引

Debian 勉強会資料

2011 年 5 月 21 日　　初版第 1 刷発行

東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）