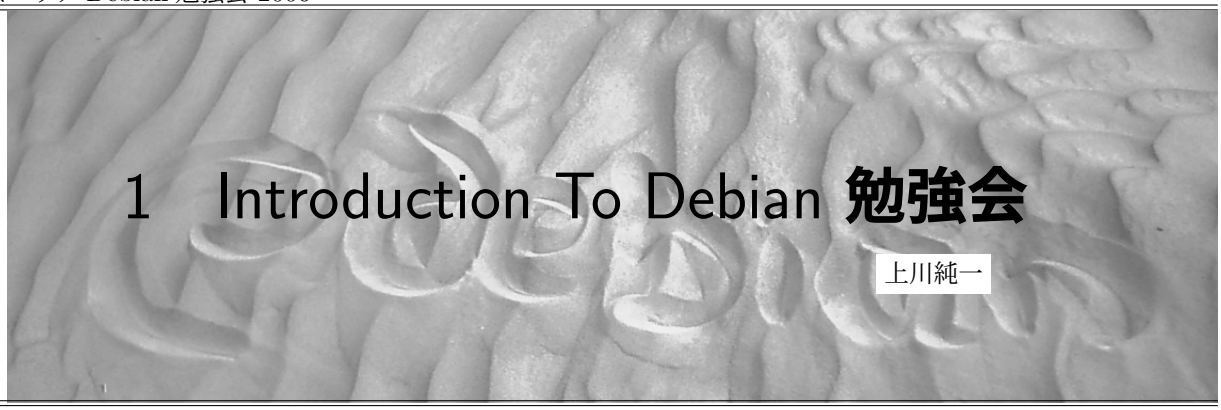
第 21 回 東京エリア Debian 勉強会
事前資料

Debian 勉強会会場係 上川純一*

2006 年 10 月 21 日

* Debian Project Official Developer

目次



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

- 武藤さん i18n meeting に参加してきた報告をします。
- 松山さん flash　について今日は語ります。
- 岩松さん 普段は SuperHっこなのですが、今日は RPM について説明します。
- 上川純一 宴会の幹事です。

1.2 事前課題紹介

今回の事前課題は「あなたの /etc/networking/interfaces をさらしてください」もしくは「Debian のネットワーク設定について思うこと」というタイトルで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 柏木さん

インターネットサービスの開発の仕事をしております。Debian を使い始めて数年になり、すっかり手放せなくなりました。自作のソフトウェアは deb 化してますが、よくわかってません。

これまで使ってばかりなので、いつか恩返しというか貢献したいと思っていましたが、Debianっていうくらいでなかなか敷居が高く、コミュニティに入って修行したいと思っていたところ、この勉強会のことを知りました。初めてですが参加させて頂きたく思います。よろしくお願いします。

今回の課題ですが、まずは周囲のマシンの /etc/network/interfaces をいくつかさらします。幸か不幸か mapping とかを駆使するまでは至っていません。

(1) Ethernet 4 ポートある小型 PC で、ブリッジとして使っています。

bridge-utils パッケージにより /etc/network/if-pre-up.d/bridge が作られ、ifupdown パッケージ自身に手をいれなくてもこのように新しい構文が使えるようになるというのは便利ですね。（そういえば、/etc/network/interfaces はてっきり ifupdown パッケージの conffile かと思ってましたが違うのですね）

```
auto lo
iface lo inet loopback

auto eth0

auto eth1

auto eth2

auto eth3

auto br0
iface br0 inet manual
       bridge_ports all
       bridge_stp off
       bridge_maxwait 10
       bridge_fd 3
```

(2) もうひとつ、IPv6 ルータになっているサーバ機の例です。eth0 が上流、eth1-4 が内側です。NAT や DHCP 等を提供しています。

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
       address 210.xxx.xxx.154
       netmask 255.255.255.240
       broadcast 210.xxx.xxx.159
       gateway 210.xxx.xxx.145

iface eth0 inet6 static
       address 2001:db8:1234:5678::154
       netmask 64
       up route -6 add default gw fe80::10 dev eth0

auto eth1
iface eth1 inet static
       address 172.20.0.1
       netmask 255.255.255.0
       broadcast 172.20.0.255
iface eth1 inet6 static
       address 2001:db8:1234:1020::1
       netmask 64

auto eth2
iface eth2 inet static
       address 172.20.1.1
       netmask 255.255.255.0
       broadcast 172.20.1.255
iface eth2 inet6 static
       address 2001:db8:1234:1021::1
       netmask 64

auto eth3
iface eth3 inet static
       address 172.20.2.1
       netmask 255.255.255.0
       broadcast 172.20.2.255
iface eth3 inet6 static
       address 2001:db8:1234:1022::1
       netmask 64

auto eth4
iface eth4 inet static
       address 172.20.3.1
       netmask 255.255.255.0
       broadcast 172.20.3.255
iface eth4 inet6 static
       address 2001:db8:1234:1023::1
       netmask 64
```

さて、Debian のネットワーク設定について結構文句はあるので、いくつか愚痴らせていただきます。全体的に RedHat 系に負けているような気がします。

- 前述と矛盾するかもしれませんが、設定できるパラメータ一覧がないのはわかりにくいです。/etc/network/if-*.d/ スクリプトには設定可能パラメータを表示するオプションをつけることを義務化して（イメージとしては init.d script の status 引数みたいなもの）　ifup –list-options とかで全部表示できたらいいなと思います。
- iface ethX inet static の中で、今時、netmask と broadcast を両方指定しなければならないのは面倒。ifconfig(8) がそうなのだから仕方ないが、デフォルトで勝手に補ってくれてもよいのでは。

- IPv6 まわりの設定方法がかなりいまいち。
  - autoconf 関係を制御できない。inet6 static と書いても autoconf でアドレスが付いてしまう。止めるには、/etc/sysctl.conf に net/ipv6/conf/default/autoconf=0 としなければならない。ここで net/ipv6/conf/all,eth0,.../autoconf=0 のように書いても動作しない。(/etc/init.d/procps.sh で sysctl を行うタイミングがかなり早く、インターフェースが作られる前だから？)　　inet6 autoconf なんてのがあるといいなあ。(ちなみに Redhat では変数で制御可能です)
  - inet dhcp があるんだから inet6 dhcp もほしい。DHCPv6 で、IPv6 の DNS サーバアドレスを配りたい。
  - netmask 64 って当たり前だから省略させてほしい。というかこれは用語が間違っていて、netmask じゃなくて prefixlen なのですが。
  - inet6 loopbackって意味あるのでしょうか。マニュアルには書いてありますが、書かなくても勝手に ::1 が付きます。
- ifupdown のエラーがわかりにくい。また、エラーがあったばあいに/etc/network/run/ifstate がおかしくなり、–force を連発することになる。(そういえば ifstate はなぜ /var にないのでしょう?)

長くなってしまいましたが以上です。

1.2.2 前田さん

```
interfaces -> interfaces.static

interfaces.dhcp
----------------------------
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp

interfaces.static
----------------------------
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
       address 192.168.222.xxx
       netmask 255.255.255.0
       network 192.168.222.0
       gateway 192.168.222.1
       broadcast 192.168.222.255

interfaces.union
----------------------------
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
       address 192.168.77.xxx
       netmask 255.255.255.0
       network 192.168.77.0
       gateway 192.168.77.254
       broadcast 192.168.77.255
```

自宅サーバのは全て固定です。普段利用しているクライアント（ノート PC）は、下記のように設定ファイルをいくつか用意して、シンボリックリンクを張り直して使っています。もっとうまいやり方があればご教授ください。(とはいえ別に今でも困ってはいませんが)

1.2.3 須藤さん

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# The first network card - this entry was created during the Debian installation
auto eth0
iface eth0 inet dhcp
    post-up wget -O /dev/null http://XXX:YYY@ZZZ.ZZZ.ZZZ.ZZZ/AAA.BBB.CCC.DDD/
    post-up wget -O /dev/null http://XXX:YYY@ZZZ.ZZZ.ZZZ.ZZZ/AAA.BBB.CCC.DDD/

auto eth1
iface eth1 inet static
    address 192.168.0.1
    netmask 255.255.255.0
    broadcast 192.168.0.255
    network 192.168.0.0
    pre-up /sbin/iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -j MASQUERADE
    post-down /sbin/iptables -t nat -D POSTROUTING -s 192.168.0.0/24 -j MASQUERADE

#iface eth1 inet6 static
#  address 2001:2f8:c2:201::fff3
#  netmask 64
#  gateway 2001:2f8:c2:201::1

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.1.1
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    wireless_essid ZZZ
    wireless_mode master
    wireless_key 0000-1111-2222-3333-4444-5555-66
    pre-up /sbin/iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j MASQUERADE
    pre-up /etc/init.d/dhcp stop
    pre-up /etc/init.d/bind stop
    post-up iwpriv wlan0 set_mac_mode 1
    post-up /etc/init.d/dhcp start
    post-up /etc/init.d/bind start
    post-down /sbin/iptables -t nat -D POSTROUTING -s 192.168.1.0/24 -j MASQUERADE
```

1.2.4 武藤 健志さん

常用の interfaces ファイルは noauto eth0; iface eth0 inet dhcp という感じでたいして面白いものではない。んで、Debian(および Linux) のネットワーク設定については不満だらけである。まず今の interfaces+ifupdown のフォーマットがすでに考古学博物館収録に値する代物 (有史以前遺産の netbase よりはマシだが) であり、CUI/GUI 問わず未だにまともなセットアップツールを提供できていない。インストーラのセットアップのほうがまだ気合いが入ってるぞ。エディタでこう記述しろなんて言われて喜んでるのは設定オタだけで十分である。特に無線周りは、Windows 辺りと比べてしまうと、そのヘタレっぷりに Debian CD を真っ二つにして窓から捨てたくなったとしても止めることはできないだろう。だいたい歴史的負の遺産な/etc/resolv.conf なんてファイルも気味が悪い。こんなのはカーネル内部に持たせてやれば DHCP のときにいちいち実ファイルを書き換える必要もないじゃないか。追加 NIC だってネットワーク接続するために付けたのであろうから、いちいちエディタで interfaces に追記する必要なく、新しい NIC デバイスを発見した段階でとりあえず DHCP でつなぎに行っても罰は当たらなそうなもんだ。まぁこういったことはデスクトップ開発者には 100 も承知なことなわけで、GNOME や KDE ではネットワークマネージャソフトの開発が進んでる。Debian はどうする。

1.2.5 野首さん

```
auto lo
iface lo inet loopback
auto eth0
mapping eth0
       script /usr/local/sbin/netenv
       map vmware vmware
       map native native
iface native inet static
       address 10.0.22.147
       netmask 255.255.252.0
       network 10.0.20.0
       broadcast 10.0.23.255
       up route add -net 10.0.0.0 netmask 255.0.0.0 gw 10.0.20.1
       up pon
iface vmware inet static
       address 192.168.157.45
       netmask 255.255.255.0
       gateway 192.168.157.2
```

netenv の中身はこんな感じです。

```
#!/bin/sh
lspci | grep -i vmware > /dev/null
if [ $? -eq 0 ]
then
       echo vmware
else
       echo native
fi
exit 0
```

物理ディスクにインストールした Debian を、native/vmware のどちらでも起動できるようにしています。pon しているのは、外部に出るための VPN を張るためです。

1.2.6 鍋太郎さん

普段持ち歩いている Thinkpad の /etc/network/interfaces です。

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface                職場の設定（有線）
noauto eth0
iface eth0 inet static
       address 172.x.x.x               # 職場の IP アドレス
       netmask 255.255.255.0               # 職場のネットマスク
       network 172.x.x.0               # 職場のネットワークアドレス
       gateway 172.x.x.x               # 職場のゲートウェイ
#        # dns-* options are implemented by the resolvconf package, if installed
       dns-nameservers 172.x.x.x       # 職場の dns サーバ

# The wireless network interface       自宅の設定（無線）
noauto eth1
iface eth1 inet dhcp
       wireless_essid any

# 使ってない pppoe の設定
iface dsl-provider inet ppp
provider dsl-provider
```

この他に Air-EDGE を使ってるので、pppconfig で作った設定が /etc/ppp 以下にあります。

また、見ての通り resolvconf パッケージで、/etc/resolv.conf を管理しようとしていますが、中途半端ですね。問題としては、Air-EDGE の ppp と 無線 LAN を切り替えると、通信に失敗するアプリケーションがあって、不便に思いながらも放置中。

放置しっぱなしと言うのもアレなので、今後の予定を挙げておきます。

1. まず、どのレイヤで通信ができていないのかを確認。切り替え後にも通信できるプログラム (firefox 等) があるので、ルーティングがうまくいっていないわけではなさそう。パケットキャプチャあたりで捕まえてみる必要あり。おそらく、参照する DNS サーバが違っていて、名前解決に失敗してるのではないか?
2. ppp の on/off と ifup/ifdown で設定を切り替えなければならないが、その方法の調査。

……でも、sylpheed を立ち上げ直すだけで解消してるんで、優先順位が限りなく低くなってたりします……

1.2.7 **キタハラさん**

しれっと、下記の様に回答すると「もしかして罠にはめられている?! 」と考えてしまうのは、少々被害妄想気味でしょうか？(実は本当に罠？)

```
k3@TPX22:~$ cat /etc/networking/interfaces
cat: /etc/networking/interfaces: No such file or directory
k3@TPX22:~$
```

Debian のメインマシンは、有線 Only 接続でこんな感じ。

```
k3@TPX22:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

k3@TPX22:~$
```

某所で設定中の colinux はこんな感じ。

```
colinux:~# cat /etc/network/interfaces
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo eth0

iface eth0 inet static
    address 192.168.0.2
    gateway 192.168.0.1
    netmask 255.255.255.0

iface lo inet loopback

colinux:~#
```

共に、まったく面白みが無いのですが、こんなのでよいのかな？

1.2.8 mhatta

「あなたの /etc/networking/interfaces をさらしてください」ということで、そもそも/etc/*networking*/interfaces というファイルは存在しないような気もしますが、私の/etc/network/interfaces は以下の通りです。

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# This is a list of hotpluggable network interfaces.
# They will be activated automatically by the hotplug subsystem.
mapping eth0
       script grep
       map eth0

# The primary network interface
allow-hotplug eth0

iface eth0 inet dhcp
iface eth1 inet dhcp
```

なんのひねりもなくてすみません。

1.2.9 **高杉さん**

debian のネットワーク設定に関しては、マシンの用途、接続方法などでいくつか設定方法があると思います。私の知る限りでは以下の設定があります。

1. /etc/network 以下 interfaces とインターフェース起動時の処理のスクリプト群
2. /etc/ppp 以下 ppp 接続時の処理のスクリプト群
3. dhcp 接続時
4. network-manager を使った設定

これらの設定の解説や設定例があるとよいとおもいます。

```
******* DMZ 家庭内 LAN 間 FW *******
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
# automatically added when upgrading
auto lo
iface lo inet loopback

# The first network card - this entry was created during the Debian
installation
# (network, broadcast and gateway are optional)
# automatically added when upgrading
auto eth0 eth1
iface eth1 inet static
       address 210.150.XXX.XXX
       netmask 255.255.255.XXX
       network 210.150.XXX.XXX
       broadcast 210.150.XXX.XXX
       gateway 210.150.XXX.XXX

#IP Masquarade
       up iptables -t nat -A POSTROUTING -s 192.168.X.0/24 -o eth1 -j MASQUERADE
#For Diablo
#        up iptables -t nat -A PREROUTING -p tcp --dport 4000 -i eth0 -j DNAT
--to 192.168.X.37:4000
#        up iptables -t nat -A PREROUTING -p udp --dport 4000 -i eth0 -j DNAT
--to 192.168.X.37:4000
#        up iptables -t nat -A PREROUTING -p tcp --dport 6112:6119 -i eth0 -j
DNAT --to 192.168.X.37:6112-6119
#        up iptables -t nat -A PREROUTING -p udp --dport 6112:6119 -i eth0 -j
DNAT --to 192.168.X.37:6112-6119

#Filtering
       up iptables -N block
       up iptables -A block -i eth1 -j DROP -p tcp --dport 137:139
       up iptables -A block -i eth1 -j DROP -p udp --dport 137:139
       up iptables -A block -o eth1 -j DROP -p tcp --dport 137:139
       up iptables -A block -o eth1 -j DROP -p udp --dport 137:139
       up iptables -A INPUT -j block
       up iptables -A FORWARD -j block
#routing
       up route add -net 192.168.X.0 netmask 255.255.255.0 eth0

iface eth0 inet static
       address 192.168.X.1
       netmask 255.255.255.0
       network 192.168.X.0
       broadcast 192.168.X.255
```

```
********** 家庭内ファイルサーバ ********
注）DHCP で IP を取得してますが、DHCP で IP 取得時に BIND に名前を登録しているの
で、書いていないでの運用上大きな問題はない。
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# This is a list of hotpluggable network interfaces.
# They will be activated automatically by the hotplug subsystem.
mapping eth0
       script grep
       map eth0

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp
```

1.2.10 小林儀匡さん

「あなたの/etc/networking/interfaces をさらしてください」。これに対する正しい回答は「そんなファイルありません」ですよね? 間違ってわざわざ恥ずかしい/etc/network/interfaces をさらしてしまった人はお疲れ様、という罠……。でも折角なので (?)、その罠に健気にまんまとはめられてみます。

以下、某所のもの (一部編集済み) です。……まったく面白くありませんが。

```
# The loopback interface
auto lo
iface lo inet loopback

# eth1 = Realtek (outer side)
auto eth1
iface eth1 inet static
  address xxx.xxx.xxx.xxx
  netmask 255.255.254.0
  gateway xxx.xxx.xxx.xxx

# eth0 = Intel PRO/100 VE (inner side)
auto eth0
iface eth0 inet static
  address 192.168.1.xxx
  netmask 255.255.255.0
```

これだけでは面白くないし折角なので、「Debian のネットワーク設定について思うこと」についても簡潔に回答してみます。

「**ダサい**」。以上。

Debian というか Linux 全般の問題だと思いますが、新しいデバイスが次々とカーネルでサポートされ、新しいソフトウェアがどんどん開発され、デスクトップも (まだまだという評価も否めませんが) 見違えるようにカッコよく使いやすくなっている一方で、なんだかダサいところは誰も (興味がないのか) 手をつけずダサいままになっているような気がします。別に設定ファイル全般が悪だとは思いませんが、設定まわりのインタフェースはもう少し改良できるのではないでしょうか。まぁ、だったらお前やれと言われても今の自分にはできないので、いつかその辺りが改善されるといいですね、と無難にまとめて回答とします。

1.2.11 やまねさん

vi で `/etc/network/interfaces` をいじるのもいいのですが、対話的インターフェイスが欲しい今日この頃です。base-config には設定がなかったっぽい。どこでやるんだっけ？やっぱりない？あとは

- 無線の設定がもっと簡単にならないかなーとか
- 一旦スリープ後、復帰したときにさくっと DHCP で IP 取ってくれないかなーとか思います。

あとは VPN なども統一して管理などできればいいんですが。色々と弄れるのはいいんだけど、簡単にある程度まで設定というのは難しいもんだなぁ、と。Windows だと AccessConnections[*1] などを使ってるんですが、同様に扱えればいいですよね。

1.2.12 中野さん

/etc/network/interfaces は下記の通りです。　普通のゲートウェイですね。

ネットワーク周りで昔嵌まったことですが、普段は dhcp でアドレスを取っているノート PC を、スイッチ周りのデバッグに用いたときのこと。

NIC のアドレスを sudo ifconfig で直接書換えたりしたかったのですが、どうも ifup/down と変は風に絡んで動いたり動かなくなったりで悩んだ結果、結局 interfaces を書換えて ifup / ifdown という面倒な手順を踏むことになった記憶があります。

ifup が単に ifconfig (とか route とか) のラッパではなく、なんか妙なことをしているのかなあ、と思いつつ深追いせずに放置して現在に至るのですが、このへんちゃんと理解したい、という気がしないでもありません。

[*1] `http://www-06.ibm.com/jp/pc/think/thinkvantagetech/accessconnections.shtml`

```
# The loopback interface
auto lo
iface lo inet loopback

# The first network card - this entry was created during the Debian installation
# (network, broadcast and gateway are optional)
auto eth1
iface eth1 inet static
       address 133.220.xxx.yyy
       netmask 255.255.255.0
       network 133.220.125.0
       broadcast 133.220.xxx.255
       gateway 133.220.xxx.zzz

auto eth0
iface eth0 inet static
       address 192.168.0.1
       netmask 255.255.255.0
       network 192.168.0.0
       broadcast 192.168.0.255
```

1.2.13 ysjj さん

```
$ cat /etc/network/interfaces
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
auto lo eth0 eth0:1

# The loopback interface
iface lo inet loopback

iface eth0 inet dhcp
iface eth0:1 inet static
   address 192.168.1.2
   netmask 255.255.255.0
iface eth0:2 inet static
   address 192.168.2.2
   netmask 255.255.255.0
```

1.2.14 えとーさん

IP 等は変更しました。madwifi のバグでレジュームからの復帰やインターフェースを落すと反応しなくなるのでどうにもちゃんと設定詰められていない。

wpasupplicant は wpa2 で AES にして AP の制限の最大値の値をキーにしています。BSSID も使ってるかな、、

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# This entry denotes the loopback (127.0.0.1) interface.
auto lo
iface lo inet loopback

# This entry was created during the Debian installation
# (network, broadcast and gateway are optional)
#auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
       address 192.168.XXX.XXX
       netmask 255.255.255.0
       network 192.168.XXX.XXX
       broadcast 192.168.XXX.255
       gateway 192.168.XXX.XXX

allow-hotplug ath0
iface ath0 inet manual
       wpa-driver madwifi
       wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

# no id_str, 'default' is used as the fallback mapping target
iface default inet dhcp

iface room inet static
  address 192.168.XXX.XXX
       netmask 255.255.255.0
       network 192.168.XXX.0
       broadcast 192.168.XXX.255
       gateway 192.168.XXX.XXX

iface dsl-provider inet ppp
    provider dsl-provider
# please do not modify the following line
    pre-up /sbin/ifconfig eth0 up # line maintained by pppoeconf
```

1.2.15 上川

/etc/network/interfaces であり、/etc/networking/interfaces ではありませんでしたね。無駄にハードルあげちゃいました。

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。

2.1 2006 年 38 号

http://www.debian.org/News/weekly/2006/38/ にある 9 月 19 日版です。
問題 1. Josselin Mouette が GNOME 2.16 について説明したのは

☐ A stable へのバックポート

☐ B experimental への投入

☐ C unstable への投入

問題 2. Russel Coker が議論する際の道具として提案したのは？

☐ A 意味の無い議論をベイジアンフィルタではじく

☐ B killfile の積極的な活用

☐ C 議論のサマリをまとめるための wiki の利用

問題 3. 最近 Debian にはいったパッケージである「ttf-vlgothic」の vl はどういう意味か？

☐ A ディストリビューションの名前

☐ B 私の名前はやまねです。

☐ C very local

2.2 2006 年 39 号

http://www.debian.org/News/weekly/2006/39/ にある 9 月 26 日版です。
問題 4. プロジェクトリーダーを罷免する決議が出たのはなぜか？

☐ A Dunk-Tank プロジェクトを発足したから

☐ B やるきがまったくみられないから

☐ C タイの政変の影響

問題 5. 一般決議に関しての規則がかわったのはなぜか

☐ A 現在多数提起されているから

☐ B くだらないものが提起されすぎだから

☐ C Dunk-Tank が気に入らないから

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 20 回目報告

東京エリア Debian 勉強会報告。9 月の第 20 回 Debian 勉強会を実施しました。Debian specific の話題、翻訳のすすめ、oprofile の使いかた、についての話がありました。

今回の参加人数は 12 人でした。

参加者は小林さん、あけどさん、さわださん、小室さん、キタハラさん、えとーさん、野首さん、高杉さん、前田さん、みつかさん、iwamatsu さん、上川でした。

まず、事前課題について紹介しました。 ruby の Debian パッケージに関してどうするべきか、という議論が起こりました。perl などの例をみると、 cpan はそれなりな品質を提供しているのですが、それでもひとつひとつのモジュールをパッケージにするのには手間をかけて作業しています。必要に応じてライブラリをパッケージ化するのがよいのではないか、という話でした。gems をパッケージに自動でするという仕組もよいですが、むしろ Debian パッケージをがりがり作成していって最新版を利用するのが Debian のほうが楽、と思われるくらいがよいねぇ、という話をしました。

恒例のクイズでは、いろいろと景品がたくさんあって、余りました。

澤田さんが「あなたが知らないうちに使っている Debian Specific」という題材で発表しました。adduser, ifup あたりはみんな実感として困っている経験があるということでした。Xsession の扱いについては、gdm のメンテナの Ryan Murray が Debian としての歩調をあわせてくれないので困っている、という話題がでました。ユーザのアカウント管理については、ID の重複などの問題があり、UNIX の世界にも ActiveDirectory を導入できないものか、ということで盛り上がりました。既存システムはそれぞれ作りこまれたシステムがあるので、移行が難しそうです。

小林さんが「翻訳のすすめ」について話をしました。DDTP が復活しているということで、みんな驚きを隠せないようでした。気軽に翻訳できるということで、OSC にて体験翻訳コーナーというのを作ってみるのもよいかも知れませんね。

上川が oprofile の使いかたの基本について説明しました。apt/dpkg を例にして、oprofile でプロファイルをした場合の例を紹介しました。来月は最適化した例を出せるとよいな、と思っています。

上川が LLGONG の報告として、realksh について説明をしました。興味をもってもらえたら幸いです。

最後に宴会を土間土間で開催しました。たべものを調子にのってたくさん頼みすぎました。おなかいっぱい。ロシアンルーレットにあたったのは小林さんと小室さんでした。辛そう … 。

二次会はデニーズでデビルズチョコレートサンデー。誰も電車にのれない状況になって、解散。

4 Jamon, tinto vino, por favor. ～スペイン Extremadura 州 Debian i18n 会議報告

武藤 健志 <kmuto@debian.org>

去る 2006 年 9 月 7 日～9 月 9 日の期間、スペイン Extremadura 州 Casar de Cáceres(図 1) において開催された「第 1 回 Debian 国際化会議 (*The first Debian internationalisation meeting*)」の模様を、技術トピックを中心に紹介する。

図 1　Extremadura 州 Cáceres(39°.28'15.72"N/6°22'18.87"W) CREOFONTE

4.1 概要

スペイン Extremadura 州は、支出の圧縮のために官公庁や教育機関に Debian GNU/Linux ベースのディストリビューション **LinEx**(`http://www.linex.org/`) の導入を進めているが、その過程で、Debian Project への謝意および、今後の開発と改良を継続への期待を込めて、Debian Project が必要とする各種会議を支援している。今回の会議も、この一環として行われ、各国から集まった参加者たちの航空券、食事、宿泊兼会議場「CREOFONTE」の提供といったすべてが同州の支援によって賄われた。

本会議は、Debian 公式開発者であり Debian インストーラなどの翻訳のとりまとめやドキュメント整備でリーダーシップを発揮している、フランスの Christian Perrier 氏の呼びかけで開催に至ったもので、世界各国から国際化に関して積極的な活動を行っている 23 名[*2]が集まり、3 日間にわたって充実した議論を行った。

主な議題については次のとおりである。

[*2] `http://wiki.debian.org/I18n/Extremadura2006` を参照。出身国はスペイン、ルーマニア、リトアニア、フランス、オーストリア、ベルギー、ドイツ、イタリア、イギリス、イスラエル、南アフリカ、ブラジル、米国、インド、カンボジア、バングラデシュ、日本と幅広い。

- Pootle 翻訳支援システムの採用推進
- DDTP/DDTSS の本格的な活用
- i18n タスクフォースの結成と活動の開始
- Debian インストーラおよび安定版における国際化の諸問題とその対策
- 非ラテン文字圏における、フォントおよび入力メソッドの概要紹介とドキュメント化の必要性
- その他国際化作業に向けたインフラストラクチャ整備

以降で、それぞれの技術詳細を述べていこう。

■キーワード

i18n 「Internationalization[a]」(国際化) の略称。一般に、アプリケーションを、技術的に大きな変更を要することなく、特定の言語・地域・文化に依存する部分 (メッセージ、アイコンなど) を分離してほかの言語・地域・文化に対応できるようにした設計およびその作業を指す。

l10n 「Localization」(地域化) の略称。特定の言語・地域・文化に合わせた作業およびその成果を指す。たとえば翻訳は、l10n 活動の 1 つである。

po 「Portable Object」の略称。GNU gettext で実装された i18n フレームワークにおける、メッセージカタログファイル。原文メッセージと対訳が 1 対 1 で構成され、短い翻訳については作業や再利用が容易である。詳しくは後述。

[a] -sation と z の代わりに s で表記されることもある。

4.2 gettext で実現される i18n と po ファイルの概要

本題に入る前に、Debian のアプリケーションの i18n および l10n で欠かすことのできない、GNU gettext の i18n フレームワークと、その中で重要な役割を担う po ファイルについて簡単に説明しておく。

GNU gettext は、GNU アプリケーション向けに開発された、メッセージカタログ向け i18n フレームワークで、GNU libc で全面的にサポートされている (図 2)。元々の概念は、Uniforum によって NLS 標準として発案され、Sun の Solaris で実装されたものだ。

GNU gettext の動作の流れは大まかに次のようになる。まず、i18n 化対象のアプリケーションのコードから `xgettext` などのツールを利用してメッセージ部分をテキスト形式の **pot ファイル**(*Portable Object Template*) として抜き出し、コードを gettext フレームワークを利用するよう改変する[*3]。

生成された pot ファイルは原文メッセージのマスターファイルとなる。このファイルを < **言語コード** >**.po** の名前を持つ **po** ファイル (*Portable Object*) としてコピーし、実際の翻訳を進めることになる。たとえば日本語であれば、`ja.po` が po ファイル名となる。po ファイルには翻訳ファイル自体のエンコーディングや、訳者名、日付などを記したいくつかのヘッダの後に、原文となる `msgid` とその対訳指定箇所となる `msgstr` のペアが羅列される。次にいくつかの例を示す。

[*3] 基本的には_("**メッセージ**") のようにアンダースコア 1 文字の関数で囲む。その他設定の詳細については GNU gettext の Info ファイルを参照。

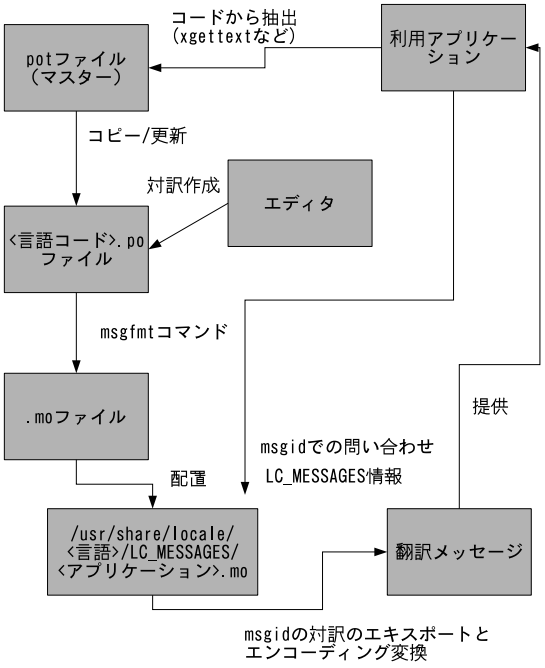


図 2　GNU gettext の仕組みの概略

```
# msgid に原文文字列、msgstr に訳を記述する
msgid "Japanese"
msgstr "日本語"

# plural 指示で単数型と複数型を分けることができる
msgid "an apple"
msgid_plural "apples"
msgstr[0] "1 個のリンゴ"
msgstr[1] "複数のリンゴ"

# 複数型を無視する場合は msgstr[1] に同じ文字列を並べるのでは
# なく、[0] だけにする
msgid "an apple"
msgid_plural "apples"
msgstr[0] "リンゴ"

# printf 書式の変換指定子を使う（実装依存）
msgid "user %s has %d files"
msgstr "ユーザ %s は %d 個のファイルを持っている"

# printf 書式文字列を使い、順序を入れ替える（実装依存）
# 順序を指定する場合、すべての変換指定子に番号付けする必要がある
msgid "%d files in %s directory"
msgstr "%2$s ディレクトリに %1$d 個のファイル"
```

GNU gettext の場合、po ファイルのままでは msgid からのメッセージ取り出しが複雑 (時間がかかる) になるので、msgfmt コマンドを利用してバイナリ形式の **mo ファイル**(*Machine Object*) に変換する。さらに、libc の gettext サポートを使ってこの mo ファイルを読ませるために、/usr/share/locale/< **言語** >/LC_MESSAGES/配下に「< アプリケーション名 >.mo」(正確には「ドメイン名」) で配置しておく。

これで準備は完了である。アプリケーション側でアプリケーション名 (ドメイン名) を宣言してバインディングした後、翻訳対象の msgid 文字列と、現在のメッセージロケール (LC_MESSAGES 環境変数。定義されていない場合は LANG 環境変数) で問い合わせると、ロケールに基いた上述のディレクトリから検索され、対応する msgstr 文字列が返される。このとき、メッセージロケールのエンコーディング情報に基いてエンコーディング変換も行われる。

GNU gettext はこのように mo ファイルと libc のサポートを利用しているが、po ファイルの構成自体は比較的単純であり、Debian における各 i18n フレームワークでも流用されている。たとえば、パッケージの構成についての質問やインストーラの質問を司る debconf インターフェイステンプレートの翻訳には po を流用した po-debconf 機構が使われており、これから説明する Pootle や po4a は、po をベースにしたより汎用的な i18n 翻訳手法である。

po ファイルはテキストファイルなので、どのようなテキストエディタでも操作可能であるが、未翻訳あるいは msgid が更新されたために曖昧 (fuzzy) になった翻訳などを順に追っていったり、訳語候補を提供したりできる支援ツールを使うことで、生産性を向上できる。このようなオフラインの po 翻訳支援ツールとしては、Emacs の po-mode、KDE の KBabel、GNOME の Gtranslator などがある (図 3)。

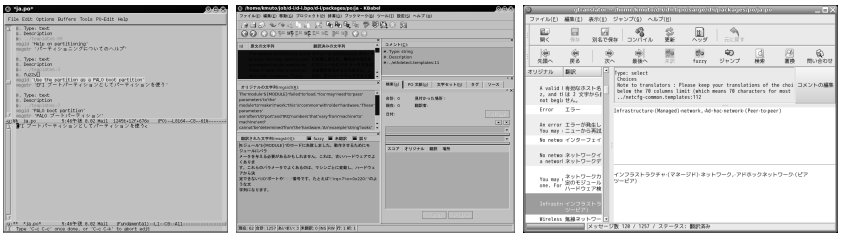
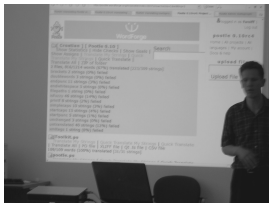


図 3　Emacs po-mode、KBabel、Gtranslator

4.3　Pootle **翻訳支援システム**

さて、今回の会議での主要な議論の 1 つが、Pootle 翻訳支援システムである。**Pootle**(http://pootle.wordforge.org/) は、WordForge Project(http://www.wordforge.org/) で開発が進められている Web ベースのオンライン翻訳管理サーバーだ。GNU GPLv2 でライセンスされており、Debian でも pootle パッケージをインストールすることで自身のサーバーを構築できる。なお、メインプログラムは Python で記述されている。

類似のものとしては、Canonical 社 (Ubuntu GNU/Linux の開発元) のコラボレーションサイト Launchpad.net(https://launchpad.net/) で使われている **Rosetta** システム (https://launchpad.net/rosetta) があるが、Rosetta はまだ Debian フリーソフトウェアガイドラインに沿ったソースコード公開はなされていない[*4]という問題があり、Debian での積極的な採用には反対の声が大きい。



会議では、Wordforge Project 創設者の Javier Solá 氏、Pootle 開発者の Friedel Wolff 氏、Google Summer of Code(GSoC) で Pootle の改善を進める Gintautas Milauskas 氏、それに Debian でのパッケージメンテナ Nicolas Francois 氏を中心に、Pootle の実装と今後の改良方針について議論が行われた。

Pootle は、その名のとおり po 形式を中核として翻訳を管理しており (内部表現は UTF-8 エンコーディング)、po の特性を生かしてデータベース内に存在する同一メッセージの再利用が可能となっている。po 形式のほかに、XLIFF 形式 (*XML Localization*

[*4] 「*Rosetta is not Open or Free Software at the moment. Rosetta will become open source sometime in the future but we don't have a date, although some parts of the Launchpad have already been released under the GPL by Canonical Ltd.*」

Interchange File Format)、Qt.ts 形式 (XML カタログ)、CSV 形式での入出力をサポートしている。

実際に Pootle のサイトに行き、使ってみればわかるように、Pootle はまだ開発中であり、実用に耐え得るほどの出来ではない。特に Web インターフェイスの能力が不十分であり、これをクライアントとして利用するにはかなりの苦難が予想される。Web インターフェイスの向上は今後の大きな課題ではあるが、GSoC の支援で Milauskas 氏がデータベースバックエンドと Web フロントエンドの分離 (抽象化) に成功したと会議において表明したこともあり、今後は既存のオフラインフロントエンドとの連携や、新たなオンラインフロントエンドの開発が進められるかもしれない[*5]。Pootle についてのドキュメントの整備はまだ追いついていないが、Pootle の開発メーリングリストでは日々活発な議論が行われており、我と思わん方はぜひ参加して頂きたい。

また、会議において提案され現在継続議論中の話題として、Pootle に「翻訳の翻訳」を実装してほしいという希望 (wishlist) がある。忘れがちなことであるが、世界各地の翻訳従事者の誰もが英語を理解できるわけではない。たとえば母国語以外の第二言語としてスペイン語・フランス語・ロシア語といった言語を使っている国は存在し、こういった国々で英語を解釈し無償で作業に従事する翻訳者を見つけるのは容易でない。たとえば Pootle 側で英語のほかに指定の第二言語が訳出済みであれば、次のようにそれを提示すれば、訳者は第二言語に従って翻訳できる。

```
msgid "English"
msgid[es] "Spanish"
msgstr "スペイン語を読んだ上での翻訳"
```

将来的には Debian における全翻訳を Pootle で賄うという壮大な構想が予定されているが、その上で必要となるのが既存の各種フォーマットと、Pootle で使う po 形式との相互変換である。

Debian パッケージの質問インターフェイスである debconf の翻訳テンプレートに対しては、Denis Barbier 氏の開発した **po-debconf** というテンプレート ↔po 間の相互変換の仕組みがある (テンプレート利用の際には po-debconf の利用を強制するようなポリシー改訂も本会議で提案された)。

その他のフォーマットへの対応としては、**poxml** と **po4a**(*po for anything*) がある。poxml は XML↔po 間の相互変換ツールであり、po4a は、XML、SGML、LaTeX、Dia ダイアグラム、POD(Perl の *Plain Old Document*)、man ページ、カーネルヘルプメッセージと多種に対応した相互変換ツールである。たとえば、Debian インストーラのドキュメントには poxml が採用されており、現在 XML ファイルを直接扱っている日本語についてもいずれは採用が求められることになるだろう。po4a は各種マニュアルや aptitude のドキュメントなどに使われており、実際の使い方については、小林儀匡氏の日記に詳しい (`http://dolphin.c.u-tokyo.ac.jp/~nori1/w/?cmd=view;name=Log200610`)。

なお、多数の po ファイルを格納して翻訳の再利用や用語集 (glossary) 作成が可能であることが Pootle その他の管理ツールにおける最大のメリットであるが、debian-www@debian.or.jp メーリングリストにおいて Seiji Kaneko 氏が疑念を呈したとおり、特に日本語においては英文の単語とユニークに 1 対 1 対応できることが少なく、訳者の著作権を認められ得る文が構成されるため、翻訳のライセンスの衝突に注意を払う必要がある。たとえばある用語集や翻訳が GNU GPL と衝突するライセンスのソフトウェアに由来する場合、GNU GPL のソフトウェアの翻訳においてそれを利用することは、ライセンス上の問題を抱えかねない。これについては、候補提供時にライセンス衝突の可能性を警告したり、今後翻訳者に何らかの (翻訳についての扱いをできるだけ自由にするための)「翻訳ライセンス同意書」を求めていくといった必要が出てくるだろう。

4.4 DDTP/DDTSS の展望

DDTP(*Debian Description Translation Project*) は、Michael Bramer 氏らによって長らく開発と作業が進められてきた、Debian の各パッケージの 1 行説明および長文説明 (Description) の翻訳インフラストラクチャである。再設計を行ったりサーバーがクラッシュしたりと長期にわたる活動停止をたびたび起こしていたが、ようやく簡易かつ堅固なサーバーシステムが実装され、活動が再開した。日本では現在角田慎一氏と田村一平氏

[*5] なお、Pootle の実験台および後述の i18n タスクフォースのためのインフラストラクチャとして、i18n.debian.net がセットアップされ、Extremadura のデータセンターでホスティングされている。

が “**ものすごい勢いで**”(©tsuno 氏) 翻訳を進めており、1 位のドイツに次ぐ翻訳数を叩き出している (http://svana.org/kleptog/temp/ddts-stats.html)。

現在の DDTP は、メールインターフェイスを使って翻訳対象の要求および提出を行うようになっており、内部表現は UTF-8 エンコーディングで統一されている。詳しくは Web の DDTP の説明 (http://www.debian.org/international/l10n/ddtp) を参照されたい。

DDTSS(*Debian Distributed Translation Server Satelite*。http://kleptog.org/cgi-bin/ddtss2-cgi/xx) は、DDTP の Web フロントエンドで、メールインターフェイス同様に翻訳対象の要求と提出、ほかの訳者によって提出された翻訳のレビューおよびコメント添付を Web ブラウザ上で容易に行うことができる。

このように、DDTP/DDTSS は強力で、かつ協力者の参入障壁の低いシステムだが、唯一の難点は翻訳データを実際に利用できる場面が極めて限られていることである。翻訳データは各 Debian ミラーにはかつて伝播されていたものの、現在は ddtp.debian.net ホストのみでの提供となっている上 (ミラーにあるものは古くて更新されていない)、現時点でユーザー環境で利用するには、次のように experimental 版から APT パッケージを取得しなければならない。

1. unstable または testing 環境において、experimental 版の APT リポジトリを加える。

```
deb http://ftp.jp.debian.org/debian experimental main
```

2. apt-get update; apt-get install apt/experimental↵を実行し、experimental 版の APT をインストールする。このとき、ライブラリバージョンが unstable のものと異なるため、aptitude などの APT ライブラリを利用しているパッケージは一旦 remove されることになる。
3. ddtp.debian.net の APT リポジトリを加える (etch も存在)。

```
deb http://ddtp.debian.net/debian sid main
```

4. 環境変数 LANG に ja_JP.UTF-8(あるいは ja_JP.EUC-JP) を設定した状態で、apt-get update↵を実行する[*6]。これにより、DDTP 日本語翻訳データである <**公式ミラー**>/dists/sid/main/i18n/Translation-ja.gz(古い。5 月 16 日以来更新されておらず、エンコーディングは EUC-JP)、および http://ddtp.debian.net/dists/sid/main/i18n/Translation-ja.gz(最新。エンコーディングは UTF-8) がダウンロードされる。
5. 環境変数 LANG に ja_JP.UTF-8(あるいは ja_JP.EUC-JP) を設定した状態で、たとえば apt-cache show apt↵を実行することで、日本語翻訳文が表示される (図 4)。

図 4　DDTP から取得した日本語メッセージ

[*6] デフォルトとなっている APT 設定 APT::Acquire::Translation "environment"; を environment の代わりに ja などとすれば環境変数に寄らずとも設定できるはずなのだが、現時点では environment が優先されることを避けられないようだ。

ただし、ddtp.debian.net に Packages ファイルが存在しないために、毎回警告メッセージが出てしまうという問題がある。aptitude や synaptic などの APT フロントエンドは、experimental 版の APT ライブラリでビルドし直すことで、同様に DDTP 翻訳を扱える。

今回の会議においては、Bramer 氏自ら現状の DDTP/DDTSS の説明の後、Pootle と DDTP の連携について可能かどうかの検討が行われた。現在の DDTP はシンプルながらそれなりに機能しており、あえて Pootle のように比較的複雑なシステムと連携する必要がどこまであるかという懸念を Bramer 氏は示していたが、用語の統一や訳の再利用という面でメリットがあることも同時に認めていた。なお、DDTP の試験的な Pootle への投入はすでに行われており、Pootle において改善すべきパフォーマンス上の問題があることがわかっている。

既存の experimental 版 APT の DDTP 対応実装が安定しており、各種派生パッケージでもうまく動作していることから Etch にマージを試みることが会議において提案されたが、残念ながら、この DDTP 対応の APT は、ABI 変更を伴うために現在リリースフリーズ中の Etch への収録は見送られることが決定した。今後、より検証を重ねた上で、Debian unstable へのマージが行われる予定だ。現時点で見る限り、experimental APT の動作は unstable のものと同程度に安定しており、ABI 変更を受けるパッケージを再ビルドしなければならないことを除けば、試用して問題になることも少ないだろう。また、せめて Web 版のパッケージ説明である `http://packages.debian.org/`において DDTP の成果を使えないかという提案が出されており、こちらも作業が進められることを期待したい。

4.5 i18n タスクフォース

翻訳などの l10n や、関連する i18n 改善を行う上で重要となるのがパッケージメンテナとの協調である。しかし不幸なことに、必ずしもメンテナがこのような改善に積極的であるとは限らず、翻訳が放置されたりあるいは理解の浅さからパッチが拒絶されたりすることが多々発生している。

リリース作業の一環として、リリースアシスタント Luk Claes 氏の合意の下、本会議では Debian **i18n タスクフォース**の結成と、**NMU キャンペーン**の実施が決定された。

i18n タスクフォースは、i18n/l10n における各種の問題についてのスペシャリスト集団として、窓口となるグループである。具体的な活動予定としては、ユーザーからの i18n/l10n におけるバグ報告の追跡と返答、メンテナや上流開発者への働きかけ、翻訳者/チームへの連絡を行う (図 5)。

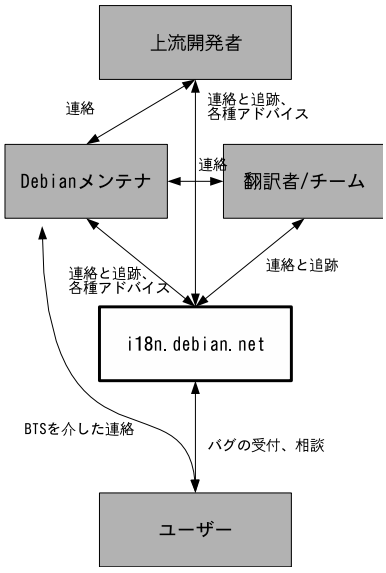


図 5　i18n タスクフォース

メーリングリスト debian-i18n@lists.debian.org、Wiki サイト http://i18n.debian.net/、IRC チャネル #debian-i18n@irc.oftc.net で活動する。図からも予想されるように、タスクフォースのメンバーの負荷は著しく高くなる恐れがある。各部分での自動化やテンプレート化、メンバーの勧誘といったことが今後の課題となるだろう。

NMU キャンペーンは、i18n タスクフォースの作業の 1 つである。i18n/l10n の翻訳・パッチ (特に po-debconf 関連と gettext 0.15 への移行) を受けていながら動きの見られないメンテナのパッケージに対し、一定の過程[*7]を踏んだ後に i18n タスクフォースに属する Debian 公式開発者が、NMU(*Non-Maintainer-Upload*) と呼ばれるパッケージの修正アップロードを行う。すでにこのミッションは開始しており、メーリングリスト debian-i18n において NMU に際して未翻訳言語の取り込みも行う旨の声明が Perrier 氏などからたびたび出されている。po-debconf 翻訳者・翻訳希望者の方はメーリングリストでの連絡に注意を払っておくとよいだろう。

4.6 インストーラおよび安定版の問題と解決

現在、Debian インストーラには世界人口の 67% に対応する 74 種の言語が登録されており、今後も Debian の普及のために全世界の言語をカバーすべくさらなる増加が予想される[*8]。しかし、この言語データの増加は、当然ながらその翻訳を収録するパッケージのサイズ増大を引き起こし、実メモリに RAM ディスクを展開して作業を行う Debian インストーラにとっては、抜本的解決策がない限りこれ以上の言語収容は厳しいというインストーラマネージャFrans Pop 氏からの非公式見解が示されている。また、インストーラに限らず、各パッケージの翻訳データの種類やサイズが増大することは、インストール環境でのディスクの圧迫 (特に組み込み環境) や、各言語の更新がパッケージメンテナにとって大きな負荷となり得る。さらに、翻訳は最終パッケージ形態になってからようやく追従されることも多く、安定版での翻訳更新のニーズもある。

本会議のブレーンストーミングセッションでこれらの問題が取り上げられ、いくつかの提案が寄せられた。

4.6.1 翻訳データの分割

Debian のインストーラは Debian パッケージとその debconf 翻訳の仕組みを生かして設計されている。インストーラは udeb パッケージというコンポーネントに分けられており、依存関係を使って動的にロードされる。翻訳は各 udeb ごとに収録されている。

前述したように、実メモリを利用するインストーラのサイズには、実用上、限りがある。これに対処する上で次の 2 つの提案がなされた。

1 つは、Christian Perrier 氏らによる「言語ごとにインストーラを分ける」というものである。つまり、ラテン語圏、アラビア圏、アジア圏、…といった具合だ。付随して、インストーラの第 1 段階で起動して言語を選んだ後に本当の (各言語の) インストーラを起動するという案も出された。ただ、いずれにしてもこの方法ではパッケージの分割が煩雑になる上、たとえば「アジア圏」と一緒くたにできるほどこれらの言語は等質ではない。日本語・中国語・韓国語を一緒にすることは結局アジア圏のサイズがほかの言語圏に比して圧倒的膨大になる上、特に GUI インストーラの場合には日本語と中国語のフォントは形状の違いから共有できない上に同じ文字コード番号で衝突する箇所があるなど、問題が大きい (むしろ日本語、中国語、韓国語は互いに独立した関係にしたほうがましであろう)。

もう 1 つは、武藤の提案した「選択した言語以外の言語については、インストーラコンポーネントの動的ロード時に取り除く」というものである。インストーラにはすでに「lowmem」という、メモリが少ないときに英語以外の言語データを切り詰める機構が用意されており、これを流用すれば比較的実装は容易だろう。Frans Pop 氏からも debian-boot メーリングリストにおいて同様の提案と賛同が寄せられており、おそらくこの方向で実装が進めることになる見込みだ。

[*7] NMU campaign for pending l10n bugs (http://people.debian.org/~lwall/i18n/)

[*8] Sarge では 42 言語 (50%)、Etch では現時点で 62 言語 (61%)。http://d-i.alioth.debian.org/i18n-doc/languages.html を参照。

4.6.2 ランゲージパックと tdeb

インストール後環境での翻訳の増大と安定版における更新を行う上で提案されたのが、ランゲージパックと tdeb である。

ランゲージパックは、Ubuntu でも採用されている方式で、大きくなりがちなパッケージの mo ファイルをひとまとめにし、言語ごとに別の deb パッケージとして一括化するものだ。現行の Debian における Firefox や OpenOffice.Org、KDE などで採用されている言語パッケージ (firefox-locale-ja など) を、もっとグローバルにして、各バイナリパッケージの翻訳を 1 つのパッケージにしたものと考えればよいだろう。Ubuntu では gcc や aptitude、console-tools などの比較的大きな mo ファイルを持つものがランゲージパックに分割されており、また GNU libc にパッチを当ててランゲージパックがインストールする`/usr/share/locale-langpack/<` **言語** `>/LC_MESSAGES` の中からもメッセージカタログを参照するように手が加えられている。

tdeb(*translation deb*) も同様に、オリジナルのバイナリパッケージから翻訳部分を抜き出し、別配布とするアイデアである。ただし、既存の deb ではなく各バイナリパッケージに対応する「tdeb」という新たなフォーマットを提唱している。Aigars Mahinovs 氏の提案している手順は次のとおりだ。

1. パッケージメンテナのビルド時 (debhelper でのフック)、またはアップロードしてアーカイブに入るまでの間に、バイナリパッケージから翻訳部分を tdeb として抽出する。たとえば日本語データの場合には hello-1.0-4.ja.tdeb のようにして、このパッケージには翻訳データのみを含める。
2. 各 FTP ミラーは、tdeb ファイルと、`Translations.gz` のようなインデックスファイルを提供する。インデックスファイルは、「<*packagename*>-<*version*>: <*separated-lang-list*>」の書式で、たとえば「`hello-1.0-4: es,fr,ja`」のようになる。
3. APT 側には、`/etc/apt/languages.list` のようなファイルでどの言語が必要かを指定しておく。パッケージのインストールやアップグレード時にバイナリ deb と合わせて tdeb をダウンロードする。
4. パッケージの実際のインストールを担当する dpkg では、バイナリパッケージがインストール済みであることを確認した後、tdeb パッケージを展開・インストールし、そのファイル一覧情報をバイナリ deb 同様 `/var/lib/dpkg/info/tdebs/hello.ja.list` のような形で配置する。そして、システムのパッケージ状態を示す`/var/lib/dpkg/status` ファイルの該当パッケージに `Installed-Translations` フィールドを追加し、ここにインストール済み言語を記述する。

上記に示したとおり、ランゲージパックにせよ tdeb にせよ、実際の対応に当たっては、C ライブラリ、ソースパッケージ、dpkg、APT、ミラー等々と各所での大きな変更が必要となるため、紆余屈折が予想される。tdeb の実装については、debian-i18n メーリングリストおよび Wiki(`http://wiki.debian.org/TranslationDebs`) で目下議論が行われているので、積極的にご参加頂きたい。

4.7 フォントと入力メソッド

非ラテン語圏の参加者も多かったため、この機会を使ってフォントと入力メソッドの説明も行われた。

カンボジアの Javier Solá 氏は、クメール文字の入力のデモンストレーションを行った。この南インド文字に似た表音文字は、母音と子音の組み合わせで文字がどんどん変形し、また縦横に伸縮していくものであり、入力側のほか、表現する上でツールキット側の対応も必要となる[*9]。デモは Windows で行われたが、(未確認ではあるものの)SCIM と OpenOffice.org の組み合わせで入力および表現できるようだ。

インドの Guntupalli Karunakar 氏は、ヒンディ語入力におけるコンソールと X のキーボードマップの差異の問題について語った。現在、コンソールのキーマップは console-tools によって提供され、

[*9] クメール文字の一部は Unicode 3.0 以降に収録されている。

X のキーマップは xkb によって提供されているが、両者のデータの持ち方に互換性がないため、管理が煩雑になっている。これについては、xkb 側をマスターとして、動的に console-tools 用のデータを生成できないかという提案がなされている。

このほか、インド系アメリカ人の Jaldhar Vyas 氏は SCIM を、武藤は類似の仕組みとして UIM を紹介した。

これらのトピックについては、開発元などによってある程度のドキュメントは揃えられているものの、全体を俯瞰して体系だった開発者向け・ユーザ向けの文書というのはまだ不足している。久保田智広氏の筆による i18n を俯瞰したドキュメント『Introduction to i18n』(`http://www.debian.org/doc/manuals/intro-i18n/`) は本会議においても大きな賞賛を受けていたが、フォントや入力メソッドについてのガイドのさらなる拡充が必要であろうという見解で一致した。日本人にとってもこれらのトピックは関心の高い分野であり、積極的協力を期待する。

4.8 まとめ

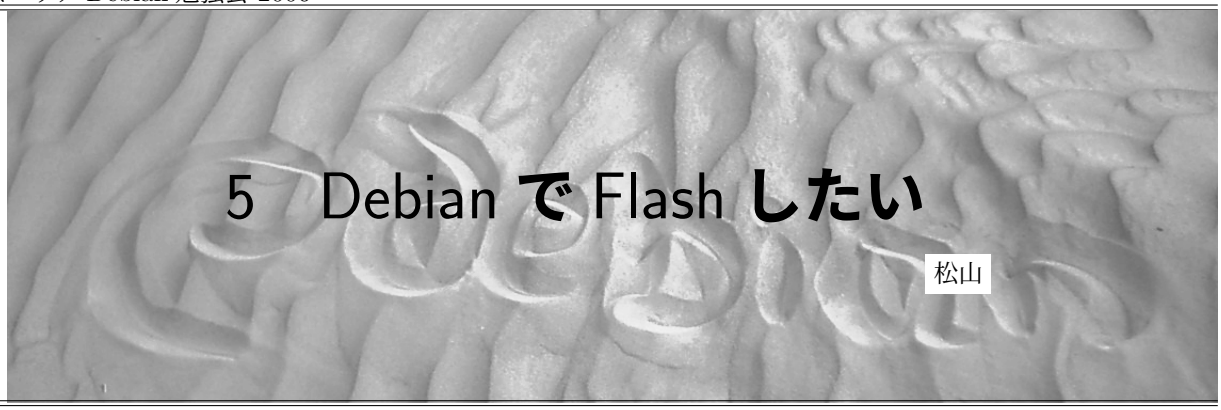
少人数で形成された本会議の会期中は、ほぼすべての時間が議論に費やされ、食事の場でも激論が大いに繰り広げられることもあった。その甲斐あって、人数の多さのために個々の「いつものグループ」に分かれがちな Debconf カンファレンスとは一味違った、中身の濃い、相互の情報交換と各種の有益な提案が生み出されることとなった。現地のロジスティックスをすべて担当した César Gómez Martin 氏の奮闘と、 コーディネータ役の Christian Perrier 氏の巧みな議事進行により、議論に集中できる環境が整っていたことも大きいだろう。

i18n/l10n 活動は 1 人だけで行うものではないし、またこなし切れるものでもない。同時に、この活動はプログラミングやハッカー倫理に熟知しなくとも参入しやすく効果の見えやすい分野のひとつである。できるだけ多くの人々を活動に招き入れ、Debian における日本語を含めた i18n/l10n の質と量の向上を図っていこうではないか。

なお、本会議の成功に伴い、来年も「Debian 国際化会議」は開催される予定である。この第 2 回には、やはり Debian での i18n/l10n 活動を進めておられる鍋太郎 (KURASAWA Nozomu) 氏にバトンを渡し、人脈の形成や議論への積極的な参加をお願いする予定だ。

了

—October 16th, 2006 Kenshi Muto



5 Debian で Flash したい

松山

5.1 Debian の Flash 事情

Debian の Flash 事情は、再生、作成ともに少し寂しい状況のようです。再生に関しては、Debian ではバージョンが古いものの (バージョン 7.Windows 版はバージョン 9)、Flash Player がされているので、これをインストールすれば、Flash が再生可能です。Debian unstable には gnash というプレーヤのパッケージがあり、フリーのものではこれが少し有名なようです。また、swf-player というパッケージもあり、いくらか制限があるものの、これでも Flash が再生可能です。このように、Debian では、Flash を何とか再生する環境はあるものの、Flash を作成する環境についてはかなり寂しい状況になっているようです。

5.2 Debian で Flash を作成したい

人が Flash を作成したい理由にはいろいろあると思いますが、とにかく Debian には Flash を作成するためのツールというものがほとんどないようです。Debian でなんとか Flash を作成できないかと探していると、ming というライブラリに出会いました。これは、Flash ファイルを作成するプログラムのためのライブラリです。つまり、ming というライブラリを使用してプログラムを作成し、その作成したプログラムを実行すると Flash ファイルが出力されます。このあたりが少しややこしいのですが、今のところ単なるライブラリとして提供されているだけなので、Flash を作成するのが困難であるという問題があるものの、私達が直接作成するのは「Flash を作成するプログラム」なので、作成 Flash を動的に変更するというメリットもあります。ming はコア部分は C でできていて、C++、Perl、PHP、Python、Java などに拡張されているようです。ming 関連のパッケージは oldstable や unstable、testing にはあるようですが、stable にはないようです。これは当時のメンテナ Erich Schubert が orphan した結果 2002 年に削除された影響です。166973[*10] このように、ming はちょっと怪しい雰囲気がありますが、今回はこれを検証してみました。

5.3 ming の検証

ming は単なるライブラリです。別途 GUI なラッパーアプリを作成すれば、きっと Debian のキラーアプリになると思います。ただ、私の気力や力量もあり、また、とにかく土台となる ming 自体、どんな Flash でも作成できるのかどうかを確認しておく必要があると思い、今回は ming の検証をしてみました。検証作業は、「ming を使ってこんな Flash は作れるのかな?」といういくつかの観点に立って実際に ming を使用した Flash を作成するプログラムを作成し、できた/できないという結論をつけていきました。[*11]

[*10] `http://bugs.debian.org/166973`

[*11] できた場合にはそれでよいのですが、できない場合は私が Flash についてよくわかっていなくてできないのか、ming の問題でできないのかの切り分けができていません。

表 1　ming の検証結果

分類	観点		結果	備考
描画	テキスト		可	フォントの埋め込みが必要?
	線		可	
	画像		可	PNG 画像、JPEG 画像の取り込みが可能。
動画	音楽再生		一部可	WAV ファイルの読み込みは可。MP3 読み込みの API はあるが、音が潰れる。
	動画埋込		不可	mpeg 等の動画を埋め込む API がない。
	動画再生		可	フレームの概念はある。線やテキスト等をプログラムによって移動させることが可能。
インタラクティブ	ボタン操作		一部可	テキストをボタンとすることが可能。画像、動画はボタンにできない。
	テキストフィールド (ボックス)		可	入力値を ActionScript で取り込む方法がよくわからない。
	マウス操作			
データ	XML			
	HTTP			

6 rpmstrap を活用する

岩松

6.1 始めに

みなさん、rpmstrap を御存じでしょうか。「これは Debian 勉強会なんじゃないの？RPM の話なんて関係ねーじゃねーか！」と思った人もおられると思いますが、今回は Debian 環境上で RPM な chroot 環境を構築することができる rpmstrap について説明しようと思います。

6.2 rpmstrap とは？

Debian では chroot 環境等を構築するツールとして、debootstrap[*12] がありますが、RPM を使って、chroot 環境を構築するツールとして rpmstrap というものがあります。debootstrap と同様、wget[*13]を使って、http/ftp 経由でパッケージを取得します。なので、基本的にインターネットにつながった環境が必要になります。

Debian では testing と sid にあり、sarge にはありません。次期リリースの コードネーム Etch には収録される予定です。

6.3 インストール

```
# apt-get install rpmstrap
```

でインストールできます。

6.4 使い方

rpmstrap は root 権限が必要です。root 権限を持ったユーザー等で実行する必要があります。

6.4.1 とりあえず、chroot 環境を構築してみる

rpomstrap を使って、CentOS 4.0 の環境を構築してみます。chroot を構築するには以下のコマンドで行います。

```
# rpmstrap centos4 install_path
```

第 1 引数に対象ディストリビューション、第 2 引数にはインストール先を指定します。

実行すると、ネットワーク経由で RPM パッケージをダウンロードしてきます。

[*12] http://packages.debian.org/unstable/admin/debootstrap
[*13] http://packages.debian.org/unstable/web/wget

```
iwamatsu@rahute:~/rpm # rpmstrap --verbose centos4 ./centos/
rpmstrap: debug: Preparing variables
rpmstrap: debug: Loading /usr/lib/rpmstrap/scripts/centos4 suite
rpmstrap: debug: Working out mirror
rpmstrap: debug: Work out RPMS
rpmstrap: debug: setup_env()
rpmstrap: debug: Install RPMS
rpmstrap: debug: setup_env()
rpmstrap: debug: get_rpms(): Getting RPM from http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/
rpmstrap: debug: wget  http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/setup-2.5.37-1.3.noarch.rpm
--21:56:09--  http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/setup-2.5.37-1.3.noarch.rpm
           => `setup-2.5.37-1.3.noarch.rpm'
mirror.centos.org を DNS に問いあわせています... 72.21.40.10
mirror.centos.org|72.21.40.10|:80 に接続しています... 接続しました。
HTTP による接続要求を送信しました、応答を待っています... 200 OK
長さ: 31,051 (30K) [application/x-rpm]

100%[=====================================================================>] 31,051        64.83K/s

21:56:10 (64.69 KB/s) - `setup-2.5.37-1.3.noarch.rpm' を保存しました [31051/31051]

rpmstrap: debug: get_rpms(): Getting RPM from http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/
rpmstrap: debug: wget  http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/filesystem-2.3.0-1.i386.rpm
--21:56:10--  http://mirror.centos.org/centos/4/os/i386/CentOS/RPMS/filesystem-2.3.0-1.i386.rpm
           => `filesystem-2.3.0-1.i386.rpm'
mirror.centos.org を DNS に問いあわせています... 72.21.40.10
mirror.centos.org|72.21.40.10|:80 に接続しています... 接続しました。
HTTP による接続要求を送信しました、応答を待っています... 200 OK
長さ: 15,608 (15K) [application/x-rpm]

100%[=====================================================================>] 15,608        48.90K/s

21:56:11 (48.77 KB/s) - `filesystem-2.3.0-1.i386.rpm' を保存しました [15608/15608]

.............（中略）

rpmstrap: debug: Installing pass number 53...
rpmstrap: debug: Installing nano-1.2.4-1.i386.rpm to /home/iwamatsu/rpm/./centos...
警告: nano-1.2.4-1.i386.rpm: Header V3 DSA signature: NOKEY, key ID 443e1821
rpmstrap: debug: Installing pass number 54...
rpmstrap: debug: Installing openldap-2.2.13-4.i386.rpm cyrus-sasl-2.1.19-5.EL4.i386.rpm
 cyrus-sasl-md5-2.1.19-5.EL4.i386.rpm to /home/iwamatsu/rpm/./centos...
警告: openldap-2.2.13-4.i386.rpm: Header V3 DSA signature: NOKEY, key ID 443e1821
rpmstrap: debug: Installing pass number 55...
rpmstrap: debug: Installing libuser-0.52.5-1.el4.1.i386.rpm to /home/iwamatsu/rpm/./centos...
警告: libuser-0.52.5-1.el4.1.i386.rpm: Header V3 DSA signature: NOKEY, key ID 443e1821
rpmstrap: debug: Installing pass number 56...
rpmstrap: debug: Installing passwd-0.68-10.1.i386.rpm to /home/iwamatsu/rpm/./centos...
警告: passwd-0.68-10.1.i386.rpm: Header V3 DSA signature: NOKEY, key ID 443e1821
rpmstrap: debug: Installing pass number 57...
rpmstrap: debug: ...nothing left to do.
rpmstrap: debug: Done
```

これで構築の完了です。

6.5 chroot 環境にログインする

chroot 環境にログインするためには root 権限で chroot を実行します。

```
# chroot ./centos
```

6.5.1 RPM データベースを作成

chroot 後に最初しないといけなことです。/var/lib/rpm に RPM のデータベースが構築されていないので、構築する必要があります。

```
# rpm --rebuilddb
```

6.6 rpmstrap の仕組み

rpmstrap の仕組は以下の通りです。

- /usr/lib/rpmstrap/scripts 以下の設定ファイルをパーサする。
- wget でパーサしたファイルを取得する。
- rpm コマンドで 取得した RPM ファイルをインストールする。

```
rpm--install --root インストール先 --dbpath インストールする RPM パッケージ
```

という感じで行われます。

6.7 設定ファイル

RPM を取得するパッケージのレポジトリ等の設定を行っているファイルが

```
/usr/lib/rpmstrap/scripts/
```

にあります。rpmstrap で取得可能なレポジトリはこのディレクトリ下のファイルのみになります。新しいディストリビューションを追加する場合は設定ファイルを追加する必要があります。現在は

- centos3 (Cent OS 3)
- heidelberg(Fedora Core 3)
- sl402 (Scientfic Linux 4.02)
- suse10.0 (Suse 10.0)
- tettnang (Fedora Core 2)
- centos4 (Cent OS 4)
- mandriva10 (Mandriva 10)
- sl304 (Scientfic Linux 3.04)
- stentz (Fedora Core 4)
- suse9.3 (Suze 9.3)
- yellowdog4 (YelloDog Linux 4.0)

をサポートしています。pdk というファイルで設定ファイルの雛型があるので、それを見て設定ファイルを作成するとよいでしょう。今回は日本で人気のある RPM を使ったディストリビューションのひとつである、 VineLinux [*14] がサポートされていないようなので、追加してパッチ[*15]を送りました。

6.8 rpmstrap の気になるところ

rpmstrap を使ってみて、気になるところがありました。

- 構築までに時間がかかる。
  無駄なファイルが多く、構築までに 30 分ほど時間がかかります。設定ファイルに記述する RPM を吟味するといいかもしれません。
- 設定ファイルが書きづらい。
  RPM を使ったディストリビューションは多いのですが、相互でバージョンが一致していなく、設定ファイルにバージョンも記述しないといけません。よって、RPM がひとつでもアップデートされると書き直す必要があります。Debian の場合はファイル名だけなのでこのような問題は発生しません。また、ディストリビューションが増える毎に設定ファイルが増えていくという問題もあります。
- ダウンロードできないファイルがあるところどころダウンロードができない RPM パッケージがあります。ダウンロードできないパッケージがあるため、環境を構築することができないディストリビューションもあります。
  テストしたところ、以下のような結果になりました。

6.9 Debian ユーザから見た rpmstrap の使いどころ

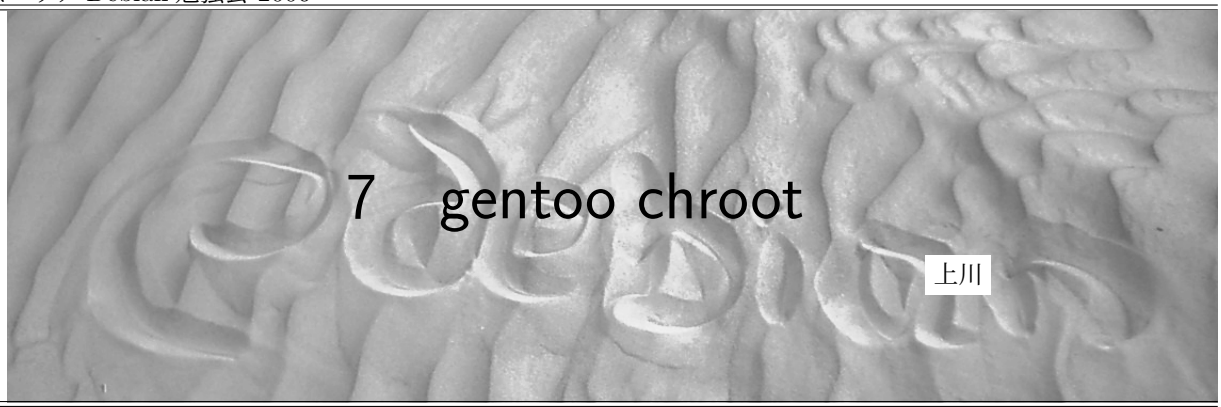
Debian ユーザとして rpmstrap をどのように使えばいいのか考えてみました。

[*14] http://www.vinelinux.org
[*15] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=392942

表 2　rpmstrap テスト結果

ディストリ	構築 可/不可
centos3 (Cent OS 3)	不可
heidelberg(Fedora Core 3)	可
sl402 (Scientfic Linux 4.02)	不可
suse10.0 (Suse 10.0)	可
tettnang (Fedora Core 2)	可
centos4 (Cent OS 4)	不可
mandriva10 (Mandriva 10)	可
sl304 (Scientfic Linux 3.04)	可
stentz (Fedora Core 4)	可
suse9.3 (Suze 9.3)	可
yellowdog4 (YelloDog Linux 4.0)	不明

- Debian が動作しているマシンで RPM のパッケージをコンパイルするために chroot 環境を構築したり...。
- RPM を使っている ディストリビューション上で別の RPM ディストリビューションを構築したり...。



7 gentoo chroot

上川

Debian 上で、 gentoo を chroot にインストールする方法について説明します。変態度合が伝われば幸いです。この手順、つくってから気づきましたが、実はあまり Debian 関係ないです。

7.1 gentoo の最低限のインストール

まず、gentoo の stage1 の tarball を取得してきます。適当なミラーにおいてあります。ここでは `http://mirror.datapipe.net/gentoo/releases/amd64/2006.0/stages/` から取得してきました。適当な場所にインストール先のディレクトリを作成し、そこで stage1 の tarball を展開します。アプリケーションの動作に最低限必要な proc ファイルシステムをマウントし、resolv.conf を chroot 内部にコピーし、chroot します。これで emerge ができる状況になったので、 emerge しまくるようです。

```
Debian$ sudo tar xfjp stage1-amd64-2006.0.tar.bz2
Debian$ sudo mount -t proc proc/ proc/
Debian$ sudo cp /etc/resolv.conf etc/resolv.conf
Debian$ sudo chroot .
Gentoo# env-update
>>> Regenerating /etc/ld.so.cache...
Gentoo# source /etc/profile
Gentoo# emerge --sync

ここで大量の出力

Gentoo# emerge portage
```

7.2 gentoo 自身のブートストラップ

wiki の手順では下記のようにすると順番にブートストラップしてくれるようです。あらゆるプログラムをコンパイルしてインストールするので時間が非常にかかります。個人的にはすでに飽きてしまったのでもう検証していません、続きはまた誰かが後でやってくれることを期待しつつ。

```
Gentoo# env-update && source /etc/profile && emerge --oneshot --nodeps
 gcc-config && USE="-* build bootstrap" emerge linux-headers && \
/usr/portage/scripts/bootstrap.sh && emerge -O libperl && emerge -O
 python && emerge --deep system && \
emerge syslog-ng xinetd grub hotplug coldplug vixie-cron reiserfsprogs
 reiser4progs sysfsutils udev dhcpcd && \
emerge --nodeps acpid ntp && rc-update add syslog-ng default &&
 rc-update add net.eth0 default && rc-update add vixie-cron default && \
rc-update add xinetd default && rc-update add sshd default && rc-update
 add hotplug default && rc-update add coldplug default && \
rc-update add acpid default
```

参考文献

- Gentoo wiki `http://gentoo-wiki.com/HOWTO_Install_Gentoo_-_The_Gentoo_Developers_Method_with_NPTL_and_2.6_from_Stage1`

8 apt を最適化してみる

上川

前回までで、oprofile を利用して apt のプロファイリングをしてみました。それでは、実際にどういう考え方で高速化を検討するか、を考えてみましょう。

最適化の必要な部分の解析

プロファイル結果を利用して、解析します。

`apt-get update` の結果を確認したところ、下記のようになるということがわかりました。どうも、SHA1Transform と MD5Transform という関数の負荷が高いようです。

```
sudo apt-get update
[中略]
CPU: Core Solo / Duo, speed 1833 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Unhalted clock cycles) with a unit mask of 0x00 (Unhalted core cycles) count 180000
samples  %        image name               app name                 symbol name
23823    46.3519  libapt-pkg-libc6.3-6.so.3.11.0 apt-get
SHA1Transform(unsigned int*, unsigned char const*)
12732    24.7724  libapt-pkg-libc6.3-6.so.3.11.0 apt-get
 MD5Transform(unsigned int*, unsigned int const*)
4282      8.3314  processor.ko             processor                acpi_processor_idle
2584      5.0276  libc-2.3.6.so            apt-get                  (no symbols)
2012      3.9147  vmlinux                  vmlinux                  __copy_to_user_ll
503       0.9787  gpgv                     gpgv                     (no symbols)
222       0.4319  vmlinux                  vmlinux                  timer_interrupt
166       0.3230  libapt-pkg-libc6.3-6.so.3.11.0 apt-get
 MD5Summation::Add(unsigned char const*, unsigned long)
```

最適化方法の検討

プロファイル結果解析ををうけて適用できる最適化方法を検討します。

`apt-pkg/contrib/sha1.cc`, `apt-pkg/contrib/md5.cc` を見ると md5 については、 dpkg の実装を、 sha1 についてはどこからか拾ってきた実装を利用しており、 C++ で書かれた汎用のコードを利用しているということがわかります。仮説として、アッセンブリでばりばりにチューニングした実装を利用すれば高速になるのではないか、と考えてみます。

GPL 互換の既存の sha1 と md5 の高速な実装を探してみます。

代表的な GPL 互換の実装である GNUTLS に含まれている sha1 / md5 の実装は、`gl/sha1.c`, `gl/md5.c` にあます。確認したところ、特に高速化されていないようです。

そこで、 git のソースを見てみます。ppc と arm 用の最適化されている sha1 の実装が含まれていますが、 i386 用はないようです。

予備試験をしてみます。600MB 程度の iso ファイルの sha1 をとるのに、mozilla 実装と openssl 実装でどらくらい違うのかを比較してみました。mozilla 実装で 14 秒程度、 openssl 実装で 12 秒程度です。

ここまでで、既存の実装を応用する限りにおいて、試験環境において劇的に高速化できる方策は無いということがわかりました。残念。

Debian 勉強会資料

2006 年 10 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）