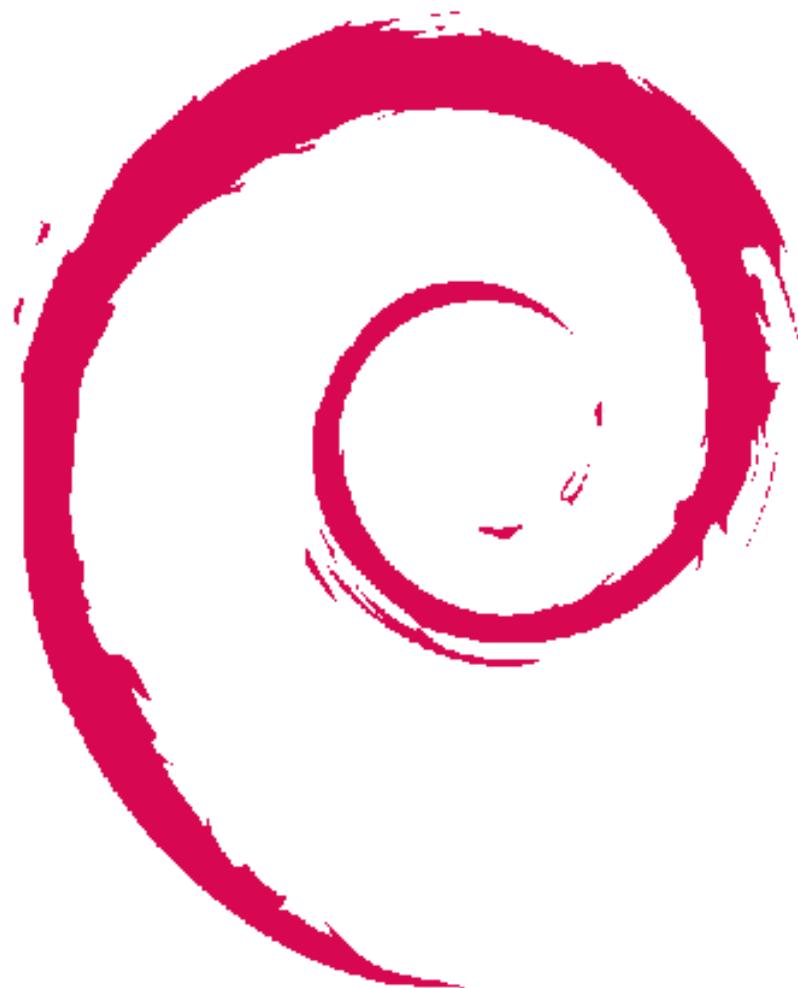


2006年冬号

あんどきゆめんてつど でびあん



東京ニアリアDebian勉強会 著

目次

1	MacBook に Debian をインストール	2
2	あなたが知らないうちに使っている Debian specific	14
3	翻訳へのさそい	17
4	dpkg, apt のプロファイリング	27
5	apt を最適化してみる	33
6	Jamon, tinto vino, por favor. ~スペイン Extremadura 州 Debian i18n 会議報告	34
7	Debian で Flash したい	46
8	rpmstrap を活用する	48
9	gentoo chroot	52
10	パッケージングについて	53
11	sid を日常環境として使うための注意	64
12	bugreport 論	65
13	Debian Weekly News trivia quiz	67
14	Debian Weekly News 問題回答	72

――『あんどきゅめんてっど でびあん』について――

本書は、東京周辺で毎月行なわれている『東京エリア Debian 勉強会』で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。収録範囲は勉強会第 18 回から第 22 回まで。内容は無保証、つっこみなどがあれば勉強会にて。

1 MacBook に Debian をインストール

上川

Apple が 2006 年春に発売開始した Intel ベースの MacBook に Mac OS X と Debian を dual-boot でインストールするときの流れを紹介します。

Mac OS X を削除して Debian のみをインストールする方法については、おそらく lilo を MBR から起動するように設定すれば最新ファームウェアは起動してくれますが、検証していません。

1.1 インストール用にパーティション準備

購入直後の状態では、Mac OS X が全部の領域を占めています。その Mac OS X パーティションを縮小し、Debian がインストールできるようにします。Mac OS X は 20GB 程度の領域を必要とするようですので、20GB まで縮小してしまいましょう。

diskutil resizevolume コマンドでボリュームサイズを動的に変更することができます。^{*1}

```
Mac OS X $ df -h
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/disk0s2     74G   17G   57G   23%       /
devfs          95K   95K    0B   100%      /dev
fdesc          1.0K  1.0K    0B   100%      /dev
<volfs>        512K  512K    0B   100%      /.vol
automount -nsl [171]    0B   0B    0B   100%      /Network
automount -fstab [179]   0B   0B    0B   100%      /automount/Servers
automount -static [179]  0B   0B    0B   100%      /automount/static
/dev/disk0s1     197M  512B   197M   0%       /efi

Mac OS X $ sudo diskutil resizevolume disk0s2 20G
Started resizing on disk disk0s2 Macintosh HD
Verifying

Resizing Volume
Adjusting Partitions

Finished resizing on disk disk0s2 Macintosh HD
WARNING: You must now reboot!

# diskutil list
/dev/disk0
 #:          type name           size      identifier
 0: GUID_partition_scheme          *74.5 GB  disk0
 1:        EFI                  200.0 MB  disk0s1
 2:        Apple_HFS Macintosh HD    20.0 GB  disk0s2
```

1.2 rEFIt のインストール

rEFIt は EFI 専用ブートローダです。rEFIt^{*2} イメージを Mac OS X にインストールします。インストールする場所はどこでもよいのですが、ドキュメントに従ってみましょう。/efi あたりにファイルを展開し、rEFIt に含まれている、./enable.sh を実行します。スクリプト内部で bless コマンド^{*3}を実行してくれます。これで、起動時に自動で rEFIt が実行されるようになります。

^{*1} resizevolume コマンドは Mac OS X 10.4.6 の機能拡張のようです。

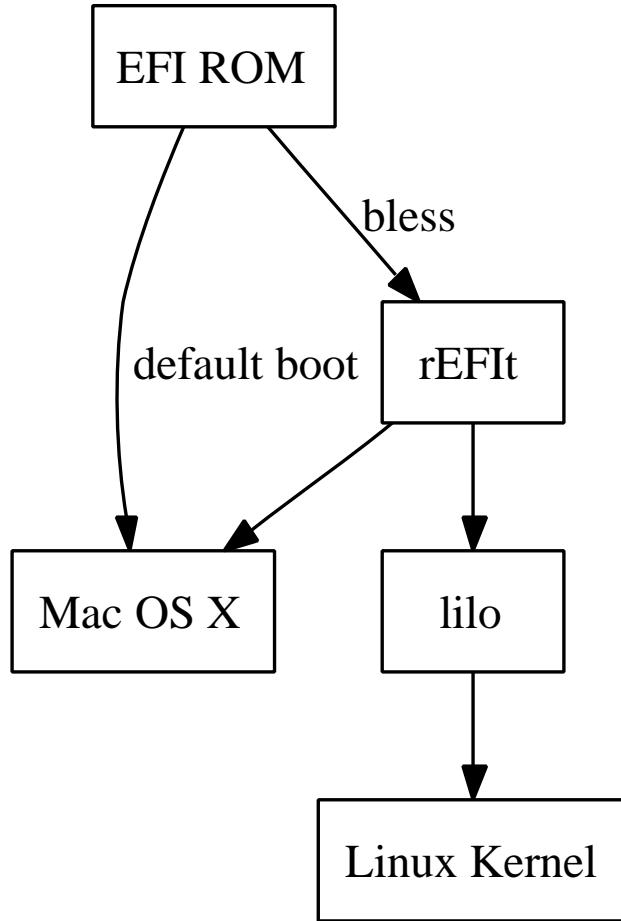
^{*2} <http://refit.sourceforge.net/> 執筆時点のバージョンは 0.7 でした。

^{*3} EFI での OS 起動優先順序を変更してくれるツール

ます。

Debian の rEFIt パッケージを利用してインストールする場合には、バージョン 0.7-3 時点では enable.sh が提供されていません。直接 bless コマンドを入力してください。

`sudo bless --folder [refit.efiのあるディレクトリへのフルパス] --file [refit.efiへのフルパス]`



1.3 Debian のインストール

2006 年 7 月版以降の etch^{*4}のインストーラを利用してインストールします。

CDROM から起動するためには、CDROM を挿入してから、C を押しながら起動すればよいです。もしくは、option キーを押しながら起動するとファームウェアの選択画面が起動します。rEFIt のメニューからも CDROM からの起動を選択できます。^{*5}

パーティションを切る部分^{*6}を過ぎ、パッケージがインストールされたら、LILO をインストールする直前の部分まで実施します。

この時点では LILO が現在動作できない状態になっています。^{*7}ここで、MBR を GPT に同期

^{*4} これ以前については動作確認をしていません。

^{*5} 2006 年 7 月時点で Debian Installer で利用している Linux カーネル 2.6.15, 2.6.16あたりでは Intel Mac に対応できていない問題があり、5 回に 4 回程度は「APIC エラー」なるものが発生し、起動に失敗するので、根気よく起動するまでがんばってください。2.6.17 以降では Intel Mac 向けの修正が一部マージされているので、状況は改善しています。

^{*6} 注意事項としては、既存の EFI FAT と Mac OS X のパーティションは削除しないこと。LILO をインストールする予定のパーティションはパーティション番号 3 か 4 にすること、ということがあります。5 番目以降のパーティションは MBR の制限があるので利用できません。

^{*7} parted が GPT の仕様に準拠しており、partition 1 のみしかない MBR 上のパーティションテーブルを再作成していることによるよう

させる作業を実施します。ここで、Alt-F2 で仮想コンソールを切替え、コマンドラインにうつります。gptsync コマンドを実行してください^{*8}。現状のインストール方法としては、chroot /target bin/sh としてインストール先の chroot に入り、そこから apt-get install refit でパッケージをインストール、そして gptsync コマンドで GPT から MBR に同期させます。

```
Shell> hd23a2:\efi\tools\gptsync

Current GPT partition table:
#  Start LBA    End LBA  Type
1      40      409639  EFI System (FAT)
2      409640    42352679  Mac OS X HFS+
3      42352680    44305805  EFI System (FAT)
4      44305806    83368386  EFI System (FAT)
5      83368307    89227682  Linux Swap

Current MBR partition table:
# R  Start LBA    End LBA  Type
1          1      156301487  EE  EFI Protective

Status: MBR table must be updated.

Proposed new MBR partition table:
# R  Start LBA    End LBA  Type
1          1      409639  EE  EFI Protective
2 *      409640    42352679  AF  Mac OS X HFS+
3      42352680    44305805  EF  EFI System (FAT)
4      44305806    83368386  EF  EFI System (FAT)

May I update the MBR as printed above? [y/N] -
```

この状態で、インストーラの画面に Alt-F1 で戻り、LILO を MBR ではなく、Linux 用のパーティションにインストールします。再起動すると rEFIt から Linux を指定して起動できるようになっています。

1.4 各種デバイスの設定

1.4.1 X の設定

X は i810 ドライバで設定します。915resolution パッケージをインストールします。解像度は 1280x800 です。

/etc/default/915resolution の例です：

です。

^{*8} 今後はインストーラから実施できるように改善したいです。

```
#  
# 915resolution default  
#  
# find free modes by /usr/sbin/915resolution -l  
# and set it to MODE  
# e.g. use MODE=54  
MODE=32  
#  
# and set resolutions for the mode.  
# e.g. use XRESO=1024 and YRESO=768  
XRESO=1280  
YRESO=800  
#  
# We can also set the pixel mode.  
# e.g. use BIT=32  
# Please note that this is optional,  
# you can also leave this value blank.  
BIT=
```

xorg.conf の例です*9：

*9 デフォルトで外部出力もするように設定してあります

```

Section "Files"
    FontPath      "/usr/share/fonts/X11/misc"
    FontPath      "/usr/X11R6/lib/X11/fonts/misc"
    FontPath      "/usr/share/fonts/X11/cyrillic"
    FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic"
    FontPath      "/usr/share/fonts/X11/100dpi/:unscaled"
    FontPath      "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
    FontPath      "/usr/share/fonts/X11/75dpi/:unscaled"
    FontPath      "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
    FontPath      "/usr/share/fonts/X11/Type1"
    FontPath      "/usr/X11R6/lib/X11/fonts/Type1"
    FontPath      "/usr/share/fonts/X11/100dpi"
    FontPath      "/usr/X11R6/lib/X11/fonts/100dpi"
    FontPath      "/usr/share/fonts/X11/75dpi"
    FontPath      "/usr/X11R6/lib/X11/fonts/75dpi"
# path to defoma fonts
FontPath      "/var/lib/defoma/x-ttcidfont-conf.d dirs/TrueType"
EndSection

Section "Module"
    Load          "i2c"
    Load          "bitmap"
    Load          "ddc"
    Load          "dri"
    Load          "extmod"
    Load          "freetype"
    Load          "glx"
    Load          "int10"
    Load          "type1"
    Load          "vbe"
EndSection

Section "InputDevice"
    Identifier    "Generic Keyboard"
    Driver        "kbd"
    Option        "CoreKeyboard"
    Option        "XkbRules"      "xorg"
    Option        "XkbModel"     "pc104"
    Option        "XkbLayout"    "us"
    Option        "XkbOptions"   "ctrl:nocaps"
EndSection

Section "InputDevice"
    Identifier    "Configured Mouse"
    Driver        "mouse"
    Option        "CorePointer"
    Option        "Device"       "/dev/input/mice"
    Option        "Protocol"    "ExplorerPS/2"
    Option        "Emulate3Buttons" "true"
EndSection

Section "InputDevice"
    Identifier    "Synaptics Touchpad"
    Driver        "synaptics"
    Option        "SendCoreEvents" "true"
    Option        "Device"       "/dev/psaux"
    Option        "Protocol"    "auto-dev"
    Option        "HorizScrollDelta" "0"
EndSection

Section "Device"
    Identifier    "Generic Video Card"
    Driver        "i810"
    Screen        0
    Option "MonitorLayout" "CRT,LFP"
    BusID         "PCI:0:2:0"
EndSection

Section "Device"
    Identifier    "Device1"
    Driver        "i810"
    Screen        1
    Option "MonitorLayout" "CRT,LFP"
    BusID         "PCI:0:2:0"
EndSection

```

続く

```

Section "Monitor"
    Identifier      "Generic Monitor"
    Option          "DPMS"
    HorizSync       28-64
    VertRefresh     43-60
EndSection

Section "Monitor"
    Identifier      "External Monitor"
    Option          "DPMS"
    HorizSync       28-64
    VertRefresh     43-60
EndSection

Section "Screen"
    Identifier      "Default Screen"
    Device          "Generic Video Card"
    Monitor         "Generic Monitor"
    DefaultDepth    24
    SubSection "Display"
        Depth          1
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          4
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          8
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          15
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          16
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          24
        Modes          "1280x800" "1024x768" "800x600" "640x480"
    EndSubSection
EndSection

Section "Screen"
    Identifier      "Secondary Screen"
    Device          "Device1"
    Monitor         "External Monitor"
    DefaultDepth   24
    SubSection "Display"
        Depth          1
        Modes          "1024x768" "800x600"
    EndSubSection
    SubSection "Display"
        Depth          4
        Modes          "1024x768" "800x600"
    EndSubSection
    SubSection "Display"
        Depth          8
        Modes          "1024x768" "800x600"
    EndSubSection
    SubSection "Display"
        Depth          16
        Modes          "1024x768" "800x600"
    EndSubSection
    SubSection "Display"
        Depth          24
        Modes          "1024x768" "800x600"
    EndSubSection
EndSection

Section "ServerLayout"
    Identifier      "Dual-monitor Layout"
    Screen 0 "Default Screen"
    Screen 1 "Secondary Screen" LeftOf "Default Screen"
    # Option "Clone" "On"
    #Option "Xinerama" "On"
    InputDevice "Generic Keyboard"
    InputDevice "Configured Mouse"
    InputDevice "Synaptics Touchpad"
EndSection

Section "DRI"
    Mode           0666
EndSection

```

キーバインドは .xsession^{*10}の中で次のような設定をしています。右の apple キーを押すと全

^{*10} 最近はデフォルトでは .gnomerc というファイルが使われるようです。 GDM からデフォルトのシステムセッションを明示的に選択すれば

角・半角キーに割り当てられています。option と apple キーはよく押し間違えるので、両方を Alt_L として設定しています。また、イジェクトキーとキーボードの下の部分にある ENTER キーをマウス用のキーとして定義しています^{*11}。また、外部マウスを USB で接続した場合も問題なく動作します。

```
xmodmap -e "keycode 115 = Alt_L"
xmodmap -e "keycode 116 = Zenkaku_Hankaku" # right-apple
xmodmap -e "keycode 108 = Pointer_Button3" # KP-ENTER
xmodmap -e "keycode 204 = Pointer_Button2" # eject
xkbset m
```

1.4.2 lilo の設定

いつもの癖で boot(/dev/sda3, ext2) と root(/dev/sda4, ext3) をわけてしまっているのでちょっとややこしい例ですが、現在利用している lilo.conf の例です：

```
boot=/dev/sda3
root=/dev/sda4
map=/boot/map
delay=20
default=Linux-20060705

image=/boot/vmlinuz-2.6.17dancer-20060701
    label=Linux-20060701
    read-only

image=/boot/vmlinuz-2.6.17dancer
    label=Linux-20060705
    read-only

image=/vmlinuz
    label=Linux
    read-only

image=/vmlinuz.old
    label=LinuxOLD
    read-only
    optional
    initrd=/initrd.img.old
```

デフォルトでインストールされているカーネルが 2.6.17 以前のものであれば、よく起動時にパニックをおこすので、Intel Mac 対応の 2.6.17 以降のものに変更しましょう。

1.4.3 サウンドカード設定

サウンドカードは snd_hda_intel ドライバで対応できる ALSA のオーディオデバイスです。

```
$ cat /proc/asound/cards
0 [Intel] HDA-Intel - HDA Intel
          HDA Intel at 0x90440000 irq 50
```

1.4.4 CPU の動的周波数設定

cpufreq は speedstep_centrino で動作します。apt-get install cpufreqd でインストールして、cpufreqd を動作させてあげると、動作します。

1.4.5 USB の設定

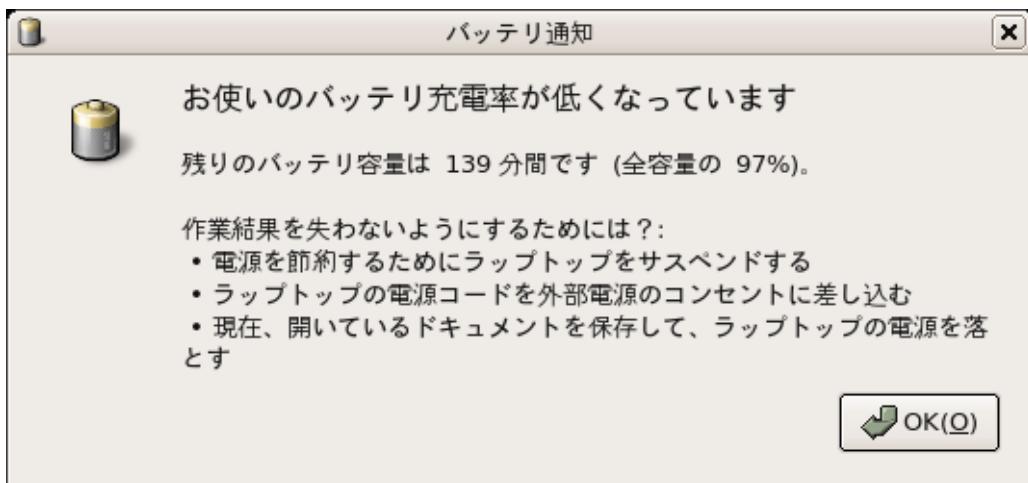
USB は UHCI, EHCI です。通常は特に設定は必要ないはずです。

1.4.6 電源設定

バッテリーはまともにサポートしているようです。ただ、電源の全容量が出ていないので、gnome から変なメッセージは出ました。

^{*}xsession を実行してくれるようです。

^{*11} xkbset パッケージが必要。



1.4.7 ネットワークの設定

有線ネットワークは SKY2 のドライバを利用します。

無線ネットワークは madwifi で対応できます。インストール方法は下記です。

- sudo apt-get install madwifi-source madwifi-tools madwifi-doc
- sudo m-a prepare
- sudo m-a a-i madwifi
- sudo modprobe ath_pci

放っておくと hotplug により、起動時に自動ロードされて有効になります。/etc/hotplug/blacklist.d/にファイルを作成し、下記のような内容を追加しておくと、手動でロードしないと有効にならないようにできます。飛行機にのる場合などのためには必要かもしれません。

```
ath_pci
```

以下、インストール時のログの例です。

```
$ sudo apt-get install madwifi-source madwifi-tools madwifi-doc
$ sudo m-a prepare
Getting source for kernel version: 2.6.17dancer
/lib/modules/2.6.17dancer/source のカーネルヘッダを利用できます
 symlink を作成中...
apt-get install build-essential
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... 完了
以下のパッケージが新たにインストールされます:
build-essential
アップグレード: 0 個、新規インストール: 1 個、削除: 0 個、保留: 94 個。
6916B のアーカイブを取得する必要があります。
展開後に追加で 20.5kB のディスク容量が消費されます。
取得:1 http://ftp.jp.debian.org sid/main build-essential 11.2 [6916B]
6916B を 0s で取得しました (81.0kB/s)
パッケージフィールドを読み込んでいます... 完了
パッケージ状態を読み込んでいます... 完了
バグレポートを取得しています... 完了
(データベースを読み込んでいます ... 現在 101852 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。)
(.../build-essential_11.2_i386.deb から) build-essential を展開しています...
build-essential (11.2) を設定しています ...

完了!
$ sudo m-a a-i madwifi
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... 完了
madwifi-source はすでに最新バージョンです。
アップグレード: 0 個、新規インストール: 0 個、削除: 0 個、保留: 94 個。

1 パッケージについての情報を更新しました
Extracting the package tarball, /usr/src/madwifi.tar.bz2, please wait...
```

```

dancer@coreduo: /tmp
ファイル(F) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
dancer@coreduo: /home/d... X dancer@coreduo: /home/d... X dancer@coreduo: /tmp X
madwifi-source をビルド中。ステップ 1。お待ちください...
80211_crypto_ccmp.o

8%

```

/home/dancer/shared/git/madwifi-modules-2.6.17dancer_0.svnr1644.0.9.0-2+20060705_i386.deb が完了しました。
未選択パッケージ madwifi-modules-2.6.17dancer 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 101861 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。)
(.../madwifi-modules-2.6.17dancer_0.svnr1644.0.9.0-2+20060705_i386.deb から) madwifi-modules-2.6.17dancer を展開しています...
madwifi-modules-2.6.17dancer (0.svnr1644.0.9.0-2+20060705) を設定しています ...

```

$ sudo modprobe ath_pci
$ lsmod | grep ath_pci
ath_pci           82212  0
ath_rate_sample   11776  1 ath_pci
wlan            167132  4 wlan_scan_sta,ath_pci,ath_rate_sample
ath_hal          192208  3 ath_pci,ath_rate_sample
$ dmesg | tail -20
eth1: no IPv6 routers present
ath_hal: module license 'Proprietary' taints kernel.
ath_hal: 0.9.17.2 (AR5210, AR5211, AR5212, RF5111, RF5112, RF2413, RF5413)
wlan: 0.8.4.2 (svn r)
ath_rate_sample: 1.2 (svn r)
ath_pci: 0.9.4.5 (svn r)
Device '[PX52] is not power manageable<6>ACPI: PCI Interrupt 0000:02:00.0[A] -> GSI 17 (level, low) -> IRQ 169
PCI: Setting latency timer of device 0000:02:00.0 to 64
wifio: 11a rates: 6Mbps 9Mbps 12Mbps 18Mbps 24Mbps 36Mbps 48Mbps 54Mbps
wifio: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
wifio: 11g rates: 1Mbps 2Mbps 5.5Mbps 11Mbps 6Mbps 9Mbps 12Mbps 18Mbps 24Mbps 36Mbps 48Mbps
wifio: H/W encryption support: WEP AES AES_CCM TKIP
wifio: mac 10.3 phy 6.1 radio 10.2
wifio: Use hw queue 1 for WME_AC_BE traffic
wifio: Use hw queue 0 for WME_AC_BK traffic
wifio: Use hw queue 2 for WME_AC_VI traffic
wifio: Use hw queue 3 for WME_AC_VO traffic
wifio: Use hw queue 8 for CAB traffic
wifio: Use hw queue 9 for beacons
wifio: Atheros 5424: mem=0x90100000, irq=169

$ /sbin/ifconfig ath0
ath0      リンク方法:イーサネット ハードウェアアドレス 00:16:CB:BA:76:E7
          inet アドレス:192.168.22.42 ブロードキャスト:192.168.22.255 マスク:255.255.255.0
          inet6 アドレス: fe80::216:cbff:feba:76e7/64 範囲:リンク
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1176 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1607 errors:0 dropped:0 overruns:0 carrier:0
          衝突 (Collisions):0 TX キュー長:0
          RX bytes:230678 (225.2 KiB) TX bytes:1306390 (1.2 MiB)

```

1.4.8 リモコン

赤外線のリモコンは使えるようです。カーネル用のデバイスドライバが存在します。2.6.18 以降にとりこまれるのではないでしょうか？ユーザ空間で利用できるドライバは作成しておきまし

た。^{*12}

1.4.9 iSight

iSight は linux-uvc デバイスです。ファームウェアのロードが必要です。次の手順でインストールができます。

- apt-get install linux-uvc-tools linux-uvc-source
- module-assistant auto-install linux-uvc

アプリケーションは ekiga などを利用しましょう。v4l2 デバイスなので、v4l2 対応のソフトウェアが必要です。

- apt-get install ekiga libpt-plugins-v4l2

実際にロードするには、Mac OS X のデバイスドライバに入っているファームウェアをロードしてからモジュールをロードします。ドライバのある場所のディレクトリ階層が深いので注意。

- sudo mount /dev/sda2 /mnt/macossx
- sudo macbook-isight-firmware-loader
/mnt/mac/System/Library/Extensions/IOUSBFamily.kext/Contents/PlugIns(次 の
行に続く)
/AppleUSBVideoSupport.kext/Contents/MacOS/AppleUSBVideoSupport
- modprobe uvcvideo

1.4.10 未確認のデバイス、手法

Debian を自動起動させる方法がわかりません、rEFIt はデフォルトでは、MacOSX もしくは eLILO を起動しようとしてしまいます。eLILO を起動すると起動できない。優先度の変更はどうやつたらよいのか、というのがいまいち不明です。

サスペンドの方法。

スリープの方法。

CD-R の動作はまだ確認していません。PATA パッチが必要という噂です。

```
# cdrecord -scanbus
scsibus0:
    0,0,0      0) *
    0,1,0      1) 'ATA      , 'ST98823AS      , '7.01' Disk
    0,2,0      2) *
    0,3,0      3) *
    0,4,0      4) *
    0,5,0      5) *
    0,6,0      6) *
    0,7,0      7) *
```

バックライトの制御ができるドライバは作成されているので、2.6.18 か 19 くらいには入るのではないでしょうか。

bluetooth については未調査。

^{*12} <http://www.netfort.gr.jp/~dancer/diary/daily/2006-Jul-12.html.ja>

1.5 発表履歴

本資料は下記の場所での発表資料として作成されたものです。内容については隨時更新しながら、いくつかの場所で発表しています。

- 2006年7月2日 秋葉原、CodeFestAkihabara 2006: 最終報告
- 2006年7月6日 恵比寿、SGI ホール、カーネル読書会: mixi.jp の話の前座
- 2006年7月15日 北海道、OSC-Do 2006: 「Debian 勉強会」のセッション
- 2006年7月29日 日々谷、TLUG: 「MacBook に Mac OS X と Debian を dual-boot でインストール」

1.6 参考文献

ファームウェアの bootcamp まわりの開発の影響で、ほとんどの web 上の手順を書いてある文献は現在の時点で手順が古くなっているので、参考にならない場合が多いですが、今後更新されるかもしれません。

- MacBook Developer Note: MacBook の論理構成図、ハードウェアの概観が解説されています。http://developer.apple.com/documentation/HardwareDrivers/Conceptual/MacBook_0605/index.html
- 赤外線リモートコントロール用、IR Receiver パッチ http://sourceforge.net/mailarchive/message.php?msg_id=16309282 <http://www.madingley.org/macmini/kernel/ir.patch>
- 赤外線リモートコントロールで XPDF プрезентーションするパッチ。<http://www.netfort.gr.jp/~dancer/diary/daily/2006-Jul-12.html.ja#2006-Jul-12-00:00:06>
- MacBook の仕様：簡単に概要だけが説明されています。<http://support.apple.com/specs/macbook/macbook.html>
- iSight (IEEE1394 外部デバイス) のプログラミングガイド <http://developer.apple.com/documentation/Hardware/Conceptual/iSightProgGuide/iSightProgGuide.pdf>
- bluetooth のドキュメント:http://developer.apple.com/documentation/HardwareDrivers/Conceptual/HWTech_Bluetooth/index.html#/apple_ref/doc/uid/TP40003032
- mactel linux のページ <http://mactel-linux.org/>、ここからたどれるメーリングリストで有用な情報が交換されています。
- rEFIt のページ <http://refit.sourceforge.net/>
- <http://sharealike.org/index.php?m=200605>
- バックライト制御 <http://modular.math.washington.edu/macbook/backlight/>
- Ubuntu のインストールについてのまとめページ <http://desrt.mcmaster.ca/macbook.xhtml>
- Gentoo の情報ページ http://gentoo-wiki.com/HARDWARE_Apple_MacBook
- MadWifi Wiki <http://madwifi.org/wiki/UserDocs/Distro/Debian/MadWifing>
- Macbook Pro build-in iSight <http://blogs.gnome.org/view/rbultje/2006/07/08/0>
- linux usb video class, linux-uvc <http://linux-uvc.berlios.de>

- Debian wiki MacBook <http://wiki.debian.org/MacBook>

2 あなたが知らないうちに使っている Debian specific

澤田さん

2.1 はじめに

Debian パッケージを管理するためのツールである apt や dpkg などは一目で Debian specific とわかります。しかし、日常的に利用しているコマンド、設定ファイルなどでも実は Debian 固有のものであったりパッケージ化する際に変更が加えられていたりするものがあります。ここではそのようなあなたの知らない Debian specific を紹介します。

2.2 adduser

Debian で

```
# adduser hoge
```

を実行すると hoge ユーザが作られパスワードの入力が求められます。Fedora で同じコマンドラインを実行した場合、hoge ユーザは作られます但しパスワード入力は求められません。パスワードは別途 passwd コマンドで設定する必要があります。

この違いは Debian の adduser と Fedora での adduser の実体の違いによるものです。Fedora の adduser は useradd へのシンボリックリンクです。Debian の adduser は Perl スクリプトで、useradd、passwd 等を呼び出してユーザを作成しています^{*13}。

ちなみに、FreeBSD の adduser はシェルスクリプトで書かれており、pw というコマンドを呼び出すことでユーザの追加を行っているようです。Debian GNU/kFreeBSD は FreeBSD カーネルの上に GNU のツールや glibc、Debian のツールを乗せたものであるため、Debian の adduser が使われており、useradd と passwd でユーザの追加を行っていました。

2.3 ifup

ネットワークの設定を変えたのでインターフェースを再起動したいという場合、Debian では以下のコマンドラインを実行すると auto に設定されているインターフェースをすべて再起動することができます。

```
# ifdown -a && ifup -a
```

一方 Fedora では-a オプションは使えず、また、複数のインターフェースを指定することはで

^{*13} パスワード入力のプロンプトは passwd コマンドが表示しています

きません。

インターフェースの設定ファイルも Debian では /etc/network/interfaces、Fedora では /etc/sysconfig/network-scripts/ifcfg-<インターフェース名> となっており、フォーマットはまったく違います。

2.4 Xsession

startx の man を見ると X の起動時に実行されるスクリプトは /.xinitrc となっています。しかし、Debian では /.xsession に書いておけば startx を実行したときでもグラフィカルログインしたときでも同じ環境にすることができます。

これは、Debian では素の X に対して変更を加えているためです。

startx したときのフローは次のようになっています。

1. /usr/bin/startx
2. /.xinitrc があったら /.xinitrc を実行。なかつたら /etc/X11/xinit/xinitrc (Debian 向けに修正されています) を実行
3. /etc/X11/Xsession を実行

グラフィカルログイン (xdm) した場合のフローは次のようになります。

1. /etc/X11/xdm/xdm-config の DisplayManager*session に書かれたコマンド (通常、/etc/X11/xdm/Xsession (Debian 向けに修正されています)) を実行
2. /etc/X11/Xsession を実行

というわけでどちらの場合も /etc/X11/Xsession が実行されます。この /etc/X11/Xsession は Debian specific なもので /etc/X11/Xsession.d ディレクトリにあるスクリプトを順に実行します。このうち、50x11-common_determine-startup で起動プログラムの検出が行われ、/.xsession がある場合、/.xsession が起動プログラムに選ばれます。99x11-common_start で 50x11-common_determine-startup で選ばれたプログラムが実行されることで /.xsession に書かれた内容が有効になります。

ちなみに、Fedora では /.Xclients に書くと startx でもグラフィカルログインでもスクリプトを実行してくれるようです。ただし、/etc/X11/Xsession.d ディレクトリのような仕組みはないようです。

2.5 lesspipe

less で gzip 圧縮されたファイルを開こうとすると通常次のようになります。

```
$ less hoge.txt.gz
"hoge.txt.gz" may be a binary file. See it anyway?
```

ひょっとしたら上のようなメッセージは表示されずに hoge.txt の内容が表示されている方もいらっしゃるかもしれません。その場合、次の環境変数が設定されているはずです。

```
$ printenv | grep ^LESS
LESS=-M
LESSOPEN=| /usr/bin/lesspipe '%s'
LESSCLOSE=/usr/bin/lesspipe '%s' '%s'
```

less には LESSOPEN という環境変数が設定されているとファイルを読み込む前に LESSOPEN で指定されたコマンドにファイル名を渡してコマンドの標準出力をファイルの内容として読み込む

という機能があります。これにより、gzip 圧縮されたファイルを

```
$ less hoge.txt.gz
```

というコマンドラインで表示することができます。また、この機能を応用することで

```
$ less hoge.tar.gz
```

とした場合に tar されているファイルの一覧を取得するといったこともできます。

/usr/bin/lesspipe は Debian specific なものです。他のディストリビューションにも同様の機能を持つものが含まれていることが多いのですが、Debian の lesspipe は対応している拡張子の数が多いことが特徴です。

language-env を実行すると LESSOPEN の設定をしてくれるため気づかず使っているという方もいらっしゃるのではないでしょうか。

2.6 Debian specific の見つけ方

それが Debian specific であるか知るために man を参照するという方法があります。man を見ると、

```
Debian GNU/Linux      Version 3.97      ADDUSER(8)
```

のように書かれているので Debian specific と推測することができます。しかし、それ以外に Debian specific かを知るよい方法はないようです。

Xsession や lesspipe のようにオリジナルの配布内容からファイルが追加されている場合、ソースパッケージの debian ディレクトリに追加ファイルが格納されています。debian ディレクトリにあるファイルリストから control や postinst などの制御ファイルを除いたものを取得すればそのパッケージに Debian specific な変更がありそうか推測することができると考えられます。

3 翻訳へのさそい

小林さん

3.1 はじめに

国際化は Debian の一つの特徴です。その国際化の達成には、フレームワークの整備から各ソフトウェアの対応、そしてメッセージやドキュメントの翻訳まで、多岐に渡る非常に膨大な作業を必要とします。ここでは、それらの作業のうち、最も大量の作業を必要とする一方で一般ユーザが最も取り組みやすい翻訳について、主に Debian JP まわりで行われている日本語訳作業をまとめます。

3.2 共通の作業用インフラストラクチャ

まず、翻訳作業で共通に使われるインフラストラクチャをまとめて説明します。これらは、後述する各種作業の説明でも頻繁に登場します。

3.2.1 作業用メーリングリスト

翻訳作業に関するやりとりには主にメーリングリストが使われます。Debian 本家のものと Debian JP のものがありますが、どちらについても、翻訳関連のメーリングリストは誰でも (Debian および Debian JP のメンバーでなくても) 自由に参加できます。

日本語訳関連の作業に関するやりとりによく使われるのは、Debian JP の debian-doc^{*14} および debian-www^{*15} メーリングリストです。登録に使うアドレスは、それぞれ debian-doc-ctl@debian.or.jp と debian-www-ctl@debian.or.jp です。これらのメーリングリストに関する情報が、<http://www.debian.or.jp/MailingList.html>^{*16} にあるので、参照してください。過去にこれらのメーリングリストに投稿されたメールのアーカイブは、<http://lists.debian.or.jp/debian-doc/> および <http://lists.debian.or.jp/debian-www/> で完全に公開されています。

さらに、パッケージの更新に伴う debconf-po の翻訳更新の依頼など、Debian 本家の開発者との英語でのやりとりには、主に Debian 本家の debian-japanese メーリングリスト^{*17} が使われます。登録およびアーカイブは <http://lists.debian.org/debian-japanese/> で利用可能です。メーリングリストを経由せずに、前のバージョンの翻訳者や、翻訳者として活発に活動しているかた、あるいは本家で活動している日本人開発者のところに直接メールが行ったりすることも

*14 debian-doc@debian.or.jp

*15 debian-www@debian.or.jp

*16 準備中の新サイトでは <http://www-internal.debian.or.jp/community/ml/>。

*17 debian-japanese@lists.debian.org

あります。

また、ドキュメントの翻訳なら Debian 本家の debian-doc メーリングリスト^{*18}に、ウェブページの翻訳なら Debian 本家の debian-www メーリングリスト^{*19}にそれぞれ登録しておくと、内容に関する質問や間違いの修正などに関するやりとりを本家の方々とできます。それぞれ <http://lists.debian.org/debian-doc/> および <http://lists.debian.org/debian-www/> で、登録やアーカイブ閲覧ができます。

3.2.2 対訳表

日本語訳の対訳表は、Debian JP debian-doc メーリングリストでたまに話題になりますが、なかなか整備までいかないのが現状です。メーリングリストで訳語に関する問い合わせをしたり、査読依頼を出してコメントをもらったりできるので、そこまで気になることはないでしょう。一応、既存のいくつかの対訳表をピントとして示しておきます。

- Debian JP の『略語の解説』: <http://www.debian.or.jp-devel/abbreviation.html>
- Debian JP 対訳表
 - ソース: http://www.debian.or.jp/Documents/trans_table/
 - HTML での出力: http://www.debian.or.jp/Documents/trans_table/trans_table.html
 - dict 形式での出力: http://www.debian.or.jp/Documents/trans_table/trans_table.dict
- APT, dpkg 関連の表記に関する用語集 (武藤健志さんの Wiki): <http://kmuto.jp/open.cgi?DebianGlossary>
- かねこさんによる『Security 関連用語対訳集』: <http://lists.debian.or.jp/debian-www/200607/msg00120.html>
- 小林による Debian Weekly News 関連の ja.po^{*20}: <http://dolphin.c.u-tokyo.ac.jp/~nori1/dwn/ja.po>

対訳表というわけではありませんが、これらの他に小林が訳語の選択によく利用するのは、Google です。Debian のウェブサイト全般から訳語を探したければ「site:www.debian.org」を、Debian Weekly News から探したければ「site:www.debian.org/News/weekly」をつけて検索し、引っ掛けたページを見ながら訳語を決めるということをよくやっています。

3.3 各種ソフトウェアの po や付属ドキュメントの翻訳

3.3.1 作業方法

各種ソフトウェアのメッセージカタログ (po) や付属ドキュメント、manpage などの翻訳に関する議論は、Debian JP の debian-doc メーリングリストで行われています。これらの翻訳はソフトウェアの更新に伴って更新作業をする必要があるので、主に開発元 (upstream) のソフトウェア作者などと (主に英語で) やりとりをしながら作業することになります。しかし、特に Debian と密接に関連したソフトウェアについては訳語が統一されているほうがよいので、訳語選択などについて debian-doc メーリングリストで査読を依頼することが推奨されています。

*18 debian-doc@lists.debian.org

*19 debian-www@lists.debian.org

*20 Debian Weekly News 用 wml ファイルの後半の Security Updates および Removed Packages から用語を抽出して po としたものです。

3.3.2 翻訳状況の確認

poについては、翻訳状況に関する情報が以下のページで得られます。

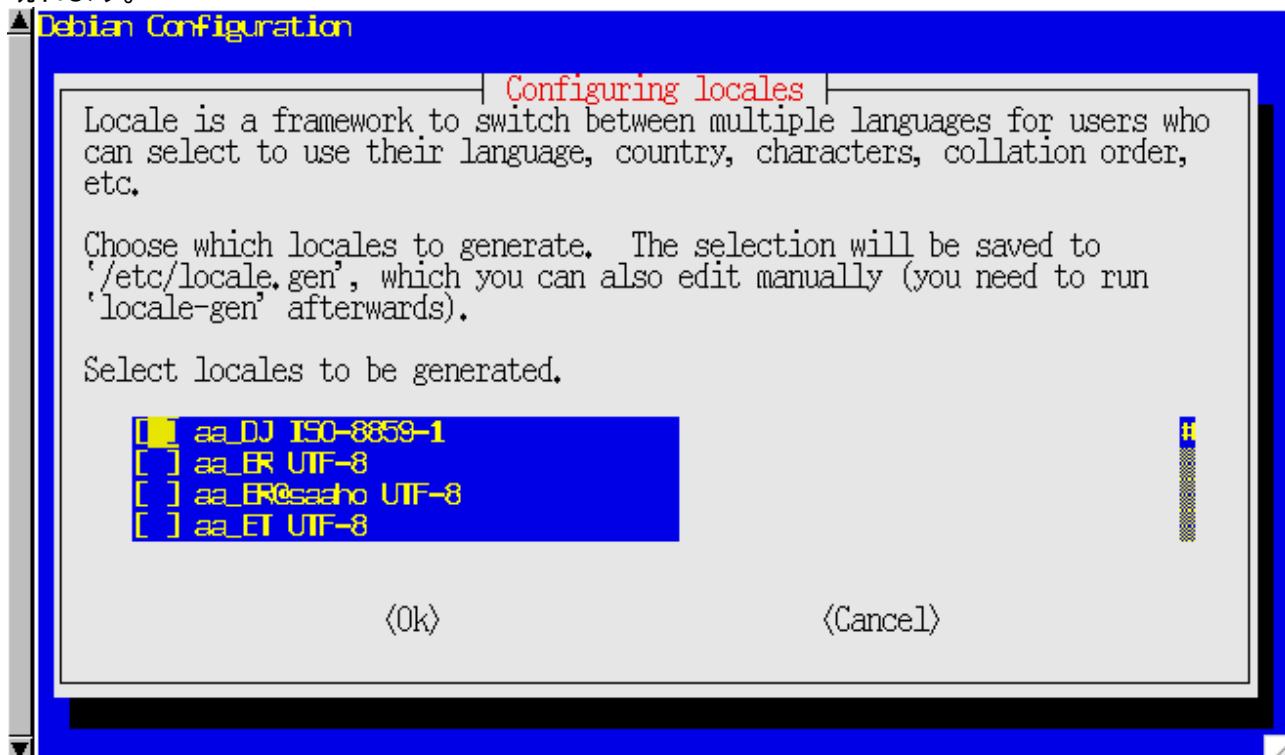
- po ファイル翻訳のページ (<http://www.debian.org/international/l10n/po/>)
 - 日本語の翻訳状況: <http://www.debian.org/international/l10n/po/ja>
 - 言語ごとの翻訳ランキング: <http://www.debian.org/international/l10n/po/rank>
- Debian-Installer の各言語翻訳状況
 - 不安定版 (unstable):
<http://d-i.alioth.debian.org/l10n-stats/translation-status.html>
 - テスト版 (testing):
<http://d-i.alioth.debian.org/l10n-stats/translation-status-testing.html>

3.4 debconf-po 関連

3.4.1 debconf-po とは

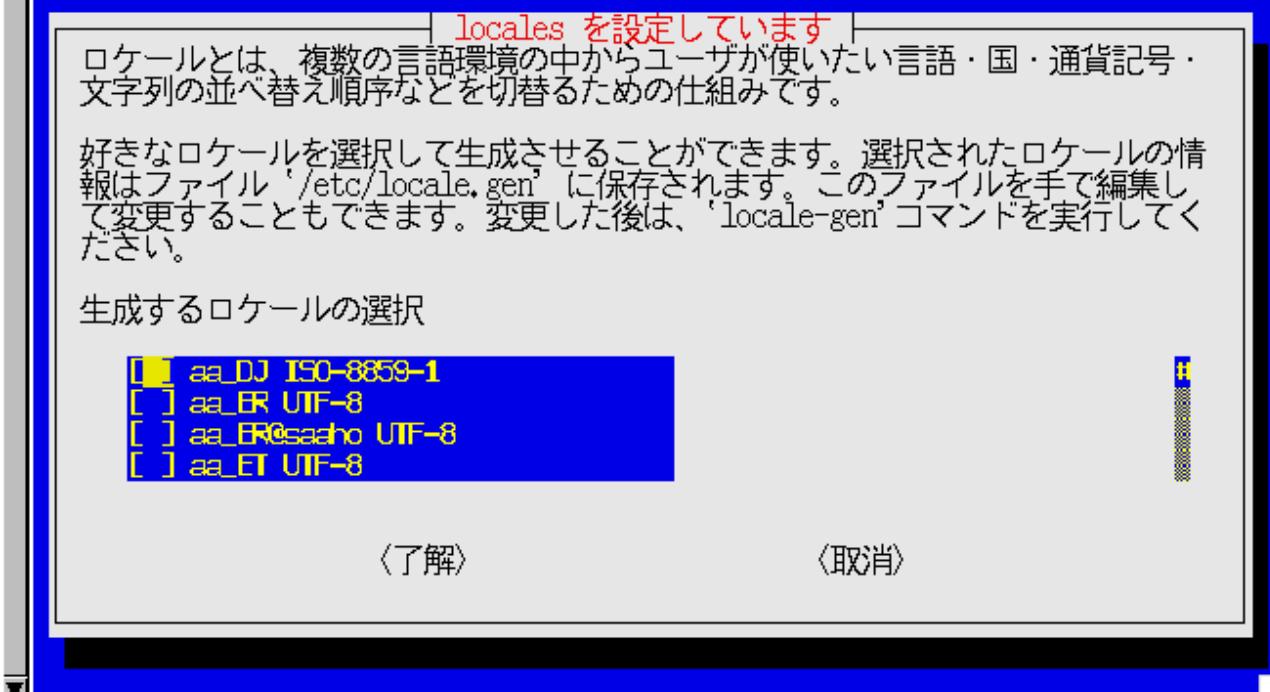
debconf-po とは、Debian パッケージをインストールする際になされる設定関連の質問 (debconf の質問) に翻訳を提供し、localizeされたインターフェースでユーザが質問に答えられるようにするための po ファイルです。

例えば、sarge で locales パッケージを設定する場合、英語のロケールでは次のような画面が現れます。



これに対して日本語のロケールでは次のようにになります。

▲Debian の設定



このようにロケールに応じた質問を提供するのが debconf-po です。

この debconf の質問自体は、次のように、パッケージ作者によって、ソースパッケージの debian ディレクトリの下のバイナリパッケージ用 templates ファイルに英語で書かれています。

```
Template: locales/locales_to_be_generated
Type: multiselect
Choices: ${locales}
_Description: Select locales to be generated.
Locale is a framework to switch between multiple languages for users who can
select to use their language, country, characters, collation order, etc.

Choose which locales to generate. The selection will be saved to
'/etc/locale.gen', which you can also edit manually (you need to run
'locale-gen' afterwards).

Template: locales/default_environment_locale
Type: select
_choices: None, ${locales}
Default: None
_Description: Which locale should be the default in the system environment?
Many packages in Debian use locales to display text in the correct
language for users. You can change the default locale if you're not
a native English speaker.
These choices are based on which locales you have chosen to generate.

Note: This will select the language for your whole system. If you're
running a multi-user system where not all of your users speak the language
of your choice, then they will run into difficulties and you might want
not to set a default locale.
```

オリジナルは英語ですが、非英語圏のユーザにとっては、自分の母語で質問されるほうがよいでしょう。そこで、localize された debconf 質問をユーザが利用できるようにするのが、この debconf-po です。

3.4.2 作業方法

作業を開始する前に、まずは 3.4.3 で述べる翻訳状況調整ページで既に翻訳作業が行われていないか確認するとよいでしょう。その上で翻訳対象とするパッケージを決めたら、<http://www.debian.org/intl/l10n/po-debconf/pot> からそのファイルの templates.pot ファイルをダウンロードします。もちろん、ソースパッケージを手に入れ、目的のバイナリパッケージの templates ファイルに対して po-debconf パッケージの debconf-gettextize コマンドを実行し、

templates.pot を生成してもかまいません。

templates.pot ファイルを取得したら、名前を ja.po に変更した上で翻訳しましょう。

翻訳を終えたら、該当パッケージに対して、severity を「wishlist」、tags を「l10n, patch」としてバグ報告しましょう。ただ、慣れていないうちは Debian JP の debian-doc メーリングリストで查読してもらうことを強くお勧めします。

3.4.3 翻訳状況の確認

debconf-po については、翻訳状況に関する情報が以下のページで得られます。

- debconf-po ファイル翻訳のページ (<http://www.debian.org/international/l10n/po-debconf/>)
 - 日本語の翻訳状況: <http://www.debian.org/international/l10n/po-debconf/ja>
 - 言語ごとの翻訳ランキング: <http://www.debian.org/international/l10n/po-debconf/rank>
- 武藤健志さんが提供する debconf-po 日本語翻訳作業調整ページ (<http://kmuto.jp/debian/po-trans/>) 各パッケージの debconf-po の翻訳状況および最終訳者の情報を得ることができます。

3.5 ウェブページ関連

3.5.1 Debian ウェブページの仕組み

Debian のウェブページは WML というファイル形式を利用しています。WML とはウェブサイトメタ言語 (web site meta language) のことで、Debian では wml パッケージとして提供されています。ここでは詳しくは述べませんが、翻訳が原文に追従できているかの確認などがこの WML の機構を用いて行われている、ということだけ書いておきます。

次の例は、<http://www.debian.org/News/weekly/2006/35/index> のソースとなっている、cvs.debian.org の webwml モジュール^{*21}の、webwml/japanese/News/weekly/2006/35/index.wml です。

```
#use wml::debian::weeklynews::header PUBDATE="2006-08-29"
#SUMMARY="Firmware, FrOSCon, Events, Cuba, Translations, GIT, Sarge,
#Etch"
#use wml::debian::translation-check translation="1.8"

<p>Welcome to this year's 35th issue of DWN, the weekly newsletter for the
Debian community. Bug squashing parties have been announced for September 8th
to 10th in <a
href="http://lists.debian.org/debian-devel-announce/2006/08/msg00012.html">Vienna</a> and for September 15th to 17th in <a
href="http://lists.debian.org/debian-devel-announce/2006/08/msg00013.html">J&uuml;lich</a>, Germany. OSDir has taken <a
href="http://shots.osdir.com/slideshows/slideshow.php?release=724&slide=2">Debian installer</a>. Petr Stehlik <a
href="http://lists.debian.org/debian-68k/2006/08/msg00234.html">reported</a>
that the installation of <a href="$(HOME)/releases/sarge/">sarge</a> and <a
href="$(HOME)/releases/etch/">etch</a> worked flawlessly in the recently <a
href="http://lists.debian.org/debian-68k/2006/08/msg00226.html">fixed</a>
version of <a href="http://packages.debian.org/aranyM">ARAnyM</a>, a 32bit
Atari ST/TT/Falcon virtual machine.</p>

[snip]

#use wml::debian::weeklynews::footer editor="Sebastian Feltel, Mohammed
Adn&egrave;ne Trojette, Tobias Toedter, Martin 'Joey' Schulze"
```

このうち#で始まる行が WML の命令です。例えば、

```
#use wml::debian::translation-check translation="1.8"
```

^{*21} <http://cvs.debian.org/?root=webwml>

という行は、原文 (`webwml/english/News/weekly/2006/35/index.wml`) の r1.8 に基づいているという意味です。

3.5.2 作業方法

翻訳作業を開始するには、目的のページの WML ファイル入手する必要があります。CVS を使い慣れている場合は、コマンドラインから日本語のツリー (`webwml/japanese`) と英語のツリー (`webwml/english`) をチェックアウトするとよいでしょう。CVS を使い慣れていない場合は `http://cvs.debian.org/?root=webwml` からリポジトリビュア ViewCVS を使って英語または日本語の目的のファイルをダウンロードしましょう。ただし、この方法では後述する latin-1 文字の置換作業ができないため、ページ内に latin-1 文字が含まれていた場合には自分で何とかして対処しなければなりません。したがって、こちらはあまりお勧めしません。

新規翻訳の場合、英語のファイルを取得したら、まずはそれを日本語訳用に変換しなければなりません。それには `webwml/copypage.pl` を用いて次のように実行します。

```
nori1[6:12]% DWWW_LANG=japanese ./copypage.pl english/News/weekly/2006/37/index.wml
Unable to open language.conf. Using environment variables...
Processing english/News/weekly/2006/37/index.wml...
Destination directory japanese/News/weekly/2006/37/ does not exist,
Copied News/weekly/2006/37/index.wml, remember to edit japanese/News/weekly/2006/37/index.wml
```

こうすると、オリジナルのファイルのリビジョンを元にして、前述の `wml::debian::translation-check translation` の値が適切に設定されます。また、latin-1 でエンコードされた文字列があっても適切な文字実体参照に置換され、日本語の文字と欧米の文字が共存できるようになります。あとは自由に翻訳してください。

新規翻訳ではなく、翻訳が古くなったページの更新であれば、英語のページを日本語用にコピーする必要はありません。`wml::debian::translation-check translation` の値を適切に設定し、原文の差分を見ながら翻訳を更新しましょう。

翻訳の際のルールについては、<http://www.debian.or.jp/devel/www/WebTranslation.html> を参照するとよいでしょう。また、ウェブページは Mozilla Firefox のような GUI のウェブブラウザでも w3m のようなテキストブラウザでも美しく見えてほしいので、改行位置には気をつけることになっています。<http://lists.debian.or.jp/debian-www/200408/msg00046.html> や <http://lists.debian.or.jp/debian-www/200609/msg00102.html> などを参考にしてください。

翻訳が終わったら、Debian JP の `debian-www` メーリングリストに査読・コミット依頼を出します。現在はコミットは主に今井伸広さんがしてくださっています。

3.5.3 翻訳状況の確認

ウェブページについては、翻訳状況に関する情報が以下のページで得られます。

ウェブサイト翻訳状況 (<http://www.debian.org/devel/website/stats/>) 言語ごとの翻訳状況の統計が載っています。

日本語のウェブサイト翻訳状況 (<http://www.debian.org/devel/website/stats/ja.html>) 日本語の各ファイルの翻訳状況がわかります。

3.6 The Debian Description Translation Project (DDTP)

3.6.1 DDTP とは

Debian Description Translation Project (DDTP) とは、現在すべて英語で提供されている Debian パッケージ説明文 (Description) に翻訳を提供し、それらの翻訳情報が使えるインフラを整えようというプロジェクトです。 <http://ddtp.debian.net/> がプロジェクトのウェブサイトです。

おそらく皆さん御存知でしょうが、パッケージ説明文とはパッケージ付隨情報の一つで、パッケージの内容を説明するとともに、`aptitude search` などでパッケージを検索する際に便利になるよう提供されています。以下は、`sarge` で `aptitude` パッケージの情報を表示させたときの様子で、「詳細:」で始まる行^{*22}以降がパッケージ説明文です。

```
norii[12:04] aptitude show aptitude          whale:~/svnwc/deb/skkdir/trunk
パッケージ: aptitude
ステータス: インストール済み
自動的にインストールされる: no
バージョン: 0.2.15.9-6bpo3
優先度: 任意
分類: admin
保守担当者: Daniel Burrows <dburrows@debian.org>
展開サイズ: 5288k
依存: libapt-pkg-libc6.3-5-3.11, libc6 (>= 2.3.2.ds1-21), libgcc1 (>=
 1:3.4.1-3), libncurses5 (>= 5.4-1), libsigc++-1.2-5c102, libstdc++5 (>=
 1:3.3.4-1)
提案: aptitude-doc-en | aptitude-doc
詳細: terminal-based apt frontend
aptitude is a terminal-based apt frontend with a number of useful features,
including: a mutt-like syntax for matching packages in a flexible manner,
dselect-like persistence of user actions, the ability to retrieve and display
the Debian changelog of most packages, and extreme flexibility and
customization.

aptitude is also Y2K-compliant, non-fattening, naturally cleansing, and
housebroken.
```

このパッケージ説明文のエントリ自体は、次のように、パッケージ作者によって、ソースパッケージの `debian/control` ファイルに英語で書かれています。

```
[snip]
Description: terminal-based apt frontend
aptitude is a terminal-based apt frontend with a number of useful
features, including: a mutt-like syntax for matching packages in a
flexible manner, dselect-like persistence of user actions, the
ability to retrieve and display the Debian changelog of most
packages, and extreme flexibility and customization.

.
aptitude is also Y2K-compliant, non-fattening, naturally cleansing,
and housebroken.
[snip]
```

説明文は、`Description:`と同じ行に書かれる `short description` と、その後の `long description` に分かれます。

オリジナルは英語ですが、非英語圏のユーザにとっては、自分の母語でパッケージ説明文を読めるほうがよいでしょう。そこで、localizeされたパッケージ説明文をユーザが利用できるようにすることを目的として作られたのがこの DDTP というプロジェクトです。

このプロジェクトは数年前から存在しており、日本語についても日本語チームコーディネータの田村一平さんなどが積極的に作業を進め、一時は翻訳率でトップになったこともありました^{*23}が、Debian のホストの問題で暫く停止していました。完全ではありませんが、最近ようやく復活の兆しが見え始めました^{*24}。最近は以下のような状況です。

*22 `aptitude` 0.4.2 以降では「説明文:」という訛語に変わっています。

*23 <http://d.hatena.ne.jp/denson/20050315/p2>

*24 <http://www.debian.org/News/weekly/2006/31/> や <http://www.debian.org/News/weekly/2006/35/> に関連記事があります。
<http://lists.debian.org/debian-devel/2006/07/msg01323.html> から始まる “Translated packages descriptions progress” というスレッドも盛り上がってきました。

- Debian 本家のウェブサイトにも DDTP のページ^{*25}が作られた。
- プロジェクトウェブサイトが復活し、翻訳状況を見られるようになった。
- メールインタフェース（後述）が復活した。
- ウェブインタフェース（後述）が作られた。

3.6.2 作業方法

DDTP の作業方法については、Debian JP のサイトに日本語の説明があります^{*26}。ただしこれは復活前のもので情報がやや古くなっているので、最近作られた Debian 本家のウェブサイトの DDTP のページ^{*27}を参照するのがよいでしょう。このページは本資料執筆現在は英語でしか利用できないので、ここでは本家の説明に基づいて、簡単に説明します^{*28}。

メールインタフェース

DDTP は、誰でも気軽に作業できるよう、非常に簡単なインタフェースを通じて作業できるようになっています。現在 Debian パッケージ数は 15000 を超えており、debconf とは異なりパッケージ説明文はすべてのパッケージに含まれているので、それらをすべて localize しようという目的をもつこのプロジェクトは、とても壮大で大量のマンパワーを必要とするからです。新しいパッケージでパッケージ説明文が改良されることがあるので、それらの変化にも追従できなければいけません。

公式インタフェースはメールで、それ以外にもウェブインタフェースが開発されています。

メールインタフェースを使うには、次のような件名 (Subject) で pdesc@ddtp.debian.net にメールを送ってください。

GET n lang

n はパッケージ説明文の数で、9 以下の数値を指定してください。lang は言語コードで、日本語では ja です。lang の後ろにドット (.) に繋げてエンコーディングを指定することも可能です。

メールを送ると、指定された数だけパッケージ説明文が添付されたメールが返信されます。これらのパッケージ説明文はしばらくの時間ロックされ、メールで取り寄せた人のみが作業できるようになるので、安心してゆっくり作業しましょう。各添付ファイルは次のような形式になっているので、パッケージ説明文中の<trans>と記された部分を翻訳してください。

```
# Source: aolserver4-nsopenssl
# Package(s): aolserver4-nsopenssl
# Prioritize: 45
# This Description is active
# This Description is owned
Description: AOLserver 4 module: module for SSL mode.
This module adds SSL capabilities to aolserver, and gives Tcl scripts
an API to access openssl functions.

.
This is currently a beta release! Use at your own risk.
Description-ja.euc-jp: <trans>
<trans>
.
<trans>
#
# other Descriptions of the aolserver4-nsopenssl package with a translation in ja:
#
```

*25 <http://www.debian.org/international/110n/ddtp>

*26 <http://www.debian.or.jp/devel/doc/Description-ja.html>

*27 <http://www.debian.org/international/110n/ddtp>

*28 本家の DDTP のページは近々日本語で利用可能になる予定です。今後は Debian JP のページではなくそちらのページをメインの日本語情報として参照するとよいでしょう。

翻訳するのは、<trans>だけです。英語のパッケージ説明文は変更しないでください^{*29}。また、ドットだけの行も段落のセパレータとして重要なので、変更を加えないでください。ただし、上の例では short description と long description の各段落の内容がすべて<trans>となっていますが、一部の段落に既に翻訳が入っている場合もあります。それは、他のパッケージに同じ段落が含まれておりそれが翻訳済みの場合です。これらは修正してもかまいません。

エンコーディングは正しいものを用いるよう注意してください。例えば、GET 1 ja という件名でパッケージ説明文を取り寄せるとき、GET 1 ja.euc-jp noguide という件名のメールが返ってきます。これは、日本語のデフォルトエンコーディングが EUC-JP となっているからです。この場合、翻訳したファイルのエンコーディングは EUC-JP とし、メールで送る際にも ISO-2022-JP で送ってしまわないよう気をつけてください。ただし、上の例の翻訳部のフィールドが Description-ja.euc-jp となっていることからもわかるように、エンコーディング指定は変更可能です。UTF-8 がいいというのであれば、フィールド名を Description-ja.UTF-8 として翻訳文字列を UTF-8 で記入してください。

翻訳を終えたらファイルを pdesc@ddtp.debian.net に送り返します。翻訳文は base64 エンコードするとよいでしょう。中野武雄さん作の ddts-send^{*30} や ddts パッケージ^{*31}などのヘルパーも利用可能です。

ウェブインターフェース

Martijn van Oosterhout さんが作成したウェブインターフェースは DDTSS と呼ばれ、<http://kleptog.org/cgi-bin/ddtss2-cgi/xx> にあります。翻訳と査読・校正の作業をウェブで簡単に行えるようになっています。

3.6.3 翻訳状況の確認

DDTP では、翻訳状況の確認は以下のページでできます。

プロジェクトウェブサイト (<http://ddtp.debian.net/>) トップページには言語ごとの翻訳状況の統計が載っています。パッケージごとに各言語への翻訳状況を表示することも可能です。

3.6.4 翻訳内容の取得

DDTP によるパッケージ説明文の翻訳は、Debian のミラーの <http://ftp.jp.debian.org/debian/dists/sid/main/i18n/> などから取得可能です。例えば日本語なら、上記のディレクトリの Translation-ja.gz や Translation-ja.bz2 が利用できます。

3.7 おわりに

本節では、国際化の一環として重要な翻訳について、作業方法および各種情報を取得できるページをざっと説明しました。翻訳は非常に手間がかかる作業で、膨大なマンパワーを必要とします。Debian ではあなたの参加を心待ちにしています。

^{*29} 誤りを見たらそのパッケージのバグとして BTS にバグ報告してください。

^{*30} <http://surf.ap.seikei.ac.jp/~nakano/linux/ddts-send.ja.html>

^{*31} <http://packages.debian.org/ddtc>

3.8 参考文献

以下のページも参考にしてください。

- 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>

4 dpkg, apt のプロファイリング

上川

apt や dpkg のどの部分が一番遅いのか、実際にプロファイリングしてみます。この例をケーススタディーとして、一般的にどういう作業をすればパフォーマンスチューニングが必要な部分を抽出できるのか、をあきらかにしてみましょう。

4.1 oprofile のインストールと設定方法

Debian のデフォルトのカーネルは oprofile をサポートしています^{*32}。もし、自分でコンパイルしていたりして oprofile サポートを追加していない場合は、カーネルを oprofile サポート付きでコンパイルしなおします。オプションは CONFIG_OPROFILE です。メニューでは

Instrumentation support : Profiling Support : Oprofile system profiling (experimental)^{*33}

にあります。

カーネルがサポートしている場合、oprofile を利用するのに追加で必要なのは oprofile パッケージです。apt-get install oprofile でインストールしましょう。

カーネルのシンボルのプロファイリング^{*34}をするために、vmlinux ファイルが必要です。カーネルを自分でコンパイルした場合には、ビルドしたディレクトリに vmlinux ファイルがあります。make-kpkg を利用してビルドしたのであれば、

```
/lib/modules/$(uname -r)/build
```

から適切にリンクがはられているはずです。探してみてください。^{*35}

4.2 oprofile が自分の利用している CPU をサポートしていない場合

残念ながら 9 月現在時点の Debian Package では、Intel core duo CPU 上では oprofile が動作しません。oprofile は認識できていない場合、cpu_type 変数が unset という値になります。カーネル側は cpu_type として i386/core を出力しているので、この時点でどうやらカーネル側のサポートは追加されているらしいということがわかります。

*32 i386, amd64 などのアーキテクチャ以外での利用は現時点では難しい可能性があるので確認してください。

*33 2.6.18-rc1 現在

*34 無い場合はカーネルの内部のどこかで実行していることはわかるが、実際どの関数で時間がかかっているのか、ということがわからない。

*35 oprofile メーリングリストには vmlinux ファイルよりは普及している System.map を利用するパッチというのも存在するので、それを適用してみるのもよいかかもしれません。

```

$ sudo opcontrol --init
cpu_type 'unset' is not valid
$ opcontrol --list-events
Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
$ cat /dev/oprofile/cpu_type
i386/core
$ uname -a
Linux coreduo 2.6.18-rc1dancer #2 SMP Sun Jul 9 09:57:01 JST 2006 i686 GNU/Linux

```

プロファイルを取得するという目的を考えると手段としてはいくつか考えられます。

- プロファイルの仕組はあまりかわらないだろうと見込み、arch/i386/oprofile/nmi_int.c の ppro_init を修正、piii とかに見せてしまう
- まじめに oprofile のユーザ空間アプリケーションを修正、core duo の仕様書を読み、対応を追加
- おそらくすでに修正されていることを見越して、oprofile の CVS レポジトリをみにいく
- 実験することが目的なのでサポートされている CPU のマシンを準備する

今回は oprofile の開発メーリングリストを見たところ、5月の時点でだれかがパッチを書いているのを発見したので、それをとりこみます。念のため、今後作業する人のために BTS にも登録しました (Bug#380462^{*36})。

確認してみると、どうやら動作してくれていることがわかりました。ここで、当面重要なのは、CPU_CLK_UNHALTED でしょう。CPU サイクルがどの関数で消費されているのかということをトラッキングできます。まず CPU の処理負荷がかかっている部分を目視して、何か問題がないか眺めてみて、何も問題なく、それなりに問題が追求できにくくなつた後に、L2 キャッシュのイベントの発生度合とかを確認していけばよいでしょう。

```

$ sudo opcontrol --init
$ sudo opcontrol --list-events
oprofile: available events for CPU type "Core Solo / Duo"

See Intel Architecture Developer's Manual Volume 3, Appendix A and
Intel Architecture Optimization Reference Manual (730795-001)

CPU_CLK_UNHALTED: (counter: all)
    Unhalted clock cycles (min count: 6000)
    Unit masks (default 0x0)
    -----
    0x00: Unhalted core cycles
    0x01: Unhalted bus cycles
    0x02: Unhalted bus cycles of this core while the other core is halted
INST_RETIRE: (counter: all)
    number of instructions retired (min count: 6000)
L2_RQSTS: (counter: all)
    number of L2 requests (min count: 6000)
    Unit masks (default 0xf)
    -----
    0x08: (M)odified cache state
    0x04: (E)xclusive cache state
    0x02: (S)hared cache state
    0x01: (I)nvalid cache state
    0x0f: All cache states
    0x10: HW prefetched line only
    0x20: all prefetched line w/o regarding mask 0x10.

```

[省略]

4.3 デバッグシンボルを収集する : dpkg と apt をコンパイルしなおす

まず、デバッグ情報がすでにあるパッケージについては、インストールします。今回では、大きいものとして、libc6-dbg パッケージがあるので、それはインストールします。プロファイルの結果、イメージが上位に出現するなどで、必要そうであれば、あとで他のライブラリなどについてもデバッグ情報のあるバージョンを追加しましょう。

^{*36} <http://bugs.debian.org/380462>

今回プロファイル対象の dpkg と apt はデフォルトではデバッグ情報がありません、プロファイル出力を確認しやすいうように、デバッグシンボルを追加してコンパイルしなおします。

```
$ debuild -e DEB_BUILD_OPTIONS=nostrip
```

その後、インストールします。

まず、oprofile を実行するのを便利にするために、スクリプトを仕込みます。入力されたコマンドを 10 回実行してそのプロファイルを取得するというものです。

```
read CMD
sudo opcontrol --shutdown
sudo opcontrol --reset
sudo opcontrol --setup \
--vmlinux=/lib/modules/$(uname -r)/build/vmlinux \
--event=CPU_CLK_UNHALTED:180000:0:1:1 --separate=library
sudo opcontrol --start
for A in $(seq 1 10); do
$CMD
done
opcontrol --dump && \
opreport -l -p /lib/modules/$(uname -r)/kernel 2>/dev/null \
| head -30
```

まず、デバッグ用のバイナリが正常に作成できているか簡単に確認します。まず、apt-get update をループでまわしてみます。libapt-pkg のシンボルレベルで確認できているので、デバッグシンボルが存在しているということがわかります。

```
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)
160 0.3113 vmlinux vmlinux page_fault
158 0.3074 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
SHA1Summation::Add(unsigned char const*, unsigned long)
140 0.2724 libstdc++.so.6.0.8 apt-get (no symbols)
134 0.2607 vmlinux vmlinux find_get_page
125 0.2432 ld-2.3.6.so http do_lookup_x
123 0.2393 vmlinux vmlinux sysenter_past_esp
95 0.1848 libapt-pkg-libc6.3-6.so.3.11.0 apt-get .plt
94 0.1829 ld-2.3.6.so gpgv do_lookup_x
89 0.1732 libc-2.3.6.so http (no symbols)
89 0.1732 vmlinux vmlinux do_generic_mapping_read
88 0.1712 ld-2.3.6.so file do_lookup_x
87 0.1693 vmlinux vmlinux memcpy
72 0.1401 ld-2.3.6.so http strcmp
69 0.1343 vmlinux vmlinux _spin_lock
65 0.1265 vmlinux vmlinux vfs_read
64 0.1245 ld-2.3.6.so gpgv _dl_elf_hash
62 0.1206 vmlinux vmlinux __handle_mm_fault
60 0.1167 ld-2.3.6.so http _dl_elf_hash
58 0.1128 oprofiled oprofiled (no symbols)
```

dpkg についてもプロファイルリングしてみます。^{*37}

*37 ここで問題が出ました。libc6 の dbg パッケージの情報を oprofile が処理できていないようです。strace で解析してみましたが、ファイルをひらくところまでは何かできているようです。これは別途バグ報告してみます (Bug#385704^{*38})。

```

sudo dpkg -i ./dselect_1.13.22_i386.deb
[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
41009 32.6009 libc-2.3.6.so dpkg (no symbols)
27485 21.8497 processor.ko processor acpi_processor_idle
14863 11.8156 dpkg dpkg parsedb
4845 3.8516 dpkg dpkg findnamemode
2691 2.1393 dpkg dpkg findpackage
1994 1.5852 dpkg dpkg f_dependency
1742 1.3848 vmlinux vmlinux get_page_from_freelist
1660 1.3196 dpkg-deb dpkg-deb inflate_fast
1552 1.2338 dpkg dpkg iterpkgnext
1520 1.2084 dpkg dpkg .plt
1500 1.1925 dpkg dpkg varbufaddbuf
1192 0.9476 dpkg dpkg filesdbinit
1080 0.8586 dpkg dpkg w_dependency
1001 0.7958 dpkg dpkg varbufdependency
988 0.7854 dpkg dpkg nfmalloc
884 0.7028 dpkg dpkg illegal_package_name
849 0.6749 vmlinux vmlinux page_fault
802 0.6376 vmlinux vmlinux __copy_from_user_ll_nocache_nozero
687 0.5461 dpkg dpkg ensure_packagefiles_available
633 0.5032 dpkg dpkg varbufaddc
568 0.4515 dpkg dpkg f_filecharf
516 0.4102 vmlinux vmlinux __copy_to_user_ll
473 0.3760 dpkg dpkg copy_dependency_links
458 0.3641 dpkg dpkg parseversion
427 0.3395 dpkg dpkg ensure_package_clientdata
406 0.3228 dpkg dpkg nfstrsave
384 0.3053 dpkg dpkg varbufrecord

```

パッケージをインストールして削除する、というループを回してみましょう。apt-listbugs と apt-listchanges が含まれており、ruby と python の処理負荷が高いことがわかります。また、libc6 のなかで何か重たい処理をしているのがわかります。

```

sudo apt-get install -y dsh; sudo apt-get remove -y libdshconfig1
[中略]
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
195383 24.3783 processor processor (no symbols)
114285 14.2595 libc-2.3.6.so dpkg (no symbols)
67157 8.3793 libruby1.8.so.1.8.4 ruby1.8 (no symbols)
48893 6.1005 libc-2.3.6.so dpkg-query (no symbols)
41537 5.1826 dpkg dpkg parsedb
30685 3.8286 perl perl (no symbols)
28353 3.5377 dpkg-query dpkg-query parsedb
26135 3.2609 python2.4 python2.4 (no symbols)
13951 1.7407 libc-2.3.6.so ruby1.8 (no symbols)
10023 1.2506 libc-2.3.6.so apt-get (no symbols)
9138 1.1402 dpkg dpkg findnamemode
7914 0.9874 vmlinux vmlinux get_page_from_freelist
7656 0.9553 dpkg dpkg findpackage
5963 0.7440 vmlinux vmlinux read_hpet
5777 0.7208 libc-2.3.6.so perl (no symbols)
5465 0.6819 dpkg dpkg f_dependency
5108 0.6373 dpkg-query dpkg-query findpackage
4525 0.5646 dpkg dpkg filesdbinit
4452 0.5555 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
    pkgDepCache::CheckDep(pkgCache::DepIterator, int,
    pkgCache::PkgIterator&)
4237 0.5287 vmlinux vmlinux page_fault
4182 0.5218 dpkg dpkg .plt
4101 0.5117 dpkg dpkg varbufaddbuf
3874 0.4834 dpkg-query dpkg-query f_dependency
3744 0.4671 dpkg dpkg iterpkgnext
3645 0.4548 libstdc++.so.6.0.8 apt-get (no symbols)
3287 0.4101 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
    pkgProblemResolver::MakeScores()
3277 0.4089 vmlinux vmlinux delay_tsc

```

4.4 テスト環境の作成

テスト用の環境を作成します。今回は chroot 内部で大量の apt-get update, apt-get install と apt-get remove をループで実行してベンチマークをとってみましょう。

dpkg と apt を変更すると最悪システムが動作しなくなるため、テスト用に環境を準備することは大切です。

chroot 内部では、chroot 外部のファイルにアクセスすることができません。そのため、bind-mount をを行い、外部のファイルを中に見せます。中で必要になるファイルとしては、apt/dpkg のデバッグ版、oprofile の修正版パッケージ（あれば）、そして実行中の Linux kernel に対応する

vmlinux ファイルです。

```
$ sudo cowbuilder --login --bindmount $(pwd)
# apt-get install gnupg
# apt-get update
# apt-get install oprofile libc6-dbg
# dpkg -i (bind-mount したところに おいた事前準備した apt/dpkg/oprofile)
# apt-get -y install gnome; apt-get -y remove libglib2.0-0
# unset LD_PRELOAD
# unset COWDANCER_ILLISTFILE
# mount -t oprofilefs nodev /dev/oprofile >/dev/null
```

chroot で調べてみると、perl が一番重たい処理をしているということがわかりました。こりやチューニングしにくいでしょ。依存関係の解決などの処理に時間がかかっているかと仮説をたてていたのですが、特に露骨に目立って負荷の高い関数というのは見付けることはできませんでした。

```
apt-get install -y dsh; apt-get remove -y libdshconfig1
[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
51258  32.2132 processor        processor      (no symbols)
7929   4.9830 libc-2.3.6.so    apt-get       (no symbols)
7321   4.6009 libc-2.3.6.so    dpkg         (no symbols)
5239   3.2925 vmlinuz        vmlinuz       read_hpet
4377   2.7507 libc-2.3.6.so    perl         (no symbols)
4282   2.6910 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  pkgDepCache::CheckDep(pkgCache::DepIterator, int,
  pkgCache::PkgIterator&)
2888   1.8150 libstdc++.so.6.0.8 apt-get      (no symbols)
2570   1.6151 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  debVersioningSystem::CmpFragment(char const*, char const*, char const*,
  char const*)
2548   1.6013 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  pkgProblemResolver::MakeScores()
2045   1.2852 dpkg           dpkg          parsedb
2019   1.2688 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  debVersioningSystem::DoCmpVersion(char const*, char const*, char
  const*, char const*)
1722   1.0822 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  pkgDepCache::Update(OpProgress*)
1571   0.9873 dpkg           dpkg          findnamenode
1558   0.9791 perl          perl         Perl_sv_gets
1461   0.9182 perl          perl         Perl_yyparse
1408   0.8849 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  debVersioningSystem::CheckDep(char const*, int, char const*)
1222   0.7680 vmlinuz        vmlinuz      __copy_to_user_ll
1181   0.7422 vmlinuz        vmlinuz      get_page_from_freelist
1129   0.7095 perl          perl         S_hv_fetch_common
1055   0.6630 perl          perl         Perl_yylex
1054   0.6624 ldconfig      ldconfig      (no symbols)
1051   0.6605 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  OpProgress::CheckChange(float)
1050   0.6599 vmlinuz        vmlinuz      page_fault
911    0.5725 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  pkgPolicy::GetCandidateVer(pkgCache::PkgIterator)
816    0.5128 dpkg           dpkg          filesdbinit
815    0.5122 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
  pkgDepCache::DependencyState(pkgCache::DepIterator&)
793    0.4984 libapt-pkg-libc6.3-6.so.3.11.0 apt-get      .plt
```

まず、opreport の結果を確認します。カーネル空間で 16%, apt-get で 10%, perl で 7% であることがわかります。ここで重要なのは、この時点で dpkg をチューニングしても大して結果に反映しなさそうだということが明確になったことです。

```

# oreport
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
CPU_CLK_UNHALT...|
samples| %|
-----
182440 51.3356 processor
59664 16.7885 vmlinux
38517 10.8380 apt-get
CPU_CLK_UNHALT...|
samples| %|
-----
25578 66.4070 libapt-pkg-libc6.3-6.so.3.11.0
7929 20.5857 libc-2.3.6.so
2888 7.4980 libstdc++.so.6.0.8
1701 4.4162 apt-get
381 0.9892 ld-2.3.6.so
12 0.0312 anon (tgid:31689 range:0xb7f47000-0xb7f48000)
11 0.0286 anon (tgid:31917 range:0xb7f49000-0xb7f4a000)
9 0.0234 anon (tgid:31841 range:0xb7fc000-0xb7fcc000)
8 0.0208 anon (tgid:31765 range:0xb7f9d000-0xb7f9e000)
27091 7.6230 perl
CPU_CLK_UNHALT...|
samples| %|
-----
22216 82.0051 perl
4377 16.1567 libc-2.3.6.so
273 1.0077 libpthread-2.3.6.so
214 0.7899 ld-2.3.6.so
3 0.0111 libnss_compat-2.3.6.so
2 0.0074 libdl-2.3.6.so
2 0.0074 Fcntl.so
1 0.0037 libnss_files-2.3.6.so
1 0.0037 libnss_nis-2.3.6.so
1 0.0037 IO.so
1 0.0037 gettext.so
23916 6.7296 oreport
CPU_CLK_UNHALT...|
samples| %|
-----
14091 58.9187 oreport
5445 22.7672 libc-2.3.6.so
4119 17.2228 libstdc++.so.6.0.8
257 1.0746 ld-2.3.6.so
3 0.0125 libgcc_s.so.1
1 0.0042 libpopt.so.0.0.0
16010 4.5049 dpkg
CPU_CLK_UNHALT...|
samples| %|
-----
8611 53.7851 dpkg
7321 45.7277 libc-2.3.6.so
74 0.4622 ld-2.3.6.so

```

この後試行錯誤して apt-get update の処理についてはチューニングできそうだ、ということがわかったのですが、それはまた別の機会に。

4.5 まとめ

この文章では Debian で oprofile を利用してボトルネックを検出する作業をするための手順についてまとめました。

4.6 参考文献

- rpm のプロファイルリング <https://www.redhat.com/magazine/012oct05/features/oprofile/>

5 apt を最適化してみる

上川

前回まで、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、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 vmlinu x vmlinu x __copy_to_user_ll
503 0.9787 gpgv gpgv (no symbols)
222 0.4319 vmlinu x vmlinu x 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あり、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 にあります。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 実装でどちらか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。

6 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($39^{\circ}28'15.72''$ N/ $6^{\circ}22'18.87''$ W) CREOFONTE

6.1 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³⁹が集まり、3 日間

^{*39} <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

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。

- 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。

6.2 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^{*40}。

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。
^{*40} 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。

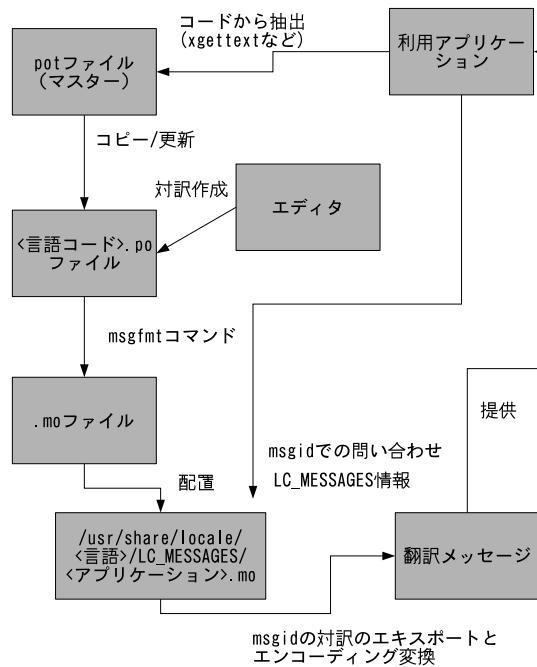


図 2 GNU gettext の仕組みの概略

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。

```

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

# plural 指示で单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 インターフェイスインターフェースの翻訳には 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）。

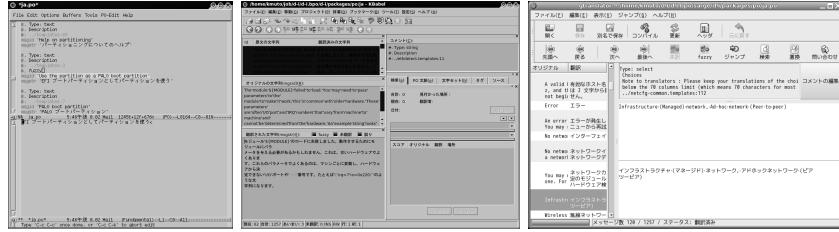


図 3 Emacs po-mode、KBabel、Gtranslator

6.3 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^{*41} 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 形式を中心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 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ません

^{*41} 「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.」

い⁴²。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ざ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あり、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。

6.4 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

*42 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。

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)、現時点では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^{*43}。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)。

^{*43} 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。

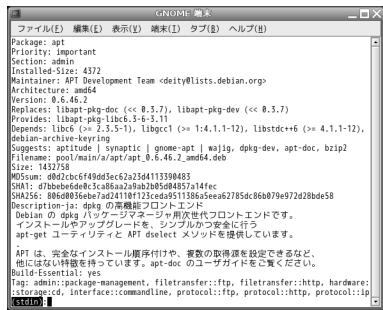


図 4 DDTP から取得した日本語メッセージ

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 ライブリリース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。

6.5 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な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）。

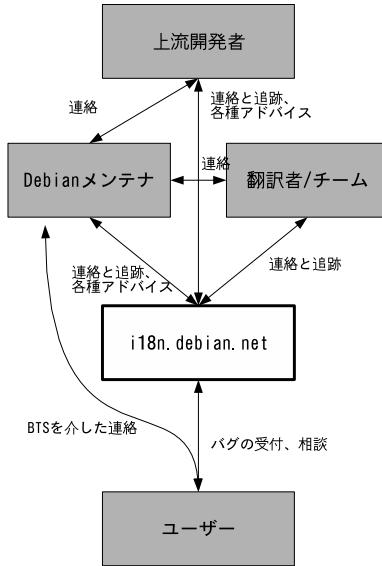


図 5 i18n タスクフォース

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^{*44}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。

6.6 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^{*45}。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



^{*44} NMU campaign for pending l10n bugs (<http://people.debian.org/~lwall/i18n/>)

^{*45} 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。

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。

本会議のプ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。

6.6.1 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。付隨して、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・韓国語と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。

6.6.2 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。

6.7 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^{*46}。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、ヒンディー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、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

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

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。

6.8 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になった。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

7 Debian で Flash したい

松山

7.1 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 が提供されているので、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、いくつ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。

7.2 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 (Bug#166973^{*47})。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。

7.3 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

*47 <http://bugs.debian.org/166973>

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。^{*48}

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			

^{*48} 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。

8 rpmstrap を活用する

岩松

8.1 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。

8.2 rpmstrap とは？

Debian では chroot 環境等を構築するツールとして、debootstrap^{*49} 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^{*50}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。

8.3 インストール

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

でインストールできます。

8.4 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。

8.4.1 とりあえず、chroot 環境を構築してみる

rpmstrap 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。

^{*49} <http://packages.debian.org/unstable/admin/debootstrap>

^{*50} <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 に問い合わせています... 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 に問い合わせています... 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。

8.5 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
```

8.5.1 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
```

8.6 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。

8.7 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 (Scientific Linux 4.02)
- suse10.0 (Suse 10.0)
- tettnang (Fedora Core 2)
- centos4 (Cent OS 4)
- mandriva10 (Mandriva 10)
- sl304 (Scientific Linux 3.04)
- stentz (Fedora Core 4)
- suse9.3 (Suze 9.3)
- yellowdog4 (YellowDog 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^{*51} がサポートされていないようなので、追加してパッチを送りました (Bug#392942^{*52})。

8.8 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い、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し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。

^{*51} <http://www.vininux.org>

^{*52} <http://bugs.debian.org/392942>

テストしたところ、以下のような結果になりました。

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 (Scientific 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 (Scientific Linux 3.04)	可
stentz (Fedora Core 4)	可
suse9.3 (Suze 9.3)	可
yellowdog4 (YellowDog Linux 4.0)	不明

8.9 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。

- 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…。

9 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 は関係ないです。

9.1 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/
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
```

9.2 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

10 パッケージングについて

岩松さん

10.1 パッケージングについての基礎知識

パッケージングとは、無数に存在するソフトウェアの配布したりソフトウェアのバージョンを管理するために、ファイルなどをまとめたものです。それらをサポートするシステムがあり、パッケージ管理システムといいます。

例えば、あるフリーソフトウェアを入手し、使おうとした場合、

```
% ./autogen.sh  
% ./configure  
% make  
# make install
```

として、ライブラリ等をチェック、コンパイル、インストールという作業を行う必要があります。これらの過程でライブラリのチェックで新たにライブラリが必要だったり、必要なライブラリがインストールされていなくてコンパイルできなかったりする事が多々あります。パッケージングシステムを使ってソフトウェアを構成すれば、これらの問題を解決したり、ソフトウェアを容易に使用することができるようになります。

パッケージングシステムの必要な機能として

- パッケージに関する情報の集約
- パッケージの作成
- パッケージのインストール
- パッケージの更新
- パッケージのアンインストール

があります。

ソフトウェアをパッケージングし、インストールやアンインストール、ソフトウェアのアップデートなどを容易に行う事が目的です。

ここで重要なのが、ユーザーだけが容易に使用できるということではなく、開発者側としても容易にパッケージングを行い、パッケージをメンテナンスできるということです。

10.2 Debian でのパッケージ管理/Debian パッケージについて

10.2.1 dpkg

Debian では dpkg と呼ばれる Debian パッケージ管理システムを使用しています。この dpkg を Debian の基礎としているため、Debian で配布されている全てのパッケージは .deb ファイル形式で提供されてなければなりません。

dpkg では

1. 依存関係の解決
 - Depends 依存
 - Recommends 推奨
 - Suggests 提案
 - Conflicts 競合
 - Provides 提供
 - Replaces 置換
2. パッケージバージョンによるアップグレードのサポート
3. インストール前後、アンインストール前後の設定機能

などが提供されています。

10.2.2 Debian Policy

Debian で配布されるパッケージは、厳格なパッケージポリシー Debian Policy^{*53} に基づいて作成されている必要があります。

この Debian Policy によって、パッケージの互換性が保たれています。

10.2.3 パッケージの種類

Debian で配布されるパッケージは、DFSG (Debian Free Software Guideline) によって 3 つのセクションに分けられます。DFSG は Debian として Free Software とはどのようなもののか、定義するものです。

1. main
DFSG に適合したパッケージは main セクションに置かれます。
2. contrib
DFSG に適合しているが、non-free なパッケージに依存しているパッケージは contrib セクションに置かれます。
3. non-free
DFSG に適合しないパッケージは non-free セクションに置かれます。non-free のパッケージは Debian の一部ではありません。

さらにこれらのパッケージを用途に応じて分けられています。

10.2.4 apt

最近は dpkg を直接使い、パッケージをインストールすることは少なくなっています。変わりに APT^{*54}を使うようになりました。その理由として、提供されるパッケージが増え、依存関係が複雑になって必要なパッケージをダウンロードするのが大変になってきたためです。そこで開発されたのが、パッケージをまとめて管理する apt です。dpkg 用のフロントエンドになっており、中では dpkg が呼ばれています。

APT の役目として

^{*53} <http://www.debian.org/doc/debian-policy>

^{*54} Advanced Package Tool

- パッケージの管理
パッケージをどのようにインストール、アンインストールをするか考える。
- パッケージをダウンロードする。
- パッケージの検索を行う。

があります。Debian Policy に基づいて作成されているからこそ実現できています。

10.3 Debian でパッケージを作る際のツール群

Debian でパッケージングを行うためのサポートするソフトウェアがあります。パッケージングを行い、品質の高いパッケージを作るために以下のソフトウェアが提供されています。

10.3.1 dpkg-dev

このパッケージには Debian ソースパッケージを展開、構築、アップロードするために必要なツール群をまとめたものです。ソースパッケージの展開に必要な `dpkg-source` や パッケージの作成に必要な `dpkg-buildpackage` が入っています。

10.3.2 debhelper

`dh_XXX` というパッケージ作成をサポートツールをまとめたものです。Debian 魔窟のひとつです。

10.3.3 devscript

`debuild` などのパッケージ作成フロントエンドが提供されています。

10.3.4 dh-make

ソースパッケージの雛型を作るツールです。ソースパッケージやバイナリパッケージを作成するためには最低限必要なファイルを生成してくれます。`perl` や `php` 用の雛型を作成する `dh-make` も存在します。

10.3.5 lintian

Debian パッケージ用のチェッカーです。Debian Policy にあわせて作られています。`linda` というパッケージ用チェッカーもあります。`lintian` は Perl で、`linda` は Python でプログラミングされたものです。

10.3.6 fakeroot

`fakeroot` は `root` 権限をシミュレートします。パッケージは、`root` の所有権でファイルがインストールされている必要があります。`fakeroot` を使用することによって、`root` にならずにパッケージを構築できます。

10.3.7 cdbs

Common Debian Build System。`dh_XXX`などをまとめて、簡潔にパッケージ作成スクリプトを記述することができるようにするためのソフトウェア。

10.3.8 GnuPG

作成されたパッケージにサインするために使います。そのパッケージがたしかにそのメンテナの PGP key によって作られていることを証明するためです。

10.3.9 dpatch

Debian のソースパッチを管理するツールです。Debian パッケージの差分は *.diff.gz という差分ファイルとして管理されるため、どの部分がどういうパッチであるかということを管理していません。その部分を実装するのが dpatch です。

10.4 実際にパッケージを作成してみる

Debian 用のパッケージを作成する方法を簡単に説明します。今回のパッケージ化の説明で使用するソフトウェアは cairo-dock^{*55} という Mac OS X Dock 風の Dock アプリケーションです。



Debian では^{*56}開発元、オリジナル配布元のことを Upstream といいます。

1. パッケージ化を行うためにパッケージをインストールします。

```
# apt-get install dh-make devscripts debhelper dpkg-dev dpatch
```

2. ソフトウェアのソースコードをダウンロードし、展開します。

```
% wget http://www.gnome-dock.org/prerelease/cairo-dock-0.0.1b.tar.gz
% tar -xzf cairo-dock-0.0.1b.tar.gz
```

3. 展開した後、ディレクトリ名を パッケージ名-パッケージのバージョン になるように修正します。

```
% ls -l
drwxr-xr-x  2 iwamatsu iwamatsu    1024 2006-08-09 01:29 cairo-dock
-rw-r--r--  1 iwamatsu iwamatsu 107560 2006-08-09 01:30 cairo-dock-0.0.1b.tar.gz

% mv cairo-dock cairo-dock-0.0.1b
```

すでに行われている場合は行う必要はありません。

4. バックアップファイルや実行ファイルが残っているので、消しておきます。

```
% ls
Makefile      clock.svg      folder.svg      lowfat.svg      start-cairo-dock.sh~  terminal.svg
cairo-dock   configure.scan  gnome-fs-home.svg movies.svg      sticky-notes.svg  user-home.svg
cairo-dock.c  development.svg im.svg        music.svg      stop.svg       user-trash-full.svg
cairo-dock.c~ editor.svg     lockscreen.svg  search.svg     tango-colors.h   web-browser.svg
chat.svg      email.svg      logout.svg     start-cairo-dock.sh  tango-colors.h~
% make clean
```

5. 対象のソフトウェアのディレクトリに移動し、dh_make -createorig を実行します。

dh_make を実行したときに、パッケージの種類を選択します。

- single binary
ひとつのバイナリパッケージを作成する。
- multiple binary
複数のバイナリパッケージを作成する。

^{*55} <http://www.gnome-dock.org/trac>

^{*56} パッケージを管理している他のディストリビューションでも使われています。

- library
ライブラリ用のパッケージを作成する。
- kernel module
カーネルモジュール用のパッケージを作成する。
- cdb
cdb (Common Debian Build System) を使ったパッケージを作成する。

実行すると、debian ディレクトリが作成されます。--createorig を指定しない場合はオリジナル用のディレクトリ（今回の場合は cairo-dock-0.0.1b.orig ）が作成されません。このディレクトリがない場合は、ソースパッケージの一部として配布される .orig.tar.gz が生成されません。

```
% dh_make --createorig
Type of package: single binary, multiple binary, library, kernel module or cdb?
[s/m/l/k/b] s

Maintainer name : Nobuhiro Iwamatsu
Email-Address   : hemamu@t-base.ne.jp
Date           : Fri, 17 Nov 2006 07:30:18 +0900
Package Name    : cairo-dock
Version         : 0.0.1b
License          : blank
Type of Package : Single
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now. You should also
check that the cairo-dock Makefiles install into $DESTDIR and not in / .
```

6. debian ディレクトリ内を編集します。

dh_make を行ったあとの debian ディレクトリは以下のようになっています。これらはテンプレートファイルなので、パッケージによって必要なものも含まれています。

- README.Debian
Debian 固有の README
- changelog
Debian 固有の変更履歴
- copyright
ソフトウェアのライセンスとコピーライト
- docs
/usr/share/doc にインストールされるファイルのリスト
- emacs-startup.ex
• emacs-install.ex
• emacs-remove.ex
 xemacs 用テンプレート
- postinst.ex
• postrm.ex
• preinst.ex
• prerm.ex
 インストール、アンインストール時に実行されるスクリプトテンプレート
- cairo-dock-default.ex
 /etc/init.d/用のテンプレート
- compat
• cron.d.ex
 crond 用のテンプレート

- init.d.ex
/etc/init.d/用のテンプレート
- rules
パッケージ作成用 Makefile
- cairo-dock.doc-base.EX
docbook 用のテンプレート
- control
パッケージのメタ情報
- dirs
- manpage.xml.ex
- manpage.sgml.ex
- manpage.1.ex
manpages のテンプレート
- menu.ex
menu システム用テンプレート
- watch.ex

7. *.ex および *.EX ファイルを削除します。

今回のソフトウェアのパッケージ化には *.ex や *.EX は必要ないので、削除します。

8. control ファイルの編集

control ファイルを編集します。

• Source

ソースパッケージ名を記述します。今回は ソースの名前から cairo-dock とします。

詳細は Debian-Policy の 5.6.1 Source を参照してください。

• Section

パッケージの種類を指定します。cairo-dock は x11 用のソフトウェアなので、x11 とします。

詳細は Debian-Policy の 2.4 Sections を参照してください。

• Priority

パッケージの優先度を指定します。今回のパッケージは特に重要でもないので、optional を指定します。

詳細は Debian-Policy の 2.5 Priorities を参照してください。

• Maintainer

パッケージメンテナ名とメールアドレスを指定します。私がメンテナンスしますので、私の名前（ローマ字）と連絡用のメールアドレスを記述します。

詳細は Debian-Policy の 5.6.2 Maintainer を参照してください。

• Build-Depends

パッケージをコンパイルするために依存するパッケージを指定します。依存しているパッケージを調べるために configure -h で指定可能なオプションから調べたり、ヘッダファイルやコンパイルエラーメッセージなどから調べるといいでしょう。

詳細は Debian-Policy の 7.6 Relationships between source and binary packages を参照してください。

- Standards-Version
Debian Policy バージョンを指定します。
詳細は Debian-Policy の 5.6.11 Standards-Version を参照してください。
- Package
バイナリパッケージ名を指定します。今回は cairo-dock となります。
詳細は Debian-Policy の 5.6.7 Package を参照してください。
- Architecture
アーキテクチャ依存 (any/各アーキテクチャ)、非依存 (all) の指定を行います。
詳細は Debian-Policy の 5.6.8 Architecture を参照してください。
- Depends,Recommends,Suggests,Conflicts,Provides,Replaces
パッケージの依存関係を記述します。
詳細は Debian-Policy の .6.10 Package interrelationship fields を参照してください。
- Description パッケージの簡単な説明と詳細な説明を記述します。
詳細は Debian-Policy の 5.6.13 Description を参照してください。

以下が今回のパッケージの control ファイルです。

```
Source: cairo-dock
Section: x11
Priority: optional
Maintainer: Nobuhiro Iwamatsu <hemamu@t-base.ne.jp>
Build-Depends: debhelper (>= 5) ,libcairo2-dev ,libgtk2.0-dev ,librsvg2-dev ,libglitz-glx1-dev
Standards-Version: 3.7.2

Package: cairo-dock
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: Dock application like dock of MacOS X
  cairo-dock is dock application like dock of MacOS X.
  This reproduces dock of MacOS X by using Xgl/Compiz and Xcompmgr.
```

9. copyright の変更

Upstream のkopieraito、ソフトウェアのライセンス、パッケージのkopieraito、Upstream のソース取得先を記述します。

Upsteam のkopieraitoはソースに含まれている（場合がある）COPYING ファイルや Web サイトを参照にするといいでしょう。ソフトウェアのライセンスに関しては、ソースに含まれている（場合がある）COPYING ファイルや LICENCE ファイルを参照するといいでしょう。たまにソフトウェアのライセンスが書いていない場合があります。このときは開発者に連絡を取り、ライセンスを確認する必要があります。

Debian-Policy の 12.5 Copyright information を参照してください。

```

Package Maintainers:
Nobuhiro Iwamatsu <hemamu@t-base.ne.jp> from Thu, 9 Nov 2006 00:19:36 +0900.

Upstream Authors:

Mirco "MacSlow" Mueller <macslow@bangang.de>
Behdad Esfahbod <behdad@behdad.org>
David Reveman <davidr@novell.com>
Karl Lattimer <karl@qdh.org.uk>

Upstream Website: <http://www.gnome-dock.org/trac>

Copyright:

Mirco "MacSlow" Mueller <macslow@bangang.de>
Behdad Esfahbod <behdad@behdad.org>
David Reveman <davidr@novell.com>
Karl Lattimer <karl@qdh.org.uk>

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, you can either send email to this
program's maintainer or write to: The Free Software Foundation,
Inc.; 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

On Debian systems, a copy of the license can be found in /usr/share/common-licenses/GPL

```

10. changelog の修正 Debian 特有の変更点を changelog に記録する必要があります。エディタ等で修正することも可能ですが、dch というフロントエンドが用意されていますので、これを使って changelog を編集します。

```

iwanmatsu@chimagu: ~/dev/maintenance/cairo-dock/cairo-dock-0.0.1b
ファイル(E) 編集(E) 表示(Δ) 端末(I) タブ(B) ヘルプ(H)
iwanmatsu@... x iwanmatsu@... x iwanmatsu@... x iwanmatsu@... x iwanmatsu@... x
GNU nano 2.0.0 File: debian/changelog.dch Modified
cairo-dock (0.0.1b-1) unstable; urgency=low
  * Initial release.
  *
-- Nobuhiro Iwamatsu <hemamu@t-base.ne.jp> Sat, 18 Nov 2006 08:56:37 +0900

  ^G Get Help ^O WriteOut ^R Read File ^P Prev Page ^K Cut Text ^C Cur Pos
  ^X Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell

```

11. Upstream のソース変更

そのままのソースコードでは パッケージ化した場合に不都合がある場合があります。パッケージを作成する前に、Debian のパッケージ作成に合うように修正する必要があります。

今回修正した点は以下のとおりです。

- Makefile の install ターゲットがないので追加します。
- Debian で配布されている libcarlio パッケージの cairo-glitz が有効になってないので、Makefile の -DHAVE_GLITZ を無効にします。

変更前

```

APP = cairo-dock

CFLAGS = `pkg-config --cflags cairo gtk+-2.0 librsvg-2.0 glitz-glx` -DHAVE_GLITZ
LDFLAGS = `pkg-config --libs cairo gtk+-2.0 librsvg-2.0 glitz-glx` -lm

SRC = cairo-dock.c

all: $(APP)

clean:
    rm -f *.o *~ $(APP)

```

変更後

```

APP?=cairo-dock
BINDIR?=/usr/bin

CFLAGS = `pkg-config --cflags cairo gtk+-2.0 librsvg-2.0 glitz-glx` -DHAVE_GLITZ
LDFLAGS = `pkg-config --libs cairo gtk+-2.0 librsvg-2.0 glitz-glx` -lm

SRC = cairo-dock.c

all: $(APP)

clean:
    rm -f *.o *~ $(APP)

install:
    install -d ${DESTDIR}${BINDIR}
    install -m 755 cairo-dock ${DESTDIR}${BINDIR}

```

- cairo-dock.c の画像ファイルの指定がプログラムのあるディレクトリになっているので修正します。
- 直接ソース等を修正してもいいのですが、差分で管理するために diff を取り、dpatch で管理します。詳細は Debian 勉強会 2005 年 07 月の資料^{*57}にあります。

10.4.1 manpage の作成

実行権限があるファイルに manpage がない場合作成し、パッケージで提供する必要があります。

12. パッケージの作成

debuild コマンドを使い、パッケージを作成します。

^{*57} debianmeetingresume200507.pdf

```

% debuild
  fakeroot debian/rules clean
dh_testdir
dh_testroot
rm -f build-stamp configure-stamp
# Add here commands to clean up after the build process.
/usr/bin/make clean
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' に入ります
rm -f *.o *~ cairo-dock
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' から出ます
dh_clean
  dpkg-source -b cairo-dock-0.0.1b
dpkg-source: building cairo-dock using existing cairo-dock_0.0.1b.orig.tar.gz
dpkg-source: building cairo-dock in cairo-dock_0.0.1b-1.diff.gz
dpkg-source: building cairo-dock in cairo-dock_0.0.1b-1.dsc
  debian/rules build
dh_testdir
# Add here commands to configure the package.
touch configure-stamp
dh_testdir
# Add here commands to compile the package.
/usr/bin/make
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' に入ります
cc `pkg-config --cflags cairo gtk+-2.0 librsvg-2.0 glitz-glx'
`pkg-config --libs cairo gtk+-2.0 librsvg-2.0 glitz-glx' -lm cairo-dock.c -o cairo-dock
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' から出ます
#docbook-to-man debian/cairo-dock.sgml > cairo-dock.1
touch build-stamp
  fakeroot debian/rules binary
dh_testdir
dh_testroot
dh_clean -k
dh_installdirs
# Add here commands to install the package into debian/cairo-dock.
/usr/bin/make DESTDIR=/tmp/cairo-dock-0.0.1b/debian/cairo-dock install
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' に入ります
install -d /tmp/cairo-dock-0.0.1b/debian/cairo-dock/usr/bin
install -m 755 cairo-dock /tmp/cairo-dock-0.0.1b/debian/cairo-dock/usr/bin
install -d /tmp/cairo-dock-0.0.1b/debian/cairo-dock/usr/share/cairo-dock
install -m 666 *.svg /tmp/cairo-dock-0.0.1b/debian/cairo-dock/usr/share/cairo-dock
make[1]: ディレクトリ '/tmp/cairo-dock-0.0.1b' から出ます
dh_testdir
dh_testroot
dh_installchangelogs
dh_installdocs
dh_installexamples
dh_installman
dh_link
dh_strip
dh_compress
dh_fixperms
dh_installdeb
dh_shlibdeps
dh_gencontrol
dpkg-gencontrol: warning: unknown substitution variable ${misc:Depends}
dh_md5sums
dh_builddeb
dpkg-deb: '../cairo-dock_0.0.1b-1_i386.deb' にパッケージ 'cairo-dock' を構築しています。
dpkg-genchanges
dpkg-genchanges: including full source code in upload
dpkg-buildpackage (debuild emulation): full upload (original source is included)
Now running lintian...
W: cairo-dock: description-synopsis-might-not-be-phrased-properly
W: cairo-dock: wrong-bug-number-in-closes 13:#nnnn
Finished running lintian.
Now signing changes and any dsc files...
  signfile cairo-dock_0.0.1b-1.dsc Nobuhiro Iwamatsu <hemamu@t-base.ne.jp>

次のユーザーの秘密鍵のロックを解除するには
パスフレーズがいります：“ Nobuhiro Iwamatsu <hemamu@t-base.ne.jp> ”
1024 ビット DSA 鍵, ID 3170EBE9 作成日付は 2003-09-29

  signfile cairo-dock_0.0.1b-1_i386.changes Nobuhiro Iwamatsu <hemamu@t-base.ne.jp>

次のユーザーの秘密鍵のロックを解除するには
パスフレーズがいります：“ Nobuhiro Iwamatsu <hemamu@t-base.ne.jp> ”
1024 ビット DSA 鍵, ID 3170EBE9 作成日付は 2003-09-29

  Successfully signed dsc and changes files

```

10.5 パッケージのテスト

パッケージができたから終わりなのではなく、できたパッケージの動作確認やパッケージ段階で不具合がないか、確認する必要があります。

10.5.1 pbuilder でビルドチェック

pbuilder^{*58} は chroot システムを構築し、その中でパッケージのビルドを行うツールです。最低限のユーザーランドの上で、パッケージの依存関係を解決し、パッケージをビルドできるの確認することができます。パッケージの依存関係に不備があった場合はパッケージがビルドできません。パッケージはできたが、実は自分の環境でしかビルドできなかつたという単純な問題を無くするために使用したほうがいいでしょう。

10.5.2 実際にインストールして、動作確認を行う

動作確認していないものを不特定多数の人に配布するのは問題ですので（やってない人もおられるようですが。）実際にインストールして、動作確認を行います。

```
# dpkg -i cairo-dock_0.0.1b-1_i386.deb  
% cairo-dock
```

10.5.3 アンインストールできるか確認する

インストールした際にアンインストールスクリプトに不具合があり、正常にアンインストールできない場合があります。アンインストールもできるか確認します。

10.6 まとめ

Debian パッケージを作成することは難しくなく、容易に作成できる環境は整っています。

肝心なのはツールを使えるようになることではなく、Debian Policy をどれだけ熟読しているか、にかかるくるように思います。

作って分からないうがあれば debian-devel@jp^{*59} 等で聞くといいでしょう。

^{*58} <http://packages.qa.debian.org/p/pbuilder.html>

^{*59} debian-devel@debian.or.jp



11 sid を日常環境として使うための注意

上川

11.1 sid とはなにものか

Debian sid は別名 unstable で、毎日リリースされている Debian の開発版です。開発者が新しいパッケージをリリースしたらまずそこに入ります。毎日日本時間午前 4 時くらいに処理されており、そのタイミングで新しいバージョンが配布されます。

開発者に近い層のユーザは、最新版のパッケージを利用するため unstable を利用します。問題があれば隨時バグ報告をしていきます。また、apt-listbugs などを利用し、他のユーザから報告された深刻なバグがないか確認しながら作業します。

11.2 インストール方法

毎日最新版になるので、安定して直接インストールできる方法というのは基本的には無いでしょう。安定版をインストールしてから、sid にアップグレードするという形が通常のやりかたです。

また、別の方法として、chroot 内部に sid を飼うという方法があります。debootstrap や、cdebootstrap などのツールを利用すると、chroot 内部で Debian sid が稼働している状況をつくれます。pbuilder などのツールを利用するとより便利に利用できます。

sid は最新版を常においかけてるので、深刻なバグのリスクに出会う危険性が常にあります。chroot 内部で常に最新版を確認して、それを外部に展開するというやりかたをしないと危険でしょう。

11.3 魅力

常に最新版が利用できます。開発が起きている場です。開発をしたいだとか、オープンソースの世界の動きを肌で感じたいというのであれば、お薦めです。

11.4 情報源

IRC や ML や勉強会で情報収集しましょう。#debian-devel IRC チャンネルのトピックが一番最新の情報が得られます。

apt-listbugs や apt-listchanges というツールも利用しましょう。apt-listbugs は今インストールしようとしているパッケージのバージョンに該当する深刻なバグレポートを表示してくれるツールです。apt-listchanges は前回インストールしたバージョンからの changelog の差分を表示してくれるツールです。

12 bugreport 論

上川

12.1 Debian BTS の特徴

Debian BTS は、ウェブとメールフロントエンドをもつバグトラッキングシステムです。他のバグトラッキングシステムと違う点として、操作が全てメールでしかできないという点と、情報が全て完全に公開されるという点があります。また、バグレポートをパッケージ単位で分類しているという点も特徴です。

12.2 使われ方

深刻なバグ（RC バグ）を登録すると、そのパッケージのバージョンはリリースできない、という意味になります。各ユーザは apt-listbugs を経由してそのようなバグを確認し、深刻なバグのあるパッケージのバージョンをインストールしないように回避できます。また、この情報はリリースマネージメントにつかわれています。

12.3 アーキテクチャ

バックエンドデータベースはプレーンのテキストファイルです。ファイル構造は下記のようになっています。

- /org/bugs.debian.org/spool
 - incoming/
 - db-h/
 - * 00/
 - ..
 - 314200.log
 - 314200.report
 - 314200.status
 - 314200.summary
 - * ..
 - * 99/
- archive/
 - * 00/
 - * ..

* 99/

- index.db – index.db.realtime へのシンボリックリンク
- index.archive – index.archive.realtime へのシンボリックリンク
- nextnumber

メールを受信したら、そのメールが incoming 以下にスプールされます。そのメールを 15 分に一回 cron から起動されるスクリプトが処理します。

ウェブ経由でバグ報告を確認するための cgi があります。その cgi 経由では BTS は内容の閲覧だけができ、変更はしません。

また、スクリプトなどから利用するために、SOAP のフロントエンドがあります。ドキュメントは一行もありません。

現状 <http://bugs.donarmstrong.com/cgi-bin/soap.cgi> でステージングされています。今後は <http://bugs.debian.org> に移行したいようです。ネームスペースは Debbugs/SOAP/Status で、そこで get_status という関数が定義されています。get_status は複数の数字パラメータをうけ、そのあたえられた数字に対応するバグレポートの情報をかえします。get_status は、バグ報告の本文ではなく、メタデータを返します。

12.4 文化

たまに BSP などの祭があります。バグトラッキングシステムに登録されているバグで最初に対応されないものについては、そのまま放置され時間がだけが過ぎてしまう傾向があります。それらに対して新たな気持ちで対応しようというものです。

12.5 参考文献

- Debian 勉強会 2005 年 10 月資料 「debbugs internal」
- Debian 勉強会 2005 年 10 月資料 「apt-listbugs の生い立ちと実装」
- `/usr/share/doc/debian/bug-maint-mailcontrol.txt` など



13 Debian Weekly News trivia quiz

上川純一

ところで、Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News。毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、意味がわからないところもあるかも知れません。みんなで DWN を読んでみましょう。

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

13.1 2006 年 25 号

<http://www.debian.org/News/weekly/2006/25/> にある 6 月 20 日版です。

問題 1. Isaac Clerencia さんは、スペインのサラゴサ市当局が、6 ヶ所ある場所に Debian ベースのシンクライアントを設置したと報告しました。その場所とはどこでしょうか

- A ラーメン屋
- B コンビニエンスストア
- C 老人ホーム

問題 2. Yaroslav Halchenko さんが Debian Packge 内のあるファイルが圧縮されていて、読むことが出来ないと気がつきました。そのファイルとは何でしょうか。

- A Word ファイル
- B PDF ファイル
- C SREC ファイル

13.2 2006 年 26 号

<http://www.debian.org/News/weekly/2006/26/> にある 6 月 27 日版です。

問題 3. 9 月にイタリアのある都市で Debian コミュニティカンファレンスがおこなわれます。そのある都市とはどこでしょう。

- A ベニス
- B デセンツァーノ・デル・ガルーダ
- C カリブ島

問題 4. 最近セキュリティチームのメンバーが増えました。それはだれでしょうか。

- A Steve Kemp
- B Hidehazu Koiwa
- C Andreas Barth

13.3 2006 年 27 号

<http://www.debian.org/News/weekly/2006/27/> にある 7 月 4 日版です。

問題 5. 最近また新しい OS に Debian を移植している噂があるらしい。それはどの OS か？

- A Plan9
- B Minix3
- C Mona

問題 6. Paul Wise さんがあたらしいグループを作成しました。それはどのグループか？

- A debian-smoker
- B debian-soccer
- C debian-flash

13.4 2006 年 28 号

<http://www.debian.org/News/weekly/2006/28/> にある 7 月 11 日版です。

問題 7. Matthew Garret はなにを断言したか

- A Debian に貢献していない者には Debian に要求する権利は無い
- B あれげはあれげ
- C 人生いろいろ

13.5 2006 年 29 号

<http://www.debian.org/News/weekly/2006/29/> にある 7 月 18 日版です。

問題 8. 7 月に不正侵入されたサーバはどれか？

- A gluck.debian.org
- B ftp-master.debian.org
- C hanzubon.jp

問題 9. 上川が発表したのは何か？

- A あれげハック
- B pbuilder やめます宣言
- C Intel Mac 向けの Debian サポートの進捗

13.6 2006 年 30 号

<http://www.debian.org/News/weekly/2006/30/> にある 7 月 25 日版です。

問題 10. ブータンの公用語向けの Debian は

- A HondaLinux
- B BongaLinux
- C DzongkhaLinux

13.7 2006 年 31 号

<http://www.debian.org/News/weekly/2006/31/> にある 8 月 1 日版です。

問題 11. Debian パッケージ内のドキュメントはビルド時にビルドするべきか？

- A ビルドするのに時間かかるからコンパイル済のものをいれるべき
- B アーキテクチャ非依存としてビルドするべき
- C アーキテクチャ依存としてビルドするべき

13.8 2006 年 32 号

<http://www.debian.org/News/weekly/2006/32/> にある 8 月 8 日版です。

問題 12. SPI の理事長は？

- A Neil McGovern
- B Bdale Garbee
- C Michael Schultheiss

13.9 2006 年 33 号

<http://www.debian.org/News/weekly/2006/33/> にある 8 月 15 日版です。

問題 13. Martin Krafft がアーカイブソフトウェアについて発表したのは？

- A チルダがサポートされた
- B 古いバージョンは自動で削除するようになった
- C セキュリティーアップデートの速度がはやくなった

13.10 2006 年 34 号

<http://www.debian.org/News/weekly/2006/34/> にある 8 月 22 日版です。

問題 14. Debian をサポートするというプレスリリースを出した会社は？

- A HP
- B IBM
- C Ubuntu

13.11 2006 年 35 号

<http://www.debian.org/News/weekly/2006/35/> にある 8 月 29 日版です。

問題 15. Alexander Wirt が FrOSCon のために準備したのは？

- A ぐるぐるの形をした金メダル
- B ぐるぐるの形をしたプレッツェル
- C ぐるぐるの形をしたぐるぐる

13.12 2006 年 36 号

<http://www.debian.org/News/weekly/2006/36/> にある 9 月 5 日版です。

問題 16. Joerg Jaspert が発表した新しいツールは

- A cdrkit
- B cdrecord
- C dvdrecord

13.13 2006 年 37 号

<http://www.debian.org/News/weekly/2006/37/> にある 9 月 12 日版です。

問題 17. Anthony Towns が提案したのは

- A BTS でのライセンス関連の問題についてタグをつけること
- B あらゆるライセンス問題はなかったことにすること
- C ライセンスなんて所詮ただの文章さ

13.14 2006 年 38 号

<http://www.debian.org/News/weekly/2006/38/> にある 9 月 19 日版です。

問題 18. 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 19. 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なく
- 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 20. 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

13.15 2006 年 39 号

<http://www.debian.org/News/weekly/2006/39/> にある 9月 26 日版です。

問題 21. 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 22. 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

13.16 2006 年 40 号

<http://www.debian.org/News/weekly/2006/40/> にある 10月 31 日版です。

問題 23. Frank Küster がカーネル 2.6.18 パッケージについて発表したのは？

- A まだ安定していないけどどんどん利用してください
- B General Resolution の結果、Linux じゃないカーネルを今後利用する
- C firmware blob を Debian package に含めるようにした

問題 24. mplayer パッケージになにがおきたか？

- A NEW キュー滞在時間の新記録をさらに更新した
- B Debian unstable に入った
- C もうあきらめることになった

14 Debian Weekly News 問題回答

上川

Debian Weekly News の問題回答です。あなたは何問わかりましたか？

1. C
2. B
3. A
4. A
5. B
6. C
7. A
8. A
9. C
10. C
11. B
12. B
13. A
14. A
15. B
16. A
17. A
18. B
19. C
20. A
21. A
22. A
23. C
24. B



あんどきゅめんてっど でびあん 2006 年冬号

2006 年 12 月 XX 日 初版第 1 刷発行
東京エリア Debian 勉強会（編集・印刷・発行）
