



第 20 回 東京エリア Debian 勉強会 事前資料

Debian 勉強会会場係 上川純一*

2006 年 9 月 16 日

* Debian Project Official Developer

目次

1	Introduction To Debian 勉強会	2
1.1	講師紹介	2
1.2	事前課題紹介	2
2	Debian Weekly News trivia quiz	3
2.1	2006 年 XX 号	3
3	最近の Debian 関連のミーティング報告	4
3.1	東京エリア Debian 勉強会 19 回目報告	4
3.2	Lightweight Language Ring	5
4	あなたが知らないうちに使っている Debian specific	6
4.1	はじめに	6
4.2	adduser	6
4.3	ifup	6
4.4	Xsession	7
4.5	lesspipe	7
4.6	Debian specific の見つけ方	8
5	翻訳へのさそい	9
5.1	はじめに	9
5.2	共通の作業用インフラストラクチャ	9
5.3	各種ソフトウェアの po や付属ドキュメントの翻訳	10
5.4	debconf-po 関連	11
5.5	ウェブページ関連	12
5.6	The Debian Description Translation Project (DDTP)	14
5.7	おわりに	16
5.8	参考文献	17
6	apt/dpkg のプロファイリング	18
6.1	oprofie のインストールと設定方法	18
6.2	oprofile が自分の利用している CPU をサポートしていない場合	18
6.3	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	19
6.4	テスト環境の作成	21
6.5	最適化の必要な部分の解析	23
6.6	最適化例	23
6.7	結果の検証	23
6.8	回帰テストの貢献	23
6.9	修正をフィードバック	23
6.10	参考文献	24
7	次回	25

1 Introduction To Debian 勉強会

上川純一

今月の Debian 勉強会へようこそ。これから Debian のあやしい世界に入るという方も、すでにどっぴりとつかっているという方も、月に一回 Debian について語りませんか？

目的として下記の二つを考えています。

- メールではよみとれない、もしくはよみとてられないような情報を情報共有する場をつくる
- まとまっていない Debian を利用する際の情報をまとめて、ある程度の塊として出してみる

また、東京には Linux の勉強会はたくさんありますので、Debian に限定した勉強会にします。Linux の基本的な利用方法などが知りたい方は、他でがんばってください。Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりを作りながらスーパーハッカーになれるような姿を妄想しています。

Debian をこれからどうするという能動的な展開への土台としての空間を提供し、情報の共有をしたい、というのが目的です。次回は違うこと言ってるかもしれませんが、御容赦を。

1.1 講師紹介

- 上川純一 宴会の幹事です。
- 小林儀匡 Debian Weekly News (DWN) および Aptitude の ja.po の翻訳をやっています。最近パッケージメンテナンスにも手を出し、そしてようやく Debian JP に入りました。

1.2 事前課題紹介

今回の事前課題は「XXX」というタイトルで 200-800 文字程度の文章を書いてください。というものでした。その課題に対して下記の内容を提出いただきました。

1.2.1

1.2.2 上川

2 Debian Weekly News trivia quiz

上川純一

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

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。

2.1 2006 年 XX 号

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

問題 1.

- A
- B
- C

問題 2.

- A
- B
- C

問題 3.

- A
- B
- C

問題 4.

- A
- B
- C

3 最近の Debian 関連のミーティング報告

上川純一

3.1 東京エリア Debian 勉強会 19 回目報告

東京エリア Debian 勉強会報告。8 月の第 19 回 Debian 勉強会を実施しました。今回は岩松さんが Debian Conference 開催進捗報告をしました。また、Lightning talk を開催しました。

今回の参加人数は 17 人でした。

参加者は gotom さん、山下さん、山根さん (中央線の事故の影響で結局宴会の最後になって到着)、岩井さん、南谷さん、前田耕平さん、たかやさん、きたはらさん、岩松さん、あけどさん、平川さん、小林さん、えとーさん、吉田@板橋さん、みつかさん、野首さん、上川です。

まず、Debian のソフトウェアに関するポリシーである、Social Contract の内容を確認しました。

異様な盛り上がりを見せながらユーザの声を紹介。事前課題をねたにしました。たかやさんが howm の Debian パッケージをオフィシャルパッケージにすることなので、期待です。

Debian Weekly News Quiz は今回は正解者には Debian シールをプレゼントしました。問題数を少なめに、早押しならぬ、早手あげで競争してもらいました。

岩松さんが北海道で下見をしてきた内容を踏まえて Debconf in Japan の検討事項について報告しました。東京や大阪など国際的にトランジットの利便のよい場所がよいですね、という話などが出ました。

Lightning talks の一発目は野首さん。プロフェッショナルな感じで、しっかりと執念を感じさせてくれる話でした、ありがとうございます。

吉田さんには、大衆向け IPv6 サービスを利用してみた話をしていただきました。まだいろいろと問題があるようで、すぐに使えるのかなあ、という印象はうけましたが、今後オフィシャルパッケージにマージされたりするとより利用しやすくなるでしょうから、今後の活躍に期待です。

Henrich さんは鉄道事故の関係で間に合いませんでした。残念。

山下さんには誕生日に思うことに付いて語っていただきました。淡々と語っていただきありがとうございます。

上川が module-assistant の使いかたについて話しました。module-assistant は意外にもつかっている人が少なく、話をしがいがありません。kernel module は必要な場合もあるので、ぜひパッケージできるようにしておくといえます。特に、module-assistant は使う側からすると従来の手法より非常に使い易くなっているのでよいよ、という話でした。

上川が最後に「board@jp お仕事日記」として発表しました。ここにかけるような内容ではないので、割愛。

今後のイベントにどのような参加方法をとるかということを検討しました。OSC-Fall にはブースも出して、いろいろなデバイスをハックしている姿を展示しましょう、ということになりました。また、OSC の沖縄には岩松さんとわたしは参加する、ということになりました。さらなるメンバー募集します。KOF については 4 人くらい参加すると宣言していたので、参加することになりそうです。

宴会は「土間土間」にて開催。けっこうよい場所を独占できたので、よい感じでした。

3.2 Lightweight Language Ring

上川が、軽量言語の祭典、LLRING に参加してきました。

発表した内容は、real kernel shell についてです。C 言語をライトウェイトに使いましょう、という話題です。

Haskell が異様にはやっていたのと、ウェブフレームワークでがりがりとやっているかたがたを横目にカーネルの話をしてきました。



4 あなたが知らないうちに使っている Debian specific

澤田さん

4.1 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。

4.2 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がパスワード入力はありません。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 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 等呼び出してユーザを作成しています*1。

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。

4.3 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。

*1 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

4.4 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。なかったら `/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。

4.5 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が。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。

4.6 Debian specific の見つけ方

それが Debian specific であるか知るためには 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思われます。

5 翻訳へのさそい

小林さん

5.1 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。

5.2 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。

5.2.1 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`^{*2}および `debian-www`^{*3}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>^{*4}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^{*5}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されているかた、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^{*6}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^{*7}にそれぞれ登録しておく、内容に関する質問や間違いの修

^{*2} `debian-doc@debian.or.jp`

^{*3} `debian-www@debian.or.jp`

^{*4} 準備中の新サイトでは <http://www-internal.debian.or.jp/community/ml/>。

^{*5} `debian-japanese@lists.debian.org`

^{*6} `debian-doc@lists.debian.org`

^{*7} `debian-www@lists.debian.org`

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。

5.2.2 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ポイ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^{*8}: <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](http://www.debian.org)」を、Debian Weekly News から探したければ「[site:www.debian.org/News/weekly](http://www.debian.org/News/weekly)」をつけて検索し、引っ掛かっ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。

5.3 各種ソフトウェアの po や付属ドキュメントの翻訳

5.3.1 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ため、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。

5.3.2 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>

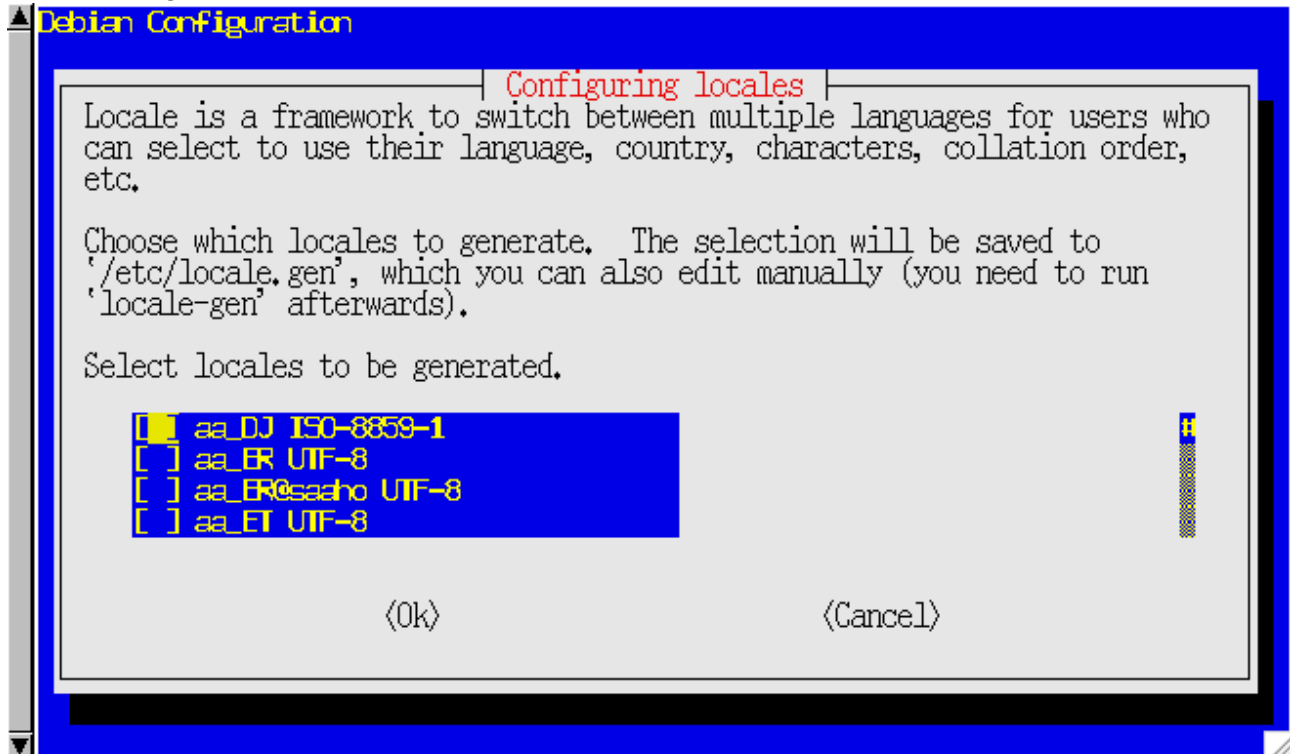
^{*8} Debian Weekly News 用 wml ファイルの後半の Security Updates および Removed Packages から用語を抽出して po としたものです。

5.4 debconf-po 関連

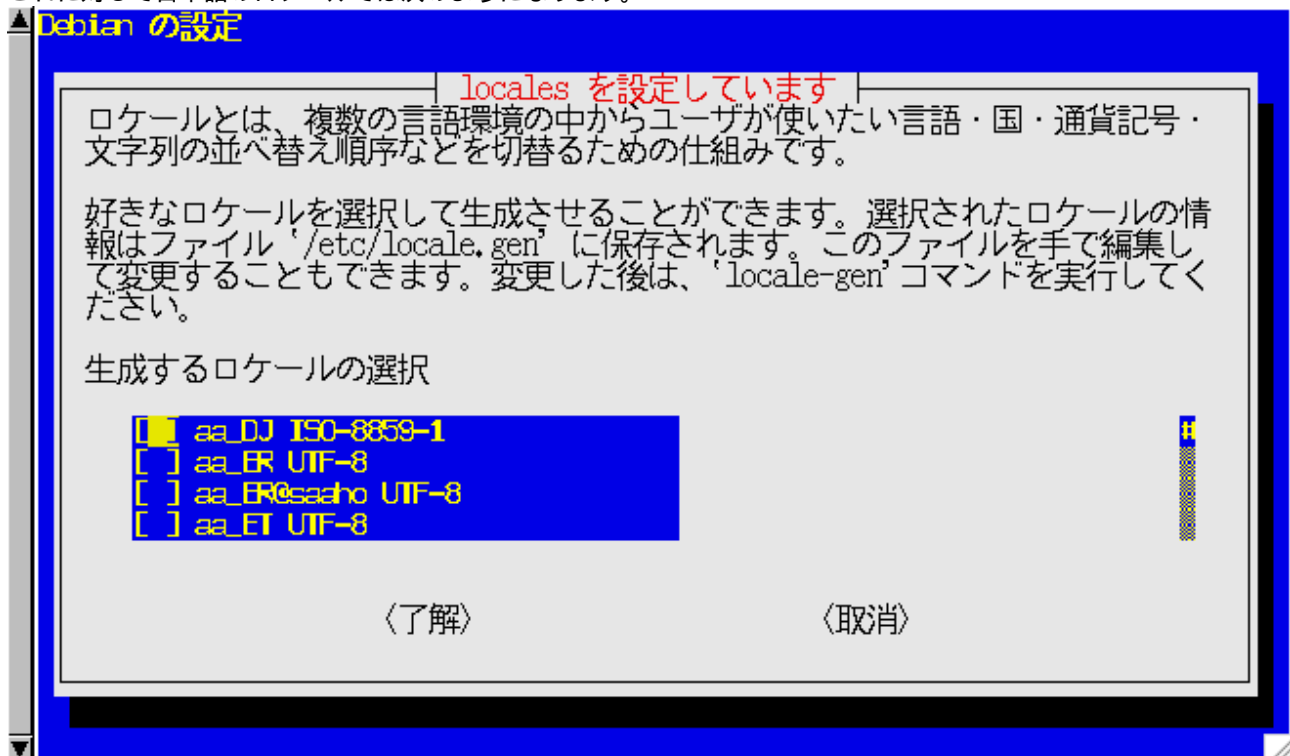
5.4.1 debconf-po とは

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なります。



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
_Chchoices: 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。

5.4.2 作業方法

作業を開始する前に、まずは 5.4.3 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 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査読してもらうことを強くお勧めします。

5.4.3 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。

5.5 ウェブページ関連

5.5.1 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^{*9}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&amp;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"
```

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。

5.5.2 作業方法

翻訳作業を開始するには、目的のページの WML ファイルを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ーア 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。

^{*9} <http://cvs.debian.org/?root=webwml>

5.5.3 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。

5.6 The Debian Description Translation Project (DDTP)

5.6.1 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随情報の一つで、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^{*10}以降がパッケージ説明文です。

```
nori1[12:04]% aptitude show aptitude          whale:~/svnwc/deb/skddic/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が^{*11}、Debian のホストの問題で暫く停止

^{*10} aptitude 0.4.2 以降では「説明文:」という訳語に変わっています。

^{*11} <http://d.hatena.ne.jp/denson/20050315/p2>

していました。完全にではありませんが、最近ようやく復活の兆しが見え始めました^{*12}。最近は以下のような状況です。

- Debian 本家のウェブサイトにも DDTP のページ^{*13}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。

5.6.2 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^{*14}。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^{*15}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^{*16}。

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間ロックされ、メールで取り寄せた人だけ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。

^{*12} <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いました。

^{*13} <http://www.debian.org/international/l10n/ddtp>

^{*14} <http://www.debian.or.jp/devel/doc/Description-ja.html>

^{*15} <http://www.debian.org/international/l10n/ddtp>

^{*16} 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。


```
# 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:
#
```

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^{*17}。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、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 ddtss-send^{*18}や ddtc パッケージ^{*19}などのヘルパーも利用可能です。

ウェブインタフェース

Martijn van Oosterhout さんが作成したウェブインタフェースは 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。

5.6.3 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。

5.6.4 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。

5.7 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

^{*17} 誤りを発見し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。

^{*18} <http://surf.ap.seikei.ac.jp/~nakano/linux/ddts-send.ja.html>

^{*19} <http://packages.debian.org/ddtc>

にしています。

5.8 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>

6 apt/dpkg 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できるのか、をあきらかにしてみましょう。

6.1 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^{*20}。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)*21
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^{*22}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。^{*23}

6.2 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 7 月現在時点で、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。

```
$ 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

^{*20} 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。

^{*21} 2.6.18-rc1 現在

^{*22} 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

^{*23} 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もしれません。

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、それを取りこみます。念のため、今後作業する人のためにBTSにも登録しました。380462^{*24}

確認してみると、どうやら動作してくれてそうです。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 の処理がかたまっている部分を目指して、何か問題がないかを眺めてみて、何も問題なく、それなりに問題が追求できにくくなった後に、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_RETIRED: (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]
```

6.3 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、上位に出現するなど、必要そうであれば、あとでデバッグ情報のあるバージョンを追加します。

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

^{*24} <http://bugs.debian.org/380462>

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リングしてみます。^{*25}

```
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 findnamenode
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 iterprgnext
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_nzero
687 0.5461 dpkg dpkg ensure_package_files_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。

^{*25} 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、ファイルを開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。385704^{*26}

```

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 findnamenode
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

```

6.4 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` を行い、外部のファイルを見せます。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 libgl2.0-0
# unset LD_PRELOAD
# unset COWDANCER_ILISTFILE
# mount -t oprofilefs nodev /dev/oprofile >/dev/null

```

`chroot` で調べてみると、`perl` が一番重い処理をしているということがわかりました。こりゃチューニングしにくいですね。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、特露骨にめだ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 vmlinux vmlinux 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 vmlinux vmlinux __copy_to_user_ll
1181 0.7422 vmlinux vmlinux 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 vmlinux vmlinux 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、この時点では 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しょう。

```
# opreport
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 (tid:31689 range:0xb7f47000-0xb7f48000)
11 0.0286 anon (tid:31917 range:0xb7f49000-0xb7f4a000)
9 0.0234 anon (tid:31841 range:0xb7fcb000-0xb7fcc000)
8 0.0208 anon (tid: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 Fcctl.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 opreport
CPU_CLK_UNHALT...|
samples|      %|
-----|-----
14091 58.9187 opreport
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
```

6.5 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。

6.6 最適化例

今回の結果で適用できる最適化を分析します。

6.7 結果の検証

さきほどのベンチマークを利用して、状況が改善していることを確認します。

6.8 回帰テストの貢献

できるようであれば、今回の修正が二度と必要ないように、パフォーマンスのデグレードがすぐに検出しやすいよう、再現しやすいテストケースを追加してみるのもよいでしょう。

6.9 修正をフィードバック

パフォーマンスの改善のための修正をフィードバックします。Debian の場合、BTS にパッチを登録します。該当するバグ番号は下記です。

- XXX
- YYY

6.10 参考文献

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



未定です。内容は本日決定予定です。
参加者募集はまた後程。



Debian 勉強会資料

2006 年 9 月 16 日 初版第 1 刷発行

東京エリア Debian 勉強会（編集・印刷・発行）
