



第 8 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*

2005 年 9 月 10 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	5
2.1	2005 年 32 号	5
2.2	2005 年 33 号	6
2.3	2005 年 34 号	7
2.4	2005 年 35 号	8
2.5	2005 年 36 号	10
3	最近の Debian 関連のミーティング報告	11
3.1	東京エリア Debian 勉強会 7 回目報告	11
4	debconf 論	12
4.1	はじめに	12
4.2	debconf を使っているパッケージの設定の仕方	12
4.3	debconf の裏側	13
4.4	debconf データベース	19
4.5	debconf を使っているスクリプトのデバッグ	20
4.6	おわりに	20
5	グループワーク用メモ	21
6	次回	22

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ふ方も，すでにどっぷりとつかっているといふ方も，月に一回 Debian について語りませんか？

目的として下記の二つを考えています．

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

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

Debian をこれからどうするといふ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．今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

- 鵜飼さん Debian JP のサーバたちにとってなくてはならない存在です．
- やまねさん DebianJP のウェブページに改革をもたらそうとしています．
- 上川純一 宴会の幹事です．

1.2 事前課題紹介

今回の事前課題は「debconf をつかっていて答えるのにこまった質問」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 澤田さん

答えるのにこまった質問、最近一番こまったのは flex のバージョンが上がるときに「POSIX 準拠じゃなくなるけどいいかい？いやなら flex-old 入れてね」という質問だ。この質問はいけていない。何がいけないかというと

* デフォルトが false な点

* false と答えると exit 1 する点

の 2 点だ。exit 1 するのでインストールエラーになる。エラーと言われるのでどきどきする人が多いのではないだろうか。特に woody から sarge に上げようとしてはまった人が多いと思われる。いや、私のことですが何か？

mpich と lam みたいに flex と flex-old を別々の場所にインストールして選んだ回答に応じて alternatives を設定とかしてくれるとうれしい気がする。

false と答えてインストール失敗した後、普通に apt-get install flex-old できるのか？と試そうとするが false と答えてるのにインストールされてしまう。何でだ？

1.2.2 岡島さん

簡単に Debian 歴を紹介しますと、Debian をはじめた原因の一つは、Redhat の Fedora 移行の混乱に嫌気がさしたことであって、ようは始めたのは最近ですし、さらに、依然として本業のほうは、顧客の意向もあって Redhat なので、結論としては私の Debian 経験は非常に乏しく、ようは単なる初心者なのですが、一通り技術はわかるが、しかし Debian は初心者、という観点からのコミュニティに対する貢献、というのもあるかと。

で、Debian の印象ですが、「上手くいくときはいいのだが、いかないときに困る」、というのが印象ですね。これは、apt-get 特有の問題ではなく、自動化を進めればどんなものにも当てはまることです。

で、お題の「Debconf で困った質問」ですが、特にありません。デフォルトで何も考えずインストールしますが、それで特段問題になったことはないですね。これは、Debian のデフォルトの選択が結構賢い、ということでもあるのでしょうか。

が、しかし、パブロフの犬状態で反射的に「いえーす、イエース」と答えてしまうインストールに不安があるのは事実ですし、それじゃあ商売にならないというのはもったいえます。

結論なんですが、出題者の鵜飼さんは、多分、勝手な推測ですが、もっと質問文をわかりやすく、、、とかお考えなんだと思いますが、そんなことより、以下の3つのほうが歓迎されると思います。どうせ、いくら質問文をわかりやすくしても、ほとんどの利用者は、本能的に「イエース、イエース、イエー——ース」の連打でしょうし、そもそも、それで使えなければ結局使えないですね。

1、Roll back, Snapshot, Checkpoint, Undo

ようは同じことなんですが、単に、apt-get remove とかを、、、というのではなく、もう、ファイルシステムレイヤーで、もろスナップショットや undo がほしいです。まあ、これは、Debian の問題ではなく Linux の問題でもあるのですが、Debian のように、システム領域にばりばり勝手なことをしていく（と、Debian を知らない人が思い込みやすい）システムにおいては、より重要かと。それ以外の理由においても、この手の機能が Linux に実装されると、システム管理上の利点は非常に大きいので、なんとかならないか、といろいろ個人的にも研究しているのですが、なかなか難しいです。

2、過去の Deb を！

個人的な印象では、Debian は、常に勝手に最新の Deb が入ってしまい、それ以外の選択がない、という印象です。Pinning とか、やり方はあるみたいですが、なんかよくわからない。そのくせ、必ずしも最新バージョンが安定しているとは限らない。Stable ブランチでもそうなんですから、Sid ならもっと。さらに、勝手にアップグレードされたパッケージが不安定だった日には、途方にくれます。どうやって戻すんですかね、これ。というか、いま、Debootstrap で遊んでるのですが、ことごとくすべてのブランチで不安定なのは、私の日ごろの素行のせいですかね。当然、Pbuilder もコケます。解決策ですが、どうも鵜飼さんがある程度やってるみたいですが、ようは、過去の Deb を保存しているアーカイブはないの？ということです。それをつかって手軽にロールバックできるだけで、ずいぶん Debian の印象は違うでしょう。

3、インストールログを。

そんなの、とっくの昔にもうあるよ、とかいわれそうですが、なにをシステム管理者がやったか、をきっちりログにしてもらえるとうれしいです。Debconf の質問一つとっても、何にどう答えたか、次の管理者がすぐわかる。さらに、何にどう答えたか、レポートにして顧客に送れる。そんなんだといいですね。

1.2.3 中島さん

質問されても読まないから困らないらいいと思う。それにたいして選択に数ないと思うので適当に打っていけば、だいたい決まってくるのだから数打てば当たるわけで、ぜんぜん困らないと思う。困るはずがないと思う。しかし困らないけれど、もしも困ったときのために、質問の内容と答えをノートに書いておくというのがいいと思う。これで万全だ。ノートに書いておけば、これで適当に打っても大丈夫だ。これで良いと思う。なんか間違っても大した支障はないと思うから適当に打っていけばいい。

1.2.4 武藤さん

メモするものが手元にないときに「 が変わったので、インストールが終わったあとに を にしてそれから してください」という注意書きだけ出てくるのが一番困るかな。

ほかは「debconf で困る点」になっちゃうけど、優先順位が適切に設定されていなくて high でも質問攻めだったり、逆に high だと重要な質問をしなかったり (XFree86 のキーボード設定とか) とか。あと、「戻る」を dialog ベースでもできないかなあ。

1.2.5 小林儀匡さん

debconf を使っていて答えるのに困った質問は色々とありましたが、あまりに多くて思い出せないので具体的には挙げません。代わりに、それに纏わることを書くことにします。

debconf を使っていて答えるのに困る質問はスキルの向上に伴いかなり減りましたが、初めてインストールしたときにはやはり多かったです。質問内容が分からず、適当にデフォルトの答えで済ませてその場をしのぎました。本当はそういったものは逐一手でメモをとっておいて後で再設定するべきなのかもしれませんが、かなり多かった上に、ディスプレイに表示された質問を紙にメモするという作業の空しさ (?) のため、途中からは閾値を上げ、多くの質問は分からなくてもメモせずにスルーしてしまいました。

自分と同様の思いをしている人が多いのか少ないのか分かりませんが、debconf で出た質問の一覧、あるいはせめて debconf で質問をしたパッケージの一覧が、APT を使った一連のパッケージインストール作業の最後に表示されたらよいのに、と感じています。質問だけでなく、「インストール後には～を読むように」という注意喚起のメッセージも、どうせなら、インストールプロセス中に表示するのではなく、最後に表示させたほうが効果的でしょう。パッケージインストール作業の最後に表示させずとも、何らかのファイルに一覧を書き出すのも効果的かと思います。

1.2.6 上川

debconf の使い方がよくわかっていなくて、先日 gotom さんにいわれるまで、質問にキャンセルという選択肢があるのに気づいていませんでした。問題答えるのに困るものがあるのと、好ましくない答えをするとインストールエラーになってしまうパッケージがあるのが困り物です。

debconf で質問されてもほぼデフォルトにしています。で、デフォルトでは動かないパッケージがあるといかりくるってバグをファイルします。

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 2005 年 32 号

2005 年 8 月 9 日です．

問題 1. あるパッケージをアップロードした場合の debian に与えるリスクを計算する方法が欲しいというリクエストに対して出て来た、現在どのパッケージがいちばん testing に影響しているというのを調査するには

A ゲーグルで検索する

B <http://bjorn.haxx.se/debian/stalls.html> にある どのパッケージがどれくらいのパッケージの testing 入りを邪魔しているかというページを見る

C `ls -l /`

問題 2. GNUStep で問題なのは何か

A FHS に全く準拠していなく、GNUStep をそのまま FHS に準拠させるのは難しい．

B 使いにくい

C 動かない

問題 3. Ian Jackson は、「Debian」 Core Consortium について何を宣言したか

A 我々が Debian をのっとった

B Debian は使えない

C Debian Core Consortium という名前は正式名称ではないので問題無い

問題 4. MySQL 4.0 から 4.1 への移行についてということが議論されたか

A libmysqlclient の移行のためにバグをファイルするのは、今は gcc 4.0 の移行中なため、避けて欲しい．

B MySQL はもうすてよう

C PySQL に名前が変わりました

問題 5. Andrews Barth は gnome2.10 を etch にいれよう、testing に入れるためにアップロードをしばらくひかえよう、とかげごえをかけましたが、出た反論が

A gnome2.10 なんでもつかっていない

B Nathanael Nerode さんが言うには、xorg の移行によってブロックされているのでしばらくは無理

C testing なんでも存在しません

問題 6. Helen Faulkner さんが作成したメーリングリストは

- A debian-science メーリングリスト
- B debian-helen メーリングリスト
- C debian-faulkner メーリングリスト

問題 7. xorg6.9 について David Nusinow が報告したのは

- A quilt ベースのパッチシステムに移行したおかげで以前までは数週間かかっていたパッチの移行が 3-4 日しかかからなくなりました .
- B 画面表示のいろあいが変わりました
- C 高速化しました

2.2 2005 年 33 号

2005 年 8 月 16 日です .

問題 8. Debian の 12 回目の誕生日はいつか

- A 2005 年 8 月 16 日
- B 2006 年 8 月 16 日
- C 2005 年 4 月 1 日

問題 9. RC バグの影響で testing からパッケージを削除される場合がある . その場合にどうしたら testing にパッケージが戻るか

- A リリースマネージャに賄賂をおくる
- B リリースマネージャを人質にとる
- C RC バグを全部修正する .

問題 10. カーネルのソースパッケージは linux-source-2.6.12 になっており , ソースの入ったバイナリパッケージは linux-2.6 になった . その理由は

- A ビリオドの消費を節約するため
- B ユーザを混乱させるため
- C 古いバージョンをアーカイブ内に維持しつつユーザのアップグレードを簡単にするため

問題 11. Debian メンテナはバグを上流に報告する義務がある , という点に関して , Eric Dorland が反論した理由は

- A firefox パッケージには 300 くらいのバグが現在 open されており , 上流にフォワードするという作業だけで時間が無駄にとられてしまう .
- B BTS なんか見てない
- C 自分では使っていないので使っている人が報告してほしい

問題 12. Joerg Jaspart が発表した , Debian Developer の追放の規則によると

- A 追放期間は一年間で , 保釈金を支払うともどつてくれる
- B DPL のわるぐちを言った Debian Developer は追放できる
- C 一旦追放されても NM プロセスを経て Debian Developer にもどつてくれる

問題 13. LinuxFund が Debian に対して実施すると発表したのは

- A 月 \$ 5 0 0 の資金提供を 1 年間実施する .
- B Debian DVD を売る
- C Debian を使う

問題 14. Hanna Wallach が Debian Women の現状について懸念を示した内容としては

- A Debian Project をのっとるためにはもっと高速に Debian Women プロジェクトが成長する必要がある
- B Debian Women という名前は性差別なので Debian people と呼ぶべきだ
- C 女性のために簡単に入れる debian-women という新しいコミュニティをつくるのが目標ではなく , Debian Developer に参加するための援助をしていくのが目標なのでそれを忘れないように

問題 15. Andrews Schuldei が求めたのは

- A Debian Developer たちが集まって作業するのと効率がよい場合もあるので , できる場所などの寄付
- B 個人的な利益
- C 世界人口の半分が Debian Developer になること

問題 16. Torsten Landschoff が提案したのは , 共有ライブラリパッケージの SONAME が変わっただけのような場合に関しては自動で ftp-master が ACCEPT する仕組みだが , その仕組みに対しての反論は

- A Joerg Jaspart 曰く , 空のパッケージがつい最近アップロードされていたので , そういう間違いが検出できるのは重要だ .
- B Joerg Jaspart 曰く , ftp-master の権限を弱めるような実装はありえない
- C Joerg Jaspart 曰く , その実装の詳細な部分が気に入らない

問題 17. Gutenprint が unstable にリリースされた . 以前はなんという名前だったか

- A gimp-print
- B gnome-print
- C kde-print

2.3 2005 年 34 号

2005 年 8 月 23 日です .

問題 18. DCC に関して , Debian 商標についての決定権を委任されたのは

- A Don Armstrong
- B Andreas Schuldei
- C Branden Robinson

問題 19. gotom さんが glibc について宣言したのは

- A カーネル 2.4 しか動かないシステムではコンパイルできないぞ .
- B ドアストップアーキテクチャについては , サポートはしない
- C よくこわれるのでインストールしないでください .

問題 20. Steve Langasek によると , woody から sarge の間のための移行パッケージというのは

- A 必要なので永遠に残す
- B 途中のリリースを省略したアップグレードはサポートしないので , woody から sarge への移行専用のパッケージは unstable では削除するべき

C バグが多いので対応してほしい

問題 21. Ramakrishnan Muthukrishnan さんと Ganesan Rajagopal があつく Debian について語ったイベントは

- A インドで実施した Debian Conference India
- B 日本で開催した Debian Conference
- C 中国で実施した Debian Asia Miniconf

問題 22. woody にしかない古いバグをクローズするのはどうしたらよいか

- A 「君の事はずっと嫌いだった」と書いたメールをそのバグに対しておくりつける
- B どのバージョンで問題が解決したのか、という情報を BTS に記録する
- C woody タグを付ける

問題 23. Lars が piuparts で実施してみたらどうなったか

- A たくさんバグが見つかった
- B Debian は完全であることが立証された
- C piuparts 自身が動かなかった

問題 24. バグ報告に GPG 鍵認証を要求する話についてどういう意見が出たか

- A Debian Developer 以外でもできるようにはすくなくともしてほしい
- B バグ報告は高尚な行為なので Debian Developer 以外は実施してはならない
- C GPG より違う認証プロトコルを使おう

問題 25. BTS の LDAP ゲートウェイはどうなったか .

- A master のポート 10101 にて稼働再開した
- B 止まったまま
- C 使えないのでステ

2.4 2005 年 35 号

2005 年 8 月 30 日です .

問題 26. Joerg Jaspart さんが発表した NEW キューで REJECT される要因は

- A DEB_AUTO_UPDATE_DEBIAN_CONTROL を使っている CDBS のパッケージは拒否
- B パッケージ名が glibc
- C メンテナ名が kmuto@debian.org

問題 27. Debian GNU/kFreeBSD では、Debian main の何 % のパッケージが利用可能か

- A 81.69%
- B 15%
- C 3%

問題 28. Andreas Barth さんがライブラリについて要求したのは

- A すでに移行が多数並行しすぎているため、ライブラリの SONAME の変更などは発生させないでほしい。
- B ライブラリは全部 static link に変更すること
- C ライブラリはすべてデバッグ情報を残す事

問題 29. バグ報告の影響をうけているユーザが投票することで、重要度をトラッキングするランキングを実施しようという意見に対して、現在すぐにでもできそうな方法は何か

- A バグに最後に活動した日付によってランキングを付ける
- B ず BTS と Popularity Contest を連携させる
- C 電波によって影響度を推察

問題 30. POSIX Shell として POSH を利用してスクリプトの試験を実施することに対しての反論は？

- A busybox のシェルですらそれよりも機能があるため、実質的な利益が無い
- B posh はつかえない
- C posh はバグだらけだ

問題 31. Mozilla についてセキュリティーフィックスとして新しい上流バージョンを利用することに関しての理由は

- A めんどくさい
- B 上流がそれじゃないと嫌だというから
- C Ubuntu 向けに Martin Pitt がセキュリティーフィックスのバックポートをけっこうな時間をさいて実施してみたが、難しかった

問題 32. Yaroslav Halchenko が十分な AM の人数がいるのに 100 日以上かかっているのは NM プロセスが忍耐力を試験しているのではないか、というコメントをした際の Marc Brockschmidt の回答は

- A 陰謀なので気づかないください
- B 忍耐力を計測しているのにきついちゃった？
- C そんなことはなく、今一番活発な AM である Marc と Joerg は担当している NM の数を減らそうとしている最中なので、AM が増えるのは好ましい

問題 33. Joey hess が debconf に依存しているけど debconf-2.0 に依存していないパッケージをなくしたいのは

- A cdebconf に移行できるため
- B debconf2.0 の機能だけにしたいから
- C debconf のバージョンを新しくしたいから

問題 34. lists.debian.org にあるウェブ経由でのメールインタフェースでメーリングリストのスパムを報告するリンクは何をするものか

- A メーリングリストにスパムを投げる
- B メーリングリストにスパムが来てますというメールをメーリングリストの参加者にランダムに投げる
- C 実はまだ統計情報を集めているだけで何をするというものでもない

問題 35. ビルドするのに non-free なシステムが必要なパッケージは DFSG-Free か

- A DFSG-free でもビルドするのに non-free なものが必要なら DFSG-Free とは言えない
- B パッケージ自身のソースがフリーならフリーだろう
- C DFSG-Free の定義による

2.5 2005 年 36 号

2005 年 9 月 6 日です .

問題 36. KDE の C++ 移行の状況はどうなったか

- A kde と arts と qt が全部移行完了した
- B 永遠におわらなさそう
- C C++ ってなんですか ?

問題 37. Wiki の内容について何が議論になったか

- A wiki に書いてある内容のライセンスは何か
- B wiki の内容がみにくい
- C wiki つかいづらい

問題 38. curl に何がおきたか

- A openssl を使うようになった
- B gnutls 専用に移行した
- C ssl を独自で再実装した

問題 39. データベースを削除するか質問するのに debconf を利用することに関して joey hess は何をいったか

- A debconf が存在するのかをきっちり確認してなら , purge される際に debconf の質問をきいてデータを消すかどうか決めるのは悪いことではない
- B そんな重要な質問に debconf を使うな
- C いかなる場合でもユーザデータは削除するべきではない

問題 40. Marc Brockschmidt が要求したのは

- A パッケージをアップロードするさいにはちゃんと変更点を報告すること
- B API を変更しないでほしい
- C API が変更したときに通知すること

問題 41. Wolfgang Borgert が意味の無い README ファイルについて苦情を述べた . その例でなかったのは

- A 「Debian パッケージはあるので , apt-get install パッケージ名でインストールすることが可能です」と書いてある README ファイル
- B 「私の名前は中村です」と書いてある README ファイル
- C 「このファイルは XXX パッケージの README ファイルです」と書いてある README ファイル

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 7 回目報告

前回開催した第 7 回目の勉強会の報告をします。

8 月の第 7 回東京エリア debian 勉強会は、Debian で新しくパッケージをアップロードするためのステップの話や、フィンランドのヘルシンキで開催された debconf5 であった議論についての報告がありました。また、debian の悪いところについての愚痴大会を開催して、いろいろな意見が飛び交いました。

DWN quiz に関しては、今回は満点は小林さんと gotom さんでした。GNU Hurd でパッケージが 40% しかビルドできていないという点について、FreeBSD より Linux に近い設計のような気がするのに変ですねえ、という話しがでていました。debbugs でバグに subscribe できるようになったり、バージョントラッキングが追加されたり debconf5 の最中に機能追加されたものがたくさんありました。

岩松さんがはじめての ITP からパッケージのアップロードの流れについて説明してくれました。ITP をするというとライセンスについて議論になり、上流の開発者がライセンスを変更したり、パッケージの存在意義について議論になったりします。また、正式に Debian Developer になるまでは、現在 Debian Developer である人に「スポンサー」を依頼し、代わりにアップロードしてもらうことになりますが、その際にパッケージの内容のチェックが入りました。スペルミスや、Description の文章の修正などからパッケージの alternatives の優先度の値の根拠についてなどの変更を経て、最終的に無事にアップロードできました。

debconf5 での参加報告を後藤さんが実施しました。multiarch サポートについてはまた新しいプロポーザルを Tollef が出して来ていますが、またメンテナンスするのが大変そうですね、という話題が出ました。dpkg については機能をいろいろ追加して dpkg2.0 を作成するとのことですが、2.0 は etch+1 でリリースする予定だとか。Debian GNU/kFreeBSD のひとたちは本当に楽しそうでしたとのことでした。

その他、宴会では半年分の資料をまとめてみようということでごそごと飲みながらノートパソコンを出して作業していました。

4 debconf 論

鵜飼文敏^{*1}

4.1 はじめに

debconf とは、Debian においてパッケージの設定を行なうためのフレームワークおよびそれを実装したパッケージです。そもそもは、元 Debian Project Leader の Wichert Akkerman 発案の configuration database framework 構想^{*2}であり、debconf は、それに基づく Joey Hess による実装です。

従来、パッケージの設定のうち、パッケージメンテナがデフォルトを決めにくいもの、つまりシステム管理者によって決定されるようなものは、メンテナスクリプト^{*3}で、プロンプトをだし管理者が入力した値に基づいて設定ファイルを生成するようにしていました。これはこれで非常に柔軟性が高い^{*4}のですが、パッケージによって^{*5} やりかたが異なってしまう、ディストリビューションとしての統一感に欠けてしまうという欠点がありました。また、パッケージのインストール時・アップグレード時に常に管理者の介入が必要になってしまうために、インストール・アップグレード中にコンソールを離れているわけにはいかないという問題もはらんでいます。

そこで考案されたのが debconf です。debconf を使うことで設定データを統一して扱うことができるようになります。

4.2 debconf を使っているパッケージの設定の仕方

apt-utils をインストールしておくで apt-extracttemplates を使って、パッケージをダウンロードしてインストール作業をする前^{*6}に debconf を使ってパッケージの設定をおこなうことができます。この時は debconf 自体の設定 debconf/frontend に従ったフロントエンド^{*7}をつかってユーザとのやりとりがおこなわれます。また、設定には優先度 (priority) ^{*8}がつけられており、debconf/priority の値より優先度が高いものだけがフロントエンドを介してユーザとやりとりをおこなわれるようになります。

またインストールした後も dpkg-reconfigure でパッケージの設定をしなおすことができます。dpkg-reconfigure は、debconf/priority の値にかかわらず低優先度 (low)^{*9}以上すなわちすべての設定をやりなおすようになっています。

また、現在そのパッケージの設定値がどうなっているかを知りたい場合は debconf-show コマンドが使えます。

```
# debconf-show debconf
* debconf/priority: critical
* debconf/frontend: Dialog
```

^{*1} Fumitoshi UKAI, ukai@debian.or.jp, ukai@debian.org, Debian Project

^{*2} /usr/share/doc/debian-policy/debconf_specification.html

^{*3} postinst など

^{*4} メンテナががんばってメンテナスクリプトを書けばいろいろできる

^{*5} メンテナによってというべきか?

^{*6} unpack をする前

^{*7} dialog, readline, gnome, kde, editor, noninteractive のどれか

^{*8} 重要 (critical)、高 (high)、中 (medium)、低 (low) のどれか

^{*9} -p オプションを使わなければ

4.3 debconf の裏側

apt-get がパッケージをダウンロードしてくると/etc/apt/apt.conf.d/70debconf というファイルにより dpkg-preconfigure -apt が実行されるようになります。

```
// Pre-configure all packages with debconf before they are installed.
// If you don't like it, comment it out.
DPkg::Pre-Install-Pkgs {"usr/sbin/dpkg-preconfigure --apt || true";};
```

dpkg-preconfigure では、apt-utils の apt-extracttemplates を使っているため、apt-utils がインストールされていないと、debconf の設定はこのタイミングではおこなわれません。

dpkg-preconfigure -apt に対して、いまダウンロードしたパッケージのリストがわたされるので、それらから template をとりだして config スクリプトの実行をおこないます。

- apt-get による*.deb のダウンロード。/var/cache/apt/archives/にパッケージがおかれる
- dpkg-preconfigure -apt の実行
 - /var/cache/apt/archives/からインストールしようとする deb をみつける
 - control 情報の templates と config をとりだす^{*10}
 - templates を load して、debconf データベース^{*11}を更新
 - config スクリプトを実行する。db_set、db_input や db_go
- dpkg -unpack
 - preinst を実行
 - パッケージを展開し、ファイルをおきかえ
- dpkg -configure
 - postinst を実行。db_get

基本的には、config スクリプトで設定情報を決定し^{*12}、postinst スクリプトでそれを読みだして実際のパッケージの設定ファイルなどに書きだすという処理をおこなうことになります。

なお、debconf データベースは registry として使う^{*13}ので、debconf データベースを参照するのはメンテナスクリプトだけにしておく必要があります。また、debconf 以外で設定ファイルを変更された場合も、その情報を上書きすることなく反映する必要があります。

debconf では、メンテナスクリプトと debconf データベース間のやりとりのプロトコルを決めています。こうすることで、フロントエンドをかえたりバックエンドのデータベースの実装を変更したりすることができるようになっていくわけです。

^{*10} dpkg -I パッケージ.deb templates config。apt-extractpackage を使う

^{*11} /var/cache/debconf/*.data

^{*12} ユーザに対して設定情報をたずねるか、デフォルト値を設定する

^{*13} 使うと debconf abuse として bug をくらう

シェルコマンド	内容	データのむき	使う場所
db_version "2.0"	バージョンネゴシエーション		
db_capb multiselect	キャパビリティネゴシエーション		
db_title タイトル	タイトル文字列の設定	スクリプト ユーザ	config
db_stop	debconf の停止		postinst, postrm
db_input プライオリティ 変数名	変数への入力	テンプレ ユーザ DB	config
db_go	入力の実行		config
db_get 変数名	変数のとりだし	DB スクリプト	postinst
db_set 変数名 値	変数の設定	スクリプト DB	config
db_reset 変数名	変数の初期化	テンプレ DB	
db_subst 変数名 鍵 置換	置き換え	テンプレ (スクリプト)	config
db_fget 変数名 フラグ	フラグのとりだし	DB スクリプト	
db_fset 変数名 フラグ 値	フラグの設定	スクリプト DB	
db_mtaget 変数名 フィールド	フィールドのとりだし	テンプレ スクリプト	
db_register テンプレート 変数名	変数の生成	元テンプレ テンプレ	config
db_unregister 変数名	変数の削除	テンプレ	
db_purge	データベースから削除	テンプレ、DB	postrm

表 1 debconf コマンド

4.3.1 テンプレートと変数

debconf では templates ファイルでテンプレートを定義し、そのテンプレートで作られる変数に対して debconf プロトコルを使って値を設定したり、読みだしたりしています。テンプレートを定義するとそれと同名の変数がつくれるので、ほとんどの場合ではテンプレート名と同名の変数を使うことがほとんどですが、同じような情報をいくつか持ちたい場合などではテンプレートから複数の変数を生成する場合があります。

テンプレートは debian/templates ファイルもしくは debian/パッケージ名.templates ファイルに記述します。

```

Template: 名前
Type: 型
Default: デフォルト値
Description: 短かい説明
長い説明

```

名前としては基本的には「パッケージ名/識別子」という文字列を使います。パッケージ名がプレフィクスについているので、他のパッケージのテンプレートと衝突することはありません。もし複数のパッケージで共有するようなテンプレートは「share/共有名/識別子」のような名前をつかうことが推奨されています。

型としては次のようなものがあります。

型名	意味
string	文字列
boolean	true か false
select	Choices:に設定されている値から一つ (“,” で区切る)
multiselect	select と違い複数選べる
note	Description:の内容を提示するだけ。メールとしても送られる
text	Description:の内容を提示する
password	パスワード用。

表 2 テンプレートの Type

debconf で使うメッセージは翻訳する場合、debconf-gettextize を使って debian/po/ディレクトリ以下に各国語版の po ファイルを置けるように準備をします。

```
% debconf-gettextize templates

To complete conversion, you must:
a. Check that new templates files contain all the localized fields
b. Add po-debconf to Build-Depends or Build-Depends-Indep in debian/control
c. Remove obsolete files:
    templates.old
d. Edit debian/rules to generate the localized templates file, either
    with dh_installdebconf or directly with po2debconf
```

このように templates ファイルを指示して debconf-gettextize を実行すると、元の templates ファイルは .old サフィックスがついたものに残され、新しい templates ファイルが作られます。新しい templates ファイルは、翻訳すべきフィールドは “_” がプレフィックスとしてつくようになります。そして po ディレクトリに POTFILES.in と templates.pot が生成されます。翻訳する場合は、templates.pot をロケール名.po にコピーして gettext を翻訳するのと同じようにして翻訳していきます。マルチパッケージで複数の templates ファイルがある場合は、それをすべて debconf-gettextize 実行時に指示します。古い templates.old は消してしまってもかまいません。

このようにして作られる翻訳を deb パッケージにちゃんととりこめるようにするには、まず debian/control の Build-Depends(か Build-Depends-Indep) に po-debconf を追加します。

後は debian/rules の binary-arch、binary-indep ルールで dh_installdebconf を dh_installdeb の直前くらいに呼ぶようにしておけば後は dh_installdebconf がしかるべき処理をしてくれます。cdbs を使っている場合は include /usr/share/cdbs/1/rules/debhelper.mk すれば中で dh_installdebconf を実行してくれます。また Depends: 行に \$misc:Depends をいれるのを忘れないようにしましょう。dh_installdebconf がしかるべき依存関係を設定してくれます。

L10N バグ (po 翻訳) を受けとった時は、そのファイルを debian/po ディレクトリにしかるべき名前 (ロケール名.po) で置くだけで OK です。

4.3.2 config スクリプト

config スクリプトは、deb パッケージがインストールされる前に^{*14}実行されます。config スクリプトでやるべきことは基本的には以下のような処理になります。

```
#!/bin/sh -e
# sample config
#
. /usr/share/debconf/confmodule
db_version 2.0
db_capb multiselect
if [ -f /etc/default/mypackage ]; then
. /etc/default/mypackage
db_set mypackage/foo "$FOOR"
db_set mypackage/bar "$BAR"
db_go
fi

db_title "My Package Configuration"
db_input low mypackage/foo || true
db_input low mypackage/bar || true
db_go
```

config スクリプトで使う debconf コマンドは以下のとおり

- . /usr/share/debconf/confmodule

まず /usr/share/debconf/confmodule を . でよみこみます。これで frontend と backend のプロセスにわかれそれぞれ通信するようになります。ちなみに frontend は perl で書かれていて open2 でスクリプトを起動しておいて通信しています。なお debconf シェルモジュールではすべてのコマンドには db_ が前についてるが、プロトコル上はこの db_ は実際にはつけません。

- (必要があれば) db_version でバージョンチェック。

通信に使う debconf プロトコルのバージョンをネゴシエーションします。これは「version 2.0 で通信したい」

^{*14} 実際には postinst などから confmodule が読みこまれた時にも

といっているわけで、それが debconf でサポートできないようなバージョンだとリターンコード 30 を変数 \$RET にいれてかえします。この場合エラーになるので sh -e によりここで実行が終了します。このコマンドは postinst なんかも使います。

- (必要があれば)db_capb で capability チェック
必要とするキャパビリティを要求します。指定できるのは backup および multiselect のふたつ。backup は前の質問に戻るをサポート、multiselect は複数のセレクトをできるようにする機能です。
- 設定ファイル^{*15}があるかどうかをチェックして、設定ファイルで指定されている現在の設定をよみます。もしあれば、その値で db_set して、debconf 変数の値に反映させます。

```
db_set 変数名 値
```

db_set はユーザとのやりとりは発生しません。

- db_title でタイトルの設定
質問ダイアログを出すときのタイトルを設定します。
- db_input でユーザに聞く質問を指示

```
db_input プライオリティ 変数名 || true
```

もしプライオリティが debconf 設定のプライオリティより高ければ変数名に対応しているテンプレートを使ってユーザにデータを入力させるようにする。つまり db_input low だとその質問はあまり重要ではなく細かい設定がしたい人だけがやればいいというような項目になります。逆に db_input critical だとユーザが与えないとまともな設定ができないような項目になります。

プライオリティ	意味
low	ほとんどのケースでデフォルト値で問題ない場合
medium	まともなデフォルト値がある場合
high	まともなデフォルト値がない場合
critical	デフォルト値では問題があるような場合。ユーザの入力が必要

表 3 プライオリティ

どのようにデータを入力するかはそのテンプレートにわりあてられている型によってかわります。

もしユーザに提示しなかったらエラーで \$RET が 30 になります。そのため通常は —— true として sh -e によりここでスクリプトが終了しないようにします。

なお、db_input だけでは入力の指示をフロントエンドに送るだけで、質問自体はまだおこなわれません。実際にフロントエンドがユーザにたずねるのは db_go を呼び出した時です。

- db_go
db_input で与えられてきた「質問してよー」というのを実際にユーザに提示します。ここでダイアログができてユーザとやりとりすることになります。

debconf は一度、db_input で入力した場合、その変数の seen フラグを true に設定します。このフラグの値を変更するには db_fset を使います。

```
db_fset 変数名 seen false
```

デフォルト値に戻したい場合は db_reset を使います。これでテンプレートに記述されていたデフォルト値に戻すことができます。

```
db_reset 変数名
```

基本的にはテンプレートと同名の変数を使うことで事足りる場合が多いですが、必要があればテンプレートから新しい変数を作ったりすることができます。変数をつくったりけしたりする操作は db_register、db_unregister を使い

^{*15} 通常/etc/default/パッケージ

ます。

```
db_register テンプレート 変数名
```

```
db_unregister 変数名
```

また、変数を新たにつくった場合は質問のメッセージの一部を変えることが多いでしょう^{*16}。そのために `db_subst` というのが使えます。テンプレートの中で `${ 鍵文字列 }` というのを埋めこんでいて、次のように `db_subst` を実行すると、`${ 鍵文字列 }` の部分が、“置換文字列”に置きかえられます。

```
Template: mypackage/baz
...
Description: ${鍵文字列} の値?
${鍵文字列}の値を入力してください
```

```
db_register mypackage/baz mypackage/baz2
db_subst 変数名 鍵文字列 置換文字列
```

```
...
Description: 置換文字列 の値?
置換文字列の値を入力してください
```

4.3.3 postinst スクリプト

`postinst` スクリプト^{*17}では `debconf` からデータをよみだして実際の設定ファイルを生成することが仕事となります。 `debconf` 以前では `postinst` 自身が `echo` や `read` コマンドなどをつかってユーザの入力をいれていたのをここでは `debconf` からとってくるようにするわけです。

```
#!/bin/sh -e
# postinst
. /usr/share/debconf/confmodule

case "$1" in
configure)
    db_get mypackage/foo
    FOO="$RET"
    db_get mypackage/bar
    BAR="$RET"
    if [ -f /etc/default/mypackage ]; then
        sed -e 's/^FOO=.*$/FOO="$FOO"/' \
            -e 's/^BAR=.*$/BAR="$BAR"/' \
            < /etc/default/mypackage > /etc/default/mypackage.dpkg-tmp
        if cmp -s /etc/default/mypackage /etc/default/mypackage.dpkg-tmp; then
            rm -f /etc/default/mypackage.dpkg-tmp
        else
            mv -f /etc/default/mypackage /etc/default/mypackage.dpkg-old
            mv /etc/default/mypackage.dpkg-tmp /etc/default/mypackage
        fi
    else
        cat <<DEFAULT > /etc/default/mypackage
# mypackage configuration file
# see /usr/share/doc/mypackage/README.Debian.
# this file is automatically managed by debconf.
#
# FOO="..."
# foo is blah blah
FOO="$FOO"
#
# BAR="..."
# bar is blah blha
BAR="$BAR"
# END OF FILE
DEFAULT
fi
;;
abort-upgrade|abort-remove|abort-deconfigure)
;;
*)
    echo "postinst called with unknown argument '$1'" >&2
    exit 1
;;
esac
db_stop
#DEBHELPER#
exit 0
```

`config` スクリプトで使う `debconf` コマンドは以下のとおり

^{*16} でないと、どれがどれかユーザにわからなくなってしまう

^{*17} `preinst` スクリプトはパッケージ展開前なので普通は `debconf` を使うことはない

- `. /usr/share/debconf/confmodule`

`config` スクリプトと同様、まず `/usr/share/debconf/confmodule` を、でよみこみます。注意すべきことは、ここで `config` スクリプトもまた実行されるということです。

- `db_get` で値のとりだし

db_get 変数名

`db_get` を実行すると、変数名であらわされる変数に格納されている値を `$RET` にとりだすことができます。

- `db_stop` で `debconf` の終了

ここで `debconf` の処理を終了させます。これ移行、標準入出力が使えるようになります。たとえば `invoke-rc.d` などで起動されるものがある場合、メッセージを標準出力にだそうとするのでこの前に `db_stop` しておかなければいけません。

`postinst` では、この例にあるように既存の設定ファイルをできるだけ維持しつつ、更新された設定値だけをいれかえるようにすることが期待されています。

4.3.4 postrm スクリプト

`postrm` スクリプトでは、このパッケージの `debconf` データを `debconf` データベースから削除しておく必要があります。

```
#!/bin/sh -e
#

case "$1" in
  purge
    . /usr/share/debconf/confmodule
    db_purge
    db_stop
    ;;
  remove|upgrade|failed-upgrade|abort-install|abort-upgrade|disapper)
    ;;
  *)
    echo "$0 called with unknown argument '$1'" >&2
    exit 0
    ;;
esac
#DEBHELPER#
```

- `. /usr/share/debconf/confmodule`

`debconf` を使う前にまず `/usr/share/debconf/confmodule` を、でよみこみます。

- `db_purge`

`db_purge` でこのパッケージに属する `debconf` データを `debconf` データベースから削除します。

- `db_stop`

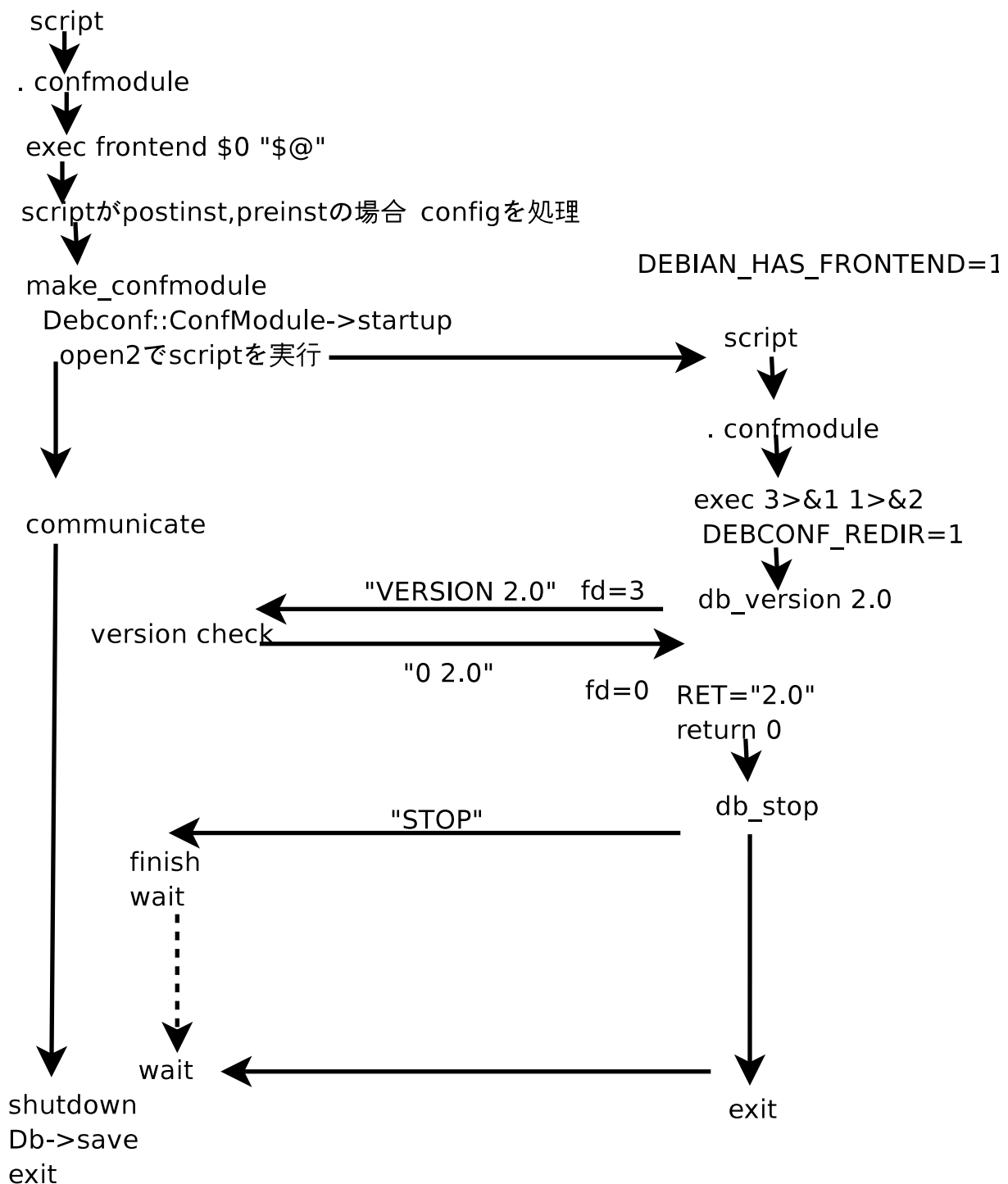
もう `debconf` は必要ないので `stop` します。

4.3.5 debconf プロトコルのやりとり

`debconf` は `confmodule` をソースすることで、`frontend` に `exec` しています。その中で `confmodule` を呼びだした `script` を `open2` で起動して `db_` コマンドを `frontend` との通信にして処理をおこなっています。

`fd=3` にコマンドを出力すると、`frontend` プログラムで解釈実行され結果がかえってきます。その結果はスクリプトの標準入力に与えられるので `read` で読みとってスペースの前がコマンドの終了コード、スペースの後が `$RET` に格納される値となります。

スクリプトがおわると `frontend` はそれを検知してデータベースに書き出して終了となります。



4.4 debconf データベース

debconf で設定したデータはどこにあるのでしょうか？ この設定は/etc/debconf.conf に記述されています。

デフォルトでは /var/cache/debconf に次のようなファイルとして格納されています。

debconf データベースの内容は debconf-get-selections を使うとダンプすることができます。この出力を debconf-set-selections の入力として渡すとロードすることができます。^{*18}

^{*18} debconf-get-selections は TAB で分割していて、debconf-set-selections では (特にタイプと値の間は) 空白文字列一つだけという点に注意したほうがいいでしょう。ファイル経由でやりとりしたりパイプ経由の場合は問題ありませんが、ターミナルの出力をコピーするとはまる場合があります (TAB が複数のスペースになってしまっているの)

ファイル名	内容
templates.dat	templates の情報
config.dat	設定データ
password.dat	パスワードデータ

表 4 debconf データベース

debconf-get-selections -installer で、debian-installer で使って debconf データベースをとりだすことができます。その他に debconf-copydb を使うことでデータベースファイルをコピーすることができます。

4.5 debconf を使っているスクリプトのデバッグ

debconf を使ってるスクリプトのデバッグは簡単ではありません。しかし基本は DEBCONF_DEBUG に値を設定してスクリプトを実行すればいい場合が多いでしょう。

```
# DEBCONF_DEBUG='.*' /var/lib/dpkg/info/パッケージ.postinst configure 最後に設定されたバージョン
```

これでパッケージの設定する時の debconf の動きを追うことができます。なお簡単にするために dpkg-reconfigure debconf で debconf フロントエンドを Readline などをしておいた方がいいでしょう。

DEBCONF_DEBUG='.*' というのは DEBCONF_DEBUG=user、DEBCONF_DEBUG=developer、DEBCONF_DEBUG=db すべてを指定したのと同じ意味になります。

debconf-communicate を使うと、debconf データベースと直接やりとりすることができます。標準入力からコマンドを与えると、標準出力に結果をかえします。コマンドは debconf シェルモジュールでつかうコマンドから db_ をとりのぞいたものになることに注意しましょう。例えば次のように使います。

```
# echo 'get debconf/priority' | debconf-communicate
0 critical
```

0 が成功を意味^{*19}し、critical が debconf/priority の値^{*20}を意味します。

4.6 おわりに

本文書では Debian で設定データの管理を司っている debconf について解説しました。

^{*19} db_get の終了値

^{*20} db_get を実行した場合だと \$RET に設定される値

5 グループワーク用メモ



6 次回



次回は 10 月 15 日土曜日の夜を予定しています．内容は本日決定予定です．
参加者募集はまた後程．