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

上川純一

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

目的として下記の二つを考えています．

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

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

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

1.1 講師紹介

- 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．
- 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

1.2.2 上川

あとで埋める予定．

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??号

問題 1.

☐ 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 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.

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い、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在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と 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るようになります。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,readlline,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-extracttempaltes 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 tempaltes 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 として使うな[*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るわけです。

[*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_metaget 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られるので、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なおして通信しています。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すます。このコマンドは postinst なんかでも使います。

- (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とります。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 を呼びだした時です。
- 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て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 ${ 鍵文字列 } というのを埋めこんでおいて、次のように db_subst を実行すると、${ 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
 purse
   . /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#
```

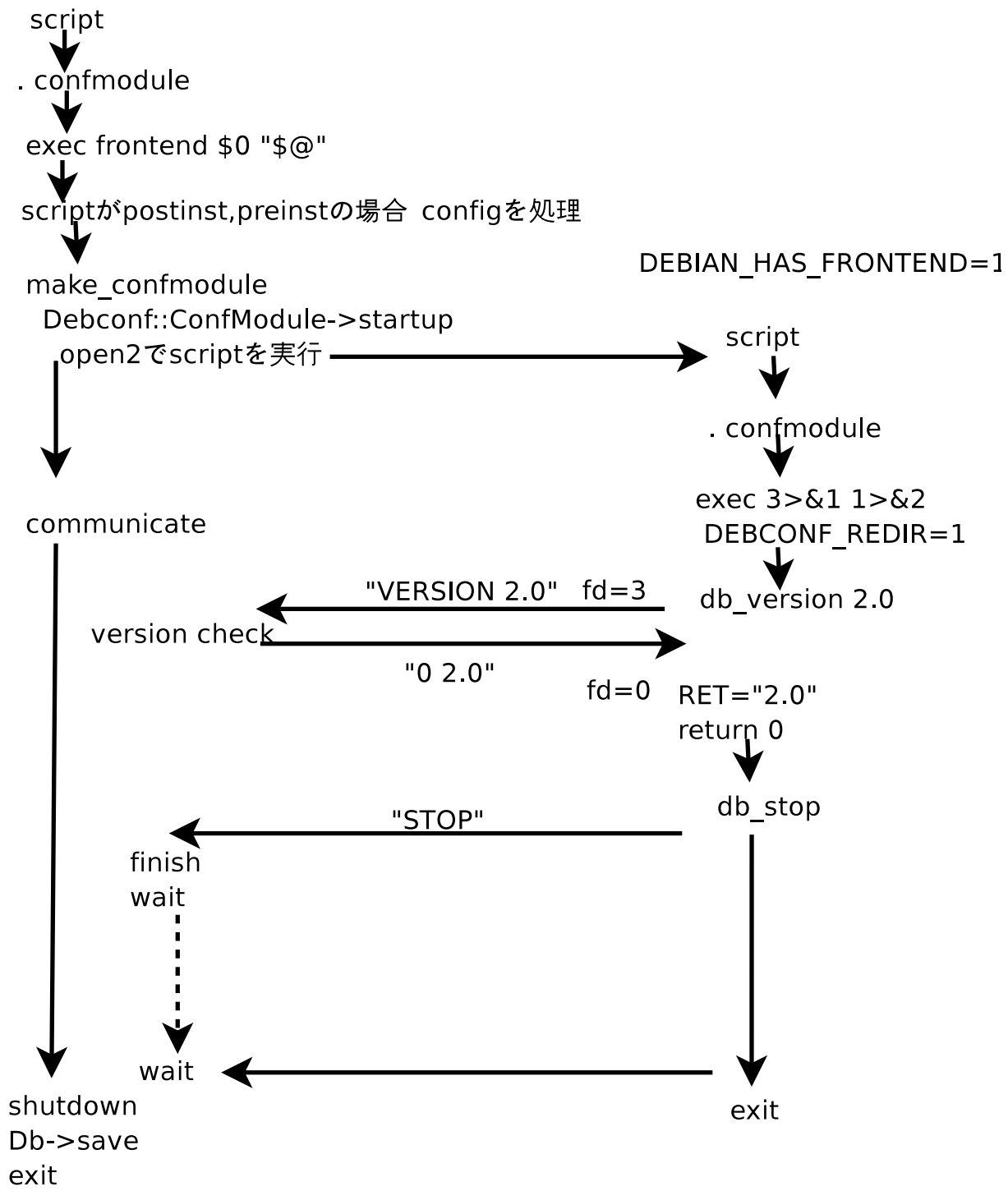
- . /usr/share/debconf/confmodule
  debconf を使う前にまず/usr/share/debconf/confmodule を. でよみこみます。
- db_purge
  db_purge でこのパッケージに属する debconf データを debconf データベースから削除します。
- db_stop
  もう debconf は必要ないので stop します。

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ペするとはまることがあります (TAB が複数のスペースになってしまっているので)

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 などをに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 ???

上川？

6 Keysigning Party

上川純一

事前に必要なもの

- 自分の鍵の fingerprint を書いた紙 (`gpg --fingerprint XXXX` の出力．)
- 写真つきの公的機関の発行する身分証明書，fingerprint に書いてある名前が自分のものであると証明するもの

キーサインで確認する内容

- 相手が主張している名前の人物であることを信頼できる身分証明書で証明しているか[*21].
- 相手が fingerprint を自分のものだと主張しているか
- 相手の fingerprint に書いてあるメールアドレスにメールをおくって，その暗号鍵にて復号化することができるか

手順としては

- 相手の証明書を見て，相手だと確認
- fingerprint の書いてある紙をうけとり，これが自分の fingerprint だということを説明してもらう
- (後日) gpg 署名をしたあと，鍵のメールアドレスに対して暗号化して送付，相手が復号化してキーサーバにアップロードする (`gpg --sign-key XXXXX`，`gpg --export --armor XXXX`)

[*21] いままで見た事の無い種類の身分証明書を見せられてもその身分証明書の妥当性は判断しにくいため，学生証明書やなんとか技術者の証明書の利用範囲は制限される．運転免許証明書やパスポートが妥当と上川は判断している

7 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．