

A large, stylized pink brushstroke graphic that forms a partial circle, framing the text on the slide.

東京エリア Debian 勉強会

第91回 2012年8月度


上川純一

dancer@debian.org

2012年8月18日

設営準備にご協力ください。

会場設営よろしくおねがいします。



Agenda

- 注意事項
 - 飲食禁止
 - 宗教禁止
 - 営利活動禁止
- 最近あった Debian 関連のイベント報告
 - 第 90 回 東京エリア Debian 勉強会
 - Debian Conference 2012 参加報告
 - 月刊 Debhelper
 - ソフト開発以外の簡単 Debian contribution(ドラフト版!)
 - Debian での C++11 開発環境



事前課題

19周年にちなんで、ということでもないが、今年は Haskell のライブラリを公開する予定なので、自分でパッケージングしてメンテナになりたい。あと Debian Haskell チームにも参加していきたい。

なかおけいすけ

変わらず使い続けます。で、ちょっとずつ、Contribute していこうと思っ
てます。


抱負: 「我がああ Debian の~力(りょく)はああああ世界いいいいチイイ
イイイイ!!!! 」とか言えると素敵かも...(~ の部分は適当に...)
ちょっとぐらい他の人がやった事がないこと1つでも Debian ネタで出来たら
いいな... 「名状しがたい何かのハードウェア」で Debian 動かしてみると
かかな...

キタハラ

諸般の都合で廃止されてしまった、会社の debian サーバを復活させたいですね。

kfreebsd を使って楽しみながら、deb していきます。

DebConf を日本で開催できるならば、是非実現させたい。





Debian Conference
2012 参加
報告



月刊 Deb-
helper

今回のお題は!

- dh_makeshlibs
- dh_shlibdeps



今回のお題は!

- dh_makeshlibs
- dh_shlibdeps

俺にとっては、まさに魔窟!秘境!
(ズガン!)

地図を手に入れた! ピロ〜ン

共有ライブラリの構造と流儀、動的リンクの方法、オブジェクトファイルの構造、deb パッケージの構造について。

- ① John R. Levine 著 榊原 一矢/ポジティブエッジ 訳, 「 Linkers & Loaders 」, ISBN10 4274064379
- ② 坂井 弘亮著, 「 リンカ・ローダ実践開発テクニック 」, ISBN10 4789838072
- ③ 高林 哲ら著, 「 Binary Hacks 」, ISBN10 4873112885
- ④ Junichi Uekawa 著, 「 Debian Library Packaging guide 」, <http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html>
- ⑤ man 5 deb もしくは wikipedia の deb(ファイルフォーマット)

今回のコマンドはそもそも何の為に!?

バイナリとバイナリに必要な共有ライブラリとの関係を適切に自動生成する為
(依存関係算出はそもそも人手では「もう、やっとなれんわーっ」という量にすぐになる為...)

「依存関係地獄 (Dependency Hell)」からの攻撃!

まあ、aptitude show とかで見ればわかる気がします...

```
$ aptitude show gnome-shell
... 中略...
依存: gir1.2-atk-1.0, gir1.2-clutter-1.0 (>= 1.9.16),
      gir1.2-cogl-1.0,
      gir1.2-coglpango-1.0, ... 中略...
      gnome-bluetooth (>= 3.0.0),
      libatk1.0-0 (>= 1.12.4), libc6 (>= 2.7),
      libcairo-gobject2 (>= 1.10.0),
      libcairo2 (>= 1.10.0), libcanberra0 (>= 0.2),
      libclutter-1.0-0 (>= 1.10.0),
      libcogl-pango0 (>= 1.7.4),
      libcogl9 (>= 1.7.4), libcroco3 (>= 0.6.2),
      libdbus-1-3 (>= 1.0.2),
... 「依存」の行は「 まだまだまだ」続くぜ...
```

... 祖はまさに「地獄」の名にふさわしい...

debian/control ファイル中のマクロ(その1)

ソースパッケージに入ってる debian/control 見ると...

```
$ apt-get source gnome-shell
$ cd gnome-shell-3.2.1
$ less debian/control
... 中略...
Package: gnome-shell
Architecture: any
Depends: ${gir:Depends},
         gjs (>= 1.29.18),
         ${shlibs:Depends},
         ${misc:Depends},
... 中略...
```

なんか、マクロっぽいものがいっぱい。dh_shlibdeps は、この場合、`${shlibs:Depends}` を置き換えて、将来構築する DEBIAN/control (deb パッケージに入る制御情報っすね) を生成します。


debian/control ファイル中のマクロ(その2)

補足: dh_shlibdeps は能力的には、

- `${shlibs:Depends}`
- `${shlibs:Suggests}`
- `${shlibs:Recommends}`
- `${shlibs:Pre-Depends}`

を依存関係算出して書き換える事が可能。が、
`${shlibs:Depends}` 以外は、どう使うのか自分的にはまだわからず..

次のスライドからとりあえず追ってみます。



召喚! objdump その1

バイナリ/共有ライブラリが必要としている他の共有ライブラリの情報一覧

```
$ objdump -p /bin/ls | fgrep -A5 'Dynamic Section:'
```

```
Dynamic Section:
```

```
NEEDED          libselinux.so.1
NEEDED          librt.so.1
NEEDED          libacl.so.1
NEEDED          libc.so.6
INIT            0x0000000000402298
```

```
$ objdump -p /lib/x86_64-linux-gnu/libglib-2.0.so.0 |  
fgrep -A5 'Dynamic Section:'
```

```
Dynamic Section:
```

```
NEEDED          libpcre.so.3
NEEDED          libpthread.so.0
NEEDED          librt.so.1
NEEDED          libc.so.6
SONAME          libglib-2.0.so.0
INIT            0x000000000001ba40
```

召喚! objdump その2

キーワード:

NEEDED この項目で指定される SONAME を持つライブラリが必要という意味

SONAME ローダがライブラリを探すときに使う共有ライブラリの名前。本当の名前 (real name) というのも別であるからややこしい... (細かいことというと、本当は soname は、libc.so.6 なら、'c' が soname。lib+'soname'+'.so.'+'バージョン番号'となる。が、ここでは大文字で SONAME とすればこちらのキーワードの値を指すことにする)

召喚! objdump その3

ライブラリの中の何が必要か? を表示してみる。

```
$ objdump -T /bin/ls
... 中略...
DYNAMIC SYMBOL TABLE:
... 中略... GLIBC_2.3    __ctype_toupper_loc
... 中略... GLIBC_2.2.5 getenv
... 中略... GLIBC_2.2.5 sigprocmask
... 中略...  GLIBC_2.2.5 raise
... 中略...
$
```

(ちょっと、行長くてプレゼン資料に入りきらないので、各行の頭の部分は「... 中略...」とさせてもらってます)

召喚! shlibs ファイル その1

すぐ思いつく(かもしれない)発想:
とりあえず、objdump -p の結果の NEEDED の SONAME から
直接依存関係起こしちゃえばいいんじゃないかね??

召喚! shlibs ファイル その2

で、shlibs ファイルが作られた。

```
$ dpkg-query --control-path libgtk2.0-0
/var/lib/dpkg/info/libgtk2.0-0:amd64.shlibs
... 中略...
$ cat /var/lib/dpkg/info/libgtk2.0-0:amd64.shlibs
libgtk-x11-2.0 0 libgtk2.0-0 (>= 2.24.0)
libgdk-x11-2.0 0 libgtk2.0-0 (>= 2.24.0)
udeb: libgtk-x11-2.0 0 libgtk2.0-0-udeb (>= 2.24.0)
udeb: libgdk-x11-2.0 0 libgtk2.0-0-udeb (>= 2.24.0)
```

意味は、例えば libgtk-x11-2.0.so.0 という SONAME を見つけたら、依存情報として、「 libgtk2.0-0 (>= 2.24.0)」と書け! という単純な構造。

(上の例の1行めの意味は libgtk-x11-2.0+'.so.'+0 が SONAME という意味)

依存関係地獄は自動でやっつけれる!

召喚! shlibs ファイル その3

debian-policy マニュアルでは、

```
$ git clone \  
  http://anonscm.debian.org/git/dbnpolicy/policy.git
```

すると shlibs について定義が読めるよ!

依存関係地獄復活! その1

一見よさそうに見える shlibs を使った方法は実は以下の問題が...

- 実はこの shlibs に記載されている依存情報はパッケージ毎に固定のデータなので、ライブラリパッケージのバージョンが上がると、ライブラリに新たに追加されたシンボルを利用するバイナリの為に、依存情報にあるバージョンを上げざるを得ない。すると、新たに追加されたシンボルを使う必要の無いバイナリのパッケージは依存関係が満たせなくなってしまうのでインストール出来なくなる。

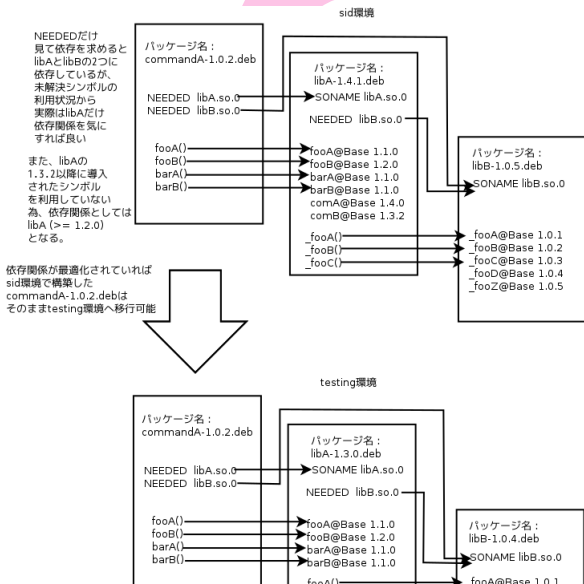
依存関係地獄復活! その2

- NEEDED に記載されている SONAME は、共有ライブラリのみが依存している他の共有ライブラリですら含んでしまっている。そのため、バイナリから見れば、直接リンクに関係無い共有ライブラリのバージョンが変わっても、パッケージはインストール出来なくなる。

このままでは、sid 環境から、testing 環境へバイナリパッケージだけを移動して動作させる事ができなくなっていく...

依存関係地獄復活! その3

本当はこうなると良かった...



魔法使い (Wizard) 現る! その1

Raphael(e はウムラウトね) Hertzog さんにより、

- shlibs ファイルを使った依存関係の問題と解決の為の提案がなされる。

Projects ImprovedDpkgShlibdeps

<http://wiki.debian.org/Projects/ImprovedDpkgShlibdeps>

- 2007/9/25 に、debian-devel-announce に、この問題を解決する為の改造を施した dpkg コマンド群のリリースがアナウンスされた。

New dpkg in experimental

<http://lists.debian.org/debian-devel-announce/2007/09/msg00004.html>

魔法使い (Wizard) 現る! その2

参考: 関係ないけど、Hertzog さんの blog サイトの名前は:
“apt-get install debian-wizard”

<http://raphaelherzog.com/>

...The DEBIAN ADMINISTRATOR'S HANDBOOK の著者です
ね...

召喚! symbols ファイル その1

さらに賢い依存関係を算出するために、symbols ファイルが導入された。

```
$ dpkg-query --control-path libgtk2.0-0
... 中略...
/var/lib/dpkg/info/libgtk2.0-0:amd64.symbols
... 中略...
$ cat /var/lib/dpkg/info/libgtk2.0-0:amd64.symbols
libgdk-x11-2.0.so.0 libgtk2.0-0 #MINVER#
* Build-Depends-Package: libgtk2.0-dev
  gdk_add_client_message_filter@Base 2.8.0
  gdk_add_option_entries_libgtk_only@Base 2.8.0
  gdk_app_launch_context_get_type@Base 2.14.0
... 中略...
```

共有ライブラリの Export しているシンボルに、最初にシンボルが現れた時の deb パッケージバージョンがついている。

召喚! symbols ファイル その2

前ページの例でいくと、バイナリに含まれる NEEDED に、libgdk-x11-2.0.so.0 が入っていれば、依存情報として、"libgtk2.0-0 #MINVER#" を記載しなさいという意味。

ここで、"#MINVER#" は、バイナリが必要とするシンボルが全部含まれるときのバージョンで置き換える。例えば、バイナリが gdk_add_client_message_filter, gdk_add_option_entries_libgtk_only, gdk_app_launch_context_get_type のみを利用しているのであれば、"libgtk2.0-0 (>= 2.14.0)" を依存関係として記載することになる。

召喚! symbols ファイル その3

さらに、バイナリのNEEDEDに、libgtk2.0のみが利用している他のライブラリが記載されていたとしても、バイナリの必要とするシンボルが含まれていなければ、依存関係の候補として除外する。

⇒ 以上が dpkg コマンド群の dpkg-shlibdeps に実装された。dh_shlibdeps は新しい版の dpkg-shlibdeps をまるごとそっくり利用して依存関係を更新する。

依存関係地獄は滅んだ!


補足: このやり方だと、ライブラリのバージョンが上がってシンボルが廃止になった場合は、自動で対応出来ないので、注意。

dh_makeshlibs は、パッケージ構築ディレクトリにある構築完了した共有ライブラリから shlibs ファイルを新規に生成し、同じように symbols ファイルをメンテナが用意したものから、新しいものへ更新するだけの debhelper プログラムです。なお、symbols ファイルの更新に、dpkg-gensymbols コマンドをそのまま用いています。

なお、前ページで述べた問題を避ける為、dh_makeshlibs は、symbols ファイルから、何かシンボルが消えた事を検知すると、エラー終了するように出来ています。(何が消えたかは diff 形式で標準出力に出力されます)

dh_makeshlibs の説明は以上っ

次回の発表者は...?





ソフト開発
以外の簡単
Debian con-
tribution(ド
ラフト版!)

ソフト開発以外の例 その1

- DDTSS の日本語訳をレビューしてみる、日本語に訳してみる。
(詳しくは「第53回東京エリア Debian 勉強会資料」
<http://tokyodebian.alioth.debian.org/pdf/debianmeetingresum200906.pdf> を参照ください。非常に良い解説とチュートリアルとなっています。)
- BTS へ BUG 報告/何かの改善提案をしてみる。
(詳しくは「第43回関西 Debian 勉強会資料」
<http://wiki.debian.org/KansaiDebianMeeting/20110123> を参照ください。非常に良い解説とチュートリアルとなっています。)

ソフト開発以外の例 その2

- <http://debtags.debian.net/edit/> で debtags をひたすら打ち込む。
(詳しくは「第63回東京エリア Debian 勉強会資料」
<http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201004.pdf> を参照ください。非常に良い解説とチュートリアルとなっています。)
- <http://screenshots.debian.net/> ヘスクリンショットを投稿しまくってみる。
- <http://www.debianart.org/cchost/> ヘオリジナルの絵を送ってみる。
- debian-doc@debian.or.jp にて投稿された翻訳のレビューをしてみる、未だ日本語に訳されていない文章を翻訳して投稿してみる。
- debian-users@debian.or.jp にて投稿された質問に回答をしてみる。

- synaptic パッケージでパッケージ調べて「 screenshots を見る 」を押すと screenshots.debian.net からスクリーンショットを落として表示します。
- ユーザ登録もなしに、誰でもスクリーンショットを登録できます。(軽い審査はされますが...)

使い方は、東京エリア Debian 勉強会資料に記載しました。超簡単に投稿出来ます。

自分もやってみた。

実例:

- renpy
`http://screenshots.debian.net/package/renpy`
- renpy-demo
`http://screenshots.debian.net/package/renpy-demo`
- zaz
`http://screenshots.debian.net/package/zaz`

等等... (実はゲームばかりです)

- Debian に使われる様々なグラフィック/サウンドデータの投稿サイト。直近では、wheezy の起動画面の背景画像、GNOME 環境のデフォルトの背景画像などのコンテストが行われた。
- 最近では、Debian プロジェクトにマスコットキャラクターが必要ということで、マスコットキャラクターの図案の募集が行われていたようです。
- デスクトップ用のファンファーレなどのサウンドアイコンもありました。

絵心のある方/写真撮影に興味のある方/サウンドアイコンに興味ある方で、我こそはという方はぜひ活用すると良いかもしれません。

補足として、投稿したもの、提供したものは基本フリーなライセンスとなります。ですので、以下は注意点となります。

- ライセンスで揉めそうなものは基本NGです。
例：有料ゲームのエミュレータのスクリーンショット、ライセンスがよく分からない画像を利用などは避けるべきかと...

ではネタだしを...

他にあつたらぜひ教えて



Debian での
C++11 開発環境

- 2011 年 2 月、ISO/IEC 委員会にて C++11 策定
- c++0x と呼ばれていた頃から各コンパイラーがドラフトを実装していたので完全ではないが準拠している
- Debian で使ってみた

wheezy での c++ コンパイラー

- g++ 4.7
<http://gcc.gnu.org/projects/cxx0x.html>
- clang++ 3.0
http://clang.llvm.org/cxx_status.html

```
# apt-get install g++ clang
```


コードの例

```
#include <iostream>
using namespace std;

int main(int argc, char **argv) {
    int hoge[] = { 1, 10, 12, 15};
    const char* fuga[] = { "hello", "world", "this is a", "message" };
    for (const auto& i : hoge) {
        cout << i << endl;
    }

    for (const auto& str : fuga) {
        cout << str << endl;
    }
    return 0;
}
```

コンパイルしてみる

```
$ g++ --std=c++11 auto.cc  
$ clang++ --std=c++11 auto.cc
```

C++プログラミングするときにあると便利な参考文献

- `stl-manual`
- `libstdc++6-4.7-doc`
- `gcc-doc`
- `libboost1.49-doc`
- `manpages-dev`
- <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3242.pdf>
- <http://www.stroustrup.com/C++11FAQ.html>

- <http://wiki.debian.org/SummerOfCode2012/StudentApplications/AndrejBelym>
- libstdc++ switched to GPL3.
- C++11