Grand Unified Debian

銀河系唯一のDebian専門誌

東京エリア/関西Ｄｅｂｉａｎ勉強会

あんどきゅめんてっど でびあん 2017 年冬号 2017 年 12 月 29 日 初版発行

デビアン勉強会

目次

1 Introduction

DebianJP

1.1 東京エリア Debian 勉強会

Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、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？

Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での “開発に関する情報” を整理してまとめ、アップデートする。
- 場の提供。
  - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
  - Debian のためになることを語る場を提供する。
  - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、 “場” としての空間を提供するのが目的です。

1.2 関西 Debian 勉強会

関西 Debian 勉強会は Debian GNU/Linux のさまざまなトピック (新しいパッケージ、Debian 特有の機能の仕組、Debian 界隈で起こった出来事、などなど) について話し合う会です。

目的として次の三つを考えています。

- ML や掲示板ではなく、直接顔を合わせる事での情報交換の促進
- 定期的に集まれる場所
- 資料の作成

それでは、楽しい一時をお楽しみ下さい。

2 Debian で Lisp を動かす

油谷知岐

2.1 はじめに

Lisp という由緒正しきプログラミング言語があります. 原初のそれは 1958 年に現れ, 2017 年現在で実に 60 年の歴史の中で洗練されてきた言語であり, Debian と同様に非常に自由なプログラミングを旨とする思想を持っています. また, Debian パッケージとして提供されている Lisp 処理系も数多くあり, その親和性は低くないと考えられます. ここでは, そんな素晴らしい言語 ”Lisp ”とはどんな言語なのかを簡単に説明します.

2.2 Lisp とは

Lisp という言語は, 御存知の通り, List Processor から命名されたものです. この言語は, プログラム自体のデータ構造がリストであるなど, データ構造として「リスト」がプログラム全体で非常に重要な役割を果たしています. プログラム自体が, 「プログラム（ソースコード）」でありながら, 「（リスト構造を持った）データ」でもあるため, 自分自身を操作しやすいという特徴を持ちます. また再帰・条件分岐・GC・動的型付けなど, 現在の PG 言語の多くが備える仕組みの先駆けとなっているという歴史も持っています.

この言語はプログラム全体で「()：括弧」が多用される, 他の言語とは異なる見た目をしているように見えるため敬遠されがちなようです. しかし, 慣れるとむしろ括弧が目印になり, ソースコードの塊がわかりやすいなどの利点もある上に, 非常に自由度の高いコーディングを可能にするという利点もあります.

また, Lisp という言語を代表する強力な仕組みの一つとして, プログラムを生成するプログラムを記述するマクロシステムが良く話題にのぼる. このあまりにも強力な仕組みもまた, Lisp という言語を難解であるかのように感じさせる原因の一つになっているようです.

2.3 Lisp の歴史

Lisp という言語は長い歴史の中で様々な処理系/仕様が「方言」として採択されており, 少しずつ変化しながら発展している. 大まかな変化の系列は Wikipedia の Lisp のページに以下のように記載されています. いつの時代も Lisp がなくなることはなく, 形を変えて生き続けていることがわかります.

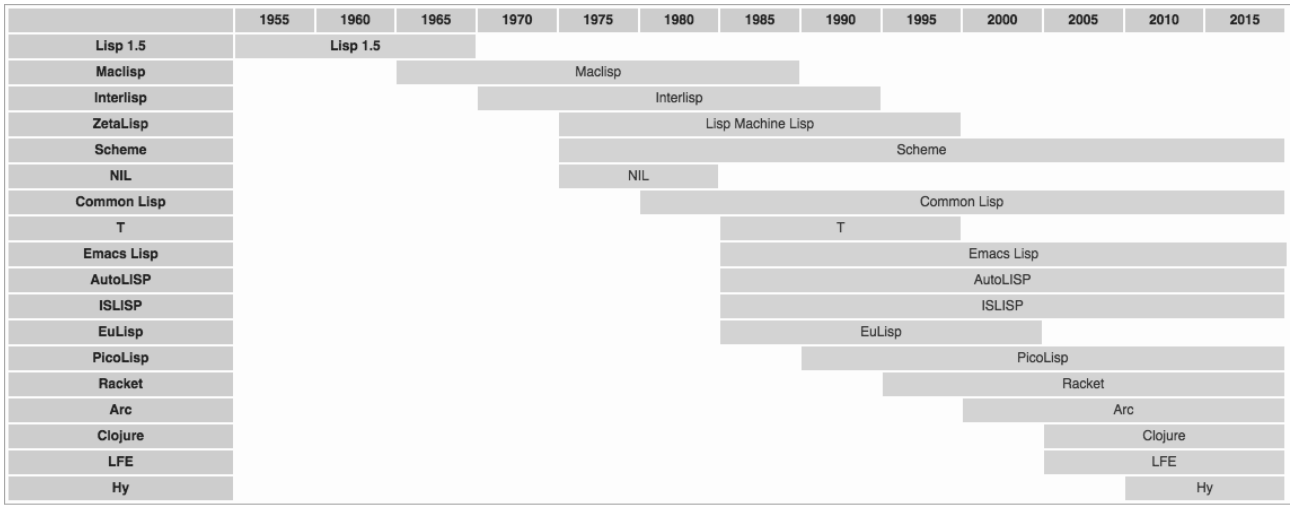


図 1　Timelines of Lisp dialects (Wikipedia: Lisp(programming language) のページより

`https://en.wikipedia.org/wiki/Lisp_(programming_language)`

Lisp は自己書き換えが容易であるという特徴のため, いわゆる人工知能の研究開発によく用いられていました.

2.4　大まかな Lisp 方言の種類

2.4.1　代表的 Lisp 方言とその特徴

表 1 に 2017 年現在で主流である 6 種類の Lisp 方言をまとめています. ただしこれらはあくまでもほんの一握りであり, 他にも現在動いている Lisp 方言は大量にあります.

現在用いられている主なLisp方言		名前空間	その他
方言名	出現時期		
Emacs Lisp	1958年	関数と変数は異なる名前空間を持つ(Lisp-2)	・最もよく知られているLisp方言 ・Emacsさえあれば動く ・速度はあまりでない
CommonLisp	1994年(2回目の標準化)	Lisp-2	・ライブラリが充実しつつある ・強力な動的オブジェクトシステム(CLOS)
Scheme	1975年	関数と変数が同じ名前空間を共有する(Lisp-1)	・CommonLispより関数型によっている ・CLよりシンプルな仕様
ISLisp	2007年(2回目の標準化)	Lisp-2	・ダイナミックスコープを表す語彙の明確化 ・CLのサブセット感が強い(完全にそうというわけではない)
Clojure	2007年	Lisp-1	・JVM上で動作 ・並列コンピューティング
Hy	(現在まだVer0.13)	Lisp-1	・Python VM上で動作する ・Clojureに似た記法

表 1　現在主流な Lisp 方言

JVM や PythonVM で動く Lisp, ErlangVM やその他様々な言語と連携するものや, 独自の機能を追加縮小したものなど, 現在でも新たな方言が活発に生まれています.

表中の名前空間のカラムは, その方言がサポートする名前空間の扱いについて言及しています. Lisp の方言には大きく 2 種類の名前空間の扱いがあります. 一つは, 関数名と変数名の名前空間を同一のものとして扱う Lisp − 1 という枠組みです. こちらの名前空間では, 関数名と同じ名前の変数を宣言すると, 当然関数の定義が変数の定義に上書きされることになります. 具体的には以下のようになります.

```
;;; 関数定義
(define sample
        (lambda (x) (* x x)))

;;; 関数の評価
(sample 2)          ; 4

;;; 変数として？評価
sample              ; #<Closure>

;;; 変数の束縛
(define sample "It's a sample valiable")

;;; 変数の評価
sample              ; "It's a sample valiable"

;;; 関数として評価
(sample)            ; Error: "It's a sample valiable" is not a function [sample, *, sample]
```

一方, 関数名と変数名それぞれに異なる名前空間を用意するのが Lisp-2 と呼ばれる枠組みです. 実際には以下のように動作します.

```
;;; 関数定義
(defun sample ()
        (format t "It's a sample function"))

;;; 関数の評価
(sample) ; It's a sample function
             ; NIL

;;; 変数の評価
sample   ; Error

;;; 変数の束縛
(defparameter sample "sample parameter")

;;; 変数の評価
sample   ; "sample parameter"

;;; 関数の評価
(sample) ; It's a sample function
             ; NIL
```

これらの違いから Lisp-1/Lisp-2 どちらが良いかといった考え方の違いが生まれたりします. 私個人としては Lisp-2 のほうが扱いやすいように感じています.

2.4.2 CommonLisp と Scheme の代表的処理系

表 2 には, Lisp 方言の中でも現在最もよく使われている 2 大方言について, 幾つかの代表的な処理系を示します. Lisp を学びはじめの時期は, 「処理系」と「方言」とを混同して考えがちな気がしますので, 簡単に説明しておきます. 特に CommonLisp と Scheme については, 非常に多くの処理系が存在しますので, その中の一部を取り上げて説明します.

処理系それぞれに依存した機能が少なからず存在するため, プログラムの可搬性を考える場合は, 何がそれぞれに備わっていて, 何が無いのかを知っていることは望ましいです. とはいえ, 最近は処理系依存の部分をラップし, 違いを吸収してくれるライブラリが現れているので, それらを用いれば問題は軽減されるといえます.

Lisp方言	処理系について	
	主な処理系	処理系の特徴 (一部)
CommonLisp	CCL (Clozure CL)	・元はMac OS用に作られたLispなのでMac OSとの親和性が高い ・コンパイル速度が高速
	SBCL (Steel Bank CL)	・安定 ・主な処理系の中で最もアップデートが活発 ・実行ファイルの速度が高速
	ACL (Allegro CL)	・商用 ・IDEが便利(らしい) ・StandAloneな実行ファイルを作れる
	LispWorks	・商用 ・IDEが便利(らしい) ・StandAloneな実行ファイルを作れる
	ECL (Embededable CL)	・Cにトランスレートする ・単独で起動するExecutableファイルを生成できる ・現代的なライブラリも使用できる(ASDF等利用可能)
	ABCL (Armed Bear CL)	・JVM上で動作する ・(当然)Javaのリソースが使える ・現代的なライブラリも使用できる(ASDF等利用可能)
	GCL (GNU CL)	・Cにトランスレートする ・高速な数値計算 ・効果的なGC
Scheme	Gauche	・スクリプト言語的利用の少なからずある ・日本人(川合史朗氏)が開発 ・R7RS(Revised^7 Report on algorithmic language Scheme)準拠
	Racket	・Racketプロジェクトとして活動している ・ライブラリも豊富
	Kawa	・JVM上で動作する ・Kawa自体はJava上で別の言語を実装するフレームワーク

表 2　現在主流な CommonLisp と Scheme の処理系

2.4.3　Debian パッケージ内として提供されている Lisp 処理系

CommonLisp

- SBCL
- ECL
- CMUCL
- CLisp
- GCL
- CCL(バグあり)

Scheme

- mit-scheme

2.5　Lisp 界隈のライブラリ事情

Lisper は各自がオレオレライブラリを書くので, 使えるリソースが少ないのでは, というような声を耳にすることがあります. 実際, Python や Java に比べると, 相対的に少ないとは思いますが, 無いわけではありません. むしろ近年活発に開発が進んでいるようにも伺えます. 特に CommonLisp は, 主に ASDF(Another System Definition Facility) というライブラリ管理ライブラリを中心にしてできている QuickLisp というパッケージマネージャの台頭によりライブラリの共有が進んでいます. Scheme の方は, CL に比べライブラリが少ないようです. 本稿では CommonLisp に焦点を絞って説明します.

CommonLisp には, 正式な仕様として, ANSI Common Lisp という仕様があります. この仕様の中で定義されているライブラリ管理システムは, 現在は非推奨であるはずの「Require/Provide」です. Emacs-Lisp ではよくお目にかかる方もいらっしゃるかと思います. しかし, 現在の主流としては, Require/Provide ではなく, 上述の ASDF というシステムがデファクトすアンダー度として使用されています. これをもとにしたパッケージマネージャが Quicklisp で

す.

Quicklisp に登録されているライブラリ数は Quickdocs `http://quickdocs.org` によると, 2017 年 9 月現在で 1557 件あり, Web 系（Web サーバー, パーサーやデータベースなど）や機械学習ライブラリ, GUI アプリケーション作成用ライブラリなどが多く見られます. まだ十分に整備されているとは言えませんが, これからのさらなる発展を考えれば, 実用に耐えるレベルである部分もあります.

2.6 Debian9 で CommonLisp 実行環境を構築する

では, ここからは実際に Lisp をプログラムする環境を Debian9 上に構築する手順を説明します. EmacsLisp で良ければ当然, Emacs さえあれば完了なのですが, 今回は CommonLisp の環境を整え, ライブラリを用いてプログラムが書けるようになるところまで進めます. ここでは Emacs を使うことを前提として話を進めます.

2.6.1 処理系のインストール（今回は SBCL)

まずは, CommonLisp の処理系をインストールします. Debian Stretch からはパッケージマネージャとして apt が推奨ですね. SBCL という処理系を apt からインストールします.

```
~ $ sudo apt install sbcl slime cl-asdf
```

インストールが終われば既に, CommonLisp を書き, 実行できる状況はできています. 実際に動くか試してみましょう.

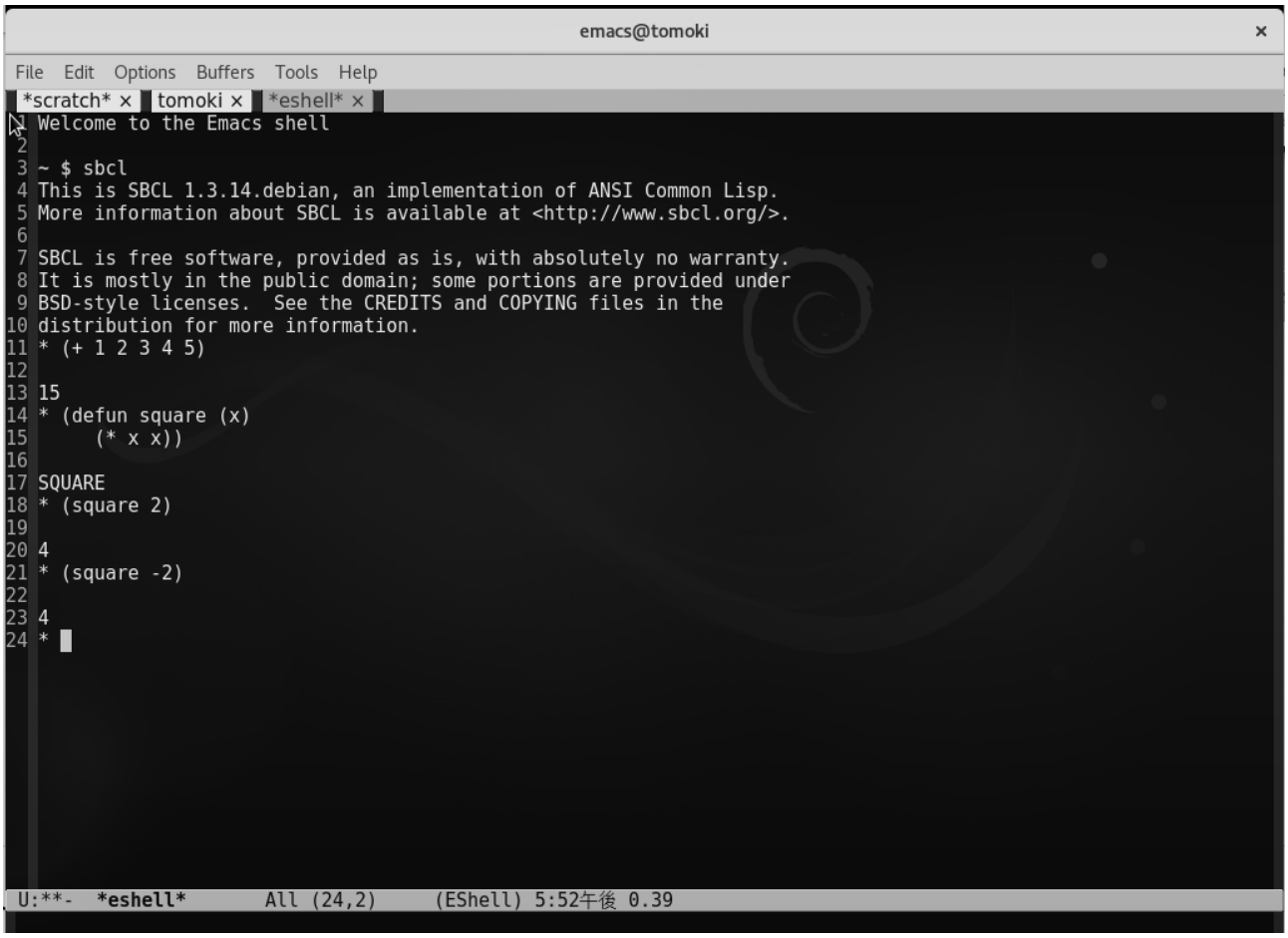


図 2 SBCL の実行画面

正しく動作していれば次の作業に移ります.

2.6.2 統合開発環境のセットアップ

この段階で, Lisp プログラムを書き, 動かすことが可能になっていますが, 実際にコーディングするとなると REPL 上だけで作業するのは不便でしかありません. Emacs では, SLIME という CommonLisp のための非常に便利な対話的開発ができる IDE を動かすことができます.

Emacs の設定ファイルの更新

Emacs の設定ファイルである /.emacs もしくは /.emacs.d/init.el に以下の設定を追加し, SLIME を使用可能にします.

```
(add-to-list 'load-path "/usr/share/common-lisp/source/slime/") ;; 自分の環境に合わせて変える
(setq inferior-lisp-program "/usr/bin/sbcl") ;; 自分の環境に合わせて変える
(require 'slime)
(slime-setup)
```

これらの設定を有効化して, ”M-x slime”を実行し Slime を起動します. 以下のような画面になれば成功です.

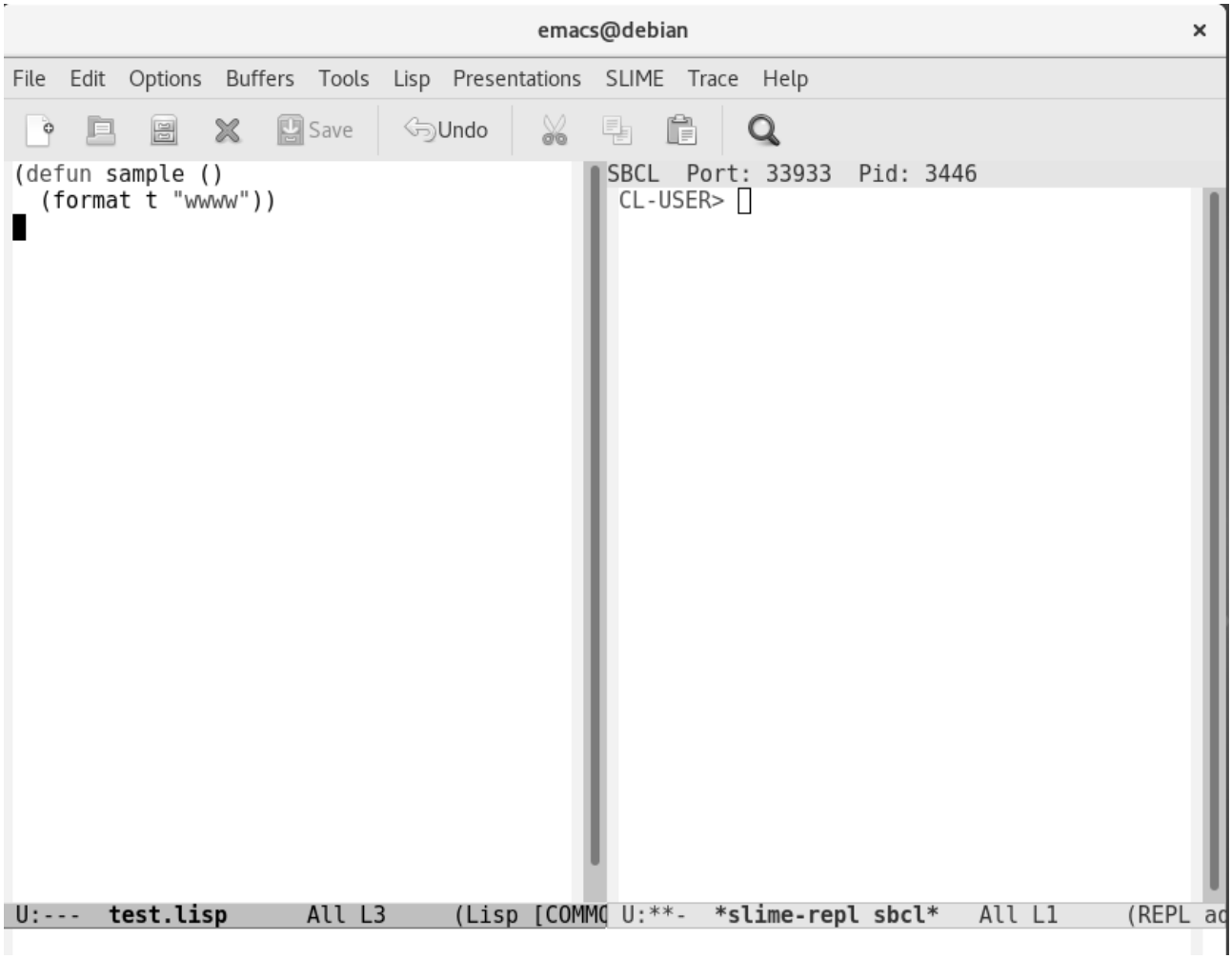


図 3　SLIME 起動画面

この画面では, 左側が Lisp ソースファイル, 右側が REPL を示しています. SLIME は, Swank サーバーという Lisp を処理するサーバにソケット通信で接続し, REPL を実現する仕組みになっている. なので, ターミナルなどで立ち上げた Swank サーバーがあれば, Emacs の SLIME からそのサーバーにアクセスすることも可能です.

2.6.3 ライブラリ管理システムのインストール

CommonLisp には Quicklisp というライブラリ管理システムがあります. 以下のコマンドでインストールできます.

```
$ curl -O https://beta.quicklisp.org/quicklisp.lisp
$ curl -O https://beta.quicklisp.org/quicklisp.lisp.asc
$ gpg --verify quicklisp.lisp.asc quicklisp.lisp
$ sbcl --load quicklisp.lisp
* (quicklisp-quickstart:install)
* (ql:add-to-init-file)
* (quit)
```

Quicklisp は Lisp で書かれており, インストールプログラムを処理系にロードして使用します. 上記のコマンドを実行すると, Homedir 以下に quicklisp/.sbclrc というファイルができるはずです. これは SBCL 用設定ファイルです. 上のコマンドではこの設定ファイルに Quicklisp のロードを行うコードが追記されるため, 処理系起動時には Quicklisp が使用可能になっています.

2.6.4 プログラムの開発

ここまでで十分に開発環境が整いました. これ以降の設定は各自の好みに合わせて変化させてください, それでは, 最後に, この環境で Lisp の環境を活かした高速な開発の手順を簡単に説明します. 次のような簡単な関数をソースファイル側で定義します.

```
(defun sample ()
       (format t "It's a sample function"))
```

これを入力した後, この関数の中の何処かにマークをおいて, ”C-c C-c”コマンドを打つことで REPL に送信し, コンパイルすることになります.

この状態ですでに REPL の方でも関数が定義された状態になります. 最後に定義した関数を実行してみます. 正しく動いていることがわかります.

プログラムの開発２

以降はファイルに Lisp プログラムをまとめて記述し, ファイル名を例えば「sample.lisp」といった名前にし, そのファイルを読み込みます. ファイル読み込みは以下に示す load コマンドです.

```
(load "~/Desktop/sample.lisp")
; t
(sample)
; It's a sample function
```

ここまでで Lisp でプログラムを書くための基本的な環境設定は整いました.

2.7 まとめ

本稿では十分に記 les すことは叶いませんでしたが、Lisp は上述の REPL をはじめとした、非常に効率のいい高速開発が可能な強力な言語です。Common Lisp などは使用の中にリバースエンジニアリングのためのディスアセンブルの昨日や、最適化オプションをを持っており、C 言語と比べても遜色のない実行速度のプログラムを書くこともできるよう設計されています。

マクロを用いた自己書き換えの能力なども相まって、人工知能的なプログラムを構築するのに有用であると考えられます。そのシンプルで思考をそのまま起こした者に近くなる言語設計は、プログラムの基礎を知るためにも有用であると言え、現在はあまり大きな注目を集めいているとは言えませんが、今後もう一度日の目を見るときが来るかもしれません。

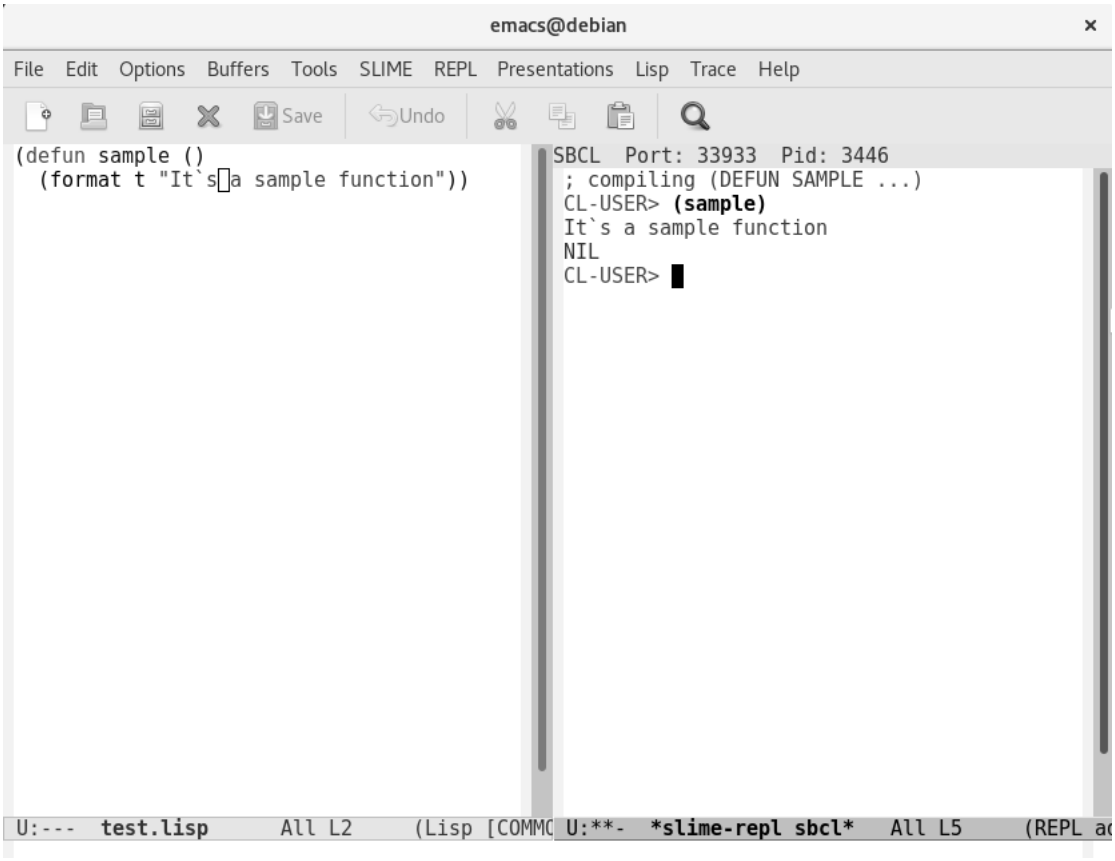


図 4　関数の実行

また、近年では、日本人デベロッパーが開発・保守している Roswell[*1]という Common Lisp パッケージマネージャーが台頭し始めており、今回解説した公式の環境構築よりも非常にシンプルな環境構築や、実用的なシステム構築を可能にする仕組みが開発され続けています。

とはいえ、モダンな Lisp 開発のためのドキュメントを集めることは容易ではありません。情報を得るためには、Lisper たちが集まるコミュニティに参加し、交流することを強くおすすめします。

2017 年現在では、関西では我々が運営している「関西 Lisp ユーザ会[*2]」が 3ヶ月に 1 度程度、関東では「Shibuya.lisp[*3]」というグループが毎月、Lisp に関する勉強会を行っています。

Lisp の知識や経験は問われず、LT を聞きに行くだけでも Lisp ユーザが今、どのようなことに興味を持っているのかを知る良い機会になると思われます。

[*1] ロズウェル：https://github.com/roswell/roswell
[*2] https://kansai-lisp-users.github.io/
[*3] https://lisp.connpass.com/

3　初めてのキーサインパーティ

ysaito

初めてキーサインパーティに参加して、どこでつまづいたかお伝えします。

3.1　Mac から caff を使おうとして

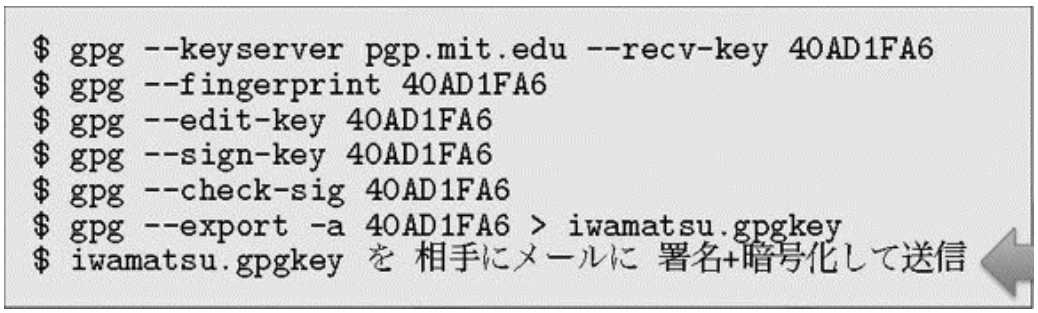
Homebrew から消えてた

```
$ brew search signing-party
 ==> Searching local taps...
 ==> Searching taps on GitHub...
 ==> Searching blacklisted, migrated and deleted formulae...
 signing-party was deleted from homebrew/core in commit e05298ad8a:
```

3.2　では gpg のみでやってみようとして

資料[*4]を参考に gpg のみでやってみようとしてみたが、つまづいた
相手の公開鍵で暗号化するところを自分の公開鍵で暗号化してしまった

```
$ gpg --encrypt --recipient 相手の公開鍵 ID 相手の公開鍵
```



[*4] http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201009-presentation.pdf

3.3 caff をつかわないなら

参考: https://github.com/thinkAmi/caffish-ps

```
# キーサーバより相手の公開鍵を取得し、
# 自分の公開鍵の鍵束に入れる
$ gpg --keyserver pgp.mit.edu --recv-key 相手の公開鍵 ID

# 表示されるフィンガープリントと、
# 手元の書類のフィンガープリントが一致していることを確認
$ gpg --fingerprint 相手の公開鍵 ID

# 相手の公開鍵に署名
$ gpg --sign-key 相手の公開鍵 ID

# 署名した公開鍵をエクスポート
$ gpg --export -a 相手の公開鍵 ID > ./foo.gpgkey

# 自分の秘密鍵で署名した相手の公開鍵を、相手の公開鍵を使って暗号化
$ gpg --no-auto-check-trustdb  --trust-model=always \
  --armor --recipient 相手の公開鍵 ID --encrypt ./foo.gpgkey
# foo.gpgkey.asc が生成される
# 暗号化した公開鍵をメールに添付し、
# メール本文を暗号化して相手へ送信
```

これは面倒です。そうだ caff 使おう。

Debian strech に caff をいれて

メールサーバを立てて...

3.4 もう一つ詰まったところ

```
$ caff -u 自分の公開鍵 ID 一人目の公開鍵 ID 二人目の公開鍵 ID...
[NOTICE] Fetching keys from pool.sks-keyservers.net, this may take a while...
[WARN] Local-user 自分の公開鍵 ID is not defined as one of your keyid in ~/.caffrc (it will not be used)
[ERROR] None of the local-user keys seem to be known as a keyid listed in ~/.caffrc
```

/.caffrc はちゃんと設定しているはずだけど...

自分の公開鍵のフィンガープリントの後半１６桁

```
$ caff 一人目の公開鍵 ID 二人目の公開鍵 ID...
```

通った

4 caff と mail-transport-agent

yy_y_ja_jp

キーサインに便利な caff とそれに使う mail-transport-agent について解説します。

4.1 caff

- caff(1) – CA - Fire & Forget
  キーサインパーティなどで便利なツール
- signing-party パッケージでインストールできる
- キーサインパーティに参加した後で、このコマンドを実行すれば必要な作業をやってくれる
  1. 参加した人の公開鍵を取得
  2. その公開鍵に自分の秘密鍵で署名
  3. 署名済みの公開鍵をその人にメールで送信

4.2 Mail Transport Agent

メール転送エージェント (MTA) は、受け付けたメッセージを他のサーバなどに転送するプログラムです。ここで

- 受け付けかた（インターフェース）には
  - SMTP（TCP 25 番ポートなど）
  - sendmail 互換（/usr/sbin/sendmail コマンドを実行して標準入力に渡すなど）
  - など

  があり、
- 転送のしかたには
  - 自力で DNS を引いて送付先サーバに転送
  - 他のサーバに渡して転送してもらう（リレー）

  があります。

受け付けかた（インターフェース）については、メッセージをマシン上にある（ローカル）MTA に渡すだけなら、わざわざメールサーバ（SMTP サーバ）を立ち上げてそれ経由で渡す必要はありません。/usr/sbin/sendmail コマンドを起動して、そのコマンド自体が転送してくれれば充分です。

次に転送のしかたですが、自力で転送してしまうと次のような様々な問題が起こります。

- そのメールサーバをインターネット公開にしていない場合は自分宛てのメールが受信できない – エラーメールも受け取れない
- 自分のメールアドレスのドメインがそのサーバ管理でない場合はなりすまし（フィッシング）になってしまう

参考: このようなメールが届いたときに Google メールが表示することがある警告メッセージ

This message may not have been sent by: xxx@example.com

Why is this message in Spam? It's similar to messages that were detected by our spam filters.

- OP25B の影響で送信できないこともある

4.3 mail-transport-agent

- Debian ポリシーで定められた仮想パッケージのうちの 1 つ
  `/usr/share/doc/debian-policy/virtual-package-names-list.txt.gz` `https://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt` で見られる

```
mail-transport-agent    a mail transport agent (e.g. Smail, Sendmail, &c)
```

- mail-transport-agent を提供しているパッケージは `apt-cache showpkg mail-transport-agent` や `https://packages.debian.org/stable/mail-transport-agent` で見られる

- Debian のデフォルトは exim4-daemon-light
  – 仮想パッケージ default-mta も提供 (Provides)
- どのパッケージも Conflicts: mail-transport-agent, Replaces: mail-transport-agent, Provides: mail-transport-agent と書いてある
  ⇒ 2 つ以上インストールできないようになっている

```
$ apt-cache show exim4-daemon-light
Package: exim4-daemon-light
Source: exim4
Version: 4.89-2+deb9u1
Installed-Size: 1167
Maintainer: Exim4 Maintainers <pkg-exim4-maintainers@lists.alioth.debian.org>
Architecture: amd64
Replaces: exim4-base (<= 4.61-1), mail-transport-agent
Provides: default-mta, exim4-localscanapi-2.0, mail-transport-agent
Depends: exim4-base (>= 4.89), libc6 (>= 2.16), libdb5.3, libgnutls30 (>= 3.5.6), libpcre3, debconf (>= 0.5) | debconf-2.0
Conflicts: mail-transport-agent
```

- MTA のパッケージが 2 つに分かれていることもある – mail-transport-agent を提供するパッケージが sendmail 互換インターフェース (/usr/sbin/sendmail) などを提供

```
$ apt-cache show msmtp-mta
Package: msmtp-mta
Source: msmtp
Version: 1.6.6-1
(略)
Replaces: mail-transport-agent
Provides: mail-transport-agent
Depends: msmtp
Conflicts: mail-transport-agent
Description-en: light SMTP client with support for server profiles - the regular MTA
(以下略)
```

```
$ apt-cache show msmtp
Package: msmtp
Version: 1.6.6-1
(略)
Depends: libc6 (>= 2.22), libgnutls30 (>= 3.5.6), libgsasl7 (>= 1.1), debconf (>= 0.5) | debconf-2.0, ucf
Recommends: ca-certificates
Suggests: msmtp-mta
Description-en: light SMTP client with support for server profiles
(以下略)
```

MTA が SMTP サーバも提供するかどうかはそのパッケージのプログラム次第です。

- SMTP サーバも提供する – exim4-daemon-light, postfix, nullmailer など
- SMTP サーバを提供しない（リレー専用） – msmtp など

4.4 caff と mail-transport-agent

caff は署名済みの公開鍵をその人にメールで送信します。しかし

- 実行されたマシン上にある（ローカル）MTA を使ってメール送信している
- デフォルトの MTA だとローカル配送のみ

なのでそのままだと次のようなエラーメールになってしまいます。

```
Subject: Mail delivery failed: returning message to sender
(略)
 Mailing to remote domains not supported
```

これは、メールがインターネットに出ていかない状態になっているためで、MTA の設定が必要です。

もし自分の（uid に使っている）メールアドレスが自分の管理するメールサーバのドメインでないのなら（例えば `@gmail.com` などの場合）、

- メールサーバ（SMTP サーバ）なし
- リレー専用

な MTA を選べば設定がかんたんです。例として msmtp-mta パッケージがあります。設定については `https://wiki.debian.org/msmtp` をご覧ください。

- 個人の設定ファイル ~/.msmtprc に Gmail アカウントと SMTP サーバの情報を書いてリレーする方法が書かれている
- パスワードを GPG 鍵で暗号化して保存する方法も書かれている

4.5 まとめ

- caff はローカルの MTA を使ってメール送信する
- ローカルの MTA は mail-transport-agent 仮想パッケージを提供するパッケージをインストールすればよい
- マシンには mail-transport-agent 仮想パッケージを提供するパッケージのうち 1 つだけをインストールできる (Conflicts/Replaces/Provides)
- MTA のパッケージが 2 つに分かれていることもある（msmtp/msmtp-mta, qmail/qmail-run など）
- MTA には SMTP だけでなく sendmail 互換インターフェースもある – caff 作業マシンがインターネット公開のメールサーバでないなら SMTP は不要
- 自分が管理していないドメインのメールアドレスなら、その外部 SMTP サーバにリレーするようにローカルの MTA をインストール（msmtp-mta パッケージがおすすめ）し設定するとよい

5 lxc について

杉本 典充

5.1 アジェンダ

- 自己紹介
- 仮想化技術について
- lxc とは
- lxc のインストール
- lxc のコマンド解説
- lxc を実用する
- おわりに
- 参考資料

5.2 自己紹介

- Norimitsu Sugimoto (杉本 典充)
- dictoss@live.jp
- Twitter: @dictoss
- Debian-3.1、FreeBSD-6.2 の頃から使っています
- Debian GNU/kFreeBSD が気になっておりウォッチ中
- 仕事はソフトウェア開発者をやってます

5.3 仮想化技術について

5.4 仮想化技術の分類

- コンテナ型仮想化
- 準仮想化型
- 完全仮想化型 (エミュレーション型)
- 完全仮想化型 (ハイパーバイザ型)

5.5 仮想化技術のメリット・デメリット

- 準仮想化型、完全仮想化型

  - 物理マシンをエミュレートした仮想マシンとして動作する
  - 仮想マシン上でもカーネルを動作させる
  - 物理マシンで動かしていたプログラムはほぼそのまま動く
  - CPU、メモリ、ディスクを多く消費する
- コンテナ型仮想化
  - ゲスト環境の動作にカーネルは不要 (ホスト環境のカーネル上で動作)
  - ゲスト環境は、ホスト環境から見るとプロセスとして扱われる
  - ゲスト環境が利用できるリソースに制約がつく場合がある

5.6 chroot

- chroot システムコールと chroot コマンド
- 1982 年にビル・ジョイが開発したとされている
- # chroot rootfsdir でコンテナ環境に入れることができる

5.7 lxc とは

- LinuX Containers のことで、省略して lxc と読んでいる
- あるディレクトリ配下に実行ファイル、ライブラリ、設定ファイルを適切に配置した rootfs を準備する
- rootfs を chroot 環境で起動し、仮想マシンのように動かすことができる
- Debian 9 Stretch では lxc-2.0.5 を採用

5.8 lxc のインストール

5.9 インストールの流れ

- ブリッジネットワークと libvirtd の準備
- lxc のインストール

5.10 ブリッジネットワークと libvirtd の準備

パッケージのインストール

```
# apt-get install libvirt-clients \
libvirt-daemon-system ebtables dnsmasq
```

仮想ネットワークの設定

```
# virsh net-autostart default
# virsh net-start default
```

5.11 ブリッジネットワークと libvirtd の準備

確認

```
$ sudo virsh net-info default
Name:           default
UUID:           78564864-f237-4059-a12a-3ec04369a27b
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         virbr0

$ ip a show virbr0
→ 192.168.122.1/24 が付与されている
```

5.12 lxc のインストール

コンテナ内のリソース制約を処理する cgroup の確認

```
# mount | grep cgroup
→ stretch は標準で mount されている
```

パッケージのインストール

```
# apt-get install lxc libvirt0 libpam-cgroup \
libpam-cgfs
```

環境を確認

```
$ ls /usr/bin | grep lxc
# lxc-checkconfig
```

5.13 lxc のインストール

- lxc-create
- lxc-destroy
- lxc-start
- lxc-stop
- lxc-console
- lxc-attach

5.14 lxc のコマンド解説

5.15 lxc-create(1)

```
# lxc-create -n demo1 -t debian -- \
--release=stretch --arch=amd64 \
--mirror=http://ftp.jp.debian.org/debian
```

- 実行するとテンプレートが debian の場合は debootstrap を実行して rootfs をダウンロードする
- lxc のゲスト環境のディレクトリは、/var/lib/lxc/” コンテナ名” 。中身は以下。
  - config (設定ファイル)
  - rootfs (コンテナの中身)

5.16 lxc-create(2)

- config を修正して、ネットワークの設定を行う
- lxc.network.type = veth
- lxc.network.flags = up

- lxc.network.link = virbr0
- lxc.network.name = eth0
- lxc.network.ipv4 = 192.168.122.60/24
- lxc.network.ipv4.gateway = 192.168.122.1

5.17 lxc-destroy

```
# lxc-destroy -n demo1
```

- コンテナを削除します

5.18 lxc-start

```
# lxc-start -n demo1
```

- 実行して何もエラーが表示されなければ、バックグラウンドで lxc コンテナが動き出します
- 起動したコンテナへの接続は、後述する lxc-console または lxc-attach で行います
- コンテナへのログインは ssh でもログインできますが、ユーザを作成する必要があります

5.19 lxc-stop

```
# lxc-stop -n demo1
```

- コンテナ環境を終了するよう指示を出します
- コンテナ環境の終了とは、コンテナ内の init プログラムを終了することをいいます
- コンテナ環境で shutdown 命令は実行できません

5.20 lxc-console

```
# lxc-console -n demo1
```

- lxc のゲスト環境のコンソールに接続します
- コンソールを抜ける場合は、「Ctrl+a q」の順に入力してください

5.21 lxc-attach

```
# lxc-attach -n demo1 {command}
```

- lxc のゲスト環境でコマンドを実行します
- コマンドを指定しない場合は、コンテナ内のユーザのデフォルトシェルが実行されます
- lxc-console でログインすることに比べ、いきなりコンテナ内でシェルを実行できるため、lxc-attach の方がコンテナ内の整備がしやすいです

5.22 lxc を実用する

5.23 コンテナ環境のセットアップの流れ

- lxc-create を実行してコンテナを生成する
- lxc のゲスト環境の config を書き換えてネットワークを設定する
- lxc-start してコンテナを起動する
- lxc-attach でゲスト環境に入る

```
# passwd
# adduser username
# apt-get install sudo vim-tiny
# visudo
```

- ssh ログインしてお好みに設定する

5.24 何に lxc を使うか

- 一時的な検証で、ホスト環境にいろいろインストールしたくない場合
- アプリケーションのクリーンビルドやクリーンインストールをテストする場合
- ホスト環境は systemd、ゲスト環境は sysvinit と使い分ける場合
- python2 系と python3 系の wsgi アプリを 1 つのホストで動かしたい場合
- ホスト環境と異なる CPU アーキテクチャのエミュレーション環境がほしい場合
  - apt-get install qemu qemu-user-static binfmt-support
  - その後、lxc-create を実行してください
  - 詳しくは CrossDebootstrap を調べてみてください

5.25 おわりに

- Debian 上で lxc を試してみました
- 発展系である LXD や docker へつなげていきましょう
- コンテナは便利ですので試してみてください

5.26 参考情報

- 「LXC」 `https://linuxcontainers.org/`
- 「LXC - Debian Wiki」 `https://wiki.debian.org/LXC`
- 「LXCLibVirtDefaultNetwork」 `https://wiki.debian.org/LXC/LibVirtDefaultNetwork`

6 Debian Strech で LXC を使う

Yosuke OTSUKI

lxc の非特権でコンテナを debian wiki に記載されている通りに作ろうとしたら、色々ハマりました。1 ヶ月ぐらいかかって起動に成功しました。

6.1 動機

仕事でも Debian の pbuilder みたいに環境に依存しないビルドができないものかと考え、コンテナを使おうと考えました。やっぱり Docker が流行っているので、最初は Docker を第一候補としていました。仕事では、CentOS を使用せざる得ないので、Docker 用の Cent OS のレポジトリを確認したところ、repomd.xml で参照されている package の db の鍵が異なりました。 これは、Docker 用に改造されているのでは？と考え LXC を使用することにしました。[*5]

- Docker : `https://yum.dockerproject.org/repo/main/centos/7/repodata/repomd.xml`
- CentOS : `http://ftp.jaist.ac.jp/pub/Linux/CentOS/7/os/x86_64/repodata/repomd.xml`

Database や Web アプリなど、middle ware 上で動く場合は、Docker でも問題ないかもしれません。しかし、native 系のアプリの Linux 上でのテストに使用したいので、できるだけ実機に近い環境が必要でした。

また、個人でも sid の開発環境が欲しかったため、作成できないかと試してみました。今回は、こちらのご紹介をいたします。

なお、東京エリア Debian 勉強会で杉本さんが lxc を取り上げています。2016 年 7 月, 2013 年 4 月の東京 Debian 勉強会の資料をご覧ください。

現在利用できる、Linux Container のバージョンとサポート終了日時を確認しておきましょう。

Linux Container

- 1.0 2019 年 6 月 1 日に サポート終了予定
- 1.1 2016 年 9 月 1 日に サポート終了
- 2.0 2021 年 6 月 1 日にサポート終了予定 (Stretch のはこれ)

1.0 系でも非特権コンテナ機能はあります。使用には Linux Kernel 3.12 以上が必要です。

Debian ならば jessie でも 3.16 なので問題なさそうです。ただし、私は、jessie では試していません。なお、Redhat 系では kernel が 3.10 なので非特権コンテナはそのままでは、利用できません。

[*5] 最も repo の参照先を変えると言う方法もあるかもしれません。

6.2 必要なものをインストール

```
# apt-get lxc libvirt0 libpam-cgroup libpam-cgroup libpam-cgfs bridge-utils
```

6.3 ブリッジを作ります

私の環境の場合、192.168.100.0/24 が host マシンで定義されているネットワークです。ここでは、作成するコンテナに 192.168.100.50/24 を与えることにしましょう。

```
iface eth0 inet manual

# auto wlp2s0
iface lxcbr0 inet static
bridge_ports eth0
address 192.168.100.50/24
```

個人的に debian は bridge device を簡単に作成できるので好きです。

6.4 カーネルが対応済み、かつ必要なツールが全部あるか確認

default の stretch ならば問題はないはず。

```
$ lxc-checkconfig
```

画面出力は、紙面節約のため省略します。すべて ”enabled” と表示されていれば問題ないです。

6.5 カーネルの設定を変更

カーネルのパラメタを変えて、コンテナ作成の許可をユーザーに与えましょう。

```
sudo sh -c 'echo "kernel.unprivileged_userns_clone=1" > /etc/sysctl.d/80-lxc-userns.conf'
```

このパラメタの意味を man 7 user_namespace と kernel.org で調べましたが記載がありませんでした。kernel.org の sysctl のパラメタに関するドキュメントは 2009 年から更新がないようです。

6.6 subid を設定

```
# usermod --add-subuids 100000-165536 yosuke
# usermod --add-subgids 100000-165536 yosuke
```

/etc/subgid と /etc/subuid に自分のユーザー名で、先程のコマンドで指定した値が反映されているか確認してください。

6.7 特定のユーザーが作成できる network interface の上限数を指定する

```
echo "$USER veth lxcbr0 10"| sudo tee -i /etc/lxc/lxc-usernet
```

こちらも、/etc/lxc/lxc-usernet に値が反映されているか確認しましょう。

6.8 非特権ユーザー用の設定ファイルを .config 以下に作成する

以下の場所にファイルを作成してください。/home/yosuke/.config/lxc/default.conf
ファイルに作成するコンテナの設定を記載します。

```
lxc.include = /etc/lxc/default.conf
lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00:16:3e:fe:3d:13

lxc.id_map = u 0 100000 65536
lxc.id_map = g 0 100000 65536

lxc.mount.auto = proc:mixed sys:ro cgroup:mixed

lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = f1:53:7f:00:00:01
lxc.network.ipv4 = 192.168.100.50/24
```

ちなみに、hwaddr にもグローバルアドレスとローカルアドレスがあるそうです。最初に、指定したアドレスが偶然マルチキャストアドレスだったため、NIC の作成ができませんでした。先頭のオクテットの下から 2 ビット目が立っていた場合、ローカルアドレス。立っていない場合はグローバルアドレスだそうです。また、オクテットの最も下位ビットが立っていた場合、ユニキャストアドレス、そうでなければマルチキャストアドレスです。詳しくは、wikipedia に記載されています。`https://en.wikipedia.org/wiki/MAC_address#Address_details`

6.9 LXC コンテナを作成

さて、ここまで非特権 LXC コンテナを作成するため、特権コンテナの場合よりも多くの設定を行っていました。いよいよ、コンテナ本体を作成します。

lxc-create コマンドは、コンテナの作成先のディレクトリを指定しない場合、/var/lib/lxc 以下に OS イメージを作成します。

では、このディレクトリの書き込み権限を見てみましょう。

```
yosuke@asusx200c:~/work/lxc$ ls -la /var/lib/lxc
total 8
drwxr-xr-x  2 root root 4096 Aug 28 07:31 .
drwxr-xr-x 49 root root 4096 Sep 24 07:54 ..
```

一般ユーザーに書き込み権限はありません。lxc-create コマンドには、コンテナの作成先を指定する - - lxcpath (-P) オプションがありますこのオプションを使用し、一般ユーザーが書き込める場所に OS image を作成すれば良いのです。

私は、はじめに下記のコマンドでコンテナを作成しました。

```
lxc-create -n stretch -t download -P ./
```

```
man lxc-create より:

   -P, --lxcpath=PATH
          Use an alternate container path. The default is /var/lib/lxc.
```

いざ、コンテナを立ち上げてみようとすると 「rootfs が見つからない」とエラーが出ます。原因は、lxc-create の –path オプションが絶対パスで指定する必要があるためでした。これに、気がつくまで結構かかってしまいました。

作成したコンテナの config ファイルを確認すると、lxc.rootfs が入力を絶対パスに変換しないで保持していました。今の所そのような仕様のようです。

lxc-create の man も確認しましたが、絶対パスで指定しろとは記述されていません。調べたところ、bug 登録されていました。orz。 `https://github.com/lxc/lxc/issues/78`

個人的には、絶対パスにしないと意図したコンテナでないコンテナが立ち上がることがあると思います。そのため、bug とは言えないのでは？ 安全面でも、絶対パスのほうが良いでしょう。man もしくは help は改善したほうが良いと思います。

```
$ lxc-create -n stretch -P /home/yosuke/work/lxc
```

とすれば問題ないはずです。

6.10 コンテナを起動します。

```
lxc-start -d -n stretch -P /home/yosuke/work/xc
```

エラーメッセジが何も表示されなければ起動成功です。lxc-ls –fancy でコンテナの動作状況が確認できます。

```
yosuke@asusx200c:~/work/lxc$ lxc-ls --fancy -P /home/yosuke/work/lxc/
NAME    STATE   AUTOSTART GROUPS IPV4                      IPV6
stretch RUNNING 0         -      10.0.3.169, 192.168.100.51 -
```

上記が 2 つの ipv4 アドレスを持っているのは、ゲスト OS がデフォルトのままなので、nic を dhcp で立ち上げているためです。

6.11 起動したら、コンテナにつなぎます

```
lxc-attach -n stretch
```

root でログインできるはずです。新しくユーザーを作り、パスワードを設定しましょう。ついでに、/etc/network/interfaces も修正して、2 つ ipv4 アドレスが与えられることがないようにしましょう。

6.12 最後に ssh でコンテナに接続します

```
ssh 192.168.100.50
```

ログインできましたら、あとはご自由にお使いいただけます。

```
yosuke@asusx200c:~/work/lxc$ lxc-attach -n stretch -P /home/yosuke/work/lxc/
```

早速パッケージを追加してみようとしました。

```
root@stretch:/# apt-get update
Reading package lists... Done
W: chown to _apt:root of directory /var/lib/apt/lists/partial failed - SetupAPTPartialDirectory (1: Operation not permitted)
W: chmod 0700 of directory /var/lib/apt/lists/partial failed - SetupAPTPartialDirectory (1: Operation not permitted)
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
```

失敗しました。一般ユーザ権限で実行していることになっているみたいですね。Ubuntu ですが似たような問題が報告されています https://github.com/lxc/lxd/issues/3310 現在、回避策を模索中です。

6.13 References

- https://stgraber.org/2016/04/06/lxc-2-0-has-been-released/ 2017/11/23 accessed
- https://wiki.debian.org/LXC 2017/11/23 accessed
- https://wiki.debian.org/LXC/SimpleBridge 2017/11/23 accessed
- http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201607.pdf 2017/11/23 accessed

7 Debian on Pomera DM200 どのように Debian マシンとして動くようにしたか

@ichinomoto

7.1 はじめに

Pomera DM200 に Debian GNU/Linux をインストールして動くようにしてみましたのでその過程をまとめてみました。

7.2 Pomera DM200 とモチベーション

Pomera DM200 とは、KING JIM 社が販売している今風のワープロ機です。ワープロであるがゆえに文書を書く機能に特化しており、一部の人が好んで使っています。

筆者はサイズ感が気に入っており、MD200 を PC として使えれば良かったのに、と感じました。

DM200 のアップデータが公開されたときに、配布ファイルの構成が Ubuntu ではないかと話題になりました。実際に筆者がアップデータファイルの内容を確認してみると、パーティション情報がそのまま結合されて保存されているように感じ、DM200 は Linux ベースな OS で動作しているのではないかと予想しました。

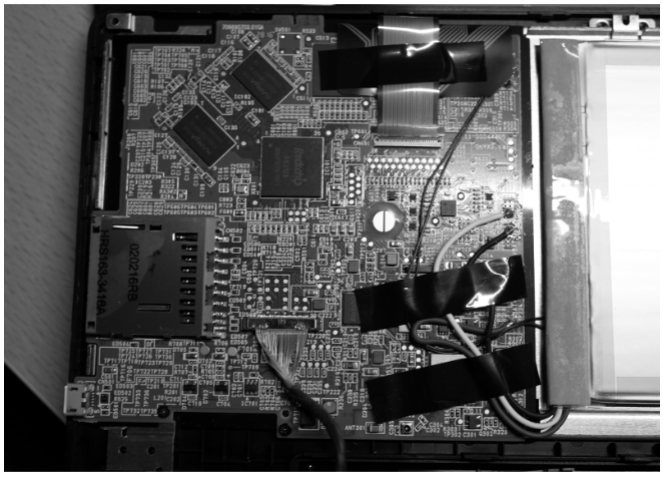
この考察から DM200 を PC のように使えるかもしれないと考え、DM200 を購入して Debian 化に挑戦することにしました。

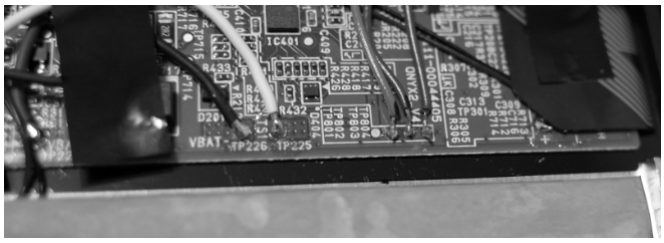
7.3 DM200 の解析

7.3.1 シリアルコンソールの場所

DM200 を普通に利用している状態では、Linux の糸口を得ることができませんでした。そのため中身を開けてみて、組み込み機器用のデバッグ用シリアルコンソールを探してみました。基板を探してみると、すぐに見つかりま

した。





7.3.2 ソースなしでがんばってみる

実験として、描画プログラムをとめて、シリアルコンソールから/dev/fb0 に適当なデータを書き込んでみました。すると DM200 の画面になにやら描画されることを確認しました。そのため、chroot 環境を作って fbterm を使えば画面に描画処理ができるのではないかと考えました。

debian の chroot 環境を作成するため、qemu-debootstrap を実行します。そして、できた rootfs を SD カードへコピーします。

```
# qemu-debootstrap --arch=armhf --variant=minbase
```

SD カードを DM200 に差し込み、DM200 のシリアルコンソールから SD カード上の debian rootfs へ chroot します。chroot 環境の fbterm を/dev/fb0 指定で起動すると、内蔵の液晶画面に描画処理を行うことができました。

しかし、動作を確認するとキーリピートが効かない、画面左上が 1 文字だけ表示されないなど、多くの問題も見つかりました。

chroot 環境で使うより debian な rootfs で起動して DM200 を使いたいと考え、任意の rootfs で起動する方法を模索しました。組み込み機器では U-Boot を使っている事例が多いため、DM200 も U-Boot を使っているとすれば起動処理に割り込むことで任意の rootfs を起動できると推測しました。

7.3.3 ソースコード開示請求

ソースコードがない状況で U-Boot であるか推測しても解析が進まないため、お客様サポートフォームへ連絡し、DVD に収めたソースコード一式を入手しました。

ソースコードを調査し、以下が判明しました。

- U-Boot での割り込み処理は無効になっている
- kernel config はそのままでは LCD にコンソールは出力できないように見える

Linux のデスクトップ環境での利用を想定した kernel config ではないことから、DM200 の eMMC の U-Boot と kernel を書き換える必要があると考えました。

ただ、U-Boot の置き換えは失敗すると二度と起動しなくなる可能性があり（＝文鎮化）、できることなら避けたい作業です。調査を進めると U-Boot から呼び出せる kernel は 2 つあるようで、片方の kernel のみを書き換えるに留めた状態でどこまで debian が動くか試してみることにしました。

7.4 kernel のビルド

7.4.1 ビルド環境の構築

DM200 向けに kernel をビルドするため、コンパイルする環境が必要になります。まずは armhf で動作する他のボード上で kernel をセルフコンパイルしてみましたが、とても遅い状態でした。[*6][*7]

そこで qemu-debootstrap で作成した armhf 環境で kernel をビルドすることにしました。

```
# qemu-debootstrap --arch=armhf --variant=buildd
```

この環境を使って PC 上で kernel のビルドを行うと 5 分程度で処理が終わります。

7.4.2 kernel config の変更と kernel のビルド

DM200 の LCD 上にコンソールを出力できるように kernel の.config ファイルへ以下を追加します。

```
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE=y
```

また、U-Boot の書き換えを行わない方針のため、kernel の起動パラメータである CMDLINE を U-Boot の現状の指定値から変更することができません。そのため、.config に必要な CMDLINE パラメータを指定します。

```
CONFIG_CMDLINE="vmalloc=496M console=tty0  \
mtdparts=rk29xxnand:0x00002000@0x00002000(uboot), \
  : (snip)
,-@0x005FA000(reserve) \
rdinit=/sbin/init root=/dev/mmcblk0p15 storagemedia=emmc \
uboot_logo=0x02000000@0x7dc00000:0x01000000 \
loader.timestamp=2016-08-29_12:54:04 \
androidboot.mode=emmc loglevel=3 rootwait"
CONFIG_CMDLINE_FORCE=y
```

7.5 rootfs の作成

rootfs がどのように作られているかは、NetBSD を使っていたときにアーキテクチャ用のパッケージを展開しただけで動作していたことを思い出し、debootstrap で作成したディレクトリツリー一式を rootfs としてそのまま使うことにしました。

rootfs を以下のコマンドで作成し SD カードへコピーして DM200 上で動作確認したところバイナリが動作することを確認しました。[*8]

```
# qemu-debootstrap --arch=armhf --variant=minbase --include=XX
```

7.6 rootfs の引き継ぎ

Pomera の標準環境の initramfs から、SD カード上の debootstrap で作成した rootfs へ switch することができれば、debian が起動するはずです。しかし、switch_root は PID 1 から実行できない制約があります。そのため、

[*6] 試したのは Raspberry Pi3(https://www.raspberrypi.org/products/raspberry-pi-3-model-b/)、dragonboard(https://developer.qualcomm.com/hardware/dragonboard-410c)。

[*7] ビルド処理が遅いのはストレージが遅いためかもしれません。

[*8] Raspberry Pi の rootfs の作成も同様に debootstrap を使っているようです。https://gist.github.com/abulte/3917357

initramfs も作成することにしました。

initramfs は busybox で作成することができます。initramfs が展開されて最初に呼ばれる/sbin/init の処理に、switch_root を呼び出し、SD カード上の rootfs を mount し、mount した SD カード内にある init を呼び出すようなスクリプトを作成しました。

これで、DM200 が debian としてブートするようになりました。

7.7 Debian 化後の問題点

7.7.1 キーリピートが効かない

キーリピートが効かない問題は、kernel のソースコード上で無効になっていたため、kernel のソースコードを修正しました。

```
arch/arm/mach-rockchip/rk312x.c
static struct tc3589x_keypad_platform_data tc35893_data = {
  .krow = 8,
  .kcol = 12,
  .debounce_period = TC_KPD_DEBOUNCE_PERIOD,
  .settle_time = TC_KPD_SETTLE_TIME,
  .irqtype = IRQF_TRIGGER_FALLING  | IRQF_ONESHOT,
  .enable_wakeup = true,
  .keymap_data    = &onxy2_keymap_data,
  .no_autorepeat  = true, ←これ。false に変更しました。
```

7.7.2 ユーザーがネットワークを使えない

以下の kernel config が有効の場合は特定のグループ以外から socket が使えないと調べて判明しました。kernel config のパラメータを無効に変更して kernel をビルドし直すことで解消しました。

```
CONFIG_ANDROID_PARANOID_NETWORK=y ←これを n に変更
```

7.7.3 Pomera の ROM と Debian で時刻がずれる

DM200 の RTC から取得した時刻は、Pomera 標準のファームウェアでは JST として扱っているようです。

今回作成した Debian な rootfs では RTC を UTC として扱う設定になっていたため、Debian 側で JST として認識させるように変更することで解消しました。

7.7.4 USB-OTG 機能が動かない

ハード的に見ると ID ラインが結線されていないため、自動認識は無理そうでした。

ソフトで強制的に有効にすると端末が死ぬことがわかり、ドライバをいじって無理やり何とかすることで対応しました。

7.8 今後の課題

DM200 を実用するために以下の課題を解決していく必要があると思います。

- 内蔵 GPU(MALI400) の有効化
- USB-OTG 有効化時に死ぬ問題の調査
- デバイスツリーの変更を行ってみる
- suspend 処理の見直し
- 標準の高速起動処理を流用できないか
- 他の起動方法ができないか調べてみる

7.9 まとめ

DM200 をハックしてみました。

組み込み機器を解析する場合にはデバッグ用のシリアルコンソールを探してみるとよいでしょう。

また、GPL なソースコードは請求して中身を見てみるといろいろ勉強になり、ソースコードがあると自分で問題を対応できて楽しいです。

qemu-debootstrap は便利なコマンドですので覚えておくとよいと思います。

8 Debian 9 Stretch のネットワークインターフェース名について

yy_y_ja_jp

8.1 ネットワークインターフェース名

Debian 9 Stretch でネットワークインターフェースのデフォルトの名前が変わりました。

- Debian 8 Jessie まで
  - 有線 LAN インターフェース: eth0, eth1, ...
  - 無線 LAN インターフェース: wlan0, wlan1, ...
- Debian 9 Stretch から
  - 有線 LAN インターフェース: enp0s1 など、ハードウェア構成により異なる
  - 無線 LAN インターフェース: wlp1s0 など、ハードウェア構成により異なる

8.2 Jessie からアップグレードしたときは

- 今まで使っていたインターフェースの名前は変わりません
- 今後新たに使うインターフェースは新しい名前の形式になります – 新しい USB NIC を差したときなど

`/usr/share/doc/udev/README.Debian.gz` によると Debian 10 Buster では昔の方式はサポートされないと言ってるので、新しい名前に移行したほうがよいかもしれません。

8.3 なぜデフォルトの名前が変わったのか

今までの名前の変え方だとうまくいかないことがあったため、今回改定することになりました。

今まで dmesg などで、`eth0: renamed from eth1` などといったメッセージを見たことがある方もいると思います。もともと、ネットワークインターフェースの名前は OS の起動中に変えている（ことがある）ものなのです。

8.4 なぜ名前を変えているのか

- Linux カーネルはネットワークインターフェースを認識するたびに名前を付けている eth0, eth1, ...
- 認識する順番は保証されない

  ⇒ ネットワークインターフェースが複数あるときは、同じインターフェースが起動するたびに別の名前になることがある

  ⇒ カーネルで認識されたらユーザーランド側で名前を修正することにした

8.5 名前の変え方 – Jessie まで

udev パッケージの
/lib/udev/rules.d/75-persistent-net-generator.rules と/lib/udev/write_net_rules
（Stretch にはもう存在しません！）が名前を変えていました。
知らないインターフェースが現れたら次のように動作します。

1. 空いている次の番号の名前を探す: eth1 など
2.「そのインターフェースが現れたらその名前に変える」という udev ルールを
/etc/udev/rules.d/70-persistent-net.rules という設定ファイルに書き込む
3. その名前に変える

Stretch ではこれらのプログラムは無くなったため、/etc/udev/rules.d/70-persistent-net.rules ファイルが今後自動で書き換わることはありません。

8.6 名前の変え方 – Stretch から

udev パッケージの/lib/udev/rules.d/75-net-description.rules、udev の net_id ビルトイン、/lib/udev/rules.d/80-net-setup-link.rules、net_setup_link ビルトイン、
/lib/systemd/network/99-default.link などが名前を変えています。
知らないインターフェースが現れたら次のように動作します。

1. そのインターフェースを特定する情報（ハードウェア配置、MAC アドレス、ユーザー設定ファイルなど）を取得する
2. その情報に基づいて名前を決める: PCI バス 0 のスロット 1 に差さっているとき enp0s1 など
ただし USB NIC の場合は、USB の差す位置で名前が変わってほしくないので MAC アドレスベースの名前にする設定になっている (/lib/udev/rules.d/73-usb-net-by-mac.rules): enxaabbccxxyyzz など

8.7 名前の変え方 – （参考）一時期の Ubuntu

Debian にはないですが、Ubuntu は biosdevname パッケージで名前を決めている時期がありました。なお、現在のインターフェース名とは互換性がありません。

- インターフェース名: em0, p1p0 など
- 後発の udev net_id はさらに別の名前形式にした。そのため互換性がない

8.8 移行するには

1. どんな名前になるか調べる。eth0 なら
udevadm test-builtin net_id /sys/class/net/eth0

```
# udevadm test-builtin net_id /sys/class/net/eth0
calling: test-builtin
(略)
Parsed configuration file /lib/systemd/network/99-default.link
Created link configuration context.
ID_NET_NAME_MAC=enx000d0bxxyyzz
ID_OUI_FROM_DATABASE=BUFFALO.INC
ID_NET_NAME_PATH=enp0s20u2
Unload module index
Unloaded link configuration context.
```

ID_NET_NAME_*のうち、順に ONBOARD、SLOT、PATH がもしあればそれが使われる

(/lib/systemd/network/99-default.link)、ただし USB NIC は MAC が使われる
2. 今までの名前を使っている設定ファイルを置き換える
3. /etc/udev/rules.d/70-persistent-net.rules を消すかどこかに移動する
4. 再起動

8.9 移行せず、名前を変えさせないには

もともとインターフェースが 1 つしかない、eth0, wlan0 しかなかったのに変えてほしくないときなどは、いくつか方法があります。

- 空っぽの設定ファイルを/etc に置いて上書きする
  `ln -s /dev/null /etc/systemd/network/99-default.link`
- カーネルコマンドライン引数に net.ifnames=0 を追加
  /etc/default/grub に書いて update-grub を実行

もちろんこうするとカーネルが決めた名前そのままになるので、インターフェースが複数あったらうまくいかないことが起きるでしょう。

8.10 名前を自分で設定するには

- net_setup_link のユーザー設定ファイル /etc/systemd/network/*.link を作る (systemd.link(5))
  例えば MAC アドレス aa:bb:cc:xx:yy:zz のインターフェースを ethernet1 という名前にしたいなら

```
[Match]
MACAddress=aa:bb:cc:xx:yy:zz
[Link]
Name=ethernet1
```

  ただし、USB NIC の名前を設定するには
  `ln -s /dev/null /etc/udev/rules.d/73-usb-net-by-mac.rules`
  が必要
- udev ルールファイル /etc/udev/rules.d/*.rules を作る (udev(7))
  Jessie まで自動で書き込まれてきた/etc/udev/rules.d/70-persistent-net.rules ファイルに同じように追加する、など

8.11 うまくいかないときは

udev をデバッグするしかないです。

- 書いた*.link が期待通りに動くか見るには net_setup_link ビルトインを試す:
  `udevadm test-builtin net_setup_link /sys/class/net/eth0`
- udev を debug モードで起動する

```
# invoke-rc.d udev stop
# /lib/systemd/systemd-udevd --debug
```

- マシン起動時の udev を debug モードにする
  /etc/udev/udev.conf を udev_log="debug" にして dpkg-reconfigure linux-image-`uname -r` して再起動

名前変更に失敗してカーネルが決めた名前のままになっていることがあったりします... 見つけたらバグレポートしたほうがよいかもしれません。

8.12 まとめ

- udev が変わり、Debian 9 Stretch からはネットワークインターフェースが新しい名前の形式になった
- 新しいインターフェースは新しい名前になる
- Jessie からアップグレードした場合は直ちに影響はないが、変えたほうがいいらしい
- 移行前に新しい名前を見るには
  `udevadm test-builtin net_id /sys/class/net/<今の名前>`
  などが使える
- 名前付け替えを一切したくないときは `/etc` で `99-default.link` を上書き設定するとよい
- 自分で名前を付けたいときは`/etc/systemd/network/*.link` を作る、または昔ながらの udev ルール `/etc/udev/rules.d/*.rules` を作る
- うまく行かないときは udev をデバッグする。名前変更に失敗して元の名前のままになっていることがある

8.13 参考文献

- PredictableNetworkInterfaceNames
  `http://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/`
- systemd ソースパッケージ `src/udev/udev-builtin-net_id.c`
- udev パッケージ `/usr/share/doc/udev/README.Debian.gz`, man ページ `systemd.link(5)`, `udev(7)`
- biosdevname パッケージ (Ubuntu)

9 Debian Stretch のインプットメソッドの現状

あわしろいくや

9.1 はじめに

去る 6 月 17 日にリリースされた Debian GNU/Linux 9.0 Stretch の GNOME 版を（仮想マシンに）インストールしてみましたが、日本語入力が難解であるという印象を持ちました。私は Ubuntu でインプットメソッド関連をごそごそしている経験があり、その知識を元に Debian 9.0 の各デスクトップ環境でスムーズにインプットメソッドを使用する方法を探っていきたいと思います。

なお、今後の提案等も含まれていますが、私自身は全く手を動かせないということをあらかじめご了承ください。

9.2 パッケージがインストールされる仕組み

Debian Installer でインストールすると、選択した項目に応じて日本語関連のパッケージもインストールされます。パッケージの選択は tasks の仕組みを使用しています。

9.2.1 task-japanese

日本語を選択するとインストールされます。実際にインストールされるパッケージは次のとおりです。

```
manpages-ja, lv, fbterm, unifont, nkf, manpages-ja-dev
```

9.2.2 task-japanese-desktop

日本語とデスクトップ環境を選択するとインストールされます。実際にインストールされるパッケージは次のとおりです。

```
firefox-esr-l10n-ja | firefox-l10n-ja, fonts-vlgothic, fonts-ipafont,
uim, uim-anthy, uim-mozc, mozc-utils-gui, anthy, libreoffice-l10n-ja, libreoffice-help-ja, poppler-data
```

9.2.3 task-japanese-gnome-desktop

日本語と GNOME デスクトップ環境を選択するとインストールされます。実際にインストールされるパッケージは次のとおりです。

```
uim-applet-gnome, icedove, icedove-l10n-ja
```

おや、icedove パッケージは thunderbird に名前が戻りましたね……。

9.2.4 task-japanese-kde-desktop

日本語と KDE SC を選択するとインストールされます。最近は KDE デスクトップ環境は KDE SC (Software Compilation) と呼んでいます。実際にインストールされるパッケージは次のとおりです。

```
kde-l10n-ja, plasma-widget-uim
```

tasks にあるインプットメソッド (uim) を自動起動するためには `im-config` パッケージが必要ですが、これは `libuim-data` パッケージに引っ張られてインストールされます。

9.3 自動実行の仕組み

前述のとおり、インプットメソッドの自動起動には `im-config` パッケージが使われています。これは各インプットメソッドの情報がまとめられており、IBus と Fcitx と uim では次のようになっています。

- /usr/share/im-config/
  - data/21_ibus*
  - data/22_fcitx*
  - data/24_uim*

これは番号が若いほうが優先度が高くなっています。すなわち、デフォルトでは IBus と Fcitx と uim が同時にインストールされている場合は IBus が優先して起動されます。

もちろん手動での設定も可能になっており、前述の 3 つが同時にインストールされている場合に Fcitx を優先して起動するためには次のコマンドを実行します。

```
$ im-config -n fcitx
```

GUI の設定ツールもあります。

9.4 GNOME Shell の場合

では、ここから各デスクトップ環境ごとの挙動を見ていきます。

インストール時に GNOME を選択し、最初にログインすると英語キーボードと（接続されている場合）日本語キーボードを認識しています。

GNOME Shell は IBus と統合されており、現状 IBus と共に使われることが意図されています。しかし、Debian では前述のとおりデフォルトのインプットメソッドは uim なので、統合機能は全く使用することができません。

半角/全角キーを押すと日本語が入力できるようになるため、uim が正常に動作していることはわかります。しかし、ステータスは全くわかりません。

図 5　GNOME Shell で uim-toolbar-gtk3 を表示したところ

ps コマンドで確認すると uim-toolbar というプロセスがいます。ls コマンドで確認すると alternatives で管理されていることがわかり、実体は/usr/bin/uim-toolbar-gtk3-systray になっています。

現在の GNOME Shell にもレガシートレイとして uim-toolbar-gtk3-systray を表示する機能がありますが、なぜか表示されません。gnome-shell-extension-top-icons-plus をインストールして有効にすれば表示されるはずですが、やはり表示されません。uim-toolbar-gtk3-systray の起動するタイミングが早すぎるのが問題と思われ、手動で起動すれば表示されます。

現実的には alternatives で uim-toolbar-gtk3 に切り替え、ツールバーを表示するのがいいでしょう。

9.5 KDE Plasma の場合

インストール時に KDE を選択し、最初にログインすると右下のトレイに妙にカラフルなアイコンがありますが、これはおそらく uim-toolbar（実体は uim-toolbar-gtk3-systray）が 1 つのアイコンの大きさに圧縮されているものと思われます。uim には uim-toolbar-qt4/5 があるので、これを選択するのがいいでしょう。

前述のとおり plasma-widget-uim がインストールされているので、これを有効にしたいところですが、追加できるウィジェットに表示されません。おそらく現状の実装が KDE Plasma 4.x 対応で 5.x に対応していないのが原因と思われます。確認したところ Debian Jessie では追加できるウィジェットに表示されていました。

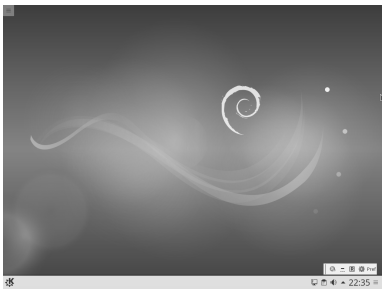


図 6　KDE Plasma で uim-toolbar-qt5 を表示したところ

9.6 Cinnamon の場合

インストール時に Cinnamon を選択し、最初にログインすると右下のトレイにアイコンが表示されます。一見このままでもよさそうですが、入力が直接入力なのかひらがな入力なのかカタカナ入力なのかそれ以外なのか極めてわかりにくいです。よって uim-toolbar-gtk3 を使用するといいでしょう。

図 7　Cinnamon のログイン直後の状態

9.7 Xfce の場合

インストール時に Xfce を選択し、最初にログインすると右上に uim のステータスがすべて表示されています。今回は紹介しませんが、MATE や LXDE を選択してもこのように表示されると思われます。理想的ではありますが、システムトレイのツールキットが GTK+ 2 か 3 かで表示方法を分けているようであり、Xfce では前者なのでいずれ対応しなくなると思われます。sid の MATE はすでに GTK+ 3 でビルドされているため、一つ分のトレイアイコンしか表示されないはずです。

図 8　Xfce のログイン直後の状態

9.8 uim-mozc の問題

uim-mozc を使用すると、Mozc の各種ツール (mozc_tool) が起動しません[*9]。正確には protobuf の問題[*10]ですが、もう何年も未解決です。IBus や Fcitx にはこのような問題がないため、uim-mozc を避けることができるのであれば避けたほうがいいでしょう。

9.9 短期的な修正の提案

以上の現状を鑑みるに、インストールする人が多そうな GNOME が特に影響が大きいので、短期的にどうにかできるものであればしたほうがいいのかもしれません。GNOME の場合は専用のタスクがあるので、これだけをいじればなんとかなるということもあります。

GNOME は現状 IBus で使用することしか考えられていないため、IBus 一式と初期セットアップ (gnome-initial-setup) を task-japanese-gnome-desktop に追加すれば、おおむね問題が解決します。初期セットアップは初回ログイン時に各種設定を行いますが、その中にキーボードやインプットメソッド（GNOME では同様に扱われる）の設定も含まれるため、設定の難易度が大幅に低下します。

図 9　初期セットアップ（Ubuntu GNOME 17.04 のもの）

9.10 長期的な修正の提案

長期的には、uim から IBus に移行するのがよさそうに思います。uim のメンテナンスは現状特定の個人に負荷が集中していますが、IBus であればある程度負荷が分散されているというのが大きな理由です。特に各種ツールキットやデスクトップ環境向けの対応を考えなくてもいいのが楽です。技術的には概ねこれまで挙げてきた問題が解決されますが、uim では問題とならなかったことが問題となり得ます。具体的には愛用者が多いと思われる SKK の実装である ibus-skk やそのバックエンドの libskk が現在メンテナーがいないこと、fbterm で使用する ibus-fbterm がパッケージになっておらず、またなったとしてもどの程度実用的なのか疑問符がつくことです。

もちろん Fcitx も選択肢に入ってきますが、現状の Fcitx は Wayland に非対応でかつツールキットは Qt 4 であり、現在最新バージョンである Fcitx5 の開発中ですが、Fcitx5 のリリースが早いか Qt 4 の削除が早いかは誰にもわかりません。ちなみに中国語（簡体字）では Fcitx が使用されているため、このリソースに乗っかれるというメリットはあります。

uim を使い続けるのであれば、uim-toolbar の切り替えを alternatives でやるのではなく、環境変数を使用してデスクトップ環境に応じてより適切な選択をするのがいいと思います。ただし Qt/KDE Plasma 5 対応をより進め、また開発が進んでいる GTK+ 4 にも対応していくのは、かなりの労力が必要なことでしょう。

[*9] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=700307
[*10] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=721791

10 Rethinking Debian release

Yamane Hideki

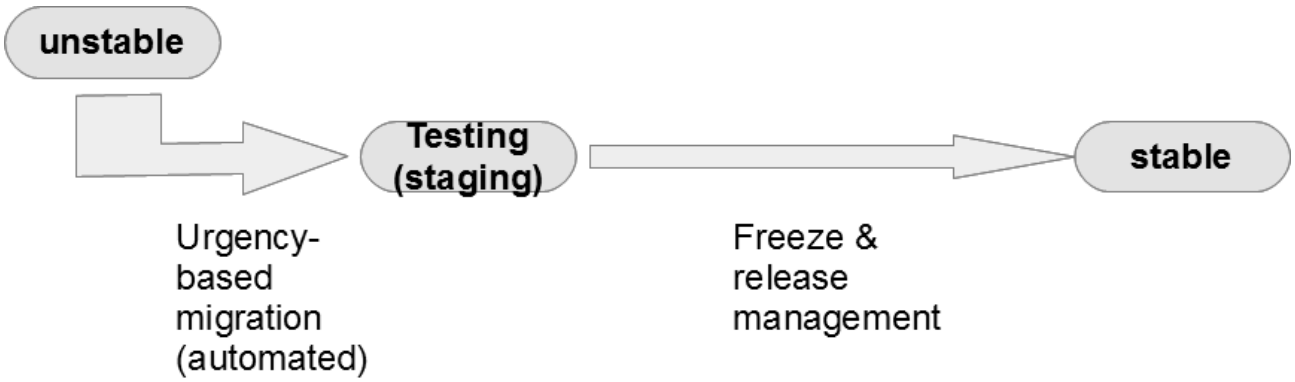
この発表は議論のたたき台として Debian 勉強会で発表されたもので、決定事項等ではありません。この資料は勉強会での英語の発表資料[*11]から編者 (吉田) がコンバートしたものです。

10.1 Why rethink release?

- “ Debian (stable) is old (for our usage) ”
- “ Some packages are still old and buggy,but no update ”

10.2 Current Release Migration

- Unstable → Testing
  - Based on urgency (high,medium,low)
    - Blocked by Release Critical bug
- Testing → Stable
  - (Looong) Freeze and release



10.3 Stable release management = many-legged-race

Push release management causes
“ many(60,000-70,000 Packages)-legged-race ”

[*11] https://wiki.debian.org/HidekiYamane/material?action=AttachFile&do=view&target=Rethinking-debian-release.pdf

10.4 Another Problem:Is that package tested, really?

- Who tests it?
  - Sometimes "Passive test" doesn't work well
- Code never matures
  - Code != Wine / Whiskey
  - But time makes features to rot...

10.5 Worst scenario

- Upload to unstable
  - → no one cares it
  - → no bugs filed
  - → migrate to testing
  - → release stable
  - → found bugs in stable, but leave it... (since put not tiny changes to stable is not easy task···)
  - → bad user experience
  - → bad reputation
  - → less user
  - → less developer...

10.6 Answer (1):" Active " migration

- Same as other distros
  - Gentoo: mask (package flag)
  - Fedora: bohdi (voting system)
  - openSUSE: openQA (automated test)
- "pull" migration system via vote by users & maintainers
  - " Package quality " is guaranteed by safety harness (pipeline)
- It ensure "it works" by someone, at least

10.7 Pull is better than push

- " Push " Testing to stable migration
  - Thousands changes in one time
  - itemizeHandled by few release managers
    - = capacity overflow → burnout...
- " Pull " migration
  - Several changes in one time
  - Handled by hundreds advanced users & maintainers

10.8 Answer (2):New distribution

Why we need "new distribution"?
Average users never use unstable or testing, they use "released" one (= stable)
" Innovators theory " (by Everett M. Rogers)

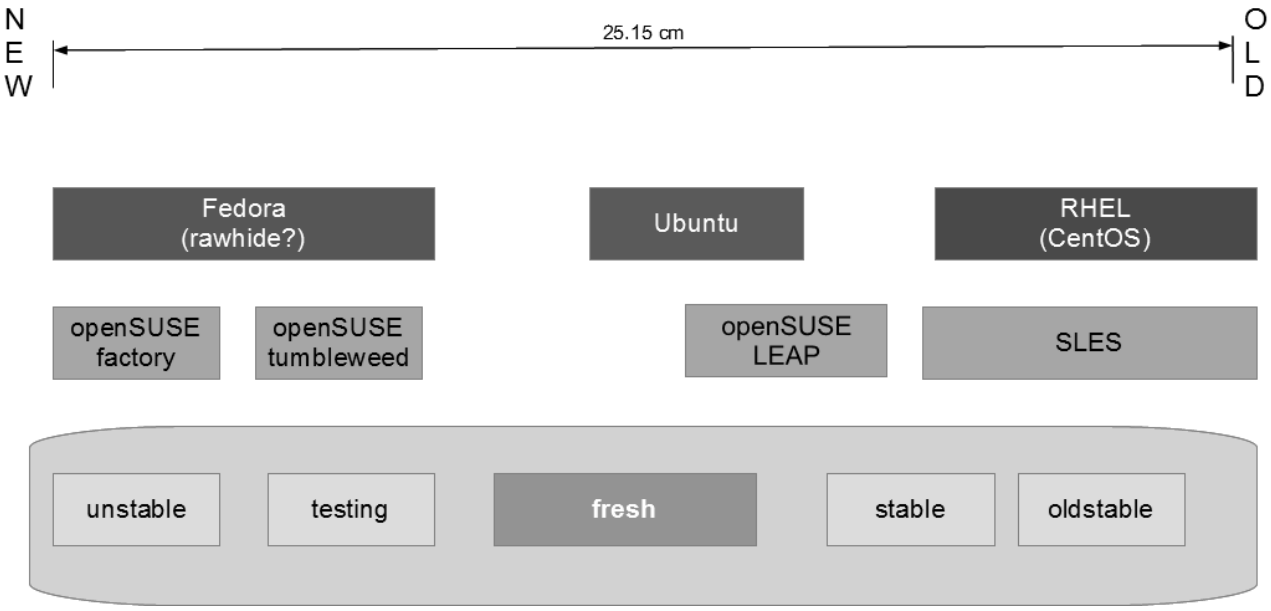
- Innovators : 2.5 % (unstable)
- Early Adopters : 13.5 % (testing)
- Early Majority : 34.0 %
- Late Majority : 34.0 % (stable)
- Laggards : 16.0 % (oldstable)

" Fresh " distribution

- Innovators : 2.5 % (unstable)
- Early Adopters : 13.5 % (testing)
- Early Majority : 34.0 % " Fresh "
- Late Majority : 34.0 % (stable)
- Laggards : 16.0 % (oldstable)

We can get more users! (100 / 66 = 150 %)

10.9 Positioning



10.10 Fresh?

Borrow name from LibO :)
Target: Average users (Early Majority)
Release every one or two week

- Rolling release
- Predictable scheduled release

Pull change sets

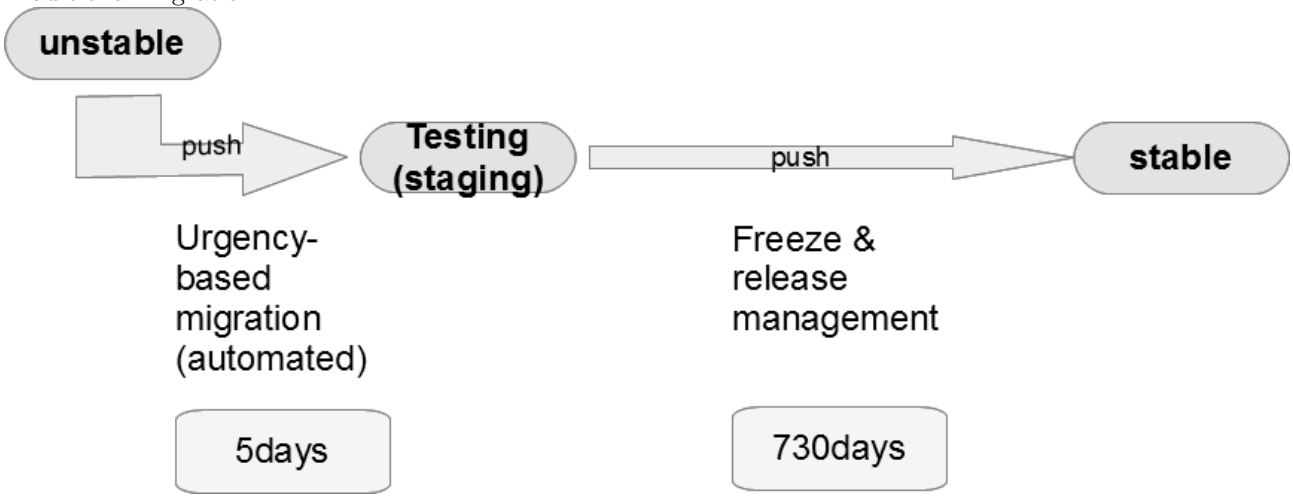
- Sustainable deploy
- Ensure changes, not break anything

Not push changes into stable directly
Why new “ fresh ” distribution?

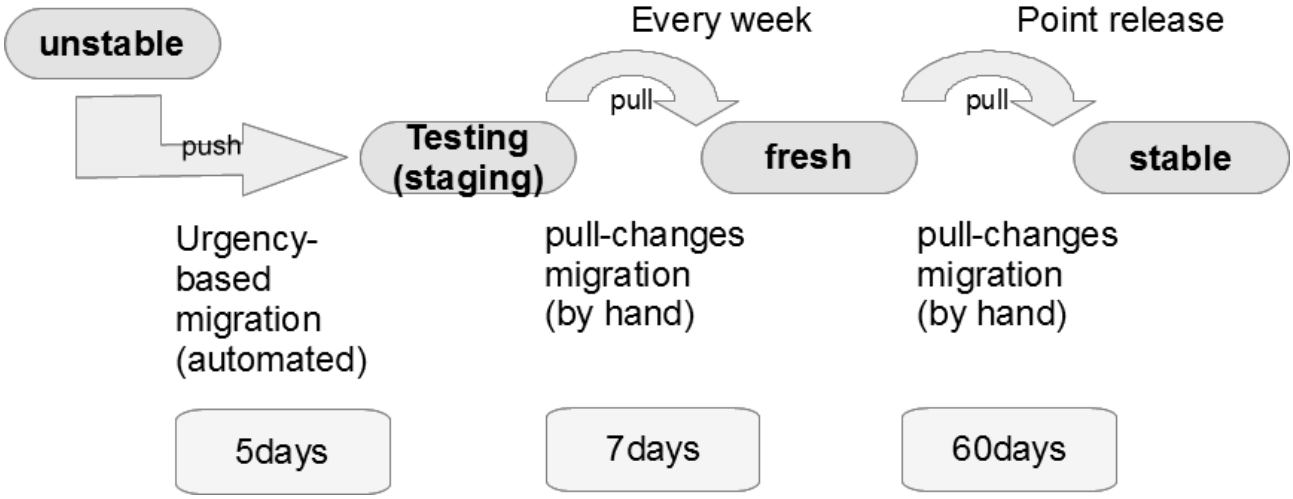
- Users expect stable as stable (not changed so much)
- We afraid to break stable release

10.11 Migration cycle time

Traditional migration



Add “ Fresh ” distribution



10.12 Shorten cycle time

Before : 730 days (minimum 180days)
After : 12 days
15-60 times faster delivery!

Maximize added-value

10.13 Change the rule!

There was a reason to make rules

- Unstable - Testing - Stable
- Long freeze term and release

But situation has changed, then rules should be changed, too. Because its rule becomes bottlenec

10.14 Faster release introduce more bugs?

Q: It may introduce more bugs!
A: “ test early and fail fast ” on fresh stage, but less bugs in stable since more test users watch it.
Testers

- Previous : 2.5 + 13.5 = 16.0
- Fresh : 2.5 + 13.5 + 34.0 = 50.0 → 300 %

10.15 “ Fresh ”: Pros & Cons

Pros)

- 150 % users, 300 % testers
- 60 times faster release
- Same cadence, its release date & changes are predictable
- Changes in each release are small, users can bite it (No Big Bang release)
  - Less freeze term for next release
  - Not need to hassle to make huge release note
  - Moe ”real acceptance test” by real users for next stable release

Cons)

- It just costs
- Infrastructure change
- Docs & website update
- More release manager & publicity work
- Prepare security fix (but delta with unstable is small, right?)
  - maybe it reduce backport effort in stable

10.16 Metrics?

More testers / More users

- BTS number
- RC in stable / bugs in stable
- Download number

11 Debian Trivia Quiz

ところで、みなさん Debian 関連の話題においついていますか？Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけでははりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容と Debian Project News からです。

問題 1. 先日、unstable の openssl パッケージが一部機能を無効な状態でビルドしたバイナリに置き換えられました。無効にされた機能とはなんでしょうか。

☐ A SSL 3.0
☐ B 3DES/RC4
☐ C TLS 1.0/1.1

本資料のライセンスについて

本資料はフリー・ソフトウェアです。あなたは、Free Software Foundation が公表した GNU GENERAL PUBLIC LICENSE の ”バージョン 2”もしくはそれ以降が定める条項に従って本プログラムを再頒布または変更することができます。

本プログラムは有用とは思いますが、頒布にあたっては、市場性及び特定目的適合性についての暗黙の保証を含めて、いかなる保証も行ないません。詳細については GNU GENERAL PUBLIC LICENSE をお読みください。

ソースコードについて

本資料のソースコードは Git を使って `git://anonscm.debian.org/tokyodebian/monthly-report.git` からダウンロードできます。以下に方法を示します。

```
$ git clone git://anonscm.debian.org/tokyodebian/monthly-report.git
```

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
    51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                        Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
```

access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

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, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year  name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Debian オープンユーズロゴ ライセンス

```
Copyright (c) 1999 Software in the Public Interest
Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN
AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR
IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

12 Debian Trivia Quiz 問題回答

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

1. 1. C TLS 1.2 に対応したアプリケーションを増やすため、あえて TLS 1.0/1.1 を動かないようにしてでバグを出すことで修正を促しています。ただ、このビルドオプションのまま Buster がリリースされるかはわかりません。`https://lists.debian.org/debian-devel-announce/2017/08/msg00004.html`

『あんどきゅめんてっど でびあん』について

本書は、東京および関西周辺で毎月行なわれている『東京エリア Debian 勉強会』および『関西 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 2017/07〜2017/11 まで内容は無保証、つっこみなどがあれば勉強会にて。

あんどきゅめんてっど でびあん 2017 年冬号

2017 年 12 月 29 日　　初版第 1 刷発行

東京エリア Debian 勉強会/関西 Debian 勉強会 （編集・印刷・発行）