Grand Unified Debian

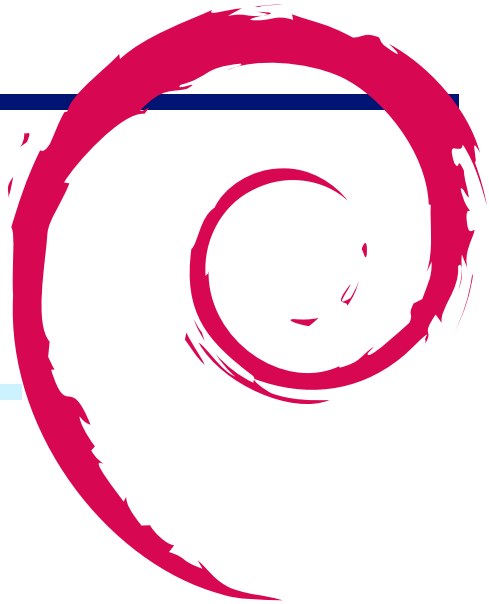
銀河系唯一のDebian専門誌

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 2015 年夏号 2015 年 8 月 16 日 初版発行

デビアン勉強会

目次



1 Introduction

上川 純一, 山下 尊也

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。

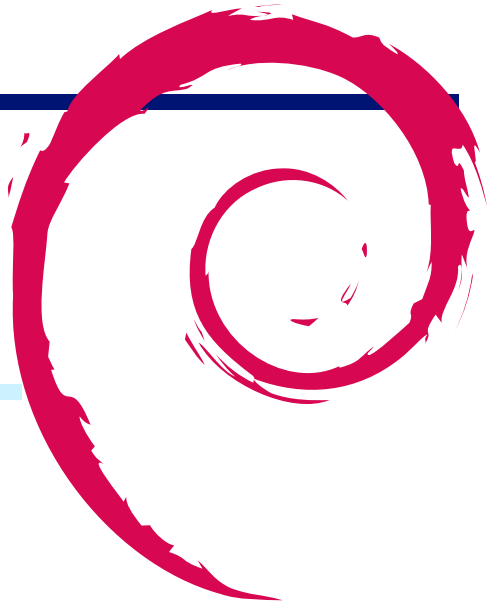
1.2 関西 Debian 勉強会

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。

- メーリングリストや掲示板ではなく、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　Wheezy から Jessie まで

かわだてつたろう

Debian 8 Jessie がリリースされました。

今回もリリースまでに色々とありましたので、Wheezy リリースから Jessie リリースまでを関西 Debian 勉強で何度かご紹介してきた Debian Project の近況から振り返ってみましょう。

2.1　Wheezy リリース

Debian 7 Wheezy は 2012 年 6 月 30 日にフリーズされ[*1]、一年近くのフリーズ期間を経て、2013 年 5 月 4 日にリリースされました。[*2]

ここから Jessie の開発が始まります。

2.2　2013 年

5 月 6 日、Jessie の開発が宣言

「Wheezy is out! Jessie is created and receives updates!」[*3]

8 月 11 日〜18 日、DebConf 13 がスイスのヴォーマルキュで開催[*4]

8 月 25 日、リリースチームより開発サイクルの提案

「Bits from the Release」[*5]

- Changing testing migration (NEW-TEST)
- Key-packages and automated removals (AUTO-RM)
- Automating Architecture (re-)Qualification (ARCH)
- Roll call for porters (ROLL-CALL)

9 月 11 日、リリースチームからリリースゴールの呼び掛け

「Call for Jessie Release Goals」[*6]

[*1] https://lists.debian.org/debian-devel-announce/2012/06/msg00009.html
[*2] http://www.debian.org/News/2013/20130504
[*3] http://lists.debian.org/debian-devel-announce/2013/05/msg00004.html
[*4] http://debconf13.debconf.org/
[*5] http://lists.debian.org/debian-devel-announce/2013/08/msg00006.html
[*6] http://lists.debian.org/debian-devel-announce/2013/09/msg00001.html

- Native systemd support in every package with sysv scripts
- Hardening of ELF binaries (carry over from Wheezy)
- debian/rules to honor CC/CXX flags
- clang as secondary compiler
- piuparts clean archive
- Cross Toolchains in the archive
- Make the base system cross-buildable
- SELinux
- UTF-8

10 月 13 日、フリーズ日が 2014 年 11 月 5 日 23 時 59 分 (UTC) になるとアナウンス

「Bits from the Release Team (Jessie freeze info)」[*7]

11 月 28 日、対象アーキテクチャから s360 と ia64 が除外

「Release sprint results - team changes, auto-rm and arch status」[*8]

2.2.1 systemd

フリーズ日のアナウンスがされる前後に、

「systemd effectively mandatory now due to GNOME」[*9]

から始まって、

「Proposal: switch default desktop to xfce」[*10]

に続き、

「Proposal: switch init system to systemd or upstart」[*11]

へと、systemd の話題で debian-devel@debian.org を賑わせることとなりました。
systemd に関する話題はその後も続くことになります。

2.3 2014 年

明けて 2014 年、2 月 11 日に Jessie から Linux アーキテクチャの init system を systemd とする CTTE の決定がアナウンスされます。

「[CTTE #727708] Default init system for Debian」[*12]

この決定に端を発して debian-devel@debian.org は荒れに荒れ、「Code of Conduct」[*13] の採択へと繋がります。
その一方で Jessie の開発は着々と進んでおり
3 月 19 日、インストーラチームから Debian Installer Jessie Alpha1 がリリース

[*7] http://lists.debian.org/debian-devel-announce/2013/10/msg00004.html
[*8] http://lists.debian.org/debian-devel-announce/2013/11/msg00007.html
[*9] http://lists.debian.org/debian-devel/2013/10/msg00444.html
[*10] http://lists.debian.org/debian-devel/2013/10/msg00496.html
[*11] http://lists.debian.org/debian-devel/2013/10/msg00651.html
[*12] https://lists.debian.org/debian-devel-announce/2014/02/msg00005.html
[*13] https://www.debian.org/vote/2014/vote_002

「Debian Installer Jessie Alpha1 release」[*14]

4 月 26 日、SPARC がリリースアーキテクチャから取り除かれる

「SPARC removed from Jessie」[*15]

5 月 1 日、リリースチームからは 6ヶ月を切ったフリーズまでのタイムテーブルが報告

「Bits from the Release Team - Freeze, removals and archs」[*16]

5 月 22 日、Debian GNU/Hurd が 80% のパッケージをカバーし SysVinit が動作することなどの活動報告

「Bits from the Debian GNU/Hurd porters」[*17]

6 月 3 日、MATE Packaging Team より MATE 1.8 の Debian 入りが報告

「MATE 1.8 has now fully arrived in Debian」[*18]

6 月 18 日、libc が eglibc から glibc に戻る

「Accepted glibc 2.19-3experimental0 (source all)」[*19]

7 月 30 日、Debian Linux カーネルチームより Jessie の Linux カーネルを 3.16 にすると報告

「Linux kernel version for jessie」[*20]

8 月 13 日、インストーラチームから Debian Installer Jessie Beta 1 がリリース

「Debian Installer Jessie Beta 1 release」[*21]

8 月 22 日〜31 日、DebConf14 がアメリカのオレゴン州ポートランドで開催[*22]

9 月 4 日、Cinnamon の全パッケージが Jessie に入ったと報告

「Cinnamon environment now available in testing」[*23]

9 月 19 日、デフォルトデスクトップ環境が GNOME に決まる

「Accepted tasksel 3.25 (source all) into unstable」[*24]

そして、11 月 5 日のフリーズを迎えました。

「Bits from the release team: Jessie Freeze」[*25]

2.3.1 systemd

2014 年も次のようなスレッドで systemd の話題が debian-devel@debian.org を賑わせ続けていました。

[*14] https://www.debian.org/devel/debian-installer/News/2014/20140319
[*15] https://lists.debian.org/debian-devel-announce/2014/04/msg00012.html
[*16] https://lists.debian.org/debian-devel-announce/2014/05/msg00000.html
[*17] https://lists.debian.org/debian-devel-announce/2014/05/msg00006.html
[*18] https://lists.debian.org/debian-devel/2014/06/msg00041.html
[*19] https://packages.qa.debian.org/g/glibc/news/20140618T220008Z.html
[*20] https://lists.debian.org/debian-devel-announce/2014/07/msg00006.html
[*21] https://lists.debian.org/debian-devel-announce/2014/08/msg00005.html
[*22] http://debconf14.debconf.org/
[*23] https://lists.debian.org/debian-devel/2014/09/msg00111.html
[*24] https://packages.qa.debian.org/t/tasksel/news/20140919T193809Z.html
[*25] https://lists.debian.org/debian-devel/2014/11/msg00174.html

「systemd-fsck?」[*26]
「How to avoid stealth installation of systemd?」[*27]
「systemd now appears to be only possible init system in testing」[*28]

そして先述の「Code of Conduct」の採択となり、さらにはデフォルト以外の init システムをどの程度サポートするかの GR(一般決議) へと繋がります。この GR は「GR では決めない」というなんともアレな結果となります。

「General Resolution: init system coupling」[*29]

このような状況のなか Debian Installer や debhelper などを開発し精力的に活動してきた Joey Hess(joeyh) が Debian を離れるというのも一つの大きな話題でした。

「so long and thanks for all the fish」[*30]

2.3.2 Squeeze LTS

Jessie でも Wheezy でもなく Squeeze ですが、LTS 提供の検討が始まり、4 月には i386 と amd64 アーキテクチャに限って 2016 年 2 月までのセキュリティサポート開始がアナウンスされ、6 月から提供が開始されました。

「Bits from the Security Team」[*31]、
「Long term support for Debian 6.0 Announced」[*32]
「Debian 6 debuts its long term support period」[*33]

2.4 2015 年

続けて本年の 2015 年を振り返ってみます。
1 月 26 日、インストーラチームから Debian Installer Jessie RC1 がリリース

「Debiain Installer Jessie RC1」[*34]

2 月 4 日、リリースチームからフリーズポリシーに最終段階に入ったとアナウンス

「Permanent removals from testing for Jessie」[*35]

3 月 8 日、リリースチームからのリリース状況で 4 月にリリースできるだろうと報告

「Status on the Jessie release」[*36]

3 月 28 日、インストーラチームから Debian Installer Jessie RC2 がリリース

「Debian Installer Jessie RC 2 release」[*37]

3 月 31 日、Jessie のリリース日は 2015 年 4 月 25 日とアナウンス

[*26] https://lists.debian.org/debian-devel/2014/05/msg00255.html
[*27] https://lists.debian.org/debian-devel/2014/07/msg00010.html
[*28] https://lists.debian.org/debian-devel/2014/07/msg00839.html
[*29] https://www.debian.org/vote/2014/vote_003
[*30] https://lists.debian.org/debian-devel/2014/11/msg00174.html
[*31] https://lists.debian.org/debian-devel-announce/2014/03/msg00004.html
[*32] https://lists.debian.org/debian-announce/2014/msg00002.html
[*33] https://lists.debian.org/debian-announce/2014/msg00004.html
[*34] https://lists.debian.org/debian-devel-announce/2015/01/msg00005.html
[*35] https://lists.debian.org/debian-devel-announce/2015/02/msg00000.html
[*36] https://lists.debian.org/debian-devel-announce/2015/03/msg00002.html
[*37] https://lists.debian.org/debian-devel-announce/2015/03/msg00015.html

「Jessie Release Date: 2015-04-25」[*38]

4 月 19 日、インストーラチームから Debian Installer Jessie RC3 がリリース

「Debian Installer Jessie RC 3 release」[*39]

4 月 23 日、CD チームから Jessie の準備は整ったと報告

「Ready for Jessie! (aka bits from the debian-cd team)」[*40]

4 月 25 日、Debian 8 Jessie リリース

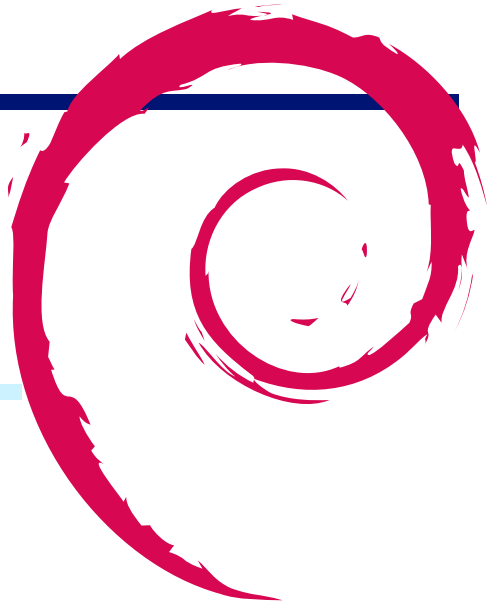
「Debian 8 "Jessie" released」[*41]

2.5 Stretch, Buster...

次の Debian 9 のコードネームは"Stretch" です。さらにはその次 Debian 10 のコードネームは"Buster" とまで決まっています。

次回はおそらく 2 年後、2017 年春頃の Debian 9 Stretch のリリースパーティでまたお祝いしましょう。

[*38] https://lists.debian.org/debian-devel-announce/2015/03/msg00016.html
[*39] https://lists.debian.org/debian-devel-announce/2015/04/msg00008.html
[*40] https://lists.debian.org/debian-devel-announce/2015/04/msg00009.html
[*41] https://lists.debian.org/debian-announce/2015/msg00001.html



3 Debian からみた Linux Mint

野島 貴英

3.1 はじめに

OSC[*42]にて、ブースを出した際、様々な来場者の方々と意見交換をしています。この時印象深かった事として、特に若い年齢の方々が、以下のディストリビューションを使っているようです。

- Arch Linux
- Linux Mint

Debian はコミュニティー主導により進化し続けるディストリビューションです。他のディストリビューションにある良い点、Debian にある他のディストリビューションとの比較で悪い点、Debian の目指すべき立ち位置などは、他のディストリビューションとの比較でわかることも多いです。これらの違いを比較して、取り込むべき良い点があるのであれば、Debian に取り込まれるべきと考えます。

今回、OSC の件もあり、ちょうど良い機会なので、先月の Arch Linux に続き、今月は Linux Mint を題材に、Debian とどのような違いがあるのかを調べてみました。

3.2 Linux Mint とは

Linux Mint は、2006 年に誕生しました [1]。Ubuntu あるいは Debian をベースにして作られています。

軽量で、かつ、Ubuntu/Debian をベースとしながら、Ubuntu/Debian ではポリシー上提供しないようなプロプライエタリなドライバ/マルチメディア系ライブラリも予め搭載されています。初心者にも使いやすい事を狙いとして作られているディストリビューションです。

3.3 Debian 上の仮想環境にインストールしてみる

ここでは KVM を使って Debian 上に Linux Mint をインストールしてみます。なお、Linux Mint の ISO イメージは、基本的に Live イメージでもあり、インストーラはこの Live イメージに梱包されている通常のアプリケーションとなります。

Linux Mint はいくつかの種類にわかれたインストール用の ISO イメージが用意されています。

- 導入されるデスクトップ環境別にイメージがあり、cinnamon 版、MATE 版、xfce 版、KDE 版があります。
- 梱包されているマルチメディア用ミドルウェアについて、国によっては配布が違法となってしまうもの (例:libdvdcss 等) を含んでいるため、これらを全て排除した NOCODEC 版がそれぞれ用意されています。日本で使うのであれば、法的問題から NOCODEC 版をおすすめします [2]。

[*42] http://www.ospn.jp/

- 基本は Ubuntu をベースにしていますが、Debian をベースにした物（LMDE) もあります。

ここでは、最近リリースされた Linux Mint 17.1(Rebecca) の MATE Nocodec の 64bit 版を debian sid 上に導入した KVM で動作させてみます。なお、Cinnamon 版ですが、筆者の元では virt-viewer との相性が非常に悪かった（cirrus エミュレーション及び、qxl エミュレーション共に）為、評価は断念しました。

Step 1. Debian マシンに KVM/libvirt/virtinst をインストールしておきます。

```
$ sudo aptitude install qemu-kvm libvirt-bin virtinst
```

Step 2. 今回使う ISO イメージファイルである、linuxmint-17.1-mate-nocodecs-64bit.iso をダウンロードしておきます。

Step 3. 仮想ディスクを 9GBytes 以上の大きさで作成し、virt-install コマンドで起動します。また、guest OS へ割り当てる仮想メモリは 1GBytes 以上は用意しておかないとインストーラが途中で終了（OOM killer で強制終了）したりします。なお、ネットワークデバイスとして指定している br0 は bridge/NAT が有効でかつ、dnsmasq で DHCP/DNS が有効にしてあります。(参考文献：[3]、[4])

```
$ sudo qemu-img create -f raw /var/lib/libvirt/images/mint-01 9G
$ sudo virt-install --connect=qemu:///system -n mint-01 --ram 1024 \
     --cdrom /home/yours/arch-linux/linuxmint-17.1-mate-nocodecs-64bit.iso \
     --disk /var/lib/libvirt/images/mint-01,bus=virtio,size=9,format=raw,cache=writeback \
     --vnc --hvm --accelerate --bridge=br0,model=virtio
```

Step 4. Linux Mint が MATE デスクトップ環境にてユーザ mint 権限の元ログイン済み立ち上がります (図 1)。

Step 5. デスクトップ上にある、インストーラのアイコンをクリックしてインストーラを立ち上げ、表示される質問に答えていきます。質問全部に答えるとインストールが開始されます (図 2)。

Step 6. インストールが完了するとリブートを促されますので、リブートします。

Step 7. ログインし、「設定」→「Languages」にて、日本語 IM の導入メニューがあるので、好きな IM を選んで追加のパッケージを導入して下さい。導入完了後、「Input method」プルダウンを導入した IM が選択されているようにしておきます (図 3)。

Step 8. 再度リブートします。

Step 9. 再度ログインすると、導入した日本語の IM が使えるようになっています (図 4)。

図 1　ブートの様子

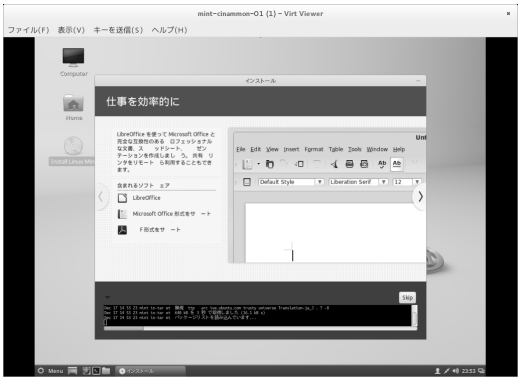


図 2　インストーラ起動の様子

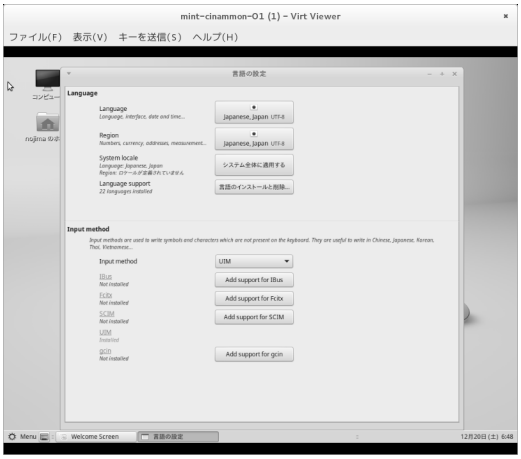


図 3　日本語 IM 導入の様子

図 4　インストール完了

3.4　Linux Mint と Ubuntu/Debian の関係

図 5 に Linux Mint と Ubuntu/Debian の関係を図示しておきます。Linux Mint 側ではデスクトップ環境及び MintTools のみを Linux Mint 側で開発しており、残りの足らない部分をすべて Ubuntu のパッケージから導入しているという作りとなります。

このため、パッケージの更新を行う場合、Debian で行うように、うっかり apt-get/aptitude などで full-upgrade すると、Linux Mint では意図していない形で副作用のあるパッケージがうっかり導入されてしまう事があるため、パッケージの更新の時は「システム管理メニュー」の「アップデートマネージャ」を使ってアップグレードしなければならない等の制約があります。

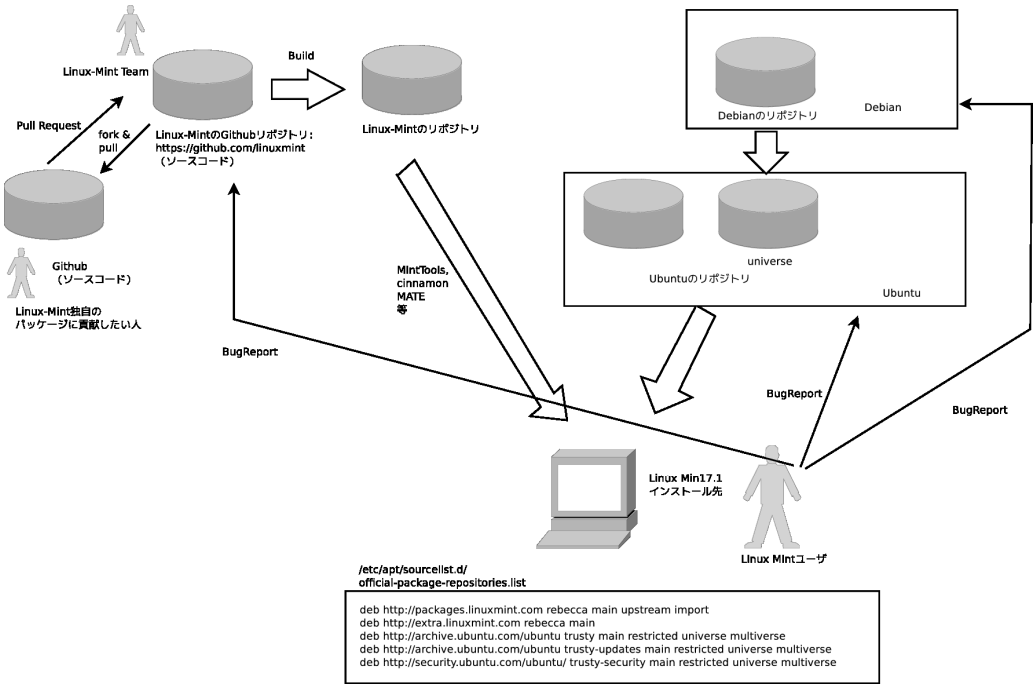


図 5　Linux Mint と Ubuntu/Debian の関係

3.5　Linux Mint の良い所

以下に Linux Mint の良い所を載せます。

- Linux Mint 側でメンテナンスされている部分の作りが非常に丁寧であり、通常の利用であれば、GUI のメニューを操作するだけですぐに使えるのが良い点です。他ディストリビューション（Debian 含む）のようにメニューの各項目の動作が不完全、あるいはメニューの項目が不完全ということは非常に少ないように思えます。
- Linux Mint 側の community サイト[*43]には、投票システムや、ユーザの投稿のランキングなどの様々な統計がすぐにみれるようになっています。さらに、アイデアの投稿場所があり、こちらに投稿されたアイデアについて、他のユーザが好感度を投票でき、どのアイデアが他のユーザに評価が高いかがすぐにわかるようになっています。

3.6 cinnamon を Debian unstable に導入してみる

2014/9/4 に Debian の testing に Linux Mint で利用されているデスクトップ環境である cinnamon が入ったとのアナウンスがありました。

ひょっとして、Debian に cinnamon を入れると、Linux Mint そっくりになったりするのかも？と思い、試しに Debian sid へ導入してみました。

Step 1. Debian sid をデスクトップ環境を全く導入しない最小限の構成で導入します。

Step 2. cinnamon デスクトップ環境を導入します。

```
$ sudo aptitude install cinnamon-desktop-envirionment
```

Step 3. Debina sid をリブートします。すると、lightdm が立ち上がり、ログイン画面が現れます。

Step 4. ログインすると、Debian らしいルックアンドフィールで cinnamon のディスクトップ環境が動作します。（図 6)

使うとわかるのですが、Debian の他デスクトップ環境の完成度としては普通の完成度です。Debian に慣れているのであれば、Debian で導入出来る他デスクトップ同様、管理の流儀も利用も全く違和感がありません。しかしながら、Linux Mint で提供されている程の、システム全体との統合はメニュー構成も含めて実現出来ていない状況です。このため、仮に Linux Mint ユーザが Debian へ乗り換えることを考えた場合、Debian に cinnamon を導入してみても、物足りなさを感じてしまうかも？と思いました。

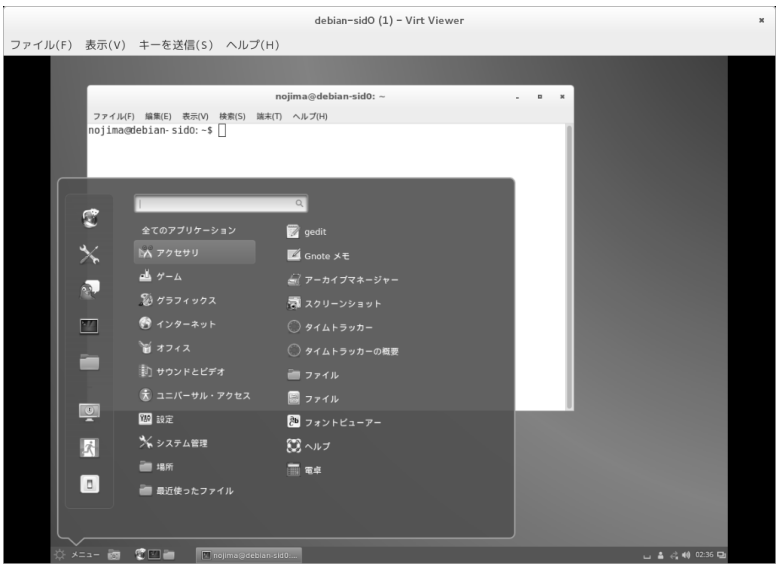


図 6 Debian に cinnamon を導入した様子

[*43] http://community.linuxmint.com/

3.7 おわりに

今回は、Linux Mint について Debian からの視点で見てみました。Linux Mint community のサイト、Linux Mint 並に GUI とシステム管理が丁寧に統合されたデスクトップ環境の構成は Debian を考える上で非常に参考になります。将来こういった点を Debian で導入・改善していくと良いと思いました。

参考文献

[1] Linux Mint UserGuide 中 History の章,http://www.linuxmint.com/documentation/user-guide/MATE/english_4.0.pdf

[2]「Linux Mint のミラーを再開しました」,http://ftp-admin.blogspot.jp/2013/06/linux-mint.html

[3]「Debian 開発者の KDE 環境あれこれ」 第 85 回東京エリア Debian 勉強会資料,http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201202.pdf

[4]「Debian で dnsmasq を使う」 第 109 回東京エリア Debian 勉強会資料,http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201402.pdf

4 Raspberry Pi 2 Model B に Debian Jessie / armhf をインストールする

岩松 信洋

4.1 はじめに

2015 年 2 月 2 日に新しい Raspberry Pi「Raspberry Pi 2 Model B」が発売されました。今回の Raspberry Pi 2（以下、RPi2）は今までの Raspberry Pi（RPi）とは異なり、SoC がアップグレードされたものになっています。FPU を持っているにも関わらず Debian では armel をつかわなければなりませんでしたが、RPi2 では Debian の armhf アーキテクチャが利用できるようになります。

本資料では Debian から見た RPi と RPi2 の違いと、RPi2 に Debian Jessie / armhf をインストールする方法について紹介します。

4.2 RPi と RPi2 の違い

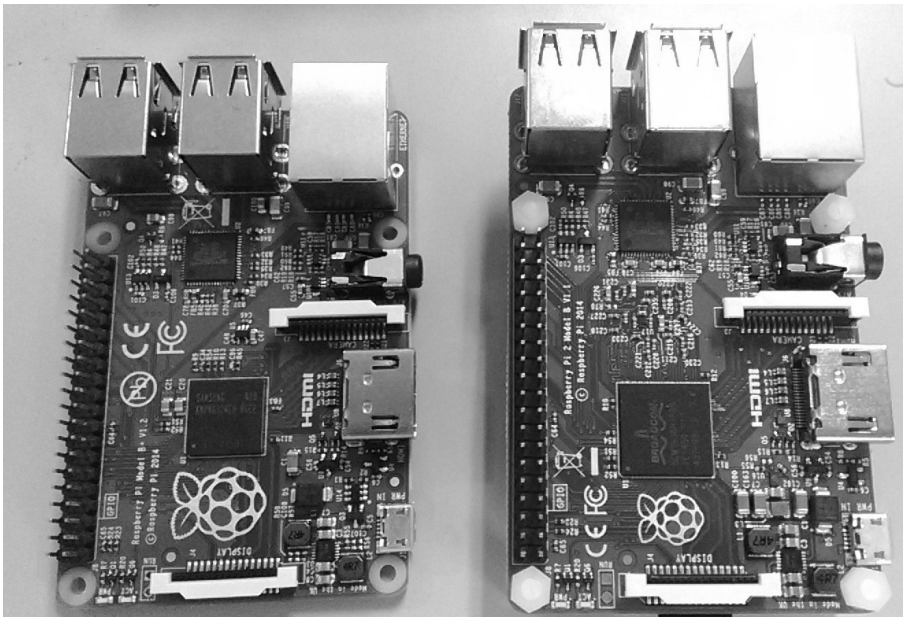


図 7 RPi Model B+（左） と RPi 2 Model B（右）

RPi (Model B+) と RPi 2 Model B のハードウェアは表 1 となります。CPU、コア数、メモリの種類とサイズが大きく異なる事が分かります。RPi2 ではコア数が増えているため、電源も大きめのものが必要になっていることも注目すべき点です。

表 1　RPi と RPi 2 B

-	RPi Model B+	RPi 2 Model B
CPU	ARM1176JZF-S 1 コア (700MHz) / ARMv6	ARM Cortex-A7 4 コア (900MHz) / ARMv7
SoC	Broadcom BCM2835	Broadcom BCM2836
CPU	Broadcom VideoCore IV (250MHz)	同左
メモリ	512MB (SDRAM)	1GB (LPDDR2 SDRAM)
ネットワーク	LAN9514 (10/100 Mbps)	同左
外部 I/O	GPIO 40 ピン	同左
ストレージ	microSD	同左
電源	600 mA (3.0W)	900 mA (4.5-5.5W)

表 2　Debian と Raspbian

-	Debian armel	Debian armhf	Raspbian
ターゲット命令セット	ARMv4	ARMv7	ARMv6
FPU	なし	VFPv3	VFPv2
Debian ネイティブ	Yes	Yes	No

表 3　UnixBench の結果

-	Debian armel/RPi	Debian armhf/RPi2	Raspbian/Rpi	Raspbian/Rpi2
System Benchmarks Index Score	66.5	450.8 (183.1)	80.1	442.9 (173.8)

Debian armel、armhf、Raspbian の違いは表 2 の通りです。各アーキテクチャでサポートする命令セットが異なり、Debian armel では RPi/RPi2 に最適化されているとは言えないことがわかります。Unixbench でベンチマークした結果を表 3 に示します。RPi では Raspbian のよい結果となり、RPi2 では Debian / armhf がよい結果となります。これは Raspbian が ARMv6 / VFPv2 に最適化されたバイナリで、RPi2 に最適化されていないためです。RPi2 では Raspbian より Debian / armhf を使ったほうが良いことがわかりました。

4.3　Debian armhf / Jessie のインストール方法

インストールには 実機、初期化されてもよい 4GB 以上の microSD カード、電源用の micro USB ケーブル等が必要です。また USB シリアル変換モジュールがあるとコンソールから操作できるので、カスタマイズが楽にできます。接続例を図 8 に示します。

またインストールの流れは以下となります。

1. microSD カードの認識確認
2. microSD カードの初期化
3. microSD カードにパーティション作成
4. microSD カードのフォーマット
5. cdebootstrap を使って microSD カードにインストール
6. RPi2 の Linux カーネルとカーネルモジュールのインストール
7. RPi2 のカーネルコマンドラインの設定
8. fstab の設定

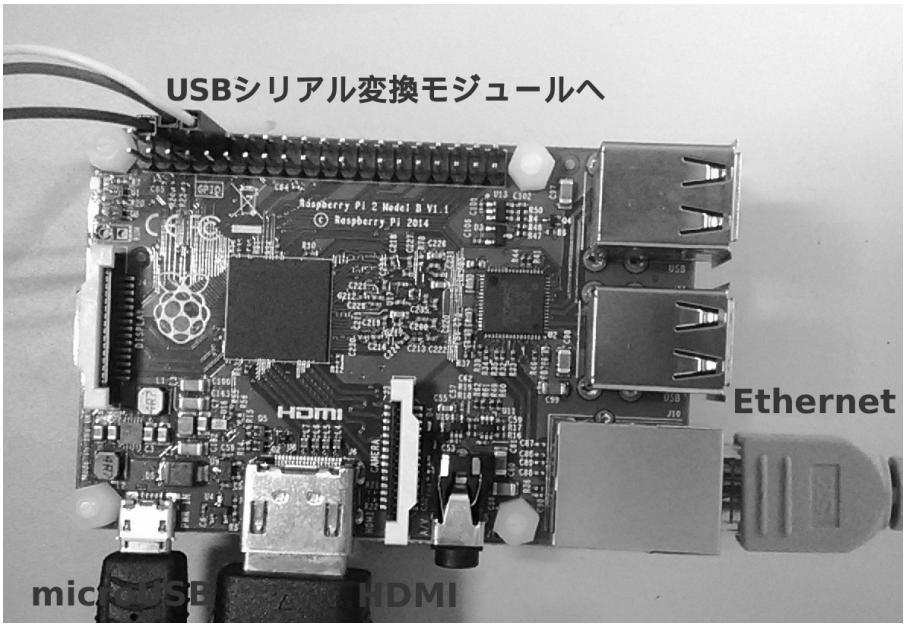


図 8　RPi2 接続例

9. ネットワークデバイスの設定
10. rootfs 用パーティションの変更
11. root のパスワードの設定と rpi ユーザの追加
12. microSD カードのアンマウントと RPi2 の起動
13. RPi2 へのログイン
14. RPi2 専用ツールのインストール

4.3.1　microSD カード の接続確認

使用している Debian に micorSD カードを挿入します。挿入すると dmesg に以下のようなメッセージが出力されるはずです。これで microSD がどのデバイスファイルに割り当てられたかわかります。図 9 では sde に割り当てられていることがわかります。

```
$ dmesg  | tail -5
[858983.896718] FAT-fs (sdf1): Directory bread(block 32775) failed
[858983.896729] FAT-fs (sdf1): Directory bread(block 1390704) failed
[858983.896731] FAT-fs (sdf1): Directory bread(block 1390705) failed
[869873.800361] sd 6:0:0:3: [sde] 15523840 512-byte logical blocks: (7.94 GB/7.40 GiB)
[869873.831121]  sde: sde1
```

図 9　microSD カードのデバイスファイル割り当て確認

4.3.2　microSD カードの初期化

購入したばかりの microSD カードは VFAT 等でフォーマットされています。MBR がある領域を 0 で埋めて初期化します（fdisk コマンド等で丁寧にやってもよいです）。

```
$ sudo dd if=/dev/zero of=/dev/sde bs=1M count=1
```

図 10　microSD カードの初期化

4.3.3 microSD カードにパーティション作成

fdisk コマンドを使って microSD カードにパーティションを作成します。図 11 に手順を示します。32MB、VFAT で boot 用のパーティションを作成し、残りを rootfs 用に Linux 用に作成します。

一気にやりたい方は図 12 を実行すればよいです。

```
$ sudo fdisk /dev/sde
Command (m for help): o

Created a new DOS disklabel with disk identifier 0x9aa4e1fa.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-15523839, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-15523839, default 15523839): +32M

Created a new partition 1 of type 'Linux' and of size 32 MiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): e
If you have created or modified any DOS 6.x partitions, please see the fdisk documentation for additional information.
Changed type of partition 'Linux' to 'W95 FAT16 (LBA)'.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (67584-15523839, default 67584):
Last sector, +sectors or +size{K,M,G,T,P} (67584-15523839, default 15523839):

Created a new partition 2 of type 'Linux' and of size 7.4 GiB.

Command (m for help): w
The partition figure has been altered.
Calling ioctl() to re-read partition figure.
Syncing disks.
```

図 11　microSD カードにパーティションを作成

図 12　microSD カードにパーティションを作成 一気にやるバージョン

```
(echo o; echo n; echo p; echo 1; echo ; echo +32M; echo t; echo e; echo n; echo p; echo 2; echo ; echo ; echo w) | fdisk /dev/sde
```

4.3.4 microSD カードのフォーマット

パーティション 1 は mkfs.msdos で、パーティション 2 は mkfs.ext でフォーマットします。フォーマットできたら適当なディレクトリを作成し、 2 つのパーティションをマウントします。今回は パーティション 1 用に /tmp/boot ディレクトリ、パーティション 2 用に/tmp/rootfs ディレクトリを作成し、マウントします。

```
$ sudo mkfs.msdos /dev/sde1
$ sudo mkfs.ext4 /dev/sde2
$ mkdir /tmp/boot /tmp/rootfs
$ sudo mount /dev/sde1 /tmp/boot
$ sudo mount /dev/sde2 /tmp/rootfs
```

図 13　microSD カードのフォーマットとマウント

4.3.5 cdebootstrap を使って microSD カードにインストールする

cbootstrap を使って、microSD カードに debian 起動イメージをインストールします。操作しているマシンが PC（i386 や amd64）の場合、通常は armhf のバイナリを実行できません。そのため PC 等で先にインストールに必要な Debian パッケージのダウンロードと展開を行い、実際のインストールは RPi2 で行うという方法を取ります。実際のインストール方法は図 14 となります。今回のインストールでは standard で指定されているパッケージの他、シリアルコンソールが使えない環境を考え openssh-server、時間設定のために ntp、証明書のために ca-certificates、エディタとして vim をインストールするようにしています。もし他に一緒にインストールしたいパッケージがある場合は −−**include** に続けて指定する事ができます。

```
$ sudo cdebootstrap --arch=armhf -f standard --foreign jessie \
  --include=openssh-server,ntp,ca-certificates,vim /tmp/rootfs
...
```

図 14　cdebootstrap を使った Debian イメージのインストール

4.3.6 RPi2 の Linux カーネルとカーネルモジュールのインストール

残念なことに RPi2 の Linux カーネルは Debian では提供されていません。その理由としてまだ完全にアップストリームでサポートされていない事と起動にファームウェアが必要ということが挙げられます。Debian で RPi2 の Linux カーネルを扱うには rpi-update というツールを使う必要があります。

図 4.3.6 では rpi-update を RPi2 の rootfs にダウンロードした後、実行権限を付加し、rpi-update を使ってカーネルとカーネルモジュールをインストールします。

```
$ sudo curl -o /tmp/rootfs/usr/bin/rpi-update https://raw.githubusercontent.com/Hexxeh/rpi-update/master/rpi-update
$ sudo chmod +x /tmp/rootfs/usr/bin/rpi-update
$ sudo mkdir /tmp/rootfs/lib/modules
$ sudo ROOT_PATH=/tmp/rootfs BOOT_PATH=/tmp/boot /tmp/rootfs/usr/bin/rpi-update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** Performing self-update
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8107  100  8107    0     0  54471      0 --:--:-- --:--:-- --:--:-- 54777
 *** Relaunching after update
...
```

図 15　rpi-update のインストールと Linux カーネルのインストール

4.3.7 RPi2 のカーネルコマンドラインの設定

RPi2 のカーネルコマンドラインを設定します。RPi は/boot/cmdline.txt に記載されているカーネルコマンドラインを読み込んで起動します。図 4.3.7 のように実行し、カーネルコマンドラインを設定します。

```
$ sudo sh -c "echo dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1
     root=/dev/mmcblk0p2 rootwait > /tmp/boot/cmdline.txt
```

図 16　カーネルコマンドラインの設定

4.3.8 fstab の設定

fstab の設定を行ないます。procfs、rootfs, boot ディレクトリの設定を記載します。

```
proc            /proc           proc    defaults        0       0
/dev/mmcblk0p1  /boot           vfat    defaults        0       2
/dev/mmcblk0p2  /               ext4    defaults,noatime        0       1
```

図 17　fstab の設定

4.3.9　ネットワークデバイスの設定

rootfs/etc/network/interfaces を編集し、ネットワークデバイスを有効にします。この操作は必須ではありませんが、USB シリアル変換モジュールを持ってない人は RPi2 に SSH でログインして操作する必要がありますので、ここでネットワークを起動時に有効するように設定しておきます。RPi2 の IP アドレスを DHCP から取得する場合は図 4.3.9 のように設定します。

```
auto eth0
iface eth0 inet dhcp
```

図 18　ネットワークデバイスの設定

4.3.10　rootfs 用パーティションの変更

rootfs/sbin/init に書かれている 2nd bootstrap の内容に rootfs をマウントする行があります。デフォルトでは rootfs を / にマウントするよう記述されているため、このままではインストールに失敗します。正しくインストールできるように rootfs を /dev/mmcblk0p2 に変更します。

```
trap 'error "Interruped!"' HUP INT TERM

mount -n -o remount,rw rootfs / <- これを
mount -n -o remount,rw /dev/mmcblk0p2 / <- これに変更

chown -hR 0:0 /
```

図 19　ネットワークデバイスの設定

4.3.11　root のパスワードの設定と rpi ユーザの追加

現状のままではインストール完了後にログインできません。2nd bootstrap 時に root のパスワードを設定する処理を追加します。また rpi ユーザを作成し、パスワードを設定する処理も追加します。rpi ユーザは説明のために使っているだけですので、他のユーザ名でも問題ありません。

```
echo 'deb http://ftp.debian.org/debian jessie main' > /etc/apt/sources.list

echo "root:root" | chpasswd <- この行を追加
useradd -m rpi <- この行を追加
echo rpi:rpi | chpasswd <- この行を追加

run rm /sbin/init
```

図 20　root パスワードの設定

4.3.12　microSD カードのアンマウントと RPi2 の起動

microSD カードをアンマウントし、PRi2 の microSD カードスロットに挿入します。挿入後、micro USB ケーブルを RPi2 に挿し、RPi2 を起動します。起動すると自動的に 2nd bootstrap が実行され、RPi2 上でインストール

が実行されます。もし HDMI 接続ができるモニターを持っている場合は、RPi2 と接続するとインストールされる様子を確認できます。インストール完了まで３０分ほど待たされるので、気長に待ちましょう。HDMI が利用できるモニターを持ってない場合、インストールが完了したか見た目ではわからないので、RPi2 に IP アドレスが割り当てられているか、ping を実行して反応があるかなどで確認する必要があります。

4.3.13　RPi2 へのログイン

インストールが完了すると自動的に init が再実行され、Debian が立ち上がった状態になっています。USB シリアルモジュール経由や、SSH 経由でログインできるようになっていますので、ログインしてください。後は通常の Debian と変わりません。

4.3.14　RPi2 専用ツールのインストール

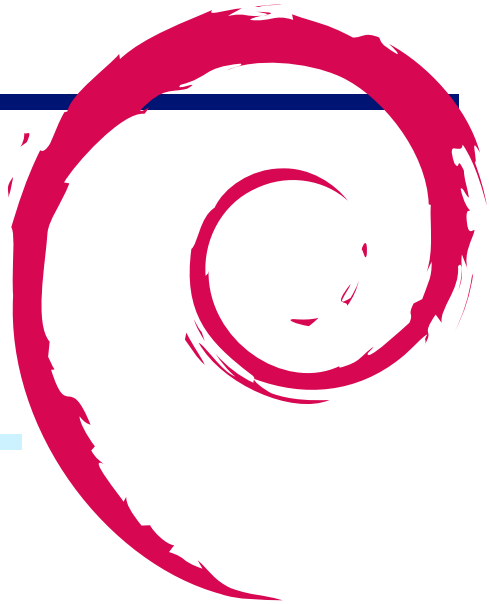
RPi2 の専用ツールである rpi-update、raspi-config はまだ Debian では提供されていません。これらはカーネルやファームウェアのアップデート、RPi ハードウェアの設定を行うための機能が搭載されており、RPi ユーザには必須のツールとなっています。これを Debian で利用できるようにするには raspberrypi.org で提供されている 各ツールの Debian パッケージをインストールする必要があります。図 4.3.14 に設定方法を示します。

```
# wget -O - http://archive.raspberrypi.org/debian/raspberrypi.gpg.key | apt-key add -
# echo deb http://archive.raspberrypi.org/debian wheezy main >> /etc/apt/sources.list
# apt-get update
# apt-get install rpi-update raspi-config
```

図 21　rpi-config、rpi-update パッケージのインストール方法

4.3.15　終わりに

RPi2 から ネイティブの Debian が利用できるようになりました。インストーラや microSD カードイメージが準備されていなくても、今回解説した方法を使うと RPi2 に自分好みの Debian をインストールできるようになります。CPU も強化されそこそこ使いやすくなった Rpi2 で Debian を触ってみてはいかがでしょうか。



5 自然言語処理チーム (pkg-nlp-ja) とパッケージ

野首 貴嗣

5.1 NOTICE

私は専門家ではありません。

5.2 pkg-nlp-ja

Debian における自然言語処理パッケージの管理 `https://alioth.debian.org/projects/pkg-nlp-ja/`

5.3 管理パッケージ

- ChaSen
  - dirts (Double Array 実装)
- 辞書
  - ipadic
  - naist-jdic
  - unidic

5.4 その他の NLP パッケージ

- 解析
  - MeCab
  - Juman
- 変換
  - Anthy (pkg-anthy)
  - KAKASI
  - mozc
    - 変換については pkg-ime で管理

5.5 何ができるか

形態素解析

- 形態素 (単語) を調べる
- 単語の属性情報を得る
  - 品詞・読み等
    - 辞書に依存

ChaSen, MeCab, Juman ref:`http://www.phontron.com/nlptools.php?lang=ja`

5.6 応用例

- 検索エンジン
  - 転地インデックスの作成
  - Groonga, Namazu 等
- 音声合成 (Text-to-Speech)
  - open-jtalk
- 分かち書き
  - word2vec の下処理
- かな漢字変換

5.7 動作原理

- 解析対象を確からしい単位で分割
  - 例: 「東京都府中市」
    - x 「東」「京都」「府」「中」「市」
    - o 「東京」「都」「府中」「市」
  - 単純なルールでは処理できない

5.8 解析の手がかり

- マッチする単語の長さ
  - 最長一致 (KAKASI)

  ref: KAKASI の実装と課題
- 品詞情報の利用
  - 前後に来やすい品詞のつながり
  - 単語のつながり
  - 単語の頻度/スコア
- 確率
  - 出現率、連接確率

5.9 アルゴリズム

- コスト最小法
  - ChaSen
- CRF (条件付き確率場)
  - MeCab

ref: 日本語形態素解析入門 (pdf)

5.10 辞書探索

Common Prefix Search

- 同じ接頭語を持つデータを高速に検索

アルゴリズム

- トライ
  - パトリシアトライ
- Double Array
  - darts パッケージ

5.11 トライ (Trie)

文字単位の木構造

```
漢-+->字-+->化
   |     |
   |     +->語
   +->音
```

ref: http://ja.wikipedia.org/wiki/基数木

5.12 Double Array

トライ構造を 2 つの配列で表現

- BASE 配列 (子ノード番号へのオフセット)
- CHECK 配列 (親ノード番号)
  - 単純な状態遷移表だと疎な配列になる
  - 構築に計算が必要
    - 配列の空き要素を見つける必要がある
    - 動的な更新には向かない

ref: ダブル配列の実装方法

5.13 辞書

- KAKASIDIC(kanwadic)
  - SKKDIC ベース
- ipadic
  - ChaSen, MeCab
- naist-jdic
  - ChaSen, MeCab
- jumandic
  - MeCab, JUMAN

5.14 単語の追加: KAKASI

辞書ファイルの作成 (EUC-JP)

```
# よみ [空白] 漢字
けいさんしょう 経産省
けいざいさんぎょうしょう 経済産業省
```

kakasi コマンドの引数に辞書のパスを追加

```
$ echo '日本の経産省 ' | kakasi -w -iutf8 -outf8 ./extdic
日本 の 経産省
$ echo '日本の経産省 ' | kakasi -w -iutf8 -outf8
日本 の 経 産 省
```

5.15 単語の追加: ChaSen

- ipadic のソースを取得
- 追加したい単語の意味に近いものを探す
- そのエントリーをコピーし、スコアはそのままで単語を修正する
- ./configure && make

注: スコアを計算するツールが付属していない

5.16 単語の追加: MeCab

1. ChaSen と同じ方法
   - 活用のある品詞はすべて開く必要がある
2. mecab-cost-train で正確なコストを算出
   - 元となるコーパスを用意する必要がある
   - cf: mecab-ipadic-neologd
   - MeCab の IPA 辞書を再学習させてみる
   - MeCab の辞書をカスタマイズする

5.17 参考

自然言語処理ツール

`http://www.phontron.com/nlptools.php?lang=ja`

日本語で読める自然言語処理のチュートリアルスライドまとめ

`http://blog.unnono.net/2015/04/nlp-tutorial.html`

6 .deb から Python パッケージへの変遷

まえだこうへい

6.1 はじめに

一昨年 1、昨年 2、現在の職場で構築および運用している Debian パッケージの自動ビルド&ローカルアーカイブの仕組みについてお話しました (以下ローカル Debian CI と呼びます)。昨年後半、部門での開発言語を Python [*44]にするという方針になり、同様の形でローカル PyPI を利用した Python パッケージ配布の仕組みを構築しました (以下 ローカル PyPI CI と呼びます)。今まで運用して分かったメリット/デメリットや、ローカル Debian CI を必要とするケースなどについて紹介します。そして私自身が主に Python 関連の Debian パッケージをメンテナンスしていますが、これに絡めた考察を行います。

6.2 ローカル Debian CI

ローカル Debian CI は、git-buildpackage で管理している Git リポジトリもしくはソースパッケージを取得してビルドを行います。前者はインハウスで開発したソフトウェア、後者はバックポートする場合です。git-buildpackage / cowbuilder でのビルド、lintian でのポリシーチェック、piuparts でのインストール/アンインストールテスト、debsign[*45]での署名、dput での reprepro へのアップロードを行います。少なくともバックポートに関しては Jenkins のプロジェクトのコピーだけで他のメンバーでも行えます。図 22 のような構成です。

6.3 ローカル PyPI CI

ローカル PyPI CI は同様に Jenkins を使用し、reprepro に相当するローカル PyPI サーバには devpi [*46] を使用しています。Jenkins のジョブスクリプトもやはり Python で実装しました。このスクリプト自体も Python パッケージ化し、ジョブ実行時にローカル PyPI からインストールして実行します。[*47]git-buildpackage/cowbuilder でのクリーンビルドに相当するクリーン環境でのテスト実行は Tox を使うことで実現しています。Tox から virtualenv 環境が作られ、Python2.7 / 3.4 でのユニットテスト[*48]、pylint、pychecker でのチェック、Sphinx ドキュメントのビルドのテストを行います。テストが成功すると devpi-client [*49] を使い、devpi-server にパッケージのアップロードを行います。

更に、ローカル PyPI CI には GitHub Enterprise からの Webhook を使ったジョブの起動と、HipChat へのテス

[*44] Web フレームワークには Django 及び django REST framework に統一しました。
[*45] 実際には TTY なしで debsign を実行できないため、Python で pydebsign を実装し、使用しています。3
[*46] `http://doc.devpi.net/latest/` PyConJP 2014 の「パッケージングの今」の資料 7 で知りました。
[*47] ローカル Debian CI 用のスクリプトは Git で管理し、ジョブごとに checkout する形式でした。
[*48] 現在の OS は Ubuntu Trusty のため、この 2 バージョンに固定。
[*49] `https://pypi.python.org/pypi/devpi-client`、`http://doc.devpi.net/latest/userman/devpi_packages.html`

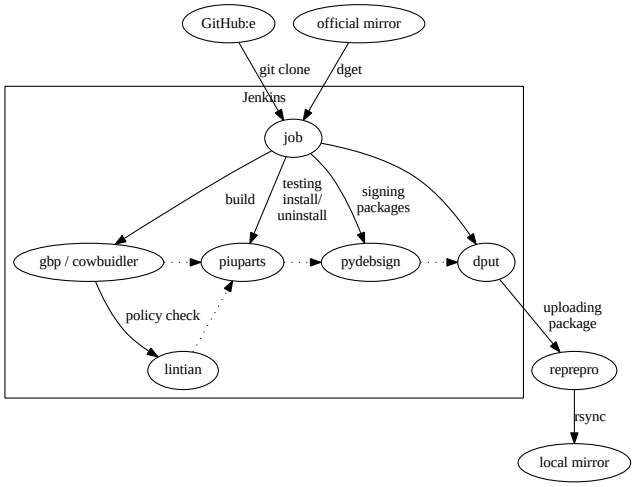


図 22　ローカル Debian CI

ト結果の通知の機能を実装しました。構成は図 23 のようになります。

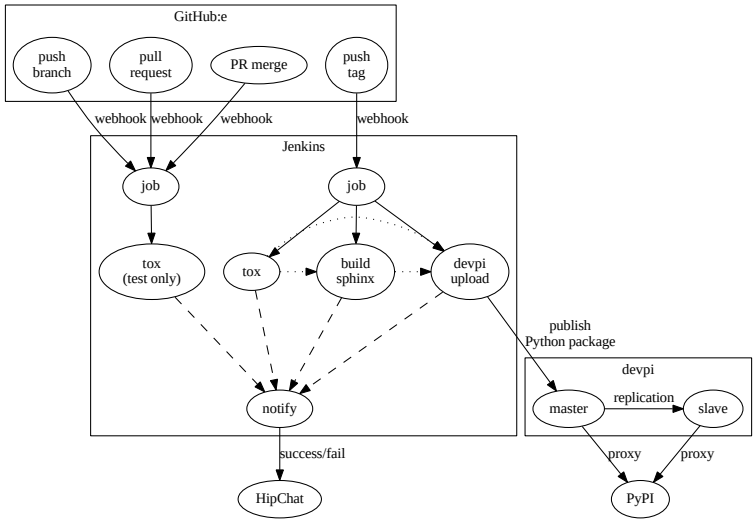


図 23　ローカル PyPI CI

Python での開発環境の標準化

チームメンバーのスキルレベルもまちまちであるため、品質の底上げのための次のような標準化も行いました。

- 開発標準化のための指針ドキュメント作成
- Python パッケージのテンプレート化
- Django アプリの Python パッケージのテンプレート化
- 独自認証システム用のモック機能付き共通ライブラリや Django モジュールの開発、テンプレートへの適用
- ローカル PyPI CI をローカルの開発環境で使うためのセットアップスクリプトの提供
- テンプレートおよびローカル PyPI CI の使い方のドキュメント作成、レクチャー
- HipChat での主要な変更点の案内、質疑応答

テンプレートには次のような機能を予め提供しています。

- Tox 経由での pytest、pytest-flake、pylint、pychecker の実行
- Sphinx automodule を使ったドキュメントの自動生成、ローカル PyPI へのドキュメントアップロード
- Django ベストプラクティスの適用 (Django プロジェクトのディレクトリ構成、環境ごとの settings.py の切替の仕組みなど) や、Django および django REST framework を含めたサンプルアプリの提供

6.4 before/after

6.4.1 パッケージ化を行えるメンバーの増加

ローカル Debian CI の構築以前に Debian パッケージの作成のレクチャーを行ったり、ローカル Debian CI の使い方のドキュメントの作成していました。しかし、敷居が高いこともありソースパッケージの作成は他のメンバーが行える状態ではありません。

一方、Python パッケージはテンプレートを使い、残りの setup.py の必要項目さえ埋めればパッケージ化できます。これにより Python は書けても Python パッケージは作ったことがないメンバーでも簡単にパッケージ化できるようになりました。

6.4.2 開発時におけるリポジトリの利用の簡易化

各オフィスビルからや各データセンターから[*50]でも各リポジトリはアクセスできるので、基本的には Debian パッケージでも Python パッケージでもその点は差がありません。しかし、パッケージ化の観点では前者は敷居が高いため、明示的にローカルの開発環境でも使う人は限られていました。[*51]後者はパッケージ化が簡易であることや、アップロードも Jenkins からだけでなく、LDAP アカウントで手元からアップロード可能であり[*52]、ローカル PyPI の URL を pip の “`--index-url`” オプションで指定することで切替可能なため、利用頻度及び明示的な利用ユーザが増加しました。

6.4.3 パッケージ化のコスト削減

Debian パッケージでは、基本私一人で行っており独自パッケージをつくるのに掛かる労力は結構かかります。特に依存パッケージが多く、かつそれらが公式パッケージ化されていない場合には大変です。Python パッケージの場合には、前述の通り用意したテンプレートを使えばパッケージ化でき、アップロードも簡単であるため、比べるまでもないでしょう。

ただし、Python パッケージ以外では Debian パッケージを作ることも当然あるので、この辺はパッケージを作成できるメンバーを増やす方法を検討する必要があります。[*53]

6.5 ローカル Debian CI が必要なケース

一方で Debian パッケージのカスタムビルドが必要なケースもまだあります。

6.5.1 公式パッケージに無い

まず、Python パッケージ以外で、公式パッケージにはないソフトウェアを Debian パッケージ化する場合です。これが一番多いケースです。理想的にはライセンス上の問題がなければ、公式パッケージ化を目指したいところですが、そこは業務との兼ね合いもあり、コストがかかり過ぎる場合は割りきって、ローカルアーカイブで管理しています。ライセンス上の問題であったり、依存関係が多すぎる場合です。

6.5.2 公式パッケージにあるがアップストリームのものを使う

これは社内的な利用実績を重きに置く場合です。他の人が該当システムの主担当でパッケージ化を要望されるとき、アップストリームの配布するバイナリを `apt-get` でインストール可能にするために Debian パッケージにしています。[*54]そうすることで外部のリポジトリを登録しなくても、ローカルアーカイブからインストールできるというメ

[*50] オフィスとデータセンター間は基本 VPN 接続ですが、一部のサービスは HTTPS でアクセスできます。これもその一つです。
[*51] VM のインストール用の preseed で設定しているため、知らずに使っている潜在的なユーザ及びサーバは非常に多いです。
[*52] devpi-ldap (`https://pypi.python.org/pypi/devpi-ldap`) というモジュールを利用。
[*53] 基本的には最初に公式パッケージ化にすることを検討しています。
[*54] Tomcat とか Oracle JDK とか。

リットもあります。[*55]

自分が主担当の場合には公式パッケージを使うか、Sid からのバックポートを行っています。

6.5.3 公式パッケージが古すぎて要件を満たさない場合

この場合はバックポートの場合を行います。Ubuntu の LTS や Debian の stable 公式パッケージが古く、backports に存在しないが、Sid や upstream の配布するソースパッケージがある場合、それを使ってバックポートし、ローカルアーカイブで管理しています。

6.5.4 Python パッケージでの特殊ケース

これは現時点ではないのですが、OpenStack の導入にあたり、Ansible で設定を行うよりも、初期のデフォルト設定を加えたカスタムビルドパッケージを使って、インストールとアップデートだけ Ansible で行った方が良いのではないか、という案です。

6.6 Python パッケージの公式 Debian パッケージ化について

最後に Python パッケージを公式 Debian パッケージにすること自体について考えてみます。

6.6.1 ユーザーの視点

Debian パッケージとして提供されていると、`apt-get` でインストールできるのでその点が最大のメリットです。ですので、次のようなものは Debian パッケージになっていると便利です。

- コマンドラインツール
- デーモン
- 上記のパッケージに依存されているライブラリなど (直接ユーザからは見えませんが)

6.6.2 Python な人の視点

Python の実行環境と標準ライブラリはシステムパッケージで良いのですが[*56]、コードを書くのに使うサードパーティライブラリ自体は pip でインストールできる方が便利です。以前は、Sid 上で必要なパッケージの全てを Debian パッケージ化し、Debian stable か Ubuntu LTS の本番環境にもそれをバックポートしていました。が、もはや黒歴史となりました。[*57]

Debian パッケージの Python を使う場合、virtualenv/pyvenv など venv 環境を作るツールも Debian パッケージを使う方が、環境毎にこれらのインストールをプロジェクトのリポジトリでケアしないですみます。Tox についても下記にある通り、Debian パッケージの python-tox を使う方が、依存するパッケージによって setup.py や tox.ini の記述を変更しないで済みます。

Tox は Debian パッケージのものを使う理由

Tox は `python setup.py test` に統合すれば、`setup()` の `install_requires` に指定したパッケージが `easy_install` でローカルにダウンロードされ実行されます。
しかし、Django のように `pip` しかサポートしないパッケージの場合、`setup.py test` に統合していても、まず `easy_install` でまずローカルにインストールされるのですが、Django と関連のパッケージのインストールで失敗します。そのまま再度実行すると、ローカルにインストールされたファイルがあるので、`setup.py test` からの `easy_install` はスキップされ、Tox の実行から始まります。ローカルで実行する場合には、これでも良いのですが、Jenkins などでジョブ毎にクリーン環境を作る場合、必ず失敗してしまいます。`setup.py test` に統合した Tox ではなく、`tox` コマンドを直接実行する場合、デフォルトでは `easy_install` ではなく `pip` が使われ、パッケージのインストールも toxinidir の下の testenv のディレクトリ毎にインストールされるのでこのような問題が発生しません。

[*55] Cassandra とか。

[*56] pyenv で実行環境そのものも自前で用意する、という人もいるかもしれませんが、私は違うので割愛。

[*57] 当時、ローカル PyPI の存在を知らなかったため、その時点で出来うる方法= Debian パッケージのローカルアーカイブを使いました。Python パッケージのファイルを HTTP サーバーに置いて公開するというのも管理が面倒ですし。

6.6.3 依存する Python パッケージに、Debian パッケージを使う場合

依存するパッケージが既に Debian パッケージ化されている場合は、`virtualenv` や `pyvenv` の`--system-site-packages` オプションを使えば、Debian パッケージでインストールされた Python パッケージを venv に使うこともできます。必要であれば、venv 環境内で個別のパッケージのアップデートも可能です。

6.6.4 開発した Web アプリを Debian パッケージにする場合

これは公式パッケージというよりは、カスタムビルドが主なケースの話になりますが、Web アプリ自体を Debian パッケージにする場合、`start-stop-daemon` などでデーモン化し、uWSGI 経由で Apache や Nginx などにリバースプロキシする設定を init スクリプトで用意やれば、ユーザとしてはパッケージのアップデートだけで基本済むので非常に楽です。基本開発が止まっているけど、ソフトウェアそのもの需要があるなら、パッケージにすると便利です。

しかし、Web アプリとして継続的に機能追加し、リリースし続ける場合、パッケージ化するコストが高いということもありますが、一度パッケージ化してしまった後、uscan、uupdate を駆使して Debian パッケージを自動アップデート及びビルドにしたとしても、コミットからリリースまでに手順が増えるので運用に則さないと思われます。良い方法があればぜひ教えてください。[*58]

6.7 Python パッケージを公式 Debian パッケージにすべきか

デーモンやコマンドラインツールのパッケージ化であれば、Python でないユーザにとってメリットがあるので、パッケージ&メンテナンスできるならやると良いでしょう。一方、ライブラリだけが目的なら、そもそも Python でないユーザには使われない上、Python な人も pip でインストールするほうが利便性が高いので基本やる必要がないと思います。一方、C binding の Python ライブラリの場合、そもそも PyPI に公開されていないこともあります。そういうケースでは Debian パッケージは必須になるでしょう。[*59]

Debian パッケージの作り方自体を学ぶのであれば、Python は比較的パッケージ化がしやすいのではないかと思います。dh-python パッケージに含まれる、`pybuild` コマンドのおかげで、以前よりも Golang のパッケージ化並に簡単になりました。**しかし、`Golang` の方が簡単です。これもあまりパッケージ化する意義が無い言語ですが…。**

ところで以前、PyPI に登録されている Python パッケージを全部自動で Debian パッケージ化する、という話もあった気がしますが、どこ行ったのでしょうか。

6.8 まとめ

基本、Debian パッケージ化するのをやめたら幸せになりました。が、部分的には Debian パッケージにすることで利便性があがるので、ケースバイケースでやると良いのではないでしょうか、というのが、私の中での結論になっています。ぜひ、他の方のお考えもお聞かせ頂ければと思います。

そういえば、Ruby でも同じ話が既にされていたと思うのですけど、結局どうなったのでしたでしょうか？

参考文献

[1] “とある Web 企業での Debian システムの使い方。“, `http://gum.debian.or.jp/2013/session/437.html`
[2] “Jenkins での Debian パッケージ自動化 “, `http://tokyodebian.alioth.debian.org/2014-07.html`
[3] “I made debsign of Python libary that can be run without a TTY“, `http://d.palmtb.net/2014/05/28/i_made_debsign_of_python_libary_that_can_be_run_without_a_tty.html`
[4] “Retrieve and generate debian package of Oracle JDK“, `http://d.palmtb.net/2014/04/16/retrieve_and_generate_debian_package_of_oracle_jdk.html`
[5] “How to build custom Debian package automatically by Jenkins“, `http://d.palmtb.net/2014/04/12/how_to_build_custom_debian_package_automatically_by_jenkins.html`
[6] “Issue deploying Jenkins to Tomcat7 Debian package in Wheezy“, `http://d.palmtb.net/2014/03/27/issue_deploying_jenkins_to_tomcat7_debian_package_in_wheezy.html`
[7] “パッケージングの今 “, `http://www.slideshare.net/aodag/ss-39068785`

[*58] 以前は、ビルド済み配布物の作成に `bdist_deb` がありましたが、今は無くなったようです。
[*59] 例えば、netsnmp-python などは Debian パッケージにしかありません。libvirt-python は PyPI にあります。

7 Emacs 関連パッケージの Debian パッケージ作成

henrich

7.1 はじめに

Debian で elisp パッケージをお作法に従って作る方法について語ってみます。
emacs 関連パッケージの Debian パッケージ化の手続きを簡単にいうと、

Step 1. emacsen-common パッケージを入れる
Step 2. /usr/share/doc/emacsen-common/debian-emacs-policy.gz を読む
Step 3. ポリシーに従って作る

以上となります。
…というのも突き放しすぎなので、step by step で作り方を説明してみます。

7.2 step by step での作り方

まず、ソースを展開しておいて、dh-make パッケージを入れて dh_make コマンドでパッケージを作ります。パッケージは dh-make、コマンドは dh_make と「-」と「_」の違いがあるのに注意。この際、*.el なファイルがあるとテンプレートファイルをコピーしてパッケージ名に合わせて修正してくれます（あるいは –with-emacs オプションを使います）。以前は特に.el なファイルがなくてもテンプレートファイルがコピーされていたのですが、毎度消すのが鬱陶しかったのでパッチを送って修正してもらいました (*60)

```
$ sudo apt-get install dh-make
$ dh_make --createorig

Type of package: single binary, indep binary, multiple binary, library, kernel module, kernel patch?
 [s/i/m/l/k/n] s

Maintainer name  : Hideki Yamane
Email-Address    : henrich@XXXXXX
Date             : Sun, 04 Jan 2015 12:26:09 +0900
Package Name     : ag-el
Version          : 0.44
License          : blank
Type of Package  : Single
Hit <enter> to confirm:
Currently there is no top level Makefile. This may require additional tuning.
Done. Please edit the files in the debian/ subdirectory now. You should also
check that the ag.el Makefiles install into $DESTDIR and not in / .
```

*60 http://bugs.debian.org/696793

```
$ ls debian/
README.Debian      changelog  emacsen-install.ex  manpage.sgml.ex  preinst.ex
README.source      compat     emacsen-remove.ex   manpage.xml.ex   prerm.ex
ag.el.cron.d.ex    control    emacsen-startup.ex  menu.ex          rules
ag.el.default.ex   copyright  init.d.ex           postinst.ex      source
ag.el.doc-base.EX  docs       manpage.1.ex        postrm.ex        watch.ex
```

emacsen-*.ex ファイルがあることがわかります。これをリネームして.ex ポストフィックスを外してやりましょう。余分なファイルも消しておきます。

```
$ ls
changelog  control    docs             emacsen-remove   rules   watch
compat     copyright  emacsen-install  emacsen-startup  source
```

スッキリしました。emacsen* ファイルの中身は特にいじる必要はありません。

次に debian/control を修正します。

```
$ vi debian/control
------debian/control ファイルここから---------------
Package: ag-el
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: <insert up to 60 chars description>
 <insert long description, indented with spaces>

Package: ag-el
Architecture: all
Depends: ${shlibs:Depends}, ${misc:Depends}, emacsen-common (>= 2.0.8), emacs,
         silversearcher-ag,
Description: Emacs frontend to ag
 The Silver Searcher (a.k.a. ag) is very fast grep-like program.
 It is faster and has an attractive features than grep.
 ag.el is simple ag frontend for Emacs, loosely based on ack-and-half.el.
------debian/control ファイルここまで---------------
```

Architecture: は elisp スクリプトなので all に変えます。ポイントは Depends 行の emacsen-common (>= 2.0.8), emacs の 2 つです。

emacsen-common (>= 2.0.8) の必要性は Emacs Policy に書いてあります。これと併せて debian/emacs-compat ファイルを置きます（中身は単に「0」とだけする）。このファイルは dh_installemacsen が処理してくれます。man dh_installemacsen とすると記載があります。

```
$ man dh_installemacsen
... 中略...
FILES
       debian/package.emacsen-compat
            Installed into usr/lib/emacsen-common/packages/compat/package in the package build directory.
... 中略...
```

emacs はメタパッケージになっており、Emacs24 などの特定バージョンに依存しています。Depends: emacs24 などとすると、Emacs25 が出た時に再度 Depends 行を書き換える必要が発生しますが、このメタパッケージを指定すればパッケージ側での都度変更は不要です。

```
$ apt-cache show emacs
Package: emacs
Source: emacs-defaults
Version: 46.1
Installed-Size: 25
Maintainer: Rob Browning <rlb@defaultvalue.org>
Architecture: all
Depends: emacs24 | emacs24-lucid | emacs24-nox
Description-ja: GNU Emacs エディタ (メタパッケージ)
 GNU Emacs は、拡張可能で自己説明的なテキストエディタです。
 これは常に推奨の最新 Emacs リリースに依存するメタパッケージです。
Description-md5: 21fb7da111336097a2378959f6d6e6a8
Tag: devel::editor, role::dummy, role::metapackage, suite::emacs, suite::gnu,
 use::editing
Section: editors
Priority: optional
Filename: pool/main/e/emacs-defaults/emacs_46.1_all.deb
Size: 1634
MD5sum: 1f115942065ac452467e02377368ee22
SHA1: dbb1343a3d24f60e5038994e3528dd7486e40943
SHA256: c1fad54e790d69b83f32f2612963baba3ea8091ff3ca72c960c7312096223e3a
```

その他にも Xemacs を含む emacsen というパッケージでの指定もありますが、これは仮想パッケージなので単独

で指定はできません。指定するとしたら"Depends: emacs | emacsen" のようにします。

…とここまで書いて「dh-make でフォローしてくれればいいんじゃねーの？」と気づきました。では、dh-make パッケージへのパッチを作りましょう…できましたので、BTS を…しました。将来的にはこの部分は「へーそんなのもあるんだー」的になるはず。あ、reportbug で複数ファイルを添付するには 1 つずつ-A で指定が必要なようです。

```
$ debcheckout dh-make
$ cd dh-make
$ git checkout -b support-modern-emacs-policy
... いろいろ変更...
$ git format-patch master
$ ls
0001-add-emacsen-compat-for-modern-Emacs-lisp-package.patch  0002-add-debian-control-file-for-Emacs-add-on.patch
debian  dh_make  dh_make.1  lib
$ reportbug -A 0001-add-emacsen-compat-for-modern-Emacs-lisp-package.patch \
-A 0002-add-debian-control-file-for-Emacs-add-on.patch dh-make
```

これで Bug#774545 として登録されました。

しかし、ここで気になることが。dh-make での debian/control ファイルはどうも種類に合わせてコピーされるようです。これをそのまま適用すると、*.el ファイルがある場合に否応なしに Emacs add-on パッケージ用の control ファイルになるわけで、複数のバイナリパッケージを生成するような場合には嬉しくありません。

多分、大本の control ファイルがあって、Emacs 用のが追加される形のほうが嬉しい…のでは？という気がします。…しかし、その場合の「Section: lisp」指定をどうするのか…

```
Source: #PACKAGE#
Section: lisp
Priority: optional
Maintainer: #USERNAME# <#EMAIL#>
Build-Depends: #BUILD_DEPS#
Standards-Version: #POLICY#
Homepage: <insert the upstream URL, if relevant>
#Vcs-Git: git://anonscm.debian.org/collab-maint/#PACKAGE#.git
#Vcs-Browser: http://anonscm.debian.org/cgit/collab-maint/#PACKAGE#.git/

Package: #PACKAGE#
Architecture: all
Depends: ${misc:Depends}, emacsen-common (>= 2.0.8), emacs | emacsen,
Description: <insert up to 60 chars description>
 <insert long description, indented with spaces>
```

これ、パッケージのところだけを「追記」する形だといいのですかね？

```
Package: #PACKAGE#
Section: lisp
Architecture: all
Depends: ${misc:Depends}, emacsen-common (>= 2.0.8), emacs | emacsen,
Description: <insert up to 60 chars description>
 <insert long description, indented with spaces>
```

そうすればソースパッケージの雛形も 1 箇所でまとめれば済んで、いい感じになるんじゃないか？とか思い始めました。

```
Source: #PACKAGE#
Section: UNKNOWN
Priority: optional
Maintainer: #USERNAME# <#EMAIL#>
Build-Depends: #BUILD_DEPS#
Standards-Version: #POLICY#
Homepage: <insert the upstream URL, if relevant>
#Vcs-Git: git://anonscm.debian.org/collab-maint/#PACKAGE#.git
#Vcs-Browser: http://anonscm.debian.org/cgit/collab-maint/#PACKAGE#.git/
```

でもそれは別ハックですね…。

あとは debuild して lintian の warning などをちょこちょこと直せば完成です。

```
$ dpkg --contents /var/cache/pbuilder/result/ag-el_0.44-1_all.deb
drwxr-xr-x root/root         0 2015-01-04 13:35 ./
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/share/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/share/doc/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/share/doc/ag-el/
-rw-r--r-- root/root       175 2015-01-04 13:10 ./usr/share/doc/ag-el/changelog.Debian.gz
-rw-r--r-- root/root      1089 2015-01-04 13:35 ./usr/share/doc/ag-el/copyright
-rw-r--r-- root/root      5279 2014-08-05 05:58 ./usr/share/doc/ag-el/README.md.gz
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/emacsen-common/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/remove/
-rwxr-xr-x root/root       465 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/remove/ag-el
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/install/
-rwxr-xr-x root/root      1280 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/install/ag-el
drwxr-xr-x root/root         0 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/compat/
-rw-r--r-- root/root         2 2015-01-04 13:35 ./usr/lib/emacsen-common/packages/compat/ag-el
drwxr-xr-x root/root         0 2015-01-04 13:35 ./etc/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./etc/emacs/
drwxr-xr-x root/root         0 2015-01-04 13:35 ./etc/emacs/site-start.d/
-rw-r--r-- root/root      1225 2015-01-04 13:35 ./etc/emacs/site-start.d/50ag-el.el
```

7.3 まとめ

- elisp パッケージは emacsen-common パッケージを入れて /usr/share/doc/emacsen-common/debian-emacs-policy.gz ファイルに従ってパッケージを作ろう
- お手軽にするには dh-make パッケージを使おう
- Policy を読んで tips を確認する手間を省くには dh-make パッケージをハックしてしまえばいい

8 Debian GNU/kFreeBSD における Jail 構築を試してみた

杉本 典充

8.1 はじめに

Debian GNU/kFreeBSD は、FreeBSD カーネルで動作する Debian です。近頃は docker を実装例としたコンテナ環境に復権（?!）の兆しがあるようにも思えます。そこで、Debian GNU/kFreeBSD をホストとして、FreeBSD Jail 環境の構築を試してみます。

8.2 Debian Ports と Debian GNU/kFreeBSD

Debian Ports [*61]とは、さまざまな CPU やカーネルで動作するように移植を行うプロジェクトのことです。

FreeBSD カーネルで起動する Debian を作るプロジェクトがあり、その Debian のことを「Debian GNU/kFreeBSD」と呼んでいます (k は kernel のこと)。現在では Intel CPU のアーキテクチャのみあります (kfreebsd-amd64、kfreebsd-i386)。Debian 6 (Squeeze) ではテクノロジープレビューとして初めて stable リリースされました。Debian 7 (Wheezy) でも継続して stable リリースされたのですが、Debian 8 (Jessie) では Drop されることが決まっています。[*62]

8.3 Debian におけるコンテナ環境の構築方法

Debian において、コンテナ環境を構築するには debootstrap を使います。通常はホスト環境と同じアーキテクチャを引数に指定して実行します。[*63]

```
# apt-get install debootstrap
# mkdir -p /srv/jail
# cd /srv/jail
# debootstrap --arch=[arch] [release] [dir] http://ftp.jp.debian.org/debian
```

ホスト環境のアーキテクチャと異なるアーキテクチャのコンテナ環境を用意すると普通は動作しません。ところが、Debian ではコンテナ環境と qemu を組み合わせることでエミュレーション動作させつつ、見た目はコンテナ環境でバイナリを実行させる方法があります。これを CrossDebootstrap と言います。[*64]

[*61] https://www.debian.org/ports/

[*62] https://lists.debian.org/debian-devel-announce/2014/09/msg00002.html において、stable を維持しつつ sid の開発を進めるにはそれなりに人手が必要であるが、kFreeBSD を作業する人手は不足していることが指摘されています。

[*63] debootstrap については昔の記事にまとめたものがあります。「debootstrap を有効活用してみよう」http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2013-natsu.pdf

[*64] https://lists.debian.org/debian-mips/2011/02/msg00031.html 2011 年頃に kfreebsd-mips 対応をしていた動きがあるようで、これが動くようになれば kFreeBSD で CrossDebootstrap を使える日がくるかもしれません。

```
# apt-get install qemu binfmt-support qemu-user-static
# debootstrap --foreign --arch=[arch] [release] [dir] http://ftp.jp.debian.org/debian
# cp /usr/bin/qemu-[arch]-static /[dir]/usr/bin/
# chroot [dir]
# /debootstrap/debootstrap --second-stage
```

8.4 Debian GNU/kFreeBSD をホストとした Jail 環境の構築

Debian GNU/kFreeBSD 上で Jail 環境を構築してみました。環境は以下で行いました。

- ホスト環境
  - Debian GNU/kFreeBSD unstable amd64
  - FreeBSD-10.1 kernel (10.1 svn274115-2)
- ネットワーク環境
  - ホストとコンテナが同一ネットワーク (192.168.0.0/24)
- コンテナ環境
  - FreeBSD Jail にて起動する

jail コマンドは apt でインストールできます。

```
# apt-get install freebsd-utils
# which jail
/usr/sbin/jail
```

ただし現状では、jls[*65]、jexec[*66]コマンドはパッケージ化されていません。そのため、今回は Jail 環境で sshd を動かし、ssh ログインできるようにすることをゴールとします。

Jail 環境を作成して起動するには以下の作業を行います。

- Jail 環境の動作に必要なバイナリや設定ファイル等のディレクトリツリーを配置する
- ネットワークデバイスに IP Alias 設定を行う
- Jail 環境内の動作に必要な/dev やファイルシステムの mount を行う
- jail コマンドを実行して Jail 環境を起動する

8.4.1 kfreebsd-amd64、kfreebsd-i386 の Jail 環境

ホスト環境と同じアーキテクチャの Jail 環境を作成します。以下の手順は kfreebsd-amd64 を指定していますが、kfreebsd-i386 も同様の手順で構築することができます。

```
# mkdir -p /srv/jail
# cd /srv/jail
# debootstrap --arch=kfreebsd-amd64 wheezy jail_kf64_1 http://ftp.jp.debian.org/debian
```

Jail 環境内の設定ファイルを変更します。

```
# cd /srv/jail
# vi jail_kf64_1/etc/resolv.conf
nameserver 192.168.0.1

# vi jail_kf64_1/etc/hostname
jail_kf64_1
```

Jail 環境内で sshd が動くように設定します。

[*65] jls は起動している jail を一覧表示するコマンドです。Jail 環境として動作しているコンテナは、kernel によって jid という番号を付与され識別されます。

[*66] jexec は Jail 環境内でコマンドを実行するコマンドです。実行例は 「# jexec 1 /bin/sh」

```
# chroot jail_kf64_1
# apt-get install openssh-server
# passwd
# adduser user
# passwd user
```

Jail 環境を起動するには debootstrap したディレクトリを jail コマンドの引数に指定します。Debian の web サイトに Jail 環境を起動する前処理を含めたスクリプトの例があります[*67]。複数の Jail 環境を構築するため、スクリプトと起動パラメータを記述した設定ファイルに分割するよう修正しました。

設定ファイルは以下です。

```
$ cat jail_kf64_1
#!/bin/sh
JID=1
JID_HOST=jail_kf64_1
JID_IPADDR=192.168.0.61
JID_NETMASK=255.255.255.0
JID_SH_PARAM=''-- -c''
JID_CMD=''/etc/init.d/rc S && /etc/init.d/rc 2''
```

起動スクリプトは以下です。

```
$ cat jail-bootstrap.sh
#!/bin/sh

. ./$1

echo ‘‘----’’
echo ‘‘start prison.’’
echo ‘‘[target] JID: $JID, name: $JID_HOST’’
echo ‘‘----’’
echo

NIC_DEV=em0

# add ip alias
/sbin/ifconfig $NIC_DEV alias $JID_IPADDR netmask $JID_NETMASK


# Linux-like /proc and /sys filesystems
mount -t linprocfs linprocfs /srv/jail/$JID_HOST/proc
mount -t linsysfs linsysfs /srv/jail/$JID_HOST/sys

# Ramdisk required for /run
mount -t tmpfs tmpfs /srv/jail/$JID_HOST/run

# A read-only /dev filesystem with restricted set of devices
mount -t devfs devfs /srv/jail/$JID_HOST/dev
# :XXX: ruleset 4 must be initialised as explained earlier in this Wiki page
devfs -m /srv/jail/$JID_HOST/dev rule -s 4 applyset
# Ensure the jail has some essential devices
for DEVICE in null random urandom zero
do if [ ! -e /srv/jail/$JID_HOST/dev/$DEVICE ]
    then
        echo ‘‘error: device ’/dev/$DEVICE’ MUST be available in the jail’’
        exit 1
    fi
done

# Ensure the jail has only a limited set of devices
#for DEVICE in mem kmem
#do if [ -e /srv/jail/$JID_HOST/dev/$DEVICE ]
#    then
#        echo ‘‘error: device ’/dev/$DEVICE’ MUST NOT be available in the jail’’
#        exit 1
#    fi
#done

# Compatibility symlink from /dev/shm to /run/shm
ln -s /run/shm /srv/jail/$JID_HOST/dev/

# Optionally enable networking
HOSTNAME=$JID_HOST
# :XXX: this IP address *must* be assigned to one of the host’s interfaces before the guest can use it
IP=$JID_IPADDR

mkdir -p /var/run/jail
jail -J /var/run/jail/$JID.jid -c jid=$JID \
  name=jail$JID \
  path=/srv/jail/$JID_HOST \
  host.hostname=$JID_HOST \
  ip4.addr=$IP \
  command=/bin/sh $JID_SH_PARAM ‘‘$JID_CMD’’
```

[*67] https://wiki.debian.org/Debian_GNU/kFreeBSD/Jails

スクリプトを実行して Jail 環境を起動します。

```
$ sudo ./jail-bootstrap.sh jail_kf64_jail

----
start prison.
[target] JID: 1, name: jail_kf64_1
----

devfs rule: ioctl DEVFSIO_SAPPLY: No such process
[info] Using makefile-style concurrent boot in runlevel S.
hostname: the specified hostname is invalid
kern.module_path: /lib/modules/10.1-0-amd64
sysctl: kern.module_path=/lib/modules/10.1-0-amd64: Operation not permitted
[....] Loading devfs rules...devfs ruleset: ioctl DEVFSIO_SUSE: Operation not permitted
 (warning).
mount: tmpfs: Operation not permitted
mount: tmpfs: Operation not permitted
[ ok ] Activating swap...done.
mount: /: unknown special file or file system
mount: /run: unknown special file or file system
mount: /proc: unknown special file or file system
mount: /sys: unknown special file or file system
[ ok ] Activating lvm and md swap...done.
[....] Checking file systems...fsck from util-linux 2.25.2
done.
[ ok ] Cleaning up temporary files... /tmp.
[ ok ] Mounting local filesystems...done.
/etc/init.d/mountall.sh: 59: kill: No such process

[ ok ] Activating swapfile swap...done.
mount: tmpfs: Operation not permitted
mount: tmpfs: Operation not permitted
[ ok ] Cleaning up temporary files....
[....] Starting device state change daemon : devddevd: Can't open devctl device /dev/devctl: Device or resource busy
 failed!
[....] Configuring network interfaces...ifconfig: up: permission denied
ifconfig: socket(family 28,SOCK_DGRAM: Protocol not supported
done.
[ ok ] Cleaning up temporary files....
[FAIL] startpar: service(s) returned failure: hostname.sh kldutils ... failed!
[info] Using makefile-style concurrent boot in runlevel 2.
[....] Starting enhanced syslogd: rsyslogdln: failed to create symbolic link '/dev/xconsole': Operation not permitted
. ok
[ ok ] Starting periodic command scheduler: cron.
[ ok ] Starting OpenBSD Secure Shell server: sshd.
```

Jail 環境へ ssh ログインします。

```
$ ssh user@192.168.0.61
user@192.168.0.61's password:

user@jail_kf64_1:~$
```

Jail 環境内のプロセスをすべて停止したい場合は Jail 環境内で以下コマンドを実行します。

```
root@jail_kf64_1:~# sh /etc/init.d/rc 0
```

8.4.2 FreeBSD-10.1-RELEASE の Jail 環境

FreeBSD のディレクトリツリーを取得して展開します。

```
# mkdir -p /srv/jail/jail_fb101_1
# cd /srv/jail/jail_fb101_1
# wget http://ftp.jaist.ac.jp/pub/FreeBSD/releases/amd64/amd64/10.1-RELEASE/base.txz
# tar xvf base.txz
# ls
COPYRIGHT  bin dev  lib      media  proc    root  sys  usr
base.txz   boot  etc  libexec  mnt    rescue  sbin  tmp  var
```

Jail 環境内の設定ファイルを変更します。

```
# vi etc/resolv.conf
nameserver 192.168.0.1
```

Jail 環境内で sshd が動くように設定します。

```
# vi etc/rc.conf
sshd_enable=''YES''
sendmail_enable=''NONE''

# cd ..
# chroot jail_fb101
# passwd
# adduser user
# passwd user
```

FreeBSD-10.1-RELEASE の Jail 環境を起動するスクリプトの設定ファイルは以下です。

```
$ cat jail_fb101_1
JID=3
JID_HOST=jail_fb101_1
JID_IPADDR=192.168.0.63
JID_NETMASK=255.255.255.0
JID_SH_PARAM=''''
JID_CMD=''/etc/rc''
```

Jail 環境を起動します。

```
$ sudo ./jail-bootstrap.sh jail_fb101_1

----
start prison.
[target] JID: 3, name: jail_fb101_1
----

mount: /srv/jail/jail_fb101_1/usr/src/sys: No such file or directory
mount: /srv/jail/jail_fb101_1/run: No such file or directory
devfs rule: ioctl DEVFSIO_SAPPLY: No such process
/etc/rc: WARNING: $hostname is not set -- see rc.conf(5).
Creating and/or trimming log files.
ln: /dev/log: Operation not permitted
Starting syslogd.
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib
32-bit compatibility ldconfig path: /usr/lib32
Clearing /tmp (X related).
Updating motd:.
Performing sanity check on sshd configuration.
Starting sshd.
Starting cron.

Mon Feb 16 22:43:36 UTC 2015
```

Jail 環境へ ssh ログインします。

```
$ ssh user@192.168.0.63
Password for user@jail_fb101_1:

user@jail_fb101_1:~ %
```

Jail 環境内のプロセスをすべて停止したい場合は Jail 環境内で以下コマンドを実行します。[*68]

```
root@jail_fb101_1:/ # sh /etc/rc.shutdown
```

8.4.3 linux-i386 の Jail 環境

【注意】debian としては動きませんでしたので、試したところまで述べます。

FreeBSD kernel には Linux バイナリ互換機能というものがあり、32bit の Linux バイナリを動かすことができます。FreeBSD において Linux バイナリ互換機能を有効にするには、linux_base Ports をインストールし、linux.ko を kldload します。Debian GNU/kFreeBSD では kernel に static link していますので linux.ko のロードは不要でした。

Linux バイナリ互換機能を動かす場合は、ホスト環境の sysctl 設定で linux kernel のバージョンを指定する必要があります。今回は linux-2.6.32 を採用している squeeze にしてみました。[*69] [*70]

[*68] `https://www.freebsd.org/doc/en/books/handbook/jails-build.html` にそう記述がありプロセスはあらかた停止しているように見えますが、kernel ではまだ jid が生きているようでゴミが残ってしまうようです。

[*69] バージョンが異なった状態で chroot してコンテナ内のバイナリを実行すると「FATAL: kernel too old」というエラーが発生します。（新しくても too old のようです）

[*70] FreeBSD Ports において「linux_base-c6」が実験的に提供されており、機能は CentOS-6 の linux kernel 相当です。CentOS-6 の linux kernel は 2.6.32 を採用しており、squeeze と同じです。

```
# vi /etc/sysctl.conf
compat.linux.osrelease=2.6.32

# sysctl -p /etc/sysctl.conf
compat.linux.osrelease: 2.6.16 -> 2.6.32
```

```
# mkdir -p /srv/jail
# cd /srv/jail
# debootstrap --foreign --arch=i386 squeeze jail_linux32_1 http://ftp.jp.debian.org/debian
```

chroot でコンテナ環境に入ってみます。コンテナ環境の中でシェルは動きますので Linux バイナリ互換機能は動いているようです。

```
# chroot jail_linux32_1
I have no name!@test-kf2:/#
```

しかし、debootstrap の second-stage でエラーになりました。

```
I have no name!@test-kf2:/# /debootstrap/debootstrap --second-stage
/debootstrap/debootstrap: 1303: /debootstrap/debootstrap: cannot create //test-dev-null: Operation not supported
E: Cannot install into target '/' mounted with noexec or nodev
```

すんなり動いてはくれない状況です。mount 絡みのエラーですので、mount できるようにコンテナの中を設定するか、debootstrap のスクリプトを修正するなど対処すると動くのかもしれません。

今回試せたのはここまででした。

8.5 おわりに

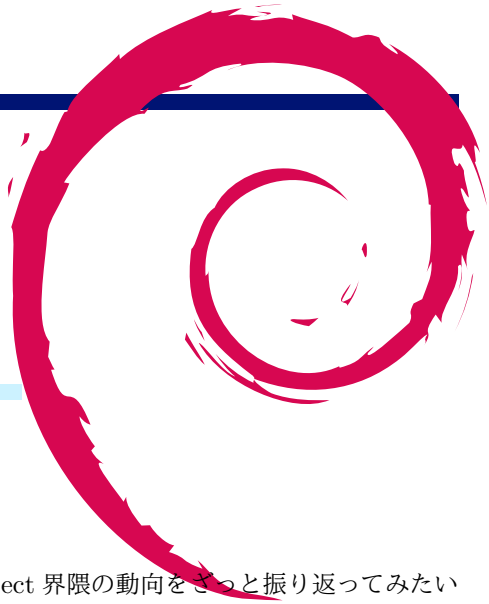
Debian GNU/kFreeBSD で FreeBSD Jail 環境の構築を試してみました。Debian GNU/kFreeBSD on FreeBSD の構成でも動作することは確認していますので試してみるのもおもしろいです。[*71]

本家 FreeBSD の Jail のように便利に利用するツールのパッケージが足りていない状況ですので、Debian GNU/kFreeBSD の開発を進めるとどんどん使いやすくなると思います。

8.6 参考資料

- Debian/kFreeBSD で jail を使う `http://hachulog.blogspot.jp/2011/11/debiankfreebsdjail.html`
- Debian_GNU kFreeBSD Jails `https://wiki.debian.org/Debian_GNU/kFreeBSD/Jails`
- あんどきゅめんてっど でびあん 2013 年夏号 「debootstrap を有効活用してみよう」 杉本典充 `http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume2013-natsu.pdf`

[*71] FreeBSD Ports に/usr/ports/sysutils/debootstrap があります。



9　DPN で振り返る 2014

かわだ てつたろう

2014 年に発行された DPN(Debian Project News[*72]) から Debian Project 界隈の動向をざっと振り返ってみたいと思います。

2014 年は 1 月 6 日号から 12 月 1 日号まで計 16 通発行されました。

9.1　1st quarter

1 月 31 日、IA64 アーキテクチャが Jessie から取り除かれました。

2 月 11 日、CTTE は Jessie の Linux アーキテクチャの init システムを systemd とする決定を行ないました。

3 月 19 日、Jessie のインストーラのアルファ1 をリリースされました。

9.2　2nd quarter

4 月 17 日、DPL 選挙の結果再選された Lucas Nussbaum さんの DPL 体制がスタートしました。

4 月 28 日、Code of Conduct が批准されました。これは先の CTTE の systemd の決定を受けて debian-devel メーリングリストが荒れに荒れたことに端を発しています。

4 月 26 日、SPARC アーキテクチャが Jessie から取り除かれました。

5 月 22 日、Debian GNU/Hurd の活動報告があり、80% のパッケージをカバーし SysVinit が動作するとのことです。

5 月 31 日、旧 stable の Squeeze はサポート終了となりました。しかし、新な試みとして限定ながら 5 年の長期サポートを目指して「squeeze-lts」が始まりました。

6 月 3 日、MATE1.8 の全パッケージングが完了したとのアナウンスがありました。

6 月 18 日、libc が eglibc から glibc に戻りました。

9.3　3rd quarter

8 月 16 日、Debian Day を迎え Debian は 21 歳になりました。

8 月 22 日から 8 月 31 日にかけて DebConf14 がアメリカのオレゴン州ポートランドで開催されました。

9 月 4 日、Cinnamon の全パッケージが Jessie に入ったと報告がありました。

9 月 19 日、再評価の結果 Jessie のデフォルトデスクトップ環境が GNOME に決まりました。

[*72] https://www.debian.org/News/weekly/

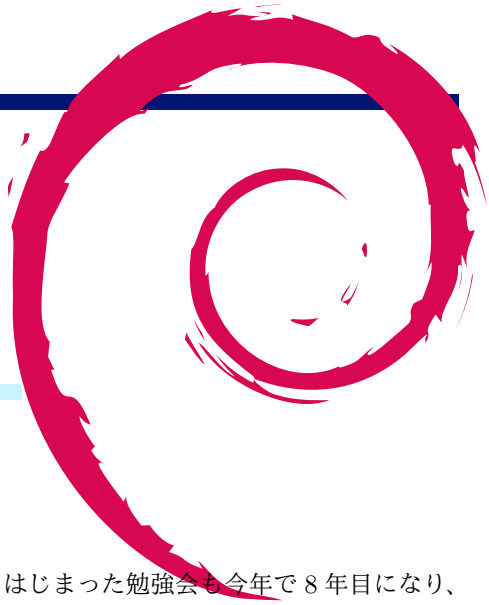
9.4 4th quarter

10 月 22 日、Debian Multimedia Maintainers からの活動報告があり、Codec のアップデート、新しいアプリの紹介などがありました。

11 月 5 日、Jessie のフリーズが行なわれました。

11 月 9 日、Debian 9 のコードネームは”Stretch”で Debian 10 のコードネームは”Buster”であるとリリースチームより報告がありました。

11 月 18 日、「init system coupling」の投票結果が「General Resolution is not required」となりました。



10　2014 年の振り返りと 2015 年の企画

Debian JP

2014 年も今月で最後の関西 Debian 勉強会になります。2007 年 3 月からはじまった勉強会も今年で 8 年目になり、来年には 100 回目を迎えようとしています。

10.1　勉強会全体について

10.1.1　もくもくの会

今年からは内容をもくもくの会を主体として開催するようにしました。各自抱えておられる作業を持ち寄って進めていくよい場となりました。しかし、参加されたことのない新規の方々にとっては、内容がよくわからないようで、障壁となってしまった面もあるようです。

10.1.2　セッション、定例ネタ

毎回のセッションは設定しませんでしたが、「Debian で楽しむ 3D プリンティング」、「自宅サーバに kvm を導入してみよう」、「Notmuch Mail」、「Debian での systemd とのつきあい方」、「Linux のドライバメンテナになった体験記」の 5 本の発表をしていただきました。

いわゆる定番のネタについてのセッションはできませんでした。来年は jessie のリリースにあわせて定番のネタ (Debian の入門的なお話、ライセンス、パッケージ作成、BTS) のセッションができたら良いかなと考えています。

10.2　運営

運営に関しては、毎月の継続した開催は行なってきていました。

申し込みのシステムとして Doorkeeper を試験的に使用しました。connpass など他の手段も試し、いわゆる今風の申し込みやすい環境を整えていきます。

昨年に続き、事前の準備、告知が不十分な場合がありましたので、今後は、定型の告知を先に投げるなど、改善していきたいと考えています。

また、運営に参加していただいていた八津尾さんが、関西から離れられることになり、運営から抜けられました。

10.3　イベント/NM 申請

イベント参加については、OSC Kansai@Kyoto、KOF2014 に参加しました。今後も継続して参加する予定です。今年は合宿や「大統一 Debian 勉強会」などのイベントの実施はありませんでした。

Debian 開発者への道、として一昨年から倉敷さんが NM プロセスの申請を進められています。今後も NM プロセスに挑む参加者や Debian Maintainer となって精力的にパッケージを開発/ 更新する人が増えると良いですね。

10.4　開催実績

関西 Debian 勉強会の出席状況を確認してみましょう。グラフで見ると図 24 になります。また、毎回の参加者、アンケート回答者の人数とその際のトピックを 表 8 にまとめました。グラフ中の黒線は参加人数、赤線は 1 年の移動平均、青線はアンケートの回答人数です。参加人数が 0 となっているところは人数が集計されていない or 開催されなかった月です。アンケートの回答人数が 0 となっているところはアンケートが実施されなかった月です。

事前課題は毎回設定しましたが、アンケートシステムは 1 回しか活用しませんでした。

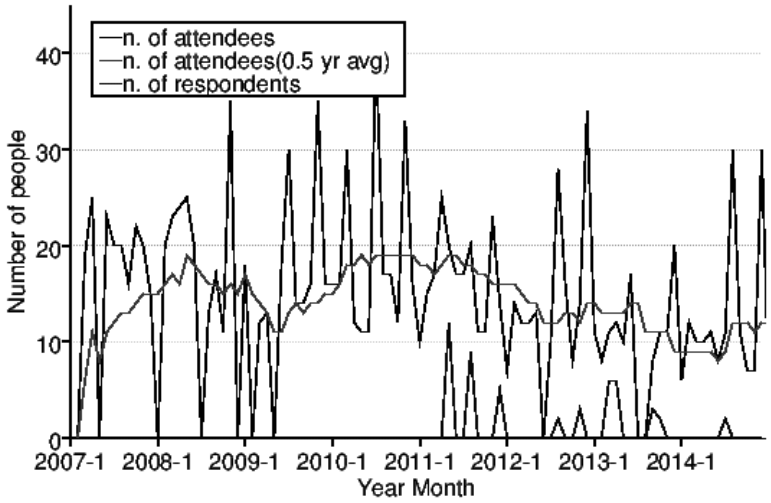


図 24　関西の参加人数推移 (参加人数と 6ヶ月移動平均、アンケート回答人数)

表 4　関西 Debian 勉強会の参加人数とトピック (2007-2008)

開催年月	参加人数	内容
2007 年 3 月	19	開催にあたり
2007 年 4 月	25	goodbye、youtube、プロジェクトトラッカー
2007 年 6 月	23	社会契約、テーマ、debian/rules、bugreport
2007 年 7 月	20 前後	OSC-Kansai
2007 年 8 月	20	Inkscape、patch、dpatch
2007 年 9 月	16	ライブラリ、翻訳、debtorrent
2007 年 10 月	22	日本語入力、SPAM フィルタ
2007 年 11 月	20 前後	KOF
2007 年 12 月	15	忘年会、iPod touch

開催年月	参加人数	内容
2008 年 2 月	20	PC Cluster, GIS, TEX
2008 年 3 月	23	bug report, developer corner, GPG
2008 年 4 月	24	coLinux, Debian GNU/kFreeBSD, sid
2008 年 5 月	25	ipv6, emacs, ustream.tv
2008 年 6 月	20	pbuilder, hotplug, ssl
2008 年 8 月	13	coLinux
2008 年 9 月	17	debian mentors, ubiquity, DFSG
2008 年 10 月	11	cdbs,cdn.debian.or.jp
2008 年 11 月	35	KOF
2008 年 12 月	?	TeX 資料作成ハンズオン

表 5　関西 Debian 勉強会の参加人数とトピック (2009-2010)

開催年月	参加人数	内容
2009 年 1 月	18	DMCK, LT
2009 年 3 月	12	Git
2009 年 4 月	13	Installing sid, Mancoosi, keysign
p 2009 年 6 月	18	Debian Live, bash
2009 年 7 月	30?	OSC2009Kansai
2009 年 8 月	14	DDTSS, lintian
2009 年 9 月	14	reportbug, debian mentors
2009 年 10 月	16	gdb, packaging
2009 年 11 月	35	KOF2009
2009 年 12 月	16	GPS program, OpenStreetMap

開催年月	参加人数	内容
2010 年 1 月	16	Xen, 2010 年企画
2010 年 2 月	16	レンタルサーバでの利用, GAE
2010 年 3 月	30?	OSC2010Kobe
2010 年 4 月	12	デスクトップ環境, 正規表現
2010 年 5 月	11	ubuntu, squeeze
2010 年 6 月	11	debhelper7, cdbs, puppet
2010 年 7 月	40?	OSC2010Kyoto
2010 年 8 月	17	emdebian, kFreeBSD
pp 2010 年 9 月	17	タイル WM
2010 年 10 月	12	initramfs, debian live
2010 年 11 月	33	KOF2010
2010 年 12 月	14	Proxmox, annual review

表 6　関西 Debian 勉強会の参加人数とトピック (2011-2012)

開催年月	参加	回答	内容
2011 年 1 月	10	0	BTS, kFreeBSD
2011 年 2 月	15	0	pbuilder, Squeeze リリースパーティ
2011 年 3 月	17	0	ライセンス, ドキュメント
2011 年 4 月	25	0	OSC Kansai@Kobe
2011 年 5 月	20	12	vi, dpkg
2011 年 6 月	17	0	vcs-buildpackage{svn, git}, IPv6
2011 年 7 月	17	0	OSC Kansai@Kyoto
2011 年 8 月	20	9	パッケージ作成ハンズオン
2011 年 9 月	11	0	vcs-buildpackage{bzr, git}
2011 年 10 月	11	0	Emacs, vim 拡張の Debian パッケージ, 翻訳
2011 年 11 月	23	0	KOF 2011
2011 年 12 月	13	5	NM プロセス, BTS

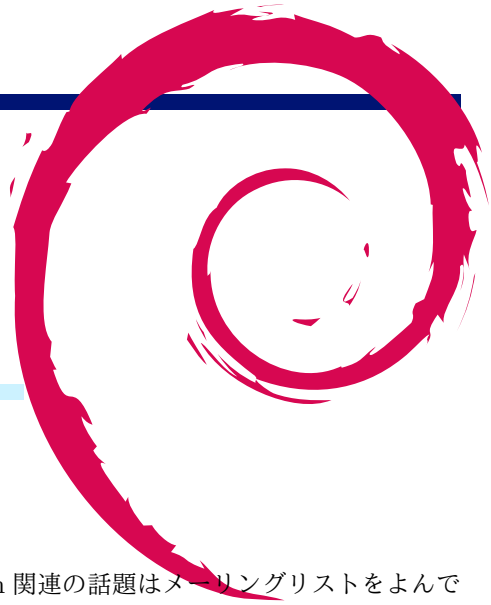
開催年月	参加	回答	内容
2012 年 1 月	7	0	Debian 温泉合宿
2012 年 2 月	14	0	autofs+pam_chroot, t-code, Debian Policy
2012 年 3 月	12	0	Konoha, t-code, Debian Policy
2012 年 4 月	12	0	フリーソフトウェアと著作権, Konoha, Debian Policy
2012 年 5 月	13	0	Debian と LDAP(頓挫), ITP 入門, Debian Policy
2012 年 6 月	-	0	大統一 Debian 勉強会
2012 年 7 月	10	0	Debian と LDAP, Debian Policy
2012 年 8 月	28	2	OSC Kansai@Kyoto
2012 年 8 月	16	0	Debian と Kerberos, News from EDOS
2012 年 9 月	8	0	clang, Debian Policy
2012 年 10 月	14	3	翻訳環境構築, DSA の舞台裏
2012 年 11 月	34	0	KOF 2012
2012 年 12 月	12	0	Debian on Android, Debian Policy

表 7　関西 Debian 勉強会の参加人数とトピック (2013)

開催年月	参加	回答	内容
2013 年 1 月	8	0	Using Drupal on Debian, 月刊 Debian Policy その 8
2013 年 2 月	11	6	Debian Installer トラブルシューティング, Ruby In Wheezy
2013 年 3 月	12	6	Ubuntu と GNOME Shell と私, 管理者視点からの GNOME の大規模な配置
2013 年 4 月	10	0	リリースノートを読んでみよう。,
			クラウド初心者が AWS に Debian をのっけて翻訳サービスの試行に挑戦してみた
2013 年 5 月	17	0	Debian と Ubuntu の違いを知ろう, Debian の歩き方
2013 年 6 月	-	0	大統一 Debian 勉強会
2013 年 7 月		0	OSC 2013 Kansai @ Kyoto, GPG キーサインパーティ
2013 年 8 月	8	3	puppet による構成管理の実践
2013 年 9 月	11	2	Linux とサウンドシステム, dgit でソースパッケージを触ってみる
2013 年 10 月	11	0	ALSA のユーザーランド解説, git-buildpackage 入門 again
2013 年 11 月	20	0	KOF 2013
2013 年 12 月	6	0	2013 年の振り返りと 2014 年の企画, 忘年会

表 8　関西 Debian 勉強会の参加人数とトピック (2014)

開催年月	参加人数	回答人数	内容
2014 年 1 月	12	0	LT, もくもくの会
2014 年 2 月	10	0	もくもくの会
2014 年 3 月	10	0	Debian で楽しむ 3D プリンティング, もくもくの会
2014 年 4 月	11	0	自宅サーバに kvm を導入してみよう, Notmuch Mail, もくもくの会
2014 年 5 月	8	0	もくもくの会
2014 年 6 月	11	2	Debian での systemd とのつきあい方, Linux のドライバメンテナになった体験記,
			キーサイン, もくもくの会
2014 年 8 月	30	0	OSC 2014 Kansai @ Kyoto
	11	0	もくもくの会
2014 年 9 月	7	0	もくもくの会
2014 年 10 月	7	0	もくもくの会
2014 年 11 月	30	0	KOF 2014
	4	0	インストーラテスト, もくもくの会
2014 年 12 月	9	0	2014 年の振り返りと 2015 年の企画, 忘年会



11 Debian Trivia Quiz

野島 貴英

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. 2014/11 月中旬にて Debconf15 のスポンサーとなった企業は何社あるでしょう？

☐ A 3 社
☐ B 9 社
☐ C 11 社

問題 2. 2014/11/18 に遂にグラフ表示機能付き電卓で Debian を動かした強者が現れた事が話題になっていました。使われた電卓のメーカはどれ？

☐ A Texus Instruments 社
☐ B CASIO 社
☐ C Hewllet Packard 社

問題 3. Debian プロジェクトにて init システムのアプリケーションの依存度に関する General Resolution の投票が行われました。投票の結果は？

☐ A パッケージは特定の init システムのみに依存してはならない
☐ B パッケージメンテナが望めば、特定の init システムに依存しても OK
☐ C General Resoultion は不要だ

問題 4. Debian Med メタパッケージのバージョンが 1.99 へ一気にジャンプする予定であることがアナウンスされました。この時初めて搭載される予定の医療関係者向けシステムは何？

☐ A 遺伝子解析システム
☐ B 外科手術用ロボットシステム
☐ C 病院情報システム (HIS)

問題 5. 2014/11/14 に Debian の BTS に、Debian のパッケージ開発初心者へ修正作業が良い演習となるようなバグに関する tag が設けられました。以下のどれ

☐ A newcomer
☐ B gift
☐ C easeofcake

問題 6. Mysql に関する仮想パッケージに定義された MysqlDB 互換 DB ソフトとして PXC というのがあるが、こちらは何の略？

☐ A Protected eXchange Controler
☐ B Posgresql eXtra Controler
☐ C Percona XtraDB Cluster

問題 7. 2014/11/20 にて、systemd 用の定義ファイル内部で使えるセキュリティを大幅に向上出来るオプションのうち、ホームディレクトリの不正アクセスを抑止するのは次のどれ

☐ A ProtectHome

☐ B ProtectSystem

☐ C PrivateTmp

問題 8. 12/29 に、Debian のサービスと連携できる chromium 向けの拡張機能がリリースされました。何と連携できるのでしょうか？

☐ A tracker.debian.org

☐ B sources.debian.net

☐ C www.debian.or.jp

問題 9. 2014/11/11 に Raphaël Hertzog さんにより提案された DEP14 の内容は以下のどれ？

☐ A git パッケージングリポジトリの使い方について

☐ B control ファイルに Debian-Homepage フィールドを入れる件

☐ C debian/upstream/metadata を用意する件

問題 10. 2015/1/8 Marvel 社より Debian ARM 用パッケージビルドマシンの寄付がありました。さて何台？

☐ A 7 台

☐ B 8 台

☐ C 10 台

問題 11. 2015/1/10 に wheezy がアップデートされました。さてバージョンは次のどれ。

☐ A 7.6

☐ B 7.7

☐ C 7.8

問題 12. 2015/1/1 にある長さ未満の Debian keyring の消去が完了したそうです。ある長さとは?

☐ A 2048

☐ B 4096

☐ C 8192

問題 13. 2014/12/6 に cdn.debian.net のレコードの示す先を、とある FQDN の CNAME としたい旨の問い合わせが debian-devel ML にありました。どこに向ける？

☐ A ftp.debian.or.jp

☐ B ftp.jp.debian.org

☐ C http.debian.net

問題 14. 2015/1/9 に debian-devel ML にて、一部パッケージの description が長すぎる事の議論がありました。最も長い description の行を持つパッケージは？

☐ A firmware-linux-nonfree

☐ B texlive-latex-extra

☐ C irssi-scripts

問題 15. 2015/2/6 に、とある国に security mirror が新設されたことがアナウンスされました。どこの国？

☐ A 日本

☐ B アメリカ

☐ C ジャマイカ

問題 16. 2015/1/26 に、Debian Installer のリリースのアナウンスがされました。こちらのバージョンはいくつ?

☐ A Jessie RC 1

☐ B Jessie RC 2

☐ C Jessie RC 3

問題 17. 過去に諸々の理由により testing から Remove されてしまった Jessie 向けのパッケージについて、再度復活できる最後のチャンスの締切り日はいつでしょう？

☐ A 2/5 中

☐ B 2/14 中

☐ C 2 月末中

問題 18. 2015/2/13 にて GSoC 2015 の Debian に関するプロジェクト募集と GSoC のメンター募集が行われました。2/15 現在寄せられていないプロジェクトは？

☐ A Apport of Debian

☐ B Coinstallable PHP Versions

☐ C Debian Gnu/Hurd enhancements

問題 19. 2015/2/13 に Reproducible Builds の報告がありました。現在 main パッケージの何 % が完了しているでしょう？

☐ A 62%

☐ B 83%

☐ C 100%

問題 20. 2015/3/3 に DebConf15 のアナウンスが行われました。スポンサードな参加の登録期限は何時までででしょう？

☐ A 2015/3/7

☐ B 2015/3/15

☐ C 2015/3/29

問題 21. 2015/3/4 に DPL の今年の選出についてアナウンスがありました。DPL の立候補〆切は何時？
☐ A 2015/3/4
☐ B 2015/3/9
☐ C 2015/4/1

問題 22. 2015/3/1 にて、cdn.debian.net がその役割を終え、ある FQDN を指すだけになるということがアナウンスされました。この FQDN は以下のどれ？
☐ A ftp.debian.org
☐ B http.debian.net
☐ C sources.debain.net

問題 23. 2015/2/17 にて、DPL の Lucas Nussbaum により立て直しの協力募集が行われたチームは？
☐ A DSA team
☐ B partners team
☐ C 東京エリア Debian 勉強会 team

問題 24. 2015/2/23 に、Debian がクレジットに載せられた映画があることが indenti.ca の zak の投稿で判りました。何と言う名前の映画？
☐ A ミレニアム ドラゴン・タトゥーの女
☐ B Toy Story
☐ C Citizenfour

問題 25. 2015/3/31 に Jessie のリリース目標の日がアナウンスされました。いつでしょう？
☐ A 4/1
☐ B 4/18
☐ C 4/25

問題 26. 2015/4/16 の lucas 最後の bit from DPL が流れました。こちらに記載されていた Debian からパッケージをリリースできるようにするための法的対策を検討中のソフトウェアは以下のどれ?
☐ A libdvdcss
☐ B OTR software
☐ C Unreal Engine4

問題 27. Debian にて開発者の属性の不均衡を正す事を目的として活動する公式チームが Debian Project に新設したと 4/16 に lucas によりアナウンスがありました。なんという名前のチーム？
☐ A Debian Release team
☐ B Debian Outreach team
☐ C Technical Comittie

問題 28. 2015/4/15 に選出された新 DPL は誰？
☐ A Mehdi Dogguy
☐ B Gergely Nagy
☐ C Neil McGovern

問題 29. Jessie が無事リリースされました。いつだったでしょうか？
☐ A 2015/4/11
☐ B 2015/4/18
☐ C 2015/4/25

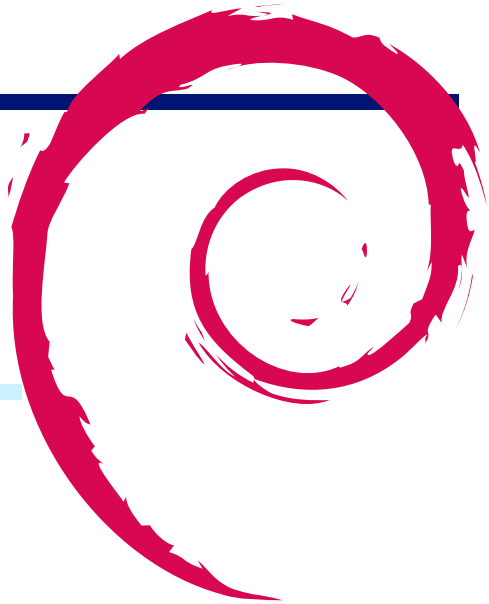
問題 30. Jessie で初めて追加されたものではないものが混ざっています。どれ？
☐ A Debian Games Blend
☐ B OpenJDK
☐ C androidsdk-tools

問題 31. Debian GNU/Hurd 2015 も 2015/4/30 にリリースされました。心臓部の GNU Mach のバージョンはいくつ？
☐ A 1.6
☐ B 1.5
☐ C 1.4

問題 32. 2015 年の GSoC に採択された、Debian MIPS ports についての開発内容は次のどれ？
☐ A 多数のビルド出来ないパッケージを、ちゃんとビルドできるようにする。
☐ B 新しい MIPS CPU への対応
☐ C 新しい MIPS CPU 搭載製品への対応

問題 33. 遂に http.debian.net が debian.org のインフラに移動となりました。新しい URL はどれ？
☐ A http://http.debian.org/debian
☐ B http://httpredir.debian.org/debian
☐ C http://www.debian.org/

問題 34. debian への libdvdcss/ZFS パッケージ搭載の件について、2015/5 の Bit From DPL で報告された状況は以下のどれ?
☐ A DPL がいろいろ議論して回っている状況
☐ B 搭載日確定した
☐ C 搭載を諦めた



12 索引

13 Debian Trivia Quiz 問題回答

野島 貴英

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

1. C スポンサー企業一覧：credativ, sipgate, Matanel Foundation, Google, Farsight Security, Martin Alfke / Buero 2.0, Ubuntu, Mirantis, Logilab, Netways,Hetzner。さあ！日本企業の皆様も是非ご一緒に！

2. A 詳しくは：http://hackaday.com/2014/11/18/ running-debian-on-a-graphing-calculator/。使われた電卓の製品名は：TI-NSpire CX というシリーズの機種だそうです。

3. C Debian でも init system とパッケージとの調整が引き続き一生懸命行われています。

4. C Debian Med チームの今回の頑張りは素晴らしく、PHYLP のライセンスを長年の交渉の末、DFSG 準拠のものに変更してもらうなど、医療関係者ら・生物学関係者らに様々にポジティブな影響を与えています。この活動は医療分野の学術系の情報サイト BioMedCentral に”Community-driven development for computational biology at Sprints, Hackathons and Codefests” ということで取り上げられたそうです。

5. A 今までパッケージ開発初心者向けに割り当てられていた BTS の tag は gift でしたが、newcomer へ変わるそうです。以前の gift という tag 名だとちょっと判りにくいとのことで、今回名前変更となったようです。

6. C Debian にて、Mysql 系の DB とみなされたものは、３つあり、MysqlDB,MariaDB,Percona XtraDB Cluster(PCX) の３つとなります。

7. A systemd の定義ファイルに用意されているセキュリティに関するオプションをうまく使いこなすと非常に強固なシステムにすることが出来るそうです。ノウハウと紹介は http://0pointer.net/public/systemd-nluug-2014.pdf

8. B zack さんの Debconf14 のプレゼンを見てインスパイアされたとのことで、Raphael Geissert さんが作った chromium 用 (chrome でも動く）のオンラインソースコードエディタを実現する拡張機能です。Debian パッケージのソースコードを直接オンラインで閲覧できる sources.debian.net 上のコードをこの拡張機能を使ってシームレスに編集可能になります。zack さんの示した深刻な問題（第 117 回東京エリア Debian 勉強会資料参照）を解決する為の画期的なアイデアの 1 つです！マヂすばらしい！自由ソフトウェアラブ、Debian ラブな人は、早速、chromium へ Nautilus からこの拡張機能を投げ込んで試すべし！

9. A パッケージングリポジトリのブランチとタグにベンダー情報（例：debian/*,ubuntu/*) やらバージョン情報の入れ方などを含む、諸々のルールの提案。詳しくは、http://dep.debian.net/deps/dep14/参照。なお、Debian-Homepage フィールドの件は DEP13。debian/upstream/metadata の件は DEP12。

10. B Marvel 社が提供したサーバは MV78460 SoC ベースのボード 8 つを提供とのこと。なお、MV78460 は、ARM v7 コア４基を搭載する Marvel 社開発の SoC。参考：http://www.marvell.com/embedded-processors/armada-xp/

11. C wheezy をお使いの方々は早速アップグレードしましょう。

12. A アナウンスと、消された該当のキー一覧は debian-devel-announce の 2015/1/1 の記事に記載があります。https://lists.debian.org/debian-devel-announce/2015/01/msg00000.html

13. C cdn.debian.net を提供している Araki さんからの問い合わせ。squeeze 以上で搭載されている apt ver 0.7.21 では http の redirect に対応しているので、利用者に最も近いパッケージダウンロード先を教えてくれる機構を http.debian.net に一本化したいとのこと。特に反対する人は居ない模様です。

14. B 2015/1/9 現在、unstable/main では、最も長い行数の順でいくと、1 位 texlive-latex-extra(実に 1935 行!!)、2 位 texlive-fonts-extra(437 行)、3 位 irssi-scripts(350 行)。unstable/non-free では、1 位 firmware-linux-nonfree(251 行) です。

15. A Debian の security mirror を新設していただいたのは、SAKURA Internet さん (http://www.sakura.ad.jp/) となります。こちらありがとうございます！OSC 2015 Tokyo/Spring(http://www.ospn.jp/osc2015-spring/) へ行かれる方は、SAKURA Internet さんのブースにてよろしくお伝えくださいませ！また、こちらの新設に尽力された、やまねさんと Debian JP Project の関係者の皆様、お疲れさまでした！

16. A Debian Installer のリリースの話で、Jessie 本体のリリースの話では無いです。で、今回 1/26 に RC1 がリリースされたことがアナウンスされました。バグ出し、特に UEFI 環境をおもちの方、動作テストしてバグ出しと BTS へのレポートよろしく

おねがいしますー。
17. A 2/5 23:59:59 UTC までだそうです。もう終わっちゃいましたね。皆様のパッケージは無事でしたでしょうか？
18. C 詳しくは、https://wiki.debian.org/SummerOfCode2015/Projects。Apport は、プログラムがクラッシュしたときに割り込み、デバッグ情報や環境の情報を BTS へ送るシステム。Coinstallable PHP Version は Debian にて、php 5.X の各バージョンごとの PHP パッケージ/PHP 環境を同居して導入できるようにするプロジェクト。
19. B Reproducible Builds とは、現在お使いの Debian の各バイナリパッケージが、本当に現在のソースパッケージから生成されたもので、悪意のあるバイナリが混じっていないかを調べる試み。簡単に言うと、ソフト製造者側の製品本体のウィルス感染チェックみたいなもの。Debian の他にも、Fedora/OpenSUSE/NixOS でも行われている。詳しくは https://wiki.debian.org/ReproducibleBuilds。また、本プロジェクトのについてのプレゼンは: http://reproducible.alioth.debian.org/presentations/2014-02-01-FOSDEM14.pdf
20. C 宿泊代と食事が DebConf 開催側持ちとなるスポンサードな参加登録の〆切は 3/29 です！〆切の時刻は UTC なのか、ドイツのローカルタイムか、細かい事がわからないので、参加を検討されている方は〆切に対して十分に日程の余裕をもって登録される事をおすすめします。DebConf15 は、ドイツ ハイデルバーグで 2015/8/15-22 の開催です。DebCamp は、8/9-8/14 となります。
21. B 毎年恒例の DPL 選挙です。DPL の立候補〆切は 3/9 で、選挙は 4/1 から行われます。今年は誰が立候補するのでしょうか？また、同時に、Debian JP Project についても、2015 年の会長の立候補者募集が行われています。
22. B cdn.debian.net は、apt による Debian パッケージの取得先について、DNS クエリの発行された場所に基づいて、ユーザに最も近いパッケージリポジトリのサーバの IP アドレスを DNS のレコードで返却する仕組みです。Araki さんによって 2007 年頃にて設計、構築、運用されておりました。2010 年に apt が HTTP リダイレクトをサポートできるようになったため、cdn.debian.net の機能を HTTP のリダイレクト機能で実現する http.debian.net が稼働を開始しました。今回、遂に、cdn.debian.net を引退させるアナウンスが行われたという状況です。もし、source.list に cdn.debian.net の FQDN を指定している場合は、http.debian.net に変えましょう。
23. B Debian Project には、Debian パートナーズプログラム (https://www.debian.org/partners/) という制度があります。こちらを統括しているチームとして partners team があるのですが、昨今、こちらの構成員の時間が取れない状態らしく、対応が滞っているという状況のようです。なお、DSA は The Debian System Administrators というチームのことでバリバリ活動されてます。東京エリア Debian 勉強会 team は当勉強会幹事の頭の中の妄想上のチームですが、頑張ってるぜ！
24. C Citizenfour というエドワード・スノーデンを題材にしたドキュメンタリー映画のクレジットの Special Thanks として Debian が入りました。imdb にも company として登録されました。Citizenfour は 2015 年アカデミー賞のうち長編ドキュメンタリー賞を受賞しました (http://www.imdb.com/company/co0504449/, http://www.huffingtonpost.jp/tatsuhei-morozumi/citizenfour_b_6741756.html) ギャガ株式会社配給で将来日本でも公開されるとのことですので、将来クレジットを確認すると良いかもしれません
25. C 遂に 4/25 が Jessie のリリース目標の日取りとなりました。ただ、最悪のケースとして、Jessie にリリースに支障があるような非常に重大な問題が見つかり修正が間に合わない場合は、ずれる可能性があるとは記載されています。現在 udd を参照すると、キーパッケージだけでも 47 個ぐらい RC バグが残ったままのようですので、Fix! Fix! なお、4/25 に東京でリリースパーティーが企画されています。詳細は connpass の Debian JP グループを見て下さいませ。
26. A なんと、libdvdcss と、ZFS が検討中らしい。もちろん、ダメになる可能性は十分にあるので、状況を静観したいと思います。libdvdcss はそのコード本体がクラッキング行為の塊なので、万一 Debian に入った場合、日本で Debian をミラーしても大丈夫かどうかは今後の議論ですね
27. B 現在、是正すべき属性のターゲットとしては性別分布を目標としているようです。現在、Debian 公式開発者の性別の割合は男性に大きく偏ってしまっています。Debian は多様性を重視するプロジェクトです。将来、性別の不均衡が是正されると良いですね。
28. C Neil McGovern さんは Debian Project で Release Team で活動されるなど多数の貢献をされている方です（所信表明は : https://www.debian.org/vote/2014/platforms/neilm ）。ちなみに、2015 の Debian JP Project 会長は岩松さんが選出されました。おめでとうございます。
29. C 無事 4/25 に Jessie(Debian 8) がリリースされました。Linux kernel は 3.16.7、GNU Compiler Collection 4.9.2、GNOME 3.14、Apache 2.4.10、PHP 5.6.7 などのバージョンのものが搭載されました。搭載されているパッケージ数は全部で 43,000 パッケージ以上もあります。
30. B Jessie での変更点はリリースノート (https://www.debian.org/releases/ jessie/amd64/release-notes/index.ja.html) にあります。Debian Games チームよりゲームの詰め合わせの Debian Blend が追加されたり、android の開発ツールの一部も追加されたりしています。
31. B 2015/4/10〜2015/4/15 周辺で GNU/Hurd（upstream 側) のアップデートが行われ、GNU Hurd は version 0.6 に、GNU Mach は 1.5 になりました。未だに KVM/QEMU などの仮想環境での動作が推奨の 32bit システムですが、多くの OS がクラウド環境で動作している世の中ですので、ホビー用途の利用でそろそろ注目されても良いかも？と思う次第です。
32. A Debian MIPS ports の多数のパッケージは再ビルドが出来ない、インストールに失敗するものが多数ある状況です。こちらを再ビルドできるように修正し、インストール出来るようにする事が採択されました。GSoC に限らず、いつでも本件の協力者

募集中ですので、MIPS ports に興味ある方はパッケージの修正に是非ご協力くださいませー。
33. B http.debian.net は HTTP の Redirect を活用し、ユーザに最も適切な場所にある、パッケージのミラー先を教えてくれるサービスです。遂に、実験的サービスの扱いである debian.net から Debian 公式サービスである debian.org へ移動となりました。http.debian.net を source.list に設定している人は、httpredir.debian.org へ修正をおねがいします。また、不具合は”mirrors” pseudo-package で BTS するようです。参照：http://bugs.debian.org/mirrors
34. A 2015/5/19 現在、FTPMasters, SFLC, FSF と議論中とのこと。さらに近々Linux Foundation とも議論予定。現在、当初想定したよりも、実現に関する状況が複雑とのことで、本件いつ決着するか見えなくなったとの事です。

『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』（第 121 回から第 126 回）および『関西 Debian 勉強会』（第 92 回から第 98 回）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。関西 Debian 勉強会資料の以下は収録無し。第 93 回は Debian Bug Squashing Party、第 94 回はライトニングトーク、第 95 回は資料無し、第 96 回は Debian8 ”Jessie” リリースパーティ。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 **2015** 年夏号

2015 年 8 月 16 日　　初版第 1 刷発行

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