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 2014 年冬号 2014 年 12 月 30 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

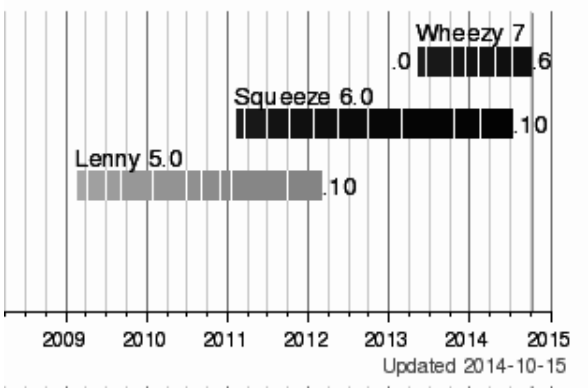
それでは、楽しい一時をお楽しみ下さい。

2 Debian Updates

岩松 信洋

2.1 ポイントリリース

- Debian 7 更新状況 (Wheezy)
  - 2014-04-26 Debian 7 更新: 7.5 リリース
  - 2014-07-12 Debian 7 更新: 7.6 リリース
- Debian 6 更新状況 (Squeeze)
  - 2014-07-19 Debian 6.0 更新: 6.0.10 リリース



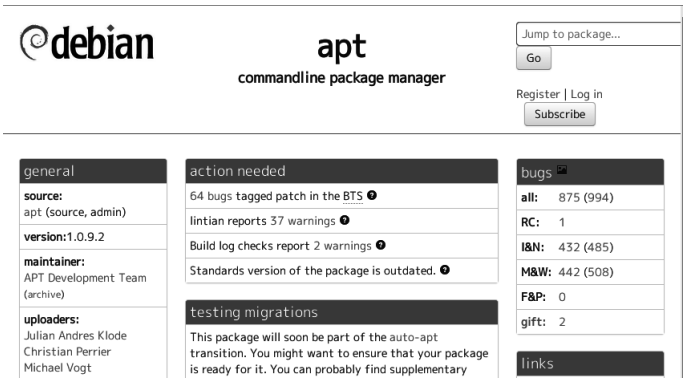
2.2 Long Term Support 開始

- 2014/5/31 に Debian 6.x サポート終了
- 2014/5/27 Debian 6.x 長期サポート (Long Term Support、LTS) 開始
  - Debian GNU/Linux 6.x 向けのセキュリティ更新を 2016 年 2 月 6 日まで提供（Squeeze のリリースから 5 年）
  - i386 と amd64 だけをサポート
  - 全てのパッケージがサポートされるわけではなく、サポート対象から外されるものもある
  - サポート対象外のパッケージは debian-security-support パッケージで管理されている（check-support-status コマンドで確認できる）

```
deb http://http.debian.net/debian/ squeeze main
deb http://http.debian.net/debian squeeze-lts main
```

2.3 tracker.debian.org

- Debian Package Tracking System(`packages.qa.debian.org`) の置き換え
- Django framework を使用



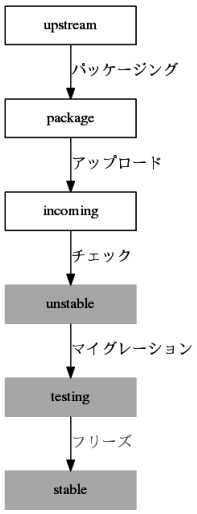
2.4 Debconf 14

- 8 月 23 日から 31 日まで、アメリカのポートランドで開催
- 参加者は約 300 名（日本からの参加は 2 名）
- カンファレンスのビデオや資料は `http://debconf14.debconf.org` から参照可能
- Debconf15 は ドイツのハイデルベルクで開催

2.5 Debian 8.0 状況

近日フリーズ

- 11/5 にフリーズ（予定）
- フリーズ: パッケージの新しいバージョンへの更新を停止
  - 9/5 に ABI 更新をフリーズ
  - 10/5 から testing へのマイグレーションを 5 日から 10 日に切り替え



2.6 リリースゴール

- systemd
- piuparts
- SELinux
- CrossToolchains
- CrossBuildableBase
- utf-8
- debian/rules honor CC and CXX
- Clang as secondary compiler
- Security hardening build flags

2.7 リリースゴールからはずされたもの

- pkg-php-tools 移行 (PEAR:PHP Extension and Application Repository サポート)

2.8 サポートアーキテクチャ

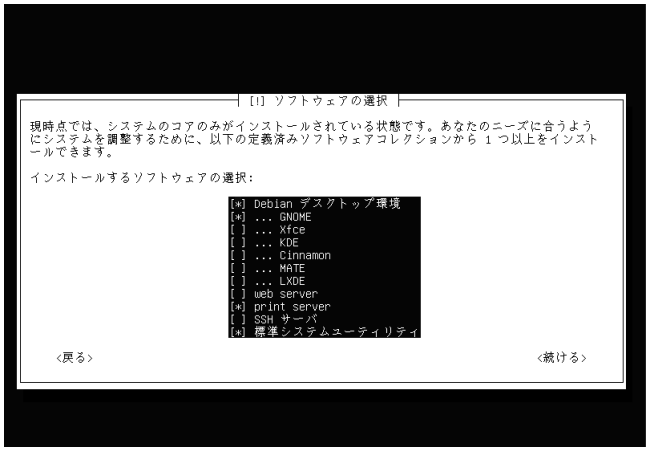
- IN: 未定
  - arm64, ppcel64 はまだ決定ではない。11 月 1 日の状況によって決まる
  - x32 は入らなかった模様。
- OUT: ia64、sparc、s390
  - ia64 は debian-ports でメンテナンス
  - s390 は s390x に移行

2.9 デフォルト init システム

- Linux のデフォルトの init システムが systemd に
- minimal でインストールしても /sbin/init が systemd

2.10 デフォルト Desktop Environment

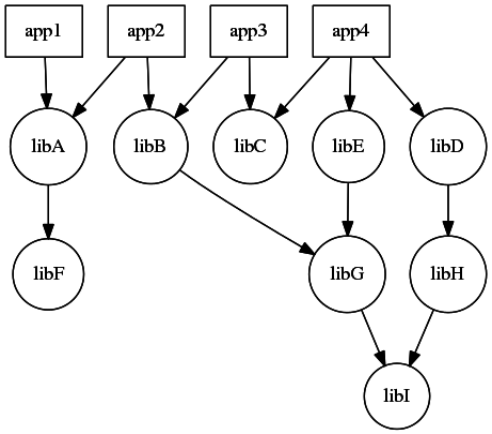
- GNOME
- 他の DE が tasksel で選択可能に



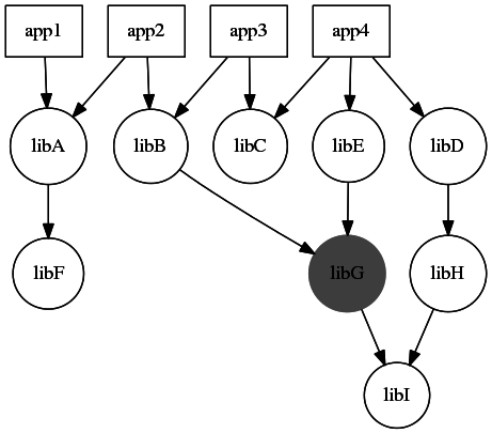
2.11 auto-removal によるリリース対策

- RC バグに 14 日なにも進展がない場合、そのバグをもつパッケージは自動的に testing から削除される。

2.12 auto-removal によるリリース対策

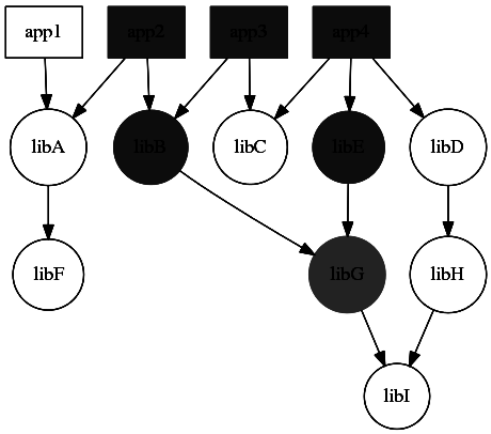


2.13 auto-removal によるリリース対策



libG が auto-rm される

2.14 auto-removal によるリリース対策



libG に依存しているパッケージも auto-rm される

2.15 auto-removal によるリリース対策

- RC バグに 14 日なにも進展がない場合、そのバグをもつパッケージは自動的に testing から削除される。
- RC バグを持つパッケージとそれに依存しているパッケージが入らない可能性大
- ただし、主要なパッケージは除く
  主要パッケージ一覧
  `https://udd.debian.org/cgi-bin/key_packages.yaml.cgi`

2.16 主要ソフトウェアの更新状況

- Linux kernel: 3.16.x (3.17.1)
- kFreeBSD: 8,9,10 (10.1)

2.17 主要ソフトウェアの更新状況

- GNOME: 3.14 (3.14)
- KDE: 4.14.1 (4.14.1)
- Xfce: 4.10 (4.10)
- LXDE: 0.7.1 (lxqt 0.8.0)
- MATE: 1.8.0 (1.8.0)
- Cinnamon: 2.2.16 (2.2.16)

2.18 主要ソフトウェアの更新状況

- Toolchain: GCC:4.9.1. binutils: 2.24.51, glibc: 2:19
- LLVM: 3.4.2, 3.5.0 (3.5.0)
- Perl: 5.20.1 (5.20.1)
- Ruby: 2.1 (2.1)
- Python2: 2.7.8 (2.7.8)
- Python3: 3.4.2 (3.4.2)

- PHP: 5.6.0 (5.6.2)
- Lua: 5.2.3 (5.3.0 alpha)
- ghc: 7.6.3 (7.8.3

2.19 主要ソフトウェアの更新状況

- Xorg: 1.16.1 (1.16.1)
- Wayland: 1.6.0 (1.6.0)
- systemd: 215 (216)
- Emacs: 24 (24.3)
- Vim: 7.4.430 (7.4)
- Pulseaudio: 5.0 ()5.0
- Apache: 2.4.10 (2.4.10
- VLC: 2.2.0 pre3 (2.1.5)
- zabbix: 2.2.6 (2.4.0)
- rails: 4.1.6 (4.1.6)
- django: 1.7 (1.7)
- QEMU: 2.1 (2.1.2
- lxc: 1.0.5 (1.0.6)

2.20 Debian Installer Jessie Beta2

- 10/6 に Debian Installer Jessie Beta 2 がリリース
- arm64, ppcel64 イメージも追加
- QEMU などで気軽に試せる

2.21 まとめ

- 11/5 にフリーズ予定
- arm64, ppcel64 はまだ入るかわからない状態
- デフォルトの init システムが systemd になる
- デフォルトの DE は GNOME。tasksel から他の DE も選択可能に
- RC バグを持つパッケージとそれに依存しているパッケージが Jessie に入らない可能性大
- 主要パッケージは新しいものが入る（と思われる）
- 既にインストーラが RC としてリリース済み。テストできる環境は整っている

2.22 リリースに向けてお願い

- 使っているパッケージでバージョンが古いものがありましたらご相談ください。ライブラリパッケージ以外はまだアップデートできる可能性があります。
- Debian 8.0 で使いたいフリーソフトウェアがあったらご相談ください。まだ取り込まれる可能性があります。
- テストにご協力ください。インストールレポート、ソフトウェア動作レポート、未翻訳など報告していただけると非常に助かります。

3 Debian での systemd とのつきあい方

佐々木 洋平

Yes, it is written systemd, not system D or System D, or even SystemD. And it isn't system d either.

Spelling - `http://www.freedesktop.org/wiki/Software/systemd/`

3.1 はじめに

少し前の話ですが、次期 Debian 安定版 Jessie でのデフォルトの Init として systemd が採用されました。勉強会参加者の皆様におかれましては、`debian-devel@lists.debian.org` へ流れた~~品の無い~~悲鳴 (?) も記憶に新しいことでしょう[*1]。

色々な意見はあるでしょうけれども、「デフォルト」として採用される以上 (好むと好まざるとにかかわらず)Debian の開発者 (含ワナビー) にとって systemd の知識は必須事項になりそうです。そんな訳で、systemd そのものについて調べた結果と Debian における現状についてまとめてみます。ちなみに主にテストした環境は以下の通り:

```
% LANG=C date
Sat Jun 21 10:15:59 JST 2014
% lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux unstable (sid)
Release:        unstable
Codename:       sid
% dpkg -l | grep systemd
ii  libpam-systemd:amd64        204-10   amd64   system and service manager - PAM module
ii  libsystemd-daemon0:amd64    204-10   amd64   systemd utility library
ii  libsystemd-id128-0:amd64    204-10   amd64   systemd 128 bit ID utility library
ii  libsystemd-journal0:amd64   204-10   amd64   systemd journal utility library
ii  libsystemd-login0:amd64     204-10   amd64   systemd login utility library
ii  systemd                     204-10   amd64   system and service manager
ii  systemd-gui                 1:3-2    all     transitional package for systemd-ui
ii  systemd-sysv                204-10   amd64   system and service manager - SysV links
ii  systemd-ui                  3-2      amd64   graphical frontend for systemd
```

本原稿の執筆時間がよくわかります[*2]。一応 whezzy + wheezy-backports でもテストはしてみましたが。

... しかし, v204 って...。

[*1] Fsck SystemD and its developers and its users. GR to override this please.
`https://lists.debian.org/debian-devel/2014/02/msg00316.html`

[*2] lsb-base と lsb-release しかインストールしていないので `lsb_release -a` の出力結果はこんなモンです。

3.2 そもそも systemd ってナニよ?

systemd は Lennart Poettering 氏[*3]によって開発されている Init の代替プログラムでした。今では単なる「Init の代替」というより「Linux のサービス (デーモン) 管理フレームワーク」となっています。開発は `freedesktop.org`[*4]で行なわれており、ライセンスは LGPL-2.1+、最新バージョンは v214 となっています。

「Linux のサービス (デーモン) 管理フレームワーク」としての systemd の特徴は

1. Init として、SysV および LSB init script との互換性の提供。
2. サービスの起動をソケットとバス (D-Bus) で行なう。
3. サービス間の依存関係を明確にし、よりアグレッシブに並列起動する。
4. オンデマンドなサービスの起動。
5. プロセス管理を pid ではなく cgroups(control groups) で行なう。

といった所です。

既に Linux カーネル用のデバイス管理ツールである udev が systemd のソースにマージされており、デバイスの状態に応じてオンデマンドにサービスが起動したりします。将来的には cron や acpid の代替機能も提供する予定らしいです[*5]。その他、systemd に関する開発者の思想、現状に関しては

- `http://0pointer.de/blog/projects/systemd.html`
- `http://0pointer.de/blog/projects/systemd-for-admins-1.html` から始まる一連のエントリ[*6]

が非常に参考になるでしょう。

3.3 Debian で使うには?

systemd の必要要件は以下の通りです:

1. Linux カーネルのバージョンは 2.6.39 以上
2. 以下の機能が有効となっていること
   - devtmpfs
   - fanotify
   - autofs4
   - cgroups

カスタムカーネルを使用されている場合には、カーネルのバージョン等にご注意下さい。

Debian でパッケージとして提供されている systemd のバージョンは

- wheezy: v44
- wheezy-backports: 204-8~bpo70+1
- jessie/sid: 204-10

です。以下、パッケージとして提供されている最新版である v204 のお話をします[*7] では導入してみましょう。wheezy をお使いの方は wheezy-backports を有効にして下さい。init を sysvinit から置き換えるために、`systemd-sysv` を

[*3] Red Hat Inc. のエンジニアさんです。systemd の開発者だけでなく様々なフリーソフトウェアの開発者でもあります。例えば PulseAudio とか Avahi なんかのメイン開発者でもありますね。

[*4] `http://www.freedesktop.org/wiki/Software/systemd/`

[*5] ここまで来るとやりすぎな感も否めませんが...。

[*6] 現在#20 まであります。長くて読むの辛い...。

[*7] ちなみに、v44 を使用する場合の手順は、`systemd` を install →起動時に `init=/lib/systemd/systemd` と指定する or grub エントリの書き換えを行なう、です。

導入します。

```
% sudo aptitude install systemd-sysv -t wheezy-backports <-- wheezy の場合
% sudp aptitude install systemd-sysv                     <-- jessie/sid の場合
```

wheezy の場合には sysvinit が core パッケージなので、インストール時の依存関係解決に多少手間どるかもしれません。以下のパッケージが依存関係でインストールされます。

```
=============================================================================
[インストール] libpam-systemd:amd64
[インストール] libsystemd-journal0:amd64
[インストール] libudev1:amd64
[インストール] systemd:amd64
[インストール] systemd-sysv:amd64
[削除] sysvinit:amd64
[更新] libsystemd-daemon0:amd64 44-11+deb7u4 -> 204-8~bpo70+1
[更新] libsystemd-login0:amd64 44-11+deb7u4 -> 204-8~bpo70+1
=============================================================================
```

インストールが終わったら reboot します。... 無事起動できたでしょうか?

systemd は boot プロセスの解析ツールがあり、起動時にかかった時間が直ぐにわかります。

```
% systemd-analyze
Startup finished in 1.632s (kernel) + 3min 2.710s (userspace) = 3min 4.343s
```

... あれ?

```
% systemd-analyze blame
       3min 12ms dnsmasq.service
        2.074s systemd-udev-settle.service
         630ms psd.service
         307ms NetworkManager.service
         127ms squid3.service
         106ms bluetooth.service
          98ms udisks2.service
          96ms exim4.service
          93ms keyboard-setup.service
          87ms bootlogs.service
          87ms avahi-daemon.service
          80ms resolvconf.service
          76ms console-setup.service
          74ms networking.service
          73ms systemd-logind.service
          69ms netfilter-persistent.service
          64ms console-kit-log-system-start.service
          ... 以下略...
```

燦然と輝く **3min 12ms dnsmasq.service**。…おかしい。これはおかしい。

3.4 現状どうなのよ?

気を取りなおして。

3.4.1 systemd の用語

systemd を理解するために、先ずは用語を確認しましょう。

- ユニット (unit):
  SysVinit の初期化スクリプトにまとめて含まれていた個々の処理を抜き出した最小単位。個々の Unit はそれぞれ以下の通り:
  - サービス (`.service`):
    デーモンを起動する
  - ソケット (`.socket`):
    systemd が指定された socket を監視し、接続があると指定の`.service` を起動し、socket を渡す。inetd の様な役割。
  - ターゲット (`.target`):

複数のユニットをまとめて、依存関係や順序関係を定義する。SysVinit の runlevel に対応[*8]

- マウントポイント (.mount):
  ファイルをマウントする
- スワップ (.swap):
  スワップを有効化
- デバイス (.device):
  udev がデバイスを認識すると有効化される。

このうち、.mount,.swap は/etc/fstab から、.device は udev から、それぞれ自動生成されます。これら「ユニット」を定義したファイルは/lib/systemd 以下に置かれ、/etc/systemd 以下の適切な場所へ symbolic link をはることで有効になります。

また、ターゲットは SysvInit の runlevel に (なんとなく) 対応しており、Debian の場合には となっています。起

SysVinit の runlevel	systemd の target
0	poweroff.target
1	rescue.target
2 - 5	multi-user.target
6	reboot.target

表 1　Debian での runlevel と.target の対応

動時に何も指定しない場合には default.target が実行されます。手元の環境では default.target は graphical.target への symbolic link になっていました.

```
% ls -la /lib/systemd/system/default.target
lrwxrwxrwx 1 root root 16 2014-04-27 19:43 /lib/systemd/system/default.target -> graphical.target
```

では graphical.target の中身を見てみましょう

```
% cat /lib/systemd/system/graphical.target | grep -v ^#

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
After=multi-user.target
Conflicts=rescue.target
Wants=display-manager.service
AllowIsolate=yes

[Install]
Alias=default.target
```

「Requires」、「After」、「Conflicts」、「Wants」が依存/順序関係を表現しています。それぞれ

- Require: ここで指定されているユニットが起動していることが必要 (依存関係)。Require で指定されたユニットの起動に失敗すると、このユニットは実行されない。
- Wants: ここで指定されているユニットが起動していることが求められる (依存関係)。ただし、Wants で指定されたユニットの起動に失敗しても、このユニットは実行開始される。
- After: ここで指定されたユニットの起動「後」に実行される (順序関係)。
- Before: ここで指定されたユニットの起動「前」に実行される (順序関係)。

となっています。

ユニットの操作は全て systemctl コマンドで行ないます。では、現状の依存/順序関係を表示してみましょう。

[*8] 厳密に 1 対 1 に対応する訳ではない。

```
  % systemctl list-dependencies [unit 名]
    unit 名省略時は default.target
  % systemctl list-dependencies [unit 名] --after
  % systemctl list-dependencies [unit 名] --before
```

```
% systemctl list-dependencies
default.target
├── bootlogs.service
├── chrony.service
├── dovecot.service
├── exim4.service
├── hyperestraier.service
├── lightdm.service
├── motd.service
├── psd.service
├── pulseaudio.service
├── schroot.service
├── systemd-update-utmp-runlevel.service
├── yaskkserv.service
└── multi-user.target
  ├── anacron.service
  ├── atd.service
  ...
```

個々の unit の状況は status で表示できます:

```
% systemctl status ssh
ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since 日 2014-06-22 14:17:07 JST; 25s ago
 Main PID: 9180 (sshd)
   CGroup: name=systemd:/system/ssh.service
           └── 9180 /usr/sbin/sshd -D

 6 月 22 14:17:07 daphne systemd[1]: Started OpenBSD Secure Shell server.
 6 月 22 14:17:07 daphne sshd[9180]: Server listening on 0.0.0.0 port 22.
```

個々の unit の開始/停止/再起動はそれぞれ start/stop/restart で可能です.

```
% sudo systemctl stop ssh
% systemctl status ssh
ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: inactive (dead) since 日 2014-06-22 14:18:27 JST; 40s ago
  Process: 9180 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited, status=0/SUCCESS)

 6 月 22 14:17:07 daphne systemd[1]: Started OpenBSD Secure Shell server.
 6 月 22 14:17:07 daphne sshd[9180]: Server listening on 0.0.0.0 port 22.
 6 月 22 14:18:27 daphne systemd[1]: Stopping OpenBSD Secure Shell server...
 6 月 22 14:18:27 daphne systemd[1]: Stopped OpenBSD Secure Shell server.
% sudo systemctl start ssh
% systemctl status ssh
ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since 日 2014-06-22 14:19:52 JST; 13s ago
 Main PID: 10819 (sshd)
   CGroup: name=systemd:/system/ssh.service
           └── 10819 /usr/sbin/sshd -D
```

といった塩梅ですね。また、ユニットファイルを編集した場合には reload で unit を再起動できます。

現状のユニットの状況を見てみましょう。インストールされているユニットの一覧は list-unit-files で表示できます.

```
% systemctl list-unit-files
UNIT FILE                                   STATE
proc-sys-fs-binfmt_misc.automount           static
dev-hugepages.mount                         static
dev-mqueue.mount                            static
proc-sys-fs-binfmt_misc.mount               static
run-lock.mount                              static
run-user.mount                              static
sys-fs-fuse-connections.mount               static
sys-kernel-config.mount                     static
sys-kernel-debug.mount                      static
tmp.mount                                   disabled
cups.path                                   enabled
systemd-ask-password-console.path           static
systemd-ask-password-wall.path              static
acpid.service                               disabled
...
```

また、現在実行されたユニットの一覧は systemctl をオプション無し実行するか、list-units を実行します。type を指定してユニットを表示することも可能です。

```
% systemctl list-units
UNIT                                                                     LOAD   ACTIVE SUB       DESCRIPTION
proc-sys-fs-binfmt_misc.automount                                        loaded active waiting   Arbitrary E...
sys-devices-pci0000:00-0000:00:03.0-sound-card0.device                   loaded active plugged   /sys/device...
sys-devices-pci0000:00-00...4.0-usb1-1\x2d4-1\x2d4:1.0-bluetooth-hci0.device loaded active plugged   /sys/device...
sys-devices-pci0000:00-0000:00:16.3-tty-ttyS0.device                     loaded active plugged   Lynx Point-...
sys-devices-pci0000:00-0000:00:19.0-net-eth0.device                      loaded active plugged   Ethernet Co...
sys-devices-pci0000:00-0000:00:1b.0-sound-card1.device                   loaded active plugged   Lynx Point-...
sys-devices-pci0000:00-0000:00:1c.2-0000:02:00.0-net-wlan0.device        loaded active plugged   Dual Band W...
...
% systemctl list-units --type=socket
UNIT                          LOAD   ACTIVE SUB       DESCRIPTION
acpid.socket                  loaded active listening ACPID Listen Socket
avahi-daemon.socket           loaded active listening Avahi mDNS/DNS-SD Stack Activation Socket
cups.socket                   loaded active listening CUPS Printing Service Sockets
dbus.socket                   loaded active running   D-Bus System Message Bus Socket
lvm2-lvmetad.socket           loaded active listening LVM2 metadata daemon socket
syslog.socket                 loaded active running   Syslog Socket
systemd-initctl.socket        loaded active listening /dev/initctl Compatibility Named Pipe
systemd-journald.socket       loaded active running   Journal Socket
systemd-shutdownd.socket      loaded active listening Delayed Shutdown Socket
systemd-udevd-control.socket loaded active running   udev Control Socket
systemd-udevd-kernel.socket  loaded active running   udev Kernel Socket
virtlockd.socket              loaded active listening Virtual machine lock manager socket
```

3.4.2 で、dnsmasq は?

起動途中の tree は以下の通りです:

```
  ...
  811 ?        Ss     0:00 /bin/sh /etc/init.d/dnsmasq systemd-start-resolvconf
  821 ?        S      0:00  \_ run-parts --arg=-a --arg=lo.dnsmasq /etc/resolvconf/update.d
  863 ?        S      0:00      \_ run-parts /etc/resolvconf/update-libc.d
  901 ?        S      0:00          \_ /bin/sh /etc/resolvconf/update-libc.d/squid3
  902 ?        S      0:00              \_ /bin/sh /usr/sbin/invoke-rc.d squid3 reload
  936 ?        S      0:00                  \_ systemctl reload squid3.service
  ...
```

さて...

```
% cat /etc/resolvconf/update-libc.d/squid3
#!/bin/sh

PATH="/usr/sbin:/usr/bin:/sbin:/bin"

# Make squid aware of changes to resolv.conf
invoke-rc.d squid3 reload || true
```

`invoke-rc.d` の呼び出しは `systemctl` に行っているわけですが、ここで止まっているように見えます。dnsmasq, resolvconf, squid3 の状況を見てみましょう。

```
% systemctl status dnsmasq
dnsmasq.service - A lightweight DHCP and caching DNS server
   Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled)
  Drop-In: /run/systemd/generator/dnsmasq.service.d
           └─50-dnsmasq-$named.conf, 50-insserv.conf-$named.conf
  ...
```

ユニットファイルの中身は以下:

```
% cat /lib/systemd/system/dnsmasq.service | grep -v ^# | sed '/^$/d'
[Unit]
Description=A lightweight DHCP and caching DNS server
[Service]
Type=dbus
BusName=uk.org.thekelleys.dnsmasq
ExecStartPre=/usr/sbin/dnsmasq --test
ExecStart=/etc/init.d/dnsmasq systemd-exec
ExecStartPost=/etc/init.d/dnsmasq systemd-start-resolvconf
ExecStop=/etc/init.d/dnsmasq systemd-stop-resolvconf
ExecReload=/bin/kill -HUP $MAINPID
[Install]
WantedBy=multi-user.target
```

dnsmasq は dbus 経由で起動されているようです。

```
% systemctl status resovconf
resolvconf.service - Nameserver information manager
   Loaded: loaded (/lib/systemd/system/resolvconf.service; enabled)
   ...
```

ユニットファイルの中身は以下の通り:

```
[Unit]
Description=Nameserver information manager
Documentation=man:resolvconf(8)
DefaultDependencies=no
[Service]
RemainAfterExit=yes
ExecStartPre=/bin/mkdir -p /run/resolvconf/interface
ExecStartPre=/bin/touch /run/resolvconf/postponed-update
ExecStart=/sbin/resolvconf --enable-updates
ExecStop=/sbin/resolvconf --disable-updates
[Install]
WantedBy=network.target
```

resolvconf は network.target で呼び出されているのでネットワークの状況が変わる度に resolvconf が呼び出されます。

そして squid3 の起動は

```
% systemctl status squid3
squid3.service - LSB: Squid HTTP Proxy version 3.x
   Loaded: loaded (/etc/init.d/squid3)
   ...
```

squid3 は systemd 用の service が提供されていませんので、multi-user.target においてこれまで通り `/etc/init.d/squid3` が呼び出されます。結果として、

1. squid3 の起動を試みる
2. network の状態が変更されたので、resolvconf.service が呼び出される
3. resolvconf.service において`/etc/resolvconf/update-libc.d/squid3` が呼び出され、その中で squid3 が reload される
4. (1) に戻る

となって timeout まで止まっている様です。

```
% cat /etc/resolvconf/update-libc.d/squid3
#!/bin/sh

PATH="/usr/sbin:/usr/bin:/sbin:/bin"

# Make squid aware of changes to resolv.conf
# invoke-rc.d squid3 reload || true
```

とした所

```
% systemd-analyze
Startup finished in 1.646s (kernel) + 3.396s (userspace) = 5.042s
```

となりました (やったね)。…さて、これはどうしたら良いのでしょうか?

3.5 まとまりませんが

というわけで、次回 (があったら)、こういった「既存のサービス」の systemd への移行や、jounrnald、timedatectl、systemd-cron あたりについてまとめてみたいと思います。

4 Debian x LibreOffice

野島 貴英

4.1 はじめに

今回の東京エリア Debian 勉強会は、関東 LibreOffice さんとコラボで開くという初の試みを行います。ここでは、Debian から見た LibreOffice に関して、Debian の開発に関わる点についていくつか述べてみたいと思います。

4.2 Debian と upstream コミュニティの関係

Debian は、Linux OS のディストリビューションでも、世界一コミュニティ主導で開発が行われるコミュニティです。ここで、Debian Project の文化として、Upstream First という考え方があり、Debian で、特定のソフトウェアで生じた問題などは、最初に upstream へフィードバックすることが求められます。当然ですが、upstream の意向も汲む必要がありますし、upstream とは良好な関係を築き、建設的な情報交換を行う事が、Debian Project の方針として定まっています。

4.3 LibreOffice のバージョン

2014/10/25 現在のリポジトリを確認すると、LibreOffice のパッケージは以下のバージョンとなっています。

Debian	Debian アーキテクチャ	LibreOffice バージョン	備考
squeeze-backports	amd64 armel i386 kfreebsd-i386 mips mipsel powerpc s390 sparc	3.5.4	
	ia64 kfreebsd-amd64	3.3.3	
wheezy(stable)	amd64 armel armhf i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips mipsel powerpc s390 s390x sparc	3.5.4	
wheezy-backports	amd64 armel armhf i386 ia64 mipsel powerpc s390 s390x	4.3.2	
	sparc	4.2.6	
	mips	4.2.5	
	kfreebsd-(amd64—i386)	4.0.3	

表 2　Debian のバージョンと LibreOffice のバージョン (2014/10/25)

Debian	Debian アーキテクチャ	LibreOffice バージョン	備考
squeeze-backports	amd64 armel i386 kfreebsd-i386 mips mipsel powerpc s390 sparc	3.5.4	
	ia64 kfreebsd-amd64	3.3.3	
wheezy(stable)	amd64 armel armhf i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips mipsel powerpc s390 s390x sparc	3.5.4	
wheezy-backports	amd64 armel armhf i386 ia64 mipsel powerpc s390 s390x	4.3.2	
	sparc	4.2.6	
	mips	4.2.5	
	kfreebsd-(amd64|i386)	4.0.3	
jessie(testing)	amd64 arm64 armel armhf i386 kfreebsd-amd64 kfreebsd-i386 mips mipsel powerpc ppc64el s390x	4.3.2	
sid(unstable)	amd64 armel armhf i386 kfreebsd-amd64 kfreebsd-i386 mipsel powerpc ppc64el s390x sparc	4.3.3~rc2	
	arm64 mips alpha ppc64	4.3.2	
	hppa	4.1.6~rc2	
	powerpcspe	4.1.4	
experimental	amd64 armhf i386 ppc64el	4.4.0~alpha1	

表 3　Debian のバージョンと LibreOffice のバージョン (2014/10/25)

4.4　使う側の Debian 固有の事情の情報源

Debian で LibreOffice を使う場合、Debian 固有で生じる事も知っておいた方が良いです。こちらの情報源についていくつか紹介しておきます。

種別	場所	備考
ファイル	/usr/share/doc/libreoffice/ README.Debian	
wiki	`https://wiki.debian.org/LibreOffice`	
bug レポート	`https://bugs.debian.org/libreoffice`	

表 4　Debian 固有の LibreOffice 情報

4.5　Debian での LibreOffice のパッケージ開発についていくつか

東京エリア Debian 勉強会は開発者の集まりですので、ここではパッケージ開発固有の事情に関していくつか紹介します。

4.5.1　メンテナ

Debian での LibreOffice のパッケージ開発は、Debian LibreOffice Maintainers (debian-openofficelists.debian.org) というコミュニティが用意されています。ただ、実際に ML を見るとわかるのですが、Debian については、ほぼ Rene Engelhard さんにより精力的にメンテナンスが行われている状態です。

4.5.2　パッケージのソースコード

Debian のパッケージの開発の git リポジトリは、`http://anonscm.debian.org/cgit/pkg-openoffice/libreoffice.git` で管理されています。見ると判りますが、各ブランチが切られており、ubuntu 用なども見えます。

また、apt-get source libreoffice すると、debian/README.debian-source が入っています。こちらを読むと判

りますが、

- バージョンアップしたパッケージをどう作るかについて、工数が少なくなるような、一定のやり方を定めています。debian/rules ファイルも、非常にうまく作られていて、このやり方で多くの事が出来るようになっています。
- LibreOffice 自体巨大なソースですので、必然的に debian/control ファイルも巨大化します。こちらをメンテしやすくするため、パッケージの種別毎に contorl ファイルが多数分割されており、debian/rules で control ファイルを生成できるようになっています。
- debian/*.in ファイルを使って、パッケージ内部に梱包する様々な設定ファイルを各 Debian の環境に合わせて修正・合成しています。
- dpkg-dev で梱包されている/usr/share/dpkg/*.mk ファイルに基づき、configure に必要なオプションが作られていきます。
- DFSG Free にこだわる必要があるため、openoffice 由来のソースから、DFSG Free でないものを抹消しています。

4.5.3 Debian 固有のパッチの紹介

Debian 固有の事情に合わせてパッチが debian/patch 以下に収められています。Debian だけにとどまらない内容のパッチも若干含まれていますが、いずれ（あるいはすでに）upstream へフィードバックされる予定のはずです。

いくつもパッチがありますが、表 5 に Debian 固有で面白いものを抜き出して列挙してみます (用語を表 6 に載せます。)

4.6 おわりに

ここでは、LibreOffice が Debian ではどうパッケージ化されているかについて説明してみました。Debian では、arm64 アーキテクチャ対応が目玉となっていますが、LibreOffice の Debian パッチとして独自で搭載していたりします。こういった Debian 側からの改造が upstream に取り込まれ、upstream も対応アーキテクチャが増えていくのは良いことだと思います。同じ FOSS の仲間である Debian と LibreOffice が相互に発展していく感じがソースからも読み取れて感慨深いです。

パッチ名	内容	備考
aarch64.diff	arm64 用のパッチ。uno を取うための arm64 用のコードが追加されている。	
aotcompile-256M-default.diff	MAX_CLASSES_PER_JAR = 256,MAX_BYTES_PER_JAR = 262144 をビルドマシンの搭載メモリサイズによらず固定にするパッチ。	
config-sub-guess-update.diff	システムが glibc/ulibc/dietlibc であるかを検出して適切な LIBC 変数を設定する。さらに、古いシステム（Next、hp300、386BSD 等...) の判定コードを削除。	注：debian に関わるもののみ説明
debian-hardened-buildflags-CPPFLAGS.diff	Jessie リリースゴールの 1 つである hardening を搭載する。	
sensible-lomua.diff	Debian の持つ各種デスクトップ環境に合わせるためのパッチ。MUA を適切に設定。	
split-evoab.diff	gnome の MUA である evolution のアドレス帳連携のドライバで EVOAB2 を有効にするパッチ。	
earch-usr-share-for-images.diff	画像のサーチパスの検索順番を debian にあわせるべく変更（プログラムの改造含む)。	
help-msg-add-package-info.diff	ヘルプファイルがインストールされていない場合に出るエラーメッセージに、libreoffice-help-en-us パッケージ、libreoffice-help-¡language-code¿パッケージを導入して欲しい旨表示するようにするパッチ	
debian-debug.diff	-g1 を gcc のデバッグオプションとして利用し、デバッグシンボルファイルのサイズを減らす。	
gcj-safe-jni-h-include.diff	gcj で、jni.h がシステムにあわせて include されるように変更。	
mention-java-common-package.diff	Java に関する問題があった場合、libreoffice-java-common パッケージを入れるように促すメッセージを追加。	
jurt-soffice-location.diff	soffice を指定された場合に、debian でのインストール位置を正確に返却	
reportdesign-mention-package.diff	Oracle Report Builder の機能が必要な場合、libreoffice-report-builder を入れてくれと促す。	
system-coinmp.diff	coinmp に対応する。	

表 5　Debian 固有のパッチの中身

用語	説明
uno	Universal Network Object。CORBA とか COM(DCOM) 等のオブジェクト指向の通信により、LibreOffice の API 呼び出しを実現する仕組み。`https://wiki.openoffice.org/wiki/Documentation/DevGuide/ProUNO/Introduction`
hardening	コンパイラにセキュリティ強化の施策を打たせる事。`https://wiki.debian.org/Hardening`
coinmp	COmputational Infrastructire for Operations Research(CON-OR) のライブラリ。OR 用途。`https://projects.coin-or.org/`

表 6　用語

5 Debian からみた Arch Linux

野島 貴英

5.1 はじめに

OSC[*9]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、Arch Linux 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。

5.2 Arch Linux とは

Arch Linux は、i686/x86_64 で使える Linux ディストリビューションの 1 つであり、単純さ、小ささ、Arch Linux 特有の機能は簡潔なコードで維持するというのを徹底して目指しているディストリビューションです [1]。なお、これらのポリシーは、The Arch Way[2] という開発ポリシーに明記されています。

公式のリリースは、いわゆるローリング・リリースを採用しています。さらに、AUR(Arch User Repository) というユーザ同士でパッケージの build に必要なファイルを登録して公開できるリポジトリが用意されており、公式リポジトリに含まれていないようなソフトウェアはこちらからインストールすることが出来ます。

2001 年に Judd Vinet さんにより開発が開始され、2002 年 3 月 11 日に最初の公式リリースである Arch Linux 0.1 がリリースされました。2007 年後半には、開発リーダは Aaron Griffin さんへ引き継がれたようです [3]。

5.3 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 Arch Linux をインストールしてみます。なお、Arch Linux の ISO イメージは、基本的に Live イメージであり、Debian でいうところのインストーラというプログラムが公式にはありません。一旦 Arch Linux の ISO イメージで起動したら、ネットワークの有効化、インストール先のディスクのパーティションテーブル作成、フォーマット、ベースシステムインストール、root ユーザ設定、ブートローダインストールを

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

全て手動で行う必要があります。

Step 1. Debian マシンに KVM/libvirt/virtinst をインストールします [5]。

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

Step 2. Arch Linux の iso イメージを入手します。基本的には Arch Linux の Download のページから辿れるミラーサイトにある、archlinux-YYYY.MM.DD-dual.iso を入手する事になります。以下の例は jaist から 2014.11.01 版（2014/11/25 にて最新) を入手する例となります。

```
$ mkdir arch-linux
$ cd arch-linux
$ wget http://ftp.jaist.ac.jp/pub/Linux/ArchLinux/iso/2014.11.01/archlinux-2014.11.01-dual.iso
```

Step 3. 仮想ディスク (5GBytes) を作成し、virt-install コマンドを使って起動します。

```
$ sudo qemu-img create -f raw /var/lib/libvirt/images/arch-01 5G
$ sudo virt-install --connect=qemu:///system -n arch-01 --ram 512 \
     --cdrom /home/yours/arch-linux/archlinux-2014.11.01-dual.iso \
     --disk /var/lib/libvirt/images/arch-01,bus=virtio,size=7,format=raw,cache=writeback \
     --vnc --hvm --accelerate
```

Step 4. virt-viewer が立ち上がり、Arch Linux の起動メニューが表示されます。Installation Guide (日本語)[4] を見ながらインストール作業を進めて下さい。(具体的な手順は長いので割愛。ほぼ手動で作業を行う必要あり。)

Step 5. ブートローダをインストールし、インストール先のディスクをアンマウントしたら、reboot コマンドを打ち込んでリブートして下さい。

Step 6. 無事 Arch Linux がディスクイメージから立ち上がり、login プロンプトが表示されれば一旦完了です。Step 4. の手順の途中で作成した root ユーザでログインし、必要に合わせて追加のパッケージを導入したり、一般ユーザを作ったりして、カスタマイズを進めて下さい。

5.4 Debian との違い

Arch Linux の公式 wiki に、Debian を含む他のディストリビューションとの違いについて記載があります [6]。ここでは、表 7 に、Debian との違いを記載してみます。

さらに Debian と Arch Linux について図 1、図 2 に示します。

項番	項目	Debian	Arch Linux	備考
1	基本方針	Debian 社会規約、DFSG	The Arch Way	
2	自由ソフトウェアへのこだわり	こだわる	あまり気にしない	
3	パッケージ管理	apt,aptitude,dpkg	pacman,yaourt	
4	リポジトリ	backports/old-stable/stable/testing/unstable/experimental, main/contrib/non-free/updates	official repository/AUR, core/extra/community/multil testing/community-testing/multilib-testing	
5	開発者	DD,DM	Developer,TU	
6	ネットワーク設定	/etc/network/interface ファイル	/etc/netctl/以下のファイル群	
7	ネットワーク設定コマンド	ifup/ifdown	netctl など	
8	パッケージ開発コマンド	dpkg-buildpackage/debuild	makepkg	
9	パッケージ	deb ファイル	pkg.tar.xz ファイル	
10	パッケージデータ形式	ar アーカイブ +tar+(gzip,xz)	tar+xz	
11	パッケージ作成用ファイル	debian/rules,debian/control, debian/changelog ファイル等	PKGFILE ファイル	
12	パッケージ作成用ファイル	各種ステージ用の定義ファイル (仕様は debian-policy マニュアルに記載。)	bash シェルスクリプト	
13	パッチの方針	Community/パッケージ開発者の方針・議論で決まる	upstream の設計をほとんど変えない	
14	設定ファイルのポリシー	インストールしたらほぼそのまま動くまで設定済	upstream のものがほとんどそのままで使われるため、場合によっては環境に合わせて Arch Linux の wiki を見ながら逐一設定する事が必要になる場合がある。	
15	パッケージ開発難易度	Arch Linux に比べたら若干複雑	容易 (upstream の想定しているインストールの仕方を変更しないのが方針の為)	
16	パッケージ依存	パッケージ構築時に依存するもの、バイナリ側で依存するものが異なる (-dev パッケージ等)	基本的にパッケージ構築時に依存＝バイナリ側で依存 (-dev パッケージがない)	
17	Debug シンボルパッケージの有無	一部有り	無し	
18	公式サポート OS/公式アーキテクチャ	Linux/kFreeBSD,x86/amd64/armel/armhf/powerpc/s390x/mips/...	Linux,x86/x86_64 のみ	kFreeBSD は Jessie には落ちた
19	パッケージ数	unstable: 42,810 (11/28 調べ。dbg パッケージは除く)	official: 11,707,AUR: 52,644	
20	upstream の最新版にどれだけ近いか	unstable: 近いものが多い。stable: 最大 1 ～ 2 年遅れ	近いものが多い	

表 7 Debian と Arch Linux の違い

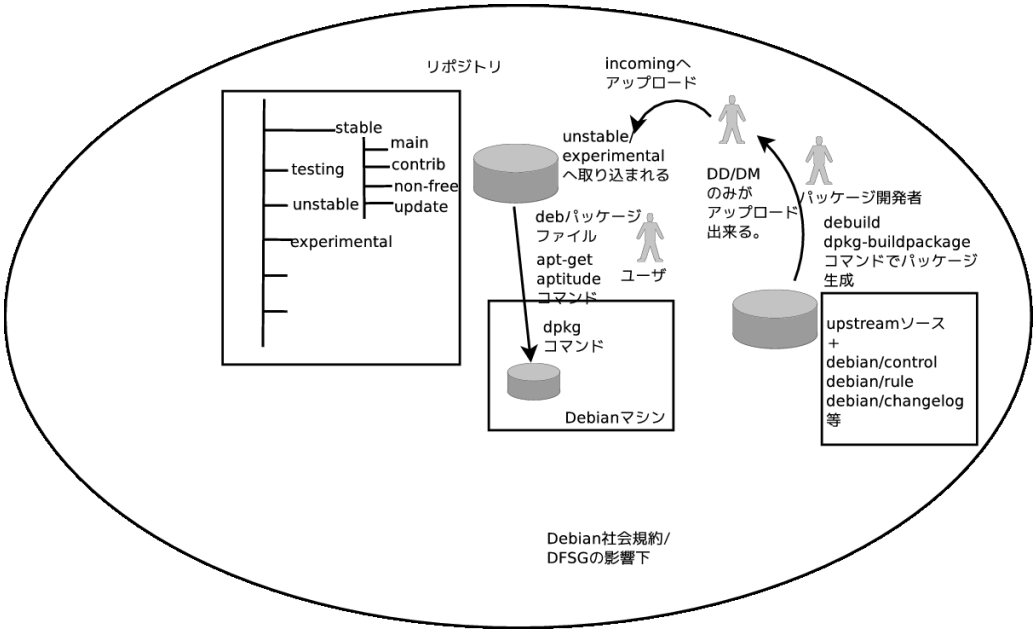


図 1　Debian

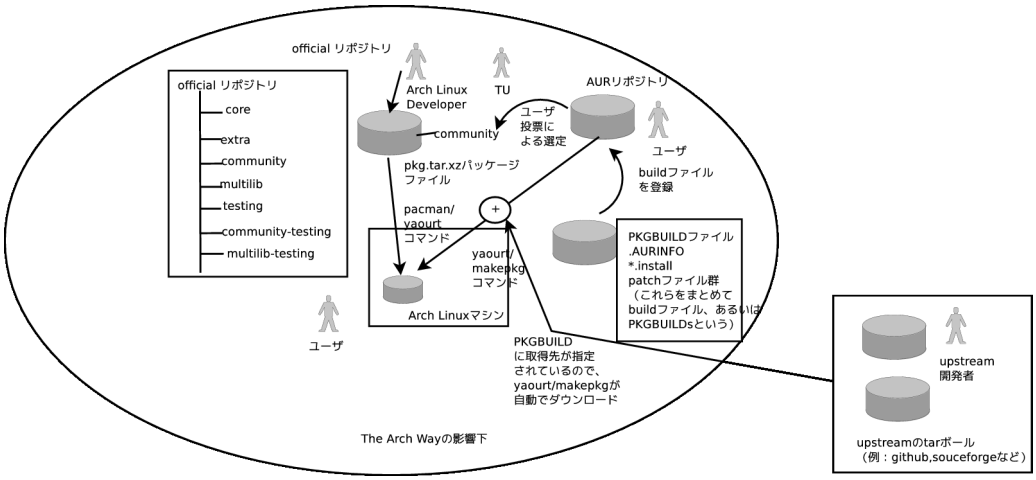


図 2　Arch Linux

5.5　AUR とは

Arch Linux には、official リポジトリに含まれないソフトウェアについて、ユーザが PKGFILE 等の build ファイル一式をアップロードして他のユーザと共有して使う Arch User Repository(AUR) というリポジトリが用意されています。AUR から入手できる yaourt（ヨーグルトと読む）コマンドを使うと、AUR をあたかも pacman で扱ったかのように便利に使う事ができます。

AUR は、`https://aur.archlinux.org/` にて、アカウントを取得さえすれば、build ファイルを登録して公開できますので、非常に手軽に、新しいソフトウェア用の build ファイルを他ユーザと共有して使うことができます。

なお、AUR はその使われ方から、登録された build ファイルは誰も精査していない場合があるため、基本的には自己責任 (AT YOUR OWN RISK) での活用が求められます。

AUR に登録された build ファイルはユーザの投票により、一定量の支持が得られると、Trusted User(TU) らにより、official リポジトリの community リポジトリに取り込まれる仕組みのようです。

5.6 Arch Linux の良さ

使ってみてわかった Arch Linux の良さを列挙します。

- The Arch Way に記載されているとおり、Arch Linux を構成するあらゆるソフトウェアは最小限主義に貫かれています。従って、upstream のソフトウェアとの変更点も最小としている為、設定ファイルの見通しが非常に良いです。
- ユーザフレンドリということには重おきをおかず、ユーザの嗜好を極力邪魔しない (User Centerized) 事をモットーとしているため、ユーザ自身が欲しいソフトウェアだけを導入という事がやりやすい作りになっています。
- AUR のように、利用者が自由に build ファイルを登録できる仕組みがあるため、気軽にパッケージを公開することが出来ます。また、upstream との変更点を最小限に保つ方針のため、パッケージ化にかかる労力が少なくて済み、upstream 側のリリースにあわせてスピーディーにパッケージ側のバージョンを追従させるが可能です。
- 軽量です。パッケージを厳選して導入する事がしやすいため、マシンのスペックが低くても問題になりにくいです。

5.7 終わりに

今回は Arch Linux と Debian を比較してみました。Arch Linux はシンプルかつ最小限をモットーとしており、ソフトウェアの個別設定を自力で行う必要があります。そのため、Linux システムの勉強を熱心にしたい人、あるいは、導入するソフトウェアに強いこだわりがある人は、うってつけのシステムかと思います。

その一方で、最初からある程度便利に使えるようにいろいろと自動で設定が行われ、マシン資源を消費するものの多くの便利なパッケージをある程度の量勝手に導入しておいてくれることを期待する人向けには、Debian の方がよくできています。

Debian の良い所を正確に知る、あるいは目指すべき方向性の確認には、他のディストリビューションと比較することも重要かと思います。機会があれば、他のディストリビューションも使ってみて、Debian との比較を行うのもよいのではないでしょうか。

参考文献

[1] Arch Linux (日本語), https://wiki.archlinux.org/index.php/Arch_Linux_%28%E6%97%A5%E6%9C%AC%E8%AA%9E%29

[2] The Arch Way (日本語), https://wiki.archlinux.org/index.php/The_Arch_Way_(%E6%97%A5%E6%9C%AC%E8%AA%9E)

[3] History of Arch Linux (日本語), https://wiki.archlinux.org/index.php/History_of_Arch_Linux_(%E6%97%A5%E6%9C%AC%E8%AA%9E)

[4] Installation Guide (日本語), https://wiki.archlinux.org/index.php/Installation_Guide_(%E6%97%A5%E6%9C%AC%E8%AA%9E)

[5] Debian wiki の KVM の章, https://wiki.debian.org/KVM

[6] Arch Compared to Other Distributions (日本語), https://wiki.archlinux.org/index.php/Arch_Compared_to_Other_Distributions_(%E6%97%A5%E6%9C%AC%E8%AA%9E)

6 Debian でタイルマップサービス作ってみた

なかおけいすけ

6.1 タイルマップサービスとは

タイルマップサービス (TMS: Tile Map Service) は、OSGeo 財団 (Open Source Geospatial Foundation) が策定した、地図をタイルとして提供するプロトコルです。タイルは、地図を領域とズームレベル毎に 256px × 256px の画像に分割したもので、図 3 のようにピラミッド構造になっています。大きな画像を小分割することで、必要な領域、ズームレベルだけを取得し、不要になった部分を開放することで、メモリの消費量を削減し、効率良く通信を行うことができます。タイルは、ズームレベルが一つ大きくなると、表示する範囲が 1/4 になります。例えば、ズームレベル 0 では、地球全体を 1 枚のタイルで表し、ズームレベル 1 では 4 枚、ズームレベル 2 では 16 枚.... ズームレベル n では $2^n \times 2^n$ 枚になります。ズームレベル 16 になると、$2^{32} = 4,294,967,296 \simeq 4.3 \times 10^9$ 枚ものタイルが必要になります。

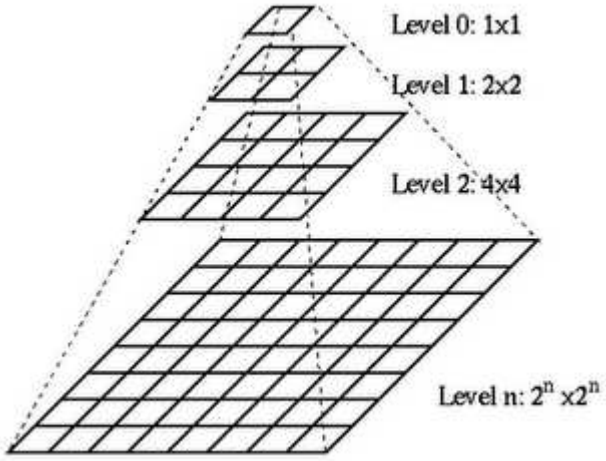


図 3　タイルのピラミッド構造 [1]

タイルは、HTTP で、`http://BASEURL/VERSION/TILENAME/z/x/y.png` という形式の URI で取得できます。現在、VERSION は 1.0.0 です。ここで z はズームレベル、x、y は領域を表す整数です。x、y は、lat、lon を緯度、経度として以下のように書けます [2]。

$$x = \frac{2^z(lon + 180)}{360}$$

$$y = 2^{z-1}(1 - \frac{\ln(\tan(lat\frac{\pi}{180})) + \frac{1}{\cos(lat\frac{\pi}{180})}}{\pi})$$

6.2　タイルの生成

タイルを生成するためには、表示するデータが必要です。今回は JAXA の水循環変動観測衛星「しずく」に搭載されている AMSR2 というセンサーで観測された海水表面温度データをタイルにしてみましょう。

「しずく」のデータは、登録が必要ですが一般公開されています [3]。登録をすると、sftp でデータをダウンロードできます。

```
$ sftp -oPort=2051 username@gcom-w1.jaxa.jp
sftp> get AMSR2/YYYY/YYYY.MM/L3/SST_10/1/GW1AM2_YYYYMMDD_01D_EQOD_L3SGSSTHA1100100.h5.gz
```

ダウンロードしたファイルは、gzip で圧縮された HDF5 というファイルフォーマットの数値データです。HDF5 は科学技術用に開発された数値データを格納するためのバイナリーフォーマットで多次元の数値データだけでなく、観測日時や観測者、スケーリングファクタ、解析アルゴリズムといった情報を階層構造で格納することができるフォーマットです。もちろん仕様は公開されており Python や Ruby のモジュールもあります。

この中に海水面温度の数値データが北緯 0 度、東経 0 度から 0.1 度刻みのメッシュで格納されています。まずこのメッシュ1 つを 1px とする画像を作ります。そうすると地球全体で 3600px×1800px の画像ができます。

私は Python を使って、メッシュのデータを取り出し、あらかじめ作っておいたカラーマップで数値を RGB に変換し、ASCII テキストでピクセルを表現できる PPM で出力しました。欠損値は (R,G,B)=(0,0,0) としています。

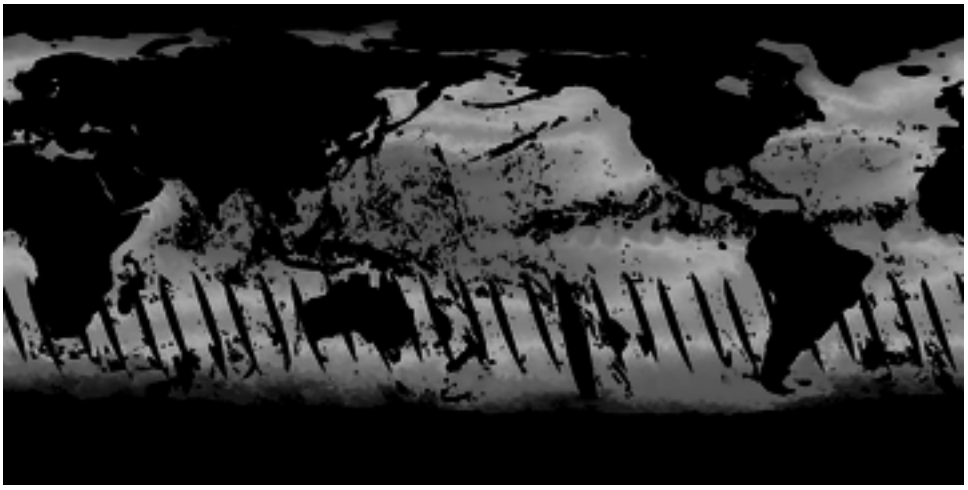


図 4　生成したマップ

この画像を ImageMagick の convert コマンドで tiff 画像にしつつ、欠損値を表す黒を透明に変換します。

```
$ convert -transparent black converted.ppm map.tiff
```

ここでできる Tiff 画像（図 4）は、地図ではありますが、ピクセルと緯度経度の対応関係の情報を持っていません。そこで GDAL を使って、この Tiff 画像を位置情報を持つ GeoTiff というフォーマットに変換します。GDAL は、地理空間データのスイスアーミーナイフ的な存在で、GeoTiff や JPG、PNG といったラスタデータフォーマットや HDF、NetCDF 等のファイルを読んで他のフォーマットに変換したり、測地系の変換等を行うことができるライブラリ、ユーティリティ群です。（ここまででやってきた、HDF5 ファイルから数値データを取り出してカラーマップにすることも、GDAL だけでできる気がします。）GDAL は、gdal-bin というパッケージになっています。Tiff ファイルを GeoTiff に変換するには、gdal_translate コマンドで、緯度経度が分かっているピクセルを教えてあげます。そして gdalwarp コマンドで、gdal_translate で指定したピクセルと緯度経度を元に幾何変換して測地系の情報を追加します。

```
$ gdal_translate -q -gcp 0 0 0 90 \
                -gcp 3600 0 360 90 \
                -gcp 0 1800 0 -90 \
                -gcp 3600 1800 360 -90 map.tiff tmp.tiff

$ gdalwarp -q -s_srs EPSG:4326 -r cubic tmp.tiff map.tiff
```

各ピクセルが位置情報を持った画像ファイルが完成しました！これでようやくタイルを生成する準備ができました。

タイルの生成には、python-gdal パッケージの gdal2tiles.py コマンドを使います。ただこのコマンドは、指定したズームレベルのタイルをすべて生成してしまいます。前述の通り、ズームレベルを増やすと指数関数的に生成する画像が増えていくのでタイルの生成に数時間かかってしまいます。数時間かけて苦労してタイルを作っても、現実にはユーザーがすべてのタイルを見てくれるわけではありません。だったら必要なタイルだけを、必要な時に生成すれば良いのです。

6.3 タイルマップサーバの実装

では、リクエストがあったときに必要に応じてタイルを生成するサーバーを作りましょう。今回は Web サーバは Apache で、mod_python と mod_rewrite を用います。mod_rewrite の説明は不要でしょう。正規表現にマッチしたリクエストの URL を書き換えるモジュールです。

mod_python は、Apache の Python binding で、CGI の置き換えを目的としたモジュールです。インストールするには、以下のコマンドを実行します。

```
# apt-get install libapache2-mod-python
# a2enmod python
# service apache2 restart
```

設定は、/etc/conf.d/ の中に適当なファイル名のファイルを作って

```
<Directory /some/path>
  SetHandler mod_python
  PythonHandler mod_python.publisher
</Directory>
```

としすればよいでしょう。

mod_python の mod_python.publisher は、HTTP リクエストを Python の関数の呼び出しに変換します。たとえば、www.example.org の DocumentRoot に、hello.py という以下の Python スクリプトがあったとしましょう。

```
def sayHello(req, name):
  return ‘‘Hello %s’’%name
```

もしも、`http://www.example.org/hello.py/sayHello?name=Debian` に GET リクエストがあると、Web サーバは、sayHello 関数の name 引数に Debian を代入して sayHello 関数を呼び出し、”Hello Debian” と表示されます。

タイルマップサービスにアクセスするクライアントは、`http://BASEURL/VERSION/TILENAME/z/x/y.png` という URI で GET しようとするので、mod_rewrite で URI を書き換えましょう。たとえば、`www.example/org` の/etc/apache/conf.d に以下のように設定すると、

```
<Directory "/var/www/tile/1.0.0/sst">
    RewriteEngine   On
    RewriteBase     /tile/1.0.0/sst/
    RewriteRule     ^([0-9]+)/([0-9]+)/([0-9]+).png    gettile.py/get?z=$1&x=$2&y=$3

    AddHandler  mod_python    .py
    PythonHandler   mod_python.publisher
</Directory>
```

`http://www.example.org/tile/1.0.0/sst/4/12/34.png` が 、`http://www.example.org/tile/1.0.0/sst/gettile.py/get?z=4&x=12&y=34` に変換されるので、gettile.py の中の、req, x, z, y の引数をとる関数が呼び出されます。ここに指定された URI のタイルだけを生成処理を書けば、オンデマンドにタイルを生成するタイルマップサービスができます。

単一タイルだけを生成するには、gdal2tiles.py を少し修正します。generate_base_tiles 関数の中の 1180 行目付近にすべてのタイルを作っているループが存在するので、

```
    def generate_base_tiles(self):
        """Generation of the base tiles (the lowest in the pyramid) directl y from the input raster"""

                            (Snip...)

        for ty in range(tmaxy, tminy-1, -1): #range(tminy, tmaxy+1):
            for tx in range(tminx, tmaxx+1):

                if self.stopped:
                    break
                ti += 1
                tilefilename = os.path.join(self.output, str(tz), str(tx), "%s.%s" % (ty, self.tileext))
                if self.options.verbose:
                    print(ti,'/',tcount, tilefilename) #, "( TileMapService     : z / x / y )"
```

それをひも解いて指定されたタイルを生成している部分を、関数として抜き出します。

```
    def generate_base_tiles(self):
        """Generation of the base tiles (the lowest in the pyramid) directl y from the input raster"""

                            (Snip...)

        for ty in range(tmaxy, tminy-1, -1): #range(tminy, tmaxy+1):
            for tx in range(tminx, tmaxx+1):
                generate_tile(tx, ty, self.tmaxz)

    # -------------------------------------------------------------------------
    def generate_tile(self, ty, tx, tz):
        ds = self.out_ds
        tilebands = self.dataBandsCount + 1
        querysize = self.querysize
```

あとは、generate_tile 関数を、mod_python が呼ぶスクリプトから呼べばできあがりです。

```
def get(req, z, tx, ty):
        req.content_type = 'image/png'
        #req.write('z: %s\ntx: %s\nty: %s'%(z,tx,ty))

        g = GDAL2Tiles(['/home/chome/public_html/tile/sst/map.tiff','/var/www/tile/1.0.0/sst'])
        g.open_input()
        g.generate_tile(int(ty),int(tx),int(z))

        with open('/var/www/tile/1.0.0/sst/%s/%s/%s.png'%(z,tx,ty), 'rb') as f:
                req.write(f.read())
```

6.4 まとめ

Debian でパッケージングされているオープンソースのプログラムやライブラリを組み合わせてタイルマップサービスを作ってみました。gdal2tiles.py を少し修正し mod_python から呼ぶことによって、オンデマンドで必要なタイルだけを生成しています。

今回開発したサービスは `http://www.hi-rezclimate.org` で公開されています。データの取得、GeoTiff にするところまでは自動化されているので、毎日最新のデータをダウンロードして自動で更新されています。「しずく」は海水面温度だけでなく、海上風速や水蒸気量、積雪深といった水に関する現象を観測しています。今後サポートするデータを増やしていく予定です。

参考文献

[1] “Tile Map Service in Geoide”, `http://geoikia.idgis.eu/wiki-english/index.php/Tile_Map_Services_in_Geoide`

[2] “Slippy map tilenames”, `http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames`

[3] GCOM-W1 データ提供サービス `https://gcom-w1.jaxa.jp/auth.html`

7 Linux のドライバメンテナになった体験記

坂本貴史

7.1 今日の話題

私は個人でサウンドデバイスドライバを書いています。そのコードが、サブシステム経由で、6 月頭に Linux 3.16-rc1 にマージされました。サブシステムから Linux にコードが pull される流れを、実体験を基にして簡単にお話します。

7.2 Linux の開発サイクル

「前の」、「今の」、「次の」バージョンと表現したとき、以下のようになります。

0 週　前のバージョンをリリース。今のバージョンのマージウィンドウ開始。
0〜2 週　この間、新機能が pull される。
2 週　マージウィンドウ閉鎖。今のバージョンの rc1 をリリース。
2 週〜(N-1) 週　この間、バグ修正が pull され、だいたい 1 週間おきに rc がリリースされる
(N-1) 週　今のバージョンの rc(N-2) をリリース
N 週　今のバージョンをリリース。次のバージョンのマージウィンドウ開始。

7.3 Linux のサブシステム

機能別にサブシステムに分割されており、それぞれのサブシステムを開発するプロジェクトがあります。

- Scheduler
- Memory management
- Networking
- Filesystems
- Read-Copy update (RCU)
- ...
- IEEE1394 bus
- Sound

7.4 Linux のサブシステムのメンテナ

- サブシステム内の開発をまとめる役割を果します
- 開発者から送られたパッチは、メンテナがマージするかどうかを決定します
- マージウィンドウが開いたら、メンテナが Linus に pull request を出します

- Linus が request を ack し、tree にマージします
- メンテナはあらかじめ、Linus との信頼関係を築いているため、拒否されることは珍しいようです

7.5 Linux の開発者の活動

- 典型的には、サブシステム開発プロジェクトのメーリングリストで活動します
- 自分が書いたパッチのマージに向け、メンテナや他の開発者を説得します
- RFC (Request for comment) を出し、他の開発者の反応を見ることが有効です
- 他の人が投げたパッチをレビューすると喜ばれます

7.6 私がやったこと

- サウンドサブシステム
- ALSA firewire stack の開発
- firewire のプロトコルスタック
- プロトコルスタックの実装
- ドライバの実装

7.7 開発の初期

2013 年 1 月〜2013 年 8 月
目標は、デバイス仕様を把握することと、自分の活動をアピールすることです。

- ライブラリモジュールの拡張
- ドライバの作成
- Linux firewire subsystem のバグ

7.8 開発の中期

2013 年 9 月〜2014 年 1 月
目標は、パッチセットの最終候補を作成し、テスターを募ることです。

- 別なデバイスチップの調査
- ドライバのブラッシュアップ
- 既存実装のリグレッションテスト
- 最終 RFC と CFT

7.9 開発の後期

2014 年 2 月〜2014 年 6 月
目標は、ALSA の上流にマージされるよう、コミュニケーションすることです。

- 影響のあるプロジェクトとの調整
- 3.14 のマージウィンドウの閉鎖
- マージリクエスト 1 〜 3
- 3.15 のマージウィンドウの閉鎖
- マージリクエスト 4

- メンテナの緩い同意
- topic/firewire ブランチ
- 0day kernel testing[*10] [*11]
- testing backend のレポートへの対応
- sound.git の for-next へ

7.10 マージ以降

2014 年 6 月
目標は、Linux へのマージを見届けることです。

- 3.16 のマージウィンドウオープン
- サブシステムメンテナから Linus へのプルリクエスト
- コードの軽微な修正
- サブシステムメンテナから Linus へのプルリクエスト
- 3.16 のマージウィンドウの閉鎖

7.11 まとめ

Linux とそのサブシステムの間で、開発サイクルがどのように進められるかを、実体験を踏まえて説明しました。

7.12 質問への回答

7.12.1 キャラクタデバイスとは何ですか？

キャラクターデバイスとは、ユーザープロセスが、カーネルランドにあるコードを利用するためのエントリーポイントの一種です。ユーザープロセスがエントリーポイントに対して特定の操作を行なうと、カーネルランドにあるコードが実行され、ハードウェアとデータが交換されます。

一般的に、オペレーティングシステムは、ハードウェアの違いを吸収するレイヤを設け、同じプログラムルーチンで、同じ機能を持つ異なるハードウェアを扱えるようにしています。これにより、異なるハードウェアに対して異なるソフトウェアを書く必要がなくなるわけです。古い文献では、この機能を仮想マシン (virtual machine) と呼びます[*12]。

この「仮想マシン」という文脈からキャラクタデバイスに言及すると、ハードウェアを抽象化してソフトウェアから利用できるようにしたもののひとつ、となります。

また一般的に、オペレーティングシステムは、複数のユーザープロセスが、擬似的に同時に走る機能を持ちます。これを Time Sharing System と呼びます。Time Sharing System は、ユーザープロセスがある条件を満たしたときあるいはシステムである事象が発生したとき、そのプロセスの実行を休止して状態を保存し、別なプロセスの状態を復元して実行を再開することで成立します。

Time Sharing System では、様々なプロセスが、同時にシステム資源へアクセスすることがあります。そのためオペレーティングシステムは、システム資源に対するアクセスを管理する機能を持ちます。オペレーティングシステムは、ユーザープロセスに対し、ハードウェアへのアクセス方法をエントリーポイントに限定することで、システム資源を管理します。このシステム資源管理方法は、最近のメニーコア (CPU を複数持つ) マシンでも役に立ちます。

[*10] 0day kernel testing back-end (2012/05/17) https://lkml.org/lkml/2012/5/17/126
[*11] KS2012: Kernel build/boot testing (2012/09/05) http://lwn.net/Articles/514278/
[*12] 「オペレーティング・システムの機能は、ユーザーに拡張マシン (extended machine) またはその下のハードウェアよりプログラムしやすい仮想マシン (virtual machine) を提供することであるといえる。」(ANDREW S. TANENBAUM (1989) 『MINIX オペレーティングシステム)』坂本文 監修、大西照代 翻訳、アスキー出版局)

Linux において、大抵のエントリーポイントはファイルの形となっています。これは Linux が、システムリソースをファイルで表現するという UNIX の設計思想に影響を受けているからです。キャラクタデバイスやブロックデバイスのエントリーポイントとなっているファイルを、特別に、「特殊ファイル (special file)」と呼びます。ユーザープロセスは、基本的に、特殊ファイルに対してファイル操作を行い、ハードウェアとのデータ交換を行います。なお、このデータ交換を入出力 (input/output = I/O) と呼びます。

キャラクタデバイスは、バイト単位で入出力可能なハードウェアを表現します。身近なところでは、マウスやキーボードはキャラクタデバイスとなります（最も、これらはシステムへの入力限定、つまりキャラクタデバイスからの読み出し限定ですが)。Linux にはこの他のエントリーポイントとして、ブロックデバイスがあります。固定サイズでデータを入出力可能なハードウェアを表現します。例えばハードディスクドライブが該当し、/dev/sda1 となります。

さて、ユーザープロセスが特殊ファイルに対してファイル操作を行うと、システムコールが呼ばれて CPU の動作モードがユーザーモードからカーネルモードに変わり、ハードウェアにアクセスできるようになります。この時に実行されるコードのうち、カーネルランドにあるものをドライバと呼びます。キャラクタデバイスに対するドライバはキャラクタデバイスドライバ、ブロックデバイスに対するドライバはブロックデバイスドライバとなります。

Linux のデバイスドライバにはこの他に、ネットワークデバイスドライバがあります。ネットワークデバイスドライバは他の 2 種類のドライバとは異なり、エントリーポイントがファイルではありません。ユーザープロセスは、専用の関数を実行し、ネットワークデバイスのエントリーポイントを取得します。このエントリーポイントに対する操作も、専用の関数となります。これら関数群をソケットインターフェイスと呼びます。

キャラクタデバイスおよび Linux における他の 2 種類のデバイスに関して、オペレーティングシステムの基本機能やユーザープロセスからの利用方法も踏まえて説明しました。

7.12.2 デバッグはどのように行いましたか？

デバッグですが、printk などのカーネル API でログを出力してカーネルスペースのデータ構造の状態を確認することと、簡単なプログラムを実行してデバイスの状態を確認することで行いました。便利なツールがたくさんありそうな気がしますが、私が気にしなければならないことに対して学習コストが高すぎると考えました。

私が書いたドライバの構造を簡単に述べると、下の方 (ボトムハーフ) は IEEE 1394 バス上のユニット (デバイス) と入出力を行い、上の方 (トップハーフ) は ALSA Core を使ってユーザープロセスと入出力を行います。ドライバの仕事は、ALSA Core がパターン化したユーザープロセスからの要求に合わせ、デバイスを操作し、特定の挙動をさせることです。ユニットに相当するスタブドライバを書くことも考えたのですが、Linux firewire subsystem が持つ OHCI1394 コントローラードライバをもうひとつ書くような大掛かりな仕事になるので、諦めました。

実はデバイスへの入出力は、ユーザースペースからも行うことができます。Linux firewire subsystem は、バス上にあるそれぞれのユニットに対して fw キャラクタデバイスを設け、libraw1394 を通じて API を提供しています。デバイスの状態は、この API を利用する簡単なプログラムを実行して確認することができました。デバイスの挙動の調査は、だいたいこのユーザープログラムから行いました。

デバイスの挙動が明確になりさえすれば、あとはそれを C 言語で書くだけです。git でコードを管理し、コードの修正がある度にテストデバイスすべてを使って挙動を確認し、printk のログ出力で構造体の状態を確認するという作業がデバッグの中心でした。コードの分量が、個人がその実行パスを把握できる程度 (10,000 行以下と言われています) だったのも幸いしました、

こうして振り返ってみると、kgdb を使ってカーネルスペースの構造体の確認をしたり、Linux firewire subsystem の学習のためにスタブドライバを書いたりするべきだったと思います。

8　GPG 秘密鍵取り扱い方法の提案

吉田 晋

8.1　はじめに

皆さん、GPG のキーサインパーティーに参加した事ありますか？私は、昨年の大統一 Debian 勉強会 2013 で初めて参加し、GPG の秘密鍵も、その時初めて作りました。ところで、この秘密鍵はどのように保管すれば良いのでしょうか？

鍵はいつでも再発行出来るとはいえ色々と手間がかかりますし、古い鍵で暗号化したファイルを複合化出来なくなると困ります。記録メディアの破損による鍵のロストには要注意です。一般的に、記録メディアの破損に備えるには複数のメディアを複数の場所で保存する事は良い方法です。

一方、最低限のマナーとして盗難にも気をつける必要が有るでしょう。厳重に管理するには、バックアップの数は少ないほど有利です。

さて、メディア破損に備えて冗長性を持たせる方法と盗難に備えて厳重に管理する方法は真っ向から矛盾してしまいました。結局、どうすれば良いのでしょう？

8.2　アイディアのベースは RAID

例えば 3 本のディスクを用いた RAID5 のディスクアレイでは、2 個のディスクにデータを分散して記録しながら残りの 1 個のディスクにパリティデータを保存します。同じ要領で秘密鍵を 2 個のファイル + 1 個のパリティファイル = 計 3 個のファイルを PC、USB メモリ（肌身離さず）、DVD（金庫）の 3 カ所で保存すれば、

- 普段は、USB メモリと PC を使って、都度鍵ファイルを復元して使用
  （使用後は鍵ファイル削除。削除忘れ防止のため、tmpfs 上で作ると良いかも。）
- PC や USB メモリが盗まれても、しばらくは安全
- ディスクのクラッシュや USB メモリの破損時には金庫の DVD を使って復元

という冗長性と秘匿性を兼ね備えた状態に成ります。完全な鍵を世界中のどこにも保存する必要が無いところがポイント高！

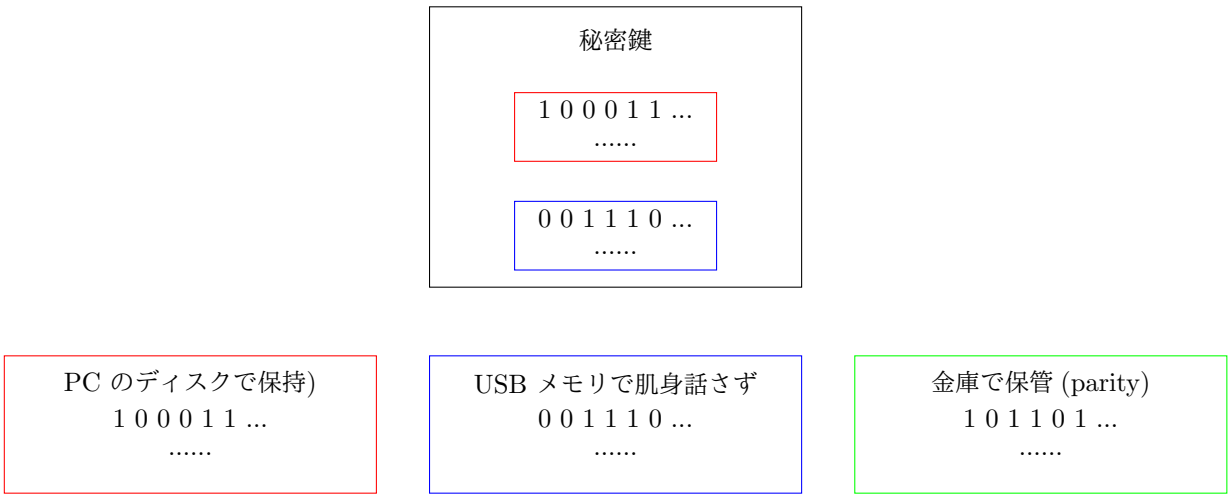


図 5　分散管理イメージ図

ちなみに、分散するファイルの数を 2 倍にしたらどうなるでしょう？秘密鍵を 4 個のファイル + 2 個 = 計 6 個のパリティファイルに分散すれば、パリティ増による冗長性の向上と、ファイル 1 個あたりの情報量減による秘匿性の向上を両立可能です。

8.3　プロトタイプ作ってみた

プロトタイプは Python 2.7 で作成し、 Debian Sid で動作確認を行いました。
dVault (distribute vault) と命名

`https://github.com/wbcchsyn/dVault_prototype`
（src/dvault が本体）

8.3.1　注意

あくまでプロトタイプなので、絶対に自分の秘密鍵に対して使わないで下さい。
例えば、

- 作成したファイルのパーミッションとか、全然気にしてません。
- 問答無用でファイルの上書きをします。
  秘密鍵を上書きしながら途中で失敗すると、秘密鍵が失われます。

8.3.2　使用方法

origin というファイルを split0, split1, parity の 3 個のファイルに分割する場合

```
$ dvault split -u origin -s split0 -s split1 -s parity
```

split0, parity の 2 個のファイルから origin を復活させる場合

```
$ dvault unite -u origin -s split0 -s split1 -s parity
```

8.3.3 仕様

- パリティは 1 個で固定
- 分散した各ファイルの最初にはメタ情報を json で保存。
  その後に 1 行空行を入れて、実際のデータを記入
  とりあえず、メタ情報は以下の 3 個
  - united_length 元ファイルの長さ
  - split_count 何個に分割したか。（最低 2。）
    この数とパリティファイル 1 個、最低 3 個のファイルが出来る
  - index 何番目のファイルか。（0 始まり。）
    index = split_count の時は、パリティファイル
- 元ファイルを 1 byte ずつ分割。
  united_length が split_count で割り切れない場合は 0 でフィリング
- 最後のファイル書き出し時以外は全てオンメモリで処理。
  一時ファイルは作りません

split_count = 2 の場合、奇数目 byte を index = 0 のファイルに、偶数目 byte を index = 1 のファイルに、パリティ情報を index = 2 のファイルに保存

8.4 実演

8.4.1 分割

中身が `abcedf\n` の 7 byte のファイルを 2 個 + parity の計 3 個に分割してみる

```
$ hexdump -C origin

00000000  61 62 63 64 65 66 0a                              |abcdef.|
00000007

$ dvault split -u origin -s split0 -s split1 -s parity

$ hexdump -C split0

00000000  7b 22 69 6e 64 65 78 22  3a 20 30 2c 20 22 75 6e  |{"index": 0, "un|
00000010  69 74 65 64 5f 6c 65 6e  67 74 68 22 3a 20 37 2c  |ited_length": 7,|
00000020  20 22 73 70 6c 69 74 5f  63 6f 75 6e 74 22 3a 20  | "split_count": |
00000030  32 7d 0a 0a 61 63 65 0a                           |2}..ace.|
00000038

$ hexdump -C split1

00000000  7b 22 69 6e 64 65 78 22  3a 20 31 2c 20 22 75 6e  |{"index": 1, "un|
00000010  69 74 65 64 5f 6c 65 6e  67 74 68 22 3a 20 37 2c  |ited_length": 7,|
00000020  20 22 73 70 6c 69 74 5f  63 6f 75 6e 74 22 3a 20  | "split_count": |
00000030  32 7d 0a 0a 62 64 66 00                           |2}..bdf.|
00000038

$ hexdump -C parity

00000000  7b 22 69 6e 64 65 78 22  3a 20 32 2c 20 22 75 6e  |{"index": 2, "un|
00000010  69 74 65 64 5f 6c 65 6e  67 74 68 22 3a 20 37 2c  |ited_length": 7,|
00000020  20 22 73 70 6c 69 74 5f  63 6f 75 6e 74 22 3a 20  | "split_count": |
00000030  32 7d 0a 0a 03 07 03 0a                           |2}......|
00000038
```

8.4.2 リストア

フィアル split1、origin を無くしたとして、split0、parity から origin を復活させる

```
$ rm origin split1

$ dvault unite -u origin -s split0 -s parity

$ hexdump -C origin

00000000  61 62 63 64 65 66 0a                              |abcdef.|
00000007
```

8.5 悩み

- Python で実装したが、果たして良かったのか？
  Debian の最小構成で Python が入っていない
  パッケージ化を目指すなら、C で再実装した方が良いかも
- メタ情報を json で保存して良かったのだろうか？
  今後メタ情報が増えた場合に、文字列と数字の区別が出来たら便利だと思って json にした。
  しかし、C で実装する場合は json のパースは面倒。
  何より、現状では json の中で改行 x 2 が入ったらバグるので何とかする必要あり。
- オンメモリの処理にどこまでこだわるか？
  OS 上のファイルとしては存在しなくとも、ディスクに少しでもデータが残ると盗難時に解析される恐れがあるのでオンメモリの処理にこだわった。
  でも、よく考えると SWAP したら意味が無いし、最初に秘密鍵を生成する際にディスク上に一時保存したら終わり。
  大容量のファイルを分割する事に備えて、名無しの一時ファイル（開いた直後に OS で削除）くらいは使っても良いかも。
- 複数パリティに対応した方が良いか？

8.6 今後の予定

自分以外の人が使ってくれるなら、メンテナンスを行うつもりです。
（ドキュメントも作成する必要あり）
「使ってみたい」という人がいらしたら、教えて下さい。

9 Jenkins での Debian パッケージ自動化

前田耕平

昨年の大統一 Debian 勉強会 2013 で、「とある企業での Debian システムの使い方。」というセッション[*13]の中で、社内向けの Debian パッケージの作成と、reprepro を使ったパッケージの配布方法についての話をしました。今回はその続編です。

9.1 背景

管理するパッケージが少ない場合は、手作業でのビルドでも間に合っていましたが、管理するパッケージ数が増え、対象ディストリビューションが増え (Precise, Wheezy, Trusty)、一方メンテナンスする人を増やすのは時間が掛かる、という状況でした。

そこで、別件[*14]で Jenkins を用意したこともあったので、ソースパッケージを作成するプロセス以外は、全て Jenkins で実行して自動化することにしました。

9.2 自動化するプロセス

パッケージのビルドからローカルアーカイブへの登録までに行うプロセスは図 1 のようになります。1a は公式パッケージのソースパッケージを dget などで取得する場合、1b は独自にソースパッケージを作成する場合です。ビルドする対象のソースパッケージによって実際の手段自体は異なってきますが、実行するプロセス自体は図の通り変わりません。

[*13] http://gum.debian.or.jp/2013/session/437.html
[*14] Java や Python のプロジェクトのビルドやテストの実行目的。

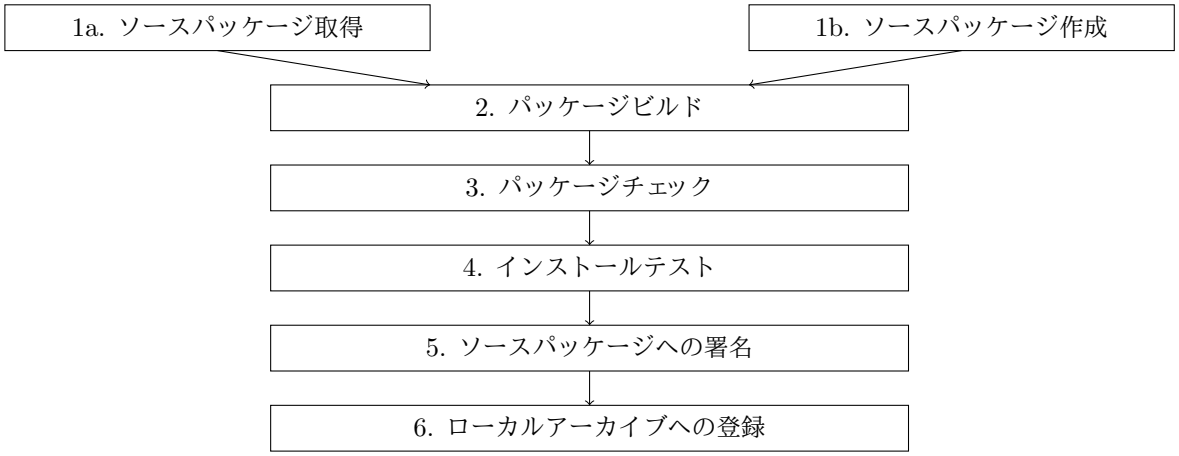


図 6　パッケージビルド～ローカルアーカイブへの登録までのプロセス

各プロセスの実行内容は下記のようになります。

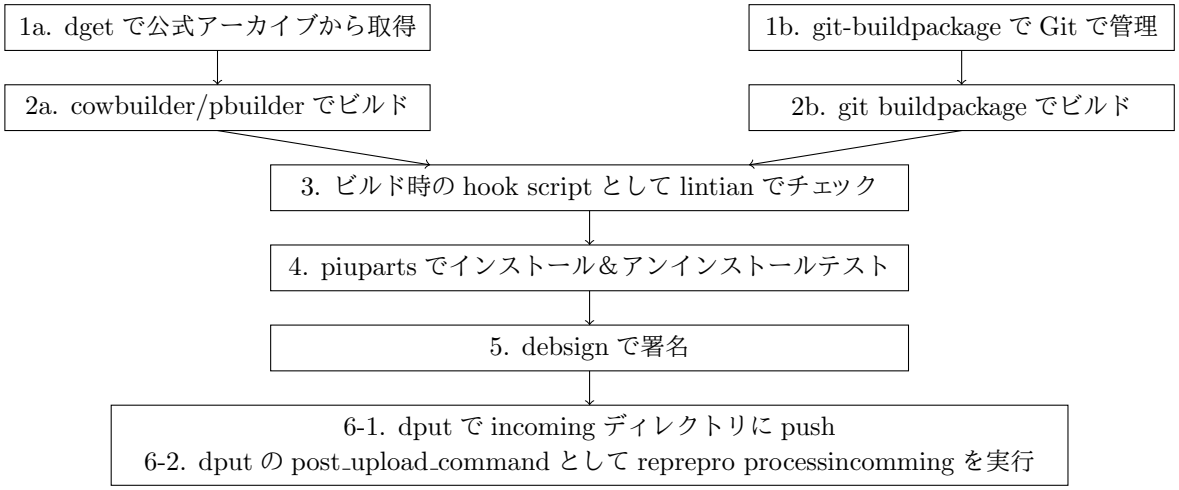


図 7　各プロセスの実行内容

これらを Jenkins でジョブとして実行するためには、対話形式の処理をなくし、リモートホスト上の処理も実行可能な状態で、一連の処理を Jenkins のジョブスクリプトを作成する必要があります。

9.2.1　Jenkins で実行する上での課題

Jenkins で実行する上で問題となるのは、GPG と SSH 公開鍵のパスフレーズの入力と、reprepro でのアーカイブの登録です。

reprepro でのアーカイブ登録は `reprepro processincoming` コマンドで行います。Jenkins と別ホストで reprepro を実行する場合、dput.cf の `post_upload_command` を使えば解決できます。例えば、Ubuntu Trusty 用のローカルアーカイブに dput する場合は、下記のような設定を dput.cf にしておきます。

```
[trusty]
fqdn                    = repo.example.org
method                  = scp
incoming                = /var/lib/debpkg-custom/ubuntu/incoming/
login                   = repoadm
post_upload_command     = ssh repoadm@repo.example.org /usr/bin/reprepro -b /var/lib/debpkg-custom/ubuntu \
--keepunreferencedfiles processincoming trusty #実際は一行
```

公開鍵の入力は、次の 3 箇所で必要になります。[*15]

[*15] Jenkins と reprepro を同じホストで実行する場合 (つまり一台で完結させる場合) には SSH 公開鍵のパスフレーズ入力は特に考慮するこ

- GPG
  5. debsign での署名

  6-2. dput の post_upload_command として reprepro processincomming を実行
- SSH

  6-1. dput で incoming ディレクトリに push

一方、GPG については既存環境で使っている鍵をそのまま利用する関係上、避けて通れません。Jenkins のジョブスクリプトで

```
$ echo -e 'passphrase\npassphrase\n' | debsign some.changes
$ echo -e 'passphrase\npassphrase\n' | dput some.changes
```

と実行しても標準入力からはパスフレーズは渡せません。しかし、gpg コマンド自体には`--no-tty --batch` オプションを使って、対話形式でなくてもパスフレーズを入力することができます。そのため、方法としては、

- debsign, reprepro processincomming を TTY なしで対話形式でなくても GPG パスフレーズを入力できるようにする
- expect を使ってプロンプトで判断してパスフレーズを入力させる

の 2 つの方法が考えられます。で、先に着手した debsign については、expect で処理するのは負けだと思ったので、別途 pydebsign という Python ライブラリを実装しました[*16]。しかし、reprepro については時間の都合上、結局 Python 版の expect の pexpect を使いました。下記のようになります。[*17]

```
def upload_files(codename, changes_path, reprepro_passphrase):
    """ uploading to local archive incoming directory of reprepro """
    os.environ['LANG'] = 'C'
    command = "dput %s %s" % (codename, changes_path)
    exp = pexpect.spawn(command, maxread=4000)
    exp.logfile_read = sys.stdout
    exp.expect('Please enter passphrase:')
    exp.sendline(reprepro_passphrase)
    exp.expect('Please enter passphrase:')
    exp.sendline(reprepro_passphrase)
    exp.expect(pexpect.EOF)
    exp.close()
    return exp.status
```

9.2.2 Jenkins で必要なプラグイン

パスフレーズは渡せるようになりましたが、Jenkins のジョブを実行するとパスフレーズが丸見えです。そのため、"Mask Passwords Plugin"[*18]というプラグインを使います。他に必要なプラグインは下記の通りです。また、Jenkins のジョブスクリプト管理用と、git-buildpackage 用と複数の Git リポジトリを扱えるようにするため、"Git Plugin"[*19]と"Multiple SCMs Plugin"[*20]も必要です。これらのプラグインは後述の Jenkins 環境構築後に、Jenkins の管理画面からインストールします。

9.2.3 その他の考慮点

- Jenkins で自動ビルド＆ローカルアーカイブへ登録したパッケージに依存するパッケージを作成することもあります。そのため、後述の pbuilder と cowbuilder のイメージの APT line には、ローカルアーカイブの URL を登録し、ローカルアーカイブの公開鍵を apt-key で登録しておく必要があります。
- バックポートする場合、orig.tar.gz,xz が.changes ファイルのリストにはデフォルトでは登録されず、reprepro で登録する際にエラーになります。そのため、debbuildopts='-sa' を指定する必要もあります。

とはありません。

[*16] Git: `https://github.com/mkouhei/pydebsign`, PyPI: `https://pypi.python.org/pypi/pydebsign`

[*17] debsign のこれで良かったですね。。。

[*18] `https://wiki.jenkins-ci.org/display/JENKINS/Mask+Passwords+Plugin`

[*19] `https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin`

[*20] `https://wiki.jenkins-ci.org/display/JENKINS/Multiple+SCMs+Plugin`

- pbuilder や cowbuilder でパッケージをビルドすると、デフォルトでは/var/cache/pbuilder/results ディレクトリ以下に作成されます。このままでは、Jenkins のジョブごとに処理するには不都合です。pbuilderrc で BUILDRESULT に、Jenkins の workspace 以下のパスを指定するようにします。

以上の点を考慮し、Python で Jenkins ジョブ用のスクリプトを作成しました。[*21]都合により、Golang 版も作成しました。[*22] これらは、前述の hook script で行う lintian でのテストを、dput の`-l` オプションで実行できるようにしています。[*23]

9.3 Jenkins 及び reprepro 環境の構築

Jenkins で自動実行するための環境として、下記の準備が必要になります。

- Jenkins のインストール＆設定
- Jenkins 用ユーザへの sudo 設定
  - /usr/sbin/pbuilder
  - /usr/sbin/cowbuilder
  - /usr/bin/git-buildpackage
  - /usr/bin/piuparts
  - /bin/chown
- pbuilder and cowbuilder イメージの作成
  - ビルド対象のディストリビューションの数分だけ作成
- debsign および reprepro 用の GPG 鍵の作成
- reprepro への upload 用の SSH 公開鍵の作成[*24]
- dput.cf の設定
- rererpro のインストール＆設定
- reprepro 管理 pool の公開設定
- ローカルアーカイブ用の GPG 公開鍵の export

このうち、pbuilder と cowbuilder イメージの作成の一部と、GPG 鍵の作成、SSH 公開鍵の作成については、手動で行う必要がありますが、他は Ansible で自動化しました。[*25]

9.4 まとめ

ソースパッケージの作成と Jenkins のジョブの作成以外は自動化できました。これにより、Debian パッケージの作成をやったことが無い人でもバックポートなどはできるようになりました。しかし、実際にはジョブを実行するとパッケージの依存関係や、lintian、piuparts などのエラーが発生します。そのため、Debian パッケージ作成について知らなければ、それらのエラーに対応できないことが予想されます。

また、ソースパッケージの作成は相変わらず、属人化しています。[*26]なので、今回の Debian パッケージング道場でパッケージ作成ができる人が増えるとイイですね。[*27]

[*21] デモのみ。
[*22] `https://github.com/mkouhei/godebbuild`
[*23] backport などの場合、lintian の Warning, Error を修正するためのパッケージの変更せず、許容してしまいたいこともあるため。
[*24] Jenkins と reprepro が別ホストの場合
[*25] デモ参照。Playbook は、`https://github.com/mkouhei/playbook-tokyodebian115` に用意してあります。
[*26] つまり、職場では自分しか作成できない。
[*27] 原稿作成時点では職場の同僚の参加者は未定。

10　DebConf14 のビデオ紹介

野島 貴英

10.1　DebConf14

DebConf とは毎年 1 回 Debian Project 主催で開かれる大きなカンファレンスのことです。

そこでは、世界中から Debian Project の活動に興味を持つ方々が集まり、Debian の開発や貢献についての発表や活発な議論が行われています。

今年は第 15 回目のカンファレンスとなり[*28]、8/23〜8/31 の間、米国オレゴン州ポートランドにて、DebConf14 として開かれました。

ここでは、DebConf video チームにより公開されている DebConf14 のカンファレンスの各セッションのビデオを視聴し語ってみます。

10.2　DebConf14 のビデオに関する情報

DebConf14 のスケジュールと、開かれたセミナの概要などの情報は、DebConf14 の公式ホームページの `http://debconf14.debconf.org/` を見て下さい。

セミナと BOF の動画ファイルは、`http://meetings-archive.debian.net/pub/debian-meetings/2014/debconf14/webm/` に webm フォーマットでおいてあります。video codec は vp8、audio codec は vorbis という、Debian らしくオープンな技術に拘ったデータ形式となっているのが特徴です。

10.3　視聴するにあたって補足

大量のビデオが公開されており、また時間も 1 時間弱程度はあるので、きっと本勉強会にいらっしゃるような熱心な Debian 関係者の方は、そのままのスピードで視聴するのではなく、早回しで視聴したいという要望を持たれる方がいらっしゃると思います。そんな時には、mplayer2 をお勧めしておきます。

```
$ apt-get install mplayer2
$ mplayer2 <動画ファイル名>
```

また、起動するとわかるのですが、あまり親切なメニューは出てきません。代わりに man mplayer するとわかるのですが、豊富なショートカットキーがあり、こちらを使う事により再生を制御できます。(表 8 参照)

また、英語の聞き取りが苦手な方には、DebConf subtitle project らによる、英語による書き起こしのテキストデータが入手できます。git からの入手の仕方は以下のとおりです。

[*28] DebConf0 が存在するので、全部で 15 回目です。

キー	操作	キー	操作
[	10% スロー	]	10% スピードアップ
←	1 分戻し	→	1 分スキップ
↑	10 分戻し	↓	10 分スキップ
o	残り時間/再生時間表示 (複数回押す)	q	mplayer を終了

表 8　mplayer のショートカットキー

```
apt-get install git
git clone http://anonscm.debian.org/git/debconfsubs/debconfsubs.git
cd debconfsubs/2014/debconf14/english/wip/
```

2014/9/25 現在のところ、未完成のものも含めて 9 つのセッションに関する書き起こしデータが手に入るようです。なお、.srt の拡張子を持つデータは字幕データであり、字幕ファイルを扱える動画プレイヤーで字幕を表示しながらセッションの動画を視聴することが出来ます。こちらについての操作方法の例は、第 105 回東京エリア Debian 勉強会のセミナ資料 [1] に具体的に記載していますので、こちらを参考にしてください。

さらに、git に格納されるよりも早く字幕の英文を見たい、あるいは、そもそも自分で英文字幕を起こす手伝いをしたい人は、`https://wiki.debconf.org/wiki/Videoteam/Subtitles` で紹介されている `http://www.amara.org/` の字幕書き起こし用の WEB サービスにアクセスしてください。こちらのページから起こし途中の字幕を閲覧したり、字幕付きの DebConf14 のセッションの動画を実際に見ることが出来ます。

10.4　DebConf14 大物ゲスト

今回は、Debian に直接の関係はあまりないのですが、大物ゲストが呼ばれており、専門のセッションが開かれています。

10.4.1　Q&A with Linus Torvalds

Linux カーネルの最初の作者である Linus Toravalds 本人を呼んで、セミナ参加者から一人づつ質問を受け続けるというセッションです。

- Linus さんの関心のほとんどは Linux カーネルそのものに費やされており、ディストリビューションの動向についてはあまり気にしてないようです。Debian も 1 度試したきりで結局 Mac に入れようとして失敗しただけのようで (Ubuntu も同様)。
- gcc の件や、systemd の件など、次々と一見答えにくい（？）質問が次から次へと Linus さんにぶつけられていました。特に systemd の件では Linus さんもコミュニティでの扱いについて熱い答弁をされていました。
- GPU と kernel の関係などについて昨今の事情についての質問もありました。

最初の質疑で Linus さんも非常に気楽に答えていたためうっかり、品位に欠ける表現がところどころ飛び出してしまい、女性の DebConf スタッフ (Debian Women の方）に表現の注意を受ける一幕もありました。

10.4.2　Weapons of the Geek

クラッカー集団である Anonymous についての社会面、文化面の研究で有名な、Gabriella Coleman さんのセッション。Anonymous と Anonymous を取り巻く独特のハッカー文化の解説と考察を行うというセッション。日本にはほとんど馴染のないサブカルチャー（例：XENU,Chanology,Scientology など）の紹介と、これらに対する Anonymous の考え方への影響などの紹介があって面白いです。

10.5 Debian の主なトピック

10.5.1 Bit From the DPL

2014 年も DPL 続投の Lucas Nussbaum さんのプレゼン。

プレゼン資料は、`http://blop.info/p/201408-dc14-dpl.pdf` で公開中。

- Debian Project の収支の件の話が出ました。Debian Project は資産を Trusted Organization として認めた機関にあずけているのですが、全部あわせると、日本円で約 2,800 万円ぐらいある模様です。保有割合としては、SPI(米国/US$預かり）が最も多い状況です。さらに、DebConf の度にスポンサーからの寄付を使い切れず資産が増えてしまう模様で、ここ 2 年がその傾向が顕著とのことです。
- さらなる予算の活用についてのディスカッションが行われました。予算利用の候補としての案は以下の 4 つ。
  1. Debian の普及に使う（例：mini-DebConf をもっと開く。景品 (T シャツなど）を新しい貢献者へ配る、Debian のブースの何かに使う）
  2. セキュリティに関して強化。Debian 公式開発者へ暗号化用スマートカードを配る。
  3. 開発の効率化に活用（例：Sprint の活発な開催、開発用ハードウェアを充実させる、DebConf への travel スポンサーの増資）
  4. upstream との積極的なコミュニケーションの為に活用（例：upstream についてのカンファレンスへの参加の旅費。upstream の接待など）

  会場では参加者の挙手を募り、賛否の割合を確認していました。

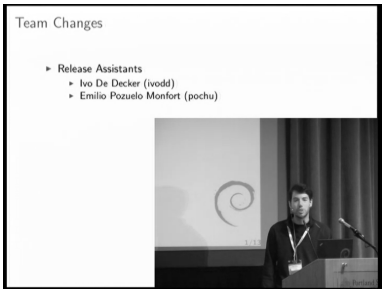
- Debian Project に関して SWOT 解析をされていました。SWOT のうち、弱み (Weaknesss) としては、中核部分に関しての完全な人手不足、技術的でない部分への興味のなさと協力者の不足、Debian 開発者同士でノウハウの共有化が行われていない場合がある、パッケージ化が難しい、メンター不足やそもそも必要な技術力が高いなどで新参者の開発参加のハードルが非常に高い、upstream とのコミュニケーションが薄いという点が挙げられていました。脅威（Threats）としては、他のディストリビューションではすでに解決済みのことに対応できていない、Debian の活動をするのに必要なスキル（開発とシステム管理作業など）を習得するような大学のカリキュラムがない、他プログラミング言語が独自で持つパッケージシステムと Debian パッケージの比較をされてしまうが挙げられていました。

10.5.2 Jessie bits from the release team

Release チームのセッションです。

スライドは `https://release.debian.org/talks/debconf14/rt-debconf14.pdf`

主な内容として、



- freeze までのタイムスケジュールと内容は以下の通り。
  - 9/5 に新規の transition を止める（ライブラリのアップグレードはここで終了）
  - 10/5 より緊急のアップロードを無視しはじめ、testing への移行に 10 日かかるようになる。また、セキュリティチームからサポート不可のパッケージの吟味が行われるようになる。
  - 11/5 Freeze する。
- 現在の RC bug の残りは 450 個。2/5 以降、testing に移動するのが望ましくないと判断されたパッケージは remove される。基本的にどのパッケージも remove から無事だとは思わないでほしいとのこと。
- 今から注意してほしい点として、今からはもう新規の transition を提案しないでほしい、Jessie に入れる気の無いパッケージのアップロードは一旦やめてほしい、とにかくインストールテスト (特に UEFI 対応の PC を持っている人はできるだけ協力タノムとの事）と、バグを潰してほしいとのことです。

10.6 日本の参加者の方の発表

今回、２名の日本からの参加者の方が発表をされていましたのでここに紹介しておきます。

10.6.1 My PGPGPG key is RSA 2048bit but I put the private key on Gnuk Token

新部さんのセッション。

内容は Gnuk Token の歴史と構造、動作の仕組みについてのセッションです。動作デモもありました。

スライドは、`http://gobby.debian.org/export/debconf14/bof/gnuk`

- Gnuk Token は、gpg のセキュリティスマートデバイスとして動作できる USB ドングルの事。新部さん開発。このドングルを利用して gpg サインを行えば、暗号処理もドングル内部で行うため秘密鍵を不正に取り出されることもなくセキュアに署名・暗号化が出来る。
- Gnuk Token では、乱数発生機として、未接続の内蔵 AD コンバータの 1 ビット目を使ったとのこと。
- Gnuk Token は最大 3 つの鍵を扱えるとのこと。ストア可能なキーサイズは 2kbytes はストアできる。
- 動作速度として、1.5 秒で Debian パッケージの gpg サインが可能。

途中、ドングル売ってくれとの聴講者の要望があったのが印象的でした。

10.6.2 find & imporove some bottleneck in Debian project

やまねさんによるライトニングトーク (以下 LT) 中の発表。動画ファイル名 Lightning_Talks_4.webm 中 0:43:14 あたりで発表。

スライドは `http://www.slideshare.net/henrich_d/find-improve-some-bottleneck-in-debian-project-debconf14-lt`

- NEW キューの ftpmaster によるチェックに時間がかかる事を解決したいという内容。現在 ftpmaster だけが膨大な量のパッケージのレビュー作業・差し戻し作業をやっている事により、作業量が多すぎて NEW キューの受け入れが滞りがちという問題がある。

- ここで review contributor という人を募集し、ftpmaster が現在行っている NEW キューのパッケージチェック作業を、彼ら（複数人）にやらせ、ftpmaster は最終の受け入れの OK/NG のみ出す役割にする。
- review contributor は、Debian 開発者候補としての訓練にも良いし、ftpmaster の作業が過多になって NEW キューが滞るのも解決できて一石二鳥でウマーというのがメリットなので、どうでしょう？という提案。

こちらをきっかけに、Debian Project に採用されると良いと思いました。

10.7 その他

その他発表で興味深い内容のものを紹介します。

10.7.1 Debian in the Dark Ages of Free Software

2012 年の DPL であった Stefano Zacchiroli(以下 zack) さんの発表です。

- Debian は DFSG Free な distoribution を作り普及させたことでは一定の成功を収めた。
- OSS も大変身近なものになり、ユーザは、たくさんのソフトウェアについて改変の自由が提供されるようになった。
- しかしながら、これらが成功した一方で、クラウドサービスも進化したため、せっかく勝ち取ったはずのソフトウェアの自由が、クラウドサービスの普及により、結局ユーザの手から取り上げられつつある。
- また、自由 (Free)Software を作るには Free の開発ツール/開発環境が究極的には必須であるにもかかわらず、github/Gmail などユーザからみれば自由でないサービスが益々開発ツール/環境としての地位を強固なものにしている。
- このような時代にある事を認識し、これを自由 (Free) Software の暗黒時代と呼ぼう。

セッションでは、zack によるアイデアも披露され、また聴講者からも提案があったものの、zack 曰くはっきりとした良い解決方法はまだ自分にもわからないという事でした。

このセッションで示されている問題は、難しく由々しき問題です。共感される方は是非議論に提案に加わっていただけると幸いです。

10.8 おわりに

他にもいろいろなセッションの動画が公開されています。ここでは紹介しきれなかったセッションにも面白いものが多数あります。

興味がある方は是非視聴くださいませ。また、英語のヒアリングに自信のある方は DebConf Video subtitle チームに作業のご協力をおねがいします。

参考文献

[1] OSC 2013 Tokyo/Fall 東京エリア Debian 勉強会セミナ資料, http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201310-presentation.pdf

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. MATE desktop 環境が先日 Debian のパッケージに入りました。対象の MATE のバージョンは？

☐ A 1.7
☐ B 1.8
☐ C 1.9

問題 2. ARM64 アーキテクチャへの対応の協力者募集が行われました。残念ながら Debian は動きませんが、民生品で手に入る ARM64 搭載の機材は次のどれでしょう？

☐ A iphone 5s
☐ B NEXUS 7
☐ C OpenBlocks A7

問題 3. edos.debian.net が qa.debian.org/dose で復活しました。ところで、このサイト何するサイト？

☐ A 各パッケージの依存関係がお互いに満たされている状態かをチェック
☐ B 各パッケージがどれだけ upstream から乖離しているかチェック
☐ C 各パッケージがどれだけ人気があるかをチェック

問題 4. 先日 Debian GNU/Hurd の中間報告がなされました。init が変更になったそうですが、どれになったでしょうか？

☐ A Hurd 独自実装の init
☐ B systemd
☐ C Sysvinit

問題 5. DEP-12 が PTS によりサポートされたそうです。ところで、DEP-12 の内容って何？

☐ A CI ツールによる自動テストの為の定義ファイルの提案
☐ B upstream に関する様々な情報を記載したメタデータの提案
☐ C copyright ファイルを機械処理がしやすいフォーマットにする提案

問題 6. dput する前には duck で必ずテストしてほしいとのこと。ところで duckって何？

☐ A debian/以下のファイルに含まれる URL/メールアドレスの存在をチェックするツール
☐ B 状況・様子から名前を断定するツール
☐ C 依存関係をチェックするツール

問題 7. mentors.debian.net でレビュー待ちのパッケージはいくつある？

☐ A 40〜50
☐ B 50〜60
☐ C 70 以上

問題 8. libc が変更になりました。変更前と変更後の組み合わせで正しいのはどれ？
☐ A eglibc から glibc へ変更
☐ B glibc から eglibc へ変更
☐ C uClibc から glibc へ変更

問題 9. ネットワークは Tor を使い、ファイル/メール/メッセンジャーソフトは暗号化対応という特徴を持つ debian ベースの Live OS の名前は何でしょう？
☐ A Plan9
☐ B Tails
☐ C GNU/Hurd

問題 10. Debian 6 LTS 開発の為の寄付を受け付ける会社名は？
☐ A Canonical
☐ B SquareEnix
☐ C Freexian

問題 11. jessie のデフォルトの gcc のバージョンは何になった？
☐ A 4.9
☐ B 4.6
☐ C 2.6

問題 12. jessie の freeze は 11 月 5 日です（注：リリースじゃないよ。）ここで、スケジュールで新規の transition が出来なくなる日はいつでしょう？
☐ A 10 月 5 日
☐ B 9 月 5 日
☐ C 8 月 5 日

問題 13. 2014/8/16 に Debian は誕生日を迎えました。さて今年で何歳?
☐ A 25
☐ B 21
☐ C 20

問題 14. 2014/7 時点の各国のアクティブな Debian Developer の数とそれぞれの国の人口との比率が最も多いのはどこの国?
☐ A Finland
☐ B Ireland
☐ C New Zealand

問題 15. OpenAmbit が 2014/7 にパッケージ化され sid へリリースされました。ところで何するパッケージ?
☐ A AmazonTV 用アプリ開発環境
☐ B ChoromeCast 用アプリ開発環境
☐ C SUNNTO AMBIT 用アプリ開発環境

問題 16. 2014/7/31 に technical committee により Debian のデフォルトの JPEG 圧縮伸長ライブラリとして採択されたのは以下のうちのどれ?
☐ A libjpeg6b
☐ B libjpeg8/9
☐ C libjpeg-turbo

問題 17. 2014/7/31 に technical committee から、「Debian パッケージは複数の init システムに対応する事」ということが再アナウンスされました。何のパッケージのドタバタがきっかけでしょう?
☐ A ftp
☐ B tftp-hpa
☐ C ncftp

問題 18. 2014/7/20 に squeeze（注：squeeze-lts ではない）の最後のアップデートが行われました。何回目のアプデートでしょうか?
☐ A 10
☐ B 9
☐ C 8

問題 19. 2014/7/31 に Jessie に搭載の可能性のある linux kernel のバージョンのアナウンスが行われました。バージョンはいくつでしょう?
☐ A 3.14
☐ B 3.12
☐ C 3.16

問題 20. VCS-*フィールドの VCS 情報を参照し、パッケージの changelog が合致しているかのチェックが行われるようになりました。以下のどのサイトで確認できる?
☐ A http://qa.debian.org/cgi-bin/vcswatch
☐ B http://bugs.debian.org/
☐ C http://codesearch.debian.net/

問題 21. パッケージに含まれているドキュメントに関して、lintian が新たな警告をするようになりました。以下のどれ?
☐ A HTML ファイル中の画像/CSS/JS/video リンクがローカルファイルを指していない場合、警告
☐ B HTML ファイル中の Debian の綴りが間違っている場合、警告
☐ C HTML ファイルが入っていない場合、警告

問題 22. 2014/7 に BTS の WEB サイトに reply のリンクがつくようになりました。このリンクをブラウザからクリックすると何が起きるようになった?
☐ A BTS 上で reply している次のメッセージが表示されるようになった。
☐ B BTS 上の reply 用のフォームがブラウザに表示されるようになった。
☐ C メーラが起動し、正しい Subject/宛先/引用が入るようになった。

問題 23. 2014/8/14 に Debian Installer Jessie Beta 1 がアナウンスされました。このバージョンのインストーラで導入される init システムは結局どうなった?
☐ A systemd になった
☐ B sysvinit になった
☐ C upstart になった

問題 24. FSF が Debian Project へ案内をしてきた、自由ソフトウェアのみの元で動かすことの出来るハードウェアについてのデータベースは次のうちどれ?
☐ A h-node.org
☐ B wiki.debian.org/Hardware
☐ C openbenchmarking.org

問題 25. Debconf14 の参加人数は結局何人？
☐ A 900 人
☐ B 300 人
☐ C 1000 人

問題 26. 8/17 に buildd にて使われるアーカイブがどこからもアクセスできるようになりました。url はどれ？
☐ A ftp.debian.or.jp/debian/
☐ B ftp.jp.debian.org/debian/
☐ C incoming.debian.org/debian-buildd/

問題 27. 2014/8/19 に登録商標として Debian ロゴが正式に登録されたそうです。どこの国の登録商標でしょうか？
☐ A 米国
☐ B 日本
☐ C スイス

問題 28. 2014/8/24 の BitFromDPL によれば、Debian Project は仮想通貨による寄付をはじめて受け付けたそうです。具体的には何という仮想通貨でしょう？
☐ A Gree コイン
☐ B Crysta
☐ C BitCoin

問題 29. 検索エンジンの DuckDuckGO より、収入が入ったとのことです。2014/8/24 現在、月当たりの DuckDuckGO からの平均収入は月額いくらでしょう？
☐ A $10
☐ B $152
☐ C $1400

問題 30. 2014/8/27 に Debian archive に搭載された 2 つの新しいアーキテクチャは、arm64 以外には以下のどれ?
☐ A sparc
☐ B mips
☐ C ppc64el

問題 31. 2014/8/31 にて、arm64 ポートの Debian 開発用に、無償の ARM64 用のコンパイラ・デバッガ等の開発キットの提供が行われたようです。製品名は以下のどれ？
☐ A Microsoft Visual Studio
☐ B IAR Embedded Workbench
☐ C DS-5 Development Studio

問題 32. Debian keyring からある大きさ以上の秘密鍵長を持たないキーが 2014/12/31 以降で削除される事についてのリマインドのアナウンスが流れていました。ある大きさとは以下のどれ？
☐ A 512bit
☐ B 2048bit
☐ C 4096bit

問題 33.　2014/9/17 に Debian Policy が改定されました。改定後のバージョンはいくつ？
□ A　3.9.5.0
□ B　3.9.6.0
□ C　4.0.0.0

問題 34. 2014/9/14 時点で Jessie でサポートされると残念ながら「言われなかった」アーキテクチャはどれ
□ A amd64
□ B powerpc
□ C sparc

問題 35. 2014/11/1 時点で Jessie でサポートされるかどうかについて依然として懸念のあるものはどれ
□ A hurd
□ B kFreeBSD
□ C s390x

問題 36. 2014/9/4 に testing 入りしたデスクトップ環境は以下のどれ
□ A xfce
□ B cinnamon
□ C unity

問題 37. 2014/10/18 にリリースされた wheezy の Debian のバージョンはいくら？
□ A 7.7
□ B 7.8
□ C 7.9

問題 38. OPW の呼びかけが debian-devel-announce で 2014/9/30 に行われました。ところで OPWって何の略?
□ A One-time PassWord
□ B One-Piece-Woven technology
□ C Outreach-Program-for-Women

問題 39. 2014/10/14 にサービスを近いうちに閉じるよとアナウンスのあったサイトはどれでしょう？
□ A githubredir.debian.net
□ B tracker.debian.org
□ C rtc.debian.org

問題 40. 2014/10/5 にリリースされた Debian Installer Jessie Beta2 の変更点はどれ
□ A syslinux まわりで過去の互換の無い大きな変更点が出た。
□ B デフォルトの init が systemd となった。
□ C GNOME デスクトップ環境がデフォルトになった。

問題 41. 2014/10/16 に投票が開始されました。内容は次のうちどれ？
□ A 特定の init とプログラムが依存しても良いか？ダメか？程度次第か？
□ B GNOME をデフォルトにしてよいのか？
□ C 野島が勉強会幹事をやり続けて良いのか？

問題 42. Debian Project 関係者の PodCast のサイトが公開されました。以下のどれ？
□ A www.debian.org
□ B www.debianandstuff.com
□ C www.debian.or.jp

問題 43. 2014/10/27 の DPN に Ada initiative から寄付のアナウンスの件が載っていました。Ada initiativeって何？
□ A オープンなテクノロジに関して女性活躍の支援をする団体
□ B プログラミング言語 Ada の普及促進をする団体
□ C Ada さんの政治後援会

問題 44. 2014/10/27 に Debian の whois コマンドが入れ替わりました。特徴はどれ？
□ A サイズが小さくなった
□ B DFSG に準拠した
□ C 作者独自の調査により IANA の情報より正確になった

問題 45. 2014/10/15 時点で、Freexian と契約した Debian の LTS のスポンサーは全部で何社？
□ A 14 社
□ B 13 社
□ C 12 社

問題 46. 2014/10/27 の DPN にて Debian Multimedia の進捗状況報告がありました。libav6:11 で搭載された新しい機能は次のうちどれ。
□ A libx265-encoder
□ B libx265-decoder
□ C libx264-encoder

問題 47. 2014/11/5 にて Freeze が行われました。この時残っている RC bug は何個だったでしょう？

☐ A 200 個

☐ B 310 個

☐ C 400 個

問題 48. 2014/10/27 にて初めて Jessie ベースの DebianEdu がリリースされました。DebianEdu は Debian の用語ではどのしくみに分類されるでしょうか？

☐ A Derivative

☐ B Blend

☐ C PureBlend

問題 49. Jessie から取り除かれる予定の Qt のバージョンはいくつでしょう？

☐ A Qt3

☐ B Qt4

☐ C Qt5

問題 50. main パッケージの Depends フィールドに”package-in-main | packages-non-free” と書いて良いかどうかの決定が 2014/10/31 に TechnicalCommitee により下されました。結論は以下のうちのどれ？

☐ A 状況次第で OK だったり、NG だったり

☐ B NG

☐ C OK

問題 51. 2014/11/9 の Release Team から Debian 9,Debian 10 のコードネームが決まりました。Debian 10 のコードネームは次のうちのどれ？

☐ A Buster

☐ B Stretch

☐ C Jessie

問題 52. 2014/11/9 の Release Team のメールにて、arm64, ppc64el, kfreebsd について、Jessie の公式リリースに含むかどうかの決断が行われました。「含まない」とされたのは次のうちどれ？

☐ A arm64

☐ B ppc64el

☐ C kfreebsd

問題 53. 2014/11/14 にて、Debian Med チームから、とあるパッケージをやっと DFSG 準拠にすることが出来たとの報告がありました。そのパッケージ名は以下のどれ？

☐ A abyss

☐ B arb

☐ C phylip

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. B Gnome 2.28 ベース以降の Gnome からフォークしたデスクトップ環境とのことです。読み方は「マテ」（マテ茶のマテ）と読むとのこと。
- 2. A iphone 5s から搭載されている A7 SoC は ARMv8 搭載で ARM64 動作も出来るそうです。ARM64 を搭載した民生品はこれが最初かも。一方、Debian の ARM64 ポーティングでは全く別の開発用向けに提供されている機材が利用されています。我々が Debian の動く ARM64 機材を民生品で手に出来る日は何時ー（涙）？
- 3. A dose-*パッケージに含まれるチェックコマンドを使い、各アーキテクチャの debian リポジトリを毎日チェックして、依存関係が満たされていないパッケージをチェックした結果を教えるサイトとなります。膨大な量のパッケージの依存関係を迅速にチェック出来るところが凄い点です。仕組みについて IEEE の論文まで出ているようです。参考：http://hal.archives-ouvertes.fr/ docs/00/14/95/66/PDF/ase.pdf
- 4. C 去年の GSoC の成果がそのまま開発継続し、遂に Sysvinit に init が変更されたそうです。このおかげで、Network、halt/shutdown 時の動作、ブート時の filesystem のマウント方式がやっと Debian 流になったようです。
- 5. B DEP-12 はソースパッケージ中の”debian/upstream/metadata” 等のファイルをに用意し、upstream に関する様々な情報を記載しようという提案です。一方、「CI ツールによる自動テストの為の...」は DEP-8 であり、こちらは”debian/tests/” 以下のファイルを使い、ci.debian.net でパッケージ開発について継続的インテグレーションを実施するものとなります。先日紹介のアナウンスありましたね。
- 6. A debian/control ファイル、debian/upstream 関係のファイルに記載されている URL, メールアドレスのドメインパート (例:hoge@fuga.com の fuga.com 部分) が実際に存在するか/アクセス可能かを調べてくれるツールが duck となります。
- 7. C 東京エリア Debian 勉強会へいらっしゃるような皆様は、ぜひ debian-devel@debian.or.jp などで DD に mentor 役をお願いする、あるいは debian の開発チームでパッケージがメンテナンスをされている場合はチームの誰かに mentor 役をお願いするのがパッケージアップロードの道として早いです。不定期に開かれる Debian Hack Cafe で面前で直接頼むという手もあります。
- 8. A eglibc が採択されてから 5 年が経過し、glibc に戻ってきた状況です。
- 9. B Trails は The Amnesic Incognito Live System の略です。利用者の出来る限りの匿名性を保つ事を目的として作られた Debian ベースの live os となります。
- 10. C Freexian 社は、Debian の公式開発者の Raphaël Hertzog により設立された、Debian の有償サポートをするための会社です。
- 11. A 各パッケージは gcc-4.9 でもコンパイルが通るように修正が必要となります。
- 12. B 9 月 5 日以降は新規の transition が出来なくなります。新しいパッケージを jessie に合わせてリリース

しようと目論む方々は期日にご注意下さい。

- 13. B 1993/8/16 に Ian Ashley Murdock さんにより、Debian は創設されました。というわけで、21 歳です。Debian Day はもう過ぎちゃいましたが、本日 (2014/8/23) の勉強会後の宴会で祝いたいと思います！
- 14. A Finland は本割合は 2009 年からずっと毎年 1 位 (今年 3.61ppm。)2 位 Ireland 3 位 New Zealand。日本の本割合は 0.28ppm で 30 位でした。単にアクティブ Developer の絶対数が多い国は 1 位 USA、2 位 Germany、3 位 France。
- 15. C SUNNTO AMBIT という腕時計型の GPS 搭載の端末のアプリ開発環境となります。楽天とか、Amazon で買えます。スポーツ好きな方は Debian でアプリ開発やってみてください。
- 16. C libjpeg-turbo は libjpeg6 ABI 互換の JPEG 圧縮伸長ライブラリで、libjpeg6 を遥かに凌ぐ速さで動作します。ここで、JPEG 圧縮伸長ライブラリのデフォルトとして、libjpeg8/9 か、libjpeg-turbo かで議論が分かれていましたが、technical comittee は libjpeg-turbo をデフォルトの JPEG 圧縮伸長ライブラリとしました。
- 17. B experimental に tftp-hpa 5.2-17 が入ったが、その changelog として”Removing upstart hacks, they are ugly and upstart is dead now.” という内容が入っていた件が物議を醸しました。bug746715 参照。
- 18. A 最後の squeeze のアップデートです。実はこのアップデートを行っても、Debian ですでに報告されている squeeze のセキュリティホールの一部は残ったままです。しかしながら、これ以上のアップデートは無いので、wheezy にアップデートするか、squeeze-lts の採用を検討しましょう。
- 19. C 2014/8/14 に kernel.org で linux kernel3.16 のリリースがアナウンスされました。Debian Kernel チームのアナウンスでは、「3.16 が Jessie に採用される可能性があるので、現在のパッケージが 3.16 と互換のある Kernel API の元で動作するか確認し、問題があれば然るべき対応してほしい」とのこと。
- 20. A vcswatch のページにアクセスし、見たいパッケージ名入れると、状況が得られます。詳しくは vcswatch のサイトの説明をご覧ください。
- 21. A ブラウザが HTML ファイルを表示する場合、ローカルのファイルであっても、表示時にブラウザが自動でリンク先をアクセスしてしまう作用を持つリンクがあります。ここに外部サイトの URL が入っていた場合、その外部サイトは、HTML ファイルが閲覧された時の情報を収集することが出来てしまいます。こういったことはセキュリティ上良くないので、該当するような HTML ファイルが含まれている場合、lintian が警告するようになりました。
- 22. C やってみれば判りますが、Bug レポートに対する返答をする場合に、非常に便利です。BTS から見て、正しい内容と宛先で、返答出来るように下書きメールがメーラに準備されるようになりました。
- 23. A このバージョンの Debian Installer から、init システムとして systemd が入るようになりました。ちなみにこの件、Installer プログラム本体の話であって、Jessie 本体の Beta 版リリースという意味では無い事に注意。
- 24. A 日本語の本件のニュースは sourceforge.jp の記事参照:http://sourceforge.jp/magazine/14/09/11/062900 。FSF は main リポジトリのみのパッケージで構成される Debian は自由ソフトウェアとみているとのこと。
- 25. B 300 人とのことです。参考 : Debconf13 は 290 人、Debconf12 は 176 人、Debconf11 は 335 人でした。
- 26. C 今までは、どこからもアクセスできたわけではなかったようです。
- 27. A United States Patent and Trademark Office になります（つまり米国。）登録された Debian ロゴのデザインは http://tdr.uspto.gov/search.action?sn=86037470　からたどると閲覧できます。
- 28. C Debian Project は BitCoin をそのまま受け付けることが出来るシステムを持たないため、その場限りの方法で受け取ったとのことです。今後、こういった仮想通貨での寄付の受け取りと取り扱いについて意見がほしいとのことです。
- 29. B Debian パッケージに含まれるブラウザにデフォルトで登録されている検索エンジンの候補として DuckDuckGO が搭載されていることによる収入となります。DuckDuckGO はプライバシーに配慮した検索エンジンです。最近は、iphone の safari ブラウザにあらかじめ登録される検索エンジンの候補としても上がり有名になりつつあります。URL は https://duckduckgo.com/

- 30. C 64 bit powerpc の little endian モードのポーティングとのことです。すでに存在する ppc64 は big endian のバイナリのポートティングとなります。
- 31. C Debian Edition とのことです。アナウンスによれば、ダウンロードリンクは http://ds.arm.com/debian/ からダウンロード可能とのことですが、日本からはダウンロードが現在出来ない模様です。残念！もちろんですが、この開発キットは無償ではあるものの自由ソフトウェアではないので誤解なきよう。
- 32. B キーサインにつかう gpg の鍵も 2048bit 以上にしましょう！
- 33. B パッケージ開発をする前に、新しい Debian Policy の変更差分は読んでおきましょう。
- 34. C 2014/9/14 時点でサポートされると言われたアーキテクチャは：amd64,armel,armhf,i386,kfreebsd-amd64/kfreebsd-i386/mips/mipsel/powerpc/s390x。なお、arm64/ppc64el は好調な頑張りとのことで、この調子が続けば入るかも ?? という状況。
- 35. B kFreeBSD は頑張って欲しいとのこと。最終ジャッジは 2014/11/1 に行われる。なお、s390x は 9/14 では懸念なし、hurd はすでにリリースは無理との判断になっている。
- 36. B kFreeBSD は頑張って欲しいとのこと。最終ジャッジは 2014/11/1 に行われる。なお、s390x は 9/14 では懸念なし、hurd はすでにリリースは無理との判断になっている。
- 37. B cinnamon は GTK+3 を利用して作られたデスクトップ環境です。元々は Linux MINT 向けに GNOME Shell から fork したデスクトップシェルでした。開発が続き、デスクトップシェルから発展して遂にデスクトップ環境となりました。
- 38. A 7 回目のアップデートとなります。早速アップグレードしましょう！新規にインストールする安定版なら Debian7.7 からやりましょう!
- 39. C Outreach-Program-for-Women(略して OPW) は、GNOME Foundation が始めた、FOSS のプロジェクトの貢献者にもっと女性を増やそう！という活動となります。FOSS の貢献者の男女構成比は、圧倒的に男の割合が高いといういびつな傾向があるため、こちらを是正しようとする活動です。
- 40. A githubredir.debian.net は、github 上にある upstream のソースについて、リリースの為にバージョン番号で tag 打ちされたバージョンのソースの tar ボールに対する直接のリンクを統一的な書式の URL で生成するサービスです。つまり、debian/watch に upstream のソースの有りかを書きやすくするために使われます。github が改善され今となっては不要となってしまったということもあり、近いうちに廃止するというアナウンスが行われました。
- 41. C 一度は Xfce デスクトップ環境と言われていましたが、accessibility の完成度の高さには勝てず、再び GNOME に戻ってきた形となりました。なお、他の選択肢は Beta1 の時の変更点となります。
- 42. A Debian の各種 init システムと他アプリケーションの依存についての規定に関する投票となります。投票内容について詳しくは https://www.debian.org/vote/2014/vote_003。ところで、Debian のイベントの幹事は、もちろん、いつ、誰が、どのようにやってもよくてよ？勇者の数を増やせ！
- 43. B 英語です。記念すべき第 1 回目は「MoinMoin Vs. MediaWiki」であり、パーソナリティーは Asheesh Laroia さんと、Sam Erbs さんとなります。収録は DebConf14 中に収録したそうです。
- 44. A オープンなテクノロジに関して女性活躍の支援が活発になってきました。Debian では、Debian Women というプロジェクトがあります。Gnome Foundation が 2006 年に Free & Open Source Software Outreach Program(略して OPW) を開始したのがきっかけで、Debian Project もこちらの動きに賛同している状況です。
- 45. C 今まで BSD 由来の whois コマンドが Debian で使われてきましたが、この度 Marco さん作の whois コマンドに変わりました。Marco さんは 1 年を費やして、IANA よりも正確に情報を得られるサーバ群を独自の調査で探し当て、こちらで対応するようにしたそうです。なお、Marco さんの whois コマンドは全 Linux ディストリビューションの標準の whois コマンドになるそうです。
- 46. A `http://raphaelhertzog.com/2014/10/15/freexians-second-report-about-debian-long-term-support/` に掲載されています。こちらのスポンサーのお陰で、LTS を担当できる Debian 開発者らのフルタイムのうち、25% の時間を割く事ができるようになったとのことです。古い Debian を使っている会社さんは是非スポン

サーになってくださいませ。

- 47. A libav6:11 は jessie 搭載予定の Multimedia 用 codec ライブラリです。遂に x265 のエンコーダが搭載されたようです。x265 は、ワンセグなどで使われている高性能なコーデックの H.264/MPEG-4 AVC の後継である H.265 の互換実装となります。H.265 は H.264 の 2 倍の圧縮率を誇るとのことで、Jessie での動画鑑賞が楽しみですね。
- 48. B Freeze 時に 310 個しか RC bug がなかったのは、昨今の Freeze ではなかったほどの快挙だそうです。さあ、RC bug 潰しまくりましょう！
- 49. C PureBlend は、特定用途向けの Debian に仕上がるようにインストールを行う場合、control ファイルに必要なパッケージを Recommends で指定しただけのパッケージを用意することによって、簡単に Debian パッケージ群のみをまとめてインストールできるようにするためのしくみです。詳しくは、「第 108 回東京エリア Debian 勉強会、2014 年 1 月勉強会」(http://tokyodebian.alioth.debian.org/2014-01.html) の資料に詳しいです。
- 50. B Qt4 は upstream 側で 2015 年に開発を終了する決定となりました。Jessie に搭載予定の Qt のバージョンは Qt5 となります。
- 51. C 例えば、”Depends: unrar-free ｜ rar” というようなパターンがありえます。通常は、main パッケージで構成される Debian システムは DFSG 準拠であるべきなので、「non-free パッケージのみ」に依存するようなパッケージを main 側に作ってはいけないというルールがあります。今回の場合は、main パッケージのリポジトリ指定時に、non-free のパッケージが優先して導入されることは無いので OK となりました。
- 52. A ちなみに Debian 9 は”Strech” だそうです。
- 53. C 大変遺憾ながら、kfreebsd は、期日までに Jessie 公式リリースに必要とされる品質に達しなかったとの事です。ただ、kfreebsd が Debian プロジェクト自体から消えるわけではないので、Jessie 公式リリース時期に、条件が揃えば非公式のリリースとして扱う事が可能とのことです。このパターンになったのは、Debian GNU/Hurd 2013 が 2013 年にそのようなリリースを行ったことがあります。
- 54. C upstream はワシントン大学である phylip は、bioinfomatics では有名なソフトウェアとのことです。しかしながら、ワシントン大学の課しているライセンスは、非営利の研究用途のみに利用を許諾している状態でした。こちらについて長年の Debian Med の交渉により、遂に phylip は DFSG 準拠の自由ライセンスにしてもらえたとのことです。また、phylip が原因で non-free にせざるを得なかった seaview というパッケージも無事 main にすることが出来るとのことでした。

『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 2014/06〜2014/11 まで東京エリアは第 114 回から第 120 回までおよび、関西エリアは第 85 回から第 91 回まで (第 86 回関西 Debian 勉強会 は OSC2014 Kansai@Kyoto のため無し第 87-89 回はもくもく会のため収録無し、第 90 回 関西 Debian 勉強会 は 関西オープンフォーラム 2014、第 91 回 関西 Debian 勉強会　Jessie インストーラテスト会のためなし)。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 **2014** 年冬号

2014 年 12 月 30 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）