.Deb

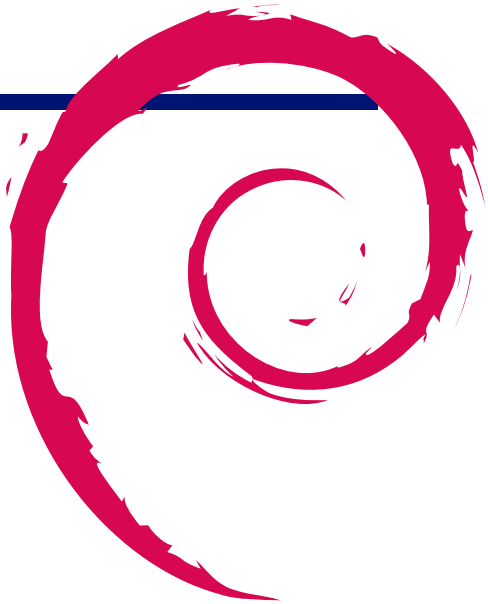
銀河系唯一のDebian専門誌

2014 年 06 月 14 日

特集：GPG 秘密鍵取り扱い方法の提案

デビアン勉強会

目次



1 事前課題

野島 貴英

今回の事前課題は以下です:

1. 本日、何の作業をやるかを宣言ください。

この課題に対して提出いただいた内容は以下です。

1.1 wbcchsyn

発表予定のツールをブラッシュアップする。
`https://github.com/wbcchsyn/dVault_prototype` フィードバックがあれば、取り込み。
無ければ、ドキュメント作成とか自分で気がついている部分の改修とか。

1.2 野島 貴英

- Debian ARM64 まわりを調べる。
- OSS 化した softether を debian パッケージとしてどうするか考える
- Debian Gnu/Hurd ネタを調べる

まあ、将来の東京エリア Debian 勉強会ネタの仕込み作業となりますなぁ。

1.3 yyuu

先々月に続いて、pyenv というスクリプトの deb 化のための作業を行なおうと考えています。
`https://github.com/yyuu/pyenv`
可能であれば pyenv-virtualenv プラグインも deb 化したいです。
`https://github.com/yyuu/pyenv-virtualenv`

1.4 zinrai

direnv の deb パッケージ化
`https://github.com/zimbatm/direnv`

1.5 koedoyoshida

あんどきゅめんてっどでびあんの編集作業

1.6 regonn(Kenta Tanoue)

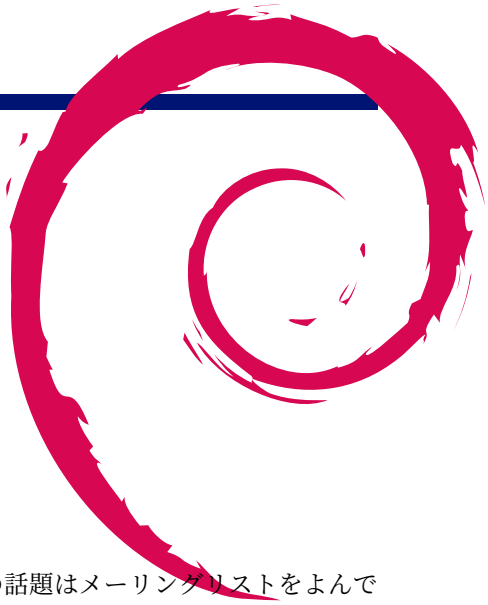
AWS debian 環境で、Rails+publify を使って自分のブログを作ります。(作業工程を記事にしたい) インストールの隙間時間で DDTSS の作業もできたらいいな。

1.7 なかおけいすけ

7 月の勉強会で発表せよとご指名いただいたので、その準備をします。

1.8 野首 (@knok)

- KAKASI のバグフィックス
- gnu.org 翻訳
- ECS Liva での debian kernel での mmc 関連修正



2 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ほりあいがないので、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` に投稿された内容などからです。

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？

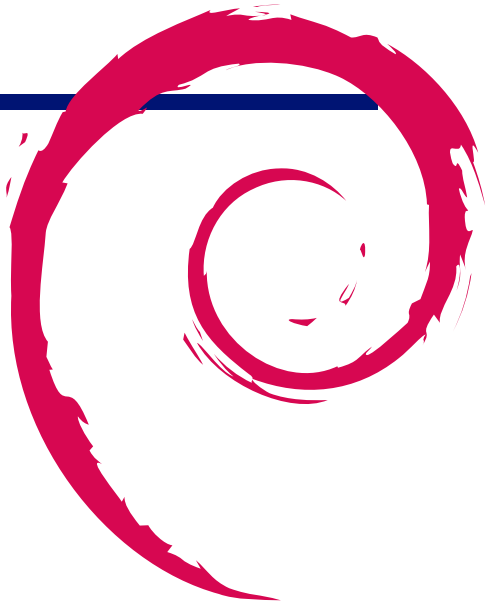
☐ A CI ツールによる自動テストの為の定義ファイルの提案
☐ B upstream に関する様々な情報を記載したメタデータの提案
☐ C copyright ファイルを機械処理がしやすいフォーマットにする提案

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



3 最近の Debian 関連のミーティング報告

野島 貴英

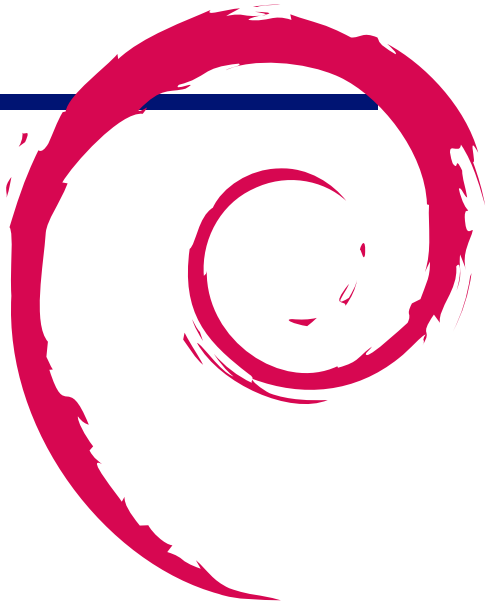
3.1 東京エリア Debian 勉強会 113 回目報告

東京エリア Debian 勉強会 113 回目は (株) スクウェア・エニックスさんで開催されました。

8 名の参加者がありました。また、野島さんより、仮想コンテナのフレームワークである docker.io についての発表が、実演ベースで行われました。

また、残った時間は各自でもくもく作業をして、結果を発表しました。

宴会は、会場近くの「南国亭新宿店」にて行いました。



4　GPG 秘密鍵取り扱い方法の提案

吉田 晋

4.1　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？

4.2　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！

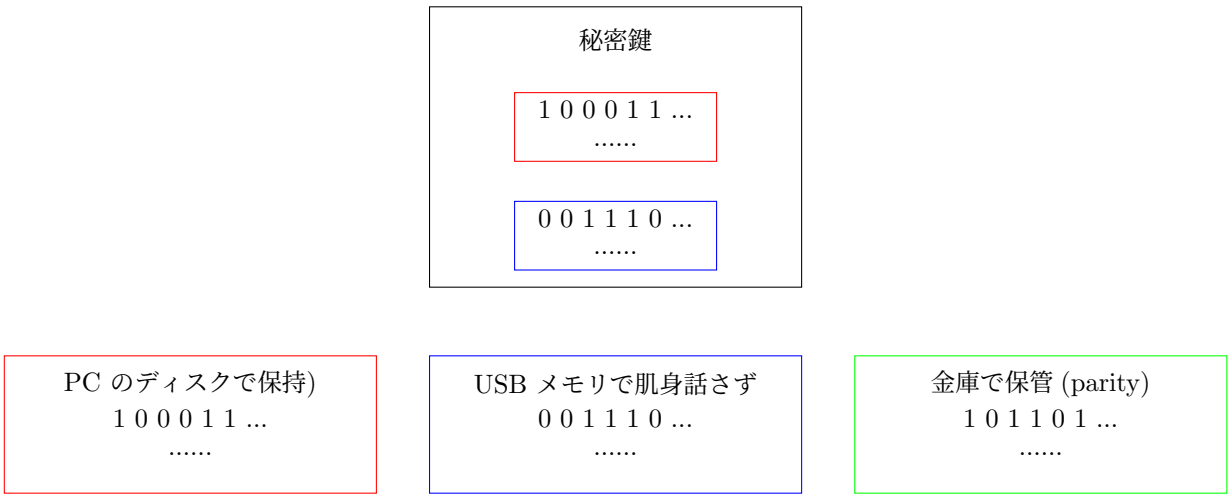


図 1　分散管理イメージ図

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。

4.3　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)

4.3.1　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。

4.3.2　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
```

4.3.3 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

4.4 実演

4.4.1 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
```

4.4.2 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

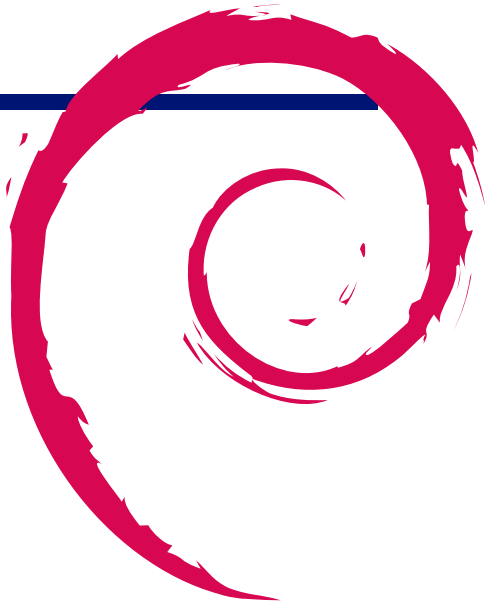
```
$ rm origin split1

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

$ hexdump -C origin

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

4.5 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？

4.6 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。



5　会場での無線 LAN のつなぎ方

野島 貴英

5.1　はじめに

今回試験として、会場側でフィルタ無しのグローバル回線を用意しました。ただ、会場側のセキュリティポリシーにより、wpa-psk AES hidden SSID という方式での提供となります。

以下に Debian マシンでの接続方法を記載します。

また、自分の環境では違うやり方でつながったという方は、野島まで教えて下さい。こちらでもノウハウとして溜めていく予定です。

5.2　wpasupplicant 及び/etc/network/interfaces を利用の場合

もっとも良いマニュアルは、/usr/share/doc/wpasupplicant/README.Debian.gz となります。困った場合はこちらも合わせてご参照下さい。

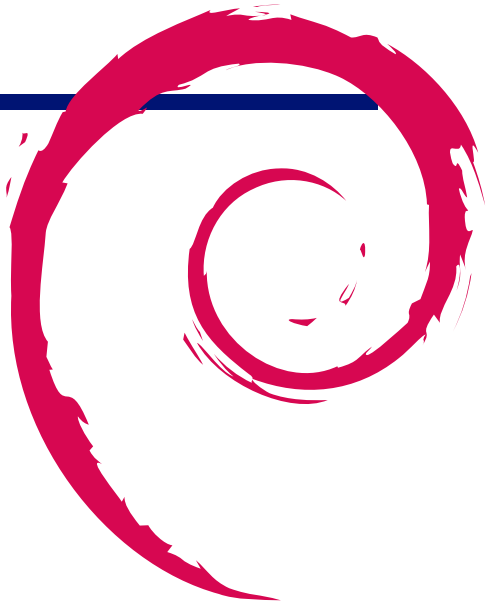
以下に/etc/network/interfaces の定義について会場の例を記載します。

```
$ sudo aptitude install wpasupplicant
# hidden ssid の元では必ず ap-scan 1,scan-ssid 1 を指定する事。
# 参考：http://bugs.debian.org/358137
$ sudo vi /etc/network/interfaces
-----以下のエントリを追記ここから----------
iface wlan_tokyodebian inet dhcp
     wpa-ssid <<会場の SSID>>
     wpa-psk  <<会場のパスワード>>
     wpa-ap-scan 1
     wpa-scan-ssid 1

-----以下のエントリを追記ここまで----------
#  無線 LAN を有効にする。
$ sudo ifup wlan0=wlan_tokyodebian
#  無線 LAN を無効にする。
$ sudo ifdown wlan0
```

また、ハマってしまった時のデバッグ方法は、/usr/share/doc/wpasupplicant/README.Debian.gz 中の”4. Trubleshooting” の章が便利です。

5.3　その他の無線 LAN 用パッケージを利用の場合

すみません、自分が情報を持たないため、現場で教えて下さい。



6 索引

Debian 勉強会資料

2014 年 06 月 14 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）