

A large, stylized pink brushstroke graphic that forms a circular shape, partially enclosing the text. It has a textured, hand-painted appearance.

東京エリア Debian 勉強会


第 102 回 2013 年 7 月度

上川純一

2013 年 7 月 20 日

設営準備にご協力ください。

会場設営よろしくおねがいします。



Agenda

- 注意事項
 - 飲食禁止
- 最近あった Debian 関連のイベント報告
 - 第 100 回 東京エリア Debian 勉強会
 - 2013 大統一 Debian 勉強会
- Debian Trivia Quiz
- 事前課題紹介



イベント報告

第100回 東京エリア Debian 勉強会



大統一 Debian 勉強会 2013





DWN quiz




Debian 常識クイズ

Debian の常識、もちろん知ってますよね？ 知らないなんて恥ずかしくて、知らないとは言えないあんなことやこんなこと、みんなで確認してみましよう。

今回の出題範囲は

`debian-devel-announce@lists.debian.org`,
`debian-devel@lists.debian.org` に投稿された内容と
Debian Project News などからです。

問題1. Stefano Zacchiroli さんが新しく作成したサービスは?

-  A chiebukuro.debian.net
-  B 2ch.debian.net
-  C sources.debian.net

問題1. Stefano Zacchiroli さんが新しく作成したサービスは?

答えは:



A `chiebukuro.debian.net`



B `2ch.deb`



C `sources.`





C

問題1. Stefano Zacchiroli さんが新しく作成したサービスは?

解説: Stefano Zacchiroli さんが、Debian パッケージで提供されているソースコードすべてを閲覧・検索出来る sources.debian.net を作った

問題2. 新しく FTP master チームに入った人は?

-  A Gergely Nagy
-  B Kouhei Maeda
-  C Joerg Jaspert

問題2. 新しく FTP master チームに入った人は?

-  A Gergely Nagy
-  B Kouhei
-  C Joerg Ja

答えは:



A

問題2. 新しく FTP master チームに入った人は?

解説: Paul Tagliamonte, Scott Kitterman, Luke Faraone,
Gergely Nagy が加入

問題3. Debian GNU/Hurd がリリースされましたが、バージョンはいくつでしょう。



A 2013



B 3.141592



C 7.0

問題3. Debian GNU/Hurd がリリースされましたが、バージョンはいくつでしょう。



A 2013



B 3.14159



C 7.0

答えは:



A

問題3. Debian GNU/Hurd がリリースされましたが、バージョンはいくつでしょう。

解説:





事前課題

- ① お使いのマシンに ARM がありますか？ もしあるのであれば、どのように使っているか教えてください。
Raspberry Pi を使ってます。とりあえずリモートアクセス用がメインですが、I2C とかを使って外部 IO をやってみたいと思っています。
- ② Debian の ARM に期待していること、お願いしたいことがあれば教えてください。
省電力な環境を生かして ReadOnlyboot(でも定期的にセキュリティ update はしたい) で放置できる環境とかが作れるとよいですね。

- ① 地図マシンです
- ② なし



- ① CuBox を持っている。eSATA 端子があるのでファイルサーバのバックアップをするマシンとして使おうとしたが、USB ポートから電源供給量が不足のため、HDD が動かず使えていない。そのため単なる armel バイナリのお試しマシンになっている。
- ② 新 Nexus7 で Debian が動いてほしい。

- ① お使いのマシンに ARM がありますか？
ないっす
- ② Debian の ARM に期待していること、お願いしたいことがあれば教えてください。
今のところないっす

- ① 昔売っていた B&N の NookColor を ARM 機材実験ボードとして未だに使ってます。この電子書籍端末は 2012/4 の東京エリア debian 勉強会, 2012/6 の大統一 debian 勉強会で喋ったとおり、特定ブートフォーマットの SD カードを用意して SD カードスロットへ入れちゃうと、うっかりそのままブートしちゃうという隠れ(?) 機能が大変便利です。おまけに連続 8 回起動失敗するとリカバリーが始まるという非文鎮化機能まで搭載しています。そのままでも pdf リーダ、ポータブル mp4 ビデオ/mp3 鑑賞機材として、また、debian ARM の native ブートの可能性を秘めた hack 機材としても楽しくお使いいただけます。
- ② NookColor 用の debian ネイティブブートが可能なイメージ (というかインストーラ) が欲しいとしてみるテスト。

- ① お使いのマシンに ARM はありますか？ もしあるのであれば、どのように使っているか教えてください。
たくさんある。開発用とかビルドマシンとして使っています。
Raspberry Pi は家のゲートウェイマシンとして動いています。
- ② Debian の ARM に期待していること、お願いしたいことがあれば教えてください。
マルチメディア系が弱いので整備して欲しい。

なし



まえだこうへい

- ① Armadillo J で自宅内の DHCP サーバとして使っています (not Debian)。OpenBlockS AX3 を、昨年夏ごろにカットとなって作った iori というツール (最近話題の docker みたいなもの) の開発環境として使っています。
- ② 今は特にはないですが、今後 ARM サーバの製品版が出てきた時に d-i でインストールできることでしょうか。



armmp



月刊 Deb-
helper



dh_strip

dh_strip の機能

- 実行バイナリ、共有ライブラリ、静的ライブラリから、デバッグシンボルを取り除く。
- 取り除いたデバッグシンボルをデバッガが見つけられるようにして、*-dbg パッケージが作りやすいようにビルドディレクトリ以下に配置する

どうやって*-dbgパッケージを作る？

そんなあなたに、

2012年 大統一Debian 勉強会
「 debug.debian.net 」の
プレゼン資料がおすすめ!

<http://gum.debian.or.jp/2012/>

とりあえず、すでに発表済みなので、ここでは省略。

dh_strip のコマンドラインオプション (その1)

- man debhelper に記載されている debhelper 共通オプション
他の月刊 debhelper の発表でなされているとおりなので、ここでは割愛。
- -Xitem, -exclude=item
item という名前を持つファイル名に対しては処理を行いません。複数指定したければ、“dh_strip -Xfoo1 -Xfoo2”と複数ならべて指定ができます。
- -dbg-package=package
*-dbg パッケージを作る際に、デバッグシンボルを格納するパッケージ名を指定する為に利用します。debhelper の COMPATIBILITY LEVEL によって振る舞いが変わります。(後述)

dh_strip のコマンドラインオプション (その2)

- -k
デバッグシンボルをバイナリパッケージに含めてしまいます。つまり、バイナリがインストールされると同時に、対応したデバッグシンボルも `/usr/lib/debug/` 以下にインストールされるようなパッケージを作成するときに使います。なお、`-dbg-package` も指定されると、`-dbg-package` の方の動作が優先されます。

dh_strip の COMPATIBILITY LEVEL の動作の違い (その1)

以下は-dbg-package 指定時の dh_strip の振る舞いの違い。

- 4 以下

“dh_strip -dbg-package=xxxx -dbg-package=yyyy” のように-dbg-package を複数指定する事が出来ます。ここで、こちらで指定した名前 (xxxx や、 yyyy) に合致するパッケージを処理する際、 “パッケージ名-dbg” というパッケージ名を*-dbg パッケージのパッケージ名として自動的に利用します。また、この COMPATIBILITY LEVEL の場合、ソースパッケージに含まれる debian/control には、これら*-dbg パッケージの為の記述を必要とはしていません。

dh_strip の COMPATIBILITY LEVEL の動作の違い (その2)

- 5 以上

-dbg-package は 1 つ指定するのが原則となります。もし複数指定した場合は、最初に指定された内容のみ利用されます。また、-dbg-package=xxxxx とすると、デバッグシンボルを格納するパッケージ名として、xxxxx がそのまま使われます。例えば、libfoo のパッケージと、foo パッケージを構築し、これらパッケージに含まれているバイナリのデバッグシンボルを foo-dbg パッケージに全部入れるには、-dbg-package=foo-dbg と指定します。また、ソースパッケージに含まれる debian/control には、生成予定の*-dbg パッケージの為の記述は必須となり、万一記載されていない場合はエラーとして扱われ、この場合は dh_strip はエラーを表示して終了します。

dh_strip の COMPATIBILITY LEVEL の動作の違い (その3)

以下は-k/-dbg-package 指定時の出来上がるデバッグシンボルファイルの違い

- 8 以下

“/usr/lib/debug/+バイナリのインストール先パス/+バイナリファイル名” に格納されるようにデバッグシンボルファイルが生成されます。また、デバッグに関する情報はコンパイラが生成したままの形で保存されるため、デバッグシンボルファイルのサイズは大きくなりがちです。

dh_strip の COMPATIBILITY LEVEL の動作の違い (その4)

- 9以上
バイナリが BuildID を含んでいる場合は、
“/usr/lib/debug/.build-id/BuildID の上位 2 桁/+BuildID
の残りの桁+.debug” に格納されるようにデバッグシンボル
ファイルが生成されます。 BuildID を含んでいなければ
COMPATIBILITY LEVEL 8 以下と同様のファイル名とな
ります。また、BuildID の有無にかかわらず、デバッグに
関する情報は zlib により圧縮されて格納されます。そのた
め、デバッグシンボルファイルのサイズはできるだけ小さ
くなるようになっています。

ところでBuildIDって

file コマンドでバイナリを調べると出てきます。

```
$ file /usr/bin/gst-launch
/usr/bin/gst-launch: ELF 64-bit LSB executable,
x86-64, version 1 (SYSV), dynamically linked
(uses shared libs), for GNU/Linux 2.6.26,
BuildID[sha1]=9b42db476118ab2b81041acd80e45e89e4289ea2,
stripped
```

詳しくは2013年大統一Debian勉強会「gdb+python拡張を使ったデバッグ手法」

(http://gum.debian.or.jp/2013/slide_data_list)に記載してます。

デバッグシンボルファイルの形式

Debian の i386/amd64 用 linux システムでは、デバッグシンボルファイルの形式としてほぼ DWARF なデバッグシンボルファイルが利用されています。DWARF は実行ファイルの形式である ELF フォーマットをベースに作られているため、DWARF を理解する時には、ELF もあわせて知っておくと良いです。

DWARF について:

- 本家
<http://www.dwarfstd.org/>
- 結構良い解説記事
<http://www.ibm.com/developerworks/jp/opensource/library/os-debugging/>
<http://ja.wikipedia.org/wiki/DWARF>
- DWARF を理解する為のお供に ELF もどうぞ!
http://ja.wikipedia.org/wiki/Executable_and_Linkable_Format

デバッグシンボルファイル覗いてみる

まず、俯瞰してみたいので、各ヘッダ、各セクションを俯瞰してみます。

```
$ sudo aptitude install binutils/sid php5-dbg
$ env LANG=C readelf -e /usr/lib/debug/usr/bin/php5
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00
  Class:                               ELF64
.. 中略...
  Start of program headers:            64 (bytes into file)
  Start of section headers:           19669080 (bytes into file)
... 中略...
Section Headers:
 [Nr] Name                               Type                               Address                               Offset
     Size                               EntSize                              Flags  Link  Info  Align
... 中略...
 [ 1] .interp                               NOBITS                             0000000000400238 00000238
     000000000000001c 0000000000000000  A      0    0    1
... 中略...
 [28] .debug_aranges                          PROGBITS                             0000000000000000 000002d0
     00000000000006190 0000000000000000           0    0    1
... 中略...
```

ここでは紙面がせまいので、皆さんお手元のdebianマシンで確認してみてくださいませ。

readelf -e の結果を図示してみる

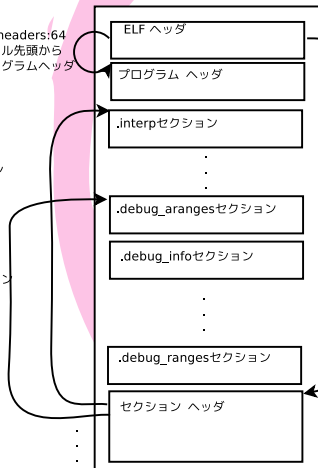
デバッグシンボルファイル：
/usr/lib/debug/usr/bin/php5

ファイルフォーマットとしては
elfが使われている。

ELFヘッダにて、
Start of program headers:64
とあるので、ファイル先頭から
64バイト目からプログラムヘッダ
が始まる。

セクションヘッダにて、
.interp はoffset 0x238
とあるので、ファイル先頭から
568バイト目から.interpセクション
が始まる。

セクションヘッダにて、
.debug_aranges はoffset 0x02d0
とあるので、ファイル先頭から
720バイト目から.debug_arangesセクション
が始まる。



ELFヘッダにて、
Start of section headers:19669080
とあるので、ファイル先頭から
19669080バイト目からセクションヘッダ
が始まる。

こちらあたりがデバッグ
用情報として使われる。
この情報の約束事がDWARF
となる。

ソースの情報を見してみる (その1)

.debug_info セクションに DWARF のとおりに格納されている。で、こちらのデコードには readelf -wi が便利。

```
$ env LANG=C readelf -wi /usr/lib/debug/usr/bin/php5
Contents of the .debug_info section:

Compilation Unit @ offset 0x0:
  Length:          0x15bf3 (32-bit)
  Version:         4
  Abbrev Offset:   0x0
  Pointer Size:    8
<0><b>: Abbrev Number: 1 (DW_TAG_compile_unit)
  <c>  DW_AT_producer      : (indirect string, offset: 0x1832):
      GNU C 4.8.1 -mtune=generic
      -march=x86-64 -g -O2 -fstack-protector -fPIC
      --param ssp-buffer-size=4
  <10> DW_AT_language     : 1          (ANSI C)
  <11> DW_AT_name         :
      (indirect string, offset: 0x1955): /tmp/buildd/php5-5.5.0+dfsg/ext/date/
      php_date.c
  <15> DW_AT_comp_dir    : (indirect string, offset: 0x6f0):
      /tmp/buildd/php5-5.5.0+dfsg/cli-build
```

ソースの情報を見してみる(その2)

readelf -wi の結果から、こちらの Compilation Unit(以下 CU) に記載されているソースファイル(以下 DW_AT_name) は、 /tmp/buildd/php5-5.5.0+dfsg/ext/date/php_date.c であり、コンパイルが行われたディレクトリ(以下 DW_AT_comp_dir) は /tmp/buildd/php5-5.5.0+dfsg/cli-build となります。

Use the SOURCE!(その1)

.debug_info の CU の DW_AT_name が絶対パスで記載されているような場合は、gdb で、set substitute-path に DW_AT_name の絶対パス部分を指定します具体例

2013年 大統一Debian 勉強会 「gdb+python 拡張を使ったデバッグ手法」のプレゼン資料

http://gum.debian.or.jp/2013/slide_data_list

Use the SOURCE!(その2)

.debug_info の CU の DW_AT_name が相対パス (あるいは、ファイル名そのもの) で記載されているような場合は、gdb で、set substitute-path には DW_AT_comp_dir の絶対パス部分を指定します。具体例

```
$ env LANG=C readelf -wi /usr/lib/debug/.build-id/9b/42db476118ab2b81041acd80
... 中略...
  <11> DW_AT_name      : (indirect string, offset: 0x57a): gst-run.c
  <15> DW_AT_comp_dir  : (indirect string, offset: 0x36b):
    /tmp/buildd/gstreamer0.10-0.10.36/tools
... 中略...
$ gdb --args gst-launch
(gdb) set substitute-path /tmp/buildd/ /home/yours/debian-src/gstreamer/
(gdb) b main
(gdb) run
(gdb) l
313     return candidates;
314   }
315
316   int
317   main (int argc, char **argv)
319     GHashTable *candidates;
320     gchar *dir;
```

Use the SOURCE!(その3)

`gdb` は、ソースの位置を求めるときに以下のアルゴリズムに従ってソースの位置を求めようとします。

- CU に記載されているファイル名が絶対パスの時:
そのままソースファイルの位置として扱う。また、この時 `DW_AT_comp_dir` の情報は無視される。
- CU に記載されているファイル名が相対パスの時:
ソースのファイル名を "`DW_AT_comp_dir` の値" + "/" + "`DW_AT_name`" として扱う。

なので、先ほどの通り `set substitute-path` の元ディレクトリ情報として指定するパス情報が異なります。

参考: COMPATIBILITY LEVEL9だと...(その1)

デバッグシンボルがCOMPATIBILITY LEVEL9の元で作成されていると...

```
$ sudo aptitude install libgstreamer0.10-0-dbg
$ env LANG=C readelf -e /usr/lib/debug/.build-id/9b/42db476118ab2b81041
acd80e45e89e4289ea2.debug
... 中略...
  [28] .zdebug_aranges   PROGBITS           0000000000000000    000002d0
          0000000000000002f 0000000000000000           0   0   1
  [29] .zdebug_info       PROGBITS           0000000000000000    000002ff
          00000000000000ccd 0000000000000000           0   0   1
... 中略...
```

全部、".z+debug+なんとか" にセクション名が変更されています。

参考: COMPATIBILITY LEVEL9だと...(その2)


.zdebug_info セクションをダンプしてみた。

```
$ env LANG=C readelf -x .zdebug_info /usr/lib/debug/.build-id/9b/42
db476118ab2b81041acd80e45e89e4289ea2.debug
Hex dump of section '.zdebug_info':
0x00000000 5a4c4942 00000000 00001b3f 789c7d59 ZLIB.....?x.}Y
0x00000010 7b7454c5 199fefee 6e76b3d9 2497cd7b {tT.....nv..${
0x00000020 370b2109 012e0482 243c13d8 84f01079 7.!......$<.....y
0x00000030 05437988 0887444f 44501e25 94a788b6 .Cy...DODP.%....
```

というわけで、ZLIBで圧縮されてます。

おわりに

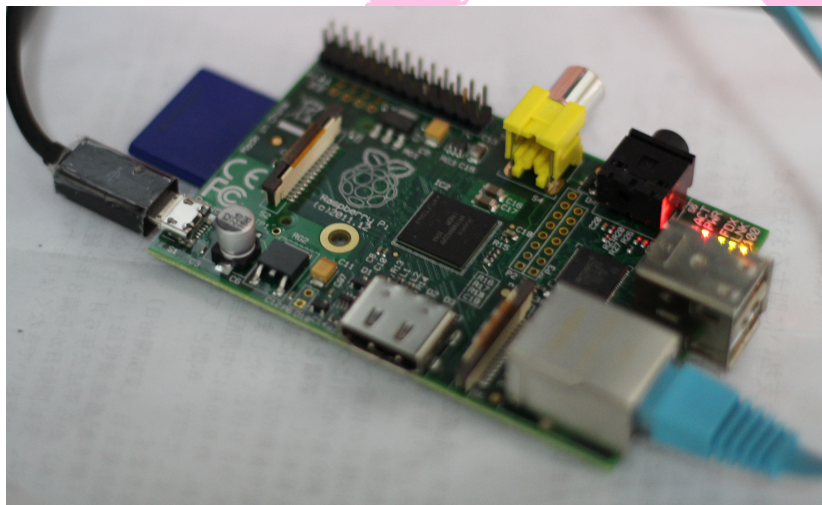
質問などおねがいしますー






raspberrypi

rasberry pi



- armel: armv4
 - raspbian: armv6 (VFP2)
 - armhf: armv7 (NEON)
- 

- SD-card にイメージを書き込んでそれから起動する

サブマシンを使う時の便利技

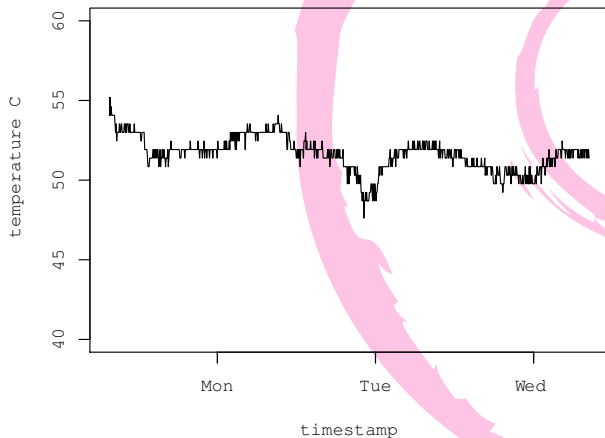
- mosh: ssh 張りっぱなしな感じにできて、ノートパソコンサスペンドしてもリジュームしたらまた繋がられる。
- avahi-daemon: DHCP の環境で面倒なことしなくてもホスト名指定でアクセスできるようになる。raspberrypi.local
- sshfs: ssh でつながるホストのディレクトリを sshfs を使って透過的にマウントできる。NFS でユーザID を調整したり SMB の設定を頑張ったりしていたのは過去のこと。

小技: sshfs

```
sshfs corei7.local:path/to/work ./mnt/
```


raspbian のセンサー

- CPU 温度センサーを使ってみた



時系列データを sqlite で保存

```
create table temperature(  
  timestamp TEXT default current_timestamp,  
  temperature INTEGER);
```

定期的にシェルスクリプトで温度を書き込み

```
while sleep 5m; do  
  temp=$(cat /sys/class/thermal/thermal_zone0/temp)  
  echo "insert into temperature(temperature) " \  
  "values($temp);" | \  
  sqlite3 sensor.db  
done
```

R でグラフ作成

タイムゾーン UTC の POSIXct 形式の文字列として解釈されるのでそれで処理してグラフ作成

```
library("RSQLite")
drv <- dbDriver("SQLite", max.con=1)
conn <- dbConnect(drv, dbname='sensor.db')
r <- dbGetQuery(conn, 'SELECT * from temperature')
r$posix <- as.POSIXct(r$timestamp, tz='UTC')
plot((temperature / 1000) ~ posix, r,
      xlab='timestamp',
      ylab='temperature C',
      ylim=c(40,60), type='l')
```

僕の raspberry pi の今後

多分こんなことします。

- 1000 円くらいで手に入る WebCam つけて監視システム
- 温度計つけて温度測定



今後のイベント

今後のイベント

- 2013年8月 Debian 勉強会



今日の宴会
場所

今日の宴会場所

未定

