第 72 回 東京エリア Debian 勉強会資料

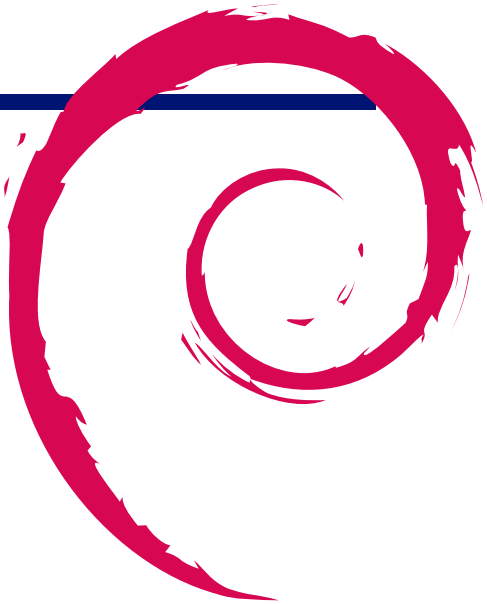
月刊 Deb専

日本唯一のDebian専門月刊誌

2011 年 1 月 15 日

特集1: キネクト

特集2: アンケートシステム



1 Introduction

上川 純一

今月の 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?

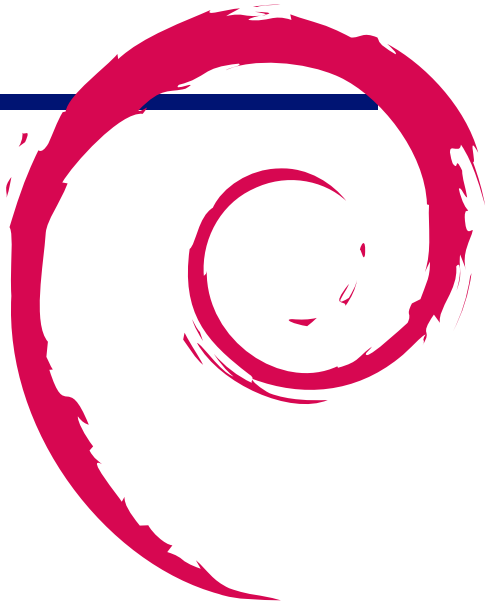
Debian 勉強会の目的は下記です。

- 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。

デビアン勉強会

目次



2 事前課題

上川 純一

今回の事前課題は以下です:

1. 2011 年の勉強会でなしとげたいことを宣言してください
2. 2015 年の Debian がどうなっているか大胆に妄想してください。

この課題に対して提出いただいた内容は以下です。

2.1 上川 純一

2.1.1 2011 年の勉強会でなしとげたいことを宣言してください

今年は LAMP スタック、 KVS など、クラウドインフラを Debian で一通り構築する仕組みを実現する方法について一通り勉強して関連したパッケージングにまつわる問題について整理したい。

2.1.2 2015 年の Debian がどうなっているか大胆に妄想してください。

Debian はまだ生き残っていて、 Squeeze+2 のリリース準備で忙しい。デバイスは携帯電話とタブレットデバイスのサポートが追加されており、メインストリームの CPU アーキテクチャは x86_64 と arm。

デバッグ用の情報がデフォルトで全部のパッケージについて提供される。

全 VCS を共通で利用できるようなインタフェースが整備され、 Debian のソースパッケージを標準的な VCS として提供されるようになる。

Debian のサーバリソースは P2P オーバレイネットワーク経由で提供され、共有のビルドサーバなどが利用できるようになっている。

2.2 まえだこうへい

2.2.1 2011 年の勉強会でなしとげたいことを宣言してください。

スマートフォン関連の Web アプリ用ライブラリとネイティブアプリ変換ツールなどの開発環境を Debian で整備する方法、何がどう優れているかを勉強したい。 Debian で環境を整える為にはどれが Debian パッケージになっているのか、どれは対応可能で、どれが不可能なのか。 Debian 上で生成したアプリを Debian から配信する、あるいは既存の配信の仕組みと連動するための仕組みはできないか。

2.2.2 2015 年の Debian がどうなっているか大胆に妄想してください。

意外と Wheezy の次のリリースが出てしまっている。スマートフォン向けのインストーラが登場し、 jail break や rooted しなくても、簡単に Debian をインストールできてしまう。 webOS が ipkg をやめて、 Debian パッケージを使うように先祖返りしている (webOS には起死回生ホームランで生き残っていてほしい)。そもそも 2015 年にはスマートフォンとかクラウドとかいう言葉は 消えて、また別の新しいバズワードに変わっているに違いない。

2.3 日比野 啓

2.3.1 2011 年の勉強会でなしとげたいことを宣言してください。

Debian 勉強会の常連に関数型言語の愛好者を一人でも多く増やす。関数型言語を使ったハンズオンとかやれたら楽しそう。

2.3.2 2015 年の Debian がどうなっているか大胆に妄想してください。

Debian 8.0 リリース?

2.4 キタハラ

2.4.1 2011 年の勉強会でなしとげたいことを宣言してください。

（ 東京エリアでの）勉強会に皆勤。もちろん無遅刻。

2.4.2 2015 年の Debian がどうなっているか大胆に妄想してください。

自らカーネルを作るようになっていたり、 AT コンパチの PC を飛び出し、省電力やクラスタ等の特別な機能を実現するハードウェアリファレンスを展開してたり、サポート請負会社が出来てたり、妄想は幾つか考えたのですが、実際は今とあまり変わらず淡々と開発が続けられているように思います。

2.5 henrich

2.5.1 2011 年の勉強会でなしとげたいことを宣言してください

成し遂げる目標は今のところありません…多少なりとも刺激が受けられればいいな、と考えています。

2.5.2 2015 年の Debian がどうなっているか大胆に妄想してください。

Hurd がついに { リリースされる, 終了宣言が出される }

2.6 村田信人

2.6.1 2011 年の勉強会でなしとげたいことを宣言してください

パッケージへの理解を深める。 Debian へのアップロードの流れを知る。

2.6.2 2015 年の Debian がどうなっているか大胆に妄想してください。

サーバ用のディストリビューションと言ったら Debian!になっている。

2.7 山田

2.7.1 2011 年の勉強会でなしとげたいことを宣言してください

- MiniConf の実現（ 勉強会的に最大のトピック? ）
- パッケージ作成の再勉強会をして、自己流を改める
- 手がけるパッケージを増やす＆自分で書いたのを配ってみる

2.7.2 2015 年の Debian がどうなっているか大胆に妄想してください。

- /var/lib/dpkg/available が行数換算で 50 万行を超え（ てさすがに重くて dpkg が超改良される（ といいな））
- 仮想化コンテナが超普及しつつ公式の超高機能 unionfs が遂に完成し、 apt-get で conflict したら即分岐して両方の環境を両立しつつ bind 統合できるようになる（ といいな）
- ファイルシステムのスナップショット・タイムマシン機能が普及して、誰も sid を使うことに懸念を感じなくなる（ といいな）

2.8 本庄

2.8.1 2011 年の勉強会でなしとげたいことを宣言してください

ずいぶん前ですが、依頼されたような気のする libsvm に関して発表したいです。

2.8.2 2015 年の Debian がどうなっているか大胆に妄想してください。

ちょっと想像つかないですね。

2.9 emasaka

2.9.1 2011 年の勉強会でなしとげたいことを宣言してください

ここは発表とかいうといいんでしょうか。

2.9.2 2015 年の Debian がどうなっているか大胆に妄想してください。

TV が CPU ばんばん積むようになって、そいつを hack して Debian が動いたりするとすごいですね。

2.10 higashiyama

2.10.1 2011 年の勉強会でなしとげたいことを宣言してください

今年は毎回参加して、ただ Debian を使う人からステップアップしたい

2.10.2 2015 年の Debian がどうなっているか大胆に妄想してください。

X がひっそりと消えていく

2.11 yamamoto

2.11.1 2011 年の勉強会でなしとげたいことを宣言してください

今年こそハッカーになる。

2.11.2 2015 年の Debian がどうなっているか大胆に妄想してください。

Debian が sid だけの、ローリングリリースになっており、stable のリリースは派生のディストリビューションに任せている。

2.12 kmuto

2.12.1 2011 年の勉強会でなしとげたいことを宣言してください

ますます忙しいんだけど、数回は出席したい。また講師できるようなテーマも作れるといいな。

2.12.2 2015 年の Debian がどうなっているか大胆に妄想してください。

（ wheezy ではなく） wheezy+1 のリリースエンジニアリング中、だよね? ? Debian を Ubuntu にマージすべきだというGR を出すかどうかで debian-vote がフレームに。

当日は CAcert assurer の仕事をやる予定。

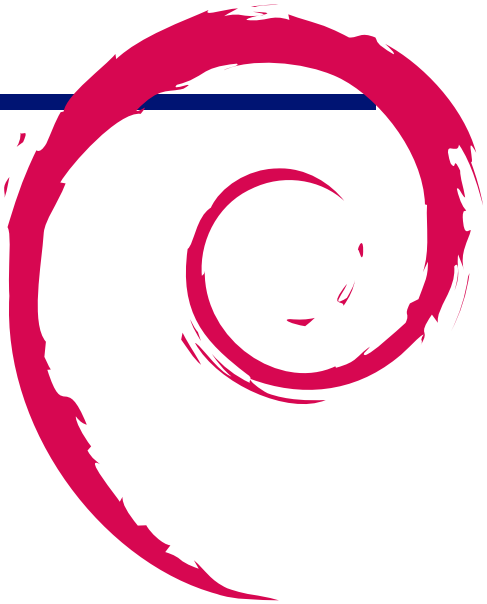
2.13 野島 貴英

2.13.1 2011 年の勉強会でなしとげたいことを宣言してください

3 つぐらいは何か発表してみたいと思いました。

2.13.2 2015 年の Debian がどうなっているか大胆に妄想してください。

1. Oracle Client ライブラリのサポート OS リストに Debian stable が入っている(本当に欲しい...)
2. X.org 以外の Window System が標準採用になっている。
3. 起動画面 (grub/gdm) や、ログインメニューがフルアニメーションしており、待機状態では普通に BGM が鳴っている。(ゲームのメニュー画面みたいな凝った演出が標準)
4. 通常のモニタは 3D モニタで利用されており、 kinect のような身体を使って操作するようなデバイスが入力デバイスとして標準対応。
5. Blue-Ray Disk4 枚組みの配布となる。
6. Mysql が upstream の最新版と同等となる。



3 最近の Debian 関連のミーティング報告

上川純一

3.1 東京エリア Debian 勉強会 71 回目報告

東京エリア Debian 勉強会。 2010 年 12 月の勉強会が荻窪にて開催されました。参加者は、松鵜さん、前田さん、山田 (泰) さん、山本浩之さん、 yyuu さん、野島さん、やまねさん、岩松さん、あらきやすひろさん、野首さん、本庄さん、吉田@板橋さん、上川の 14 人でした。

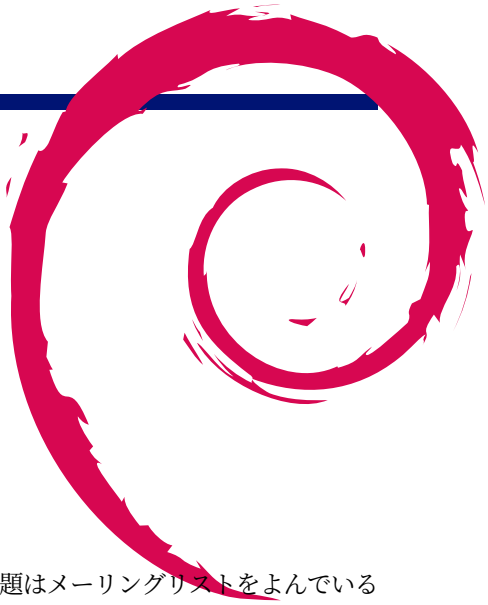
2010 年を振り返り、 2011 年にむけてやりたいことを提案しました。 2010 年 12 月分のプレゼン資料 PDF にまとめてありますのであとでご覧ください。

本庄さんが libsane について紹介しました。感想として、 Debian で libsane のコード書くの、簡単じゃん! 簡単なりにいろいろとハマりどころがあったようで、プレゼンテーションの軽妙な中身とともに楽しませていただきました。

山田さんが Kinect のハック進捗について簡単に紹介しました。来月までにはおもしろいデモができるそうです。

山田さんが CACert の認証について必要なことについて紹介しました。 CACert での Assurance(認証) を行うには Assurer により必要な内容が多数あるようで、大変そうでした。 2011 年 1 月は実際に認証をしてみようということになりました。

Debian 勉強会アンケートシステムの稼働開始と案内について。まぁ当日回答いただいた内容についてはシステムがまだまともにう動いていなかったようです先月回答全然もらえなかったのでまともにデバッグできてなかったよう。これは鋭意改善中。



4 Debian Trivia Quiz

上川 純一

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

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

問題 1. RC バグの現状ははどこで確認できるか

☐ A `http://bugs.debian.org/release-critical/`

☐ B `http://localhost/`

☐ C `http://debianmeeting.appspot.com/`

問題 2. Debian 勉強会予約システムの URL はどれか

☐ A `http://www.2ch.net/`

☐ B `http://atnd.org/events/`

☐ C `http://debianmeeting.appspot.com/`

問題 3. events@debian.org はどこと統合されたか

☐ A merchants@debian.org

☐ B hoge@debian.org

☐ C fuga@debin.org

問題 4. antiharassment@debian.org のうらにいないのは誰か

☐ A Amaya Rodrigo Sastre

☐ B Patty Langasek

☐ C Kouhei Maeda

問題 5. 現在いくつかの Sprint が開催され、企画されている。現在企画すらされていない sprint はどれか

☐ A -www sprint

☐ B security sprint

☐ C tokyo sprint

問題 6. DACA はどこを見ればよいか

☐ A `http://qa.debian.org/daca/`

☐ B `http://daca.debian.org/`

☐ C `file:/tmp`

問題 7. DEP は何の略か

☐ A Debian Enhancement Proposal

☐ B Device Enhancement Protocol

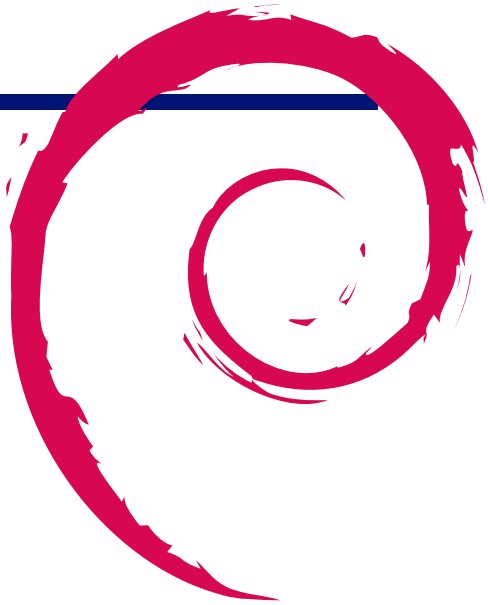
☐ C でっぷ

問題 8. DEP5 で提案されている debian/copyright の機械可読形式はどういうものか

☐ A S 式

☐ B RFC822 風

☐ C XML



5 Debian 勉強会予約システムにアンケート追加

上川 純一

5.1 アンケートシステムの目的

Debian 勉強会は各自のブログに記述するということが当初事後課題として機能していました。勉強会を開催したあと、参加者は勉強会で得られた内容を整理することができて、開催者は内容がどのようにうけとられたのかをある程度推し量ることができました。

しかし、近年勉強会の感想がブログに記述される率が低下してきています [1]。原因を想像するに、最近はコミュニケーションの道具としての twitter の流行などにともない、ブログに記述するという習慣が減ってきているからなのかもしれません。

時代の変遷によりブログエントリーがなくなるとその機能を代替するものは何でしょう。

ブログの機能としては、フィードバックのシステムと、あとその他のユーザに勉強会の記録を伝えるという面がありました。記録の面は twitter のログをまとめる togetter をなどを使えばよいだろうということが 2010 年 12 月の勉強会で提案され、それで問題は解決しそうです。残る課題はフィードバックシステムとしての機能です。

勉強会へのフィードバックシステムとしてアンケートシステムを提案します。

5.2 アンケートシステムの利用方法

5.2.1 管理者の事前準備

イベントの管理者はイベントの一覧画面からアンケート作成画面に遷移することができます。そこでは、ユーザに表示するための全体的なメッセージと、各項目を改行区切りで記述する欄があります。

ユーザにメールを送信すると、ユーザにはリンクの書いてあるメールが届きます。そのリンクをクリックすると、ユーザはアンケートに回答することができます。

5.2.2 ユーザ

ユーザは各項目については 0 から 5 までの値で回答することができます。そして、全体について自由記述の項目があります。[*1]

二つ目的があります。勉強会の企画の中で、どの部分が評価されているのかというのを出していきたいのと、任意なメッセージを入力できるようにすることで会に対してやりたいことなどの要望が伝えられるようにすることです。

[*1] 勉強会の会場では 5 択だと中心に回答が偏りがちなので、 4 択にしたほうがよいのではないかという意見が出ました。 Debian 勉強会アンケートシステムでは中心に偏り過ぎているとい傾向はまだ出ていないのでこのまま進めてみようと考えています。

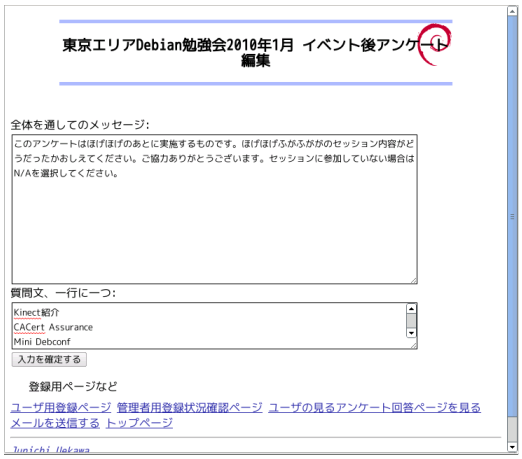


図 1　アンケート編集画面

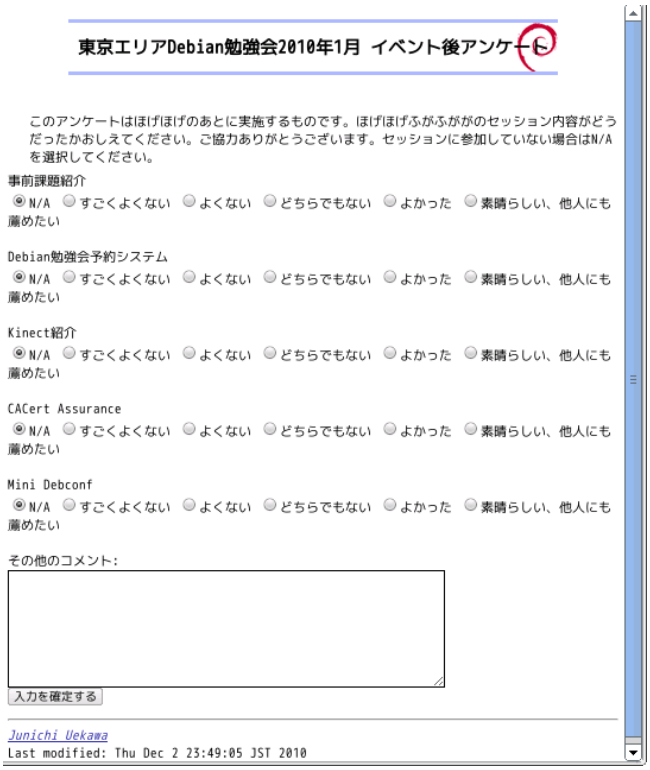


図 2　アンケート回答画面

表 1　アンケート回答結果の内部表現

0. N/A [*2]
1. すごくよくない
2. よくない
3. どちらでもない
4. よかった
5. 素晴らしい、他人にも薦めたい

5.2.3　管理者が事後に確認

管理者は、アンケートの [アンケート集計 CSV] リンクをたどることにより、イベント単位のデータの CSV ダンプを見ることができます。

特に集計処理はしていません。ここから先は CSV をとりだして処理することになります。

ここで、アンケートが匿名であるべきか記名であるべきかを最初考えていませんでした。どちらがどういう効果があるのかよくわかっていないので今後の課題としたいと思います。ブログ代わりなのであれば記名でよいと思うのですが、アンケートが記名だとバイアスがかかるというのであれば匿名のほうがよいかもしれません。システム的には今は匿名として扱っています。[*3]

[*3] Debian 勉強会の会場での議論では、回答の匿名性については。参加者のアンケート回答は匿名扱いにしたほうが答えにバイアスがかかりにくいだろうという意見が出ました。

5.3 アンケートシステムの設計と実装

アンケートシステムの設計と実装について見てみましょう。アンケートは Debian 勉強会予約システム [2] の一機能として実装しています。

5.3.1 データストア

ここの設計で悩んだ点は、 EventEnqueteResponse を Attendance と一緒にするべきかどうかです。 Datastore が join をサポートしていないためです。今回は分離して必要なときに手動で join[*4]するアプローチにしてみました。

```
class EventEnquete(db.Model):
    """Enquete questions for an event."""
    eventid = db.StringProperty()
    overall_message = db.TextProperty()
    question_text = db.StringListProperty()
    timestamp = db.DateTimeProperty(auto_now_add=True)

class EventEnqueteResponse(db.Model):
    """Enquete respnose for an event by one person."""
    eventid = db.StringProperty()
    # responses for 1-5 questions. 0 is N/A
    question_response = db.ListProperty(long)
    overall_comment = db.TextProperty() # a general comment from user.
    timestamp = db.DateTimeProperty(auto_now_add=True)
    user = db.UserProperty()
```

5.3.2 ユーザインタフェース

ユーザインタフェース部分は enquete.py に記述しています。メールや HTML ファイルの表示部分には、 Django のテンプレートを利用しています。

- EnqueteAdminEdit.html 管理者用のアンケート編集画面
- EnqueteAdminSendMail.txt 管理者が参加者にアンケートを依頼するメール送信するときのメールのテンプレート
- EnqueteAdminShowEnqueteResult.txt アンケートの結果を CSV 形式で表示する。
- EnqueteRespond.html 参加者がアンケートを返答する際に表示される HTML。
- EnqueteRespondDone.txt 参加者がアンケートを回答したときに送信される確認メール。

5.3.3 バックグラウンドタスク

ウェブインタフェースからアンケートの送信リクエストがあった場合、クリックした瞬間にメールの送信処理が発生するわけではなく、メールの送信処理に taskqueue を利用しています。

送信元は noreply@debianmeeting.appspotmail.com を使っています。多分ログインユーザの権限でメールを出すということができないのだと想定してこうなっています[*5]。

5.4 先月のアンケート集計結果

それでは、例として 2010 年 12 月のアンケート結果をみてみましょう。 0 は欠損値なので、 R に処理させる場合は「 NA」として処理します。これで R で処理した結果を見てみましょう。まだこの時点ではデータが少ないので、今回のなかでどのセッションが比較的ポジティブな感想を集めたのかということが分かるだけです。

事前準備として、統計解析ツール R[3] をインストールします。ここでは基本パッケージの r-base とドキュメントの r-doc-info[*6]をインストールしています。

```
# apt-get install r-base r-doc-info
```

[*4] 同じキーで二回別のテーブルを query することになるので必要なときには効率が悪いけど、あまり必要にならないと予想した設計

[*5] 未確認

[*6] R ではなにかわからないことがあればとりあえず info を見るか、 help() コマンドをつかえばよいかんじです。

R を起動して CSV ファイルをロードして、まず基本的な統計量を調べます。

```
$ R
[中略]
> enquete <- read.csv('201012enquete.csv')
> summary(enquete)
  事前課題紹介   X2010 年の Debian を振り返って.2011 年を企画する
 Min.   :3.00   Min.   :3
 1st Qu.:3.75   1st Qu.:3
 Median :4.00   Median :4
 Mean   :3.75   Mean   :4
 3rd Qu.:4.00   3rd Qu.:5
 Max.   :4.00   Max.   :5
 NA's   :1.00   NA's   :1
 CACert の準備に何が必要か 俺の libsane が火をふくぜ Debian.Miniconf. 企画
 Min.   :4.000            Min.   :4.000           Min.   :1.000
 1st Qu.:4.000            1st Qu.:4.500           1st Qu.:2.000
 Median :5.000            Median :5.000           Median :3.000
 Mean   :4.556            Mean   :4.714           Mean   :2.778
 3rd Qu.:5.000            3rd Qu.:5.000           3rd Qu.:4.000
 Max.   :5.000            Max.   :5.000           Max.   :4.000
                          NA's   :2.000
> quantile(enquete$俺の, na.rm=TRUE)
  0%  25%  50%  75% 100%
 4.0  4.5  5.0  5.0  5.0
> quantile(enquete$Debian)
  0%  25%  50%  75% 100%
   1    2    3    4    4
```

ざっと数字を眺めただけですが、今回特に評価の高かったセッションは sane ネタと、 CACert ネタで、評価の低かったセッションは Debian Miniconf 企画ネタでした。今回のアンケートの結果を受けると、 Debian miniconf についてはプレゼンテーションの方法を考え直した方がよいだろうと思います。事前課題資料が無かった、プレゼンテーションしようとしたマシンがプロジェクタにうまくつながらなかった、そもそも話の内容もあまり練られていなかったということを考えるとそれがそのままアンケートの統計値にも反映したと考えられます。

一つ目線を変えて相関を調べてみました。しかし、まだこれをどう判断するべきか考えがまとまっていません。

```
> cor(enquete$CAC, enquete$俺の, use="complete.obs")
[1] 0.7302967
> cor(enquete$俺の, enquete$事前, use="complete.obs")
[1] -0.2581989
> cor(enquete$CAC, enquete$事前, use="complete.obs")
[1] 0
```

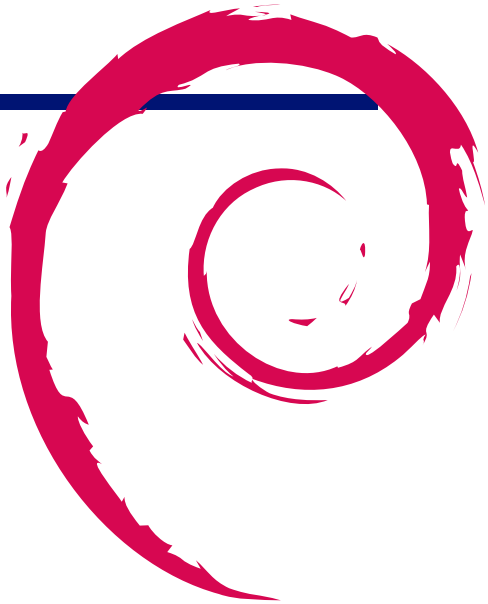
また、勉強会で議論した結果出てきた懸念点としては、点数のフィードバックは毎回出てこられても講師のやる気がそがれるなどの悪影響がある可能性がある点でした。講師には一年に一回程度まとめて返すのがよいだろうという提案がありました。

参考文献

[1] 上川純一,「 2010 年の東京エリア Debian 勉強会をふりかえって」東京エリア Debian 勉強会 Deb 専 2010 年 12 月号

[2] 上川純一,「 東京エリア Debian 勉強会予約システムの構想」東京エリア Debian 勉強会 Deb 専 2010 年 2 月号

[3] R Development Core Team, "R: A Language and Environment for Statistical Computing", R Foundation for Statistical Computing, Vienna, Austria, 2008 http://www.R-project.org



6 Kinect を Debian で使ってみた

山田 泰資

6.1 Kinect って何?

Kinect は Microsoft が 2010/11 に XBox360 の新しい操作デバイスとして発売したゲームデバイスで、強力な画像認識機能を持ち、一切コントローラなどを手にしないゲームプレイを実現します。

ハードウェアとしての最大の特徴は、画像認識と言っても単なるカメラではなく、深度センサを搭載していることです。この種の深度センサ付カメラは従来もあったのですが、 Kinect は大量生産されることもあって価格が 1 万 4 千円前後と従来と比べてまさに桁違いの安さで、このため PC などの周辺機器として流用できないかと発売前から高い注目を集めました。

6.2 発売、そして Kinect ハックへ

そして発売されると普通に USB 接続ができることが判明(単品販売品のみ。 XBox360 同梱版はケーブルを加工する必要あり) し、すかさず画像・深度データのキャプチャ方法が解析されます。こうしてまず生まれたのが libfreenect です。libfreenect は急速に開発が進み、

1. カメラ(通常、深度、赤外線) 画像の取得
2. カメラのチルト制御
3. 加速度センサの読み出し
4. 筐体 LED の制御

とサウンド関係以外はほぼ制御できるようになり、各種言語へのラッパーが作られたり OpenCV などの既成の画像認識ライブラリと組み合わせてデモが作成されるなど、あっというまに普及が進みました。ここまでが発売後 1 ヶ月程の話になります。

しかし、 libfreenect だけでは限界がありました。 XBox360 では kinect を使ったゼスチャ認識などの高度な機能を持ったアプリケーション(ゲーム) を作成できる環境が用意されているのですが、 libfreenect はあくまでハードウェアにアクセスするまでなので、その部分が欠けていたのです。

実は前評判では「本体側の負荷を下げるため、 Kinect 側でジェスチャ認識などの高水準認識も分担する能力がある」という話もあったのですが、最終的には深度センサまでがハードウェアで、残りは XBox360 向けのソフトウェアライブラリで実現する形となっていたのでした。

しかし、ここで新展開が起こります。 Kinect の深度センサと上記ライブラリの原型提供を行った PrimeSense 社から、

1. Linux, Windows 用の Kinect 搭載センサ用ドライバを OSS として公開する
2. ゼスチャ認識機能を備えた OpenNI フレームワークを OSS として公開する

3. 骨格認識および高度なジェスチャ認識を行う NITE ライブラリを無償提供する[*7]

という発表が行われ、純正の XBox360 開発環境と遜色がなくなり、人気がさらに高まりました。

これを生かして年末にかけては 3D モデリングツール（ MikuMikuDance）にモーションキャプチャエンジンとして組み込んだり、年明けには AR ウルトラセブン（ 画面中の自分の姿にウルトラセブンのコスプレをオーバレイし、ジェスチャにあわせて様々な必殺技を繰り出すことができる）が登場するなど、日本では現在進行形で爆発的な盛り上がりを見せています。

ここでは OpenNI および NITE の利用方法を紹介しつつ、それを元に Debian 勉強会のプレゼンテーションをジェスチャで行う支援ツールの作成までを紹介します。

6.3 OpenNI と NITE の導入

まずはインストールです。

1. 独自に拡張が始まっている OpenNI とドライバ (SensorKinect) は github から
2. NITE はバイナリのため PrimeSense 社サイト (http://www.openni.org/) から

それぞれ入手します。

現段階ではバージョンの組み合わせによっては動作しないことがあり、ここでは以下の組み合わせで導入します:

1. OpenNI (stable) / https://github.com/OpenNI/OpenNI.git
2. SensorKinect (avin2 version) / https://github.com/avin2/SensorKinect.git
3. NITE-Bin-Ubuntu-x64-1.3.0.17.tar.bz2

いくつかのパッケージを入れるなどビルド環境を整えておく必要はあります[*8]が、後は付属のドキュメントの通りに進めればトラブルなく導入できました。

まずは OpenNI 本体をインストールします:

```
$ git-clone https://github.com/OpenNI/OpenNI.git
$ cd OpenNI/Platform/Linux-x86/CreateRedist
$ ./RedistMaker # ビルドおよびインストール用バイナリセットを作ります
...
$ cd ../Redist
$ pwd
/d/src/openni/OpenNI/Platform/Linux-x86/Redist
$ sudo ./install.sh
copying shared libraries...OK
copying executables...OK
copying include files...OK
creating database directory...OK
registering module 'libnimMockNodes.so'...OK
registering module 'libnimCodecs.so'...OK
registering module 'libnimRecorder.so'...OK

*** DONE ***
$
```

次にドライバ（ といっても、 Linux 版のものは単なる共有ライブラリです）：

[*7] OSS ではなく、無償利用のためのライセンスコードが公開された

[*8] apt-get install gcc python2.6 libusb-1.0-0-dev freeglut3-dev doxygen graphviz あたりで整うかと思います

```
$ git-clone https://github.com/avin2/SensorKinect.git
$ cd SensorKinect/Platform/Linux-x86/CreateRedist
$ ./RedistMaker # ビルドおよびインストール用バイナリセットを作ります
...
$ cd ../Redist
$ pwd
/d/src/openni/SensorKinect/Platform/Linux-x86/Redist
$ sudo ./install.sh
creating config dir /usr/etc/primesense...OK
copying shared libraries...OK
copying executables...OK
registering module 'libXnDeviceSensorV2.so' with OpenNI...OK
registering module 'libXnDeviceFile.so' with OpenNI...OK
copying server config file...OK
setting uid of server...OK
creating server logs dir...OK
installing usb rules...OK

*** DONE ***
$
```

そして最後に NITE のインストールです:

```
$ tar xf NITE-Bin-Ubuntu-x86-1.3.0.17.tar.bz2 -C /d/src/openni/
$ cd Nite-1.3.0.17
$ sudo ./install.bash -l=0KOIk2JeIBYClPWVnMoRKn5cdY4=
... バイナリを/usr/lib 等にコピーした後、サンプルコードをビルドする...
$
```

最後の NITE の導入で指定した「 0KOIk2JeIBYClPWVnMoRKn5cdY4=」が無料利用のための公開ライセンスキーになります。

これらのインストールが終わると

1. /usr/lib に libnim*.so, libXn*.so, libXnV*.so
2. /usr/include/ni,nite に各種ヘッダファイル
3. /usr/bin に niReg, niLicense コマンド(モジュールやライセンスの管理用)
4. /var/lib/ni にモジュールやライセンスの管理用 XML ファイル
5. /usr/etc/primesense に OpenNI やモジュールの設定・データファイル

という構成になります。/usr/etc/primesense は LFS 的におかしな位置のため、/var/lib/ni/modules.xml から参照しているパス設定を変更の上、/etc に移動させるとよいかもしれません。[*9]

あと、サンプルコードを動かせるよう一部ファイルを修正しておきましょう。 NITE を展開した Nite-1.3.0.17/Data/ 下の XML ファイルを以下の内容にします。ライセンスキーの書き込みと、キャプチャ設定の修正・追加になります。

Sample-Scene.xml の内容:

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="true" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="false"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="Scene" />
  </ProductionNodes>
</OpenNI>
```

Sample-Tracking.xml の内容:

[*9] OpenNI のパッケージを作成予定ですが、このあたりは検証後直す予定

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="false" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="true"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth" name="Depth1">
      <Configuration>
      <MapOutputMode xRes="640" yRes="480" FPS="30"/>
      <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="Gesture" />
    <Node type="Hands" />
  </ProductionNodes>
</OpenNI>
```

Sample-User.xml の内容:

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="true" writeToFile="false">
    <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
    <LogLevel value="3"/>
    <Masks>
    <Mask name="ALL" on="false"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
    <Node type="Depth" name="Depth1">
      <Configuration>
      <MapOutputMode xRes="640" yRes="480" FPS="30"/>
      <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="User" />
  </ProductionNodes>
</OpenNI>
```

これらの内容の意味については次項で解説します。

6.4 Kinect を叩いてみる - OpenNI を使ってみよう

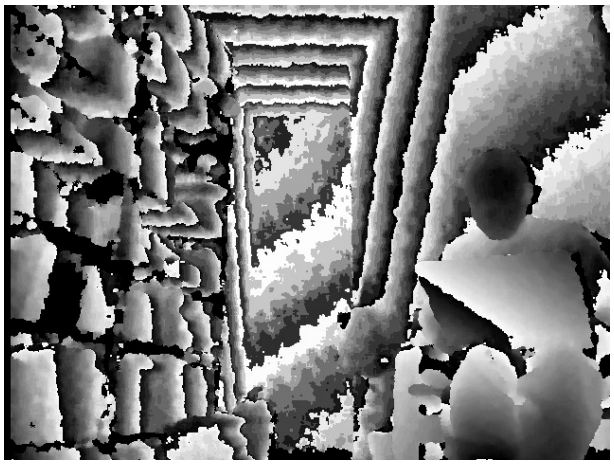
それでは OpenNI から使ってみましょう。以下はサンプルを少しいじって、深度カメラを叩いて深度情報を取得し、それを PPM 画像の形でダンプするようにしてみたものです。

```
/*BINFMTCXX: -Wall -I/usr/include/ni -I/usr/include/nite -lOpenNI
 *
 * 深度情報のキャプチャを行い、 PGM 画像として保存する。
 * カメラからの距離がピクセルの濃淡で表される画像になる。
 *
 */

#include <XnCppWrapper.h>

#define log(...) do { \
        fprintf(stderr, __VA_ARGS__); fprintf(stderr, "\n");    \
    } while (0)

#define die(...) do { \
        fprintf(stderr, __VA_ARGS__); fprintf(stderr, "\n"); exit(1);   \
    } while (0)

// 深度センサが返すデータは深度が 16bit 値で左上から並んでいるので、
// そのまま 16bit PGM としてダンプすれば画像になる。
void
ppmsave(xn::DepthMetaData& meta, const char *outfile) {
    const XnDepthPixel* pix = meta.Data();

    XnUInt32 xmax = meta.XRes();
    XnUInt32 ymax = meta.YRes();

    // dump 16bit depth data as 16bit PGM
    FILE* out = fopen(outfile, "w+");
    fprintf(out, "P5\n%d %d 65535\n", xmax, ymax);
    fwrite(pix, sizeof(XnDepthPixel), xmax * ymax, out);
    fclose(out);
}

int
main(int argc, char **argv) {
    xn::Context context;
    xn::EnumerationErrors errors;
    xn::DepthGenerator dgen;
    xn::DepthMetaData meta;

    if (argc < 3) {
        die("Usage: %s kinect.xml depth.pgm", argv[0]);
    }

    // XML 構成ファイルでの初期化、深度カメラへの参照取得、そしてキャプチャ
    context.InitFromXmlFile(argv[1], &errors);
    context.FindExistingNode(XN_NODE_TYPE_DEPTH, dgen);
    context.WaitOneUpdateAll(dgen);

    // キャプチャデータの取得と保存
    dgen.GetMetaData(meta);
    ppmsave(meta, argv[2]);

    context.Shutdown();
    return 0;
}
```

これを実行すると、

```
\$ ./camera3d.cc kinect.xml depth.pgm
```

次のように各ピクセルの深さが画像として取れます:



正直何がなんだか? という画像ですが、右側にいるのがノート PC を膝において座っている私です。

ここでの中核部分は以下の３行です。

```
context.InitFromXmlFile(argv[1], &errors);
context.FindExistingNode(XN_NODE_TYPE_DEPTH, dgen);
context.WaitOneUpdateAll(dgen);
```

OpenNI ではシステム中にどのようなデバイスが存在するか、また、それらの構成（ カメラなら解像度など）はどうなるかを外部の XML で記述でき、それを元に初期化を行います。コードだけでも書くことはできるのですが、その場合は構成を変更するとコードの書き換えが必要になります。

このコンテキスト（ context）を作成し、そこにデバイスやフィルタ（ 未登場ですが、センサデータの入力を受けてゼスチャ認識を行い、登録されているフックをトリガするものなどがあります）といった OpenNI モジュール群を接続する、というのが OpenNI/NITE のプログラムフローになります。

最後に、上で利用した kinect.xml も引用しておきます:

```
<OpenNI>
  <!-- ログ設定。以下の設定で標準出力は使わず Log/*.log に書く -->
  <Log writeToConsole="false" writeToFile="true">
    <LogLevel value="0"/>
    <Masks>
      <Mask name="ALL" on="true"/>
    </Masks>
    <Dumps />
  </Log>

  <ProductionNodes>
    <Node type="Image" name="Kin2D">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>

    <Node type="Depth" name="Kin3D">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>

    <Node type="Scene" />
    <Node type="User" />
    <Node type="Gesture" />
    <Node type="Hands" />
  </ProductionNodes>
</OpenNI>
```

以後の例もすべてこの XML を使って動かしています。

6.5 ジェスチャの検出方法 - NITE を使う

それでは次はジェスチャの認識をさせてみましょう。こちらでは NITE ライブラリを利用します。

これも OpenNI フレームワーク上のプログラミングなのでコンテキストを生成する点は同じですが、 NITE ではコンテキストを介した深度センサデータのやりとりのような低レベルの操作は隠され、

1. ジェスチャー操作の流れを管理する、コンテキストをラップするセッションマネージャ
2. セッションマネージャに登録される各種のジェスチャー検出器
3. それら検出器に登録される、操作検知時に呼び出されるコールバック

という構造のコードになります。操作中・操作終了をセッションマネージャが判定し、「 セッション中」の動作のみが検出器にかかるよう調整しています。

実際にどのようなものになるか、見てみましょう:

```
/*BINFMTCXX: -Wall -I/usr/include/ni -I/usr/include/nite -lXnVNite -lOpenNI
 *
 * 簡単なジェスチャ認識のテスト。
 * 手を振ってセッション開始すると、プッシュとスワイプ動作を認識する。
 *
 */

#include <XnCppWrapper.h>
#include <XnVNite.h>

/**************************************************************************
 * Utility Functions
 **************************************************************************/

#define log(...) do {                       \
        fprintf(stderr, __VA_ARGS__);       \
        fprintf(stderr, "\n");              \
    } while (0)

#define die(...) do {                       \
        fprintf(stderr, __VA_ARGS__);       \
        fprintf(stderr, "\n");              \
        exit(1);                            \
    } while (0)

/**************************************************************************
 * セッションマネージャや検出器に登録するコールバック関数
 * 現在は個々のジェスチャーを認識した等のログを出すのみ。
 **************************************************************************/

static void
OnSessionStart(const XnPoint3D &p, void *data) {
    log("session start");
}

static void
OnSessionEnd(void *data) {
    log("session end");
}

static void
OnFocusStartDetected(const XnChar *focus,
                     const XnPoint3D &p, XnFloat progress, void *data) {
    log("focus start");
}

static void
OnSwipe(XnVDirection dir, XnFloat speed, XnFloat angle, void *data) {
    log("swipe detected. dir=%d, speed=%f, angle=%f", dir, speed, angle);
}

static void
OnPush(XnFloat speed, XnFloat angle, void *data) {
    log("push detected. speed=%f, angle=%f", speed, angle);
}

/**************************************************************************
 * Main
 **************************************************************************/

int
main(int argc, char **argv) {
    xn::Context context;

    if (argc < 2) {
        die("Usage: %s kinect.xml", argv[0]);
    }

    context.InitFromXmlFile(argv[1]);

    // セッションマネージャと各種検出器の生成
    XnVSessionManager* sm = new XnVSessionManager;
    XnVSwipeDetector*  sd = new XnVSwipeDetector;
    XnVPushDetector*   pd = new XnVPushDetector;

    // セッションマネージャの初期化
    //
    // 手を振って( Wave) セッションに入り、休止状態からのセッション再開は
    // 手を上げる( RaiseHand) 。セッション中は下で登録された検出器が操作を
    // 検出するようユーザの動きをそれらに伝達する。
    sm->Initialize(&context, "Wave", "RaiseHand");
    sm->RegisterSession(NULL,
                        OnSessionStart, OnSessionEnd, OnFocusStartDetected);

    // ジェスチャー検出器と検知した際のコールバックを登録
    sm->AddListener(sd);
    sm->AddListener(pd);
    sd->RegisterSwipe(NULL, OnSwipe);
    pd->RegisterPush(NULL, OnPush);

    // 後はずっと入力を待ってはイベント処理するループ
    context.StartGeneratingAll();
    for (;;) {
        context.WaitAndUpdateAll();
        sm->Update(&context);
    }
    return 0;
}
```

最初の深度データの画像を保存する例よりも簡単になりました。上では XnVSwipeDetector と XnVPushDetector の２種類だけを使いましたが、 NITE には 10 を超える様々な基本ジェスチャーの検出器があり、また、複数のジェスチャーを組み合わせた操作に対応するための複合処理などを行う補助クラスも用意されています。

このように、多種多様な検出器を用意して、煩雑な体の各部の動きのトラッキングや照合の手間を NITE は省いてくれます。

各種の操作と認識率

なお、動かしてみるとわかりますが、動作によって「きちんと認識される」度が結構違います。スワイプは「一瞬手を止めて、それからさっと流す」動作なのですが非常に認識率が悪いです。一方、プッシュ操作はほぼ確実に認識されます。プッシュ（ Push, Click）動作は手振り（ Wave）動作と並んで「セッション開始のシグナル動作（ focus gesture）」として実用的とマニュアルに書かれていますが、よくわかります。快適な操作感のためには結局カスタムの検出器を作成する必要がありそうです。

6.6 アプリケーションの制御方法 - XTest の話

さて、これまで OpenNI, NITE と例を出してきましたが、次はこれで認識したジェスチャでどう他のアプリケーションを制御するかです。今回は X11 の xtst(XTest) 拡張でマウスやキーイベントを送り、それで（ この資料の）プレゼンテーションが行えるようにします。

XTest 自体の使い方は以下のようになります。

```
/*BINFMTCXX:-Wall -lXtst -lX11
 * XTest を使ったキーとマウスの制御テスト
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/extensions/XTest.h>

#define log(...) fprintf(stderr, __VA_ARGS__)
#define die(...) do { fprintf(stderr, __VA_ARGS__); exit(1); } while (0)

enum { DOWN = 1, UP, DOWNUP };

void
sendkey(Display *disp, unsigned int keysym, int mask) {
    unsigned int kc = XKeysymToKeycode(disp, keysym);
    if (mask & DOWN) { XTestFakeKeyEvent(disp, kc, True, CurrentTime); }
    if (mask & UP)   { XTestFakeKeyEvent(disp, kc, False, CurrentTime); }
    XSync(disp, False);
}

int
main(int argc, char **argv) {
    Display *disp;

    if ((disp = XOpenDisplay(NULL)) == NULL) {
        die("Unable to open display\n");
    }

    // キー入力のテスト - とりあえず ls させてみる
    unsigned int key[] = { XK_L, XK_S, XK_Return };
    for (int i = 0; i < (int)(sizeof(key) / sizeof(*key)); i++) {
        sendkey(disp, key[i], DOWNUP);
    }

    // 相対位置移動
    XTestFakeRelativeMotionEvent(disp, -100, -100, CurrentTime);
    XSync(disp, False);
    sleep(3);

    // 絶対位置移動
    XTestFakeMotionEvent(disp, -1, 100, 100, CurrentTime);
    XSync(disp, False);
    sleep(3);

    // XTest の FakeMotion でなぜか XMing 上のポインターが動かないので、
    // XWarpPointer を使ってみるテスト・・・これだと動くようだ。
    XWarpPointer(disp, None, None, 0, 0, 0, 0, 200, 200);
    XSync(disp, False);
    sleep(3);

    // Control 押しながら右クリックを実行( メニューが出るか確認)
    sendkey(disp, XK_Control_L, DOWN);
    XTestFakeButtonEvent(disp, 3, True,  CurrentTime);
    XTestFakeButtonEvent(disp, 3, False, CurrentTime);
    sendkey(disp, XK_Control_L, UP);
    XSync(disp, False);

    // 終了
    XTestDiscard(disp);
    XCloseDisplay(disp);

    return 0;
}
```

操作したいアプリケーションに合わせて

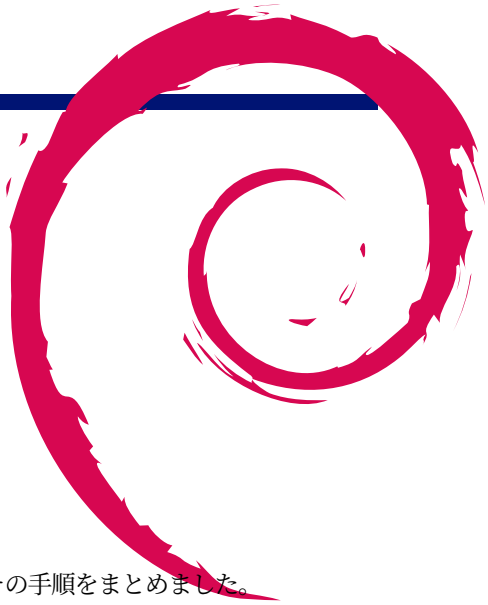
1. XTestFakeKeyEvent
2. XTestFakeButtonEvent
3. XTestFakeMotionEvent
4. XTestFakeRelativeMotionEvent

で操作イベントを送りながら、 XSync で同期を取る(ポインタを移動させて入力、のような操作で、移動が完了してから入力がされるようにする）というのが基本的な処理になります。

あとはこれを先の OpenNI/NITE のジェスチャ検知をトリガに発行してやれば、今回のデモは完成になります。

6.7 デモ

さて当日うまくいくか・・・



7 CACert Assurance

山田 泰資

先月の CAcert 報告に引き続き、今回は CAcert の assurance 会[*10]を行うので、その手順をまとめました。

また、手順について cacert.org の ML で質問していたところ、色々と話が転がって、 3/4-5 のオープンソースカンファレンス(OSC2011 Tokyo/Spring @ 早大) で、関係者が来日の上、 CAcert の公式イベントとして CAcert ATE (Assurer Training Event) を開催できることになりました。日本初となります。これについても報告&協力募集します。

7.1 今日の手順

今日は練習を兼ねたミニサイン会を行います。必要な資材は用意してあるので、今日参加された方は政府発行の身分証明書があり、 cacert.org にアカウントがあればポイントを受けられます。

以下、手順:

1. 認証を受けたい方は、 CAP Form に以下の内容をまず記入して下さい
   (a) 表面は、 **署名および Assurer 用フィールド以外の各欄 に記入。名前や誕生日はローマ字と西暦で**
   (b) 裏面は、白紙部分に名前(漢字&カタカナ) と和暦での誕生日を補記として記入[*11]
2. Assurer は、以下の確認を行って下さい
   (a) CCA(CAcert Community Agreement) [*12]に同意することの口頭確認、および、その旨のチェックボックスにチェックがあること
   (b) CAP Form の内容がすべて記入されており、また、提示された身分証明書の通りであること
   (c) 身分証明書が真正であり、内容が以下を満たすこと[*13]
      i. 誕生日記載の政府発行の物が 1 つはあること
      ii. 写真確認できるものが 1 つはあること
3. 最後に、 **CAP Form に目の前で署名をし、そのまま Assurer に渡して別れて下さい。署名がないと無効です!**
4. **Assurer は、以下の追加作業を行って下さい**
   (a) CAP Form に Assurer として署名し、それを最低 7 年間保管する
   (b) CAP Form 記載のメールアドレスを使って http://cacert.org で検索し、各人にポイントを付与する

ポイントが 100pt を超えた方、超えそうな方は https://cats.cacert.org/ のオンライン試験を通しておくと、 100pt 以上になった時点から他の人を認証できるようになります。なお、アクセスには CAcert 発行のクライアント証明書が必要です。

[*10] 認証会? サイン会? もう GPG と合わせてサイン会でいいか?
[*11] これは来る OSC2011 で外国の方に認証してもらうための手順として策定中の方法です
[*12] CAcert に参加するにあたっての遵守事項
[*13] 可能ならば紫外灯などでの偽造チェックや、他の証明書(政府・民間発行は問わない) での追加確認をします

7.2 CAcert@OSC2011 の計画

元々 OSC2011 の Debian 枠の空き時間で GPG キーサイン兼 CAcert サイン会を企画していましたが、冒頭の通り cacert.org の方より打診があり、

> 日本初の CAcert 公式トレーニングイベント（ ATE Tokyo）

を開催できることになりました。

ATE(Assurance Training Event) というのは通常のサイン会とは若干異なり、主に Assurer になる・なれる人を対象にした、上位 Assurer 養成（ & CAcert の信頼性向上）のためのサイン会を兼ねたトレーニングイベントです。

たまたま期間中に来日する cacert.org の方（ Peter Yuill 氏）がおり、 OSC2011 の枠も別途確保できたため、日本での CAcert 立ち上げの好機として緊急で開催決定となりました。

検討中の日程は以下の通りです:

1. 3/5 昼に OSC2011 の正式セッションとしてトレーニング講座を実施する（ サインも残り時間やブースで実施）
2. その上で、 3/5 夕刻から追加サイン会を実施する。実施場所・時間は以下の 2 案で検討中

（ a） 近隣の会議スペースや会場の空きスペースでサイン会をする

（ b） 飲み屋やレストランのスペースを借りて、その中でサイン会をする

OSC2011 は 17:00 に終了なので、サイン会は前者なら 17:00-19:00、後者なら 18:00-エンドレス (酔っ払うまで）で考えています。飲み会内開催案は

> 「 酔っ払いそうだが大丈夫か? 」「 大丈夫だ、問題ない」[*14]

と、先にサインをするなど手順を工夫すれば、場所や時間の確保問題をクリアする案としてよいかもという事です（ ドイツの ATE でもやってみるとか）。

なお、準備のため、特に運営やサインに協力して下さる方を絶賛大募集中です。現在の準備状況は以下の通り:

必要なもの	準備状況
当日参加できる Assurer の方	未調整（ 来場者に 100pt 渡せるとベスト）
当日参加できる運営の方	未調整（ Assurer 担当と別に、 CAP Form 確認や引率に数名）
場外会場の確保	未了（ 隣接の区のスポーツセンタの優先権がある新宿区の人求む）
事前告知&申込ページの用意	未了（ wiki.cacert.org で行う? atnd 等を使える? ）
CAP Form や CCA などの紙資料	準備可能（ 各 100 部程度なら市の印刷機で激安）
ATE プレゼンや CCA の翻訳	未了（ これが次の山か? 頑張ります・・・ ）
案内チラシ（ 場所案内など）	未了（ 内容が未作成。印刷は可能）
ブラックライト（ Assurer 人数分）	未了（ 製作計画中）
プリンタ（ 紙資料印刷用）	準備 OK （ 持って行けます）
プロジェクタ（ 場外でプレゼンの場合）	準備 OK （ 持って行けます）
終了後レポート	未了（ 実施レポートを cacert.org に出すまでが公式イベント、らしい。これはやります）

なお、開催規模ですが、 CAcert の方には、「 Assurer または 100pt 近い Assurer 見込み」の状態の ATE 参加者は 10 名内外、また、その中で 150pt 達成済みの ExperiencedAssurer は数名だろうという見込みを伝えています。

[*14] 意訳

Debian 勉強会資料

2011 年 1 月 15 日　　初版第 1 刷発行

東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）