
体験版
うぶんちゅ まがじん ざっぱ～ん♪ vol.5

うぶんちゅ！　まがじん ざっぱ〜ん♪
Vol.5 体験版

team zpn　著

2016.08.21 版　　発行

Chap. 1

続 OpenNebula で PCI passthrough

おおたあきひこ

青羽家は OpenNebula で家庭内クラウドを構築し、ここなちゃんとママの二人で計算機リソースをやりくりしています。

ここなちゃんは 14 歳の誕生日に以前から欲しかった Intel の X520-DA2 10GbE NIC を買ってもらい、OpenNebula の仮想マシンテンプレートに RAW セクションを書いて、ゆるふわっぽく仮想マシンに PCI passthrough していました。

でもこの方法ではどのホストのどの PCI デバイスをどの仮想マシンが使用しているのか OpenNebula は管理できません。「X520-DA2 を SR-IOV してママも使ってみたいわ」という要望にも応えにくく、ここなちゃんは大弱りです。

1.1 OpenNebula ノススメ セカンドシーズン

「あの」

はい何でしょうここなちゃん。

「ざっぱ〜ん vol.3 でママに買ってもらったのは Mellanox の ConnectX-3 InfiniBand HCA だったような気がしますけど」

ごめんなさい原稿用に InfiniBand HCA はやっぱり高くて買えませんでした。その代わり Intel X520-DA2 を用意したのでこれで我慢してください。

「えー」

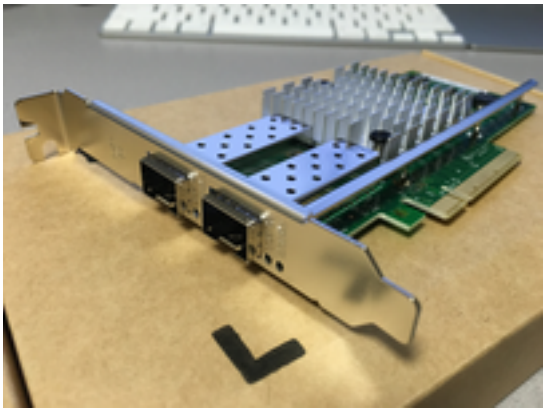

図 1.1　Intel X520-DA2

シンプルでフレキシブルなクラウド管理ツールであるところの OpenNebula では、こんなにも簡単にクラウド環境に PCI passthrough を導入できるんですよ、というお話を vol.3 で書きました。

いや書いたのですが、vol.3 で取り上げた方法は「libvirt Domain XML 文をテンプレートに直接記述する」というわりと力技の部類で、手軽な反面 OpenNebula で PCI デバイスをリソースとして管理できませんでした。本来はスケジューラが担うはずのリソース管理を、運用者がパーミッション等で管理する必要があり、書いた本人が言うのも何ですが微妙に残念感があったのは否めません。

しかし！ OpenNebula はバージョン 4.14 でついに PCI passthrough にネイティブ対応しました。ホスト上の PCI デバイスを OpenNebula がリソースとして認識し、利用者の「ホストの PCI デバイスを使いたい」という要望に対してスケジューラがよしなに計らってくれるのです。

PCI passthrough の恩恵を得やすい PCI デバイスとして、InfiniBand や 10GbE NIC といった高速なインターコネクトがあります。これらは SR-IOV(Single Root I/O Virtualization) に対応している製品も多く、クラウド環境との親和性も高かったりします。最近では 10GbE NIC の価格が随分下がっており、個人でもなんとか手が出せる価格帯になってきました。

そこで今回は、OpenNebula の PCI passthrough ネイティブ対応と、10GbE デバイスの SR-IOV、という 2 つのテーマに取り組みます。具体的には、

1. ホストの 10GbE NIC を SR-IOV で仮想化する
2. 仮想化したホストの 10GbE NIC を OpenNebula に管理対象リソースとして登録する
3. OpenNebula から PCI passthrough で 10GbE NIC を使用する仮想マシンを作成する

を試してみたいと思います。

1.2　前置き

- OpenNebula 環境での、OpenNebula サーバープロセスが稼働するマシンを「フロントエンド」と呼称します。
- OpenNebula 環境での、仮想マシンが稼働するマシンを「ホスト」と呼称します。
- フロントエンドとホストはいずれも Ubuntu 16.04 LTS server で構築しています。

- 使用する OpenNebula のバージョンは 5.0.2 です。
- 仮想マシンのイメージファイルは、Marketplace の OpenNebula Public からインポートした「Ubuntu 16.04 - KVM」イメージを使用しています[*1]。
- ハイパーバイザは KVM を使用します。
- OpenNebula が一通りセットアップされ、各種リソースが登録されているものとします。
- ホストは CPU、チップセット、BIOS/UEFI が I/O 仮想化支援に対応しているものとします。また、kernel やドライバモジュールのオプションが適切に設定されているものとします[*2]。

1.3 ホスト設定

まずはホストの環境を整えます。PCI デバイスを SR-IOV で仮想化した後、libvirtd の設定を一部変更し、仮想化した PCI デバイスを仮想マシンにパススルーできるようにします。

1.3.1 ホストの 10GbE NIC を SR-IOV で仮想化する

X520-DA2 はコントローラに Intel 82599 が載った 2 ポートの 10GbE NIC です。ドライバは ixgbe を使用します。SR-IOV に対応しており、物理的なポート上に仮想的なポートを複数作成し、それぞれ独立したポートとして利用できます。この仮想ポートはバーチャルファンクションと呼ばれます。

最近の kernel であれば、sysfs に SR-IOV を使用するためのインタフェースが用意されています[*3]。SR-IOV に対応したデバイスであれば、**/sys/bus/pci/devices/ドメイン番号:バス番号:デバイス番号.ファンクション番号/sriov_numvfs** に作成するバーチャルファンクション数を書き込むことで、そのデバイス上にバーチャルファンクションを作成できます。

ドメイン番号、バス番号、デバイス番号、ファンクション番号は毎度おなじみ lspci コマンドで確認します。ドメイン番号は通常 **0000** なのでデフォルトでは省略されますが、-D オプションをつけることで確認できます。筆者の環境では以下のように表示されました。

```
% lspci -D | grep 82599
0000:04:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Con-
nection (rev 01)
0000:04:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Con-
nection (rev 01)
```

0000:04:00.0 が X520-DA2 のポート A、**0000:04:00.1** がポート B に該当します。

[*1] OpenNebula AppMarket: http://marketplace.opennebula.systems/appliance

[*2] この原稿を執筆時点の Ubuntu 16.04 LTS の kernel には、オプションに intel_iommu=on を指定すると boot 時にパニックするバグがありました。そのためホストはパッチを当ててリビルドした kernel を使用しています。https://bugzilla.kernel.org/show_bug.cgi?id=119281

[*3] ixgbe ドライバのソースの README によれば、kernel 3.8 以上であれば sysfs からのバーチャルファンクション設定が利用できるようです。kernel 3.7 以下の環境は ixgbe ドライバの max_vfs オプションを使用してください。

Chap. 2

snap パッケージを作ってみよう

kazken3(@kazken3)

Ubuntu 16.04 LTS では snap パッケージが実験的に利用できるようになったので、今回は snap パッケージを作成してみてみることにしました。さてその命運やいかに。

2.1　はじめに

今回は、「Ubuntu 16.04 LTS リリース記念オフラインミーティング」もあり、snap パッケージに関する発表を行う機会がありました。さて、今回ざっぱ〜んでも何を書こうかと考えていましたが、この機会ですので snap アプリケーションを作って Ubuntu store に登録までやってみようかと思います。

2.2　snap パッケージって？

snap パッケージとは、これまで Ubuntu Phone で利用されてきた click パッケージをさらに発展させた『ユニバーサル Linux パッケージ』です。

snap パッケージの特徴をまとめると以下のような感じになります。

- squashFS を利用して、アプリケーションと snap.yaml[*1] と呼ばれるメタデータを含んだパッケージで読み取り専用のファイルシステムと書き込み可能領域を持つ
- システムコンポーネントやアプリケーションを自身でコンテナ化している。よってシステムや他のアプリケーションに影響を与えず更新や元に戻すことが可能
- セキュリティ機構を介して OS や他のアプリケーションから隔離されているが、ユーザーや OS が提供するきめ細かいポリシーに従えば必要な機能を利用し、コンテンツのやりとりを行うことが可能

[*1] snap に含まれるメタデータは snap.yaml で管理されています。後述の snapcraft を利用して snap パッケージを作成する際の設定ファイルである snapcraft.yaml とは異なります。

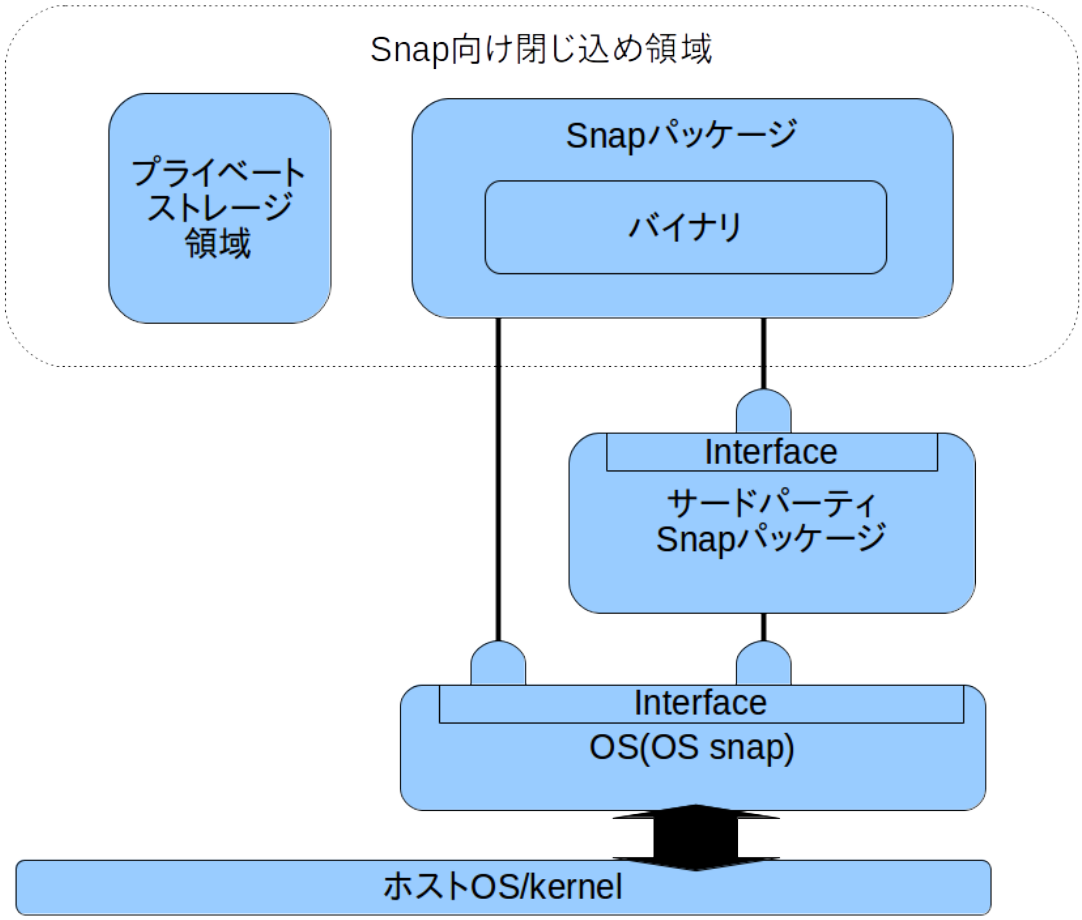

図: snap パッケージの動作構成

上図の snap パッケージが動作構成にもあるように、snap パッケージは OS snap[*2]上でのみ動作するため、「コンテナっぽい」動作を行うアプリケーションパッケージと捉えても差支えはないでしょう。

2.2.1　snap のメリット

では、snap パッケージのメリットは何でしょうか？

・ポータビリティが高い

これは、アプリケーションの仕組みが Ubuntu Core[*3]経由で snap パッケージのアプリケーションが実行可能であるためで、Ubuntu Core が動作する環境であれば、ディストリビューションの構成を意識することなく実行できます。

[*2] Ubuntu Core と呼ばれる snap パッケージです。
[*3] snap パッケージをインストールしたら必ずインストールされます。

・ホスト OS とアプリケーションが隔離された環境で動作している

snap パッケージでインストールしたアプリケーションは、ホスト OS から隔離された環境で動作しています。もし snap パッケージの脆弱性があったとしても、この脆弱性を突いてホスト OS を攻撃することは困難です。

・差分更新の仕組みがある

更新が発生した際に、必要な分だけを更新する差分更新が可能です。

・作成しやすい

後述しますが、snap パッケージの作成方法はわりと簡単です。

2.2.2　snap のデメリット

当然ながらデメリットもあり、以下の内容が挙げられます。

・ファイル容量が大きい

メリットであるポータビリティが高いことが、システムとの隔離に基づいているため、『必要なものはすべて詰め込む』スタイルとなっているためです。

・知名度

仕方ない話ではあるのですが、snap パッケージはまだ知名度はありません。

・進化途中

デスクトップにおける snap パッケージは Ubuntu 16.04 LTS では『Technology preview』という扱いになっています。また後述する snap パッケージを作成する snapcraft は毎週リリースされている状況であるため、今後何らかの変更がある可能性が考えられます[*4]。

2.2.3　snap はどのようなアプリケーションに向いている？

では、どのようなアプリケーションが snap パッケージに向いているかとなると、現時点では Technology preview であるため、snap パッケージをメインに置いてパッケージ配布を行うことは難しいかもしれませんが、以下の条件に当てはまれば snap パッケージに向いているのではないかと筆者は考えます[*5]。

・ネットワークを利用する
・更新頻度が高い
・もしもの場合でもセキュリティが担保できる

[*4] この記事は 2016/07/31 時点のものです
[*5] Web アプリケーションなどとか向いていそうな気がしますね。

Chap. 3

HummingBoard/PT3 でつくる小型録画ベアボーン

ryunuda(@ryunuda)

SolidRun 社の ARM ボード「HummingBoard」と PT3 を利用して録画ベアボーンの作成を試みた記録です。

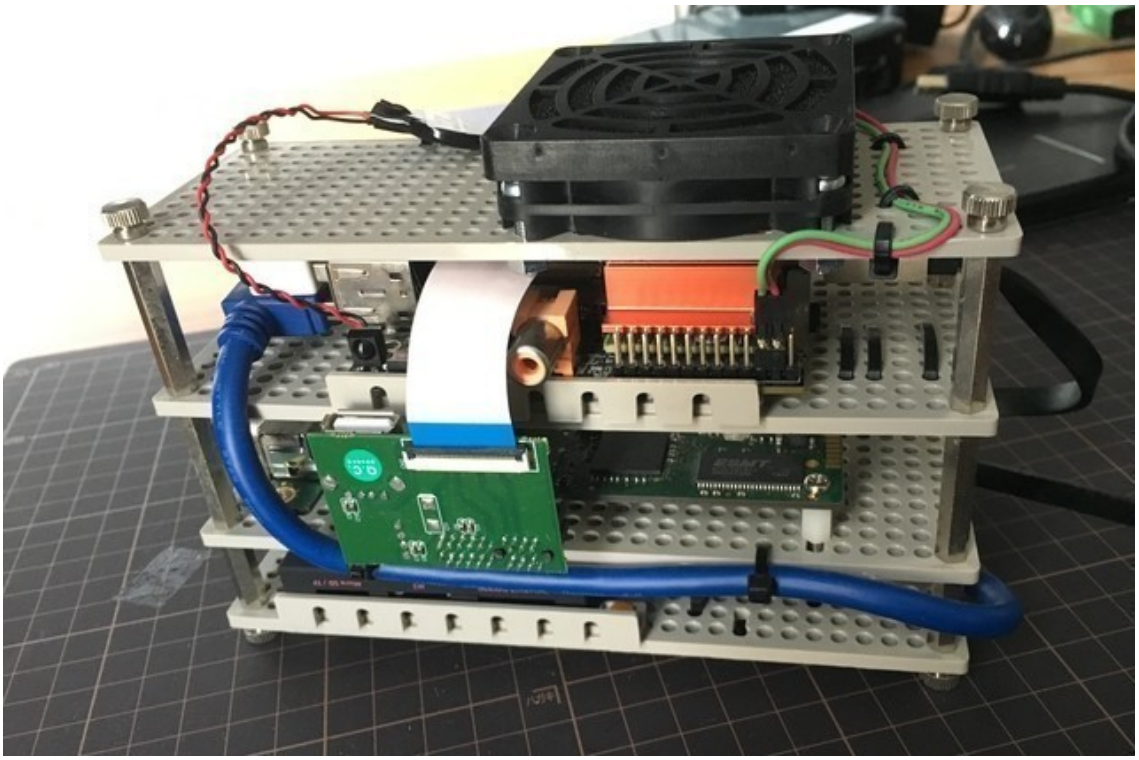

図: 録画ベアボーン

3.1　ハードウェア編

3.1.1　HummingBoard

HummingBoard はイスラエルの SolidRun 社が販売する、Freescale iMX6 を搭載した ARM ボードです。2016 年 7 月時点では Base、 Pro、 Gate、 Edge の 4 種類 (グレード) が存在しており、Base 以外には mPCIe と mSATA port を備えているという Raspberry Pi などの ARM ボードにはあまり見られないハードウェア上の特徴が存在します。また、注文時に BTO パソコンのように搭載する SoC/メモリの容量の選択も可能です[*1]。

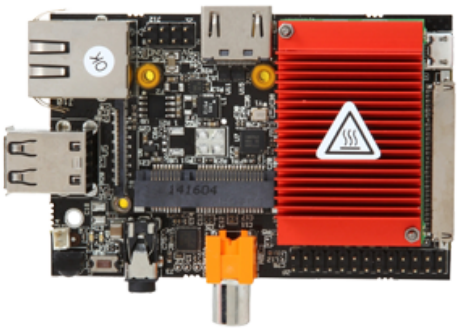

図: HummingBoard Pro

筆者は PCIe が付いているのは面白いと思ったことと、本来は WiFi カードや SIM カードを搭載するために設けられもののように見えるが、PT3 を始めとした他の PCIe デバイスもうまくすれば動くかもしれないと考えて HummingBoard Pro の i2eX/1GB memory モデルを購入しました。購入したのは HummingBoard が発売され始めた 2014/8 頃で価格は$120 でした。まだ Gate、Edge はなくこのモデルが最も性能の良いモデルでした。

3.1.2　実験に必要なもの

HummingBoard Pro i2eX/1GB と PT3 を接続する実験をするために以下のものを用意しました。

- HummingBoard Pro i2eX/1GB
- USB AC アダプター (2A)/電源用 USB ケーブル
  - HummingBoard を動かすためには 2A が必要です。2A より少ないアンペア数の AC アダプターないしは USB ポートに接続すると LED は光るのだけれども OS はいつまで経っても起動しないという憂き目に会います。
- PT3

[*1] 物理的な取り外しも可能なので、SoC の交換も可能です。

- mPCIe <-> PCIe 変換アダプター
  - HummingBoard 側 PCIe の物理インターフェースは mPCIe なので、PT3 を接続するために必要でした。筆者は Amazon でビープラス・テクノロジーのアダプターを購入しました。
- M2.5 ネジ
  - HummingBoard と変換アダプターの mPCIe 側の基板を接続するために必要でした。M2.5 ネジはそもそも物がなく、近所のホームセンターでたまたま見つけました。
- IC カードリーダー (NTT SCR3310-NTTCom)
  - 筆者が予備として所持していたものを使用しました。
- 同軸ケーブル
- 分配器
- microSD カード (16GB; class10)
  - HummingBoard の OS インストール用です。
- OS インストール用 PC(Ubuntu)
  - 今回は Ubuntu 14.04 LTS Desktop を使用しました。16.04 LTS でも大きく変わらないと思います。
- ディスプレイ・HDMI ケーブル
- キーボード

3.2　ソフトウェア編

3.2.1　armbian

armbian は ARM ボード向けの Debian/Ubuntu ベースのディストリビューションです。HummingBoard や同じ SolidRun 社が販売している Cubox-i や、Banana Pi、Cubieboard などの ARM ボードにも対応しています[*2]。

armbian の HummingBoard 用 image で提供される kernel はデフォルトで mPCIe や mSATA に対応しており、パッケージ管理も apt が使用可能であるため自前で kernel やパッケージをクロスビルドするといったやや高度かつ煩雑な作業を回避できるため armbian を選択しました。

また、今までは Ubuntu 版は Trusty 版 image のみでしたが、Xenial のリリースに併せて Xenial 版の image も作成されていたため今回は Xenial 版で新しく OS をインストールすることにしました。

armbian の提供する kernel には古いバージョンを使用する「Legacy」と最新バージョンを使用する「Vanilla」の 2 種類が存在しています[*3]。今回は Hummingboard 用 image のページを確認して、ハードウェア周りの対応が良いように読めた Legacy Kernel を選択しました。

[*2] http://www.armbian.com/download/ に対応 ARM ボードの一覧があります
[*3] http://docs.armbian.com/Hardware_Allwinner/#legacy-or-vanilla-kernel/ にも Legacy と Vanilla の違いがあります

Chap. 4

Cinnamon のおかしな翻訳にツッコミを入れる

あわしろいくや

今回は Cinnamon デスクトップ環境 (以下 Cinnamon) の翻訳をする決意をした経緯と、作業中に見つけた妙ちくりんな翻訳の指摘と解説を入れてみます。

4.1　GNOME Shell の辛いところ

Software Design 2016 年 8 月号の Ubuntu Monthly Report 第 76 回に、Ubuntu 16.04 LTS に Cinnamon 3.0 をインストールする方法を書きました。ここだけの話、もともとはネタも時間もなくて苦し紛れに出したネタだったりするのですが、実際に使用してみるとすごくしっくり来ました。Cinnamon は GNOME Shell から派生するところから始まっており、筆者はその GNOME Shell のヘビーユーザーなのでそのあたりが原因かも知れません。

GNOME Shell を使い始めてもう結構経ちましたが、辛くなってきたのは事実です。理由としてはいろいろあって、まずは IBus を使うことだけしか考えておらず、Fcitx を使うといろいろと不都合があるということです。端的な例は lp: 1594681 で、まさかまさかの環境変数ハードコーディングです。`export QT_IM_MODULE=ibus` と `export XMODIFIERS=@im=ibus` がハードコーディングされているため、Qt と XIM を使用したアプリケーションで IBus 以外が使えません。ほかにもいくつか IBus しか意図していないだろうという実装があります。

次に辛いところとしては、Ubuntu Weekly Recipe 第 395 回でも愚痴ったように GNOME Shell が AppIndicator/KStatusNotifierItem に対応しないことです。AppIndicator は Unity の、KStatusNotifierItem は KDE SC の通知トレイにアイコンを表示する仕組みで、ほぼ同じものになっています。一方、GNOME Shell は独自路線をひた走っており、歩み寄る様子はありません。確かに GNOME Shell のほうが先発ではあるものの、AppIndicator/KStatusNotifierItem がデファクトスタンダードであることに疑念を挟む余地はありません。

最後に挙げる点としては、GNOME Shell Extensions の仕様がダサいということです。対応する GNOME Shell のバージョンを拡張機能に埋め込まないと対応しません。現在 GNOME は半年に一度のリリースを行っているため、同じタイミングでのメンテナンスが必要ということです。場合によっ

ては本当にバージョンを追加するだけのこともありますが、GNOME Shell の仕様が変わってそれに追随しないといけないということもあります。要するに、拡張機能の開発者の苦労なんて全く考えていないということです。互換レイヤーを準備し、API のバージョンで変更を管理するという普通に考えられる仕組みを用意すればこのようなことにはならないわけですが、拡張機能開発者にひたすらスパルタを強いるというのが現状の仕様です。ということは、手の込んだ拡張機能ほど死にやすいというわけで、現に KStatusNotifierItem/AppIndicator Support という、まんまの機能を提供する機能がメンテナンスモードという名の放置状態になってしまいました。

Cinnamon にすると、これらの問題は概ね解消されます。Cinnamon では特定のインプットメソッドを統合するような機能はないため、IBus も Fcitx も、その他でも好きなものが使えます。2.8 からだったか 3.0 からだったかは忘れましたが、AppIndicator/KStatusNotifierItem にも対応しています。Cinnamon にも拡張機能やアプレットがありますが、実のところそれらに依存しなくても特に問題なく使えそうです。

4.2　Cinnamon の翻訳の現状

Cinnamon にもいろいろと問題はあります。最新版を使おうと思ったら自力でビルドする必要があるとかもそうですが、一番は現状の翻訳です。

そもそも Cinnamon はどのように翻訳されているのかというと、主たる部分は cinnamon-translations でひとまとめになっているのですが、この中にはなくて直接 github で管理しているものもあり、わかりにくくなっています。というわけで表にしてみましょう。

表 4.1　翻訳方法の分類

ソース名	まとめ/個別
cinnamon-bluetooth	まとめ
cinnamon-control-center	まとめ
cinnamon-desktop	個別
cinnamon	まとめ
cinnamon-screensaver	まとめ
cinnamon-session	個別
cinnamon-settings-daemon	個別
muffin	個別
nemo	まとめ

「まとめ」は cinnamon-translations にまとめられているもの、「個別」は直接 github で管理しているものです。

cinnamon-translations にまとめられている分に関しては、Launchpad 管理されており、翻訳者のチームももちろんあるわけです。翻訳が行われていない、あるいはクオリティが低いのはここにいる 17 人が原因ということになります。

いやいや、数が多すぎるとか難しいとかで対応できないんじゃないの、と思ったので、筆者も調査してみました。Poedit が表示した翻訳数 (Poedit の用語では件数) は次のとおりです。

合計 4247 件といっても、全部を完全に翻訳し直す必要はありません。誰がどう訳しても同じになる

表 4.2　翻訳件数

ソース名	件数
cinnamon-bluetooth	47
cinnamon-control-center	474
cinnamon-desktop	31
cinnamon	1435
cinnamon-screensaver	37
cinnamon-session	69
cinnamon-settings-daemon	571
muffin	361
nemo	1222
合計	4247

のもありますし、設定項目など無視してもメニュー上に現れてこないものもあります。そう考えると、どんなに多くても 2500 件ぐらい対象かなと思います。GNOME Shell から派生しているということで、gnome-settings-daemon や gnome-control-center や nautilus など派生元の翻訳を参考にできる部分もありますし、そう考えると更に少なくなります。

というわけで、そのぐらいの数だと個人ですべて面倒見るというのも現実的な視野に入ってくるわけです。ただ、個人だとどうしても得手不得手があるので、すべてをカバーするのは難しいですが[*1]、それでも LibreOffice よりはずいぶんマシで、まだなんとかなりそうな気がしました。

6 月ぐらいからちょいちょいと翻訳を進め、まだまだ道半ばではありますが、このぶんだと 9 月か遅くとも 10 月には一通り完了しそうなペースで進んでいます。

完了後は、github で管理している分には Pull Request を送って取り込んでもらおうと思っていますが、cinnamon-translations で管理している分は特に何かをするつもりはありません。["Japanese translation team for Linux Mint" team] に入ることは絶対にありません。そんな屈辱には耐えられません。

Cinnamon の翻訳を改善すべきと思っているのはどうやら筆者だけのようなので、それでいいのかなと思っています。もしも他にもいるのであれば、その人が頑張ればいいでしょう。筆者とは無関係にです。

4.3　Cinnamon のおかしな翻訳を見る

前置きがずいぶん長くなりましたが、いよいよここから本題です。Cinnamon 関連にある、筆者がおかしいなぁと思った翻訳をピックアップして、ここで晒していくことにします。見やすいように差分形式にします。もちろん修正前が現在の、修正後が筆者による翻訳です。

4.3.1　キングオブおかしな翻訳

何はなくともこちらをご覧ください。

[*1] 実際にやってみて、筆者はアクセシビリティ関連の翻訳が弱いということがよくわかりました。

Chap. 5

Ubuntu 16.04 と Windows 10 でデュアルブート

Rakugou

5.1　前々回のざっぱ〜ん♪

ざっぱーんの原稿の作成で Ubuntu を入れることになったらくごうさん。「ありのまますぎるよ」「内容がお粗末です」そんな中、初めて起きた問題が...「Let's note にはどの OS がふさわしいか、次に入れた OS が正 OS よ」色々やったけれど、結局問題は解決せず「じゃあいいんじゃないかな、そのままで」「いえ、もう (Ubuntu に) 決まってますよ」「じゃあ、Ubuntu で」そんな時、なにやら大変なことが...

5.2　あれから 1 年

Let's note のメーカー保証が 5 月に切れた。基本的に家電量販店で高価なものを購入した場合、普段は長期保証に入って有事の際に何とかできるようスタンバイしているのだが、この Let's note は展示品で購入した最後の一台であるため、通常のメーカー保証のみとなり、購入から 1 年経ったことで保証期間終了を迎えることとなった。また、国会答弁でも槍玉に上がるまでになった[*1]Windows 10 へのアップグレードについて、奇しくも 2016 年 7 月 28 日に Windows 10 の無償アップグレードが終了してしまった[*2]。その前に Let's note にデフォルトで入っている Windows 8.1 をこの機会に Windows 10 にアップグレードしておきたいと考えた。

前々号で触れた「Windows 入りの SSD を使ってから、Ubuntu 入りの SSD を使うと、起動できなくなる問題」を解決しないまま、Windows 10 にアップグレードをしようとして、Windows 入りの SSD に換装して Windows を起動してしまったため、前々回に頑張って設定した Ubuntu 入りの

[*1] http://www.sangiin.go.jp/japanese/joho1/kousei/syuisyo/190/meisai/m190134.htm

[*2] Windows 10 への無償アップグレードは終了したものの、障碍者向けの支援技術製品の利用者に対して Windows 10 への無償アップグレードが継続されている (2016 年 8 月 7 日現在)。https://www.microsoft.com/ja-jp/enable/windows10upgrade.aspx?tduid=(14c348e893a91089cb2980533c6ee123)(256380)(2459594)(TnL5HPStwNw-0FBZfNm6YPFXNhiu3o52rA)()

SSD から起動することができなくなってしまい、まっさらにしないといけなくなった。Ubuntu 一本で頑張っていこうと思ったのだけれど、ネット回線のセットアップには Windows が必要となること[*3]、また、Scansnap で電子書籍を取り込む際、どうしても傾きが気になる。画像が一つであれば Shotwell などを使って画像の傾き補正ができるのだが、数百単位の画像ファイルの傾きを補正するとなると、一つ一つ補正するにはどうしても時間がかかるため、やはり Scansnap の同梱ソフトに任せるのが早くて楽だ。ということで、まだまだ Windows にお世話になることも多いと見込んだ。どうにかして Windows を残したまま、Ubuntu を使えるようにできないかと考えた末、「デュアルブートしようではないか」というところに行き着いた。

5.3 材料

今回必要な材料は下の通り。全てを新しく揃えようとするとなかなか手間がかかる。

・2.5 インチの SSD

Windows をまるまるクローンして保存するために必要。Let's note に搭載されている容量以上の SSD があれば安心。

・M.2 SSD

Let's note に元々搭載されている Windows の環境をそっくりそのまま残しておくためにも、デュアルブート先として準備しておこう。

・外付け HDD

Windows のシステムイメージの保存場所として使用する。アプリケーションをいくつか入れただけの状態であったとしても、50GB ほどの容量が必要となる。

・USB メモリー (回復ドライブ用)

回復ドライブを作成するために必要。容量は 4GB あれば十分作成可能だ。

・USB メモリー (Ubuntu 16.04 LTS のライブイメージ作成用)

ライブイメージを作るためのもの。こちらの容量も 4GB あれば十分だ。

5.4 Windows 側でクローンの作成

今回デュアルブートを始めるにあたって、クローンを複製してバックアップを取ろうと考えた。失敗して万が一にも Windows が起動できなくなるというのは困る。OS の新規購入をする必要があるし、過去に Ubuntu をインストールしようとした時に、システム修復ディスク + システムイメージの組み合わせでバックアップを取ることに失敗したほろ苦い経験がある。そのため、前述の方法によるバックアップの他、もう一重の保険をとっておこうと考えたからだ。その際、現在の環境のバックアップを取

[*3] オペレータさんに OS を聞かれた時、「Ubuntu ならあるんですけどねぇ」と言うと「はぁ」と返されたのはいい思い出。

るのに一番頭を悩ませたのが、M.2 SSD を Let's note に接続する手段がないということだ。そういえば 2.5 インチの SSD をパソコンにつなぐ接続ケーブルは使ったことはあったのだが、M.2 SSD をパソコンに接続するケーブルというのは見たことがない。ヨドバシの店員さんに「M.2 SSD を外付けで接続できるケーブルみたいなのありませんか」と聞くと「ないみたいですね」と言われたので、すごすご諦めることにした[*4]。

というわけで、今の Windows 10 のバックアップを、今回のために購入した 2.5 インチの SSD にクローンしてバックアップ。必要に応じて USB 接続で起動して、本体の M.2 側にクローンを取り直して復旧させるという方針にした。

今回クローンを作成するにあたり、EaseUS Todo Backup を用いた。Intel 製の SSD だったりすると、手厚くクローン用のソフトも同梱してくれたりもするが、今回は同じような容量で、価格の安い Sandisk の SDSSDA-120G-J26 にすることにした。ついでに、USB3.0 のケーブルによってパソコンに接続できる Transcend の SSD ケースを購入し、クローンを開始することにした。メイン画面から「Clone」を選択し、その後はクローン元の SSD を選択し、移動したいフォルダを設定する。SSD の項目を選ぶと中身をまるごと選択してくれるので、今回はそちらで行った。その後、移動先の SSD を選択し、「次へ」をクリックすると、クローンが始まる。

さて、BIOS の設定を変更して SSD から立ち上げるぞ、と思ったら、エラー画面が出てくる。2 回に 1 回、強制的に電源が落ちる画面と Windows キーをクリックすると、BIOS 設定に戻る画面と出てくるのだが、どちらでやっても起動しない。

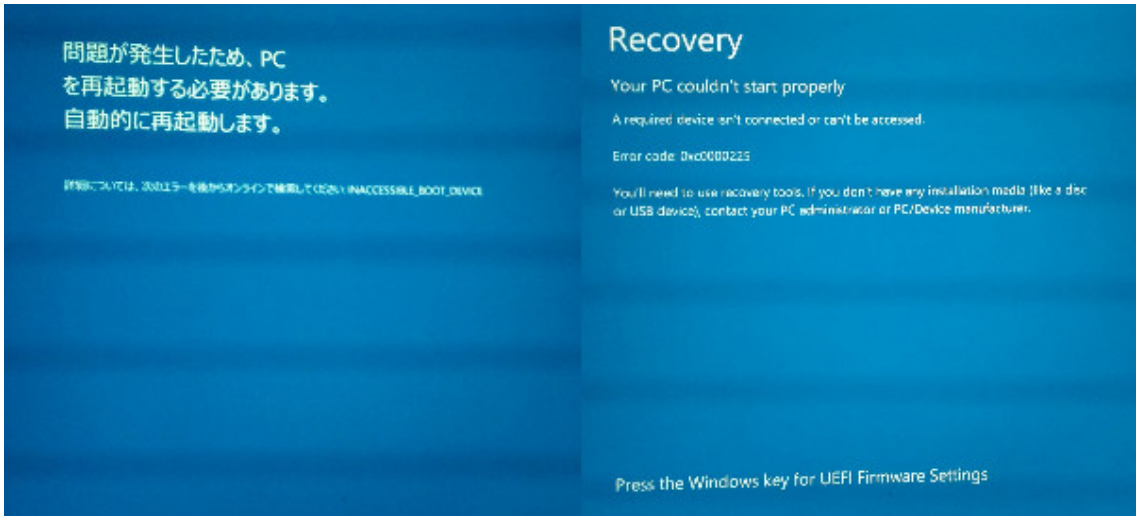

図: 起動すると図の 2 種類の画面が出てくる、再起動するか、BIOS を起動するかのいずれかになる。

仕方がないので、念の為作っておいた回復ドライブ（作り方などは後述する）の「詳細オプション」→「スタートアップの修復」を試みても、どうやら同じなようだ。筆者はヘタレなので、このあたりで外付けした SSD からのクローンを諦めることにした。何事も諦めが大事である。

[*4] 後々調べてみると、Amazon にはあるらしいということが発覚。慣れないものをむやみに買うのは怖いという精神から今回は購入を見送った。

Chap. 6

LibreOffice Online で遊んでみよう

おがさわらなるひこ (@naru0ga)

こんにちは。LibreOffice の方から来ました。

みなさま LibreOffice Online、通称 LOOL（ろーる）はご存知でしょうか？ 平たくいえば、ブラウザーで操作できる LibreOffice です。Google ドキュメントや MS Office Online のような、といえばわかりやすいでしょうか。

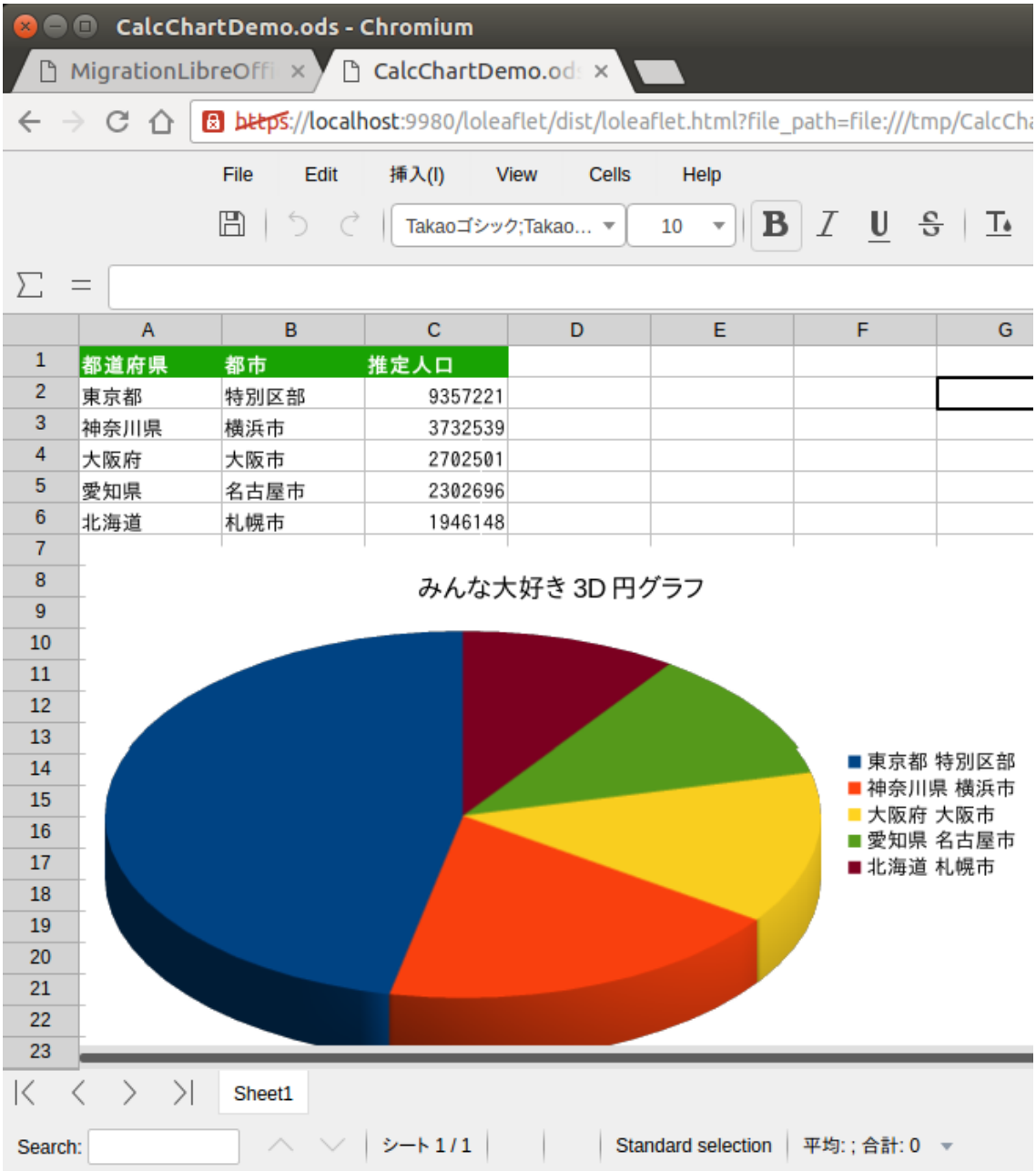

図: LibreOffice Online

しかしこれらの競合と違った大きな魅力は、LOOL はオープンソースであり、自分でビルドして自分でサーバーを立てて、自分で遊べることです。ソースコードのボリュームも、サーバーサイドが C++、クライアントサイドが JavaScript でそれぞれ 15000 行程度と、LibreOffice そのもの（約 750 万行で 80% 弱が C++）に比べると気後れしないですむ分量です。技術的にもなかなかチャレンジしていて、遊びがいがあると思います。

本稿では Ubuntu 16.04 LTS にて LOOL をビルドする手順を紹介し、さらにはちょっとだけ中身

も覗いてみます。

6.1　なにはともあれ LibreOffice のビルド

くわしくは後で述べますが、LOOL は文書を目に見える形に描画する……つまりレンダリングを、サーバーサイドで LibreOffice を動かすことによって行います。

LOOL が LibreOffice をキックするために使っているインターフェースが LibreOfficeKit（以下 LOKit）です。LOKit は C/C++ から LibreOffice の様々な機能、例えばタイルレンダリングなどを呼び出せる超強力なインターフェースですが、Android 版ビューアーや LOOL などの開発に伴って頻繁にインターフェースが変化しています。なので、最新の LOOL をビルドするには、まず最新の LibreOffice をビルドする必要があります。

そんなわけで LibreOffice をビルドしましょう。幸いなことに Ubuntu ならものすごく簡単です。

最初に行うのはソースコードの clone です（さすがに git の導入とかは説明は不要ですよね？）。公式リポジトリ[*1] から取ってくるのが王道ですが、ちょっと重いので github にあるリードオンリーミラー[*2] を使ってもよいです。今回は公式を使いましょう。ホームディレクトリ直下にソースを置きます。

```
$ cd
$ git clone git://gerrit.libreoffice.org/core.git LibreOffice
```

次にビルドに必要なパッケージを導入しましょう。だいたい次で一発です[*3]。

```
$ sudo apt-get build-dep libreoffice
```

いよいよビルドです。

```
$ cd LibreOffice
$ ./autogen.sh --with-lang="ja"
$ make
```

autogen.sh には指定できるパラメータが山ほどありますが、ただビルドするだけならこんな感じでよいです。ちょっと時間がかかるので、間にご飯でも食べてください。

[*1] https://gerrit.libreoffice.org/core.git

[*2] https://github.com/LibreOffice/core

[*3] これは「現在インストールされている LibreOffice のビルドに必要な（依存している）パッケージを全部取ってくる」という意味です。これからビルドするのは LibreOffice のマスターブランチ（開発版）なので、場合によっては依存関係が変わっていて、ビルドが失敗することがあります。その場合は適宜、エラーメッセージをみながら必要なパッケージを追加で入れるなりなんなりしてください。LibreOffice のメーリングリストで聞くのもよいでしょう。

あとがき

あわしろいくや

毎回それなりに苦労して発行しているのですが、今回はさまざまなことが重なってこれまで以上に苦労しました。

まずは組版システムの Re:VIEW が 1.7 から 2.0 にバージョンアップし、非互換が発生しました。ついでにバグも踏抜きました。kazken3 さんと柴田さんが対応作業を行ってくださいました。

Jenkins はこれまで柴田さんが借りていた AWS を間借りしていたのですが、今回からは ConoHa で同居させることにしました。設定ドキュメントは柴田さんが執筆し、それを水野さんが修正して私が作業という段取りだったのですが、LXD も Jenkins も知らない私にとっては難しいことだらけでしたが、手助けもありなんとか動作させることができました。

この『ざっぱ〜ん♪』は、可能な限り自由な環境で制作するようにしています。もちろん個々人の制作環境には関知しませんが、ConoHa 自体はプロプライエタリなサービスではあるものの OpenStack で構築されていますし、Ubuntu、Re:VIEW、フォント、OpenJDK、GitBucket、LXD/LXC、Jenkins、エトセトラ……、すべて自由なライセンスで提供されているものばかりで、かつファイルフォーマットは DRM なしの PDF と EPUB で、オープンな規格なのでビューアーも自由なライセンスで提供されているものもあります。それが私のこだわりのポイントです。

私自身の執筆環境は Ubuntu GNOME 上の gedit で Fcitx+Mozc を使用しており、完全に自由なライセンスです。フォントは Source Han Code JP を使用しています。Mozc は最新版を自分でビルドしたもので、今回その Mozc を本誌を購入してくれた希望者に配布したらどうかなとちょっと考えましたが、とりあえずはやめておくことにしました。リクエストが多ければ前向きに検討してみようかと思います。というか吊るしの Mozc だと辛くないですか。そんなことないのかな。

閑話休題。Ubuntu 16.04 LTS のリリースに伴い、ConoHa の OS は引き続き 14.04 としましたが、Jenkins が動いている LXC は 16.04 にしました。どうも TeX Live がイマイチのバージョンのままリリースされたらしく、最新版では修正されているバグがいろいろ残ったままとのことでした。このあたりも柴田さんが対応してくれました。

著者紹介にもあるとおり、もともと執筆予定のなかった記事を書いたわけですが、GNOME の松沢二郎さんにご意見をいただき、反映しました。

PDF 版は今まで A4 サイズだったのですが、やっぱり同人誌なら B5 だよねということで変更しました。簡単にできるのかなと思ったら意外と大変のようで、これも柴田さんに対応いただきました。

前々から TeX ってどうしてデフォルトだと英字フォントと日本語フォントが別なんだろう、一緒のほうが読みやすいのにと思っていたのですが、今回からどちらも IPAex フォントに変更しました。簡単にできるのかなと以下略って、そんなことばかりで本当に苦労をおかけしました。

今回も表紙のイラストはよかぜさんにお願いしました。テーマは「おもてなしのアラゲジリス」です。なるほどおもてなしといえばメイドさんですか胸のリボンがかわいいですねハァハァ……ってのはさておき、デザインは kazken3 さんによって 2 パターンが作成され、「総理、ご決断を！」(例の話題の映画ネタです) とのことで私が決めました。明らかに私がやるよりもクオリティが高く、設定した発売日も守ることができたので大変に助かりました。

そんなこんなで制作に携わったすべての皆さんに感謝しつつ、多くの人たち、みんなまとめて「我々」の思いの詰まった一冊が読者の皆様に気に入っていただけたのであれば、これほど嬉しいことはありません。

Vol.5 があまりにも予定から延びてしまったため (結局今回は前までやっていた事前告知でもできず)、続けて Vol.6 の制作に入ります。発売は年内を目指していますが、どんなものでしょうか。Vol.6 のテーマは「アプリケーション」にしようと思います。

とりあえず Vol.6 の製作期間中は長南さんとスムーズに連絡が取れるといいのですが……。どれだけ忙しいんですか……。

著者紹介

よかぜ @xxyokazexx

今回もご縁がありまして、表紙を描かせていただきました。ありがとうございます。個人的に女の子を久しぶりに描いたので楽しかったです。あと、最近できた夢はコードネームコンプです。いつか実現できたらいいなぁ…！ vol.5 お疲れ様でした！

おおたあきひこ @skywalker_37th

釣りとか宇宙とか UNIX とかアニメとか。最近は Excel 方眼紙にも興味があります。ヤマノススメの 3 期はまだか…

kazken3 @kazken3

今回は snap パッケージについてわりと真面目に書きました。本当は「ああひふみ先輩はひふみ先輩なんだよ...」とか、「ひふみ先輩の宗次郎になりたい...」といった話も書きたかったのですが、それはまた別の機会にしたいと思います。みなさまお疲れ様でした。

ryunuda @ryunuda

今回のテーマからちょっと外している気はしないでもないものの、このまま埋めておくのもなんだからということで HummingBoard で録画システム構築する記事を書かせていただきました。……IDF で発表された Intel の新しい小型開発ボード「Joule」にも PCIe が付いているそうじゃないですか奥さん。

いくや @ikunya

本来は著者にはならない予定でしたが、分量的にあれかなと思ったので急遽参戦しました。それと感じない内容になったと思うのですが、いかがだったでしょうか。オープンソースは多くの人の役に立つこともできるし、自分のためだけに何かをすることもできるのがいいところですね。

Rakugou @rakugou

フットワークの軽い他称ノマド。何かを忘れては右往左往する日々。Bash on Ubuntu on Windows の話題も記憶に新しいですし、Ubuntu と Windows の共存もいいものですね。あ、ざっぱ〜ん♪ Vol.5 発行おめでとうございます。

おがさわらなるひこ @naru0ga

LibreOffice Online、通称 LOOL の紹介っぽいことを書きました。オフィスソフトってどうしても地味っぽく思われてるようなのですが、結構おもしろいことをやってるじゃない、って思ってもらえれば嬉しいです。あと、カヌースラロームの羽根田卓也選手、リオ五輪銅メダルおめでとう！

うぶんちゅ！　まがじん ざっぱ〜ん♪ Vol.5 体験版

2016 年 8 月 21 日　v1.0.0 版発行
著　者　team zpn
発行所　team zpn