special feature 1 Linux学習のコツ

special feature 2 Ubuntu 14.04最新情報

Software Design

2014 6

2014年6月18日発行
毎月1回18日発行
通巻350号
（発刊284号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価 本体 1,220円 +税

［ソフトウェア デザイン］
OSとネットワーク、
IT環境を支える
エンジニアの総合誌

［新人Linux使い養成講座］

設定ファイルの読み方・書き方でわかる

special feature 1

Linuxのしくみ

上達のヒントは設定ファイルを大事に扱うこと

special feature 2 Windows XPからの乗り換えにいかが?

Ubuntu 14.04 “Trusty Tahr”

過酷な環境でも信頼できるLTSバージョン

special issue

- Google Glass GDK 先取り解説
- リアルタイム監視ツール「OpenTSDB」前編

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

第66回

IT エンジニア必須の

最新用語解説

TEXT：㈲オングス 杉山 貴章 SUGIYAMA Takaaki
takaaki@ongs.co.jp

テーマ募集
本連載では、最近気になる用語など、今後取り上げてほしいテーマを募集しています。sd@gihyo.co.jp 宛にお送りください。

Hack

Facebook 製新言語「Hack」

「Hack」はFacebookが独自開発したHHVM（HipHop Virtual Machine）向けの新しいプログラミング言語です。HHVMとは、同じくFacebookが開発したPHP実行環境で、JITコンパイル技術を用いることでPHPコードの極めて高速な実行を実現しています。HackはこのHHVM上で利用することを前提としてPHPをベースに作られた言語であり、HHVMと同様にオープンソース版がBSD Licenseに基づいて公開されています。

Hackの最大の特徴が、関数のシグネチャおよびクラスメンバに対してアノテーション方式[注]による型情報の付加ができるという点です。型検査はインクリメンタルに行われ、矛盾を検知した場合にはエラーを表示してくれるようになっています。動的型付け言語であるPHPに対して、静的な型付けのしくみを取り入れることで、開発の手軽さを維持しながら安全性やメンテナンス性の向上を実現しようというのがHackの狙いです。

静的型付けのほかに、次のような機能も盛り込まれています。

- ジェネリクス
- null 許容型
- コレクション
- ラムダ式
- 型エイリアス

コレクションでは、PHPの配列の代替となる、型安全性を備えた各種コンテナが提供されます。初期段階ではVector、Map、Set、Pairの4種類が用意されています。ラムダ式では、クロージャを作成するための簡潔なシンタックスが提供されます。

HackとPHPの互換性は高く、PHPで書かれたコードのほとんどはそのままHackのコードとして再利用できるとのことです。同一ファイル内で、一部のコードのみHackを導入し、そのほかは従来のPHPコードのままという使い方もできます。このような導入の敷居の低さもHackの大きな強みとなっています。

Hack 誕生の経緯

HHVMやHackは、もともとFacebook社内におけるシステム開発で使用するために生み出されました。HHMVの主な目的はアプリケーション実行速度の向上で、爆発的なユーザ数の増加に対応できる新しいPHP実行の枠組みとして開発されました。一方Hackの主な目的は、従来どおりの開発スピードを維持しながらの生産性の改善ということになります。

PHPのような動的型付け言語の場合、高い開発スピードを実現できる反面、初期段階でエラーを検出するのが難しいことや、コードの可読性やメンテナンス性が犠牲になりやすいといった問題があります。この問題は、Facebookほどの大規模なシステムの場合にはとくに顕著に影響します。一方で静的型付け言語には、開発の迅速さに劣るという弱みがあります。

そこで同社が取り組んだのが、動的型付けと静的型付けの“いいところ取り”です。最初の取り組みはHPHPc（HipHop PHP-to-C++ compiler）と呼ばれるPHPコンパイラで行われたそうです。HPHPcでは、PHPの言語仕様に若干の手を加えて静的型付けの機能を追加していたとのこと。しかし、事前コンパイルが必要なHPHPcのしくみではPHPの手軽さが損なわれるデメリットがあったため、これに代わるツールとして、JITコンパイル技術を活用したHHVMが開発されました。

HHVMが採用されたことで、事前コンパイルを利用した独自の静的型付け機能は利用できなくなりました。そこで次の一手として行われたのが型アノテーション方式を採用したPHP言語そのものの拡張で、これがHackの誕生につながったとのことです。

HHVMおよびHackには、PHPの高速化や安定性強化にかけるFacebookの強い姿勢を見て取ることができます。同社では2013年からの約1年間で、社内のほとんどのPHPコードをHackに移行させたそうです。SD

Hack
http://hacklang.org/
HHVM
http://hhvm.com/

注）各要素に関するメタデータをプログラム中に組み込めるしくみ。コンパイルなどの過程で無視されてしまうコメントに対し、アノテーションではコンパイル時や実行時に参照させることができる。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

［第1特集］

［新人Linux使い養成講座］

設定ファイルの読み方・書き方でわかるLinuxのしくみ

技術評論社の本が電子版で読める！

電子版の最新リストは

Gihyo Digital Publishingの

サイトにて確認できます。

https://gihyo.jp/dp

GIHYO Digital Publishing

amazon.co.jp

Google play

楽天 Rakuten KC　kobo

法人などまとめてのご購入については
別途お問い合わせください。

■お問い合わせ
〒162-0846
新宿区市谷左内町21-13
株式会社技術評論社　クロスメディア事業部
TEL：03-3513-6180
メール：gdp@gihyo.co.jp

Contents

Software Design Jun. 2014

［第2特集］

［一般記事］

［巻頭Editorial PR］

［アラカルト］

OSとネットワーク、
IT環境を支えるエンジニアの総合誌

Software Design

毎月18日発売

PDF電子版
Gihyo Digital Publishingにて
販売開始

年間定期購読と電子版販売のご案内

1年購読(12回)

14,880円(税込み、送料無料)1冊あたり1,240円(6%割引)

PDF電子版の購入については

Software Designホームページ
http://gihyo.jp/magazine/SD

をご覧ください。
PDF電子版の年間定期購読も受け付けております。

PDF電子版発売キャンペーン

2014年5月17日までにGihyo Digital Publishingにて電子版年間定期購読を新規にお申し込みの方は、年間購読料**12,000円**(税込み)でお求めいただけます。

・インターネットから最新号、バックナンバーも1冊からお求めいただけます!
・紙版のほかにデジタル版もご購入いただけます!
デジタル版はPCのほかにiPad/iPhoneにも対応し、購入するとどちらでも追加料金を支払うことなく読むことができます。

※ご利用に際しては、／~＼Fujisan.co.jp(http://www.fujisan.co.jp/)に記載の利用規約に準じます。

Fujisan.co.jpからのお申し込み方法

1 >> ／~＼Fujisan.co.jpクイックアクセス
http://www.fujisan.co.jp/sd/

2 >> 定期購読受付専用ダイヤル
0120-223-223(年中無休、24時間対応)

Contents

Software Design Jun. 2014

3

Column

Development

OS/Network

Logo Design ロゴデザイン > デザイン集合ゼブラ+坂井 哲也
Cover Design 表紙デザイン > Re:D
Cover Photo 表紙写真 > 101cats /gettyimages
Illustration イラスト > フクモトミホ、aico
Page Design 本文デザイン > 岩井 栄子、 ごぼうデザイン事務所、 近藤 しのぶ、 SeaGrape、 安達 恵美子
> [トップスタジオデザイン室] 轟木 亜紀子、阿保 裕美、佐藤 みどり
> [BUCH+] 伊勢 歩、横山 慎昌、 森井 一三、 Re:D、 [マップス] 石田 昌治

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

紙面版
A4判・16頁
オールカラー

電脳会議

DENNOUKAIGI

一切無料

新規購読会員受付中!

『電脳会議』は情報の宝庫、世の中の動きに遅れるな!

『電脳会議』は、年6回の不定期刊行情報誌です。A4判・16頁オールカラーで、弊社発行の新刊・近刊書籍・雑誌を紹介しています。この『電脳会議』の特徴は、単なる本の紹介だけでなく、著者と編集者が協力し、その本の重点や狙いをわかりやすく説明していることです。平成17年に「通巻100号」を超え、現在200号に迫っている、出版界で評判の情報誌です。

今が旬の情報満載!

新規送付のお申し込みは…

Web検索か当社ホームページをご利用ください。
Google、Yahoo!での検索は、

電脳会議事務局　検索

当社ホームページからの場合は、
https://gihyo.jp/site/inquiry/dennou
と入力してください。

一切無料!

◉「電脳会議」紙面版の送付は送料含め費用は一切無料です。そのため、購読者と電脳会議事務局との間には、権利&義務関係は一切生じませんので、予めご了承下さい。

毎号、厳選ブックガイドも付いてくる!!

『電脳会議』とは別に、1テーマごとにセレクトした優良図書を紹介するブックカタログ（A4判・4頁オールカラー）が2点同封されます。扱われるテーマも、自然科学／ビジネス／起業／モバイル／素材集などなど、弊社書籍を購入する際に役立ちます。

株式会社 技術評論社　電脳会議事務局　〒162-0846　東京都新宿区市谷左内町21-13

Pocket Reference

小さくても、中身充実！

「あれ何だったっけ？」
「こんなことできないかな？」
というときに、すぐに調べられる
機能引きリファレンス。
軽くてハンディなボディに
密度の濃い内容がギュッと凝縮！
関連項目への参照ページもあって、
検索性もバツグン！

岡本隆史、武田健太郎、相良幸範　著
四六判／272ページ
定価（本体2,480円+税）
ISBN978-4-7741-5184-7

石田つばさ　著
四六判／448ページ
定価（本体2,180円+税）
ISBN978-4-7741-4836-6

高江賢　著　山田祥寛　監修
四六判／536ページ
定価（本体2,580円+税）
ISBN978-4-7741-4592-1

古籏一浩　著
四六判／592ページ
定価（本体2,380円+税）
ISBN978-4-7741-4819-9

田口貴久ほか　著　日本仮想化技術株式会社　監修
四六判／352ページ
定価（本体2,980円+税）
ISBN978-4-7741-5600-2

沓名亮典、平山智恵　著
四六判／576ページ
定価（本体2,280円+税）
ISBN978-4-7741-3816-9

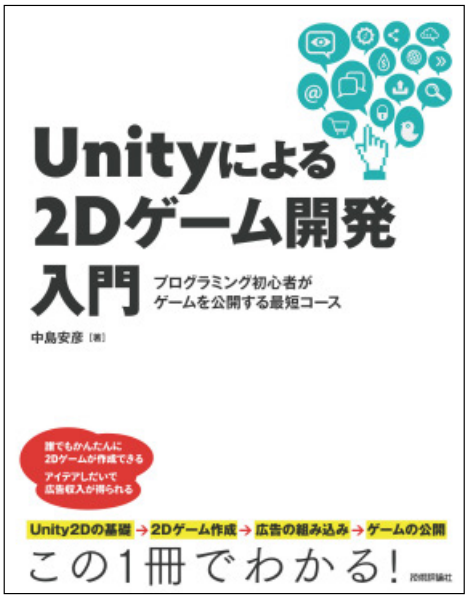
若杉司　著
四六判／400ページ
定価（本体2,980円+税）
ISBN978-4-7741-6332-1

片渕彼富　著　山田祥寛　監修
四六判／512ページ
定価（本体2,780円+税）
ISBN978-4-7741-6127-3

しげむらこうじ　著
四六判／512ページ
定価（本体3,200円+税）
ISBN978-4-7741-6335-2

山森丈範　著
四六判／304ページ
定価（本体2,380円+税）
ISBN978-4-7741-4396-5

技術評論社　当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
〒162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150 FAX.03-3513-6151



ISBN978-4-7741-6376-5
B5変形判／224ページ
定価（本体2,380円+税）

Unityによる2Dゲーム開発入門

プログラミング初心者がゲームを公開する最短コース

中島安彦　著

Unityを使った2Dゲームアプリで広告収入が入る！

スマートフォンユーザを中心に2Dゲーム（カジュアルゲーム）市場は盛り上がりを見せています。3Dゲームに比べリソースの管理がかんたんですので、アイデア次第でヒットを狙うことができ、ゲーム開発初心者は2Dゲームから取り組むのがお勧めです。本書では、Unity4.3で強化された2Dゲーム開発の方法の基礎を解説します。2Dゲームを制作し、広告を組み込み、ゲームアプリを公開するまでをゴールとし、ゲーム開発初心者にもわかるようにやさしく解説しています。

ISBN978-4-7741-6410-6
B5変形判／544ページ
定価（本体3,500円+税）

Ruby on Rails4 アプリケーションプログラミング

山田祥寛　著

Ver.4に完全対応！いちばんわかりやすいRailsの本！

MVCフレームワークとして人気のあるRuby on Railsは、Rubyを活用してREST原則に基づくWebアプリケーションを手軽に開発できる点が大きな特徴です。本書では、最新のRuby on Rails 4を対象に、Ruby on Railsの基本から、MVCモデルに則ったWebアプリ開発、ルーティングやテストの方法のほか、クライアントサイド開発としてAsset PipelineやCoffeeScript、SCSS、Ajaxなどの最新技術も網羅します。また、キャッシュ機能、国際化対応、Bundleによる機能拡張や、本番環境としてHerokuなどのPaaSへ導入する方法なども解説します。

技術評論社　当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
〒162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150 FAX.03-3513-6151

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

DIGITAL GADGET

Volume 186

安藤 幸央
EXA Corporation
[Twitter] »@yukio_andoh
[Web Site] »http://www.andoh.org/

視点を空中に。ドローンの台頭

Amazon Prime Airで注目のドローン

Amazonが「Amazon Prime Air」と銘打ったプロジェクトで、一般的に「ドローン」と呼ばれる小型のリモート制御ヘリコプターによる配達の実験を進めていることを発表しました。倉庫で荷物をピックアップするところから、配送先の家の前に置いてくるまで、すべてが自動化されています。

Amazonの「octocopter」と呼ばれる配送専用のヘリコプターは、故障を想定した冗長性のため、8つの回転翼を持っています。約2.3kgの荷物をつり下げ、往復16kmまで運ぶことができるそう。2.3kgというとそれほどの重量に思えないかもしれませんが、オフィスで使うようなプロジェクタ1台分くらい、または家庭用掃除機1台分くらいの重量です。そして実際のところAmazonで販売されている商品の86%は、この2.3kg重量内で運べるそうです。

Amazon Prime Airは、単なる話題集めかと思いきや、結構真剣に実現に向けて進行中のようで、すでに6世代目まで改良された機材が飛行テストされ、現在7、8世代目の機材を開発中とのこと。今後は、米国航空局や運輸安全委員会の認可や各種法規制のクリアなどが課題です。早くて2015年、遅くとも2020年頃までの実用化に向けて、着々と進行中のようです。

Amazon Prime Airで一躍注目を浴びたドローンですが、広い意味でドローンというと、UAV無人飛行体も含み、軍用の無人戦闘機などもドローンと呼ばれることがあります。今回取り上げるのは、ラジコンヘリコプターのような、小型のコンピュータ制御による複数回転翼のヘリコプターです。

最近ではドローンにカメラを搭載し、ソチ冬季オリンピックでのウィンタースポーツの中継や火山の噴火口の撮影など、一般に撮影が困難な状況の撮影、報道にも用いられるようになってきました。

従来型のラジコンヘリでは操縦が難しく、筐体が大きく高額、メンテナンスが難しいといった課題も、コンピュータ制御で安定して平易に飛ばせるようになったドローンの台頭で状況が変わってきました。GoProのような過酷な撮影状況でも活用できるアクションカメラの普及も良いタイミングでした。

Amazon Prime Air降着状態


Amazon Prime Air飛行中

※本記事で紹介しているものは国内未発表・未発売のものを含んでおります。

ドローンのしくみと応用例

ドローン普及のきっかけとして、職人芸的なラジコン操作術を必要とせず、コンピュータ制御による平易で安定した飛行を実現させたことがあげられます。自己安定アルゴリズムや、飛行モデルを想定した設計によって、どんな状況から計算をはじめてもいずれ収束、安定する方法を用い、フィードバック情報と組み合わせ、数十ミリ秒ごとに補正を繰り返します。マルチコプターの羽根（ローター）の1つが不調の場合も、多少不安定になりつつも、なんとか飛び続けることができるような冗長性を持っています。さらにヘクサコプターやオクトコプターには6つから8つのプロペラがあり、多少機体が損傷しても柔軟に対応できるアルゴリズムがあらかじめ考えられています。

各ローターの軸方向（ロー、ピッチ、ヨー、加速）の制御をまとめて搭載コンピュータが担当し、操る人のほうはローター1つ1つではなく、ヘリ全体を制御できるのです。また、ジンバルと呼ばれる回転台でカメラの揺れやブレを吸収し、滑らかな移動を実現する機材が安価で高性能になったことで、高品質のドローン撮影が実現しています。

また、損傷しやすい羽根の交換部品や大容量バッテリー、GPS機構など、周辺機器の充実も普及に貢献しています。

ドローンの応用例としては、自分を上空から撮影、行列の具合を見に行かせたりといった個人的な用途から、大規模な不動産の撮影、農業支援、山火事などの視察や消火活動、医療品や薬の緊急配送、撮影が困難な箇所の報道撮影などさまざまな応用が始まりました。

その一方で、悪用やプライバシー保護の課題なども持ち上がってきました。米国では武器の搭載禁止、個人的利益になる映像撮影の禁止、他人の個人宅への低空撮影禁止、監視目的での使用禁止などの検討が進んでいるそうです。

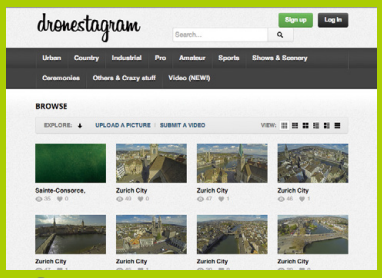
技術的アプローチでは、かすかなプロペラ音をとらえ、ドローンが接近していることを知らせてくれるDroneShield（http://www.droneshield.org/）などが開発され、スパイ映画のような世界が待ち構えているのかもしれません。

ドローンのこれからの可能性

ドローンを応用したサービスの可能性はまだまだこれから無限に広がっています。ダンサーとドローンが競演したり、ドローンジャーナリズムと呼ばれる報道の世界でも可能性が期待されています。一方で、技術的にクリアしなければいけない課題もまだまだたくさん残っています。雨風などの天候対策や着陸時の盗難を防ぐ工夫、GPSによる位置精度だけでなく画像認識による位置精度向上の工夫や、飛行時またはトラブル時の安全性の確保などといったものです。

また、運用ノウハウもいろいろあり、バッテリーの持ち時間が温度などの条件で大きく変化するため、常に残量を把握しておく、ヒーターで適度に暖めておく、バッテリー残量をバッテリーセルごとに都度計測する、航空機での移動の際はバッテリー持ち込みの制限を

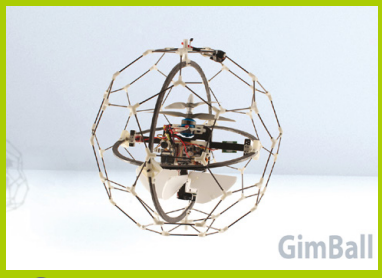
「Gofor」タクシーのようにドローンを呼べるサービス（まだ実サービスではなく、コンセプト段階）
http://www.gofordrones.com/



「Dronestagram」写真共有サイトInstagramのドローン版
http://www.dronestagr.am/browse/

「レクサス＋Drone」クルマのテレビCMでのドローンの未来的な映像
http://vimeo.com/79240111

「Pix4D」ドローン撮影した映像から、三次元地形を測定するツール
http://pix4d.com/



「GimBall」障害物や廃墟の中でも安定して飛行できるボール型のガードつきドローン
http://lis.epfl.ch/gimball

「SPAXELS」ドローンの連帯飛行をテーマにしたアート作品
http://www.aec.at/spaxels/en/

把握しておくなど、一般的なデジタル機器以上にシビアなバッテリーの把握が必要です。そのほかにも離陸前の水平位置でのキャリブレーションや、電源投入後GPSの位置検知の安定を待つ、不安定な飛行になったときに焦らないことも大切です。

操作方法では、実機と正面で向かい合ったときに、操作の左右と実機の左右とを間違わないようにしなければなりません。90度や180度横向きで練習したり、ラジコンヘリコプター用シミュレータ（パソコン上で動くもの）であらかじめ練習することが重要だそうです。

ニール・スティーヴンスンによる近未来を描いたSF小説『スノウ・クラッシュ』では、産業としてエンターテインメントと、プログラミング、高速の宅配しか残っていない世界がリアルに鮮やかに描かれています。この世界では「スマートボックス」と呼ばれるカウントダウンタイマーがついたピザボックス型の箱で、高速配送が行われているのです。

ドローンによる配送ネットワークが完備し、ドローンネットが進化すると、今までインターネット上のパケットとして転送されていたデジタルデータやデジタルコンテンツと同様に、「物質」を配送、配信、転送できる物質のためのネットワークが構築されます。航続距離がそれほど長距離ではないドローンも、バケツリレーのように配送できれば遠距離への配送も可能になります。

コンピューティングの世界で定評のあるロバスト性（外的要因に対する堅牢性を与えるしくみ）やアルゴリズムや手法が、安価に高速に実現できることによって、現実世界のサービスや物質的な事柄でも応用が始まります。物質ありきの現実世界と、デジタルな計算で成り立つコンピューティングの世界の融合がますます進むことでしょう。より進化したドローンは生き物的になり、一種のトンボのような、生物的な意思が感じられる瞬間がやってくるかもしれません。SD

GADGET 1

PARROT AR.Drone 2.0

http://ardrone2.parrot.com

フライトレコーダー搭載、老舗のクアッドコプター

PARROTは「オウム」を意味する名前が付けられたローターが4機搭載されているクアッドコプターで、iPhoneやAndroidスマートフォンで遠隔操作します。家電見本市CES 2010で最初のバージョンが初めて一般に披露されました。現在では30fps 720pのHDカメラを搭載し、GPS、フライトレコーダー搭載で、飛行経路を記録したり、指定した緯度経度に飛行できる新機種も出ています。通常バッテリーでは飛行時間は約12分、大容量かつ2倍のバッテリーを搭載すれば約36分可能。基本セットで約300ドルから。

GADGET 2

AirDroids

http://airdroids.com

折りたたみ式、小型トライコプター

AirDroidsは持ち運び時には折り畳んで運べる小型のトライコプター（ローターが3基）です。飛行時間は約20分、本体は約450gと軽量ですが、225gまでの荷物を持ち上げることができ、小型のデジタルカメラを搭載するのであれば充分な性能を持ちます（GoPro HERO3がバッテリー込みで76g）。Google Mapsと連携した、自動操縦が可能なAndroidアプリが用意されており、iOS版のコントローラも予定しているそうです。現在はまだ予約受付中の段階ですが、価格が499ドルと高性能で安価なのも特徴のひとつです。

GADGET 3

DJI PHANTOM 2

http://www.dji.com/product/phantom-2

超安定、空撮用クアッドコプター

DJI PHANTOM 2はGoProなどのアクションカメラを搭載できる機種があります。カメラの撮影をブレなく安定させる専用の3軸ジンバル"ZENMUSE"が用意されており、これによってとても安定した映像が撮影できます。機材は専用ケースも用意されており、トータルでも1kg程度、約25分間の飛行時間です。2.4GHz帯を使ったリモコンの到達距離は約1,000m。内蔵のGPS自動操縦システムによって、ある特定の位置と高度にホバリング（静止）し続けることができるため、とくに空撮時に向いているそう。ジンバル込みで1,000ドル弱。

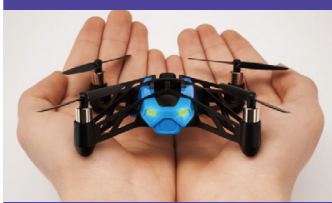
GADGET 4

PARROT MiniDrone

http://blog.parrot.com/2014/01/07/parrot-minidrone/

室内向け超小型ドローン

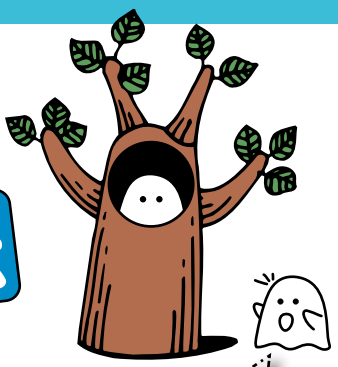
PARROT MiniDroneは手のひらサイズ、室内向けの超小型クアッドコプターです。スマートフォンやタブレットとBluetooth 4.0 LE経由で接続、コントロールします。本体は50g、プロペラ保護用のホイールを装着しても70gと軽量です。発売時期は未定で、カメラを搭載できないながらも手軽に楽しむことができそうです。スマートフォンのようにマイクロUSB端子でバッテリー充電し、約7～8分の飛行時間、約50m離れたところからもコントロールでき、より簡単に初心者でも操縦しやすくなっているそうです。



結城 浩 YUKI Hiroshi
http://www.hyuki.com/
Twitter：@hyuki

13

結城浩の再発見の発想法



Bottleneck

Bottleneck——ボトルネック

ボトルネックとは

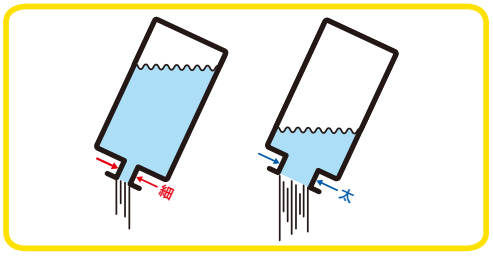
ボトルネック(bottleneck)とは、**システムのパフォーマンスを決定づける場所**の名前です。

もともと、ボトルネックというのは瓶(びん)(ボトル)の首(ネック)のことです。水がたくさん入った瓶を傾けて、瓶の外に水を出そうとしたとき、「瓶の首の太さ」が水を出すのにかかる時間を決定づけます。首が細いとなかなか水は出ませんが、首が太いとすぐに出るでしょう。すべての水を急いで外に出したいなら、首が太い瓶を選ぶべきでしょう。これは当たり前のことですね(**図1**)。

技術者はよく「**このシステムのボトルネックはどこか？**」と問います。これは、システムのパフォーマンスを改善したいと思ったら、それを決定づけている場所(すなわちボトルネック)を見つけて、そこを改善するのが効率のいい方法だからです。

多少意味合いは異なりますが、プロジェクト管理における**クリティカルパス**や、化学反応における**律速段階**[注1]もボトルネックと類似の概念です。

▼図1　ボトルネック



プログラムにおけるボトルネック

たとえば、遅いプログラムを改善してスピードアップしたいとしましょう。その場合「きっとここを直せば速くなるだろう」などと、やみくもにプログラムを書き換えてもスピードアップは見込めません。

必ずプロファイラなどでスピードの計測を行い、プログラムを遅くしている個所(ボトルネック)を見つけて、それからプログラムの修正を始める必要があります。

ソースコードをただ眺めるだけでボトルネックを見つけることは極めて困難です。たとえば、ソースコードの中に百万回のループを繰り返している関数が見つかったとしましょう。いかにもこれがプログラムの遅い原因のように見え、プログラムを修正してループ回数を減らしたくなります。でも、プログラムの実行中にその関数が一度も呼び出されないなら、ループを減らす努力をしてもスピードアップにまったく貢献しないでしょう。逆に、たった3回しかループしない別の関数があったとして、その3回がすべて、外部デバイスやネットワークへのアクセスのような重い処理を伴っていたら、その関数がボトルネックになることもあるでしょう。

注1)　多段階で進む化学反応の中で、反応速度が最も低いために全体の速度を決定づけてしまう段階のこと。

とはいえ、多数のループがボトルネックになりやすいのは事実です。とくに、多数回のループの中から呼び出される関数の中にさらにループがあると、かかる時間は倍増するので、ボトルネックを生み出しやすいものです。このような多数回のループをプロファイラで見つけることは難しくありません。

セキュリティにかかわる部分がボトルネックになることはよくあります。セキュリティというものの性質上、システムの「すべての情報」や「すべてのアクション」がセキュリティにかかわる部分を通過することが多いからです。セキュリティにかかわるコードが遅ければ、システム全体が遅くなることでしょう。「パフォーマンスを上げる」ことと「セキュリティのレベルを上げる」ことが、しばしばトレードオフの関係になるのも理解できます。

ボトルネックの発見に比べて、**ボトルネックの解消**はずっと難しいことです。ボトルネックの解消とは、パフォーマンスを低下させているボトルネックを特定したあと、それをなくすことです。ループの中で実行している処理をできるだけループの外に追い出すというのが基本的な技法でしょう。キャッシュやメモ化などもよく使われる改善方法です。

しばしば、ボトルネック以外の修正でパフォーマンスが向上します。たとえば、アルゴリズムを改善するのはパフォーマンス向上のために非常に有効な方法です。アルゴリズムを改善することで、ループの回数が劇的に減り、やっかいなボトルネックがなくなってしまうこともよくあります。

日常生活におけるボトルネック

さて、私たちの生活でもボトルネックはよく見かけます。

家庭の中で、登校／通勤前の忙しいときに洗面所やトイレがボトルネックになることがあります。家族のみんながそこを使うのに、一ヵ所しかないために停滞してしまうのですね。

交通事故や道路工事のために道が狭くなって、交通渋滞が起きることがよくあります。これは絵に描いたようなボトルネックですね。

複数人で作業を進めるときには、ボトルネックがどこにあるのかを見つけるのは非常に重要です。なぜなら、みんながいくらがんばって作業を進めても、ボトルネックが改善されなかったら全体のパフォーマンスは高くならないからです。逆にいえば、ボトルネックさえ改善していれば、あまりがんばらなくても良い結果を生む場合があるということです。

「上司の承認」がボトルネックになるのはよくあることです。作業を進めてその結果をリリースしたり、発表したりする際に、事前に「上司の承認」を必ず取る必要がある。しかし上司は忙しくてなかなか承認が取れない。そのために何日もロスしてしまう……これはありがちなことですね。

この場合のボトルネックの解消は、社内の承認フローを改善すること、あるいは上司の権限の一部を部下に委譲することなどが考えられます。いずれにせよ、やみくもにがんばって作業をするのではなく、システム全体のボトルネックを見つけて、改善することが大きな効果を生むでしょう。

社内の承認フローを改善することは、プログラムにおける「アルゴリズムの改善」に相当するといえます。ただし、ボトルネックを解消しようとするあまり、上司のチェックがうまく効かなくなってしまうことがあり得ます。これはプログラムにおける「セキュリティとボトルネック」の関係とまったく同じですね。

◆ ◆ ◆

あなたの周りを見回して、もっとパフォーマンスを改善したいシステムはありますか。そのシステムのボトルネックはどこにあるでしょう。ボトルネックを発見するにはどうしたらいいでしょう。そのボトルネックを解消することで、逆に問題が起きることはないでしょうか。

ぜひ考えてみてください。SD

enchant ～創造力を刺激する魔法～

(株)ユビキタスエンターテインメント　清水 亮　SHIMIZU Ryo
URL http://www.uei.co.jp

第14回 魔法

魔女の町へ

enchantMOONのソフトウェアに不満を抱きながらも、開発者たちの必死の作業によっていつか諸問題が解決されると信じて、僕はワールドツアーに出かけました。かねてから交流のあるアメリカの諸大学と、スウェーデンの名門ウプサラ大学で出張講義をする手はずになっていたからです。

サンフランシスコでは地元のベンチャー企業が集まるワーキングスペースでenchant.jsとenchantMOONのプレゼンテーションを行いました。講義やプレゼンはおおむね好評で、僕は時間を見つけてはenchantMOONを使って旅行記を書いてみたり企画メモを書いてみたりしていました。しかし、使うたびに動作がもたつくことや、かなりの頻度で落ちること、そして肝心の開発環境であるMOONBlockの起動に恐ろしく時間がかかってしまうことなどにとても苛立ちを感じていました。

何より悲しかったのは、これほど苛立ちを感じるデバイスを、我が社の製品として発売してしまったことでした。僕の目論見では、注意深く設計をし直せばもっと早い段階で高速化に着手できたはずですが、アップデート間隔が短いため、場当たり的な最適化しかできていないことが悔やまれてなりませんでした。

しかし、経営者としての僕はenchantMOONだけにかかわっているわけにもいかず、開発チーム全体が成長し、自立して優れた製品に磨き上げてくれることを祈るのみでした。

スウェーデンの端にあるウィズビーという小さな街で、enchantMOONで旅行記を書いていたときのことでした。

この街は『魔女の宅急便』のモデルになったと言われている、とても小さな、しかし素敵な街です。夜になると、本当に街のどこかに魔女が棲んでいるんじゃないかと思ってしまうような、そんな雰囲気たっぷりの街でした。

僕はひさしぶりのオフに、この街をあちこちめぐりながら、ハイパーリンクで旅行記を作っていました。しかし、あまりに遅い動作速度に、「もう限界だ」というストレスを感じると同時に、しかしなぜか不思議と「もっと作りたい」という欲望も湧いてくるのでした。僕は、開発チームが最も大切なものを見落としているのではないか、と強く感じたのでした。

それは、実際に使ってみればすぐにわかることで、そして使ってみなければ永久にわからないかもしれないということでした。

徹夜でやったのに

スウェーデンから帰国して最初の講義があり、僕は久しぶりに成蹊大学の学生たちと顔を合わ

せました。皆、心なしか疲れているようでした。僕はいつもどおりに講義を行い、授業を終わらせました。すると、一番先頭に座っていた女生徒がじっと僕になにか言いたそうな目で見つめてきました。僕は訳がわからず、目をそらして、そのまま帰ってしまいました。

次の週の講義も、ごく普通に行いました。しかし、やはり生徒たちはなにか疲れているように見えました。授業を終えようとすると、一人の女生徒が立ち上がり、僕に言いました。

「先生、先生が帰国したら、enchantMOONの作品発表をするんじゃないんですか？　何日も徹夜して作ったのに……」

なんてことだ。僕はとんでもないことをしてしまったと思いました。彼女たちが、まさかそんなに真剣にenchantMOONの課題をやっているとは、夢にも思っていなかったのです。

それで慌てて来週発表してもらうことにし、同時にある作戦を思いつきます。開発者たちにそれまでのアプローチが間違いだったと気づかせる、絶好のチャンスだと思ったのです。

想定外

翌週の授業はものものしい雰囲気になりました。正規の学生、いつのまにか増えていたもぐりの学生に加えて、enchantMOONの開発にかかわった全スタッフが見学に来たからです。

さあいざ発表、ということで最初の生徒が書画カメラの前にenchantMOONを置きました。

「私は、サークルの仲間を紹介するページを作ってみました！」

元気よく自己紹介して、最初のページを表示します。彼女は手書きした「次へ」というボタンをタップしました。MOONで作られたハイパーリンクであることは誰の目にも明らかでした。ところが……

「あ、あれ？　ジャンプしないね」

待てど暮らせどジャンプ先のページが表示されません。

「そうなんですよねー」

学生は苦笑しながらも、「いつもこのくらい時間がかかるんですよ」と笑います。しかし僕たちからみれば、これは明らかに異常と言ってよい遅さでした。40秒から1分近く経って、ようやくジャンプ先のページが表示されました。

「やっと出ました」

彼女が笑い、ページを見せると、僕たちは驚愕しました。ジャンプ先のページはシンプルな文字だけのページでした。ただし、その文字は何度も重ね塗りされ、飾られた飾り文字でした。のちにデータを調べさせてもらったところ、このページだけで40MBもありました。こんな使い方を僕たちは想像すらしていなかったのです。

enchantMOONではすべてのデータはベクトルデータとして保存されます。そしてページを開く、ジャンプする、といったときはベクトルデータを全部読み込み、全部レンダリングをかけていました。それは、もともとの想定が、検索を意識してシンプルな文字と図の入ったノートしか書かれないだろう、という前提があったからです。

ところが、実際に“紙”の代替として使わせてみると、学生たちは一所懸命、ページを飾ろうとしたのです。MOONBlockを起動してプログラムを書き換えなければ色を変えることもできないようになっているわけですが、極力カラフルなページを造ろうと頑張っているのがデータから見て取れました。これは見た目よりも遥(はる)かにたいへんな手間を賭して造られた作品たちであることは明らかでした。

僕は予想以上の状況に大きな失望を感じました。我々が作ってきたソフトは、僕自身の予想を遥かに超えて、ぜんぜんダメだったのです。

「無様だな」

僕は後ろの席に陣取る開発者たちに向けて言い放ちました。まったく、無様としか言いようのないことです。頭でっかちに、“こんな機能を実現しよう”ということだけを優先し、これが実際に実用的に使われるシーンをまったく想

定せずに開発を進めてきたわけですから。

僕は学生たちに感謝しました。彼女たちがここまで使ってくれたことで、根本からソフトウェアを見直す必要があることを、これ以上ないほどハッキリと我々が認識できたからです。

再設計

早急にアーキテクチャを根本から見直す必要がありました。しかし、開発期間に余裕はありません。そこで今回は現場に任せきりにせず、僕が自らアーキテクチャの再設計に関して基本方針を作ることにしました。

まず、画面周りをベクトルだけにせず、ベクトルのレンダリング結果をラスター(ビットマップ)データとして保存し、このラスターデータを表示するようにします。こうすれば、ベクトルデータがどれだけ巨大になっても、ページの表示そのものは高速に行われることになります。

ただし、ベクトルデータを完全に捨て去ってしまうと、enchantMOONの売りである文字認識や検索といった機能が使えなくなってしまうので、ベクトルデータも保存します。ベクトルデータが巨大になるとメモリも圧迫するので、ベクトルデータを世代別・領域別に分割し、ただ書き足しただけのときはその差分だけを保存し、一部を消した場合などはその都度ベクトルデータを読み込んで、部分的なデータを削除するなどの対応を行う方針にしました。

「これまでのソースはいったん全部捨てよう。あれは間違った設計思想のもと作られた最初のバージョンだ。ユースケースが明確化された今、根本部分をゼロから作り直し、そこに機能を足して行こう。この改造に3ヵ月を使う」

こうして、大規模なリファクタリング作業が始まりました。

1週間後、テストバージョンとして作られたプログラムは驚くほど高速になりました。これに機能を追加していくなかで低下するパフォーマンスをどうするか、ということも問題です。しかしベクトルデータを原則としてヒープに持たない、という方針を採用することで、メモリの利用効率が劇的に向上したことが最大の改善でした。ページのラスターイメージをキャッシュとして何枚も持てるようになるなど、できることの幅が大きく広がりました。

エフェクトへのこだわり

それから1ヵ月ほどして、布留川英一が僕のところにα版を持ってきました。

「どうかな、これで」

「どれどれ……」

僕は何の気なしにハイパーリンクをタップしました。すると、瞬時に別のページにジャンプします。あまりの素早さに、今度は頭がついていきませんでした。それで、ジェフ・ラスキンの著作『ヒューメイン・インターフェース』を思い出すのです。

ラスキンは、Macintoshの名付け親であり、独自の理論でユーザインターフェースを開発した研究者の一人です。ラスキンの研究によれば、画面が瞬時に切り替わった場合と、スクロールによって切り替わった場合とで認識速度に大きな差が出る、ということでした。画面が瞬時に別のものに切り替わると、それを脳が理解するまでに数秒を要するけれども、たとえば横方向にスクロールして切り替わる場合は一瞬で済む、といったものです。

そこで、ハイパーリンクでジャンプするときにはリンクの中に入り込んで行くようなトランジション(場面切り替え)エフェクトを実装することにしました。

メモリに余裕ができたので、キラキラエフェクトの改善も行いました。ハイパーリンクへのジャンプが高速化された一方、どの部分がリンクになっているか一目でわからないという問題が残っていました。リンク部分を一定間隔でキラキラと光らせる、という仕様が最初の企画書に盛り込まれていたのですが、動作速度があま

りに遅かったためオミットしていたのです。

とはいえ、「キラキラしたエフェクト」とは一体どのようなものなのか、言葉で指示されても僕にもわかりません。そこで樋口真嗣監督に相談すると、次々とエフェクトのアイデアが送られてきました。それをコマ送りで再生しながら、僕が自分で手付けアニメーションを作り、そのアニメーションをアルゴリズムで実現する方法をenchant.js上に実装します。

半日ほど作業して、企画担当の辻秀美に見せると「違う」と言われてしまいました。

「火垂るの墓、ホタルみたいな、ぼんやり丸くて、光って、いくつかふわふわ浮いて……そんな感じです」

そんなぼんやりした指示で作れるかよ、と思いながらも、火垂るの墓のビデオを見ながらこんな感じかなあ、とアニメーションを作ります。

「もうちょっとふわっとできないですか」

そこでいくつかの光を周期の異なるsin関数で左右に散らすことにしました。

「そう。こんな感じです」

そこまで作ってから布留川に渡すと、ものの1時間で実装されます。コンパイル、実行を繰り返す作り方に比べると、セーブしてすぐ実行できるJavaScriptはプロトタイピングに有効でした。それからは僕の指示はすべてenchant.jsで作られ、ドキュメントの代わりに短めのソースコードとコメントをプログラマに渡すようになりました。

ささもと

僕はその日、成蹊大学の教え子たちと、新宿の焼きとん屋「ささもと」で食事する約束をしていました。このお店は店主がプログラマで、授業でも紹介したため、学生たちが連れて行ってほしがりました。

「授業、どうだった？」

と僕が聞くと、ひとりの生徒はこう答えました。

「ずっと考えていました。どうすればenchant MOONを使いこなせるようになるのかって」

「君は一番熱心に書いてたからね」

僕は彼女が返してくれた端末のデータを思い返しました。あの遅いソフトウェアを使って、ぎっしりと書き込まれ、使い倒してやろうという気合いが否応なしに伝わってきたのです。

「授業を受けるまでは、自分がプログラミングとか、できるようになるとは思わなかったけど、こんなに楽しいものだと思いませんでした。ブロックでクマが動いただけで、すごく楽しくて。まるで魔法みたいですよね。プログラミングって。JavaScriptって、呪文みたいだし。MOONBlockは、魔法陣ですね」

「魔法、か……アーサー・C・クラークだな」

「えっ？」

「“充分に発達した科学技術は、魔法と区別がつかない”有名な言葉だよ」

確かに、我々は魔法を創りだそうとしているのかも……僕はそのときようやく気付いたのです。辻はずっとそう言っていたと。enchant、それは魔法を意味する動詞なのです。

エンジニアという視点から見たMOONは、アポロ計画であり、サターンV型ロケットです。しかし一方で、ロマンチストな少女たちから見たMOONは、魅惑の月であり、魔法のような存在なのです。だからこそenchantMOONは、僕が当初想像したよりも多くの人に興味を持たれ、ある面では受け入れられているのではないか。だとすると、僕がもっと注意を払うべきは、MOONのエンジニアリングマシンとしての特性ではなく、魔法の板としての性質なのではないか。だとしたら、魔法は現実世界に作用しなければ意味がありません。

「ありがとう。いいヒントになったよ」

僕は学生たちと別れ、そのまますぐにゴールデン街の馴染みの店に向かいました。電源とWi-Fiがとれるからです。現実世界に作用する“魔法”、そのプロトタイプは一夜にして完成しました。後にそれはGeminiと呼ばれることになります。SD

コレクターが独断で選ぶ!

偏愛キーボード図鑑

株式会社 創夢
濱野 聖人 HAMANO Kiyoto
khiker.mail@gmail.com
Twitter：@khiker

最終回

プログラマに人気のプロ向けキーボード

Happy Hacking Keyboard

写真1　HHKB（初代）

はじめに

当連載は、今回で最終回です。最後に紹介するキーボードは、Happy Hacking Keyboardです。プログラマの間で最も人気のあるキーボードの1つと言っても過言ではないでしょう。

Happy Hacking Keyboard

Happy Hacking Keyboard（以下HHKB）は、㈱PFUが販売しているUSBキーボードです[注1]。

A の横に Ctrl があることからもわかるように、独特のキー配列（UNIX配列）と、必要最小限のキーのみに絞った省スペースな形状が特徴です。省スペースであるため、ファンクションキーは単体では存在せず、Fn との組み合わせで入力します。DIPスイッチも存在し、スペースバー両隣にあるキーの割り当てを変更するなど、細かく挙動を変えられます。

現在、手に入るラインナップには次のものがあり、英語配列、日本語配列がそれぞれ存在します。

- HHKB Lite2
- HHKB Professional2
- HHKB Professional2 Type-S

そのほかにもう販売されていない製品で、初代のHHKB（**写真1**）やHHKB Lite、HHKB Professionalなどが存在します。ProfessionalとProfessional2の違いは、おおざっぱにはUSBハブを内蔵しているか、していないかです。内蔵していれば、Professional2です。

そのほかHHKBのUSBインターフェースをBluetoothに変換できるモバイルバッテリー、交換用のキーキャップ、キーボードトランクのようなオプションも数多く存在します。これらはPFUダイレクトという直販サイト[注2]で販売されています。

HHKB Lite2

Lite2（**写真2**）は廉価版のHHKBです。キースイッチはメンブレンスイッチで、約55gの押下圧があります。そのため、後述するProfessional2やType-Sと比較すると、タイピングしていてかなり重く感じます。また、Lite2はMac専用の刻印を施したスノーホワイトのHHKB Lite2 for Macというモデルも存在します。Lite2の値段はおよそ7,000円です。

HHKB Professional2

Professional2（**写真3**）はHHKBの高級モデルです。キースイッチに

写真2　HHKB Lite2

注1）https://www.pfu.fujitsu.com/hhkeyboard/
注2）https://www.pfu.fujitsu.com/direct/hhkb/option.html

最終回　Happy Hacking Keyboard


写真3　HHKB Professional2


写真4　HHKB Professional2 Type-S

は、東プレのリアルフォースシリーズと同じ静電容量無接点方式を採用しており、押下圧は45gです。そのため、キータッチはLite2と比べると軽く、タイピングしていて疲れません。本体が白色の白モデルとチャコールブラックの墨モデルの2種類が存在します。キートップに刻印がないものも存在します[注3]。値段は、21,000円ほどしますが、それだけの価値はあります。

HHKB Professional2 Type-S

Professional2 Type-S(**写真4**)は、Professional2の静音モデルです。Professional2との大きな違いはキースイッチです。Type-Sでは静音モデルのキースイッチを採用しており、タイピング時の静音性が増しています。現在のところ、Type-Sには白モデルしか存在しないようです。値段は3万円ほどしますが、静音のためだけにこれを選ぶ価値はあります。稀(まれ)にPFUダイレクトで安売りしていることがあるので、Twitterで「@PFU_HHKB」をウォッチしておくと良いかもしれません。

◆ ◆ ◆

HHKBは種類がいくつかあるので、どれを選べば良いのか悩むかもしれません。筆者はProfessional2もしくは、Type-Sを勧めます。

その理由は、Lite2の購入を考えるのであれば、前回紹介したHHKBライクな配列の小型メカニカルキーボードを候補として考えた方が良いためです。Lite2と値段はそう変わらないのに、Lite2よりキーボードタッチが良いです。なお、Professional2とType-Sであれば、懐に余裕があるならType-Sを勧めます。静音は数千円余計に支払ってでも得る価値があります。

キー配列は、日本語配列は少し窮屈な印象が否めないため、英字配列を勧めます。

HHKBは大きめの量販店に行けば取り扱っていますので、どれを選ぶにしろ、まずは店頭で触って試してみるのが良いでしょう。

連載の終わりに

1年以上にわたって、筆者の独断と偏見に満ちた選出でキーボードを紹介してきました。かなり趣味に走った紹介もしましたが、いかがだったでしょうか。

連載では、現在、手に入るキーボードを中心に紹介してきましたが、古いキーボードでおもしろいものはたくさんあります。たとえば、メカニカルキーボードでは、Cherry軸しか紹介しませんでしたが、ALPS軸というキースイッチを採用しているキーボードもあり、愛好家が存在します。古いキーボードはeBayやヤフオク！に出品されています。接続がPS/2やADBのものもありますが、変換コネクタは探せば販売されているので、大概のキーボードは今でも使えます。

「キーボードはタッチデバイスにおされ、人気が下がっているのではないか」という危惧もありますが、筆者はそうは考えていません。キーボードはPCを扱う際に必須のデバイスで、まだまだ人気があります。日本では2ちゃんねるのハードウェア板にあるキーボード関連のスレッド、海外ではgeekhack[注4]やdeskthority[注5]などで多くの愛好家が活発に議論を交わしています。最近では、kickstarter[注6]で「keyboard」と検索すると、キーボードに関するプロジェクトも見られます。まだまだおもしろいキーボードは出てきそうです。

最後に、本連載で1人でも多くの方がキーボードに興味を持っていただけたならば、うれしい限りです。SD

注3）余談ですが、刻印なしは格好良いですが、掃除のときにキーを外すと、どれがどのキーかわかりづらくなるので注意が必要です。
注4）**http://geekhack.org/**
注5）**http://deskthority.net/**
注6）**https://www.kickstarter.com/**

秋葉原発!

第44回

はんだづけカフェなう

Maker Faire Shenzhen

text：坪井 義浩　TSUBOI Yoshihiro　ytsuboi@gmail.com　@ytsuboi
協力：㈱スイッチサイエンス　http://www.switch-science.com/

Shenzhen Mini Maker Faire

ちょうど12回ごとに書いていますが、今年も3回目の深圳（しんせん）でのMaker Faireが開催され、筆者も参加してきました。Maker Faire Tokyoも年々大きくなっていますが、深圳でのMaker Faireも、ものすごい勢いで成長しています。前回と同様に会場は野外で、1日目には雨にも見舞われてしまいましたが、深圳市内のオシャレな町の路上で開催されるMaker Faireは非常に印象的でした（**写真1**）。

このMaker Faireには112の団体や個人がブースを出しており、IntelやAtmelといった半導体メーカーもスポンサーに名を連ねていました。また、今回のMaker Faireには多数の日本人が参加したというのも特徴だと言えるでしょう。

“Innovate with China”というスローガンを会場で多く見かけました。オーガナイザーでもあるSeeed Studioは、多大なマンパワーをこのMaker Faireに投入しているようで、それは「Maker文化を普及させることと、中国が世界に向けて開かれていることを示したいからだ」と同社の副社長が話していました。

▼写真1　Maker Faire Shenzhen

Makerムーブメント

今回のMaker Faireも、これまでと同様に、中国のホビースト（趣味に熱中している人）よりも、製品を作っている会社の出展が多くを占めていました。これはこれで中国のおもしろい商品を知れて良いのですが、筆者としては違う発想を持ったホビーストの作品を楽しみにしているので少し残念でした。

昨今、ハードウェアスタートアップですとか、ものづくり起業というのがバズワードになっていて、経済誌などビジネスパーソン向けの媒体でもよく見かけるようになりました。個人が作ったニッチな商品が、Kickstarterなどのクラウドファンディングで注目を集め、予定よりも大きな額の資金調達に成功したというのは夢のある話です。ある程度の数を作ることになれば、小ロット生産も受け付けてくれる製造業の会社を見つけたりして量産する必要も出てきます。そういった「Make:」の延長にある量産に筆者もとても興味があります。

同様に、Maker Faireも、各々が好きなモノを作って、持ち寄って展示をするMakerのお祭りだったんじゃないかなと感じています。そういったものは消費者に売れるような品質ではないし、作っている当人も消費者に売るつもりなどまったくない、そういう世界のおもしろさだったはずなのです。Makerムーブメントとは、もともとは自分で何かを作ることを楽しむはずだったように思うのです。しかし、既存の産業からは生まれてこ

なかった製品を作ることよりも、クラウドファンディングでお金を集めたり、起業することが中心に据えられている記事を見るたびに、筆者はなんだか複雑な思いをしています。

ハードウェアスタートアップというのは、構造的にはWebサービスを立ち上げて起業するというものと何ら変わらないと思います。多くの人が投資してくれそうなWebコンテンツのラフを作って、まずはお金を集めるってなんだか不健全ですよね。

オープンソースとビジネス

もう1つ、Maker Faire Shenzhenに参加して思ったことがあります。「Make:」文化のおもしろさに、Shareがあると筆者は考えています。つまり、Open source Hardwareの世界です。Arduinoを例に挙げると、オープンソースであったがゆえにさまざまな亜種が生まれ続け、既存のArduinoボードの枠から外のことをしたいと思った人へのパスが用意されていました。これが多くの人にArduinoが愛された理由の1つと言えるでしょう。

しかし、最近のクラウドファンディングなどで資金を集めているプロジェクトというのは、ソース(回路図)などが公開されていない、プロプライエタリな製品が多く見うけられます。ソフトウェアでも同様ですが、自社製品をオープンソースにしつつビジネスと両立させていくのはとても困難なことです。でも、そういったプロプライエタリな製品は、従来から売っている消費者向けの家電と何が違うのでしょうか。少なくとも、そこにはhackの楽しさはありません。自分の痒いところをかくことができないというのは、つまらないと思うのです。

MAKERSの著者としても知られる、元Wired編集長のクリス・アンダーソン氏もMaker Faire Shenzhenで講演していました。同氏は今、3D Robotics[注1]というUAV[注2]を製造販売するベンチャーを起業しています。3D RoboticsのWebサイトでは、これらUAVのコントローラの回路図、ソフトウェアが公開されており、実際にオープンソースでビジネスをしていることが見て取れます。クワッドコプターの販売は、さまざまな事業者が参入していて、すでにレッドオーシャンになりつつある分野です。しかし、コントローラがオープンソースであることで、ただ組み立てて飛ばすだけではない、楽しみの余地が残されています。同社の製品はオープンソースであることによって、少なくとも筆者にとっては、大きな魅力のあるものとなっています。

UAV

プロプライエタリな製品ばかりでしたが、会場では日本でも売っているのをみかけるUAVを見かけました。日本のAppleストアでも売っている、AR.Drone 2のPARROTも出展していました。

写真2はDJI[注3]のPHANTOM 2という製品の飛行デモをしているところを撮影しました。筆者の想像よりも、遙かに高い飛行安定性と電波到達距離で、デモを見ていてすっかり欲しくなってしまいました。技適の都合もありますので、日本に帰ってから買おうと思って、その場はぐっと我慢しました。帰って検索してみる

注1) http://3drobotics.com
注2) Unmanned Air Vehicle、無人航空機
注3) http://www.dji.com/

▼写真2 PHANTOM 2

と、Amazonで技適を通したPHANTOM 2が販売されていました。ほかにも扱っている模型店がいくつか見つかりましたので、日本でもそれなりの販売網を持っているようです。

Maker Faire Shenzhenのメインスポンサーの1つがIntelだったのですが、Intelのブースでもクワッドコプターを見かけました。なんと、Galileoを搭載しています(**写真3**)。このフレームも自作したのだろうかと思っていたのですが、先ほどのDJIのWebサイトを見ていると、同社の"Flame Wheel ARF KIT"というキットを使ったものだということがわかりました。このキットを手に入れて、筆者も自分のマイコンで飛ばしてみたいと思い始めています。

POV HDD Clock

ホビーストの作品を見て、とても興味深かったので紹介します。POV(Persistent of Vision)、つまり残像効果を使った作品をいろいろ作っている方が出展していました。HDDのプラッタ(ディスク)を加工して切れ込みを入れて、タイミング良く光らせることで写真のような模様を描いたり、時計の針を表示させたりしています(**写真4**)。

POVは、本連載の第14、15回で紹介をした、ダイナミック点灯と同じようなしくみです。高速で点滅しているものは、人間の目には常に点灯しているように見える効果を使っています。

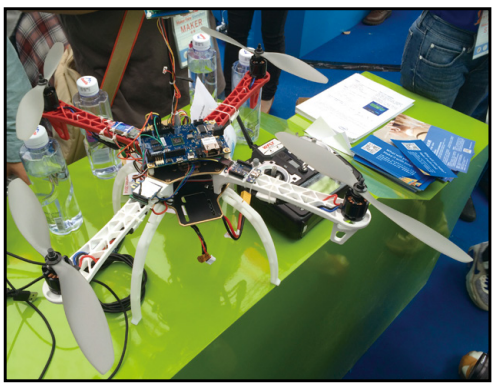
構造が気になっていろいろと質問をしていたら、出展者の方はPOVの工作事例の本を書いている方で、見本誌をいただいてしまいました(**写真5**)。いろいろなPOVによる作例が掲載されていて、なかなか充実した内容です。中国語の本ですので、書いてある文章は理解できませんが、回路図やソースコードを見ることで内容はわかりました。帰りの機内では、この本でなかなか楽しむことができました。本の値段は36元(約600円)と書いてあり、中国の書籍は意外と安いようです。

embedded world

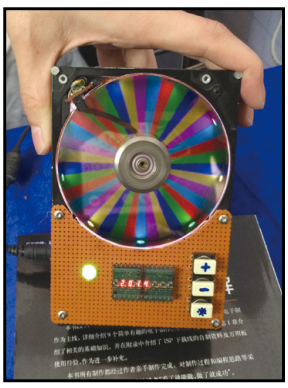
Maker Faire Shenzhenから少し遡った2月末、筆者はembedded worldというイベントにも参加してきました。このイベントは、世界でも最大級の組み込み技術展です。ドイツのニュルンベルグにて開催されていました。

筆者は組み込み技術者でもなんでもなく、あくまで趣味でエレクトロニクスを楽しんでいるだけです。が、この展示会で筆者も移植作業に参加をした、mbed LPC1549と、これに使われているLPC1549というマイコンが発表されるために参加をしてきました。mbedを提供しているARMのブースを訪ねると、mbed LPC1549が展示されていました(**写真6**)。自分の携わったプロダクトの発表が、ヨーロッパの、このような大きな規模の展示会で展示されていることがとてもうれしかったです。

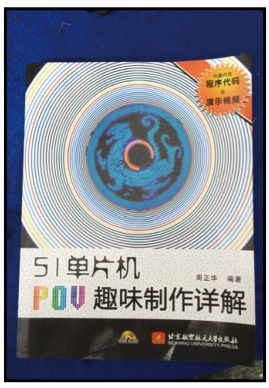
▼写真3　Galileo搭載のクワッドコプター



▼写真4　POV HDD Clock



▼写真5　POVの本



LPC1549は、PWM[注4]を高い自由度で使うことが可能なマイコンです。このことから、モーター制御などを得意とします。こういった特徴があるためにデモではモーターを回転させ、**写真7**のように、POVで“NXP”という文字を表示させていました。残念ながら、mbedのお手軽な開発環境を使うと、あらかじめ規定された方法でしかPWMを使うことができません。PWM制御の自由度の高さを発揮させるには、従来のマイコン開発を行わなければならないのが残念です。

PWM

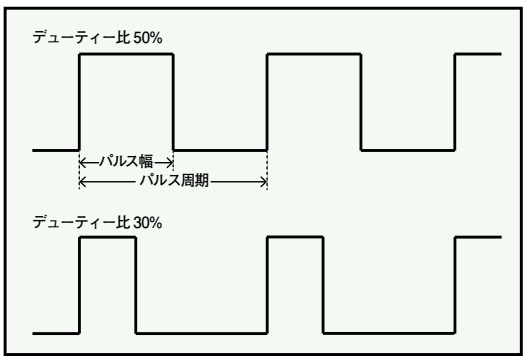
そういえば、PWMについて、この連載であまり扱ったことがありませんでした。PWMは、“パルス幅変調”というその名のとおり、信号の幅を変える手法のことです。**図1上**の場合、パルス幅はパルス周期の50％ですので、デューティー比が50％の状態です。このパルス幅を30％とすると、**図1下**のような信号になります。この中のパルス周期というのは、一般的にはPWM周波数などと呼んで使われています。たとえば、パルス周期が4ミリ秒だったとき、PWM周波数は1÷0.004=250(Hz)となります。

PWMはさまざまな分野で使われていますが、身近な例としてLEDの点灯制御が挙げられます。高速にLEDのON/OFFを繰り返すことで、人の目に映るLEDの明るさを変化させることができます。これを使って、パソコンがスリープしているときに、ほわっとLEDのインジケーターが点滅するといった機能が実現されています。また、RGBの3色のLEDの明るさを調整して組み合わせて、カラー表示を実現するのにも使われています。

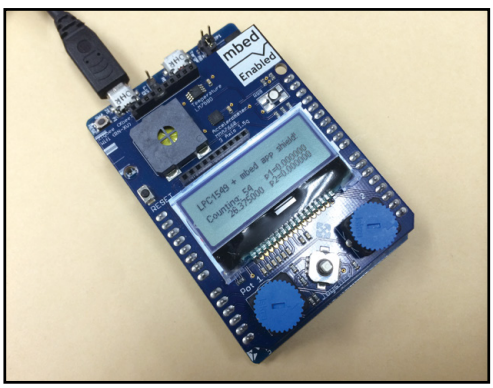
先ほど触れたモーターの回転速度を制御するのにもPWMは使われています。デューティー比を小さくすればモーターの回転速度は低くなりますし、デューティー比を高くするとモーターは高速に回転します。デューティー比とモーターの回転速度は、パルス周期によって線形だったり非線形だったりします。このため、モーターを制御する際には、制御しやすく、十分な性能が得られるパルス周期を求めて使うということが行われています。SD

注4） pulse width modulation、パルス幅変調

▼図1　パルス幅変調



▼写真6　mbed LCP1549



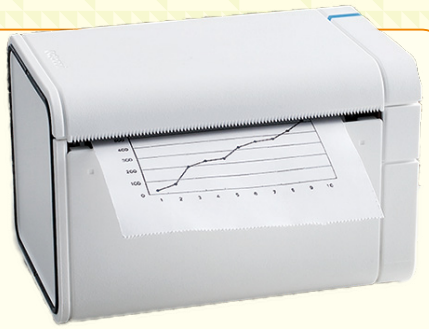
▼写真7　LPC1549のデモ

PRESENT 読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト http://sd.gihyo.jp/ の「読者アンケートと資料請求」からアクセスし、アンケートにご協力ください。ご希望のプレゼント番号をご入力いただいた方には抽選でプレゼントを差し上げます。締め切りは 2014 年 6 月 17 日です。プレゼントの発送まで日数がかかる場合がありますが、ご了承ください。

ご記入いただいた個人情報は、プレゼントの抽選および発送以外の目的で使用することはありません。アンケートのご回答については誌面作りのために集計いたしますが、それ以外の目的ではいっさい使用いたしません。ご入力いただいた個人情報は、作業終了後に当社が責任を持って破棄いたします。なお掲載したプレゼントはモニター製品として提供になる場合があり、当選者には簡単なレポートをお願いしております（詳細は当選者に直接ご連絡いたします）。

01 クリップ専用プリンター「ココドリ」 1名



PC 画面の必要な部分だけを印刷できるクリップ専用プリンタです。製品本体を PC に接続し、専用ソフトを立ち上げ、印刷したい部分をマウスで囲ってから、画面上の印刷ボタンをクリックすると、ほしい情報だけを印刷できます。マウスで囲った部分を印刷する「キャプチャモード」とコピーした文字列を印刷する「テキストモード」と、シーンに合わせた 2 つのモードを選択できます。専用ロール紙も付けてプレゼントします。

提供元 キングジム **URL** http://www.kingjim.co.jp/

02 カバンの中身 mini 3名

※製品には写真内の収納品は付属しません

鞄の中に入れることで収納力をアップできる「カバンの中身」シリーズの最新作。小型バッグやポーチ用に開発された小型サイズです。スマートフォンや小型タブレット、周辺機器も収納可能。ネオプレンやトリコット素材を採用し、電子機器保護にも対応しています。

提供元 ビー・ナチュラル **URL** http://www.kabannonakami.com/

03 無線 LAN 子機 GW-450S（手裏剣） 1名

無線 LAN の新規格 IEEE802.11ac に対応した無線 LAN 子機（11a/11n にも対応）。最大 433Mbps の転送速度で通信できます。2.4GHz Wi-Fi は内蔵しているけれど 5GHz Wi-Fi がないノート PC に最適。USB2.0/1.1 ポートを搭載した Windows PC または Mac で使えます。

提供元 プラネックスコミュニケーションズ **URL** http://www.planex.co.jp/

04 第3版 Cisco LAN スイッチ教科書 2名

シスコシステムズ合同会社 基盤技術グループ 著／
B5 変形判、400 ページ／
ISBN = 978-4-8443-3564-1

シスコ社のベテラン技術者が執筆した LAN とスイッチの構造／機能／技術を総合的に理解できる 1 冊。「日本人が書いた、翻訳書にはない読みやすさ」と「実践的な内容」が特徴です。

提供元 インプレスジャパン **URL** http://www.impressjapan.jp

05 教養としてのプログラミング講座 2名

清水 亮 著／
新書判、192 ページ／
ISBN = 978-4-12-150489-0

本誌連載「enchant ～創造力を刺激する魔法～」でお馴染みの清水亮氏が、プログラミングの成り立ち、簡単なプログラムの作り方、日常生活に役立つプログラマの思考法などについて解説します。

提供元 中央公論社 **URL** http://www.chuko.co.jp/

06 フリーソフトではじめる機械学習入門 1名

荒木 雅弘 著／
菊判、272 ページ／
ISBN = 978-4-627-85211-2

機械学習の基本から強化学習や深層学習などの最先端の手法までを解説。フリーのデータマイニングソフト「Weka」を使って、データの読み込みから学習結果の視覚化までを自分の手で行えます。

提供元 森北出版 **URL** http://www.morikita.co.jp/

07 プログラミング PHP 第3版 2名

Kevin Tatroe、Peter MacIntyre、
Rasmus Lerdorf 著、高木 正弘 訳／
B5 変形判、416 ページ／
ISBN = 978-4-87311-668-6

PHP の言語仕様から実用的なプログラミング技法まで、詳細に解説。文字列処理や配列処理、拡張モジュールと組み合わせてデータベースを使用する方法など、実践的なテクニックを網羅しています。

提供元 オライリー・ジャパン **URL** http://www.oreilly.co.jp/

新人Linux使い養成講座

第1特集

設定ファイルの読み方・書き方でわかるLinuxのしくみ

上達のヒントは設定ファイルを大事に扱うこと

本特集では、RHEL（Red Hat Enterprise Linux）を題材に、Linuxの使い方を学んでいただきます。新人の方にとって、企業システムを支えるLinuxは、なにか大きな・窺い知れない・ちょっと怖いものに感じられるかもしれません。でも、本特集を読めば大丈夫です。新人さんが理解しにくい知識をひとつひとつ解説します。その理解の鍵は、設定ファイルです。エンジニアはOSにさせたいことを、設定ファイルに書きます。そうすれば望みどおりにOSは動きます。とてもシンプルなことなのですが、忘れがちな技術です。まずはLinuxのディレクトリ構造など、基本的なしくみを知り、設定ファイルを書くための道具としてviの使い方を学びます。設定ファイルを自由に書くことができるようになると、ユーザ管理、システム管理、ネットワーク管理、アプリケーション管理ができるようになります。これが実力あるLinux使いになる第一歩です。各種設定ファイルの特徴を知っておけば、エラーが起きたときに原因の追及も楽になります。最後に新しいREHL7の紹介を少しします。本当の基本はあとで自分で自在に使えることがゴールです。本特集をじっくり読んで練習してください！

Writer　中井 悦司（なかい えつじ）
レッドハット㈱ グローバルサービス本部 シニアソリューションアーキテクト クラウドエバンジェリスト

CONTENTS

Part 1

Linuxのはじめの一歩

ディレクトリの歩き方とファイル編集の基礎

最初のパートでは、Linuxが持つファイルシステムについて、その構成と操作方法、パスの考え方、ディレクトリとパーティションの関係についてお話しします。章の後半では、設定ファイルの編集に使うviエディタについても解説します。始め方やコマンドによる操作方法など、初歩的な部分から見ていきましょう。Part2からは、これらの内容を前提として話を進めていきます。

設定ファイルで基礎を見直す

この特集では、「設定ファイル」という視点で、Linuxの基礎を見直します。Linuxをインストールしたあと、あるいは、クラウドでLinuxの仮想マシンを立ち上げたあと、デフォルト設定のままで、なんとなく使っているユーザも多いのではないでしょうか？　しかしながら、「必要なアプリケーションが動けば十分」と思っていると、思わぬところで足をすくわれます。

「性能が思うように出ない」「再起動したらうまく動かなくなった」「原因を調べようにもログファイルの場所がわからない」―― 設定ファイルを理解することは、Linuxをより効果的・効率的、そして安全に利用する基礎となります。

Part1では、Linuxのディレクトリの基礎を学び直したうえで、設定ファイルの効率的な編集に不可欠なviエディタとその他のコマンドを紹介します。そのあと、Part2～Part5では、それぞれ、ユーザ管理、システム起動プロセス、ネットワーク管理、アプリケーション設定にかかわる設定ファイルを説明します。

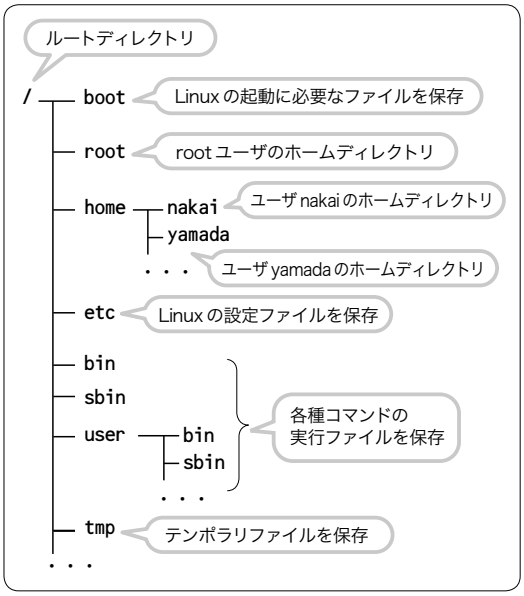
そして、最後のPart6では、少し未来の話をします。Part5までは、現在広く使われているLinuxディストリビューションである、Red Hat Enterprise Linux 6（RHEL6）を例に解説を進めますが、本年中には、新バージョンのRHEL7が登場する予定です。RHEL7では、新機能の導入に伴って、これまでとは設定方法／操作方法が変わる部分があります。その中でもとくに大きな変化をもたらす、「systemd」について解説します。

Linuxディレクトリの歩き方

Linuxサーバでは、設定ファイル、各種コマンドの実行ファイルなど、さまざまなファイル群がツリー状のディレクトリに整理して保存されます。RHEL6の場合は、**図1**のような構成になります。Linux本体とアプリケーションの設定ファイル群の多くは、ディレクトリ「**/etc**」の直下と、さらにその下のサブディレクトリに保存されます。アプリケーションによっては、その他の独自のディレクトリに保存するものもあります。

これらディレクトリ内の設定ファイルを各種

▼図1　Linuxの主要なディレクトリ



コマンドで編集するわけですが、ファイルを指定する際は「フルパス」と「相対パス」の2種類の方法があります(**図2**)。フルパスは、ルートディレクトリからの道筋を「/」区切りで示すもので、「**/etc/sysconfig/network**」のような形式です(「network」は、ホスト名が記載された設定ファイル)。最初の「**/**」だけは区切り文字ではなく、すべてのディレクトリの出発点である「ルートディレクトリ」を表すところがちょっとユニークです。

一方、Linuxにログイン中のユーザには、「現在地」を表す「カレントディレクトリ」が割り当てられます。「相対パス」は、カレントディレクトリからの道筋を「/」区切りで表したものです。たとえば、カレントディレクトリが「**/etc**」の場合、先と同じファイルは「**./sysconfig/network**」となります。頭の「**.**」はカレントディレクトリを表す記号ですが、「**./**」を省略して、「**sysconfig/network**」と書いても同じ意味になります。

ちなみに、カレントディレクトリのファイルを指定する際は、ファイル名だけで「file01.txt」のように記載しますが、これは「**./file01.txt**」の「**./**」を省略したものと考えられます。ファイル名の指定1つにも、このような理論的(?)根拠があるあたりは、UNIX/Linuxらしい気持ちよさが感じられます。

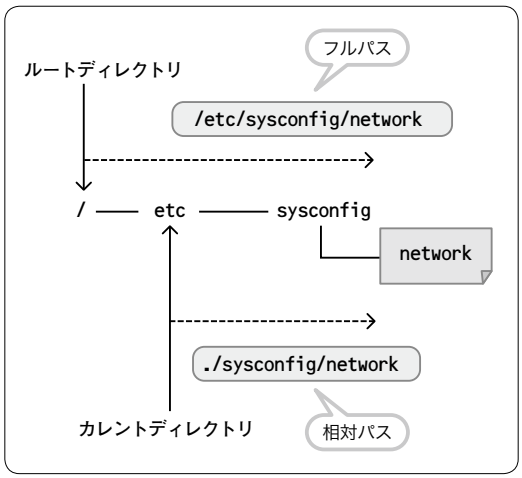
もう1つ覚えておくと良いのが、親ディレクトリを表す「**..**」という記号です。たとえば次は、ディレクトリ「**/etc/sysconfig/network-scripts**」をカレントディレクトリに変更して、その中の設定ファイル「**ifcfg-eth0**」を編集する例です。Part4で説明する、NIC(ネットワークインターフェース)のIPアドレスを設定する際の定番作業です。

```
# cd /etc/sysconfig/network-scripts
# vi ifcfg-eth0
```

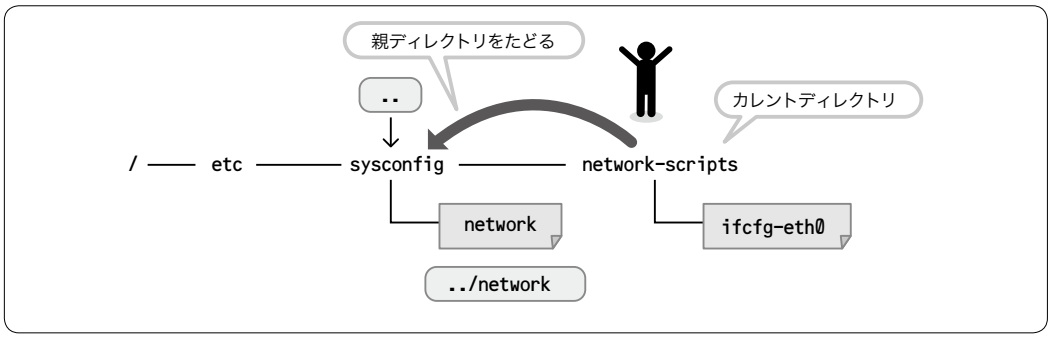
そしてこのあと、ホスト名を設定するために、1つ上のディレクトリにある設定ファイル「**/etc/sysconfig/network**」を修正しないといけないことに気がつきます。再度、カレントディレクトリを変更するという方法もありますが、次のように「**..**」を使った相対パスを利用するとスマートです(**図3**)。

```
# vi ../network
```

▼図2　フルパスと相対パス



▼図3　親ディレクトリを経由する相対パス



念のために補足すると、先のcd(Change Directory)コマンドは、カレントディレクトリを変更するコマンドです。cdコマンドで、「..」を使うこともできますので、次のようにも対応できます。

```
# cd ..
# vi network
```

このように、相対パスを駆使してディレクトリを歩きまわっていると、自分がどこにいるのかわからなくなることがあります。そのようなときは、pwd(Print Working Directory)コマンドで、現在のカレントディレクトリを確認してください[注1]。

```
# pwd
/etc/sysconfig
```

ディレクトリとパーティションの役割

ここで、**図1**に示したディレクトリの役割を簡単に整理しておきます。図の上から順に説明します。まず、「**/boot**」は、Linuxを起動する際に、最初に必要となるファイル群が入っています。サーバ起動時に最初に起動するブートローダ、そして、ブートローダが読み込むLinuxカーネルのバイナリファイルなどです。これらのファイルを誤って変更すると、サーバが起動しなくなる恐れがありますので、この中のファイルは不用意に変更しないように注意してください。

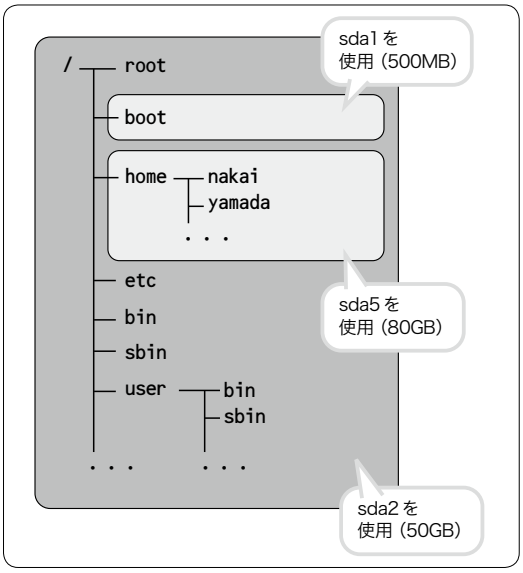
「**/root**」と「**/home/<ユーザ名>**」は、それぞれ、rootユーザと一般ユーザのホームディレクトリです。Linuxにログインした直後のカレントディレクトリは、これらのホームディレクトリです。システム全体の動作に関係しない個人的なファイルは、ホームディレクトリの下に保存するようにしましょう。「**/etc**」は先に説明したとおり、Linuxの主要な設定ファイルが保存されます。

「**/bin**」「**/sbin**」「**/usr/bin**」「**/usr/sbin**」には、各種コマンドの実行ファイルが収められています。「**/sbin**」と「**/usr/sbin**」には、おもにシステム管理者が使用するコマンド、そして、「**/bin**」と「**/usr/bin**」には、一般ユーザも使用するコマンドがあります。ちなみに、「**/sbin**」と「**/usr/sbin**」、そして、「**/bin**」と「**/usr/bin**」の使い分けには歴史的な経緯があります。このあとのディスクパーティションに関するコラムを参考にしてください。

最後に「**/tmp**」は、各種アプリケーションが一時ファイルを作成するためのディレクトリです。RHEL6の場合、「**/tmp**」以下にある長期間アクセスのないファイルは、tmpwatchというツールが定期的に削除するようになっています。逆に言うと、勝手に消えて困るファイルは「**/tmp**」には保存しないようにしてください。

ここで、ディスクパーティションとディレクトリの関係を整理しておきます。Linuxでは、1つの物理ディスクを複数のパーティションに分割して使用します。これは、Windowsでも同じです。しかしながら、Windowsではそれぞれのパーティションが「Cドライブ」「Dドライブ」のように別々のドライブとして認識され

▼図4　ディレクトリとパーティションの関係



注1）よく考えると「現在のカレントディレクトリ」とは、「頭痛が痛い」のような変な表現ですが……。

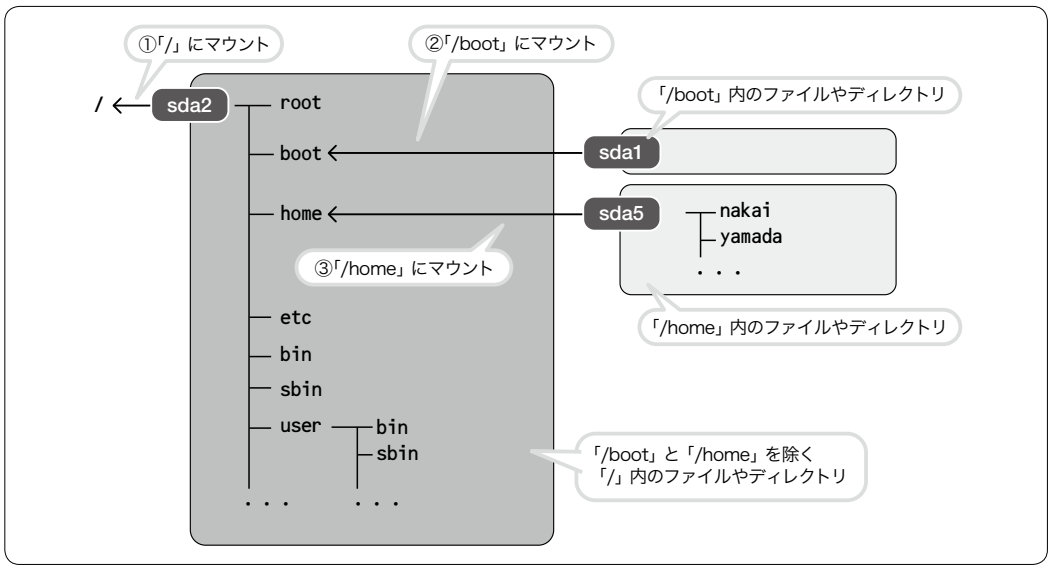
るのに対して、Linuxでは、**図1**のようなディレクトリツリーの中に複数のパーティションが混在してマウントされます。

図4の例では、ルートディレクトリ「**/**」には、パーティション「sda2」がマウントされていますが、「**/boot**」と「**/home**」には、別のパーティション「sda1」と「sda5」がマウントされています。**図4**は、サーバ起動時にパーティションがマウントされる順序(**図5**)と比較するとよく理解できます。

サーバが起動してLinuxカーネルが動き始めると、まず初めに「ルートファイルシステム」を格納したパーティション(この例では「sda2」)をルートディレクトリ「**/**」にマウントします。この時点では、「**/boot**」と「**/home**」の2つのディレクトリは存在こそしていますが、その中身は空になっています。このあと、それぞれのディレクトリに対応するパーティション(「sda1」と「sda5」)をマウントすると、パーティションの中身がマウント先のディレクトリの下に現れます。

サーバ起動時にマウントするパーティション

▼図5　ファイルシステムのマウントの流れ



▼図6　/etc/fstabの内容とblkidコマンドによる確認(コメント行は省略)

UUID　マウントポイント

/etc/fstab

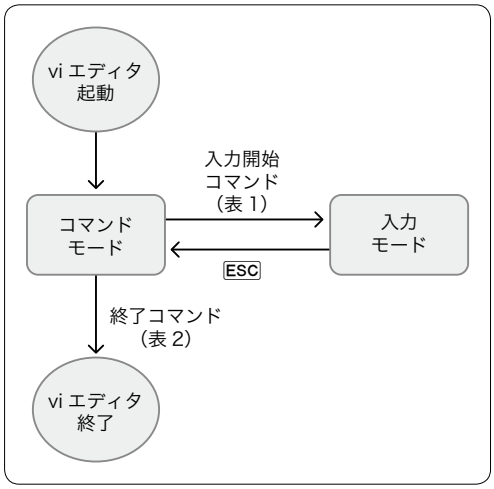
```
UUID=efb5bd05-3be7-4429-8635-feeddb546ed4 /                                 ext4    defaults        1 1
UUID=e2fddcd0-0f9b-4f4a-bfdb-ee8a1729c891 /boot                             ext4    defaults        1 2
UUID=ba11ad10-d5e7-4054-8996-bb54b48361ed /home                             ext4    defaults        1 2
UUID=9ec4306a-8824-4dcf-a25c-a493a95f5bc0 swap                              swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

```
# blkid
/dev/sda1: UUID="e2fddcd0-0f9b-4f4a-bfdb-ee8a1729c891" TYPE="ext4"
/dev/sda2: UUID="efb5bd05-3be7-4429-8635-feeddb546ed4" TYPE="ext4"
/dev/sda3: UUID="9ec4306a-8824-4dcf-a25c-a493a95f5bc0" TYPE="swap"
/dev/sda5: UUID="ba11ad10-d5e7-4054-8996-bb54b48361ed" TYPE="ext4"
```

は、設定ファイル「/etc/fstab」に記載されます。RHEL6の場合は、図6のようにパーティションの「UUID」とマウント先のディレクトリ(マウントポイント)が記載されています。それぞれのUUIDに対応するディスクパーティションは、図6の下にある、blkid(BLocK ID)コマンドで確認します。

ちなみに、UUIDの値は、パーティションをフォーマットしてファイルシステムを作成する際に、ファイルシステムの属性情報として記録されます。以前は「/etc/fstab」には、「/dev/sda1」などのディスクパーティションの名前が記載されていましたが、あとからディスクを追加した際にパーティション名の割り当てが変わってサーバが起動しなくなるなどの問題がありました。一方、ファイルシステムに記録されたUUIDは、あとから変化することはありませんので、このような問題を避けることができます。blkidコマンドで表示されるパーティション名との紐付けはサーバ起動時に更新されるようになっています。

▼図7 viエディタのモード切り替え



できる! viエディタ

ここで、ちょっと寄り道して、viエディタの使い方を確認しておきましょう。先のコマンド例にも出てきましたが、既存の設定ファイルを編集する際は、次のコマンドでviエディタを起動します。<ファイル名>の部分は、先に説明した「フルパス」、もしくは、「相対パス」が入ります。

```
# vi <ファイル名>
```

初めてviエディタを起動したときは、キーを押しても文字が入力されない、終了方法がわからない、といったことで困ってしまうのが「お約束」ですが、viエディタの操作の基本は、図7のとおりです。起動直後は、各種コマンドを

Column

分けるべきか分けざるべきか、それが問題?

Linuxサーバを構築する際、多くの場合、ディスクパーティションを細かく分けることはしません。最低限でも「/boot」と「/」、そして、スワップ領域の3つのパーティションがあれば十分です。一般ユーザが多量のファイルを保存する可能性がある場合は、「/home」を分けることもあります。万一、「/home」の使用量が100%になっても、ほかのディレクトリの空き容量には影響を与えないためです。

一方、古いUNIXサーバでは、「/usr」「/var」「/opt」「/home」「/tmp」など、ディレクトリごとに細かくパーティションを分ける習慣がありました。当時は、1つのファイルシステムの最大容量があまり大きくなかったほか、ファイルシステムが破損するという障害も多く、パーティションを分けることで、障害の影響範囲を局所化するなどの理由があったのです。このとき、図5を見るとわかるように、いくつかのファイルシステムが破損しても「/」にマウントする「ルートファイルシステム」が生き残っていれば、最低限システムを起動することができます。そこで、ルートファイルシステムに含まれる「/bin」「/sbin」には、システムのメンテナンスに必要な最低限のコマンド類を入れておき、その他の大量のコマンドは、別ファイルシステムとなる「/usr」以下に保存するという使い分けがなされていました。

ただし現在のLinuxでは、「/usr」を別ファイルシステムにすることはありませんので、この使い分けにはほとんど意味がありません。そのためPart6で紹介するRHEL7では、「/bin」と「/sbin」が、それぞれ「/usr/bin」と「/usr/sbin」へのシンボリックリンクに変更されています。

受け付ける「コマンドモード」ですので、**表1**の「入力開始コマンド」を押して、文字入力を開始します。入力が終わったらESCでコマンドモードに戻って、**表2**の「保存／終了コマンド」でファイルを保存・終了します。

入力モードでは、矢印キーでカーソル移動をしますが、コマンドモードでは、IJKLで「←↓↑→」に移動するほか、**表3**のコマンドで移動することもできます。そのほかに、コマンドモードで利用できる便利なコマンドを**表4**～**7**にまとめてありますので、本誌をコピーして「クイックリファレンス」として活用してください。

そのほかに、u（Undoコマンド：直前の操作を取り消す）、Ctrl+G（編集中のファイル情報を表示）、「:set number」「:set nonumber」（行番号の表示／非表示）などを覚えておくと良いでしょう。

設定ファイル編集に役立つコマンド

設定ファイルの編集に役立つのは、viエディタだけではありません。findコマンドで特定のファイル名の設定ファイルを探し出す、grep（Global Regular Expression Print）コマンドで特定の設定項目を抜き出して確認する、などは定番でしょう。とくにfindコマンドでは、指定の時間以内に修正されたファイルを探し出すことができます。次は、useraddコマンドで新規ユーザ「nakai」を作成したあと、「**/etc**」以下で、5分以内に変更されたファイルを検索する例です。ユーザの作成に伴って、自動的に変更された設定ファイルが確認できます。

```
# useradd nakai
# find /etc -mmin -5
/etc
/etc/group
/etc/shadow
/etc/gshadow
/etc/passwd
```

viエディタの編集コマンド（**表1**～**7**）と、findやgrepなどのテキスト編集コマンドは筆者の著書[注2]で徹底解説していますので、そちらも参考にしてください。

そのほかに覚えておくと良いのが、diff（DIFFerence）コマンドです。これは、2つの

▼表1　主な入力開始コマンド

コマンド	説明	覚え方
i	カーソルの前に挿入開始	Insert
a	カーソルの後ろに挿入開始	Append
I	行頭に挿入開始	Insert
A	行末に挿入開始	Append
o	カーソルの次の行を空けて挿入開始	Open
O	カーソルの前の行を空けて挿入開始	

▼表2　主な保存／終了コマンド

コマンド	説明	覚え方
:w	ファイルを保存する	Write
:w 〈ファイル名〉	指定のファイル名で保存する	Write
:q	終了する（ファイルが未保存だと終了できない）	Quit
:q!	ファイルを保存せずに強制終了する	Quit
:wq	ファイルを保存して終了する	Write-Quit
ZZ	:wqと同じ	ZZZ…（もう寝る）

▼表3　主な移動コマンド

コマンド	説明	覚え方
0	行頭に移動	——
^	0と同じ	——
$	行末に移動	——
gg	ファイルの先頭行に移動	Go
G	ファイルの最終行に移動	
[Ctrl]+[F]	1画面下に移動	Forward
[Ctrl]+[B]	1画面上に移動	Back
[数字]G	数字で指定の行に移動	Go
w	次の単語の先頭に移動	Word

▼表4　主な削除コマンド

コマンド	説明	覚え方
x	カーソル位置の一文字を削除	——
X	カーソルの前の一文字を削除	——
dd	カーソルの行全体を削除	Delete
d（移動コマンド）	移動先までをまとめて削除	
v（hlで移動）d	移動した部分の文字を削除	Visual
V（jkで移動）d	移動した部分の行を削除	
J	行末の改行を削除（次の行と結合）	Join

注2）中井悦司 著『「独習Linux専科」サーバ構築／運用／管理』技術評論社，2013年．

▼表5　主なコピーコマンドとペーストコマンド

コマンド	説明	覚え方
y(移動コマンド)	移動先までをコピー	Yank
yy	カーソルの行全体をコピー	Yank
v(hlで移動)y	移動した部分の文字をコピー	Visual
V(jkで移動)y	移動した部分の行をコピー	Visual
p	カーソルの直後にペースト	Paste
P	カーソルの直前にペースト	Paste

▼表6　主な検索コマンド

コマンド	説明
/文字列	カーソル位置から後ろに文字列を検索
/	同じ文字列を検索
n	「/」と同じ
N	同じ文字列を逆方向に検索

▼表7　主な置換コマンド

コマンド	説明
:s/文字列1/文字列2/	カーソル行の最初の「文字列1」を「文字列2」に置換
:s/文字列1/文字列2/g	カーソル行のすべての「文字列1」を「文字列2」に置換
:s/文字列1/文字列2/gc	カーソル行のすべての「文字列1」を「文字列2」に置換(確認つき)
:%s/文字列1/文字列2/	各行の最初の「文字列1」を「文字列2」に置換
:%s/文字列1/文字列2/g	すべての「文字列1」を「文字列2」に置換
:%s/文字列1/文字列2/gc	すべての「文字列1」を「文字列2」に置換(確認つき)

▼図8　diffコマンドによる変更部分の確認

```
# cp /etc/shadow /tmp/shadow
# passwd nakai
...(中略)...
# diff -u /tmp/shadow /etc/shadow
--- /tmp/shadow 2014-03-06 14:10:44.562998567 +0900
+++ /etc/shadow 2014-03-06 14:10:51.881001416 +0900
@@ -26,4 +26,4 @@
 sshd:!!:16135::::::
 tcpdump:!!:16135::::::
 oprofile:!!:16135::::::
-nakai:!!:16135:0:99999:7:::
+nakai:$6$k8eT4Lr/$u2DtqL...(中略)...:16135:0:99999:7:::
```

← 変更部分

設定ファイルの異なる部分を抽出します。設定ファイルを編集する前に、cp(CoPy)コマンドでバックアップを取っておき、編集後のファイルと比較することで、変更内容をチェックできます。使い方は次のとおりです。

```
# diff -u <変更前ファイル> <変更後ファイル>
```

図8では、ユーザ「nakai」のパスワードを設定した際に、設定ファイル「**/etc/shadow**」がどのように変わるか確認しています。行頭に「**-**」のある行が削除された行で、「**+**」のある行が新しく追加された行です。実際には、同じ行を書き換えていますが、出力上は、削除後に追加した形になっています。新旧のファイルを横にならべて変更部分をマークする、sdiff(Side-by-side DIFFerence)コマンドもあります。

まとめ

Part1では、設定ファイル編集の基礎として、ディレクトリやディスクパーティションの考え方、そして、viエディタの使い方を説明しました。また、find、grep、diffなど、設定ファイルの編集を助けるコマンドも紹介しました。viエディタやこれらのコマンドは、長いLinux人生を助けるすばらしい伴侶です。体の一部、生活の一部になるまで、徹底的に使いこなしてください。

次のパートからは、各種設定ファイルの役割と設定方法を説明していきます。SD

Part 2

大事な基礎部分

ユーザ管理と設定ファイル

複数の人間が参加する開発の環境として、またはサーバとしてLinuxを利用する場合、ユーザの登録や切り替えについての知識は必要不可欠となります。本パートでは、Linuxにおいてのユーザの分類とグループの基礎について説明した後、ユーザ管理に関する設定ファイルの見方、編集の仕方について、セキュリティにも触れながら詳しく見ていきます。

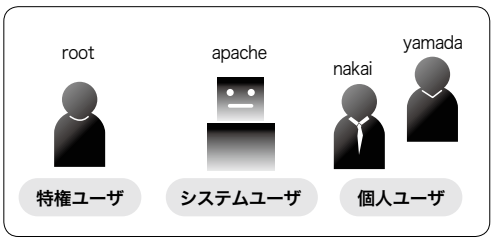
システム管理の土台としてのユーザ管理

Part2では、ユーザ管理にかかわる設定ファイルを紹介します。多数のユーザが同時にログインして使用するワークステーションはもちろんのこと、管理ユーザ以外はログインしない、サーバ用途のLinuxにおいてもユーザ／グループ管理の知識は必須です。

たとえば、サーバ上で稼働するアプリケーションは、多くの場合、そのアプリケーション専用のユーザ権限で動作します。このとき、設定ファイルのアクセス権が正しく設定されていないとどうなるでしょうか？　アプリケーションが設定ファイルを読み込めずに停止するなどは、定番の失敗例です。あるいは逆に、読めてはいけないはずのファイルが読めてしまい、セキュリティ上の問題が発生することもあります。

ここでは、ユーザ管理にかかわる設定ファイルを中心に解説します。このあとのPart5では、アプリケーションが参照するファイルのアクセス権についても説明します。

▼図1　ユーザアカウントの分類



▼表1　UIDの割り当て範囲

ユーザの種類	UIDの範囲
rootユーザ	0
システムユーザ	1～499
個人ユーザ	500～60000

▼表2　GIDの割り当て範囲

グループの種類	GIDの範囲
rootグループ	0
システムグループ	1～499
個人グループ	500～60000

ユーザとグループの基礎

Linuxに登録されるユーザ、すなわちユーザアカウントは、**図1**の3種類に分類されます。あらゆる操作が許される「特権ユーザ」のrootユーザ、Linux上で作業を行う個人に対応する「個人ユーザ」、そして、アプリケーションを実行するための「システムユーザ」です。各ユーザアカウントにはユニークな「UID（ユーザID）」が割り当てられており、ユーザの種類によって、値の範囲が決められています。RHEL6では、**表1**のようになります。

また、各ユーザは、少なくとも1つのグループに所属する必要があります。これをプライマリグループと呼びます。RHEL6では、各ユーザについて、ユーザ名と同じ名前のグループを作成して、それをプライマリグループに設定するようになっています。各グループにはユニークな「GID（グループID）」が割り当てられますが、**表1**の3種類のユーザに対応して、**表2**のように割り当て範囲が決まっています。

そして、各ユーザは、プライマリグループとは別に、追加のグループ（セカンダリグループ）に属することもできます。**図2**のユーザ「nakai」は、

セカンダリグループとして「wheel」に属しています。1つのグループに対して、複数のユーザがセカンダリグループとして所属することもあります。各ユーザが所属するグループやUID、GIDなどの情報は、**図3**のidコマンドで確認します。

ユーザ管理にかかわる設定ファイル

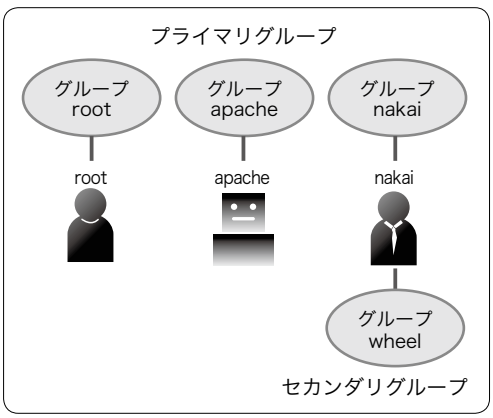
それでは、ユーザ管理にかかわる設定ファイルを見ていきます。基本になるのは、ユーザ情報、グループ情報、パスワード情報を収めた「**/etc/passwd**」「**/etc/group**」「**/etc/shadow**」の3点セットです。

まず、ユーザ情報とグループ情報の書式を**図4**に示します。「**/etc/passwd**」は、1行が1ユーザに対応しており、各行は、「:」区切りでいくつかの情報が並んでいます。ここでは、個人ユーザ「nakai」とシステムユーザ「apache」の例を記載しています。各項目の意味は**図4**のとおりですが、2列目の「x」は、このユーザのログインパスワードが、別ファイル「**/etc/shadow**」に暗号化形式で保存されていることを示します。

この2つの例を見比べると、少しおもしろい違いに気がつきます。まず、個人ユーザである「nakai」のログインシェル(ログイン時に起動するシェル)は、Linux標準の「**/bin/bash**」が指定されています。一方、システムユーザ「apache」(Apache HTTPサーバを実行するためのユーザ)では、「**/sbin/nologin**」という変わったシェルが指定されています。これは、本物のシェルではなく、ログインするとすぐにログアウトするプログラムです。システムユーザは、ログインに使用するわけではありませんので、誰かがイタズラでログインしようとしてもできない仕掛けになっています。

ホームディレクトリについては、個人ユーザ「nakai」には、標準的な「**/home/nakai**」が指定されていますが、システムユーザ「apache」は「**/var/www**」になっています。前述のようにシステムユーザはログインできませんので、そもそもホームディレクトリは必要ありません。そこで、対応するアプリケーションのデータ領域のディレクトリをホームディレクトリに指定するのが慣例になっています。フルネームについては、この例では、システムユーザのみに

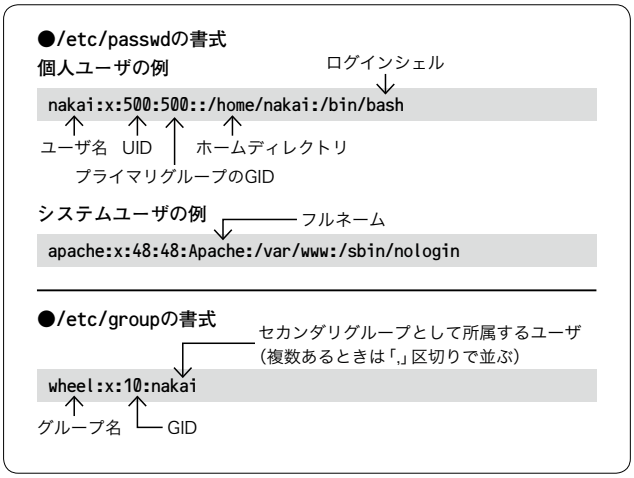
▼図2　グループの割り当て例



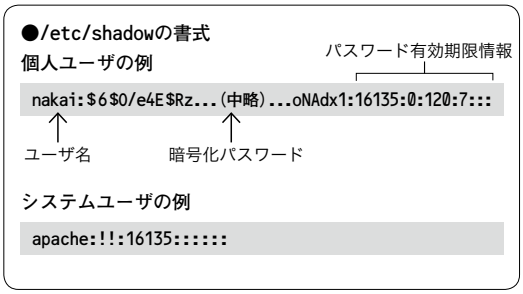
▼図3　idコマンドによる確認例

```
# id root
uid=0(root) gid=0(root) 所属グループ=0(root)
# id nakai
uid=500(nakai) gid=500(nakai) 所属グループ=500(nakai),10(wheel)
```

▼図4　ユーザ情報／グループ情報ファイルの書式



▼図5　パスワード情報ファイルの書式



設定されていますが、これには特別な意味はありません。個人ユーザについてもフルネームを登録することができます。

「/etc/group」は、項目数が少なくて単純です。2列目の「x」は、グループパスワードに関する項目ですが、これは、現在ではほとんど利用しない機能です。

パスワード情報の書式は、**図5**のとおりです。ユーザ情報と同様に1行が1ユーザに対応しており、ユーザ名、暗号化パスワード、パスワードの有効期限情報が並びます。システムユーザ「apache」の場合は、暗号化パスワードの部分が「!!」になっています。これはパスワードが無効化されているという意味で、どのようなパスワードを入力しても認証に成功することはありません。

次に重要なのが、これらファイルの編集方法です。既存のユーザやグループの情報を確認するために、ファイルの内容を見るのはかまいませんが、その内容をviエディタなどで直接変更してはいけません。たとえば、新規ユーザを作成する際は、「/etc/passwd」に必要な情報を追加するだけではなく、そのユーザのホームディレクトリを作成するなど、関連する作業がたくさんあります。useraddコマンドでユーザを追加すると、これらの作業がすべてまとめて実施されます。

仮に、「/etc/passwd」だけを手で変更した場合、関連するほかの設定との食い違いが発生して、システムが正常に稼働しなくなる恐れがあります。ユーザ／グループに関する設定は、専用の管理コマンドで行うように心がけてください。とくにユーザとグループを作成する際は、それぞれ、useraddコマンドとgroupaddコマンドを使用します。**表3**、**4**のオプションで必要な情報を指定すると、それに合わせて、「/etc/passwd」「/etc/group」などに適切なエントリが追加されます。

▼表3　useraddコマンドの主なオプション

オプション	説明
-u〈UID〉	UIDを指定(省略時は未使用番号を割り当てる)
-d〈ディレクトリ〉	ホームディレクトリを指定(省略時は「/home/〈ユーザ名〉」)
-c〈フルネーム〉	フルネームを指定
-M	ホームディレクトリを作成しない
-r	システムアカウントを作成(500番未満のUIDを割り当てる)
-g〈グループ名〉(もしくはGID)	プライマリグループを指定する(省略時はユーザ名と同じグループを自動作成)
-G〈グループ名〉(もしくはGID)	セカンダリグループを指定する(「,」区切りで複数指定が可能)

▼表4　groupaddコマンドの主なオプション

オプション	説明
-g〈GID〉	GIDを指定(省略時は未使用番号を割り当てる)
-r	システムグループを作成(500番未満のGIDを割り当てる)

既存のユーザ／グループの情報を変更する際は、usermodコマンドとgroupmodコマンドを使用します。変更可能な情報は、**表3**、**4**とほぼ同じです。詳細は、これらコマンドのmanページを参照してください。

次に、「/etc/shadow」に記載されるパスワードの有効期限は、chage(CHange AGE)コマンドで設定します。主なオプションは**表5**のとおりです。一般には、rootユーザがuseraddコマンドで新規ユーザを作成して、初期パスワードを設定したあとに、続けて有効期限の設定を行います。**図6**では、passwdコマンドで初期パスワードを設定したあとに、chageコマンドで有効期限を設定しています。

始めの-Mオプションでは、有効期限を120日に設定しています。続いて、-dオプション(パスワードの最終更新日)に「0」を指定していますが、これは、初回ログイン時にパスワードの

▼表5　chageコマンドの主なオプション

番号	オプション	説明
(1)	-d〈日付〉	パスワードの最終更新日(「YYYY-MM-DD」の形式で指定)
(2)	----	パスワードの有効期限が切れる日付((1)と(6)から決まる日付を表示)
(3)	-I〈日数〉	パスワードの有効期限が切れたあと、パスワードを変更せずにこの日数が経過するとログインを禁止する
(4)	-E〈日付〉	アカウントの使用を停止する日付(「YYYY-MM-DD」の形式で指定)(無期限にするには「-1」を指定)
(5)	-m〈日数〉	パスワード変更後に再変更が可能になるまでの日数。「0」の場合はいつでも変更可能
(6)	-M〈日数〉	パスワードの有効期限日数。パスワード変更後、この日数を過ぎるとパスワードの変更を要求する
(7)	-W〈日数〉	パスワードの有効期限が切れる前に、警告の表示を開始する日数

▼図6　パスワード有効期限の設定

```
# passwd nakai
ユーザー nakai のパスワードを変更。
新しいパスワード:  ←――――――――新規パスワード入力
新しいパスワードを再入力してください:  ←新規パスワード再入力
passwd: 全ての認証トークンが正しく更新できました。
# chage -M 120 nakai
# chage -d 0 nakai
# chage -l nakai
(1) Last password change                                    : password must be changed
(2) Password expires                                       : password must be changed
(3) Password inactive                                      : password must be changed
(4) Account expires                                        : never
(5) Minimum number of days between password change         : 0
(6) Maximum number of days between password change         : 120
(7) Number of days of warning before password expires      : 7
```

変更を強制する効果があります。このあと、ユーザ「nakai」がサーバにログインすると、新しいパスワードを設定するように指示がでます。最後の-lオプションは、有効期限の設定内容を確認しています。**図6**内の(1)～(7)の項目について、**表5**に対応する番号を記載してあります。

セキュリティ設定ファイル

ユーザ管理に関連したセキュリティ設定ファイルも重要です。まず、企業システムなどでユーザ認証を細かく制御する際に使用するのが、ディレクトリ「**/etc/pam.d**」の下にまとめて保存されている、PAM(Pluggable Authentication Modules)の設定ファイルです。単純な文字列のパスワードの使用を禁止したり、誤ったパスワードを何度も入力すると、それ以降のログインを禁止するなどの制御ができます。

たとえば、設定ファイル「**/etc/pam.d/system-auth**」を開くと、**リスト1**のように、先頭が「password」で3列目が「pam_cracklib.so」と「pam_unix.so」の2行が見つかります。これらの設定値を**リスト1**のとおりに変更すると、「パスワードは最低8文字(minlen=8)で、大文字を1文字以上(ucredit=-1)と数字を1文字以上(dcredit=-1)含まないといけない」「直近のものに加えて、さらに過去2回(remember=2)のパスワードの再利用を禁止する」という条件になります。

同じく、設定ファイル「**/etc/pam.d/su**」を開き、**リスト1**の行のコメントアウトを削除します。すると、グループ「wheel」に属するユーザのみが、su(Switch User)コマンドでrootユーザに変更できるようになります。PAMモジュールの詳細は、筆者の著書[注1]に詳しい説明があります。

ちなみにsuコマンドは、個人ユーザとしてログイン中のユーザが一時的にrootユーザに切り

注1) 中井悦司 著『プロのためのLinuxシステム構築・運用技術』技術評論社, 2010年.

▼リスト1　PAMの設定例

/etc/pam.d/system-auth

```
password    requisite     pam_cracklib.so try_first_pass retry=3 type= minlen=8 dcredit=-1 ucredit=-1
lcredit=0 credit=0
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok remember=2
```

/etc/pam.d/su

```
#auth       required      pam_wheel.so use_uid
```

↑このコメントアウト(#)を削除

替えて作業するためのコマンドです。rootユーザ専用の管理コマンドを実行する際などに使用します。切り替える際は、次のように、rootユーザのパスワードを入力する必要があります。

```
$ su -  ←個人ユーザから実行
パスワード:  ←rootパスワードを入力
#  ←rootユーザに変わった
```

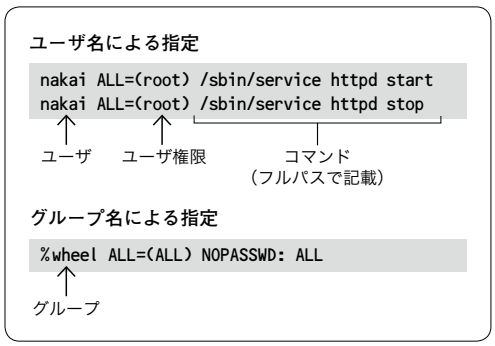
しかしながら、管理者以外の一般ユーザに対しては、rootユーザのパスワードを教えることなく、特定のコマンドだけをrootユーザの権限で実行できるようにしたい場合もあります。これを実現するには、設定ファイル「**/etc/sudoers**」を活用します。

図7は、この設定ファイルに追加する項目のサンプルです。「ユーザ名による指定」では、ユーザ「nakai」に対して、httpdサービスの起動・停止コマンドの実行を許可しています。このとき、ユーザ「nakai」は、次のように、sudoコマンドを通じて許可されたコマンドを実行します。

```
$ sudo -l
[sudo] password for nakai: ←nakaiのパスワードを入力
...(中略)...
ユーザー nakai
    は次のコマンドをこのホスト上で実行できます:
    (root) /sbin/service httpd start
    (root) /sbin/service httpd stop

$ sudo service httpd start
$ sudo service httpd stop
```

始めに、-lオプションで許可されたコマンドの一覧を確認して、そのあと、実際にコマンドを実行しています。rootユーザのパスワードは

▼図7　/etc/sudoersの書式



不要ですが、安全のために、自分自身のパスワードの入力を求めるようになっています。ユーザ「nakai」でログインした端末を放ったらかしにした場合でも、他人が勝手にsudoコマンドを利用できないためのしくみです。一度パスワードを入力すると、その後5分間は、パスワード入力なしにsudoコマンドが利用できます。

図7の例では、「**/etc/sudoers**」には、コマンドが引数を含めて記載されていますので、ユーザ「nakai」は、これ以外の引数でserviceコマンドを実行することはできません。コマンドのみを記載すると、任意の引数で実行できるようになります。また、「**/etc/sudoers**」には、コマンドをフルパスで記載する必要があります。フルパスがわからない場合は、whichコマンドで確認します。

```
# which service
/sbin/service
```

図7の「グループ名による指定」は、グループ「wheel」に属するユーザに対する設定です。ユーザ権限の部分とコマンドの部分がともに「ALL」になっていますので、グループ「wheel」に属するユーザは、任意のユーザ権限で、任意のコマンドを実行できます。さらに、「NOPASSWD:」

オプションが指定されているので、自分のパスワードを入力する必要もありません。

次は、ユーザ「nakai」から、ユーザ「apache」の権限で、idコマンドを実行する例です。つまらない例ですが、ユーザ「apache」の情報が表示されており、確かにユーザ権限が変わっていることがわかります。

```
$ sudo -u apache id
uid=48(apache) gid=48(apache) 所属グループ
=48(apache) ...(省略)...
```

最後に、設定ファイル「**/etc/sudoers**」を編集する際の注意について。このファイルを編集する際は、viエディタを使用するのではなく、「visudo」コマンドを使用します。このコマンドを引数なしで実行すると、既存ファイルの一時的なコピーが作成されて、その編集画面がviエディタで開きます。そのあと、編集を終えてviエディタを終了すると、設定内容の文法チェックが行われて、問題がなければ、「**/etc/sudoers**」に内容が反映されます。文法に間違いがある場合は、警告とともに、再度編集画面に戻るか、変更を破棄して終了するかの選択肢が表示されます。

「**/etc/sudoers**」に誤った設定があると、sudoコマンドが正しく利用できず、セキュリティ上の問題が起きる可能性もあります。このような問題を防止するために、visudoコマンドが用意されています。

ユーザ環境のカスタマイズ

最後にユーザ環境のカスタマイズにかかわる設定ファイルを紹介します。ユーザがLinuxにログインすると、ホームディレクトリにある「.bash_profile」と「.bashrc」の2つのスクリプトが実行されるようになっています。個人ユーザは、これらの中に環境変数やコマンドエイリアスの設定を自由に追加できます。

コマンドエイリアスは、長いコマンドに短い別名を付ける機能で、次のように設定します。

```
# alias la='ls -la'
```

この場合は、「la」と入力すると、「ls -la」というコマンドが実行されるようになります。

これらのファイルの大元は、ディレクトリ「**/etc/skel**」の中に入っています。useraddコマンドでユーザを作成すると、このディレクトリの中身がサブディレクトリを含めて、作成したユーザのホームディレクトリにコピーされます。すべてのユーザのホームディレクトリに共通に配布したいファイルがある場合は、事前に「**/etc/skel**」の中に入れておくと良いでしょう。

細かい点ですが、「.bash_profile」と「.bashrc」の使い分けにも注意があります。SSHなどでLinuxにログインすると、これらのスクリプトは「.bash_profile」→「.bashrc」の順に実行されます。一方、すでにログインした状態で、新たにシェル(bash)を起動した際は、「.bashrc」だけが実行されます。

とくにLinuxデスクトップ環境において、デスクトップ上で新しいコマンド端末を開いた際は、「.bashrc」のみが実行されます。一般的な設定項目は「.bashrc」に記載して、ログイン時に一度だけ実行したいものは「.bash_profile」に記載するという使い分けが必要です。

まとめ

Part2では、ユーザ管理の基礎となる設定ファイル群を説明しました。これらのファイルの多くは、viエディタで直接に編集するのではなく、専用の管理コマンドを通して設定するので、注意が必要です。それぞれに多数のオプションがありますが、すべてを覚える必要はありません。必要なオプションは、manページですぐに調べることができるので、まずは、その背後にある考え方をしっかりと理解しておきましょう。

次のパートでは、Linux全般の設定項目として、システム起動にかかわる設定ファイルを紹介します。システム起動のしくみと併せて理解しておくと良いでしょう。SD

Part 3

ちゃんと理解しておきたい
システム起動にかかわる設定ファイル

Part3では、Linuxサーバの起動処理を追いながら、関連する設定ファイルを解説していきます。これらのファイルは普段、自ら変更することは少ないかもしれません。しかし、ここをきちんと理解しておけば、Linuxの深い知識を得られるとともに、起動処理を調整したいときや、起動におけるトラブルで原因究明するときに役立ちます。この機会に頭に入れておきましょう。

システム起動の流れにみる設定ファイル

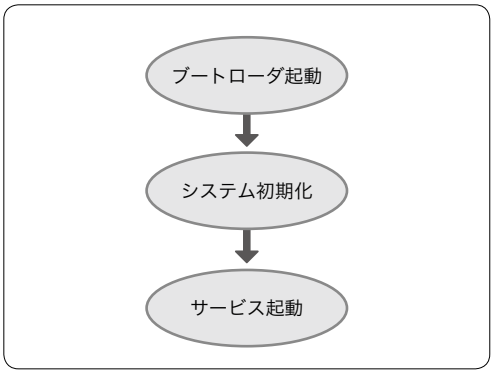
Linuxの起動処理は、**図1**の3つのステップに分かれます。ブートローダから始まるこのプロセスには、Linuxの根幹となる技術要素を理解するヒントが隠されており、「**/etc/fstab**」など、定番の設定ファイルの役割を理解するとともに、システムに問題が発生したときの対応にも役立ちます。

各ステップの内容を簡単にまとめると次のようになります。

①ブートローダ起動

サーバの電源を入れると、サーバに搭載されたシステムBIOSが起動して、ハードウェアの構成確認を行ったあと、ブートディスクから「ブートローダ」を読みだして起動します。RHEL6では、GRUB(GRand Unified Bootloader)と呼ばれるブートローダが標準的に使用されます。Linuxでは、複数のバージョンのカーネルをインストールしておき、起動時に使用するカーネルを選択できるようになっています。GRUBが起動するとサーバのコンソールに「Press any key to enter the menu」と表示されますが、ここで適当なキーを押すと、使用するカーネルの選択メニューが表示されます。その後、GRUBは選択されたカーネルをサーバのメモリに読み込んで起動します。ここで、GRUBの役割は終了です。

▼図1　Linux起動処理の流れ



②システム初期化

カーネルが起動すると、ルートファイルシステムをマウントしたあと、最初のプロセスである「**/sbin/init**」を起動します。その後、「**/sbin/init**」は、Upstartと呼ばれるしくみに従って、システム起動処理を実施していきます。最初に実行するのは、「**/etc/rc.d/rc.sysinit**」というシェルスクリプトです。このスクリプトは、fsckコマンドによるファイルシステムのチェックなど、おもにファイルシステム関連の初期化処理を行います。**Part1の図5**に示した、ファイルシステムのマウント処理、そして、スワップ領域の有効化もここで行われます。併せて、SELinuxのラベル付け処理なども実施します。

③サービス起動

最後に「**/sbin/init**」は、chkconfigコマンドで自動起動が設定された、各種サービスを起動していきます。「networkサービス」が起動したタイミングで、IPアドレス割り当てなどのネットワーク初期化が行われて、「sshdサービス」が起動したタイミングで、SSHによるリモートログインが可能になります。

それでは、この3つのステップに従って、設定ファイルの理解を深めていきましょう。

ブートローダの起動

先に説明したように、ブートローダ(GRUB)は、Linuxカーネルを読み込んで起動する役割を持ちます。カーネル本体のファイルを含めて、GRUBが必要とするファイル群は、すべて「/bootファイルシステム」に収められています。**Part1の図4**の例では、パーティション「sda1」のファイルシステムにあたります。

GRUBの設定ファイルは、この中にある「**/boot/grub/grub.conf**」です。「**/etc/grub.conf**」がGRUBの設定ファイルとして紹介されることもありますが、これは、先のファイルへのシンボリックリンクになっています。

リスト1は、2つのバージョンのカーネルが導入された環境における「grub.conf」の例です。「title」で始まる破線で囲まれたブロックが1つのバージョン(この例では「2.6.32-431.5.1.el6.x86_64」)に対する設定項目です。設定ファイルの1行目にある「default=0」は、最初のブロックをデフォルトで選択するという意味です。2個目のブロックをデフォルトにする場合は、「default=1」とします。

ブロック内の「kernel」で始まる行は、カーネルを起動する際のオプション、いわゆる「カーネルオプション」の指定です。通常は、これらのオプションを変更する必要はありませんが、サーバが起動しないなど、問題発生時の対応に向けて知っておくと良いのが、「root=」と「rhgb quiet」の部分です。

まず、「root=」は、ルートファイルシステムのディスクパーティションの指定です。この例では、**Part1の図6**で説明した「UUID」で指定されています。そのほかには、「root=/dev/sda2」のようにデバイス名で指定することも可能です。Linuxカーネルが起動すると、ここで指定されたパーティションを読み込み専用モードで「**/**」にマウントして、その中にある「**/sbin/init**」を最初のプロセスとして起動します。

「ファイルシステムは、『**/etc/fstab**』に従ってマウントされるのでは？」と疑問に思うかもしれませんが、実はルートファイルシステムだけは特別です。設定ファイル「**/etc/fstab**」はルートファイルシステムに入っていますので、ルートファイルシステムがマウントされていない状態にある起動直後のカーネルからは、参照できません。そこで、ルートファイルシステムのパーティションだけは、カーネルオプションでの指定が必要になります。この指定が誤っていると、サーバ起動時にカーネルパニックが発生して起動処理が中断します。

もう1つの「rhgb quiet」は、グラフィカルな起動画面を表示するオプションです。これらを削除すると、起動処理に伴う詳細なテキストメッ

▼リスト1　grub.confの例(コメント行は省略)

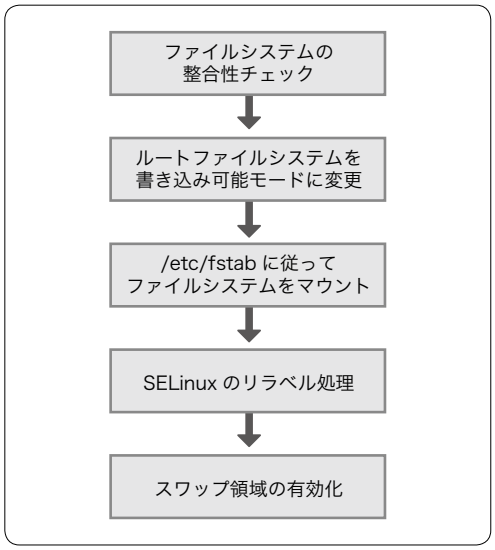
```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-431.5.1.el6.x86_64)
     root (hd0,0)
     kernel /vmlinuz-2.6.32-431.5.1.el6.x86_64 ro root=UUID=efb5bd05-3be7-4429-8635-feeddb546ed4 rd_NO_LUKS
rd_NO_MD crashkernel=auto  KEYBOARDTYPE=pc KEYTABLE=jp106 LANG=ja_JP.UTF-8 rd_NO_LVM rd_NO_DM rhgb quiet
     initrd /initramfs-2.6.32-431.5.1.el6.x86_64.img
title Red Hat Enterprise Linux (2.6.32-431.el6.x86_64)
     root (hd0,0)
     kernel /vmlinuz-2.6.32-431.el6.x86_64 ro root=UUID=efb5bd05-3be7-4429-8635-feeddb546ed4 rd_NO_LUKS
rd_NO_MD crashkernel=auto  KEYBOARDTYPE=pc KEYTABLE=jp106 LANG=ja_JP.UTF-8 rd_NO_LVM rd_NO_DM rhgb quiet
     initrd /initramfs-2.6.32-431.el6.x86_64.img
```

セージがコンソールに表示されますので、サーバ起動に失敗する際などの問題判別に活用できます。ただし、サーバ起動に失敗する場合は、そもそも「grub.conf」を編集することができません。このような場合は、サーバ起動時にGRUBのカーネル選択画面を表示したあと、[a]キーを押すとカーネルオプションを一時的に修正して起動することが可能です。

ファイルシステムの整合性チェック

最初のプロセスである「**/etc/init**」が起動すると、システム初期化スクリプト「**/etc/rc.d/rc.sysinit**」が実行されます。おもな処理内容は、**図2**のとおりです。

▼図2　rc.sysinitによる初期化処理



はじめに、ルートファイルシステムを含めて、「**/etc/fstab**」に記載のファイルシステムに対して、fsckコマンドによる整合性チェックを行います。これに成功すると、ルートファイルシステムを書き込み可能モードに変更します。ここまで読み込み専用モードになっていたのは、書き込み可能モードでマウントされた状態でfsckコマンドを実行すると、ファイルシステムを破損する恐れがあるためです。

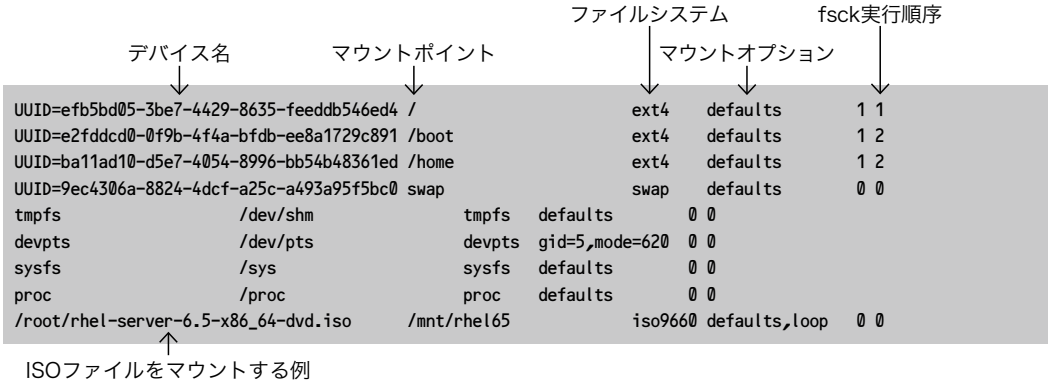
fsckコマンドによるチェックには、短時間で終わる簡易モードと、すべてのメタデータをチェックする完全モードがあります。通常は、簡易モードのチェックが行われますが、再起動したタイミングによって、突然、完全モードの処理が走って、サーバの起動に長時間かかるという問題が発生することがあります。これは、ファイルシステムのオプションで、「一定回数マウントした後」、あるいは、「一定期間が経過した後」に再起動すると、完全モードのチェックを行うように設定されているためです。

ファイルシステムのオプションは、**図3**のtune2fsコマンドで確認します。図中に白枠で示した「Maximum mount count」と「Check interval」が完全モードでのチェックを行うタイミングを決めるオプションです。インストーラで自動作成したファイルシステムでは、これらの値は「0」で無効化されていますが、mkfs.ext4コマンドで直接作成した場合は、そのほかの値が設定されている場合があります。既存の設定を変更し

▼図3　ファイルシステムオプションの確認

```
# tune2fs -l /dev/sdb1
tune2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          7c6e1f01-782d-49ff-a957-fb24b3081b4d
...(中略)...
Mount count:              4
Maximum mount count:      33          ←33回マウントすると完全チェック実施
Last checked:             Sat Mar  8 14:31:54 2014
Check interval:           15552000 (6 months)   ←6ヵ月ごとに完全チェック実施
Next check after:         Thu Sep  4 14:31:54 2014
...(以下省略)...
```

▼リスト2　/etc/fstabの設定例（コメント行は省略）



て無効化するには、次を実行します。

```
# tune2fs -c 0 -i 0 /dev/sdb1
tune2fs 1.41.12 (17-May-2010)
Setting maximal mount count to -1
Setting interval between checks to 0 seconds
```

このあとで説明する「**/etc/fstab**」の設定で、特定のファイルシステムをfsckコマンドによるチェックの対象外にすることも可能です。また、そのほかには、次の再起動時に限って、fsckコマンドの動作を変更する方法があります。まず、次のコマンドで、「**/fastboot**」という空ファイルを作成すると、fsckによる整合性チェックを行いません。

```
# touch /fastboot
```

逆に、次のコマンドで、「**/forcefsck**」という空ファイルを作成すると、強制的に完全モードでの整合性チェックを行います。

```
# touch /forcefsck
```

また、「**/fsckoptions**」というファイルにfsckコマンドのオプションを記載することで、実行時のオプションを追加することができます。これらのファイルは、システム起動時に削除されるので、次に再起動する際には影響しません。

ファイルシステムのマウント処理

続いて、「**/etc/fstab**」に記載されたファイルシステムのマウント処理が行われます。「**/etc/fstab**」の書式は、**リスト2**のとおりです。Part1でも説明したように、デバイス名は、UUID、もしくは、「**/dev/sda1**」などのデバイス名で指定します。ファイルシステムは、RHEL6では標準的に「ext4（エクストフォー）」が使用されます。4行目にあるファイルシステムが「swap」の行は、ファイルシステムではなく、スワップ領域を表します。

マウントオプションは、ほとんどの場合「defaults」で十分ですが、必要な際は「**,**」区切りで追加します。たとえば、読み込み専用でマウントする際は、「defaults,ro」のように、「ro」オプションを追加します。

また、DVDメディアなどのISOファイルを保存しておき、その内容をサーバ起動時にマウントすることも可能です。**リスト2**の最後の行にあるように、デバイス名にISOファイルを指定して、ファイルシステムに「iso9660」、マウントオプションに「defaults,loop」を指定します。

「**/etc/fstab**」の各行の最後の数字は、システム起動時にfsckコマンドでファイルシステムをチェックする順序を指定します。通常は、ルートファイルシステムには「1」を指定して、その他のファイルシステムには「2」を指定します。この部分に「0」を指定すると、ファイルシステムのチェックは行われなくなります。

なお、その前の列の数字は、dumpコマンドの処理対象の指定です。現在では、dumpコマ

ンドを使用することはほとんどありませんので、ここは気にしなくてもかまわないでしょう。

SELinuxのリラベル処理

ここで、**図2**の「SELinuxのリラベル処理」について補足しておきます。Linuxのセキュリティ機能である、SELinuxが有効化された環境では、個々のファイルに付与された「セキュリティラベル」によって、プロセスからのファイルアクセスの制御を行います[注1]。最初からSELinuxを有効にしているサーバであれば問題ないのですが、SELinuxを無効化した状態のサーバにおいて、途中でSELinuxを有効化した場合などは、あらためて、それぞれのファイルに有効なセキュリティラベルを付与する必要があります。これが「リラベル処理」です。

リラベル処理をコマンドで実施する際は、fixfilesコマンド、もしくは、restoreconコマンドを使用します。次は、fixfilesコマンドですべてのファイルのリラベル処理を行う例です。

```
# fixfiles -F restore
```

ただし、リラベル処理は、SELinuxが有効化された状態でないと実施できません。一方、SELinuxを無効化状態から有効化状態に切り替える際は、設定ファイル「**/etc/selinux/config**」の「SELINUX=disabled」を「SELINUX=enforcing」に変更したあと、一度、サーバを再起動する必要があります。そこで、このような設定変更直後の起動時は、システム初期化スクリプト「rc.sysinit」の中で、fixfilesコマンドを自動的に実行するようになっています。この際、システム起動中のコンソールには、リラベル処理の進行状況を表すメッセージが表示されます。あるいは、「**/.autorelabel**」という名前の空ファイルを作成して再起動すると、SELinuxの設定変更とは無関係に、強制的にリラベル処理を実施することも可能です。

なお、RHEL6では、SELinuxはデフォルトで有効化されています。何らかの理由で一時的に無効化する際は、先の設定ファイルを編集するのではなく、次のコマンドを実行します。

```
# setenforce 0
# getenforce
Permissive
```

最初のsetenforceコマンドで、「Permissiveモード」に変更して、次のgetenforceコマンドで結果を確認しています。これは、SELinuxに違反する動作が起きた際は、ログファイルに記録するだけで、実際には動作を許可するモードです。設定ファイルで完全に無効化する場合と異なり、ファイルのラベル付けなどの処理は継続するので、再度、有効化する際に、サーバの再起動やリラベル処理を行う必要がありません。次のコマンドで、いつでもすぐに有効化状態に戻せます。

```
# setenforce 1
# getneforce
Enforcing
```

cronジョブの設定方法

「rc.sysinit」によるシステム初期化が完了すると、ランレベルに応じたさまざまなサービスの起動が始まります。ランレベルの設定は、設定ファイル「**/etc/inittab**」に記載されており、一般には、ランレベル3(テキストログインモード)、もしくは、ランレベル5(GUIデスクトップモード)のどちらかを指定します。次は、ランレベル3の設定例で、「**:**」区切りの2列名の数字がランレベルの指定になります。

```
id:3:initdefault:
```

各ランレベルで起動するサービスは、**図4**の

注1) SELinuxの初心者向けの解説は、Part1の注2の書籍を参照してください。

Column

設定ファイルのシンボリックリンクにご用心

本文では、SELinuxの設定ファイルは「**/etc/selinux/config**」であると説明しました。Webで検索すると、「**/etc/sysconfig/selinux**」が設定ファイルとして紹介されていることがありますが、これは、先の設定ファイルへのシンボリックリンクになっています。viエディタで編集する場合は、どちらのファイル名を使っても結果は同じですが、sedコマンドを駆使して編集すると、思わぬ問題が発生することがあります。

sedコマンドは、ファイル内の文字列をさまざまに置換する機能があります。次は、指定ファイル内の「SELINUX=enforcing」を「SELINUX=disabled」に置換する実行例です。

```
# sed -i "s/SELINUX=enforcing/SELINUX=disabled/" /etc/sysconfig/selinux
```

セキュリティ上好ましいものではありませんが、SELinuxを無効化しようというわけです。ところが、残念ながら、この後サーバを再起動してもSELinuxは有効化されたままになります。どこに問題があるのでしょうか？

これは、sedコマンドの-iオプションの動作に原因があります。-iオプションでは、指定ファイルを読み込んで文字列の置換を行ったあと、その結果を同じファイル名で上書きします。このとき、元のファイルがシンボリックリンクだとすると、シンボリックリンクを削除して、普通のファイルとして同名のファイルを作成します。つまり、「**/etc/sysconfig/selinux**」自体が「SELINUX=disabled」に設定されたファイルとなり、本物の設定ファイル「**/etc/selinux/config**」は元のままになります。

筆者の知人は、一度、データセンタ内の数百台のサーバで上記のコマンドを実行してしまい、冷や汗を書きながら修正したことがあるそうです。その後、sedコマンドのmanページ(英語版)を見て、このようなときは、「--follow-symlinks」オプションが必要だと気づいたそうです。

▼図4　起動サービスの確認

```
# chkconfig --list | grep 3:on
abrt-ccpp       0:off   1:off   2:off   3:on    4:off   5:on    6:off
abrtd           0:off   1:off   2:off   3:on    4:off   5:on    6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
blk-availability        0:off   1:on    2:on    3:on    4:on    5:on    6:off
certmonger      0:off   1:off   2:off   3:on    4:on    5:on    6:off
cpuspeed        0:off   1:on    2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
...(以下省略)...
```

chkconfigコマンドで確認します。この例では、grepコマンドを使って、ランレベル3で起動するサービスだけを表示しています。サービスの起動を有効化／無効化する際は、同じく、chkconfigコマンドを使用します。--levelオプションで特定のランレベルに対してのみ、有効化／無効化を設定することも可能です。

```
# chkconfig <サービス名> on  ←有効化
# chkconfig <サービス名> off ←無効化
```

図4の結果からもわかるように、一般にシステム起動時には、多数のサービスが起動します。これらすべての設定ファイルを紹介するわけにはいきませんが、ネットワークに関連する主要な設定ファイルは、次のPart4でまとめて説明します。ここでは、システム管理のうえで重要な「crondサービス」の設定ファイルを解説しておきます。

これは、各ユーザに対して、指定時刻に指定のコマンドを定期実行する、スケジューリング機能を提供するサービスです。この機能で定期実行するコマンドを「cronジョブ」、各ユーザのジョブ設定ファイルを「crontabファイル」と呼びます。ここでは、個人ユーザ「nakai」でcronジョブを設定するものとします。

▼リスト3　crontabファイルの設定例

```
* * * * * /home/nakai/bin/script01.sh >/dev/null 2>&1    ←毎分実行
0,30 * * * * /home/nakai/bin/script02.sh >/dev/null 2>&1 ←毎時0分と30分に実行
30 8 * * * /home/nakai/bin/script03.sh >/dev/null 2>&1   ←毎日8:30に実行
30 13 * * 0 /home/nakai/bin/script04.sh >/dev/null 2>&1  ←毎週日曜日の13:30に実行
0 9 25 12 * /home/nakai/bin/script05.sh >/dev/null 2>&1  ←12月25日の9:00に実行
0 * 1-3 1 * /home/nakai/bin/script06.sh >/dev/null 2>&1  ←正月三賀日は毎時0分に実行
```

まず、既存で設定されているジョブは、crontabコマンドの-lオプションで確認します。ここでは、まだ設定されたジョブはありません。

```
$ crontab -l
no crontab for nakai
```

crontabファイルを開いて設定する際は、-eオプションを指定します。viエディタが開いて、crontabファイルが編集できるようになります。

```
$ crontab -e
```

リスト3は、典型的な設定例を並べたものです。各行の先頭5つの数字(もしくは「*」)は、実行タイミングを指定するもので、先頭から、「分」「時」「日」「月」「曜日」に対応します。「*」は「任意の」という意味になります。それぞれの部分は、「,」区切りによる複数指定や「1-3」のような範囲指定も可能です。「曜日」の部分は、0～7の数字で指定して、「0」と「7」はどちらも日曜日で、「1～6」が月曜日～土曜日に対応します。

実行タイミングの後ろには、実行するコマンドを記載します。cronジョブとして実行するコマンドは、画面出力を表示する場所がありませんので、通常は、「>/dev/null 2>&1」を付けて画面出力を捨てるようにしておきます。画面出力を捨てなかった場合は、その内容がrootユーザにメールで送信されるようになっています。

設定済みの内容を破棄して、crontabファイルを空にするときは、-rオプションを使用します。

```
$ crontab -r
```

最後に、cronジョブに類似の機能を提供する「anacron」について補足しておきます。cronジョブは、各ユーザが個別にジョブを設定するものですが、anacronは、システム全体の管理作業を定期的に実行するためのしくみで、「日次」「週次」「月次」で指定のコマンドやスクリプトを実行します。それぞれ、「**/etc/cron.daily**」「**/etc/cron.weekly**」「**/etc/cron.monthly**」というディレクトリの中にあるコマンドが上記の周期で実行されます。ディレクトリに配置したコマンドは、rootユーザの権限で実行されます。

cronジョブのように細かな時刻の指定はできませんが、逆に、毎回、実行時刻がランダムに変動するようになっています。日次ジョブの場合は、毎晩、深夜の3時～3時45分のどこかで実行されます。仮想マシン環境では、多数の仮想マシンが同時に日次ジョブを実行すると、物理ホストの負荷が跳ね上がるなどの問題が起きることがあります。これを回避するためのしくみになります。

なお、anacronは、crondサービスから定期的に実行されるようになっていますので、anacronのための独立したサービスはありません。

まとめ

Part3では、Linuxサーバの起動処理の流れを見ながら、関連する設定ファイルの説明を行いました。システム起動処理は、普段はあまり意識しない部分ですが、その背後で起きていることを理解すると、Linuxの「奥深さ」がよくわかります。システム起動処理にかかわる設定は、設定をあやまるとサーバが起動しなくなるなどの問題にもつながりますので、とくに正確な理解が必要です。

次のパートでは、ネットワーク設定にかかわる設定ファイルを解説していきます。SD

Part 4

IPのしくみを振り返りながら学ぶ

ネットワーク設定ファイル徹底攻略

Part4では、いよいよネットワークの設定ファイルを扱います。IPネットワークの基礎をおさらいしながら、Linuxではどの設定ファイルで、どのようなネットワークの定義がなされているのかを見てみましょう。IPアドレスとデフォルトゲートウェイの設定、名前解決のしくみと設定、iptablesによるパケットフィルタリングについて解説します。

ネットワーク設定の心構え

ネットワークの設定ファイルには、多数の設定項目が登場します。これらの意味を理解して、正しく設定するには、当然ながら、TCP/IPを中心とするネットワークの知識が必要です。ネットワークの基礎に不安がある読者は、まずは、『[改訂新版]3分間ネットワーク基礎講座』[注1]などの書籍で基礎固めを行いましょう。

また、ネットワークの設定において大切なのが、「接続先のネットワークに合わせて設定する」という意識です。当たり前のことのようですが、ネットワークにかかわるトラブルでは、「ネットワーク側の設定とサーバ側の設定のミスマッチ」によるものが多数あります。サーバを設定するエンジニアにネットワークの基礎知識がなければ、サーバ側の設定に必要となる、ネットワーク側の設定情報を聞き出すこともできません。ネットワーク設定ファイルの各種項目について、関連するネットワークの基礎知識と併せて確認していきましょう。

▼リスト1　ホスト名とIPアドレスの設定

/etc/sysconfig/network

```
NETWORKING=yes
HOSTNAME=rhel65
```

/etc/sysconfig/network-scripts/ifcfg-eth0
（固定IPアドレス設定）

```
DEVICE=eth0
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
```

/etc/sysconfig/network-scripts/ifcfg-eth0
（DCHPによるIPアドレス設定）

```
DEVICE=eth0
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=dhcp
```

ネットワーク基本設定

ネットワーク設定の基本は、ホスト名とIPアドレスの指定です。**リスト1**が典型的な設定例になります。まず、設定ファイル「**/etc/sysconfig/network**」の「HOSTNAME」にホスト名を指定します。「NETWORKING=yes」は、ネットワーク機能を使用するという意味です。これを「no」にすると、ネットワークの設定はいっさい行われなくなります。

設定ファイル「**/etc/sysconfig/network-scripts/ifcfg-eth0**」は、NIC（ネットワークインターフェース）「eth0」に対するIPアドレスの設定です。複数のNICを持つ環境では、それぞれのNICについて設定ファイル（ifcfg-ethX）を用意します。「DEVICE」の部分にNICのデバイス名が指定されていますので、「ifcfg-eth0」を「ifcfg-eth1」にコピーして利用するような場合は、この部分を変更するのを忘れないようにしてください。

「NM_CONTROLLED=no」は、Network

注1）網野衛二 著『[改訂新版]3分間ネットワーク基礎講座』技術評論社,2010年

▼図1　NICデバイス名とMACアドレスの紐付け

/etc/udev/rules.d/70-persistent-net.rules

MACアドレス

```
# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="52:54:00:a8:ee:3d", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth0" ←デバイス名

# PCI device 0x1af4:0x1000 (virtio-pci)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="52:54:00:5e:5d:ce", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth1"
```

現在のMACアドレスを確認

```
# ifconfig -a | grep HWaddr
eth0      Link encap:Ethernet  HWaddr 52:54:00:A8:EE:3D
eth1      Link encap:Ethernet  HWaddr 52:54:00:5E:5D:CE
```

Managerサービスの管理対象外にする指定です。NetworkManagerサービスは、ノートPCなどネットワーク構成が頻繁に変化する環境で、ネットワーク設定を自動変更する機能を提供します。しかしながら、一般的なサーバ環境では、ネットワーク設定の自動変更は不要ですので、NetworkManagerサービスの管理対象外とします。インストール時のパッケージの選択によっては、最初からNetworkManagerサービスが導入されていない場合もありますが、導入されている場合は、次のコマンドでサービス自体を無効化しておくと良いでしょう。

```
# chkconfig NetworkManager off
# service NetworkManager stop
```

NetworkManagerサービスの管理対象外のNICは、サーバ起動時にnetworkサービスが起動したタイミングで有効化されて、IPアドレスが割り当てられます。ただし、そのためには「ONBOOT=yes」の指定が必要です。これを「no」にすると、networkサービスからの有効化が行われません。次のように、手動で有効化する必要があります。

```
# ifup eth0
```

具体的なIPアドレスの割り当て部分については、このあとでまとめて説明します。なお、インストール直後の設定ファイル「ifcfg-eth0」を見ると、**リスト1**に記載のない項目として、次のようなものがあります。

```
TYPE=Ethernet
UUID=cc5702fa-3021-4caf-9c93-06eb966a117f
HWADDR=52:54:00:A8:EE:3D
```

「TYPE=Ethernet」はEthernet接続を使用するという意味で、その他の選択肢には、「Wireless(ワイヤレス接続)」「Modem(モデム接続)」などがあります。Ethernetを使用する場合は、とくに指定しなくても問題ありません。「UUID」は前述のNetworkManagerサービスが使用する項目ですので、これも必要ありません。

そして、「HWADDR」は、該当NICのMACアドレスを記載します。以前のバージョンのRHELでは、複数のNICを持つサーバにおいて、この設定によって、特定のMACアドレスのNICに対して、特定のデバイス名を割り当てることができました。しかしながら、RHEL6では、この機能はudevに移行されており、ここでMACアドレスを指定する意味はありません。むしろ、udevで割り当てたものと異なるMACアドレスを指定すると、IPアドレスが割り当てられなくなります。

udevによる、NICデバイス名とMACアドレスの紐付(ひもづ)けは、設定ファイル「**/etc/udev/rules.d/70-persistent-net.rules**」で行います(**図1**)。NICを追加するなどした場合は、サーバ起動時に自動的にエントリが追加されます。既存の内容をあえて変更する場合は、このファイルを修正して、サーバを再起動します。

NICデバイス名とMACアドレスの実際の対応関係は、**図1**のifconfigコマンドで確認します。

IPネットワークの基礎を学ぶ

続いて、**リスト1**の中で、IPアドレスの設定にかかわる項目を見ていきます。これは、固定的にIPアドレスを指定する場合と、DHCPによる自動割り当てを行う場合に分かれます。固定的に割り当てる場合は、「BOOTPROTO=static」として、**表1**の3つの項目を指定します。一方、DHCPを使用する場合は、「BOOTPROTO=dhcp」と指定するだけです。**表1**の項目は、DHCPサーバから適切な値を取得して、自動的に設定されます。

それでは、皆さんは、**表1**の項目の意味を説明できるでしょうか？　これを理解するには、「サブネット」と「ネットワークアドレス」の概念を理解する必要があります。

一般に、IPネットワークの世界は、L2スイッチで接続された機器の「島」をルータが相互接続する形になります。最近は、専用のルータ機器ではなく、L3スイッチ内部の仮想ルータ機能を利用することもあります(**図2**)。このような1つ1つの「島」を「サブネット」と呼びます。同じサブネットの機器は、L2スイッチを経由して直接にネットワークパケットを交換することができます。

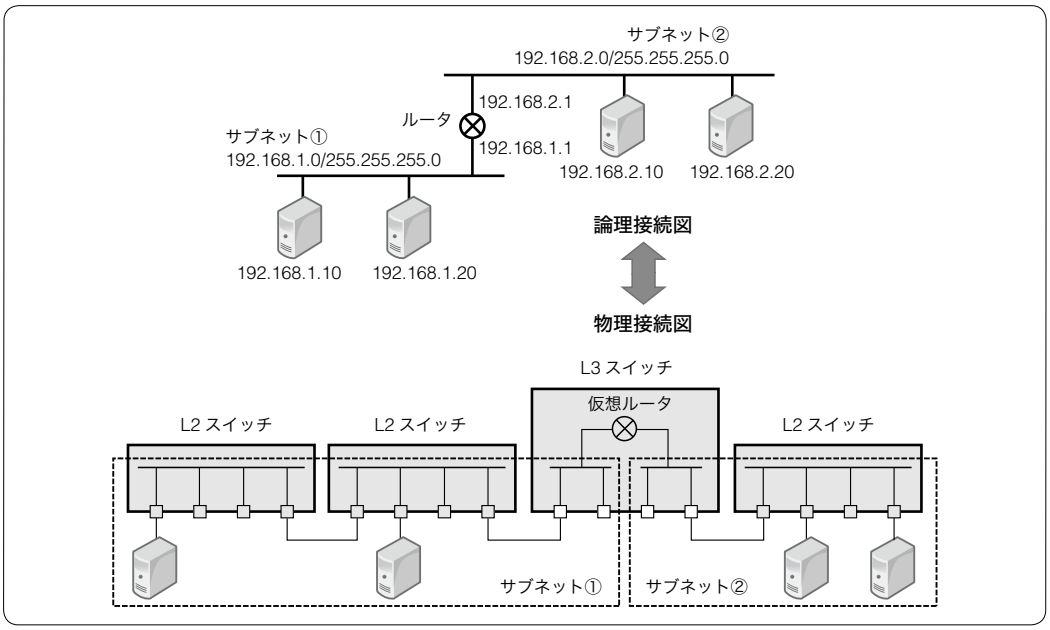
一方、異なるサブネットの機器と通信する際は、間にあるルータがパケットを中継します。つまり、サーバは、ルータのIPアドレスに向けてパケットを送信する必要があります。このときのルータのIPアドレスが「デフォルトゲートウェイ」です。**図2**の例では、サブネット①のデフォルトゲートウェイは「192.168.1.1」で、サブネット②は「192.168.2.1」になります。

また、それぞれのサブネットでは、そこに接続する機器が使用するIPアドレスの範囲が、ネットワークアドレスによって規定されます。ネットワークアドレスは、「192.168.1.0/255.255.255.0」

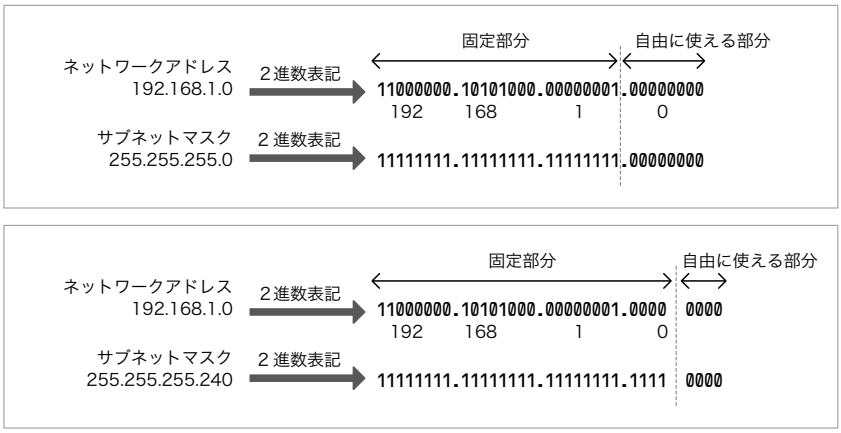
▼表1　IPアドレス設定の基本項目

項目	説明
IPADDR	IPアドレス
NETMASK	サブネットマスク
GATEWAY	デフォルトゲートウェイのIPアドレス

▼図2　IPネットワークの基本構成



▼図3　ネットワークアドレスの計算例



のようにサブネットマスク(この例では「255.255.255.0」)とセットで定義されます。ネットワークアドレスとサブネットマスクを2進数表記すると、**図3**のようにサブネットマスクは先頭から1が連続します。ネットワークアドレスのうち、サブネットマスクが1の部分は固定されており、サブネットマスクが0の後半部分が自由に使える範囲です。

図3の上にある「192.168.1.0/255.255.255.0」では、「192.168.1.0～192.168.1.255」がこのサブネットで使用できるIPアドレスの範囲になります(厳密には、最初と最後にあたる「192.168.1.0」と「192.168.1.255」は特別な意味を持つので、機器に割り当てることはできません)。一方、**図3**の下にある「192.168.1.0/255.255.255.240」では、「192.168.1.0～192.168.1.15」が使用できる範囲です。同じネットワークアドレスでも、サブネットマスクが変わると使用できる範囲が変わるので注意が必要です。

Linuxカーネルがネットワークパケットを送信する際は、宛先アドレスが自身と同じサブネットのネットワークアドレスに収まっているかどうかで、送信先を変更します。同じサブネットであれば、該当の機器に向けて直接に送信しますが、そうでない場合は、デフォルトゲートウェイに向けて送信します。このような振り分けを行うために、サブネットマスクの値が必要になります。

ちなみに、**表1**を見ると、サブネットマスクの指定はありますが、ネットワークアドレス自体の設定はありません。これは、自身のIPアドレスとサブネットマスクがわかれば、対応するネットワークアドレスは、計算で求めることができるからです(自身のIPアドレスを2進数表記して、サブネットマスクが0の部分の値を0にすれば、ネットワークアドレスになりますよね？)。

少し長くなりましたが、これで、**表1**の「3点セット」の必要性が理解できました。一般的な企業ネットワークでは、これらの設定値は、事前にネットワーク担当者から割り当てられることがほとんどですが、サーバ仮想化環境などでは、サーバ管理者自身がネットワークの構成を行う場面も増えてきました。サブネットマスクの値などにもしっかりと注意を払いながら、設定するようにしてください。

名前解決にかかわる設定ファイル

続いては、「名前解決」に関連する設定ファイルです。名前解決とは、IPアドレスの数値と「gihyo.co.jp」などのFQDN(Fully Qualified Domain Name)を相互変換する機能です。一般

▼リスト2　/etc/hostsの設定例

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.10 rhel65 rhel65.example.com
```

には、「ホスト名とIPアドレスの相互変換」と説明されることもありますが、これは厳密には正しくありません。ホスト名は、1台のサーバに1つだけ与えられるものですが、複数のNICを持つサーバでは、それぞれのNICについて、対応する名前(インターフェース名)が割り当てられます。NICが1つの環境では、慣習的に、インターフェース名とホスト名に同じものを使用していると考えてください。

サーバ上で、ネットワーク通信の宛先にFQDNを指定した場合は、最初に設定ファイル「**/etc/hosts**」を参照して対応するIPアドレスを検索します。ここで見つからなかった場合は、設定ファイル「**/etc/resolv.conf**」で指定されたDNSサーバに問い合わせにいきます。通常は、DNSサーバに登録されていない、プライベートなネットワーク上のサーバのみを「**/etc/hosts**」に登録しておきます。

「**/etc/hosts**」の書式は、**リスト2**のとおりで、IPアドレスの後ろにスペース区切りで、任意の数だけ名前を指定することができます。最初の2行は、デフォルトで用意されているもので、「localhost」という名前を自分自身を示すループバックアドレスに紐付けています(2行目はIPv6のループバックアドレス)。3行目の例にあるように、FQDN(rhel65.example.com)の先頭部分(rhel65)だけを取り出した短縮名も記載することができます。サーバ上で、短縮名を宛先に使用した場合、「**/etc/hosts**」に対応するエントリがあれば、そのIPアドレスに変換されます。

一方、「**/etc/hosts**」にエントリがなく、DNSサーバに問い合わせる場合は、少し動きが変わります。DNSサーバでは、短縮名での問い合わせは許可されませんので、何らかの方法でFQDNに変換してから問い合わせる必要があります。このときに使用されるのが、

▼リスト3　/etc/resolv.confの設定例

```
nameserver 8.8.8.8
nameserver 8.8.4.4
search example.com gihyo.co.jp
```

「**/etc/resolv.conf**」の「search」エントリです。

リスト3は、「**/etc/resolv.conf**」の設定例で、「nameserver」にはDNSサーバのIPアドレスを指定します。複数のエントリを用意すると、最初のDNSサーバが障害で停止している際などは、2つめ以降のDNSサーバに順番に問い合わせるようになります。そして、「search」の部分に短縮名を補完するドメイン名を指定します。この例の場合、短縮名「rhel65」に対しては、最初に「rhel65.example.com」でDNSサーバに問い合わせます。該当のFQDNが存在しないという返答の場合は、続いて、「rhel65.gihyo.co.jp」で問い合わせます。

「search」の代わりに、「domain」という設定を用いる場合もありますが、これは、ドメイン名が1つしか指定できないという制限を除いて、「search」と同じ役割を果たします。

なお、あまり使用することはありませんが、設定ファイル「**/etc/nsswitch.conf**」を用いると名前解決の際の検索順序を変更することができます。デフォルトでは、「hosts: files dns」という設定が記載されており、最初に「**/etc/hosts**」を検索して、次にDNSサーバに問い合わせるという意味になります。たとえば、これを「hosts: dns」に変更すると「**/etc/hosts**」を使用せずに、いきなりDNSサーバに問い合わせるようになります。

iptablesの設定を理解する

それでは続いて、ネットワーク設定の大物(?)、

iptablesの設定ファイルに移ります。iptablesの役割は、大きくは、パケットフィルタリングとNAT(Network Address Translation)に分かれます。ここでは、iptablesの基礎となるパケットフィルタリングを解説します。これは、サーバに出入りするネットワークパケットに対して、指定の条件を満たすものだけを通過させるファイアウォール機能を提供するものです。

iptablesの機能を有効にするには、iptablesサービスを起動する必要があります。デフォルトでサーバ起動時に自動起動するように設定されているはずですが、必要な際は、次のコマンドで自動起動を設定しておいてください。

```
# chkconfig iptables on
```

iptablesを設定する際は、設定ファイル「**/etc/sysconfig/iptables**」を編集します。次のコマンドでiptablesサービスを再起動すると、設定内容が反映されます。

```
# service iptables restart
```

図4は、RHEL6におけるデフォルトの設定内容です。1行目の「*filter」は、パケットフィルタリングに関する設定であることを示します。その他の項目を説明する前に、**図5**の3種類のチェーンについて説明しておきます。

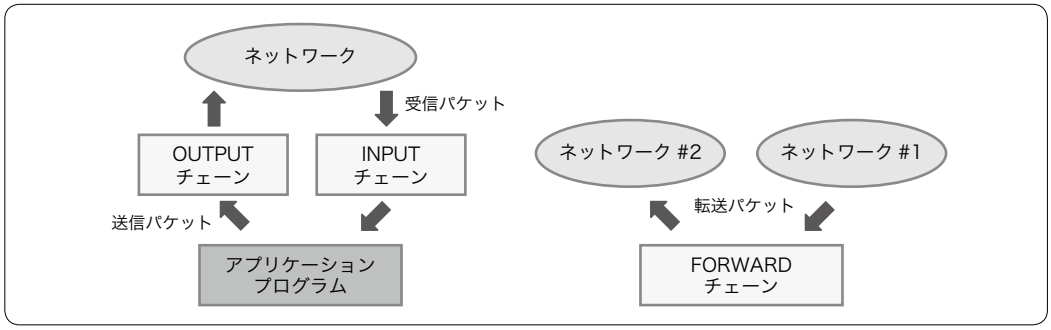
Linuxサーバに出入りするパケットは、「受信パケット」「送信パケット」「転送パケット」の3種類に分かれます。受信パケット／送信パケットは、その名のとおり、Linux上のアプリケーションが外部から受信／外部に送信するパケットです。転送パケットは、複数のNICを持つLinuxサーバをルータとして使用する際に発生します。一方のネットワークから受信して、そのまま他方のネットワークに送信するパケットがこれにあたります。

▼図4　/etc/sysconfig/iptablesの設定例（コメント行は省略）

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]   ←デフォルトポリシー
:OUTPUT ACCEPT [0:0]
①  -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
②  -A INPUT -p icmp -j ACCEPT
③  -A INPUT -i lo -j ACCEPT
④  -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
⑤  -A INPUT -j REJECT --reject-with icmp-host-prohibited
⑥  -A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

```
-A <チェーン> <パケットマッチングルール> -j <アクション>
```

▼図5　iptablesの3種類のチェーン



これら3種類のパケットは、Linuxカーネル内部で、それぞれに対応する「チェーン」を通過するようになっており、各チェーンに対して通過を許可する条件を設定します。**図4**の①～⑤は、「INPUTチェーン」の設定で、このサーバが受信するパケットに対する条件です。最後の⑥は、「FORWARDチェーン」の設定で、転送パケットに対する条件になります。

一般的な書式は、**図4**に記載のとおりです。各チェーンを通過するパケットについて、上から順に<パケットマッチングルール>を評価していき、マッチした条件の<アクション>が適用されます。アクションの種類は、**表2**のとおりです。基本的には最初にマッチしたルールのアクションが適用されて、その後のルールは無視されますが、「LOG」の場合は例外で、パケットの情報をシステムログに記録した後、さらに次のルールへと判定が移ります。

どのルールにもマッチしなかった場合は、**図4**の上部にある「デフォルトポリシー」が適用されます[注2]。デフォルトポリシーに設定できるのは、「ACCEPT」か「DROP」のどちらかですが、この例ではすべてのチェーンに対して「ACCEPT」が指定されています。つまり、ルールにマッチしなかったパケットは、通過が許可されます。

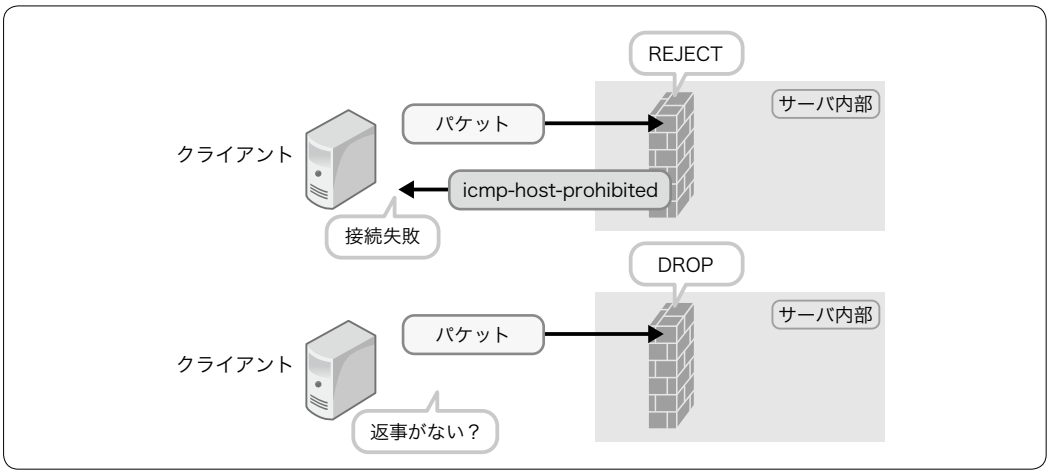
――と言いたいのですが、実際には、少し事情が異なります。⑤⑥の部分はマッチングルールの指定がありませんので、すべてのパケットがマッチして、「REJECT」が適用されます。つまり、受信パケットについては、①～④のどれかにマッチして、「ACCEPT」が適用されたものだけが通過して、それ以外は受信を拒否します。転送パケットについては、必ず⑥にマッチするので、このサーバでは、パケットの転送はいっさい許可されないことになります。デフォルトポリシーに「REJECT」が指定できれば、このような設定は不要なのですが、前述のようにデフォルトポリシーに「REJECT」は指定できないので、このような設定になっているものと思われます。

なお、「REJECT」と「DROP」の違いは、**図6**のとおりです。「DROP」の場合は、受信パケットをだまって破棄するだけですが、「REJECT」

▼表2　iptalesのアクション

アクション	説明
ACCEPT	通過を許可する
REJECT	通過を拒否する（通知あり）
DROP	通過を拒否する（通知なし）
LOG	パケットの情報を記録する

▼図6　「REJECT」と「DORP」の違い



注2） デフォルトポリシーの末尾の「[0:0]」は、iptablesがパケットの統計情報を記録するためのフィールドです。この部分は、とくに変更する必要はありません。

の場合は、受信を拒否したことをクライアントに通知します。**図4**の⑤⑥にある「--reject-with」は受信拒否を通知するパケットの種類を指定しています。つまり、「REJECT」の場合、クライアント側では即座に接続エラーが発生しますが、「DROP」の場合は、サーバからの応答をしばらく待った後にタイムアウトでエラーになります。言い換えると、「REJECT」の場合、クライアントからは、少なくともそこにサーバがあることはわかります。クライアントに対してサーバの存在そのものを隠したい場合は、「DROP」を使用する必要があるというわけです。

パケットマッチングルールの詳細

それでは、**図4**の①～④のパケットマッチングルールについて説明していきます。まず、②は、プロトコル(-pオプション)がICMPのパケットにマッチします。つまり、このサーバに対しては、pingコマンドなど、ICMPを利用した通信が許可されます。③は、受信インターフェース(-iオプション)がループバックの場合にマッチします。つまり、外部からの接続ではなく、同じサーバ上でループバックアドレス(127.0.0.1)に接続する際は、これが許可されます。

①④には「-m state」というオプションがありますが、これは、「コネクショントラッキング」と呼ばれる機能に関係します。SSHでサーバにログインするような場合、1回のログインに伴って継続的にパケットのやりとりが発生しますが、1番目のパケットとその後に継続するパケットを区別する機能になります。④の「--state NEW」は、1番目のパケットを表します。SSH接続に使用するTCP22番ポート宛のパケット「-m tcp -p tcp --dport 22」で、1番目のパケットは、この条件にマッチして受信が許可されます。

それでは、2番目以降のパケットはどうなるのでしょうか？　こちらは、①の「--state ESTABLISHED,RELATED」にマッチして、受信が許可されます。これは、2番目以降のパケット(ESTABLISHED)、もしくは、既存の通信に関連した通信のパケット(RELATED)という意味です(RELATEDの詳細は後述)。結果として、SSH接続に伴うパケットはすべて受信が許可されて、このサーバにはSSHでログインできることになります。まわりくどい設定のようですが、iptablesのルールが大量にある場合、2番目以降のパケットを最初のルールですぐにマッチすることで、残りのルール判定をスキップして処理性能を向上させる効果があります。

また、①のルールは、このサーバから外部への通信を許可する意味もあります。OUTPUTチェーンの設定がありませんので、サーバから外部に送信するパケットは、当然ながら送信を許可されますが、通信先のサーバから返ってくる応答パケットは、INPUTチェーンで受信を許可する必要があります。今の場合、応答パケットは、最初に送信したパケットに対する2番目のパケットとして、①のルールで受信が許可されることになります。

そして最後に、①の「RELATED」の説明です。たとえば、FTP通信では、コマンド操作を行う「コントロールセッション」の通信と、データ転送を行う「データセッション」の2種類の通信が発生します。最初にコントロールセッションのパケットが通過すると、その後で発生するデータセッションのパケットは、「関連するパケット(RELATED)」として認識させることができます。これにより、コントロールセッション(TCP21番ポート)の通信を許可しておけば、データセッションの通信は①のルールで自動的に許可されることになります。FTP通信のデータセッションは、宛先ポートがランダムに決定されるために、明示的に宛先ポートを指定して通信を許可することができないという問題があるのですが、この問題を回避することが可能になります。

Column

FTPとファイアウォールの相性の悪さ

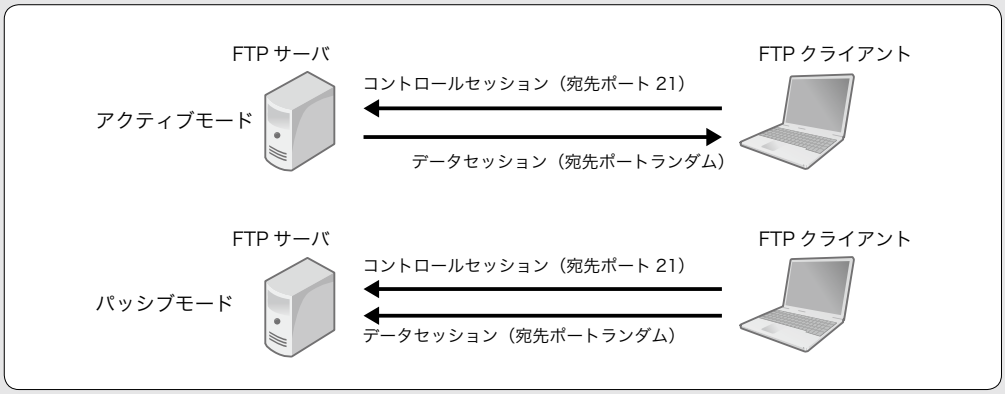
本文で例に出したFTPは、通信内容が暗号化されないなど、セキュリティ上の懸念点があるため、現在では、よりセキュアなSFTPを利用することがほとんどです。また、FTPが多用されていた過去を振り返ると、FTPとファイアウォールとの相性の悪さはよく問題になりました。

FTPでは、コントロールセッションとデータセッションの2種類の通信が発生すると説明しましたが、初期のFTPでは、**図7**の「アクティブモード」がデフォルトでした。これは、データセッションについて、サーバからクライアントに接続を行う形になります。この場合、サーバとクライアントの間に、サーバからクライアント方向の接続を禁止するファイアウォールがあれば、その時点でFTPは使えなくなります。

その後、この問題を解決するために、データセッションもクライアントからサーバに接続する「パッシブモード」が追加されました。ただし、この場合も、データセッションでは、宛先のTCPポート番号がランダムに決定されるという仕様になっており、**図4**の④のように明示的にポート番号を指定して接続を許可することができません。そこで、これを解決するために、ヘルパーモジュール「nf_conntrack_ftp」が開発されたというわけです。

iptablesの設定を適切に行うには、アプリケーションがどのような通信を行うのか、その部分の知識も必要となります。iptablesを学ぶことは、Linux、ネットワーク、アプリケーション、あらゆる知識を身につけるチャンスとも言えるでしょう。iptablesの完全理解を目指す方には、筆者の著書[注3]をお勧めします。

▼図7 FTPのコントロールセッションとデータセッション



なお、どのような通信を「関連する（RELATED）」と判断するかは、ヘルパーモジュールによって管理します。ヘルパーモジュールは、設定ファイル「**/etc/sysconfig/iptables-config**」の「IPTABLES_MODULES=」に指定します。先のFTP通信の場合は、「nf_conntrack_ftp」というヘルパーモジュールを指定する必要があります。

まとめ

Part4では、IPネットワークの基礎知識と併せて、ネットワーク関連の設定ファイルを説明しました。サーバ仮想化環境、クラウド環境では、ネットワークの知識は必須になります。IPアドレスの設定やファイアウォールの設定など、これまで「なんとなく」設定していたかもしれませんが、その背後のしくみと併せて理解しておくと、問題発生時にも「何をすれば良いのか」が自然にわかるようになります。

次のパートでは、Webサーバやデータベースサーバなど、典型的なアプリケーションの設定ファイルを説明します。さらに、ファイルのアクセス権についても学びます。SD

注3） 中井悦司 著『プロのためのLinuxシステム・ネットワーク管理技術』技術評論社,2011年

Part 5

本番環境を想定して

アプリケーション設定ファイルの例

Part5では、Linux上で動かすアプリケーションの設定ファイルについて、「Apache HTTPサーバ」と「Samba」を例として見ていきます。これらアプリケーションのファイル配置や設定手順について説明した後は、ファイルへのアクセス権について解説していきます。Part2の内容を思い出しながら見ていきましょう。

設定ファイルの配置パターンを知る

Part5では、各種アプリケーションの設定方法を見ていきます。Linux本体の設定ファイルは、ディレクトリ「**/etc**」の下に配置するのが標準ですが、アプリケーションの設定ファイルについては、とくに明確なルールはありません。ただし、RHEL6の場合、RPMパッケージで提供される標準的なアプリケーションでは、「**/etc**」の下にアプリケーション名のディレクトリがあり、その中に設定ファイルが保存されることがほとんどです。

ここでは、代表的なアプリケーションとして、「Apache HTTPサーバ」と「Samba」を取り上げて、これらの設定ファイルの配置パターン、そして、典型的な設定手順／設定例を紹介します。iptablesやSELinuxなど、これまでのパートで説明した知識も必要となります。

また、アプリケーションからアクセスするファイルのアクセス権の理解も大切です。「Part2 ユーザ管理と設定ファイル」の補足として、ファイルのアクセス権についても説明しておきます。

Apache HTTPサーバの設定ファイル

RHEL6でApache HTTPサーバ(以降では「Apache」と表記)を使用する場合は、RPMパッケージ「httpd」を導入します。yum(ヤム)コマンドを利用すると、依存パッケージをまとめて導入できるので便利です。

```
# yum install httpd
```

このとき、デフォルトで導入される設定ファイルは、次の3つです。

(1) **/etc/sysconfig/httpd**
(2) **/etc/httpd/conf/httpd.conf**
(3) **/etc/httpd/conf.d/welcome.conf**

(1)は、httpdサービスからデーモンプロセスを起動する際の起動オプションや環境変数の設定などを行います。Apacheが提供するWebサーバ機能についての設定ファイルではありませんので、通常は変更する必要はないでしょう。Apacheが標準提供するWebサーバ機能の各種設定は、(2)の設定ファイルで行います。

ただし、すべての設定を1つのファイルに書き込むのではなく、必要に応じて、設定ファイルを複数に分割できます。追加の設定ファイルは、ディレクトリ「**/etc/httpd/conf.d**」の下に任意のファイル名で配置します。これらの設定ファイルは、ファイル名のアルファベット順に読み込まれていきます。

Apacheは、さまざまな「モジュール」を読み込むことで機能拡張を利用できます。読み込むモジュールの指定は、(2)の設定ファイルで行いますが、個々のモジュールに対する設定は、別の設定ファイルに分けて、先のディレクトリに配置することが多いようです。デフォルトで用意される(3)は、モジュールの設定というわけではありませんが、**図1**のテストページを表示する設定を追加するものになっています。

Apacheの設定例

RHEL6のRPMパッケージで提供される

Apacheでは、デフォルトの設定ファイルで基本的な設定がなされており、ディレクトリ「**/var/www/html**」の下に公開したいコンテンツ(HTMLファイル)を配置するだけで、すぐに利用できます。

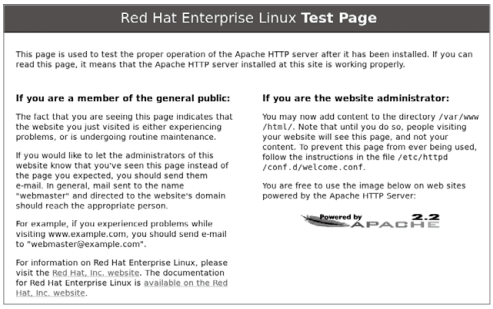
ただし、iptablesによるファイアウォールを使用している場合は、Part4で説明した設定ファイルを利用して、HTTP/HTTPS接続に使用するポート(TCP80番ポート/443番ポート)へのアクセスを許可する必要があります。**Part4の図4**における④と同じ形式で、「--dport 80」と「--dport 443」の行を追加するのが簡単ですが、そのほかには、**リスト1**のような指定もできます。「-m multiport」を使用すると、「--dports」オプションで複数のポート番号を指定できるようになります。

リスト1の変更を行ったら、次のコマンドでhttpdサービスの自動起動を有効化したあと、httpdサービスの起動とiptablesサービスの再起動を行います。

```
# chkconfig httpd on
# service httpd start
# service iptables restart
```

このあと、Webブラウザから、このサーバのIPアドレスにアクセスすると、**図1**の画面

▼図1　Apacheのテストページ



▼リスト1　Webサーバへの接続許可設定

/etc/sysconfig/iptables

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m multiport -p tcp --dports 80,443 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

この行を追加

Column 『rpm/yumコマンドの便利機能を活用していますか？』

本文で、httpdパッケージがデフォルトで提供する設定ファイルを紹介しましたが、一般に、導入済みのRPMパッケージが提供するファイル一式は、次のコマンドで確認できます。

```
# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
...(以下省略)...
```

RPMパッケージを導入する前であれば、-pオプションでパッケージファイルを指定します。逆に、導入済みのファイルがどのRPMパッケージで提供されたものか調べることもできます。

```
# rpm -qf /etc/httpd/conf/httpd.conf
httpd-2.2.15-29.el6_4.x86_64
```

YUMリポジトリからパッケージをインストールする前に、パッケージファイルだけを取得したい場合は、yumdownloaderコマンドを使用します。

```
# yumdownloader httpd
```

このように、rpm/yumコマンドの機能は、パッケージを導入するだけにとどまりません。便利な使い方がさまざまありますので、筆者のブログ記事[注1]も参考にしてください。

注1) yumによるRHELの保守的パッチ適用方法　http://d.hatena.ne.jp/enakai00/20130208/1360285927

▼リスト2　/etc/httpd/conf/httpd.confの変更個所（コメント行は省略）

```
<IfModule mod_userdir.c>
    UserDir disabled ←――――この行をコメントアウト（行頭に＃を追加）
    #UserDir public_html ←――― この行のコメントアウトを解除（行頭の＃を削除）
</IfModule>
```

▼図2　diffコマンドによる修正個所の確認（コメント行は省略）

```
# diff -u /root/httpd.conf.orig httpd.conf
--- /root/httpd.conf.orig      2014-03-11 17:59:53.869000542 +0900
+++ httpd.conf 2014-03-11 18:00:09.481004706 +0900
@@ -363,14 +363,14 @@ ←――――― 363行目から14行分を表示

-    UserDir disabled ←――――― 削除した行
+    #UserDir disabled ←――――― 追加した行

-    #UserDir public_html ←
+    UserDir public_html ←

 </IfModule>
```

が表示されます。

これだけで終わるとつまらないので、ここではさらに、Linux上の個人ユーザが自身のホームページを公開する方法を紹介します。各ユーザは、ホームディレクトリの下にディレクトリ「public_html」を作成して、好みのHTMLファイルを配置します。すると、「http://<サーバのIPアドレス>/~<ユーザ名>」というURLでその内容が公開される形になります。

これにはまず、Apacheの設定ファイル「**/etc/httpd/conf/httpd.conf**」で個人ユーザのHTMLファイル公開を許可します。具体的には、**リスト2**に示した個所を変更しますが、ここでは、変更前のファイルをバックアップしてから編集してみましょう。

```
# cd /etc/httpd/conf
# cp httpd.conf /root/httpd.conf.orig
# vi httpd.conf ←リスト2の修正を実施
```

このあと、Part1で学んだdiffコマンドで、修正個所を確認すると、**図2**のようになります。出力の見方を図にコメントしてありますが、意図通りの修正ができているか、ひと目で確認できます。このあとは、httpdサービスを再起動して、設定変更を反映しておきます。

```
# service httpd restart
```

続いて、Part3で触れた、SELinuxの設定変更を行います。SELinuxは、アプリケーションの特定の動作を禁止／許可するためのパラメータ（Booleanパラメータ）を持っており、ここでは「httpd_enables_homedirs」パラメータで、httpdプロセスが個人ユーザのホームディレクトリにアクセスすることを許可します。現在の設定値は、次のコマンドで確認します。

```
# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

現在は無効化（禁止）されているので、次のコマンドで有効化（許可）します。このコマンドは、実行が完了するまで少し時間がかかります。

```
# setsebool -P httpd_enable_homedirs on
```

これで、Apacheの準備は完了です。この後は、それぞれの個人ユーザが次の手順で、好みのHTMLファイルを作成します。

```
$ chmod og+rx ~
$ mkdir ~/public_html
$ vi ~/public_html/index.html ←HTMLファイルを作成
```

「~」は作業中のユーザのホームディレクトリを表す記号です。最初のchmod（CHange MODe）コマンドでは、ホームディレクトリに対して、httpdプロセスからのアクセスを許可しています。先ほどは、SELinuxの機能によるアクセ

ス禁止を解除しましたが、こちらは、通常のLinuxとしてのファイルアクセス権の変更です。ファイルアクセス権の詳細は、この次に、Sambaの設定ファイルと併せて解説します。

ユーザ「nakai」でHTMLファイルを作成した場合、Webブラウザから「http://<サーバのIPアドレス>/~nakai」にアクセスすると、その内容が表示されます。

Sambaの設定ファイルと設定手順の例

続いては、Sambaの設定ファイルです。ご存じのように、Sambaは、Windows PCに対して、Linux上のディレクトリを共有フォルダとして公開する機能を提供します。RHEL6でSambaを利用する際は、「samba」のRPMパッケージを導入します[注2]。

```
# yum install samba
```

Sambaの主要な設定ファイルは、次の2つです。

(1) /etc/sysconfig/samba
(2) /etc/samba/smb.conf

(1)は、先のApacheと同じく、smbサービスで、デーモンプロセスを起動する際の起動オプションなどを指定します。通常は、変更する必要はありません。(2)のほうが、Sambaで公開する共有フォルダなど、利用目的に合わせて設定するものになります。基本的には、このファイル1つに、あらゆる設定をまとめて記載します。

Apacheと同ように、RPMパッケージで提供されるデフォルトの設定ファイルで基本的な設定がなされており、各ユーザのホームディレクトリがそのまま共有フォルダとして公開できるようになっています。必要な追加手順は次のとおりです。

まず、iptablesの設定で、TCP445番ポートへのアクセスを許可します。「**/etc/sysconfig/iptables**」に**リスト3**の行を追加したら、smbサービスの起動と、iptablesサービスの再起動を行います。

```
# chkconfig smb on
# service smb start
# service iptables restart
```

次に共有フォルダを公開するための専用ユーザ「winuser」を作成します。

```
# useradd -mk /dev/null -s /sbin/nologin winuser
```

このユーザは、共有フォルダとして使用するホームディレクトリを提供するためのもので、Linuxにログインする目的では使用しません。そこで、ログインシェル(-sオプション)には、Part2で紹介した、すぐにログアウトするプログラムを指定しています。また、「-mk /dev/null」は、ディレクトリ「**/etc/skel**」の下にある「.bashrc」などのカスタマイズファイルをホームディレクトリにコピーしないためのオプションです。

また、Windows PCから共有フォルダを接続する際は、Samba独自のパスワードを使用します。これは、次のコマンドで設定します。

```
# smbpasswd -a winuser
New SMB password: ←任意のパスワードを入力
Retype new SMB password: ←パスワードを再入力
Added user winuser.
```

▼リスト3　Sambaサーバへの接続許可設定
/etc/sysconfig/iptables (追加部分のみ抜粋)

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m multiport -p tcp --dports 80,443 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 445 -j ACCEPT
```

この行を追加

注2) Sambaのデフォルトの設定ファイルについては、「samba」パッケージの依存パッケージとなる「samba-common」パッケージで提供されます。

そして最後に、SELinuxのBooleanパラメータで、Sambaのデーモンプロセスに対して、個人ユーザのホームディレクトリへのアクセスを許可します。

```
# setsebool -P samba_enable_home_dirs on
```

これですべての準備ができました。Windows PCから共有フォルダ「¥¥<サーバのIPアドレス>¥winuser」に接続して利用できます。

アプリケーションとファイルアクセス権

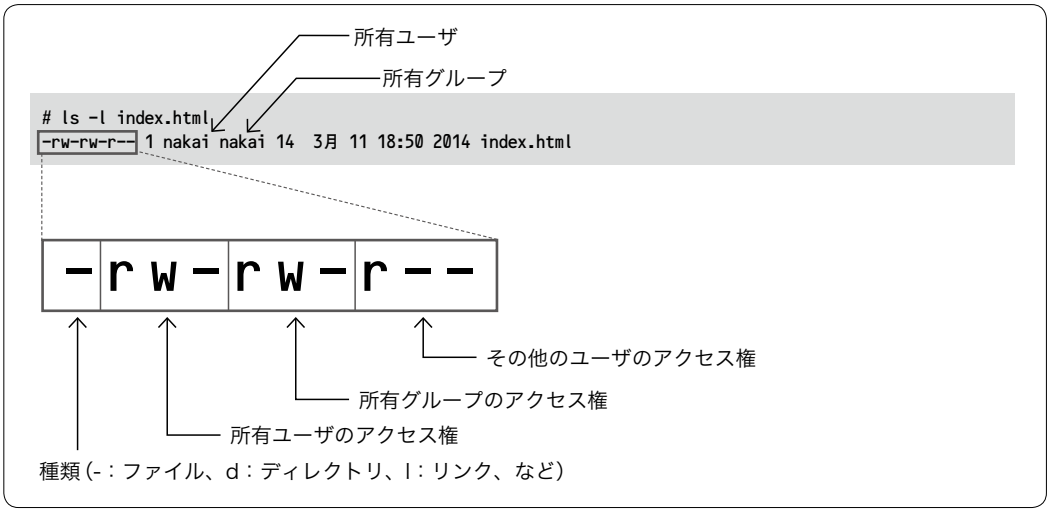
ここまで、少し駆け足で、設定手順だけを紹介しました。ここで、少し落ち着いて、ファイルのアクセス権について考えます。Linux上のファイルやディレクトリは、それを所有するユーザ／グループが決まっており、所有ユーザ、所有グループ、その他のユーザのそれぞれに対するアクセス権が設定されます。

設定されているアクセス権は、**図3**のようにlsコマンドで確認ができます。それぞれの記号の意味は、**表1**、**2**を参考にしてください。ファイルとディレクトリで、記号の意味が少し異なるので注意が必要です。

図3は、先にApacheで公開したHTMLファイルのアクセス権の例ですが、そのほかのユーザからの参照が許可されているところがポイントになります。Apacheのデーモンプロセスは、「apache」というユーザで起動しているので、このユーザがアクセスできないファイルは、Webサーバからは公開できません。そのために、所有ユーザ「nakai」のファイルに対して、その他のユーザからのアクセス許可が必要というわけです。また、このファイルのフルパスに含まれる一連のディレクトリについて、**表2**における「rx」（参照可能かつ、チェンジディレクトリ可能）のアクセス権も必要です。先の手順でホームディレクトリ「**/home/nakai**」に対して、chmodコマンドを実行したのは、このアクセス権を設定するためです。

chmodコマンドの使用方法は、「ugo」「+-」「rwx」の3つ組表現で覚えるのが簡単です。次は、

▼図3　アクセス権の見方



▼表1　ファイルのアクセス権

記号	ファイルに対する設定
r	ファイルの読み込み可能
w	ファイルへの書き込み、削除可能
x	ファイルを実行可能

▼表2　ディレクトリのアクセス権

記号	ディレクトリに対する設定
r	ディレクトリ内のファイル一覧を参照可能
w	ディレクトリ内のファイルの削除、リネーム、新規作成が可能
x	アクセス（チェンジディレクトリ）可能

file01に「u(User：所有ユーザ)」の「x(実行権)」を「+(追加)」します。

```
# chmod u+x file01
```

同じく、次はfile01の「g(Group：所有グループ)」と「o(Other：その他のユーザ)」の「r(読み取り権)」と「w(書き込み権)」を「-(削除)」します。

```
# chmod go-rw file01
```

既存ファイル／ディレクトリの所有ユーザ／所有グループを変更する際は、chown(CHange OWNer)コマンドを使用します。次のように所有ユーザと所有グループを「.」区切りで指定します。どちらか一方だけを指定してもかまいません。

```
# chown nakai.nakai file01
```

それでは、もう一方のSmabaのデーモンプロセスは、どのようなユーザ権限でファイルにアクセスするのでしょうか？ これは、**図4**のようになります。Windows PCから「winuser」で認証して共有フォルダを接続すると、Sambaサーバ側では、Linuxとしてのユーザ「winuser」でディレクトリ上にファイルを読み書きします。したがって、winuser自身のホームディレクトリであれば、アクセス権の変更なく、自由に読み書きができることになります。

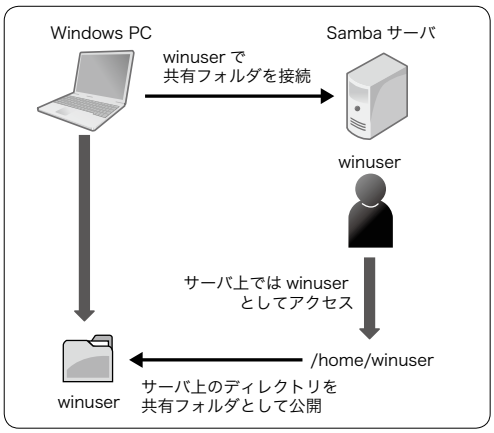
図5の例では、Windows PCの代わりに、Sambaサーバ自身から、smbclientコマンドで共有フォルダに接続してファイルを保存しています。「winuser」のホームディレクトリに保存されたファイルを確認すると、所有者は確かに「winuser」になっています。

まとめ

Part5では、最後のまとめとして、アプリケーションの設定ファイルと関連する設定作業について説明しました。アプリケーションの設定には、ユーザ管理、セキュリティ設定など、さまざまなLinuxの機能に関する知識が必要となります。Webで検索すれば、各種アプリケーションの設定方法が見つかりますが、書かれていることをそのまま実行するのではなく、1つひとつの作業の意味を理解しながら、設定作業をすすめたいものです。

次のパートは、「番外編」として、Red Hat Enterprise Linuxの新バージョン、RHEL7における新たなサービス管理機能とその設定方法を紹介します。SD

▼図4　Sambaのファイルアクセス方式



▼図5　共有フォルダに保存したファイルの所有ユーザ確認

```
# echo "Test file" > file01.txt
# smbclient -U winuser //localhost/winuser
Enter winuser's password: ←――― winuserのSamba用パスワード
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.6.9-164.el6]
smb: ¥> put file01.txt
putting file file01.txt as ¥file01.txt (4.9 kb/s) (average 4.9 kb/s)
smb: ¥> exit
# ls -l /home/winuser/
合計 4
-rwxr--r--. 1 winuser winuser 10  3月 11 20:47 2014 file01.txt
```

Part 6

新しいOSの招待 来たるべきRHEL7への準備

本稿では、近々リリースが予定されているRed Hat Enterprise Linux 7に搭載される予定のsystemdについて解説をします。すでにβ版で利用が可能ですが、その基本機能であるUnitのしくみ、systemctlコマンドによるサービス管理について、解説を行います。

RHEL7Betaで新機能「systemd」を体験

Part6では、少し話題を変えて、Red Hat Enterprise Linux 7(RHEL7)で採用予定の新機能「systemd」を紹介します。RHEL7の正式リリースは今年(2014年)中と予定されていますが、すでにRed Hatからはベータ版が公開されており、さまざまな新機能を体験することができます[注1]。

その中でも、とくにsystemdはLinuxの起動処理にかかわる役割を果たします。Part3で説明したように、Linuxカーネルが起動すると、最初のプロセスとして「**/sbin/init**」を起動します。RHEL6では、このあとUpstartと呼ばれるしくみによって、システム初期化スクリプト「**/etc/rc.d/rc.sysinit**」やサービス起動スクリプトが実行されていきます。

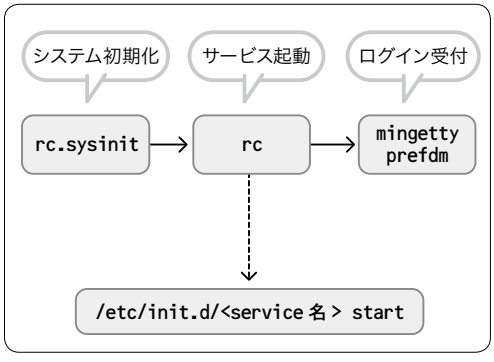
一方、RHEL7の場合ではLinuxカーネルが、「**/sbin/init**」の代わりに、systemdの実行ファイル「**/usr/lib/systemd/systemd**」を起動します。systemdは、Upstartのしくみをそっくり入れ替えるもので、各種スクリプトを実行する代わりに、「Unit」と呼ばれる細かな単位の処理を並列実行することで、システムの初期化やサービスの起動を行います。これには、システム起動時間の短縮やサービスに関連するプロセスの管理を強化する効果があります。

ただし、その一方で、サービスの起動・停止方法や各種サービスの設定ファイルなど、サービスにかかわる管理方法がこれまでとは大きく変わります。ここでは、systemdのしくみについて、その概要を説明したあと、関連するコマンドの使い方を紹介します。

systemdを支える「Unit」のしくみ

これまでのシステム起動処理では、シェルスクリプトが多用されていました。**図1**は、典型的な起動処理の流れです。Part3では、スクリプト「**/etc/rc.d/rc.sysinit**」でシステム初期化を行ったあと、さまざまなサービスを起動すると説明しましたが、サービス起動処理については、スクリプト「**/etc/rc.d/rc**」が担当します。このスクリプトは、chkconfigコマンドで自動起動が設定されたサービスについて、対応する起動スクリプト「**/etc/init.d/<サービス名> start**」を順番に実行していきます。それが終わると、最後に「migetty」「prefdm」など、

▼図1 スクリプトによる起動処理



注1) RHEL7Betaの入手方法については、次のWebサイトを参照してください。
https://access.redhat.com/site/products/Red_Hat_Enterprise_Linux/Get-Beta

コンソールからのログインを受け付けるプロセスを起動します。

一方、systemdの環境では、このようなスクリプトはもはや存在しません。systemdにおけるシステム起動処理では、これらのスクリプトが実施していた処理内容を多数の「Unit」に分解して、それぞれのUnitを個別に実行していきます(**図2**)。Unitは、役割によって種類が分かれており、Unit名の末尾の拡張子で区別します。おもなUnitの種類は、**表1**のとおりです。

ただし、これらすべてのUnitを設定ファイルで定義するわけではありません。「mount」「swap」「device」などは、systemdが環境に応じて自動的に用意します。システム管理者が設定ファイルで管理する必要があるのは、おもには「service」と「target」の2種類になります。

これらのUnitの定義ファイルは、「**/usr/lib/systemd/system**」と「**/etc/systemd/system**」の2つのディレクトリの下にあります。「sshd.service」などのUnit名がそのまま定義ファイルの名前になります。「**/usr/lib/systemd/system**」のほうには、システムがデフォルトで提供する内容の定義ファイルがあります。システム管理者が編集する際は、いったん「**/etc/systemd/system**」の下にファイルをコピーしてから変更します。両方のディレクトリに同じ定義ファイルがある場合は、「**/etc/systemd/system**」のほうが優先されるしくみになっています。

Unitの依存関係と順序関係

図2のように、システム起動時の処理が多数のUnitに分解されるわけですが、これらは、まったくバラバラに実行されるわけではありません。

▼表1　主なUnitの種類

拡張子	名称	機能
service	サービス	サービスを起動
target	ターゲット	複数のUnitをグループ化するために使用
mount	マウントポイント	ファイルシステムをマウント(/etc/fstabから自動生成)
swap	スワップ	スワップ領域を有効化(/etc/fstabから自動生成)
device	デバイス	ディスクデバイスを表す(udevがデバイスを認識すると自動生成)

▼図2　起動処理を「Unit」に分解して実行

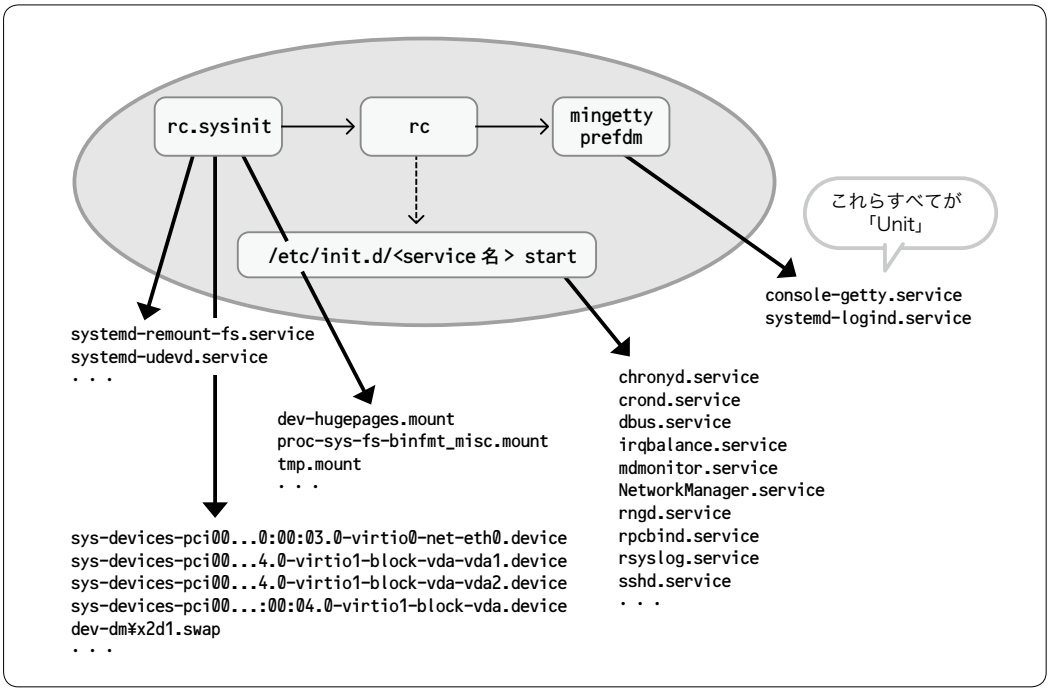


表1の「ターゲット」と呼ばれるUnitは、まとまった処理を行うUnit群を1つにまとめる役割を持ちます。**図3**は、主なターゲットの依存関係と各ターゲットに紐付いたUnitの役割を示しています。たとえば、さまざまなファイルシステムのマウント処理を行うUnit群は、「**local-fs.target**」というターゲットに紐付けられます。あるいは、従来ではランレベル3で起動していたサービスに対応するUnit群は、「**multi-user.target**」というターゲットに紐付けられます。

さらに、それぞれのターゲットは、**図3**の矢印で示した依存関係を持ちます。Linuxカーネルがsystemdを起動すると、systemdは、はじめに「**default.target**」というターゲットを検索します。このターゲットの設定ファイルは、実際にはほかのターゲットへのシンボリックリンクになっています。仮に「**multi-user.target**」へのシンボリックリンクだとすると、**図3**において、「**multi-user.target**」に連なる一連のターゲットとそれに紐付いたUnitが、起動対象としてまとめて選択されます。

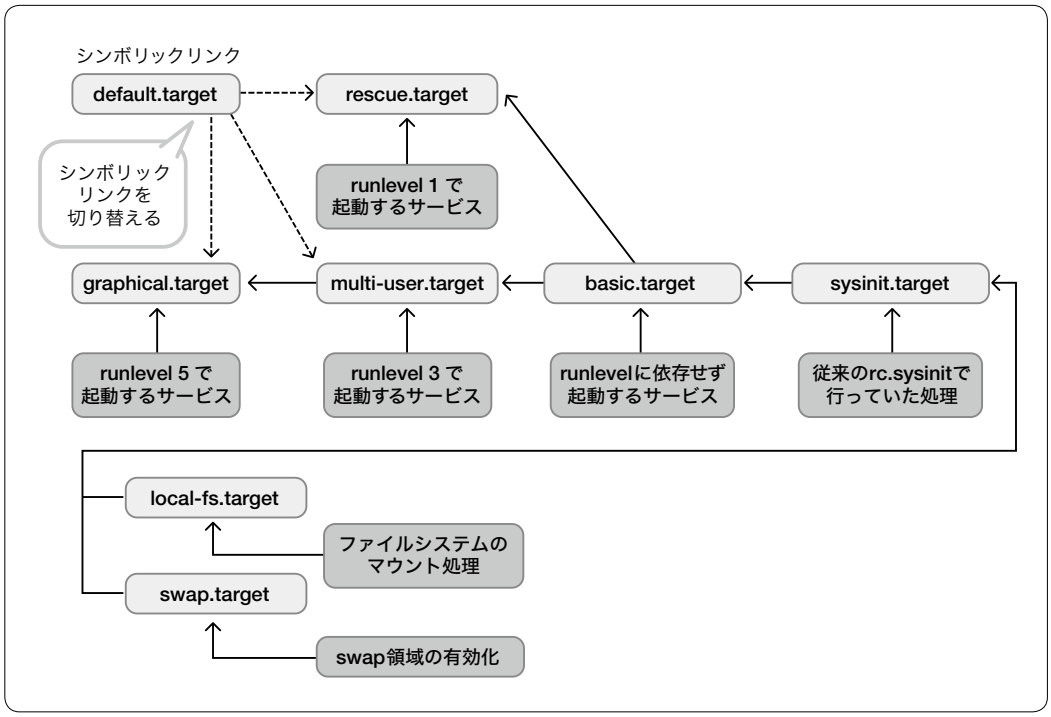
このとき、「**default.target**」のリンク先を変えることで、選択するサービスの種類が変更できます。これは、従来のしくみにおいて、「ランラベルを切り替える」という処理に相当します。systemdの環境では、ランレベルという考え方はなくなっていますが、このシンボリックリンクの切り替えによって、ランレベルの変更に相当する処理を行います。**図3**から読み取れるように、従来のランレベルとの対応は、**表2**のようにまとめられます。

これで、systemdが起動するべき一連のUnit群が決まりましたが、面白いのはこれらのUnitを起動していく順序です。**図3**を見ると、図の矢印の順にUnitが起動するように思われ

▼表2　デフォルトターゲットとランレベルの関係

ランレベル	ターゲット
1	rescue.target
3	multi-user.target
5	graphical.target

▼図3　ターゲットの依存関係



ますが、実際には少し異なります。それぞれのUnitが起動する順序は、**図3**とはまったく別に定義されており、「ネットワークを準備するUnit群」「ネットワークを使用するUnit群」のような、より大きな単位で順序付けが行われます。

この場合、「ネットワークを準備するUnit群」に含まれるUnitは、すべて同時に起動処理が行われます。これらがすべて起動したら、続いて、「ネットワークを使用するUnit群」に含まれるUnitに対して、一斉に起動処理を開始します。このようにして、できるだけ多数のUnitを並列に起動することで、システム起動時間の短縮が図られるというわけです。従来のシェルスクリプトによる起動処理では、1つ1つの処理は、スクリプトに記載された順序で行われますので、このような並列化ができませんでした。マルチコアCPUを搭載した、最近のサーバの性能を活かしきれなかったというわけです。

Unitの依存関係／順序関係の具体的な設定方法については、紙幅の都合で割愛します。詳細は、[1]の資料を参考にしてください。

systemctlコマンドによるサービス管理

それでは、systemdのしくみの解説はここまでにして、実際の使い方の説明をしていきましょう。systemdによるUnitの管理は、systemctlコマンドで行います。

まず、**図4**の上は「**list-unit-files**」サブコマンドで、定義済みのUnitを一覧表示しています。**表1**のように、Unitにはいくつかの種類がありますが、ここでは、「**--type=service**」オプションでサービスを起動するUnitだけを表示しています。「STATE」の列は、システム起動時に該当のサービスを自動起動するかどうかを示します(**表3**)。従来の「**chkconfig --list**」コマンドによる確認に対応する操作と言えます。

自動起動の有効／無効を切り替える時は、次

▼表3　サービスのステータス

ステータス	説明
enabled	自動起動が有効
disabled	自動起動が無効
static	自動起動設定の対象外

▼図4 サービスUnitの一覧表示

```
# systemctl list-unit-files --type=service
UNIT FILE                                   STATE
abrt-ccpp.service                           enabled
abrt-oops.service                           enabled
abrt-pstoreoops.service                     disabled
abrt-vmcore.service                         enabled
abrt-xorg.service                           enabled
abrtd.service                               enabled
accounts-daemon.service                     enabled
alsa-restore.service                        static
alsa-state.service                          static
…… (以下省略) ……
```

```
# systemctl --type=service
UNIT                          LOAD   ACTIVE  SUB      DESCRIPTION
abrt-ccpp.service             loaded  active  exited   Install ABRT coredump hook
abrt-oops.service             loaded  active  running  ABRT kernel log watcher
abrt-xorg.service             loaded  active  running  ABRT Xorg log watcher
abrtd.service                 loaded  active  running  ABRT Automated Bug Reporting Tool
alsa-state.service            loaded  active  running  Manage Sound Card State (restore and store)
atd.service                   loaded  active  running  Job spooling tools
auditd.service                loaded  active  running  Security Auditing Service
avahi-daemon.service          loaded  active  running  Avahi mDNS/DNS-SD Stack
chronyd.service               loaded  active  running  NTP client/server
…… (以下省略) ……
```

[1]　Linux女子部 —— systemd徹底入門！（http://www.slideshare.net/enakai/linux-27872553）

▼図5　サービスの状態確認

```
# systemctl status httpd.service
httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
   Active: active (running) since 水 2014-03-12 06:55:39 EDT; 1min 58s ago
 Main PID: 23727 (httpd)
   Status: "Total requests: 20; Current requests/sec: 0.1; Current traffic: 102 B/sec"
   CGroup: /system.slice/httpd.service
           ├─23727 /usr/sbin/httpd -DFOREGROUND
           ├─23728 /usr/sbin/httpd -DFOREGROUND
           ├─23729 /usr/sbin/httpd -DFOREGROUND
           ├─23730 /usr/sbin/httpd -DFOREGROUND
           ├─23731 /usr/sbin/httpd -DFOREGROUND
           └─23732 /usr/sbin/httpd -DFOREGROUND

 3月 12 06:55:39 localhost.localdomain httpd[23727]: AH00558: httpd: Could not reliably determixe
the server's fully qualified d...essage
 3月 12 06:55:39 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
 3月 12 06:55:48 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

関連するデーモンプロセス

関連する直近のログ

の「**enable/disable**」サブコマンドを使用します。それぞれ、指定のUnitの自動起動を有効化／無効化します。

```
# systemctl enable <Unit名>
# systemctl disable <Unit名>
```

図4の下は、systemctlコマンドをサブコマンドなしで実行していますが、これは、現在起動しているUnitの一覧表示です。先と同じく、サービスに対応するUnitのみを表示しています。本来起動しているはずのサービスが、何らかの問題で停止している場合は、「ACTIVE」の列に「failed」と表示されます。

また、従来のserviceコマンドに相当する操作として、手動でサービスを起動／停止／再起動する際は、「**start/stop/restart**」サブコマンドを使用します。

```
# systemctl start <Unit名>
# systemctl stop <Unit名>
# systemctl restart <Unit名>
```

さらに、特定のUnitの状態を確認するには、「**status**」サブコマンドを使用します。**図5**の実行例のように、サービスに関連するデーモンプロセスや直近のログなども併せて表示されます。

なお、サービスのログ出力は、従来のシステムログファイル「**/var/log/messages**」などにも出力されていますが、それとは別に、systemd自身も独自にログを保管しています。次のjournalctlコマンドを使用すると、特定のUnitに関連するログだけを抽出して確認することができます。

```
# journalctl -u <Unit名>
```

まとめ

少し駆け足になりましたが、Part6では、RHEL7の新機能「systemd」の解説をしました。システム起動処理という、OSとして基本的な処理について、このような改善の余地が残っていたのは意外に感じるかもしれません。本文でも触れたように、マルチコアCPUが標準になるなど、Linuxが稼働するハードウェア環境の進化がその背景にあります。クラウド環境での利用など、Linuxを活用する環境の変化による影響もあるでしょう。このような環境の変化に対応するべく、Linuxはこれからも進化を続けていきます。

そして、その進化に取り残されないためにも、「設定ファイルの読み方・書き方」といった基礎知識は、ますます重要になると言えるでしょう。この機会にぜひもう一度、Linuxの基礎を足元から見なおしてください。SD

SFで小飼弾

第6回

『あなたの魂に安らぎあれ』（神林長平／早川書房）

「人間とはなにか？」というのは哲学の主題にして文学の課題でもあるのですが、だとしたらSFほど哲学的にして文学的なジャンルはないのかもしれません。人のように考え、人のように動く、人でないものが他でこれほど登場することはないのですから。

そのSFの中でも、ひときわ哲学的で文学的なのが神林長平作品。最もよく知られ英訳もされた『戦闘妖精・雪風』の知的戦闘機雪風、最もよく売れアニメ化もされた『敵は海賊』シリーズの知的宇宙戦闘艦ラジェンドラ、『今宵銀河を杯にして』の知的戦車マヘルシャラルハシバズ……。同氏の描く知的兵器は多くのエンジニアに支持されてきましたし、私自身、彼／女らを肴に何晩でも語り明かせる自信があるのですが、今回取り上げる『あなたの魂に安らぎあれ』のアンドロイドは、ある意味SF史上最も地味な人造知性体かもしれません。

雪風やラジェンドラやマヘルシャラルハシバズが機械であるのに対し、こちらは生体。超音速で空をかけることもなければ、超光速で宇宙（そら）をかけることもない。もちろん電話はかけられない（苦笑）。切れば血も出るし、エッチもすれば子供も産む。人間と唯一の違いは、放射線耐性。舞台は核戦争で地表を汚染されてしまった火星。人類が破沙という洞窟の中で細々と暮らしているなかで、いつか人類が地表に戻る日に備えて地表を整備するのが汚染された地上でも生きていける彼らの任務のはずだったのが、主たる人類そっちのけで門倉京という都を築いて「人」生を満喫している。破沙の人類は、食糧もエネルギーも門倉京のアンドロイドに頼りっきりで、その代価を得るために門倉京の単純労働まで請け負う始末。同作の世界における主人はアンドロイドで、人類は彼らの慈悲にすがる乞食……。

しかし我が世の春を満喫するアンドロイドたちの間に、ひとつの神話がひそやかに伝えられていたのです。「神エンズビルが天から下り、すべてを破壊し、すべてが生まれる……」

エンズビルとは何物なのか。そもそもなぜ人類は自らを強化するのではなくアンドロイドという別種を創成することを選んだのか。破沙の人類も門倉のアンドロイドたちも知らないアンドロイドの秘密が明かされるとき、読者の魂に安らぎが待っています。SD

題字・イラスト／aico

第2特集

Windows XPからの乗り換えにいかが？

Ubuntu 14.04 “Trusty Tahr”

過酷な環境でも信頼できるLTSバージョン

4月17日にUbuntu 14.04 LTS(長期サポート)がリリースされました。本バージョンでは、Desktop、Server、Coreについては5年間、Ubuntu GNOME、Lubuntu、Ubuntu Studioなどの派生ディストリビューションは3年間のサポート期間が設定されました。愛称は“Trusty Tahr”(信頼できるタール)で、山岳地帯に生息する、絶滅危惧2類に指定されているヤギの仲間だそうです。

Ubuntuは、その取っつきやすさもあり、初心者が最初に利用することの多いLinuxディストリビューションです。本特集ではこのUbuntu 14.04 LTSをいろいろな角度から解説し、その本質に迫ります。

第1章 モバイルとデスクトップが統合される日は来るのか?

Ubuntu Japanese Team
長南 浩 CHONAN Hiroshi chonan@progdence.co.jp

本稿では、2014年4月17日にリリースされたUbuntu 14.04 LTS"Trusty Tahr"について説明します。

はじめに

Ubuntu 14.04 LTS"Trusty Tahr"が4月17日にリリースされました。2004年10月26日にリリースされた最初のバージョンのUbuntu 4.10 "Warty Warthog"から数えて、今回がちょうど20回目のリリースとなります。

Ubuntu 14.04 LTS "Trusty Tahr"

14.04 LTSのコードネームは「Trusty Thar」で、マスコットは「ヒマラヤ・タール」という、山岳地帯に生息するヤギの仲間[注1]です。コードネームには「過酷な環境でも信頼できる」思いが込められているようです。

今回のリリースは2年に一度のLTS(長期サポート)リリースで、Desktop、Server、Coreについては5年間、Ubuntu GNOME、Lubuntu、Ubuntu Studioといった派生ディストリビューションには3年間のサポート期間が設定されました。

1つ前のリリースである、13.10ではスマートフォンやタブレットと同じUIを実現するために、従来デスクトップ環境において使われてきたX11ディスプレイサーバをMir/XMirに移行する試みが行われました。今回の14.04 LTSにおいても、従来のX11とUnity 7の組み合わせが標準となり、Mir/XMirやUnity 8についてはプレビューとして用意されるという無難な路線に落ち着きました。

その代わり、Unity 7については細かな機能がいくつか追加され、よりブラッシュアップされた状況となりました。13.10で日本のユーザに評判が悪かったibusまわりも更新が入り、状況が改善されています[注2]。

今後、現在のようなX11 + Unity 7のデスクトップ環境がどれだけ長く維持されるのかは現時点ではわかりませんが、11.04で初めて搭載されたUnityデスクトップ環境の、一応の完成形が今回の14.04 LTSと言えます。

Unityデスクトップの変更点

スマートフォンやタブレットとデスクトップ環境の収束(Convergence)は次のリリース以降へと持ち越しになってしまいましたが、既存のUnityデスクトップ環境には細かな改良が加えられました。

Locally Integrated Menus

14.04 LTSの変更で最も目につきやすいのが、LIM(Locally Integrated Menus)です。Unityデスクトップ環境では、ディスプレイの縦方向のスペースを節約するために、Mac OS Xのようなグローバルメニューが実装されてきました。Macのような使い勝手については賛否両論がありましたが、マウスポインタがタイトルバーに近いときに、メニューをタイトルバー上に表示させる

注1) 通例のリリースに比べて、「動物園で飼育されている」程度にメジャーな動物となりました。とはいえ準絶滅危惧種となっています。

注2) とはいえ、まだ問題が残っているのも事実です。日本語入力環境については、72ページから詳しく解説しています。

設定ができるようになりました(**図1**、**図2**)。

Unity Spread

また、一見地味に見える変更ですが、Launcherアイコンをクリックしたときに表示されるウィンドウについて、dashのようにウィンドウタイトルで検索できるようになりました(**図3**)。

新しいロックスクリーン

一定時間操作しないと、作業を再開するためにはパスワードの入力が必要となります。その機能がロックスクリーンですが、14.04 LTSではこのパスワード入力画面がログイン時の画面(LightDM + Greeter)に変更されました(**図4**)。

▼図1　外観の設定画面。「ウィンドウのタイトルバーの中」が指定できる

▼図2　LIM設定によって、タイトルバー部分にメニューが表示されたところ

▼図3　Unity Spreadで、Firefoxのウィンドウから“weekly”をタイトルバーに含むものを抽出したところ

そのほかの変更点

そのほかは、次の細かい改良が加えられました。

- デフォルトの壁紙のデザインが変更された
- ウィンドウのリサイズ時に内容を表示したままリサイズされるようになった
- Launcherのアイコンのサイズの最小が16ピクセルとなった
- 画面全体のスケールを設定できるようになった(**図5**)
- 100%を超えた音量を設定できるようになった

インストール時の留意点

インストール時に注意しないといけない点は従来のリリースと同様で、

- データのバックアップをとる
- リリースノートに十分に目を通す

▼図4　デスクトップをロックしたところ。LightDMの表示に統一された

▼図5　メニューとタイトルバーの拡大縮小ができるようになった。解像度の高い環境でもメニューやタイトルバーの視認性を確保できる

の2点です。とくに12.04 LTSからのアップデートインストールを行う際には、12.04 LTSのリリース当時と事情が変化しているので、何の準備もなしにアップデートを強行することは避けてください。

新規インストール

新しいマシンに新規にインストールする際にはインストールメディアから起動しインストールを行います。Windowsなどがプレインストールされているノートの場合には、必要に応じてプレインストール状態でのバックアップをとっておいてください[注3]。

インストールは画面にしたがって質問に答える形で進みますが、13.10のときに質問されていた、Ubuntu Oneの設定についての質問が14.04 LTSで省略されています[注4]。なお、インストールする前にLive環境を起動して、ハードウェアの認識具合を確認したり、VirtualBoxなどの仮想PC環境で手順を確認しておくと良いでしょう。

13.10からのアップグレード

13.10からのアップグレードは、大きく分けて、

- インストールメディアから起動してアップグレードインストール
- 稼働している13.10環境からdo-release-upgradeコマンドやソフトウェアセンター経由でアップグレード

の2種類の方法があります。

インストールメディアからのアップグレードインストールは、新規インストールと同様にインストールメディアから起動し、インストール方法の質問で「アップグレード」を選択する方法です。稼働している13.10環境からのアップグレードは、インターネットへの接続があることを前提にオンラインでアップグレードする方法で、

```
$ sudo do-release-upgrade
```

として表示にしたがってアップグレードします。Unityデスクトップを使っている際には、ソフトウェアセンターから新しいリリースが利用可能な旨が通知され、アップグレードを促されますが、do-release-upgradeと同じ内容の処理が行われます。

標準的な環境の場合はそれほど大きな問題になる可能性が低いのですが、もしもの場合に備えてソフトウェアセンターの通知にしたがって気軽にアップグレードするのではなく、データのバックアップなどをとり準備を整えたうえでアップグレードすることをお勧めします。

12.04 LTSからのアップデート

前回のLTSである12.04 LTSからのアップデートインストールがサポートされています。安定性を重視する環境でLTSリリースが多く使われている事情もあるので、より注意深くデータのバックアップ、アップデート手順、12.04 LTSでの動作検証を行ったうえでアップデートすることをお勧めします。

とくに、Windows 8の発売前後に普及したUEFI[注5]搭載マシンで12.04 LTSを使っていた場合、UEFIパーティションを新たに作成する必要がある場合があります。

必要となる操作は、32/64bitの別やインストールするマシンのUEFI対応状況、12.04 LTSでのパーティション構成、Safebootの設定状況により異なります。内蔵HDDの全領域にUEFIモードで新規インストールする方法がまとめられている[注6]ので、事前に操作を確認する必要があります。

注3) 筆者の場合、使い始めてすぐにSSDや大容量HDDに交換してしまうことが多いので、「初回起動前」にストレージの交換を行ってしまい、プレインストールのものはそのまま保管してしまうようになりました。いろいろな意味で本末転倒なのですが……。
注4) 検証時にはあまり気にならなかったのですが、非常に残念なことに、Ubuntu Oneサービスが終了するというアナウンスが出されました。http://blog.canonical.com/2014/04/02/shutting-down-ubuntu-one-file-services/
注5) Unified Extensible Firmware Interface。BIOSの新仕様。
注6) https://wiki.ubuntulinux.jp/UbuntuTips/Install/UEFI

また12.04 LTSは、現段階でもサポート期間が3年間残っているので、あえてアップデートせずに12.04 LTSのまま運用を続けるという選択や、次のLTSまでアップデートを見送るという選択もとることができます。

なお、Alternate CDは12.10から廃止となったので、12.04 LTS当時とインストール手順、アップデート手順が異なる場合がありますので、必要に応じてインストール方法についても確認が必要です。

MirやUnity 8を試してみる

LTSの使い方としてはあまり好ましいものではありませんが、開発が進められているMirやUnity 8の試用法を紹介します。

Mirディスプレイサーバを使う

13.10などでの試用法が変更となり、14.04 LTSでは、

```
$ sudo apt-get install ubuntu-desktop-mir
```

としたあとに再起動することで試すことができるようになりました。とはいえ、うまく動かない場合もあるので、その場合にはSSHや仮想コンソールでログインして、ubuntu-desktop-mirパッケージを削除したうえで再起動します。

Unity 8を試用する

Ubuntu Touchと共通のものが使われる予定となっているUnity 8は、デフォルトとして搭載されなかった代わりにDesktop Previewが用意されています。

Unity 8パッケージをインストールしたうえで、

```
$ unity8
```

とすることで、試してみることができます(**図6**)。

また、デスクトップセッションとして使うためには、事前に「システム設定」の「システム」→「ソフトウェアとアップデート」の「アップデート」タブ上の「プレリリースされたアップデート」にチェックを入れたうえで、**図7**のようにすることでログインセッションにUnity 8を選択できるようになります。しかし、現状ではディスプレイの解像度にかかわらずスマートフォン用の画面が起動する状況です。

なお、「プレリリースされたアップデート」にはテストが十分行われていないものが含まれ予期せぬ挙動を示す可能性があるので、あくまでテスト用の環境で試用してください。

14.04の位置づけ

Ubuntu Touchが発表されて1年が経過し、デスクトップとタブレット、スマートフォンとの環境を統合する計画が進められてきましたが、14.04 LTSリリースの段階では未だ道半ばの状態です。

とはいえ、今まで育ててきたUnityもさらなる改良が加えられて全体のバランスがとられ、LTSにふさわしい内容になりました。前回のLTSリリース同様に、安定して長く使えるリリースとなりそうです。SD

▼図6 Unity 8デスクトップを起動したところ

▼図7 ログインセッションにUnity 8を加える

```
$ sudo apt-get update
$ sudo apt-get install unity8-desktop-session-mir
(Mirが正常に動かない場合には下記のX11ベースのパッケージを使用)
$ sudo apt-get install unity8-desktop-session-x11
```

第2章 UnityもいいけどGNOME Shellもね ピュアGNOMEのUbuntu GNOME 14.04 LTS

Ubuntu Japanese Team
あわしろいくや AWASHIRO Ikuya ikuya@fruitsbasket.info

あらたに14.04からLTSとしてリリースされることになった、GNOMEをフィーチャーしたフレーバーであるUbuntu GNOME 14.04を紹介します。

Ubuntu GNOMEとは

Ubuntu GNOMEはUbuntuの公式フレーバーの1つです。Ubuntuとの大きな違いはユーザインターフェース(UI)にUnityではなくGNOME Shellを採用していることであり、それ以外の差異はさほど大きくないので省略させてもらいます。発祥はUbuntu GNOME Remix 12.10であり、13.04から公式フレーバーとなって名称がUbuntu GNOMEと改められました。14.04は初のLTSであり、リリース後3年間サポートが継続されます。

Ubuntu GNOMEは「純粋なGNOMEデスクトップを提供する」と謳(うた)っており、確かにそうなっているのですが、コンポーネントレベルで見た場合はGNOMEのバージョンが混じっています。具体的には14.04だとGNOME 3.6[注1]/3.8/3.10/3.12のコンポーネントが混在しています。Fedoraは、たとえばFedora 20だと「純粋なGNOME 3.10デスクトップを提供」しているといえますが、Ubuntu GNOMEはそうはなっておらず、使い方などに微妙な違いがあることもあります。

ダウンロードとインストール

ダウンロード元の情報はWiki[注2]にあります。また、リリースノートもUbuntuやそのほかのフレーバーと同じところで公開されているため、事前に目を通しておいてください。インストーラはUbuntuと同じUbiquityですが、インストール中に表示されるスライドショーはUbuntuとは微妙に異なり、簡単な使い方の解説も書かれています。

GNOME Shell入門

インストーラから「直接インストール」を選択し、再起動後デスクトップマネージャー[注3]でログインします。上部にバー(トップバー)が表示されているだけでアプリケーションを起動するアイコンが見当たらず、面食らうかもしれません(**図1**)が、慌てず騒がずトップバーの[アクティビティ]をクリックするか、Superキー[注4]を押してください。クリックすると「アクティビ

▼図1 起動直後のUbuntu GNOME

注1) gnome-control-centerやgnome-terminalは3.6のままです。
注2) https://wiki.ubuntu.com/UbuntuGNOME/GetUbuntuGNOME
注3) LightDMではなく、GDMです。
注4) WindowsではWindowsキー、MacではCommandキー。

ティ」画面になります。左側にあるのが「ダッシュ(Dash)」で、ここからアプリケーションを起動します。右にちょこっと見えているのが「ワークスペース」です。ワークスペースは動的に増減することができ、これがGNOME Shellの特徴のひとつです。

ダッシュ

ダッシュの一番下にあるアイコンをクリックすると、2つのタブのうち[すべて]が表示されます(**図2**)。これは、インストールされているすべてのアプリケーションが表示されていますが、中には[諸ツール][ユーティリティ]など小分類もあります。ちなみに端末(gnome-terminal)は[ユーティリティ]以下にあります。一度起動したアプリケーションは[常用]に表示されます。すなわち、よく使うアプリケーションがここに表示されるということです。

ダッシュには起動しているアプリケーションも表示されます。そのアイコンを右クリックすると[お気に入りに追加]があり、これをクリックするとダッシュに登録されます。よく使うアプリケーションはダッシュに登録し、そこそこ使うアプリケーションは[常用]から起動、めったに使わないアプリケーションは[すべて]から起動するのが王道でしょう。あるいは、アクティビティ画面で検索文字として何文字か入力すれば、キーワードにマッチしたアプリケーションが表示されますので、キーワード(アプリケーション名がほとんど)がわかっていてアイコンから探すのが面倒な場合はその方法を取るのもいいでしょう。

ワークスペース

ワークスペースは、前述のとおり動的に増減できます。アクティビティ画面でウィンドウをドラッグし、右にある任意のワークスペースにドロップするとウィンドウを移動することができます(**図3**)。**図3**の例だと、中央に表示されているGNOME端末はドラッグ&ドロップで別のワークスペースに移動することができます。この際一番下のワークスペースにドロップすると、新規にワークスペースが追加されます。ワークスペースにあるすべてのウィンドウを終了させるか移動させると、そのワークスペースは消滅します。ワークスペースの切り替えはアクティビティ画面でそのワークスペースをクリックすることによって行えますが、Ctrl+Alt+上下矢印キーでも変更できます。また、Ctrl+Alt+Shift+上下矢印キーで現在アクティブなウィンドウのワークスペース間移動ができます。ドラッグ&ドロップする必要がないので、これを覚えておくと便利でしょう。

ちなみに筆者は1ワークスペースあたり1〜2ウィンドウにし、Ctrl+Alt+上下矢印キーで頻繁に変更するという方法を多用しています。2つのウィンドウを比較する場合はワークスペースを隣接させ、ショートカットで切り替えると

▼図2　ダッシュからアプリケーション一覧を表示

▼図3　任意のワークスペースに移動できる

作業効率が抜群にいいのでお勧めです。

トップバー

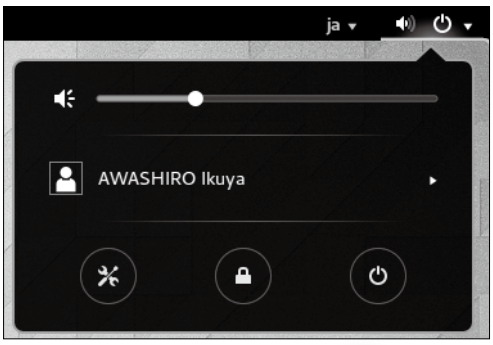
画面上部に常に表示されているトップバーにもさまざまな機能や情報があります。一部抜粋ですが解説します。

[アクティビティ]の横には、現在アクティブなウィンドウの名称が表示されます。この名称部分をクリックするとメニューが表示されますが、これをアプリケーションメニューと言います。アプリケーションメニューとして表示される項目はアプリケーションによりますが、メニューが少ないなと思ったらクリックしてみるといいでしょう。

トップバーの右端には電源などのアイコンがあり(システムステータスエリアといいます)、これをクリックするとシステムの状態が表示されます(**図4**)。**図4**下の3つのアイコンは、左から設定、ロック、電源オフです。ログアウトは表示されていませんが、ユーザ名をクリックすると表示されます。

ファイル(Nautilus)など一部のアプリケーションでは、タイトルバーとツールバーが統合したヘッダーバーが表示されています(**図5**)。これもGNOME 3.10からの新機能で、より画面を広く使用できるようになります。

▼図4 システムステータスエリアとシステム状態のポップアップ



GNOME Shell拡張機能

GNOME Shellの大きな特徴の1つに、GNOME Shell Extensions[注5]でたくさんの拡張機能が配布され、それらを容易にインストールできることが挙げられます。デフォルトでもたくさんの拡張機能がインストールされていますが、残念ながらここでは数が多すぎて紹介しきれません[注6]。

GNOME Shell Extensionsロゴの横にある[Installed Extensions]をクリックすると、インストールされている拡張機能の一覧が表示されます[注7]。中にはオン/オフスイッチの左側にツールアイコンがある拡張機能もありますが、これは設定できる機能があるものであり、このアイコンをクリックするとその拡張機能の設定画面が表示されます。

具体的には、ダッシュをドックにして、デスクトップ画面でもダッシュを表示できるようにする「Dash To Dock[注8]」とUnityのインジケータに対応した「AppIndicator Support[注9]」はお勧め

注5) https://extensions.gnome.org/
注6) デフォルトでインストールされている拡張機能の多くはGNOMEクラシックで使用されています。
注7) 後述するTweak Toolでも確認できます。
注8) https://extensions.gnome.org/extension/307/dash-to-dock/
注9) https://extensions.gnome.org/extension/615/appindicator-support/

▼図5 ファイル(Nautilus)のヘッダーバー

です。有用な拡張機能はたくさんあるので、お好みのものを探してみてください。

収録されるパッケージ

GNOME Shell固有の話からは外れて、Ubuntu GNOMEというLinuxディストリビューションについて俯瞰します。

前述のとおりUbuntu(Unity)とあまり違いはありませんが、デスクトップマネージャーがLightDMではなくGDMというのは、GNOME Shellの都合です。大きな違いはメーラーがThunderbirdではなくEvolutionになっていることでしょう。変わり種では、13.10まではUbuntuとUbuntu GNOMEともにgnome-control-center(g-c-c)とgnome-settings-daemon(g-s-d)を採用していました。どちらもUnityでうまく動作させるために大量のパッチが当たっており、容易にアップデートができませんでした。そこで14.04ではそれぞれ、unity-contorl-center(u-c-c)とunity-settings-daemon(u-s-d)としてフォークし、g-c-cとg-s-dはバージョンアップをしやすくしようという方策が取られました。しかし、蓋を開けてみるとu-c-cはそれまでの3.6から3.8ベースにバージョンアップしたものの、g-c-cは3.6のまま据え置き、g-s-dとu-s-dはともに3.8ベースのまま、という現象が起きました。なかなかままならないものです。

カスタマイズ

GNOME Shellは極力シンプルになるように作られており、各ウィンドウのタイトルバーには最大化・最小化ボタンすらありません。ほかにも設定できてしかるべきところがシステム設定[注10]から変更できません。カスタマイズしたい場合は、Tweak Tool(gnome-tweak-tool)を起動してください。これでかゆいところに手が届くようになります。たとえばタイトルバーに最大化・最小化ボタンをつける場合は[ウィンドウ]タブの[タイトルバーボタン]でオン／オフできます。

GNOMEクラシックとGNOME Fallback

GNOMEは3.0からGNOME Shellを採用し、GNOME 2.xとはルック＆フィールがまったく別のものになりました。旧来のルック＆フィールも「GNOME Fallback」としてGNOME 2.xから移植されたものが残っていましたが、それも3.6まで。メンテナンスされていないという理由で3.8からは削除され、代わりにGNOME Shellに拡張機能を追加して、GNOME Fallbackっぽいルック＆フィールにした「GNOMEクラシック」がGNOMEに搭載されました。

これはGNOME 3.10を採用しているUbuntu GNOMEからも利用でき、ログイン時に[GNOMEクラシック]を選択するだけです。GNOME ShellのUIには慣れないけれど、なんとなく新しいものを使いたい場合にはいいのではないでしょうか[注11]。

一方、確かにGNOME Fallbackは削除されたものの、新たにフォークされてGNOME Flashbackとしてフラッシュバック(逆行再現)しました。使用方法はgnome-session-flashbackパッケージをインストールしてログイン時に選択するだけです。GNOME 2.x時代のアプレットは使用できませんが、おおむねGNOME 2.xっぽいルック＆フィールを体験できます。ただ、問題点としてはUbuntu GNOMEだとテーマなどがいろいろいまいちなため、GNOME Flashbackを使用したい場合はUbuntu(Unity)にgnome-session-flashbackパッケージをインストールしたほうがいいです。SD

注10) これがgnome-control-centerです。

注11) 通常そのようなことはないので、なんというかお察しください。

日本語RemixではFcitxを採用。IBusも継続

第3章 Ubuntu 14.04のインプットメソッド事情

Ubuntu Japanese Team
あわしろいくや AWASHIRO Ikuya ikuya@fruitsbasket.info

インプットメソッドの変更点をできる限りコンパクトにまとめてお伝えします。

12.04からのアップグレードには注意が必要

本章では、Ubuntu 14.04のインプットメソッド(以下、IM[注1])について書いていきますが、主な読者ターゲットは12.04からアップグレードした/する予定の方です。13.10からアップグレードした/する予定の方はご存じのことが多いでしょうし、初めてインストールする方は前はそんなんだったのかと思いながら読んでいただければ幸いです。なお、本誌2013年10月号のUbuntu Monthly Report第42回で『インプットメソッドの変遷とIBus 1.5』という記事を書きましたので、お手元にあるなら今一度読んでいただけると幸いです。

インプットメソッド変更の背景

さて、Ubuntu 12.04ではIMに関してとくに悩むことはなかったのではないかと思います。IMのバージョンはIBus 1.4.1でした。しかし13.10からIBusのバージョンが1.5.3に上がり、使い勝手がだいぶ異なることになりました。

IBus 1.4以前ではIBusにパッチを当ててUnityのインジケーターに対応させていましたが、13.10からはこれをなくし、新たにキーボードインジケーター(indicator-keyboard)を実装し、これを使用するようになりました。もちろん、キーボードインジケーターはIBusと協調して動作していますが、いろいろとトラブルの元にもなったのです。たとえばキーボードインジケーターからは変換エンジンであるAnthyやMozcなどのステータスが確認できず、また設定ツールの起動もできませんでした。

これとIBus 1.5の設定変更が重なり、13.10でIBusを使用するのはとても難しい、ということになりました。そこで、IBus 1.4と比較的似たような挙動で、かつ機能豊富な「Fcitx」というIMを13.10から使用可能にしました。Ubuntu日本語RemixではIm-setup-helperというパッケージを追加し、MozcのデフォルトとFcitxへの変更が簡単にできるようにしました[注2]。

14.04ではIBusのバージョンが1.5.5になり[注3]、またキーボードインジケーターのバージョンも上がってほぼ実用段階まできました。なぜなら、新たにプロパティパネルという機能が実装され、現在の入力状態が半角なのか全角なのかわからないとか、変換エンジンの各種ツールが起動できないといった問題がなくなったからです[注4]。**図1**の左側、「あ」とスパナアイコンが表示されているのがプロパティパネルで、そのすぐ右に表示されているのがキーボードインジケーターです。

とはいえ、今まで(12.04まで)と同じ使い勝手というわけではないので、日本語Remixでは

注1) かつては「日本語入力」と呼ばれていましたが、入力できるのは日本語だけではないのでIMという用語を使用します。が、「日本語入力」のほうが通りがいいのは知っていますし、筆者もたまに使ったりします。

注2) 本誌2014年1月号のUbuntu Monthly Report第45回にIMセットアップヘルパーの仕様や作成に使用したツールについて書きました。

注3) 13.10ではIBus 1.5.3でした。

注4) もっとも、Ubuntu GNOMEというかGNOME Shellではもともと13.10でも問題ありませんでした。

新たにFcitxを採用することになりました。Ubuntu KylinというUbuntuフレーバーがあり、Fcitxはそこでの採用事例があるのです。IBusを使い続けたい方のために、im-setup-helperに大幅に手を加えて、FcitxからIBusに変更できるようにもしました[注5]。

Fcitx

Fcitxは聞き慣れないIMかもしれませんが、開発の開始は2002年からと、IBusよりも古いくらいです。もともとは中国語のみの対応だったのが、内部構造に大幅に手を入れて多言語対応しました。Anthyとのブリッジであるfcitx-anthyやMozcとのブリッジであるfcitx-mozcも以前からありましたが、当初はほぼ使い物になりませんでした。しかし、2013年3月頃から本格的にFcitxの翻訳を始め、バグ報告などが急速に行われるようになり、13.10がリリースされる頃には日常的に使用できるようになりました。Fcitxに関しては、13.10と14.04でほぼ違いはありません。

Ubuntu 13.10のUnityでFcitxを使用すると、Dashの入力欄で候補ウィンドウが表示されないという不具合がありましたが、これはUnityのツールキットであるnuxのバグであり、14.04で修正されました。これでUnityでも問題なく使用できるようになりました。

FcitxとIBus 1.4の違いについてはWiki[注6]に書きましたので、こちらを参照いただくのがいいかと思います。

IBus

前述のとおり、IBusとキーボードインジケーターのバージョンが上がってプロパティパネルが使用できるようになりました。これで現在の変換エンジンの状態とツールの起動ができるようになり、13.10の頃のような使いにくさはなくなったと思います。ただし、表示されたり消えたりで煩わしく思うところもあるかもしれません。ですがステータスがわからないよりはずっといいはずです。そのほかにもいろいろ使い勝手は良くなっていますが、それでも12.04と同等とはいきません。そのあたりについてはUbuntu Weekly Recipe 第319回[注7]に書きましたので、IBusを使用したい場合はご一読いただけると幸いです。

▼図1　プロパティパネルとキーボードインジケーター

IMセットアップヘルパー

13.10からGNOMEとIBusが統合することによりUnityにも影響があり、Mozcを簡単にデフォルトにするということができなくなってしまいました。そのあたりを何とかしようと考えて書いたのが、このIMセットアップヘルパーです。これはパッケージ名であり、Mozcセットアップヘルパーと Fcitxセットアップヘルパーの2つのスクリプトが収録されています。

14.04ではFcitxをデフォルトにしたのであまり使う機会はないかもしれませんが、XubuntuやLubuntuやKubuntuユーザには有用だと思います。もちろんIBusを使用したいユーザにも役に立つことでしょう。SD

注5）im-setup-helperにはmozc-setup-helperとfcitx-setup-helperが含まれており、名前どおりの役割を果たしていますが、13.10ではfcitx-setup-helperからIBusに戻す設定ができました。これは14.04では不自然なのでmozc-setup-helperからできるようにしました。

注6）https://wiki.ubuntulinux.jp/UbuntuTips/JapaneseEnvironment/Fcitx

注7）http://gihyo.jp/admin/serial/01/ubuntu-recipe/0319

サポート期間／アップデート／HES／ポイントリリース

第4章 Long Term Supportってなんだろう

Ubuntu Japanese Team
あわしろいくや AWASHIRO Ikuya ikuya@fruitsbasket.info

14.04はLTSですが、具体的にLTS(Long Term Support)にはどういう特徴があるのでしょう。また、最近はどんなポリシーの変更があったのでしょう。LTSに関する詳細を解説します。

LTSってなんでしたっけ?

まずはLTSのおさらいからです。LTSはLong Term Supportの略で、Ubuntuは偶数年の4月にLTS版をリリースします。LTSはその名のとおり長期間サポート版で、通常のUbuntuのリリース(Standard Release)がリリース後9ヵ月サポート[注1]のところ、Ubuntu(Desktop)とUbuntu Serverはともに5年間サポートが継続されます。WindowsやRed Hat Enterprise Linuxは10年間かそれ以上サポートされるので比較すると短いですが、無償で提供されていることとリリース時期が決まっていて計画が立てやすいというメリットがあります。

5年間何のサポートを継続するのかというと、もちろんセキュリティフィックスなど各種アップデートの提供です。Canonical[注2]と商用サポート契約を結んでいる場合は、そのサポート期間とも一致します。今のところ5年間以上のサポート継続については、特別にCanonicalと契約すれば提供されるといったものはありません[注3]。

なお、Ubuntuにはさまざまな公式派生版(フレーバー)があり、これらもそれぞれLTSとしてサポート期間が延長されますが、フレーバーによって3年間ないし5年間とまちまちです。

Ubuntuは通常バージョンを飛ばしてのアップグレードはできません。しかし、LTSに関しては前のLTSから新しいLTSへのアップグレードがテストされ、実行できます[注4]。ただし12.04で実際にあったことですが、10.04から12.04へのアップグレードが12.04.1リリース後に推奨になるといったケースもありますので、情報収集を密に行う必要はあります[注5]。

最初のLTSはUbuntu 6.06で、約8年を経てLTSの方針もいろいろと変わりました。続いてその変更点をお知らせします。

各フレーバーとサポート期間

各フレーバーのサポート期間は、それぞれのコミュニティから出された提案に対してTechnical Committeeが承認する、というプロセスで決定されます。12.04まではKubuntuのサポートはCanonicalが行っていました[注6]が、今回からは別の会社[注7]でサポートすることになります。よってCanonicalがサポートするUbuntuフレーバーはEdubuntuだけになりました。

各フレーバーのサポート期間を**表1**にまとめます。サポート期間はほぼコミュニティの強力さ(あるいはスポンサーの有無)に比例していると考えられます。

注1) 以前は18ヵ月サポートでしたが、13.04から短縮されました。
注2) http://www.canonical.com/
注3) オープンソースなのでCanoncal以外にもこのようなことを行う企業があっても不思議ではありませんが、今のところは聞きません。
注4) もちろんうまく行くかはまた別の話ですが。
注5) たいていはリリースノートを読むと書いてあります。
注6) 正確には現在も継続しているので「サポートしていますが」です。
注7) Blue Systemsというドイツの会社です。Wikipediaによると、少なくとも9人のKDE開発者を雇用しているとのことです。CanonicalとUbuntuの関係とだいたい同じと考えていいのではないかと思いますが、Blue Systemsという会社は調べてもよくわからないのでなんとも言えません。

14.04から新たにLTSになったフレーバーはUbuntu GNOMEとUbuntu Kylinです。Ubuntu GNOMEは第2章を参照いただくとして、Ubuntu Kylinは中国向けのフレーバーです。このフレーバーで興味深いところは、日本でもお馴染みKingsoft Office（WPS Office）が無償提供されていることでしょうか[注8]。

開発背景

通常のリリースでは新機能を搭載することに注力していますが、LTSではブラッシュアップに注力し、あまり新機能は搭載されません。14.04でもまさにそんな感じでした。意欲的な新機能はLTS後の通常リリースに先延ばしされることもまま起きています。通常リリースもLTSも開発期間は同じ半年間なので、そうなるのは当然といっていいでしょう。

開発に関して14.04で変わったことは、以前はDebianのリポジトリからパッケージを取り込む際にtestingから行われていました。しかし今回からは通常リリースと同じくunstable（sid）から行われるようになり、より新しいバージョンがインポートされるようになりました。これまで開発期間には直接そのリポジトリにインポートされていましたが、今はいったんproposedリポジトリにアップロードされ、問題ない場合にアップデートとして提供されるというポリシーの変更を受けてのことです。これはもちろんDebianからインポートされるパッケージだけではなく、全体的にそうなったことによって比較的安全に開発版の使用ができるようになったということでもあります[注9]。

▼表1　Ubuntuフレーバーのサポート期間

フレーバー名	サポート期間
Kubuntu	5年
Xubuntu	3年
Lubuntu	3年
Edubuntu	5年
Ubuntu Studio	3年
Mythbuntu	3年
Ubuntu GNOME	3年
Ubuntu Kylin	5年

注8）Debパッケージは配布されており、いちおう日本語でも使えなくはないです。

MREとSRUとバックポート

Ubuntuで提供されるアップデートはいくつかあります。セキュリティアップデートは“-security”リポジトリで提供されています。“-”の前には各リリースのコードネームが入ります。14.04だと“trusty-security”です。ある特定かつ小さな不具合修正が行われた場合、一度“-proposed”リポジトリにアップロードされてテストが行われ、問題なければ“-updates”リポジトリに入って広く提供されます[注10]。これをStable Release Updates、略してSRUと呼んでいます。

セキュリティの修正もSRUも原則としては最低限必要な差分のみを適用して提供されていますが、一部パッケージに関してはマイナーバージョンアップの場合は差分の提供を行わず、まるごとアップデートしてしまうという例外があります。これをMicro Release Exception、略してMREと呼んでいます。典型的な例としてはFirefox/Thunderbirdで、これらは常に最新版を提供しています。延長サポート版（Extended Support Release）もありますが、これではなく通常版を提供しています。Webの世界の進化は速く、これについていくには通常版を提供するのが最善であるという決定に基づいています。ちなみにChromium[注11]もMREの対象ですが、独自のカスタマイズが多くてGoogle Chromeリリースからはだいぶ遅れて提供されています。

そのほか、新しいバージョンを提供するバックポートもあります。ただし、活用されているとはいえないのが現状です。

注9）それでも無理解によるとんでもないバグが投下されて、IBusユーザが悶絶したということもありました。
注10）-proposedはデフォルトで無効になっていますが、-updatesはデフォルトで有効になっています。
注11）Google Chromeのオープンソース版です。

HES

サポート期間が長くても、対応するハードウェアが増えなければありがたみが半減します。この問題の回答として、UbuntuではLTSリリース後にリリースされたUbuntuのカーネルとXスタック[注12]をバックポートしています。新しいカーネルとXスタックは新しいハードウェアにも対応している、ということです。これをLTS Hardware Enablement Stack(HES)といいます。具体的には12.04のリリース後12.10～13.10までのカーネルとXスタックがインストールできるようになっています。

もちろん12.04リリース時点でのカーネルとXスタックもそのまま利用できます。すなわち、HESを使用するかどうかはユーザが選択できます。もちろん新しいハードウェアを使用する場合は選択の余地なくHESを使用することになりますが、通常のカーネルやXスタックで問題ない場合はそのままでもいいですし、新しくしたい場合もパッケージをインストールするだけで安全に行うことができます。

ハードウェアを更新したいけれどもUbuntuのバージョンはそのままにしておきたいという場合は、事前にHESで提供されたカーネルとXスタックをインストールしてからHDDを新しいPCに移行するということも可能です。もちろん移行の方法はHDDをそのまま継続利用してもいいですし、ddコマンドで新しいHDDにコピーしてからでもいいです。このあたりはUbuntuの強みでもあります[注13]。

HESのインストールパッケージは**表2**のとおりです。公式[注14]では**図1**の方法を推奨していますが、検証してみたところ**表2**のパッケージをインストールするだけでいいです。Ubuntu ServerなどXが不要な場合はカーネルのみを、それ以外の場合はXスタックを指定すればカーネルほか、まるごとインストールされます。

もちろんこれは12.04の実績の話ではありますが、詳細は執筆段階で未公表であるものの、14.04でも原則としては同じように進行するはずです。

ポイントリリース

LTSでは次のLTSが出るまで事前に提示し

注12) X.Orgのサーバ・クライアント・ドライバその他もろもろをまとめてこう呼んでいます。

注13) とはいえWindowsでもプリインストール版でなければ可能ですし、Red Hat Enterprise Linuxでも規約を読んだ限りでは問題なさそうなので、さほどのメリットとはいえないかもしれません。

注14) https://wiki.ubuntu.com/Kernel/LTSEnablementStack

▼表2　HESカーネルとXスタックのパッケージ名一覧

HESのベースバージョン	カーネル	Xスタック
12.10	linux-generic-lts-quantal	xserver-xorg-lts-quantal
13.04	linux-generic-lts-raring	xserver-xorg-lts-raring
13.10	linux-generic-lts-saucy	xserver-xorg-lts-saucy
14.04(予想)	linux-generic-lts-trusty	xserver-xorg-lts-trusty

▼図1　公式に推奨しているHESのインストール方法

```
【12.10ベース】
sudo apt-get install --install-recommends linux-generic-lts-quantal xserver-xorg-lts-quantal
libgl1-mesa-glx-lts-quantal
【13.04ベース】
sudo apt-get install --install-recommends linux-generic-lts-raring xserver-xorg-lts-raring
libgl1-mesa-glx-lts-raring
【13.10ベース】
sudo apt-get install --install-recommends linux-generic-lts-saucy xserver-xorg-lts-saucy
libgl1-mesa-glx-lts-saucy
```

てあるスケジュール[注15]に沿って、累積したアップデートをまとめてインストールイメージにし、XX.04.1といったポイントリリースを提供しています。12.04からはHESの導入を、12.04.2では12.10の機能を、12.04.3では13.04のHESを適用したポイントリリースを提供するようになりました(**表3**)。これで新しいハードウェアにLTSがインストールできることになり、LTSがますます便利になりました。一方、仮想マシンにインストールする場合はカーネルやXスタックが新しい必要はまったくないうえに、HES未適用のイメージが見つけにくくなるというデメリットもあります。

今のところはリリースされているすべてのHESがサポートされていますが、14.04リリース後12.04.5がリリースされると、これ以外のHESはサポート打ち切りとなり、HESが必要な場合はアップデート必須となる予定です。「予定です」というのは12.04リリース前にはこのような計画はなく、12.04.4で打ち止めとなる予定でした。しかし、それだとあと3年間すべてのHESのサポートを継続しなければいけなくなるため、メンテナンスがとても大変です。そこでこのようなことが考え出されました。現在のところ12.04.5の詳細は発表されていないため、12.04を継続して使用したい場合は続報を待つのがいいでしょう。

仮想マシンで動作させる場合は、Xスタックもカーネルも新しい必要はまったくありません。そのような場合は、HES対応前のポイントリリース(たとえば12.04.1)をold-releases.ubuntu.com[注16]からダウンロードしておくといいでしょう。アップデートに時間はかかりますが、カーネルとXスタック以外は最新のポイントリリースと同様の状態になります。

注15) とはいえポイントリリースは期日どおりにリリースされるほうが少ないですが、さまざまな兼ね合いのうえなのでやむをえないでしょう。

▼表3　これまでのポイントリリース一覧

バージョン	リリース日	HESのベースバージョン
12.04(参考)	2012/4/26	-
12.04.1	2012/8/24	-
12.04.2	2013/2/15	12.10(Quantal)
12.04.3	2013/8/24	13.04(Raring)
12.04.4	2014/2/7	13.10(Saucy)
12.04.5	2014/8/7予定	14.04(Trusty)

悩ましいPPA

LTSを長く使っていると、ピンポイントでこのパッケージだけ新しくしたいということがあります。そのような場合はPPA(Personal Package Archives)で検索してみて必要なパッケージが提供されていないか探したり、あるいは自前でパッケージのビルドを行ってPPAにアップロードするという手があります。しかしPPAは玉石混交で、リポジトリから取得できるパッケージと同じレベルのものもあれば、とりあえずパッケージにしましたというものまでまちまちです。質の低いパッケージはインストールしたくありませんが、ソースを取得して品質を確認するのがまた大変な作業で、知識が求められます。それ以前にそんな手間なんてかけていられません。一方、LibreOfficeの最新版はLTS向けにも提供されており、おおむね安心してインストールできる品質になっています。あるいはアップストリームの開発者がPPAで最新版のパッケージを配布していることも多く、こちらもおおむね安心できる品質です。

よって、過度に恐れる必要はありませんが全面的に信頼もできません。それよりLTS間アップグレードの際に問題になる可能性もあるので、仮想マシンを用意して一度実験するとか、ppa-purgeコマンドをうまく使ってアップグレードの前にPPAでインストールしたパッケージをリポジトリで提供されているバージョンに戻してしまうとか、自分なりの対策をたてたうえで、PPAを使用するのがLTSを長く使うコツのように思います。SD

注16) http://old-releases.ubuntu.com/releases/

第5章 本格的に使えるようになった! Ubuntu SDKとUbuntu Touch

Ubuntu Japanese Team
柴田 充也 SHIBATA Mitsuya mty.shibata@gmail.com

Ubuntu 14.04 LTSでは、Ubuntu SDKとUbuntu Touchが本格的に使えるようになりました。デスクトップの、とくに日本のユーザにとってはまだあまり影響はありませんが、将来のUbuntuにかかわってくる話でもありますので、ここで紹介しましょう。

Ubuntu SDK

Ubuntu 14.04 LTSにおいて、エポックメイキングになりそうな機能の1つが「Ubuntu SDK」の搭載です。Ubuntu SDKはUbuntu向けのアプリを開発するための、QtCreatorをベースにした統合開発環境(IDE)であり、そこで使われているライブラリ群やドキュメントの総称でもあります。

UbuntuにSDKは必要なのか

「Linuxディストリビューション」といえば、Linuxカーネルと既存のソフトウェアに対して独自の味付けをしたうえで1つにまとめて配布する存在です。もちろんディストリビューションごとに独自のソフトウェアを開発することもありますが、とくにGUIアプリの開発はディストリビューションよりは、GNOMEやKDEといったデスクトップ環境のコミュニティやコミュニティ内外の個人がディストリビューションを限定せずに開発することがほとんどです。だからこそGNOMEの大量のソフトウェア資産は各種ディストリビューションで使われていますし、Red Hatで開発された便利なソフトウェアをUbuntuで使うこともできるのです。

そのためAndroid SDKやWindows SDKのような位置付けとなる、「Linuxディストリビューション用のSDK」はあまり存在しませんでした。各ソフトウェアの開発者が目的や好みにあわせて自由にツールキットやフレームワーク、IDE、言語を選択すればよかったのです。

では、なぜUbuntuはあえて「独自のSDK」を搭載するようになったのでしょうか。これには「Ubuntu Touch」というモバイル向けOSの存在と、「Convergence」というUbuntuの野望がかかわってきます。

新しいモバイルOSにとって一番問題となるのは「アプリのエコシステムを作れるかどうか」です。iOSもAndroidも、立ち上げ初期はストアアプリの数やダウンロード数をさかんに喧伝していました。これはすでにFLOSSの膨大な資産をリポジトリとして持っているUbuntu Touchも例外ではありません[注1]。デスクトップとモバイルではリソースも見た目も使い勝手もまったく異なるため、モバイルならではのアプリをどんどん投入して、Ubuntu Touchの使い勝手を向上させる必要があります。

加えて、Ubuntuには「Convergence」という長期的な目標もあります。詳しい話は「第1章の14.04の位置付け」で述べられていますが、簡単に言うと「デスクトップやモバイル、そのほかのUbuntuベースのデバイスで同じコードベースのアプリを使えるようにし、その使い勝手や見た目をConvergence(収束)させる」ことが目標です。

すでに述べたとおり、デスクトップとモバイルでは前提条件がだいぶ異なります。モバイル間のちょっとしたデバイスの差異でさえ苦労しているのに、それ以上に異なる環境をアプリの開発者が

注1) 実際のところGUIを提供するソフトウェアはそのままでは使えません。CUIで完結するソフトウェアはそのまま使える可能性は高いですが、そのためにapt-getでのインストールを許すかどうかはメーカー側の判断となります。

フォローするのは非常にたいへんでしょう。

そこで、その部分をUbuntu側のフレームワークで対応しようとした結果、Ubuntu SDKが登場することになりました(**図1**)。たとえばコードでは独自単位でUI部品のサイズを指定すれば、実行時にデバイスのDPIやユーザとデバイスの距離に応じて適切なピクセルサイズを算出して表示するしくみや、モバイルとデスクトップでのメニューの表示方法を変えるしくみなどをUbuntu SDKのフレームワークで提供しています。

Ubuntu TouchのUIとコアアプリは、Qt5/QMLで作成しています。このため、Ubuntu SDKはQtのIDEであるQtCreatorをベースに、Ubuntu特有の機能を拡張して提供しています。実はUbuntu 13.04の時点でUbuntu SDKは存在はしていたものの、QtCreatorのバージョンアップやQtが5.0から5.2へ変わる中でさまざまな実験的な機能をブラッシュアップしていった結果、14.04のUbuntu SDKはだいぶ「使える」状態になりました。

そこで、ここでは実際にUbuntu SDKで実際にアプリを作る過程を見ていきましょう。なお、Qtは5.2からAndroidやiOSへの正式な対応も行われています。今後はQtを押さえておけば、AndroidやiOSのアプリも同じように作ることができるでしょう。

どれほど簡単にアプリを作れるのか

Ubuntu SDKは、リポジトリからubuntu-sdkパッケージとしてインストールできます。ただし今後はPPA(Personal Package Archive)を使って、より新しいSDKを提供する可能性もあるので、具体的なインストール手順は公式ページ[注2]を参照してください。Qt5も一緒にインストールされるので、かなり大きなデータをダウンロードします。

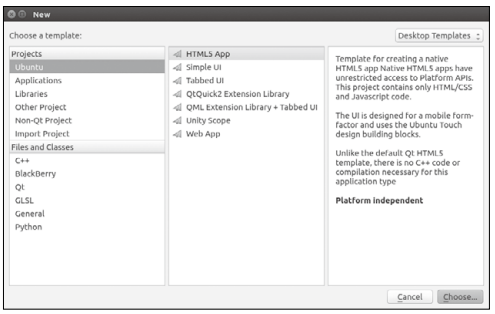
インストールが終わったらDashから「Ubuntu SDK」を検索して起動しましょう。Fileメニューの「New file or Project...」を選択すると、新しいプロジェクトの作成ウィザード(**図2**)が起動します。

QtCreatorには、もともとさまざまなプロジェクトテンプレートが存在しますが、Ubuntu SDKにはさらにUbuntuアプリ用のテンプレートがいくつか追加されています。

- Simple UI：1つのウィンドウにボタンとラベルを追加したシンプルなQMLアプリを作成します
- Tabbed UI：Ubuntu Touchのタブ形式に対応したQMLアプリを作成します
- HTML5 App：CordovaベースのHTML5アプリを作る場合に使用します
- Web App：W3C準拠のWebアプリを作成します
- QtQuick2 Extension Library：QMLを拡張するためのC++プラグインを作成します
- QML Extention Library＋Tabbed UI：QML

▼図1　Ubuntu SDKのメイン画面

▼図2　プロジェクトウィザード



注2) http://developer.ubuntu.com/apps/sdk/

とC++を組み合わせたアプリを作成します
・Unity Scope：UnityのScopeを作成します

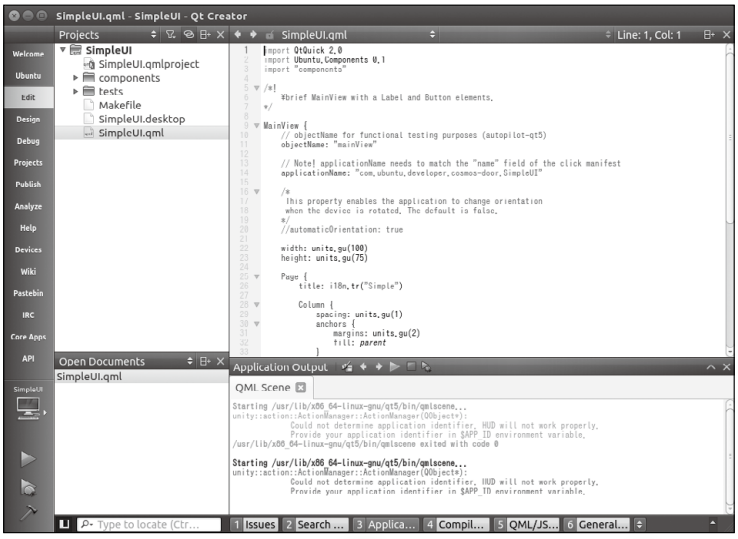
いくつもメニューはあるものの、大雑把に分けるとQMLで作るかHTML5で作るかの違いになります。特別HTML5を使いたい理由がなければ、より拡張性の高いQMLを選択しておくと良いでしょう。最初のうちはQMLだけで作成し、QMLでは実現できない機能が出てきたらC++で拡張するスタイルがお勧めです。

そうすると一番最初に選択するのは「Simple UI」か「Tabbed UI」の2択になります。後者はUbuntu Touchらしくメニューバーやツールバーが追加されるものの、若干複雑なコードが生成されます。QMLに慣れないうちはSimple UIから始める方が理解しやすいかもしれません[注3]。

HTML5 AppとWeb Appは、いずれもHTML5/JavaScript/CSSで開発するスタイルです。HTML5 AppはUbuntuプラットフォームの機能にアクセスできるのに対して、Web AppはW3C準拠のWeb Appを作成するため移植性が高くなります。ここではSimple UIをベースに少しだけ改良してみましょう。

注3) QML/Qt Quickについて詳しく知りたい場合は次の書籍がお勧めです。折戸孝行著『Qt Quickではじめるクロスプラットフォーム UIプログラミング』ISBN978-4-04-891512-0

▼図3 コード編集画面



プロジェクトの作成と実行

ウィザードでSimple UIを選択すると、プロジェクト名と保存先を設定する画面と、バージョン管理システムを選択する画面が出てきます。それぞれ設定したうえでFinishボタンを押せば、テンプレートの作成が完了です。

自動生成されるファイルのうち、QMLのコードを記述するのは「*.qml」ファイルです。「.qmlproject」ファイルはプロジェクト全体の設定を記述し、testsディレクトリとMakefileはおもにテストコードやそのビルドに使われます。「.desktop」はインストール後、メニューに追加される情報を記述したファイルです。

QMLファイルは2つあります。メインはSimple UI.qmlです(**図3**)。もう1つのcomponents/HelloComponent.qmlは、SimpleUI.qmlから使われるファイルです。具体的には**図3**の3行目の「import "components"」でcomponentsディレクトリのQMLファイルを読み込めるようにしたうえで、35行目の「HelloComponent{」において、HelloComponent.qmlの内容をHelloComponentエレメントとして取り込みます。そのため、まずはSimpleUI.qmlから解読していくと良いでしょう。

このテンプレートは実行するために必要なコードはすべてそろった状態で作成されています。そこでまずは実行して動作を確認してみましょう。ウィンドウ左下にある緑の再生ボタンを押すかCtrl+Rを入力することで、現在のプロジェクトのコードを実行できます。

「Hello..」と表示された矩形領域と「Tap me!」と書かれたボタンが表示されるはずです(**図4**)。ボタンを押すと矩形領域が「..world!」に変わります。ただそれだけのシンプルなアプリです。

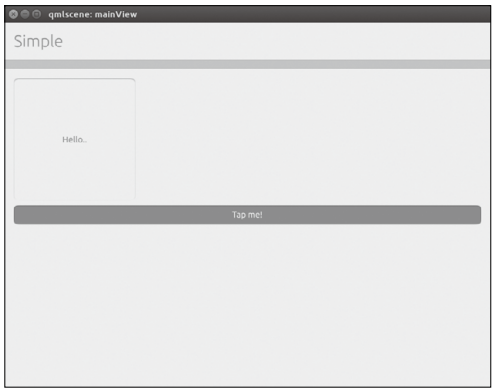
QMLの文法

QMLは、宣言型のJSONライクな文法を持ったUI構築用の言語であり、JavaScriptと組み合わせることで、より簡単にQtの豊富な機能を使用できます。ですからJSONとJavaScriptをある程度理解できれば、QMLのコードを読むことはそれほど難しくないでしょう。

コードの中でMainViewやPage、Columnなどは「エレメント」であり、UI部品やロジックのひとかたまりを構成する要素の単位となります[注4]。

注4) エレメントにカーソルを合わせてF1キーを押すとヘルプが表示されます。ESCキーを押すと編集画面に戻ります。QtCreatorのショートカットについては、「Tools > Options」のEnvironmentにあるKeyboardタブを参照してください。

▼図4 SimpleUIの実行



▼リスト1 図3から一部を抽出し、改変したQML

```
 1:HelloComponent {
 2:    id: label
 3:    objectName: "label"
 4:
 5:    // 3. tappedの値に応じて自動的にラベルの変更
 6:    text: button.tapped ? i18n.tr("..world!")
 : i18n.tr("Hello..")
 7:}
 8:Button {
 9:    id: button
10:    objectName: "button"
11:    width: parent.width
12:    property bool tapped: false // 1. プロパティの追加
13:    text: i18n.tr("Tap me!")
14:    onClicked: {
15:        // 2. クリックされるたび、tappedを反転
16:        tapped = !tapped
17:    }
18:}
```

個々のエレメントに対して、widthやheight、titleといったプロパティを設定し、onClickedといったイベントハンドラーを記述することで、アプリを構築していきます。

図3を一部変更した**リスト1**をご覧ください。たとえば「Tap me!」ボタンはButtonエレメントを使っており、その横幅は「parent.width」となっている(11行目)ことから親エレメント(Column)の横幅と同じということがわかります。さらにクリックされたとき(onClicked)に、HelloComponentエレメントの文字列(label.txt)を"...world!"に変更しています(6行目)。

このままでは、一度タップしたらラベルはもう変わらなくなるので、ボタンをトグル式にしてみましょう(**リスト1**)。まずタップの状態を保持するために、Buttonエレメントに新しいプロパティtappedを追加します(16行目)。クリックしたときはこのプロパティのみを変えるようにし、label.textはこのプロパティの値に応じて自動的に変わるようにします(6行目)。

Ctrl+Sで変更を保存したあと、Ctrl+Rで再実行してみましょう。今度はタップボタンがトグル式になっているはずです。

Ubuntuエミュレータの起動

Ubuntu SDKにはUbuntu Touchのエミュレータ機能も備わっています。そこで今回作ったアプリをこのエミュレータ上でも実行してみましょう。SDKのサイドペインにある「Devices」タブの「Create new emulator」でエミュレータを作成します。起動は「Start selected emulator」です。x86マシン上でarmhfのエミュレーションをするため作成も起動もその動作自体も実機よりはかなり緩慢になります。

起動が完了するとウェルカムスクリーンが表示されます。「Devices」タブ(**図5**)の下にある「DeviceAction」

から「Redetect devices」ボタンを押すと、開発に必要な設定がエミュレータに対して行われます。

さっそくSimple UIをエミュレータ上で動かしてみましょう。アプリメニューの「Build > Ubuntu > Run Application on Device」を選択するだけです(**図6**)。これだけで必要なファイルをコピーし、エミュレータ上でプログラムが実行されます(**図7**)。もちろんボタンを押せば、ローカルでテストしたときと同様にラベルが変わります。

アプリの終了は、同じメニューの「Close Application on Device」です。エミュレータはウィンドウにある終了ボタンで終了できます。

Ubuntu拡張の種類

メニューの「Ubuntu」は本家QtCreatorにはない、Ubuntuの拡張メニューです[注5]。この拡張メニューには、すでに見たデバイス上でのアプリの実行など、Ubuntu Touch開発において便利な機能が多く追加されています。

たとえば「Tools > Ubuntu > Showcase Gallery」はUbuntuテーマの各UIコンポーネントを実際に試すことができます(**図8**)。ギャラリー自体はただのQMLプロジェクトであり、メニューを選択するとプロジェクトが開きますので、実際のソースコードを確認することもできます。

ToolsのUbuntuには、ほかにもエミュレータや実デバイスを操作するためのメニューが存在します。SSHの接続、adbコマンドの実行、スクリーンショットの取得などは、逐一メニューをたどるよりはショートカットを覚えるか、Altキーをタップして HUDを駆使すると便利です。

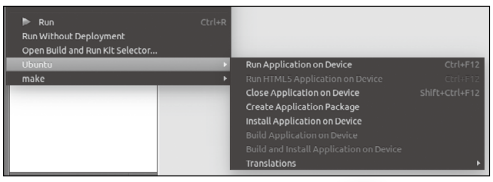
Ubuntu Touch

Ubuntu Touchはスマートフォン・タブレット向けにUIを調整したUbuntuの公式派生物の1つです。デスクトップのUnity 7に対して、Unity 8というバージョンを採用しているように、Ubuntu Touchの成果は将来的にデスクトップ版

注5) QtCreatorにはプラグイン機能があるため、QtCreatorのコードを変更することなくメニューや機能を追加できます。有効になっているプラグインは「Help > About Plugins」を参照してください。

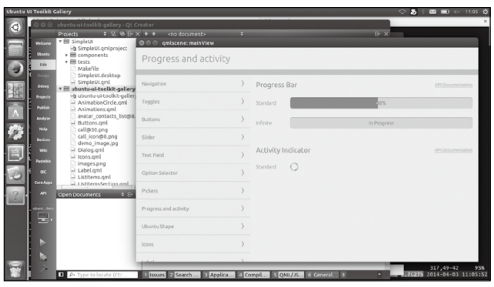
▼図5 Devicesタブ

▼図6 メニュー



▼図7 エミュレータ上での実行

▼図8 UIギャラリー



のUbuntuにも統合される予定になっています。Ubuntu Touch自体は13.04で開発プレビュー版、13.10で正式版がリリースされていましたが、いずれも不具合が多く、開発版のほうがましという状態でした。

14.04での変更点

Unity 8はQt5/QMLベースで開発を行っています。このQtのバージョンが、5.0から5.2に変更になりました。5.2はAndroid/iOSも正式に対応したバージョンであり、モバイル向けの機能も充実してきました。これまではUbuntu独自にモバイル向けAPIを実装していた部分も、Qtの機能を使うように部分的に移行しています。

Qt自体は本誌が発売されるころには5.3がリリースされる予定です。Ubuntu 14.04 LTSに5.3は取り込まれていませんが、14.10では5.3が採用されることになるでしょう。ただし、これまでQt5を使うのはUbuntu Touchだけだったのに対して、14.10ではKubuntuのQt5への移行が本格的に始まります。お互いの状況の擦り合せに時間がかかる場合は、さらに先のリリースに延期されるかもしれません。

Ubuntu Touchのカーネルや一部のサービスはAndroidのAOSP(Android Open Source Project)のコードを使用しています。14.04ではAndroid 4.4ベースのAOSPを使うようになりました。これによりサポートデバイスの見直しも行われています。Galaxy NexusとNexus 7(2012)は開発プラットフォームの対象から外し[注6]、Nexus 4がメインターゲットになりました。Nexus 7(2013)もテスト対象になっています。Androidのベースバージョンが上がったので、今後はNexus 5もテスト対象になる予定です。

細かいところでは、次世代ディスプレイサーバであるMirの不具合の修正とパフォーマンスの改善が行われ、スクリーンキャストの取得もできるようになりました(**図9**)。Mirの新APIに対応するため、Unity 8のSurfaceFlingerサポートが廃止されています。Unity 8はScopeシステムを一新し、13.10までのUbuntu Touchとは見た目ががらりと変わりました。QtWebKitに代わるバックエンドとして、Oxideの採用が進んでいます。

インストール方法

Ubuntu Touchのインストール方法も13.10からコマンド名も含めて若干変わっています。リリース前後でさらに変わる予定もあるので、詳しい手順は公式ページ[注7]を参照してください。

一般的な流れは、Android側で、必要なデータのバックアップをとり、ブートローダーアンロックを行い、最後にUSBデバッグを有効にします。その状態で、Ubuntu PC側からUbuntu Touchのインストールを実行します。ポイントとしてはUbuntuTouchインストール時に「channel」オプションを明示的に指定することです。channelを指定しない場合、14.04リリース版(stable)がインストールされますが、develの方がより新しく安定している場合があるためです。

注6) 14.04リリース直前まではイメージ自体は作成されていたので、インストールはできます。

注7) https://wiki.ubuntu.com/Touch/Install

▼図9 Unity 8/Mirのタスクスイッチャー

Ubuntu Touchをインストールすると、Androidのデータとリカバリー領域は削除されます。必要なデータは必ずブートローダーアンロックよりも前に取得してください。すでにデュアルブートを行うためにリカバリー領域を使っている場合は注意してください。ちなみに、開発者プレビューという位置付けではありますが、Ubuntu公式のデュアルブートインストーラも開発中です[注8]。

日本語化

Ubuntu TouchにはDroidフォントがインストールされていますが、いくつかの不具合[注9]によってそのままでは日本語が表示されません。そこで必要なパッケージをインストールしましょう。**リスト2**ではUbuntu Touchのルートファイルシステムを書き込み可能に変更してから、必要なパッケージをインストールしています。

最後にSystem Settingsを起動し、Language & TextのDisplay Languageから「日本語」を選択してください。再度再起動すれば、日本語化が完了です。ちなみに同じ画面のTime & DateからTimezoneを「Asia/Tokyo」に変更すると表示時刻が日本標準時に変わります。

なお、14.04リリース時点では日本語入力は開発中で、今のところ簡単に試す方法はありません。

その他のカスタマイズ

Ubuntu SDKとUbuntu Touchの連携方法はエミュレータと同じです。SDKで作成したアプリをTouch上で実行できますし、「SSH to Device(Ctrl-F10)」でSSH接続できます。

Nexus 4であればSIMも使えます。たとえばIIJmioのSIMで、音声通話やSMS、データ通信ができることを確認しました。まずSIMを挿入してデバイスを起動します。System SettingsのCellularでCarrierが正しく認識されていることを確認してください(IIJmioなら「NTT DOCOMO」)。この時点で音声通話は可能です。次に、デバイスをUSBケーブルで接続しadb shellからファイルを編集します(**リスト3**)。最後にrebootコマンドで再起動すれば、3Gでデータ通信ができるようになります。残念ながらNexus 4は日本のLTEに対応していないため3Gのみの通信になります。またNexus 7については2012も2013もWifiモデルのみの対応です。3G/LTEモデルにはWifiモデルのイメージをインストールして、Wifiのみで通信することになります。

Nexus 4や7は新モデルが出て、死蔵しているデバイスもあることでしょう。ぜひ、そんなデバイスにUbuntu Touchを入れて試してみてください。SD

注8) https://wiki.ubuntu.com/Touch/DualBootInstallation
注9) https://bugs.debian.org/661235

▼リスト2 パッケージのインストールバッチ

```
adb shell rm /userdata/.writable_image
adb reboot
adb shell apt-get install -y fonts-ipaexfont ➥
language-pack-{gnome-,}ja-base
adb shell apt-get purge -y fonts-arphic-ukai
adb shell rm /etc/fonts/conf.d/65-droid-sans-fonts.➥
conf
adb shell fc-cache -fv
adb reboot
```

▼リスト3 APN設定(値は各プロバイダの設定を確認してください)

```
$ adb shell
# editor /var/lib/ofono/(数字列)/gprs
[Settings]
Powered=true
RoamingAllowed=0

[context1]
Name=IIJmio
AccessPointName=iijmio.jp
Username=mio@iij
Password=iij
Type=internet
Protocol=ip
# reboot
```

Ubuntu 14.04の根本を支える

第6章 FoundationsとServer関連の新パッケージ

Ubuntu Japanese Team
吉田 史　YOSHIDA Fumihito　hito@kugutsu.org

本稿では、Ubuntuの根本を支える「Foundations」と「Server関連の新しいパッケージ」について解説します。

Foundations

Ubuntuでは、「Ubuntuというディストリビューション」の根本を支える要素を「Foundations」(旧称：Platform)と呼んでいます[注1]。これは、Launchpadのような基盤となるサーバだけでなく、アーカイブサーバやパッケージの構成／パッケージ管理システム／ツールチェインISOイメージのリリースに使われるサーバ群の準備や「universe」リポジトリのための自動コピーといった、さまざまな要素を含んでいます。カーネルや、Ubuntuのコア部分(最低限aptが実行できるようになる部分。この部分だけをまとめた「Ubuntu Core」という配布形態もあります)なども広義にはFoundationsに含まれています。

14.04の変更のうち、Foundationsに相当する部分を見ていきましょう。

注1) 「Foundations」とは別に、「Ubuntu財団」(Ubuntu Foundation)という組織も存在します。これはCanonicalと同じくMark Shuttleworthの出資で作られた財団で、「Canonicalが会社組織として破綻したあとも、Ubuntuの開発を継続できるようにするバックアップ」です。

カーネル

Ubuntu 14.04では、Linuxカーネル 3.13.8をベースとしたUbuntuカーネルが利用されています。リリース対象アーキテクチャはi386、amd64、armhf、arm64、ppc、ppc64、ppc64elの7種です。カーネルのフレーバーは次のとおりで、たいていのユーザはgenericを用いることになるでしょう(**図1**)。

- generic(汎用)
- lowlatency(Ubuntu Studioなどで利用される低レイテンシ特化版)
- generic-lpae(ARMのLPAE対応版)
- e500、e500mc、smp(PPC、PPC64のみ。各ハードウェア依存)

arm64は先行投資的なサポートで、ARM Foundation Modelエミュレータと Applied Micro X-Gene“Storm”を対象としたものです。X-Gene“Storm”のサポートはLSP[注2]を独自にマージする形でサ

▼図1　アーキテクチャとフレーバー

アーキテクチャ	フレーバー		
i386	generic	lowlatency	
amd64	generic	lowlatency	
armhf	generic	generic-lpae	
arm64	generic		
ppc	powerpc-e500	powerpc-e500mc	powerpc-smp
ppc64	powerpc64-emb	powerpc64-smp	
ppc64el	generic		

注2) Linux Support Package。ハードウェアベンダが対象ボードを動作させるためのカーネル差分をまとめたもので、まだLKMLのMainlineカーネルにマージされていないもの。

ポートしています。

カーネルの開発方針そのものはこれまでとは大きな差はなく、"LKMLのMainlineカーネルに必要なパッチを加えたもの"という設計思想です。主なパッチの内容は次のとおりです。

- AppArmor、AUFS、overlayfsの最新に近いリリースをマージ
- AppliedMicro X-Gene "Storm"のLSPをマージ("apm-storm")
- ドライバの更新。実ハードウェア以外にMicrosoft Hyper-Vのフレームバッファなどへの対応も含む

Trimへの対応

Ubuntuのユーザランド共通の機能として、SSDのためのtrimの自動発行に対応しました[注3]。この変更はutil-linuxパッケージの一部として提供されます。

trimは、「すでに使用しなくなった」領域をOSからSSDに伝えるためのATAコマンドです。現代的なSSDは、「使われなくなった」領域をコントローラにとって都合の良い場所に再配置することで性能を維持する特性を持っています。しかしSSD側からみると、「ある領域を使っているか否か」を判断する材料がないので、OS側にはtrimコマンドで「その領域は空きとみなして良い」ことを知らせる必要があります。

trimのLinuxでの一般的な実装はfstrimコマンドで、これを発行するとファイルシステム側の対応に基づいて、自動的に「そこは空きである」ということが通知されます。

Ubuntu 14.04 LTSでは、fstrimを用いた自動実行プログラム、「fstrim-all」が準備され、cronエントリ/etc/cron.weekly/fstrimから実行されるしくみとなっています。対応するファイルシステム(ext3、ext4、xfs、btrfs)であれば、週に一度、自動的にfstrimが発行されます。

ただしcronエントリにも記述されているとおり、現状ではホワイトリストに含まれるベンダのSSD[注4]でのみtrimが実施されます。これは、高I/O負荷時にtrimを発行すると異常な挙動(ストールしたりデータが消えたりする)を示すSSDが存在するためです。現在のホワイトリストは、Intel、Samsung、OCZ、SanDisk、Patriotです。

Python 3.4への移行

Python3パッケージのデフォルトが3.4系に更新されました。Python 3.3系は完全に削除されています。

これは、Python 3.3がセキュリティ更新のみに切り替わった[注5]ためです。予測はされていたものの、時期的にはぎりぎりの処置となり、「開発終盤である3月前半に、おもむろにsidやexperimentalにあるパッケージをそのままリビルドしてリポジトリに投入後、担当者が影響を受けたパッケージをすべて直す」[注6]という極端な方法が取られていますが、3月末の時点では大きな問題は見つかっていません。

この移行の根拠は、14.04 LTSが5年間サポートを継続しなくてはならないことと、Python 3.3が5年後までメンテナンスされなくなり、Ubuntu側で自力でメンテナンスしなくてはならない可能性があることです。端的には、"3.3のメンテナンスを継続するよりも、リスク含みではあるものの3.4に更新するほうが最終的なコストは減る"ということです。なお、さまざまな事情からPython 2.7からの完全な脱却は完了しておらず、Python 2.7パッケージも「universe」リポジトリ経由で提供されます[注7]。

注3) https://blueprints.launchpad.net/ubuntu/+spec/core-1311-ssd-trimming

注4) これは開発者の1人の「手元にあるSSDでテストしてみたよ」という報告を元にしたもので、「これで大丈夫そうだ」という報告があれば随時追加される予定です。ただし、報告者が型番指定で報告しているにもかかわらず、チェックルーチンはなぜかベンダ名で判別しており、「そもそもこれでいいのか、とくにIntelとOCZとPatriotは型番ごとにコントローラチップのOEM元が違う」といった問題が残っており、現状では「とりあえず実装してみた」レベルのものとなっています。

注5) https://mail.python.org/pipermail/python-dev/2014-March/133213.html

注6) https://bugs.launchpad.net/ubuntu/+source/python3.3/+bug/1295153参照。

注7) universe経由ですので、サポート期間は9ヵ月です。

■ Fastpath Installer

14.04では、Fastpath Installer(Curtin)の機能拡張も進められました。Curtinはdebian-installer(d-i)の代替実装の1つで、d-iとは異なり、パッケージを個別に展開してセットアップする(.dpkgをひとつひとつインストールしていく)代わりに、「展開済みファイルシステムのイメージをそのまま書き込むことができる」という特徴があります。Curtinの投入そのものは13.10の時点で行われており、今回は「そこからさらに機能を拡張する」というアプローチが取られています。

具体的には、使用するカーネルを任意に選択[注8]できる機能や、ハードウェア的なリブートではなく「kexecによる簡易再起動」「特定のボリュームは変更せずに済ませる」といった機能が追加されています。開発時点ではLVM、MD RAIDのサポートも追加される予定でしたが、今回は見送られています。

■ Upstart 1.12

Ubuntuの採用するinitシステム[注9]、Upstartは14.04では大きな変更はありません。重篤なバグ(IPv6だけを利用するプロセスのソケットを検知できない、DBusが動作していない環境ではtelinitが機能しない[注10]など)の修正が行われたものの、13.10からの大きな変更はありません。

12.04との比較では、「ユーザ権限のプロセスをinitのしくみで管理する」Upstart User Session[注11]が投入されている、という点が最大の違いです。これはUpstartそのものをユーザ権限で動作させ(= PID 1としてではなく、任意のサブプロセスとして起動し)、各種プロセスの自動起動・spawn管理(停止を検知して自動的に再起動)を行います。

Server関連の新しいパッケージ

■ Apache HTTPd 2.4と「It works!」ページの差し替え

Apache httpdパッケージが更新され、Apache httpd 2.4系がデフォルトとなりました。2.4系への移行とともに、「It works!」とだけ表示されるそっけないインデックスページから、Apacheの操作について説明したビジュアルなページに置き換えられています。

これはDebian側のデフォルトページ更新を受けたもので、バグ情報ページやアイコンがUbuntuオリジナルのものに差し替えられています(**図2**)。

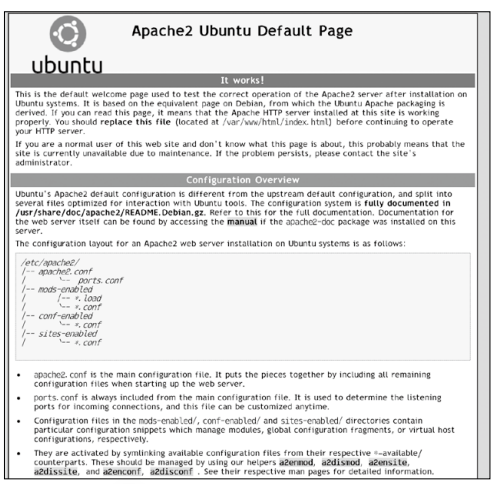
注8) 現在のUbuntuは、「HWE」と呼ばれる追加パッケージを用いることで、リリース後にカーネルのメジャーバージョンを変更できます。通常、LTSには「そのリリースの次のLTSまで」(14.04 LTSであれば16.04 LTSまでの、14.10、15.04、15.10、16.04 LTSの4種)のカーネルが提供されます。

注9) systemdへの移行も検討されていますが、少なくとも14.04世代では継続してUpstartが利用されています。また、現在のUbuntuの各種機能はUpstart(とくにUser Session)に強く依存する形で実現されており、移行は簡単な話ではありません。

注10) Upstartは伝統的なinitシステムへの互換性を重視しているため、昔ながらのtelinitコマンド(ランレベル切り替え)をエミュレートする機能が搭載されています。……が、この機能がDBusなしでは動作していなかった、というバグです。ただし、Ubuntuは基本的にランレベル1と2と6だけで動作しており、Red Hat的な「ランレベル3がコンソールマルチユーザ、ランレベル5がグラフィカルマルチユーザ」というような使い分けをしていなかったこともあり、しばらく誰にも気づかれていなかった、というオチもついています。

注11) この変更はUpstart 1.7で投入されています(https://launchpad.net/upstart/+milestone/1.7)。なおUpstartはUnixの伝統的なバージョン付与モデルを採用しており、1.9→1.10→1.11→1.12、というバージョン履歴を持ちます。

▼図2 ビジュアル化されたApacheのインデックスページ



■ Nginx

14.04 LTSから、Apacheの代替選択肢の1つ、Nginxが「main」リポジトリから提供されるようになりました。mainに含まれるようになると、Ubuntu/Canonicalが組織として責任を持った状態で、当該リリースのサポート期間の間はセキュリティ脆弱性・重篤なバグへの対応が行われるようになります(併せて、セキュリティ的な問題が見つかった場合、Ubuntu Security Notice(USN)が発行されます)。

Nginxはこれまでこうしたサポートのない「universe」リポジトリから提供されていましたが、14.04では「main」の一部として、5年の間、適切な形でサポートが提供される予定です。ただし、Seedではありません[注12]。「Ubuntu Server」としてのISOイメージにはNginxは含まれておらず、インストール後にapt-getを用いて導入する必要があります。

■ MySQLとMariaDB

これまでのMySQLパッケージ(mysql)は、Oracleがリリースする MySQLを「セキュリティアップデートが含まれるリリースのみ」パッケージングしてリリースするモデルを採用していましたが、14.04ではMRE[注13]が認められ、「すべてのマイナーバージョン」を取り込む形に変更されました。

バージョン選定としては(Pythonなどに比べると)おとなしく、5.5系が「main」リポジトリに、そして新メジャーリリースである5.6が「universe」リポジトリに投入されています。5.6系への乗り換えは「14.04 LTSリリース時点」では見送られていますが、ライフサイクルのどこか(≒ポイントリリースのいずれか)で、MySQL 5.6系への切り替えが行われる予定です。これは、MySQL 5.5のExtendet Support期間が2018年12月までと、14.04のサポート期限である2019年4月までに比べるとわずかに短いこと、また、現時点ではまだMySQL 5.6が「十分に枯れた」状態になっていないこと、5.5から5.6への切り替えに大きな困難はなさそうであることによります。

なお、MySQL「互換」のDBとしてしばしば名前の挙がる、MariaDBとPercona Serverもuniverseながら14.04に取り込まれています。

■ QEMU2.0

4月序盤(リリースまで約2週間)の時点で「Upstreamから正式なリリースが行われたら14.04に投入する予定だ」という宣言が行われており、緊張感の漂う状態となっています。

「リリースされたら14.04に」とはいえ、PPA(Personal Package Archive)によるテストリポジトリでQEMU 2.0枝のリリース直前バージョンのテストは行われており、導入しても惨事は起こらないであろう、という状態で作業が進められています。しかし、「通常であればリリース候補版が出ている時期に、まだソフトウェアが全部そろっていない」という段階にあります[注14]。

■ LXC 1.0x

Ubuntu/Canonicalが主要開発者を送り込んでいる仮想化ツール(というかコンテナ実装)として、LXCがあります。LXCはcgroupを利用したユーザプロセスの分離とchroot環境を組み合わせ、親マシンと共通するカーネルを利用した仮想マシンの一種です。Solarisなどで実現されているコンテナに近い実装です。lxc-createで仮想マシンを作成し、lxc-runでマシン実行を行います。

注12) Ubuntuの各種CD/DVDの生成には、「Seed」と呼ばれるパッケージリストが用いられます。各種Seedは、http://people.canonical.com/~ubuntu-archive/seeds/から確認できます。14.04 ServerのSeedは、http://people.canonical.com/~ubuntu-archive/seeds/ubuntu.trusty/server-shipです。

注13) MRE(Micro Release Exception)は「非常に速い速度でUpstreamのリリースが行われる」ソフトウェアに対して実施される例外措置で、「リリース後はセキュリティアップデートと重篤なバグの修正しか行わない」(= Micro Release Policy)を無視し、Upstreamのリリースをそのままリリース後にも取り込む、というものです。FirefoxやThunderbird・OpenStackに対して実施されています。

注14) この段階ではQEMU 2.0RC1(qemu-2.0.0~rc1+dfsg-0ubuntu1)がリポジトリに投入されていました。

4月頭の時点では、14.04には1.0.2が投入されており、また、1.0.x系ではコマンドの非互換な変更などは予定されていないため、実用的に利用できる環境になったと言えるでしょう。

これまでのLXCはコマンドそのものが日常的に変更されていく、オプションが正常に通らない、といった事情から、Ubuntuのアーカイブに含まれるものを使うのは少々厳しく、PPA（ppa:ubuntu-lxc/stable）にあるものを利用する必要がありました。

前述のとおりCanonical社員がLXCの開発において中心的な役割を果たしていること、そしてLXCをそのまま使うだけでなく、AppArmorを用いて「子から親へ」のアクセスを制約できることから、UbuntuはLXCを使うのに適した環境だと言えるでしょう。

■ OpenStack “Icehouse”

14.04のOpenStackは、14.04と同日4月17日にリリースされる予定のIcehouse（2014.1）が搭載されます。4月序盤の時点ではRC1ベースのパッケージが投入されていますが、OpenStack関連はMREの対象となっているため、2014.1リリース後に更新される予定です。

■ UVTool（Ubuntu Virtualization Tools）

Ubuntu Serverには、「難しい操作を簡単に行えるようにする」ための独自のユーザランドツールが準備されています。たとえば、byobu（TmuxやGNU Screenのラッパーソフトウェア）やrun-one（cronなどで利用できる、「指定されたプロセスを1個だけ」起動するためのラッパー）、UFW（Uncomplicated Firewall。iptablesのラッパー）などが該当します。

こうしたラッパーコマンドとして、libvirtとKVMをベースとした「簡単な仮想マシン管理ユーティリティ」となるUVTool（UVT。Ubuntu Virtualization Tools）が加わりました。

UVTはUbuntu/Canonicalがリリースしている「Cloud Images」[注15]を自動的にダウンロードし、KVM仮想マシンとして起動するためのラッパーです。将来的にはKVM以外のハイパーバイザをサポートする予定が述べられていますが（とくにLXC）、現状では実装されているコマンドはvvt-kvmのみです。もともとはUbuntu 13.10の時点で投入されたツールですが、13.10時点では事実上そのままでは使いものにならなかったこともあり、14.04でのお披露目となっています。

UVTの利用方法

UVTはapt-getに近い考え方のツールで、「仮想マシンデータのダウンロード」「仮想マシンの生成」「仮想マシンへの接続」といった機能を持っています。

仮想マシンイメージのダウンロードには「uvt-simplestreams-libvirt」コマンドに“sync”オプションを与えて実行します。確保済みのイメージは、“query”オプションで得られます。これによりCloud Imageの構成ファイルがダウンロードされ、KVMで利用できる形式にコンバートされます。実ファイルは/var/lib/uvtool/libvirt/以下に配置されます。なんらかの特殊な理由でファイルを破棄したい場合、“purge”オプションを用います。

マシンイメージを入手したら、「uvt-kvm」コマンドを用いてマシンを生成します。“create”オプションを与えることでマシンを起動できます。起動したマシンへの接続には“ssh”オプションが利用できます。Cloud Images由来の仮想マシンを用いる場合、初期ユーザは“ubuntu”、ログインに利用するSSH鍵はUVTを実行するマシンのホームディレクトリ上のものが暗黙で利用されます。

マシンが生成されたら、「uvt-kvm ssh test1 --insecure」とすることで接続できます（初回の

注15）Ubuntuでは、ISOイメージによるリリースに加えて、Amazon EC2やLXC・Vagrantなど環境で簡単に利用できる、「Cloud Images」というリリースを準備しています（http://cloud-images.ubuntu.com/）。HTTPでダウンロード可能なファイルとしてだけでなく、「ボタン1つでEC2仮想マシンが立ち上がる」というしくみも準備されています。

みSSHホスト鍵の確認を省略するためinsecureオプションが必要です。KVM環境との接続にIP偽装やネットワーク盗聴の可能性はあまり考えられないのですが、見た目的に非常に美しくないので将来的には改善されるでしょう)。マシンが不要になったら、「uvt-kvm destroy」で削除できます。基本的なコマンドの利用方法は、**図3**を参照してください。

uvt-kvm create時に--memory、--disk、--cpuといったオプションを与えることで、利用するメモリ・ディスク容量・CPU数を変更できます。それぞれデフォルトでは512(MB)、8(GB)、1コアです。

また、create時に--user-data、--meta-dataオプションを用いると、cloud-initを用いた初期設定も可能になります。デフォルトのイメージがCloud Images由来で、EC2上のUbuntuと同じファイル構成となっていることもあり、「手元にあるEC2マシン」として気軽に扱うことができます(ただし、EC2はXen、UVTは現状ではKVMベースという違いがあります)。一種の実験ホストとして便利に利用できるでしょう。

UVTの機能は、基本的にはlibvirtのvirshでできることと同じです。「Cloud Imagesを自動的にダウンロードし、KVM環境で使えるようにコンバートする」という処理を肩代わりしてくれるため、イメージの元ファイルの管理を意識しなくて済むようになります。現時点ではまだ開発が継続されていることもあり、universeリポジトリからの提供となっています。

▼図3　UVTの使い方

・仮想マシンイメージの操作(uvt-simplestreams-libvirt)

※デフォルト指定でのダウンロード

```
uvt-simplestreams-libvirt sync
```

※アーキテクチャを指定しない場合、暗黙でi386、armhf、amd64版がダウンロードされます。通常必要になるのはどれか1つだけですので、なるべく指定するようにしたほうが良いでしょう。たとえば、14.04のamd64アーキテクチャのみをダウンロードするには次のように指定します。

```
uvt-simplestreams-libvirt sync release=trusty arch=amd64
```

※ダウンロード済みイメージを一覧

```
uvt-simplestreams-libvirt query
```

・仮想マシンの操作(uvt-kvm)

※KVM仮想マシンの生成

```
uvt-kvm create test1 release=trusty arch=amd64
```

※稼働中のマシンのリスト

```
uvt-kvm list
```

※IPアドレスの確認

```
uvt-kvm ip test1
```

※KVM仮想マシンの起動

```
uvt-kvm create test1 release=trusty arch=amd64
```

■Juju

近年のUbuntu Serverを代表する、自動化、オーケストレーションツールとしてJujuがあります。Jujuは、Canonicalが開発する「複数のサーバを自動的に設定するだけでなく、さらにサーバ間の動作も定義する」自動化ツールです。

たとえば、WordPressとApache・MySQLで構成されたWebサイトを考えてみましょう。古典的な自動化ツール・デプロイツール(いわゆる冪等性(べきとうせい)を担保するためのフレームワーク)の類では「このホストAにはMySQLが導入されている」「MySQLの設定はこうなっている」「このサーバにはApacheとPHPが導入されていて、ホストBのMySQLを参照している」といった定義を積み重ねることで設定を行います。

Jujuもこうしたツールと同じように設定を行いますが、単に「MySQLの設定はこうなっている」といったことを定義するのではなく、「このホストの役割はMySQLサーバである」「このホストの役割はWordPressフロントエンドで

ある」といった、ホストごとに役割を指定する形で設定します。かつ、「このMySQLサーバノードはこのWordPressフロントエンドから利用される」という関係性(relation)を与えて接続する、という使い方をします。また、マシンの起動も含めて自動的に実行されます。後発であるがゆえに、十分に抽象化された自動化ツールとして利用できる、という点がメリットです。JujuにはWebUIもあり、ブラウザからサーバを起動し、関係性を定義するだけで複雑なサーバ群をセットアップできるようになっています(あるいは、「できるようになるという理想のもとに開発が進められていますが、今のところお仕着せの構成をなぞることぐらいは可能、というレベルで未完成です」)。

14.04世代のJujuは13.10からは大きな変化はありませんが、12.04時代に比べると設定ファイル(Charm)コレクションを保存するJuju Charm Storeの充実、そして、複数のJuju制御のマシンを組み合わせた「Bundle」がサポートされ、(Canonical側で準備したものと完全に一致する構成であれば、という前提はつくものの)コマンド1つでHadoopやOpenStackのような、複雑な構成のクラスタ環境を構成できるようになっています。

■ MAAS 1.5

Ubuntu Serverには、MAAS(Metal As a Service)と名付けられた、「物理マシンをクラウドサービスのように扱える」ソフトウェアが準備されています。

これは、IPMIやマジックパケットによる電源投入、PXEブートによるOSの自動インストールを組み合わせて、ec2-run-instanceと同じように、1コマンドで「Ubuntuがセットアップされたマシン」を生成できるようにする、というプロジェクトです。物理マシンを一度MAASサーバに登録しておくと、WebUIやCLIからコマンドを実行するだけで自動的に電源が投入されたうえでOSがインストールされ、SSHできるようになります。

また、物理マシンを使うだけでなく、仮想マシンを配下に置くこともできます。

こうしてセットアップしたマシンはユーザが個別にセットアップすることもできますし、cloud-initによる自動設定を用いることも、Jujuを用いて抽象化した状態で扱うこともできます。

MAASそのものの初投入は12.04時点で、当時はCobblerのラッパーとしての機能しか持っていませんでした。14.04のMAASは基本路線は13.10のものと変化なく、LXC環境でスムーズに動作するようになったほかは、MAASそのものの目新しい機能はありません。前述のCurtinによりインストールが高速化されていること、また、Ubuntuの、とくにOpenStack関連での各種デモンストレーションでしばしば利用されるようになったため、致命的なバグがなくなったことがメリットです。OpenStackのテスト環境として、Jujuとともに簡単に利用する程度であれば、大きな問題はないでしょう。ただし、いわゆる本番環境を担うには少々厳しいものがあります。

これとは別に、MAASやクラウド環境に向けて、マシンイメージを準備するためのラッパーとして、「cloud-installer」と呼ばれるラッパーソフトウェアも準備されています。ただし、4月序盤の時点ではまだ正しくデバッグされていないこともあり、「マニュアルはおろかmanページすらない」「コマンドの一部は正常に動作しないどころかPythonのimport errorが出て終了する」という状態です。4月17日のリリースを控えて、なぜか現状でも活発に開発が続けられている、という状態にあり、操作的な面でも正しい使い方は定かではない(今のところの動かし方らしきものはわかるものの、リリース版でどうなるかは定かではない)ので、現時点では「こういう名前のものが準備されている」というレベルにとどめておきます。SD

GDK先取り解説

どうなってる？「Google Glass」のアプリケーション開発

※Google Glassは現在、米国内のみでの試験運用を行っている段階です。
本記事の内容は、将来市販される商品に対して何の保証もされないことをご承知おきください。

タイトル写真提供：Google

Google発のウェアラブルデバイス「Google Glass」。いろいろな面で注目を集めているのでご存じの方も多いでしょう。昨年11月にはこのGoogle Glassに特化したAndroidアプリを開発できるGDK（Glass Development Kit）が発表されました。本稿では、Google Glassのアプリのしくみと GDKによる開発の概要を中心に現状を紹介しましょう。

有山 圭二（ありやま けいじ）（有）シーリス 代表　http://www.c-lis.co.jp

GDKにより開発環境が整備

2013年5月にGoogle Glassが発売されてから、1年が経ちます。この1年間、Google Glassはアップデートによる機能強化を繰り返してきました。また、昨年11月にはGoogle Glassに特化したAndroidアプリを開発できるGDK（Glass Development Kit）を発表しています。本稿では、6月のGoogle I/Oで何らかの発表があると思われるGoogle Glass（以下、Glass）の最新事情と、Glass用のアプリケーション（以下、アプリ）開発について解説します。

とはいえ、ほとんどの読者の方がGlassをお持ちではないでしょうから、アプリ開発といってもピンとこないかもしれません。しかし、後述するようにGoogleはGlassに続いてAndroid Wearを発表し、ウェアラブルの流れは加速しています。ウェアラブルが普及すれば、アプリ開発者としてはさまざまな要求に対応する必要があるでしょう。その準備が早いに越したことはありません。新しいパラダイムのプラットフォームに自分のアイデアをどう注ぎ込むか、イメージするのに役立ててもらえれば幸いです。

Google Glass概要

Glassは、2012年5月にGoogleが発表した眼鏡型のデバイスで、身につけて使用する「ウェアラブルデバイス」に分類されます。Glassは、スマートフォンと同じくAndroidで動作しています。搭載しているAndroidのバージョンは4.4.2（KitKat）です。

▸▸ Glassのインターフェース

Glassの操作方法は、一般的なスマートフォンとは異なる部分が多くあります。

タッチパッド

スマートフォンは通常、画面がタッチパネルになっているので、画面を直接触って操作します（**図1**）。表示に対して指をあてるため狙いが定めやすく、細かい操作が可能です。

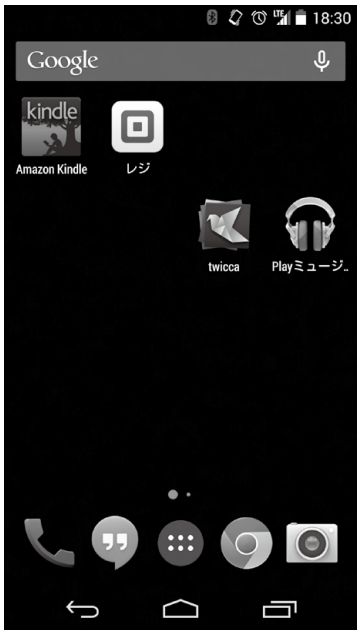
一方、Glassはおもに側面のタッチパッドで操作します（**写真1**）。表示を直接触らないので、画面の中で狙った場所をタップするという操作はありません。

タッチ操作は、前方向へのスワイプ（スワイプ・フォワード）、後方向へのスワイプ（スワイプ・バック）、下方向へのスワイプ（スワイプ・ダウン）の3種類で、これらにタップを組み合わせて操作します。

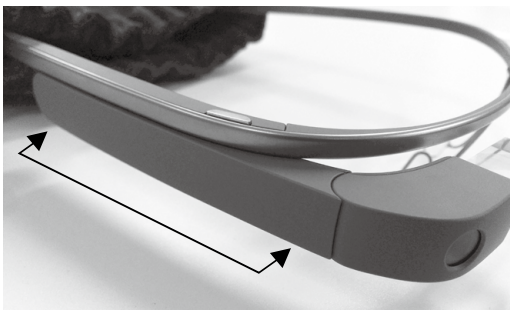
音声入力

Glassは、側面タッチパッドに加えて音声入力で操作できます。画面下部に“ok glass”と表

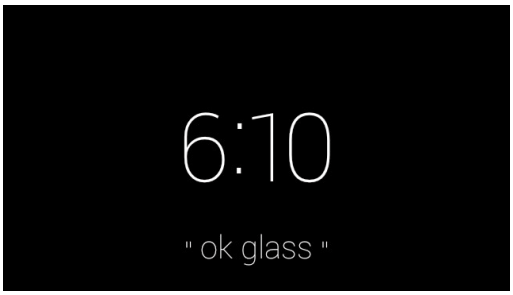
▼図1　画面を見ながら操作 できる スマートフォン



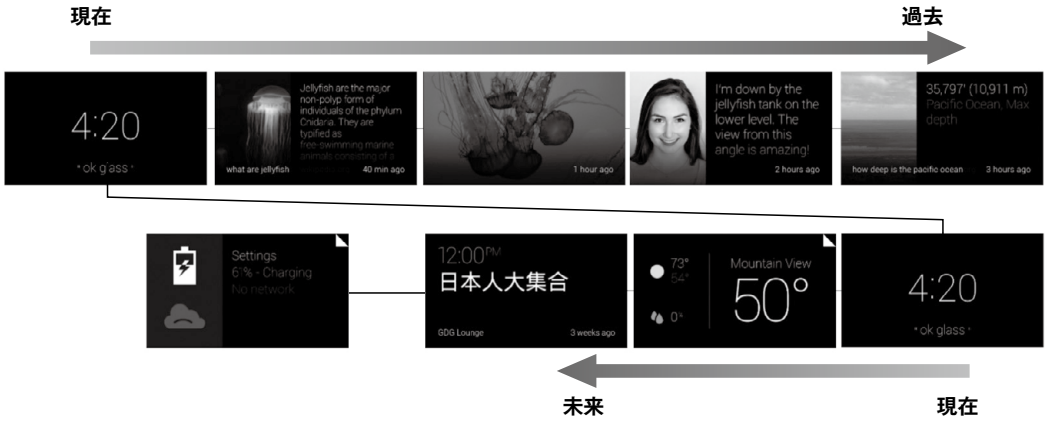
▼写真1　側面がタッチパッドになっているGlass



▼図2　ok glass



▼図3　タイムライン



示されているとき(**図2**)に、「OK Glass」と発声すると、音声による命令(ボイスコマンド)の入力を開始します。

タイムライン

Glassは、カード型の画面インターフェースを備えているのが特徴です。カードは、時計の画面から左右に広がっています(**図3**)。時計の現在時刻表示を起点に、右側にはGlassで撮影した写真や動画、受信したメッセージなど**過去の情報**が、左側には天気予報や今後のスケジュールなど**未来の情報**を表示します。

このように複数のカードを並べたものをGlassでは「タイムライン」と呼びます。

タイムラインを表示しているのは、Glassのホーム画面である「Glass Home」です。これは、通常のスマートフォンのホーム(ランチャ)アプリに相当し、ボイスコマンドの受付、撮影した写真の閲覧や操作、そのほかすべての情報に、このGlass Homeからアクセスします。

また、後述するGlassアプリからタイムラインにカードを追加したり、カードの操作を受けて処理を実行することもできます。

▶▶アップデート

Glassは発売から昨年12月まで、月一度のペースでアップデートされてきました。本稿執筆時点での最新版は、今年(2014年)4月にリリースされた「XE16.11」です(**図4**)。

XE16より前のバージョンは4.0.3(Ice Cream Sandwich)ベースでしたが、XE16からは最新のAndroid 4.4.2(KitKat)ベースとなりました。

最新のAndroidが搭載されることで、それまではソフトウェアが対応していなかったBLE(Bluetooth Low Energy)が利用できるなど、もともとのハードウェア性能をフル活用できるようになっています。

Glassware開発

最初にGlassのしくみをつかんでもらいましたが、現時点でアプリ開発者には何ができるのでしょうか。ここからはそれを紹介していきましょう。

本稿では、Glassで動作するアプリケーション全般を「Glassware」と呼ぶことにします。Glasswareを開発するには、大きく2つの方法があります。

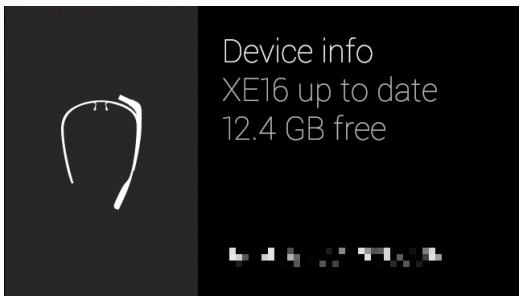
1つは、**Mirror APIを使って開発する方法**。もう1つは、**GDKを使ったAndroidアプリとして開発する方法**です。

Mirror APIは、Googleが公開しているサーバとやりとりすることで、間接的にGlassと情報を送受信するしくみです。Mirror APIを使う場合、ユーザは基本的にWebから認証をするだけで利用を開始できますが、Glassは必ずインターネットに接続している(オンライン)必要があります。

一方、GDKを使ったAndroidアプリとして開発する場合、GlasswareはGlass上で動作します。したがってオンライン/オフライン両方で動作するアプリを開発できますが、配布の手段が限定されています。現状では、原則としてAPK[注1]ファイルをadb(Android Debug Bridge)でインストールする必要があります。

それぞれの違いを簡単に**表1**にまとめます。

▼図4　Glass XE16



注1) APK(Android application PacKage file):Androidアプリの配布パッケージ。

▼表1　Mirror APIとGDK

種類	インターネット接続の要・不要	配布方法
Mirror API	必要	Webから認証
Androidアプリ(GDK)	不要	apkを配布

● column ●

技適の問題

日本におけるGoogle Glassは、技術的には使用できますが、法律の壁があります。

日本で電波を発信するには、製品に「技術基準適合証明等のマーク(技適マーク)」(**図A**)を表示する必要がありますが、Glassには技適マークがありません。

したがって日本国内では、GlassからWi-Fi、Bluetoothといった電波を発信することは違法になります。そのため原則として、Glassを日本国内で使用するにはAirplaneモードに設定する必要があります。

どうしてもWi-FiやBluetoothなどの電波を出す必要がある場合には、「電波暗室」と呼ばれる、シールドされて電波を外に漏らさない特別な施設の中で行います。

▶図A
技術基準適合証明等のマーク(技適マーク)

▸▸Mirror API

Mirror APIを使うと、Googleのサーバを経由してユーザのGlassへ情報を配信したり、逆にGlassからの情報を受け取ることができます(**図5**)。

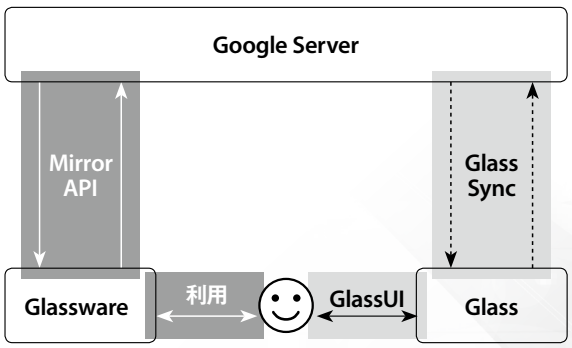
GlassはGoogleのサーバとバックグラウンドで通信して、定期的に同期しています。同期自体は自動的に行っているので、通常、ユーザが意識することはありません。

Mirror APIは、OAuth 2.0のしくみで認証して利用するRESTfulなAPIです。したがって、Mirror APIを利用するにはGoogleにあらかじめサーバ情報を登録して、認証情報(クライアントIDとクライアント・シークレット)を受け取る必要があります。

そのうえで、ユーザのGlassにアクセスする権限をGoogleに要求(リクエスト)します。Glasswareからのリクエストに対してユーザが許可をした場合、ユーザのGlassにアクセスできるようになります。

Mirror APIは、必ずGoogleのサーバを経由して情報を送受信することになりますが、左ページのコラムで述べているとおり、Glassは日本の技適を取得していないので、Wi-Fi/Bluetoothを使った通信ができません。そのため本稿執筆時点では、あまり現実的な方法ではありません。日本の技適を通ったGlassの発売が待ち遠しいですね。

▼図5　Mirror APIを使ったアプリのしくみ



▸▸Androidアプリ(GDK)

GlassはAndroidで動作しているので、スマートフォン向けのAndroidアプリをインストールできます。また、GDKを使えば、Glass固有の機能を利用することもできます(**図6**)。

インターネット接続を必要としないアプリであれば、Wi-Fi/Bluetoothを使わず、スタンドアローンで動作することも可能です。日本国内でGlassを入手した開発者にとって、昨年11月のGDKリリースは待ちに待ったものだったと言えるでしょう。本稿では、こちらを詳しく解説します。

GDKによるGlassware開発

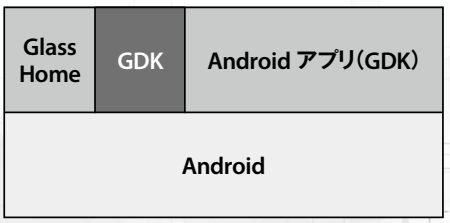
GDKを使ったアプリ開発の流れをざっと解説しましょう。ここでは、開発環境としてAndroid Studio 0.5.5を使います。

▸▸必要な準備

通常のAndroidアプリとしてGlasswareを開発するのであれば、通常のAndroidアプリ開発の準備以外はとくに必要ありません。しかし、Glass特有の機能を利用する場合は、あらかじめGDKライブラリ「Glass Development Kit Preview」のダウンロードと、GDKを利用するプロジェクト設定を**表2**のようにして作成する必要があります。当然、Glass特有の機能を使った場合、通常のAndroidスマートフォンでは動作しないので注意してください。

なお、紙幅の都合上詳細な設定方法やサンプルリストは掲載できませんが、本誌サ

▼図6　Glass Development Kit



▼表2　プロジェクト設定

項目	値
Application name	GdkSample
Package name	jp.co.c_lis.gdksample
Minimum required SDK	API 19: Android 4.4.2(KitKat)
Target SDK	API 19: Android 4.4.2(KitKat)
Compile with	API 19: Android 4.4.2(KitKat)

ポートサイト[注2]からダウンロードできますので、興味のある方はそちらをご覧ください。

Glass Homeからの起動

この状態で作成したアプリはGDKの機能を使うことができますが、インストールしたアプリをGlass Homeから起動することはできません。起動ボイスコマンドを設定することで、Glass Homeからアプリを起動できるようになります。

まずはじめに、「app/src/main/res/xml」以下に、ボイスコマンドを設定するXMLファイルを作成します。今回は、「app/src/main/res/xml/voice_trigger.xml」としました。

リスト1の`hello glassware`が、ボイスコマンド本体です。本稿執筆時点では、ボイスコマンドは英語のみに対応しています。また、この文字列は、「strings.xml」など別のリソースに分離することもできます。

次に、指定したボイストリガーのXMLを、起動したいアクティビティに関連づけます。これは「AndroidManifest.xml」に指定します。

リスト2では、アクティビティ`jp.co.c_lis.gdksample.app.MainActivity`の`intent-filter`を変更して、`action`に`com.google.android.glass.action.VOICE_TRIGGER`を指定しています。また、ボイストリガーの設定として、先ほど作成した「app/src/main/res/

注2）http://gihyo.jp/magazine/SD/archive/2014/201406/support

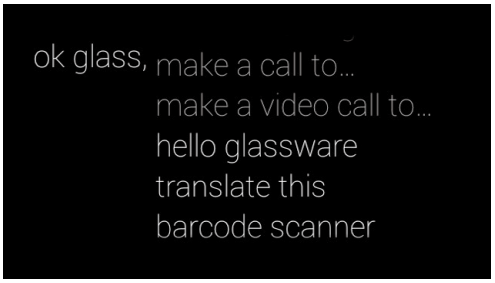
▼リスト1　app/src/main/res/xml/voice_trigger.xml

```
<? xml version="1.0" encoding="utf-8"? >
<trigger keyword="hello glassware">
</trigger>
```

▼リスト2　app/src/main/AndroidManifest.xml

```
<? xml version="1.0" encoding="utf-8"? >
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.co.c_lis.gdksample.app">

    <uses-permission android:name="com.google.android.glass.permission.DEVELOPMENT" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name="jp.co.c_lis.gdksample.app.MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="com.google.android.glass.action.VOICE_TRIGGER" />
            </intent-filter>
            <meta-data android:name="com.google.android.glass.VoiceTrigger"
                android:resource="@xml/voice_trigger" />
        </activity>

        <uses-library
            android:name="com.google.android.glass"
            android:required="true" />
    </application>

</manifest>
```

▼図7　hello glasswareのコマンドが追加されている



xml/voice_trigger.xml」を指定しています。さらにXE16からは、開発用のアプリがボイストリガーを登録するための新しいパーミッション**com.google.android.glass.permission.DEVELOPMENT**が必要です。

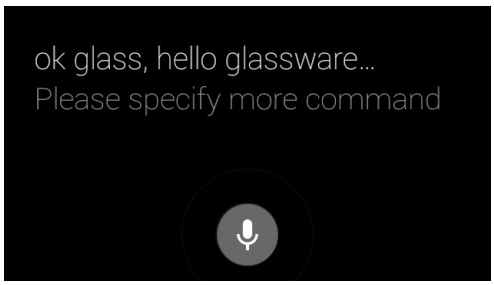
この状態のアプリをインストールすると、「ok glass」で表示されるボイスコマンドの選択一覧に「hello glassware」が表示されます(**図7**)。

なお、ボイストリガーは、アクティビティだけでなくサービス(Service)にも適用できます。AndroidManifest.xml で Service の **intent-filter** にボイスコマンドを設定すれば、ボイスコマンドから画面を表示することなく何らかの処理を実行できます。

サブコマンド

Glass Homeからアクティビティを呼び出す前に、追加で音声認識をして、サブコマンドとしてアクティビティに渡すことができます。

▼図8　サブコマンドの音声認識画面



▼リスト3　app/src/main/res/xml/voice_trigger.xml

```
<? xml version="1.0" encoding="utf-8"? >
<trigger keyword="hello glassware">
    <input prompt="Please specify more
command" />
</trigger>
```

リスト3のように、**trigger**の中に**input**を指定すると、アクティビティを起動する前に音声認識画面が表示されます。その際、**prompt**で指定している文字列が表示されます(**図8**)。ただし、サブコマンドの認識にはインターネットへの接続が必要です。インターネット接続がない状態ではエラーが表示されて認識ができません。

正しく音声認識がされた結果を、アクティビティで受け取ることができます。**リスト4**のように、サブコマンドとして認識したワードは、**RecognizerIntent.EXTRA_RESULTS**に文字列リストの形式で渡されます。

サブコマンドを受けとることで、アクティビ

▼リスト4　MainActivity.java

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ArrayList<String> voiceResults = getIntent().getExtras()
                .getStringArrayList(RecognizerIntent.EXTRA_RESULTS);

        // 認識結果を画面に表示
        TextView tv = new TextView(this);
        tv.setText(voiceResults.toString());
        setContentView(tv);
    }
    …(省略)…
}
```

ティは起動直後に目的に合った処理を実行できます。またボイスコマンドと同様に、サブコマンドはアクティビティだけでなく、サービス(Service)にも適用できます。

タイムライン

タイムラインに表示するカードをGlasswareから操作できます。XE16では、それまでカードの操作に使われていたTimelineManagerが廃止されました。また、以前はStatic Cardという静的なカードをタイムラインに表示する方法が用意されていたのですが、そちらもXE16では削除されています。

Live Card

タイムラインに表示してから内容に変化があるカードです。Live Cardにはさらに、描画スピードによって2つの方法が用意されています。

1つは、RemoteViewsを使う方法です(**図9**)。Androidの通知(Notification)のように、Live CardにRemoteViewsを設定して表示します。描画速度は数秒に1回程度です。速度としてはあまり速くありませんが、手軽に扱えるというメリットがあります。

もう1つが、SurfaceViewを使う方法です(**図10**)。また、XE16では、OpenGL ES 2.0での描画に対応し、さらに高速なレンダリングが可能になっています。

タッチパッド

アクティビティでタッチパッドのイベントを受け取る方法も用意されています。基本的にGlassは、タッチパッドのイベントをD-pad Key(方向キー)が押されたときのイベントとしてアクティビティに伝えます。タップしたイベントはonKeyDownのKEYCODE_DPAD_CENTERとして処理します[注3]。また、下方向へのスワイプはAndroid端末のバックキーを押した際のイベント(KEYCODE_BACK)として処理します。

ジェスチャ

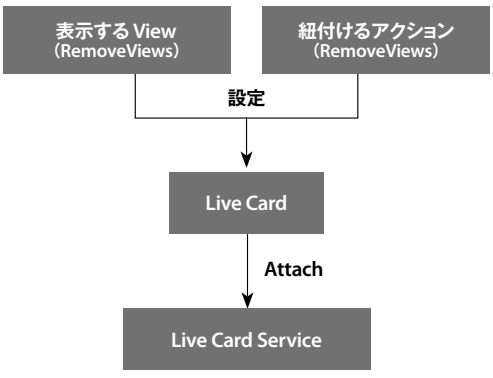
さらに複雑な、前後スワイプなどのジェスチャを認識するには、Glass用に設計された`GestureDetector`を使います。たとえば、「二本指でのタップ」「前方向へのスワイプ」「後方向へのスワイプ」のイベントを認識[注4]したり、タッチパッドに触れている指の数の変化(FingerListener)やスクロール操作のイベント(ScrollListener)を受け取ることができます。

Glassの配布方法

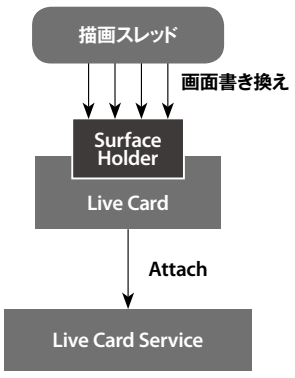
前述のとおり、本稿執筆時点ではMirror API/Android(GDK)どちらの場合でも、Glass

注3、4) サンプルコードがサポートサイトからダウンロードできます。

▼図9 Live Card - RemoteViews



▼図10 Live Card - SurfaceView



への Glassware 配布手段は Google Play のように一般に公開されていません。したがって、GDK を使った Android アプリを開発しても、現時点では APK ファイルを配布する以外の選択肢はほとんどありません。

もちろん例外もあります。たとえば、Glass は Web 上に「MyGlass」と呼ばれる管理用のインターフェースが用意されていて、Glass のユーザは、MyGlass の画面で自分の Glass の状態を確認したり、設定を変更できます。

MyGlass には Glassware に関する項目も用意されていて、インストール可能な Glassware が一覧で表示されます(**図11**)。Mirror API を使った場合でも、Android アプリの場合でも、ユーザがここで Glassware の利用開始を選択することで、手持ちの Glass から利用できるようになります。

しかし、開発したアプリを MyGlass に表示するには、Google から認められる必要があるのです。Google への申請は、Google Glass サイトのフォーム[注5]から受け付けていますが、現状では Google Play のように気軽に公開はできません。

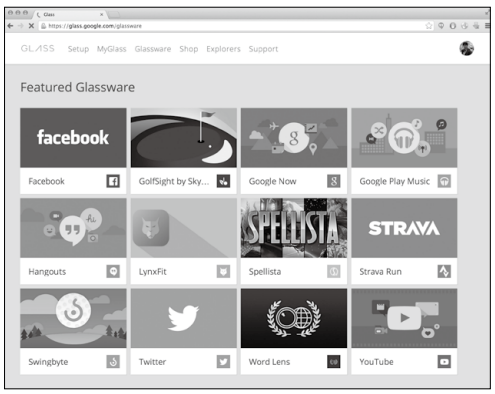
Glassから広がる未来

ここまで、GDK を使った Glass 向けのアプリケーション「Glassware」の開発について解説しました。いかがでしたか?

Glass は、Android が動作しているので、通常の Android アプリもバージョンや Glass が搭載していないセンサーやデバイスを使用していなければ、ほとんどそのままで動作します。

また、GDK を使うことで Glass 固有の機能であるタイムラインの操作や音声入力、タッチパッドのジェスチャ入力にも対応できます。

▼図11　MyGlass の Glassware 一覧



Glass と Android Wear

冒頭に述べたとおり、Glass は「ウェアラブルデバイス」に分類されます。ウェアラブルと言えば、Google は 2014 年 3 月 18 日、Android をベースにしたウェアラブルデバイス向けプラットフォーム「Android Wear」を発表しました。

Android Wear は、Android スマートフォンと連携するしくみで、スマートフォンから Android Wear に情報を表示したり、Android Wear 側の操作をスマートフォンで受けることができます。

Glass が“Google”ブランドで展開されるウェアラブルデバイスであるのに対して、Android Wear は、“Android”のエコシステムに根ざしたプラットフォームです。今後は、さまざまなメーカーがデバイスを展開することが予想されます。現在判明しているだけでも、Android Wear を搭載した製品として、Motorola 社から「Moto 360」が、LG エレクトロニクス社からは「G Watch」と、どちらも腕時計型のウェアラブルデバイスの発売が予定されています。

Android Wear との関連や Google Glass の新型など、ウェアラブルに対する Google 社の今後の展開についても、米国時間 6 月 25 日と 26 日、サンフランシスコで開催される Google の開発者イベント「Google I/O」で、何らかの発表があるものと思われます。期待しましょう。

SD

注5) https://developers.google.com/glass/distribute/form

リアルタイム／分析機能／スケーラブル が武器

複雑化するサーバ環境の監視を変える「OpenTSDB」

前編

サーバ管理者の皆さん、昨今の複雑化・肥大化するサーバ環境の監視業務は、大きな負担になってきてはいないでしょうか。あるいは蓄積されたログを分析していたのでは、障害の発見が遅れてしまったり、絶好のタイミングを逃してしまうという事態になっていませんか。本稿で紹介する監視ツール「OpenTSDB」には、もしかしたら欲しかった機能があるかもしれません。一度試してみませんか。

新井 昇鎬　ARAI Sungho
楽天(株)
PaaS Development & Operation Section

クラウド時代の監視ツール、OpenTSDBとは?

こんにちは。楽天㈱の新井です。現在、社内プライベートPaaSの開発、運用部署に所属しています。いわゆるDev and Opsチームです。

本稿は最近活用事例が増えてきている監視ツールについての話です。複雑さが増す現代のサーバ環境を安全に運用するためには監視・管理が今まで以上に重要になりますが、皆さんはどのように行っていますか？　ZabbixやNagios、Hinemos、Pandora FMSなどといった統合監視ツールを使っている方もいらっしゃるでしょう。しかし、管理する台数の増加に監視ツールが対応できなくなったり、Development、Staging、Productionなどの環境ごとやその環境の中のノードごと、ノード全体に注目して監視、分析したいときに困ったりしていないでしょうか？

今回紹介する「OpenTSDB」は、さらに数千台、数万台といった大規模環境を考慮して生まれた、サーバ上で動作しているOS、ミドルウェア、アプリケーションなど、さまざまなレイヤーのソフトウェアをリアルタイム分析するためのツールです。

OpenTSDBの特徴

最初に、OpenTSDBがどのような特徴を持った監視ツールなのかを説明しましょう。OpenTSDBは次の3つの特徴を持っています。

- リアルタイム
- 分析機能(平均／合計／最大／最小／ダウンサンプリングなど)
- スケーラブル

まずリアルタイムについてですが、"Time Series Database"と名前がついているくらい、ある一定の期間に関して詳細なデータを簡単に取得、表示することができます。OpenTSDB自体にWebブラウザからオペレートするダッシュボードが付属しているので、OpenTSDBの自動アップデート機能を利用すれば自動的にグラフがアップデートされ、リアルタイムの監視が可能となります。

また、OpenTSDBはリアルタイムに監視できるだけではなく、システム全体やある一部に注目して分析できる優れた監視ツールです。たとえば、**図1**のようにノードごとに1日のネットワークI/OのWriteスループットを表示させたり、ノードごとではなく**図2**のようにノード全体で見たWriteスループットも簡単に表示できます。また、**図3**のように、**図2**のグラフを

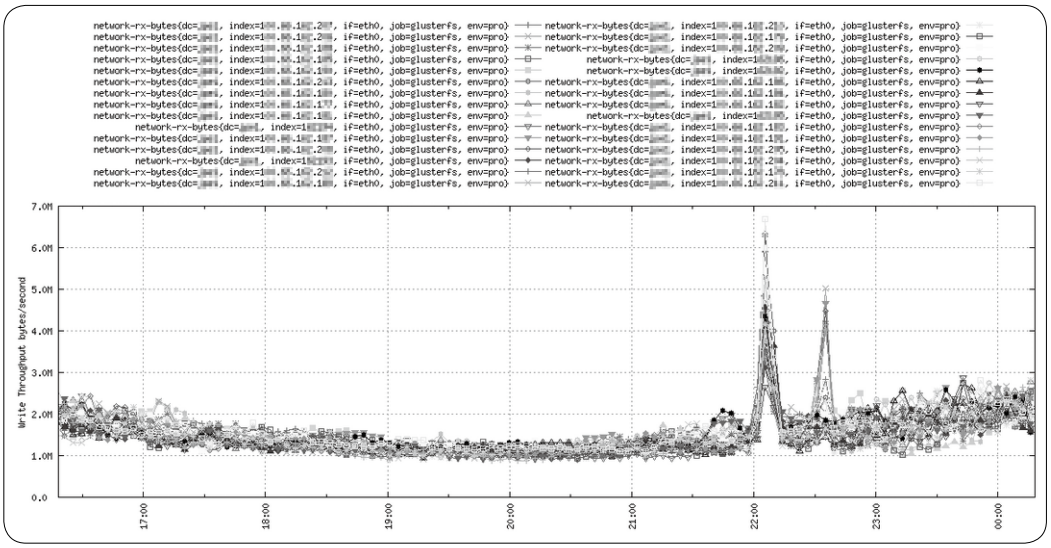
1週間という期間で1時間ダウンサンプリング（後述）して表示させたりすることも可能です。

3つめに、OpenTSDBはスケーラブルな監視ツールです。バックエンドのHBaseのおかげで、OpenTSDBはたくさんのホストやアプリケーションから大量のメトリックを処理することができます。HBaseはGoogleの「Bigtable」のオープンソースクローンであり、大量データに対応した分散ストレージシステムです。そのHBaseを用いることで、スケーラブルで信頼性のある監視が可能となります。

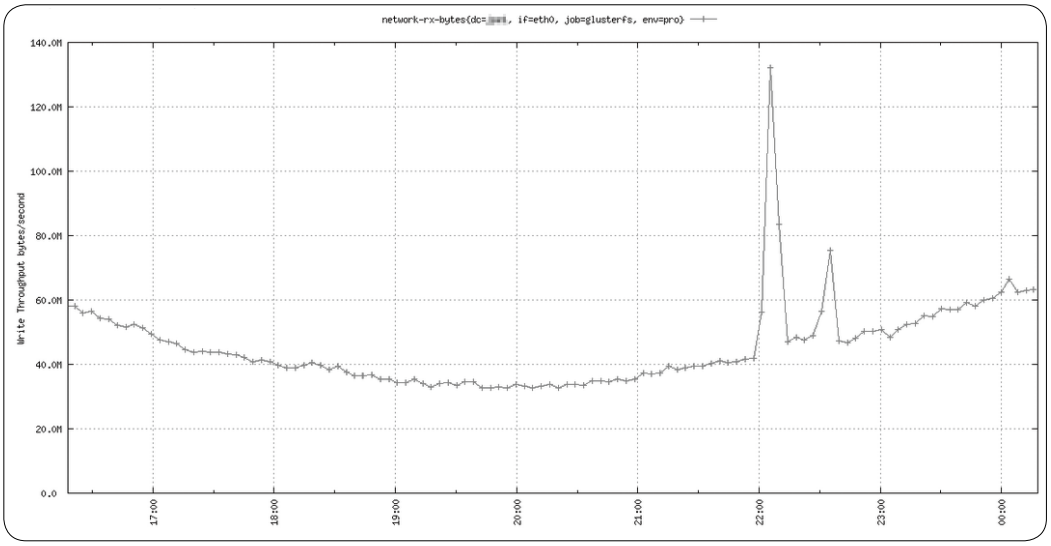
OpenTSDBの用途と導入企業

OpenTSDBの用途はさまざまです。アドミニストレータ（運用する人）が自社システムを監視するためにOpenTSDBを使用したり、システム開発者が新開発したシステムの負荷テストに使用したり、マネージャがチームのKPI（Key

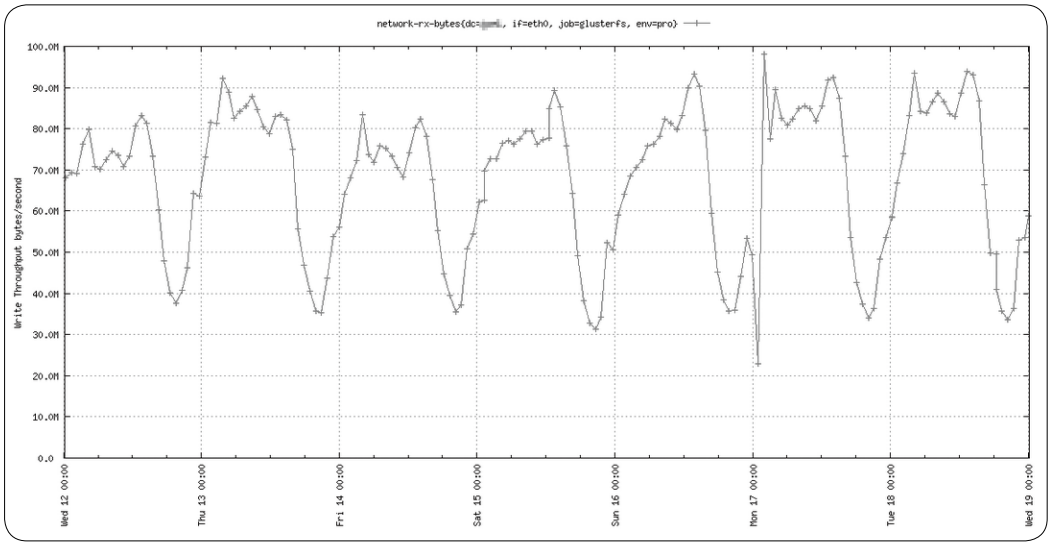
▼図1　各ノードのネットワークI/Oにおける Write のスループットをグラフにした例



▼図2　ノード全体のネットワークI/Oにおける Write のスループットをグラフにした例



▼図3　1週間のノード全体のネットワークI/Oにおける Write のスループットをグラフにした例



Performance Indicator：重要業績評価指標）やシステムの成長をグラフ化するために使用したりすることができます。

OpenTSDBのGitHubのwiki[注1]を見ると、アメリカではさまざまな有名企業がOpenTSDBをすでに導入していることがわかります。たとえば、JIRAやStashなどで有名なアトラシアンやECサイトで有名なeBayなどがOpenTSDBを使用していることが載っています。

Vagrant上でOpenTSDBの構築

OpenTSDBをもっと知るには実際に使ってみるのが一番なので、1台のUbuntu 12.04上で動作するOpenTSDBを構築しましょう。

Vagrant上にUbuntuを用意

まずはVagrantという仮想環境構築ツールを使用してUbuntu 12.04環境を用意しましょう。Vagrantを利用すると、さまざまな仮想環境を同じインターフェースで構築できるという利点があります。Vagrantについては紙幅の都合上説明の対象外とさせていただき、インストール方法や使い方などの解説は割愛します。Vagrantの詳しい情報は公式サイト[注2]で確認してください。

図4のようにVagrantコマンドを実行して、ローカルマシンにUbuntu 12.04を起動させてください。Vagrantfileという設定ファイルを編集して、4242と60010と60030ポートをフォワードする設定を入れるのを忘れないでください。

これでUbuntu 12.04がVagrant上で起動しました。仮想環境が構築されましたので、これからVagrant内の仮想環境を「Vagrant上」と呼び、Vagrant外の普段の環境を「ローカル上」と呼ぶことにします。

それでは次にOpenTSDBを動作させるために、作成したVagrant上にシングルHBaseをインストールしましょう。

注1）https://github.com/OpenTSDB/opentsdb/wiki/Companies-using-OpenTSDB-in-production

注2）http://www.vagrantup.com/

▼図4　VagrantでUbuntu 12.04を起動

```
$ vagrant box add precise64 http://files.vagrantup.com/precise64.box
$ vagrant init precise64
$ cp Vagrantfile Vagrantfile.old
$ vi Vagrantfile
$ diff Vagrantfile Vagrantfile.old
19,22c19
<   config.vm.network :forwarded_port, guest: 80, host: 8080
<   config.vm.network :forwarded_port, guest: 4242, host: 4242
<   config.vm.network :forwarded_port, guest: 60010, host: 60010
<   config.vm.network :forwarded_port, guest: 60030, host: 60030
---
>   # config.vm.network :forwarded_port, guest: 80, host: 8080


$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'precise64'...
[default] Matching MAC address for NAT networking...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] -- 4242 => 4242 (adapter 1)
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Mounting shared folders...
[default] -- /vagrant
$ vagrant ssh
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Sat May 11 05:55:03 2013 from 10.0.2.2
vagrant@vagrant:~$ uname -a
Linux vagrant 3.5.0-23-generic #35~precise1-Ubuntu SMP Fri Jan 25 17:13:26 UTC 2013 ➥
x86_64 x86_64 x86_64 GNU/Linux
```

シングルHBaseの構築

図5のようにCloudera社のレポジトリを利用して、さきほど構築したVagrant上のUbuntuにHBaseをインストールしてください。**図5**の手順が完了すれば、シングルHBaseがVagrant上で動作するようになります。

ローカルのマシン上でWebブラウザを立ち上げて、「http://localhost:60010」と「http://localhost:60030」にアクセスしてVagrant上でHBaseが正常に動作していることを確認してください。**図6**のような画面が確認できると思います。

OpenTSDBをインストール

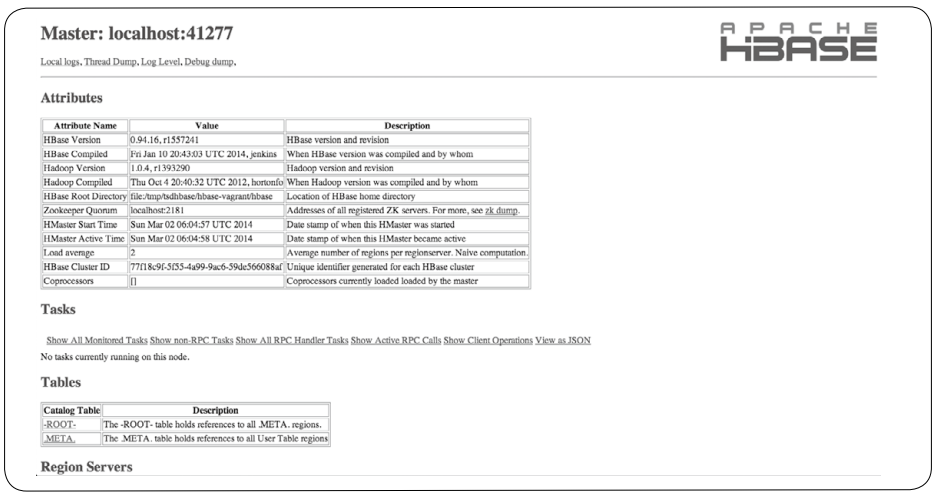
さて、シングルHBaseが動けばOpenTSDBをインストールするための準備が完了です。**図7**のようにGitHubからOpenTSDBのソースをクローンして、debianパッケージを作成してからVagrant上のUbuntuにインストールします。設定ファイルの一部を修正してからstartコマンドを実行してください。

これでOpenTSDBがインストールされてサービスをスタートしました。OpenTSDBはデフォルトで4242ポートを使用します。確認のために、ローカルのブラウザで「http://127.0.0.1:4242/」

▼図5　Vagrant上でHBaseを起動

```
vagrant@vagrant:~$ sudo apt-get update
vagrant@vagrant:~$ sudo apt-get install openjdk-6-jdk
vagrant@vagrant:~$ wget http://archive.cloudera.com/cdh4/one-click-install/precise/amd64/➥
cdh4-repository_1.0_all.deb
vagrant@vagrant:~$ sudo dpkg -i cdh4-repository_1.0_all.deb
vagrant@vagrant:~$ sudo apt-get update
vagrant@vagrant:~$ sudo apt-get install hbase-master
# Copy and paste the following to /etc/hbase/conf/hbase-size.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>file:////hbase</value>
  </property>
  <property>
  <name>hbase.zookeeper.property.maxClientCnxns</name>
  <value>1000</value>
  </property>
  <property>
  <name>hbase.zookeeper.quorum</name>
  <value>127.0.0.1</value>
  </property>
</configuration>
vagrant@vagrant:~$ sudo mkdir /hbase
vagrant@vagrant:~$ sudo chown hbase:hbase /hbase
vagrant@vagrant:~$ sudo /etc/init.d/hbase-master restart
vagrant@vagrant:~$ netstat -ant | grep 60010
tcp6       0      0 :::60010                :::*                    LISTEN
vagrant@vagrant:~$ netstat -ant | grep 60030
tcp6       0      0 :::60030                :::*                    LISTEN
```

▼図6　HBaseの正常動作確認



にアクセスしてみましょう。**図8**のような画面をローカル上のブラウザから見ることができたら準備完了です。

起動が失敗していた場合は/var/log/opentsdb/opentsdb.logがOpenTSDBのログなので、そこを確認してみてください。

Vagrant上にOpenTSDBを構築できたので、続いて実際にデータを入れてみましょう。

▼図7　Vagrant上でOpenTSDBを起動

```
vagrant@vagrant:~/opentsdb$ sudo apt-get install autoconf git gnuplot
vagrant@vagrant:~$ git clone https://github.com/OpenTSDB/opentsdb.git
Cloning into 'opentsdb'...
remote: Reusing existing pack: 4532, done.
remote: Total 4532 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (4532/4532), 26.75 MiB | 552 KiB/s, done.
Resolving deltas: 100% (2993/2993), done.
vagrant@vagrant:~$ cd opentsdb/
vagrant@vagrant:~/opentsdb$ git checkout v2.0.0RC1
vagrant@vagrant:~/opentsdb$ sh build.sh debian
vagrant@vagrant:~/opentsdb$ sudo dpkg -i build/opentsdb-2.0.0/optsdb-2.0.0_all.deb
vagrant@vagrant:~/opentsdb$ env JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64/ COMPRESSION=➥
NONE HBASE_HOME=/usr/lib/hbase/ ./src/create_table.sh
vagrant@vagrant:~$ sudo vi /etc/opentsdb/opentsdb.conf
以下の項目をtrueにしてください
tsd.core.auto_create_metrics = true
vagrant@vagrant:~/opentsdb$ sudo /etc/init.d/opentsdb start
* Starting TSD...
```

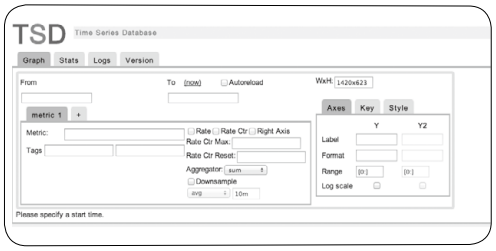
OpenTSDBにデータを入れてみよう

■putコマンド

OpenTSDBは**図9**のようなフォーマットでモニタリングするデータをinputとして受け取ります。

metric(メトリック)は時系列に監視したい項目の名前です。たとえばcpuやmemoryやload_averageなどのメトリックが使用されます。**timestamp**はそのデータを時系列上に表示するためのある1点です。UNIX/POSIX Epochタイムスタンプが使用されます。**value**はその監視項目のtimestamp時点の値です。**tag**(s)はこのメトリックに対してインデックスを張り、あとでメトリックを分析するときに使用します。

▼図8　OpenTSDBの正常動作確認



▼図9　putコマンドのフォーマット

```
put <metric> <timestamp> <value> <tagk1=tagv1[ tagk2=tagv2 ...tagkN=tagvN]>
```

▼図10　OpenTSDBのstatsインターフェース

```
vagrant@vagrant:~$ curl http://localhost:4242/stats
tsd.connectionmgr.connections 1393751578 2 type=open host=vagrant
tsd.connectionmgr.connections 1393751578 5 type=total host=vagrant
tsd.connectionmgr.exceptions 1393751578 0 type=closed host=vagrant
tsd.connectionmgr.exceptions 1393751578 0 type=reset host=vagrant
tsd.connectionmgr.exceptions 1393751578 0 type=timeout host=vagrant
tsd.connectionmgr.exceptions 1393751578 0 type=unknown host=vagrant
tsd.rpc.received 1393751578 0 type=telnet host=vagrant
tsd.rpc.received 1393751578 10 type=http host=vagrant
tsd.rpc.exceptions 1393751578 0 host=vagrant
tsd.http.latency_50pct 1393751578 4 type=all host=vagrant
tsd.http.latency_75pct 1393751578 10 type=all host=vagrant
…(省略)…
```

▼図11　statsからputコマンドを生成

```
vagrant@vagrant:~$ curl http://localhost:4242/stats | awk '{print "put " $0}'
put tsd.connectionmgr.connections 1393752700 3 type=total host=vagrant
put tsd.connectionmgr.exceptions 1393752700 0 type=closed host=vagrant
put tsd.connectionmgr.exceptions 1393752700 0 type=reset host=vagrant
put tsd.connectionmgr.exceptions 1393752700 0 type=timeout host=vagrant
put tsd.connectionmgr.exceptions 1393752700 0 type=unknown host=vagrant
put tsd.rpc.received 1393752700 76 type=telnet host=vagrant
put tsd.rpc.received 1393752700 2 type=http host=vagrant
put tsd.rpc.exceptions 1393752700 0 host=vagrant
put tsd.http.latency_50pct 1393752700 48 type=all host=vagrant
put tsd.http.latency_75pct 1393752700 48 type=all host=vagrant
…（省略）…
vagrant@vagrant:~$ sudo apt-get install tcputils
vagrant@vagrant:~$ curl http://localhost:4242/stats | awk '{print "put " $0}' | tcpconnect ➡
localhost 4242
```

▼図12　5秒おきにOpenTSDBにデータを投入

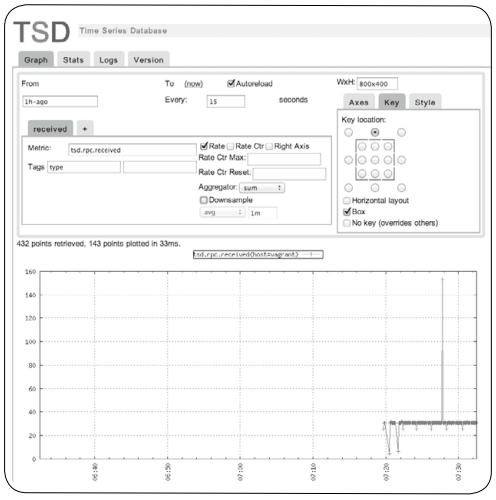
```
vagrant@vagrant:~$ while true; do curl http://localhost:4242/stats | awk '{print "put " ➡
$0}' | tcpconnect localhost 4242; sleep 5; done
```

■statsインターフェースからputコマンドを作成

OpenTSDBは自分自身の現在状態を返すためのstatsと呼ばれているインターフェースを持っています。**図10**のようにcurlコマンドでそのstatsインターフェースにアクセスすることができるので実際に取得してみましょう[注3]。

注3）curlがインストールされていない場合は"apt-get install curl"でインストールしてください。

▼図13　リアルタイムに表示



この行指向のstatsから、さきほど説明したputメッセージを作成しましょう。**図11**のようにawkを使用すれば、簡単にstatsをputコマンドに変換することができます。あとはこのputコマンドをOpenTSDBに渡してあげれば、データを表示することができます。

■リアルタイムにデータを表示

実際にOpenTSDBにデータを入れるところまでできたので、継続的にデータを投入することでOpenTSDBの自動グラフアップデート機能を利用したリアルタイム表示を体験してみましょう。

図12のコマンドを実行すると5秒ごとにOpenTSDB自身のstatsの情報がOpenTSDBに投入されます。コマンドを実行したら、ローカル上のブラウザでVagrant上のOpenTSDBにアクセスしてみましょう。「tsd.rpc.received」などの適当なメトリックを選択して「Auto reload」のチェックボックスを有効にしてください。このチェックボックスを有効にすると、**図13**のようにダッシュボード上のグラフが自動でアップデートされます。これでリアルタイム監視の完了です。

OpenTSDBの武器は、何よりもこれから説明する分析機能です。

分析ツールとTcollector

OpenTSDBには、リアルタイムにデータを表示させる機能のほかにもさまざまな分析機能が用意されてます。ここではレート、ダウンサンプリング、Aggregationと、OpenTSDBのAgentツールのTcollectorを紹介します。

レート

レート機能を利用すると、クエリ数などのメトリックを監視している場合は一定時間に増加するクエリ数、つまりQPS(Query per second)を計算することができます。また、エラー数をモニタリングしていた場合には、レートをとることでError Rateを計算することができます。なお、さきほどVagrantの上で監視した「tsd.rpc.received」もレートをとることでQPSを表示させています。

ダウンサンプリング

ダウンサンプリング機能を利用すると、長い期間でデータを表示させたいときに一定期間のデータを平均して表示させることができます。たとえば、1日の1時間平均や1年間の1日平均などを表示できます。ダウンサンプリングを使用すると平均をとるので、短い時間のスパイクなどが吸収されて見えなくなる可能性があります。そこは理解して使用してください。

Aggregation

OpenTSDBはメトリックをただ保存するだけではありません。あるメトリックに対してタグをつけることが可能です。タグ機能を利用すると、あるキーでタグづけされた値ごとに表示させたり、まとめて表示させたりできます。たとえば、タグの値にそれぞれのノードのIPを入れた場合は、あるIPのノードだけに注目したり、ノード全体で平均をとったりすることができます。ほかにも、タグにアプリケーション名を入れてCPUメトリックを監視している場合は、あるアプリケーションのCPU利用だけを監視したり、アプリケーション全体でどのくらいCPU資源を使用しているのかも見ることができます。

tcollector

tcollector[注4]は、Pythonで書かれたOpenTSDB用の監視Agentツールです。監視したいターゲットで動作させれば、OpenTSDBに定期的にデータを入れてくれます。tcollectorはMySQLやHadoop、Elastic Searchなどさまざまなソフトウェアに対応しています[注5]。自分でスクリプト作成して追加すれば、独自の環境も監視することができます。

まとめ

今回はリアルタイム分析に優れた監視ツールのOpenTSDBを紹介しました。クラウド時代の大量サーバや複雑なレイヤーを監視するのにOpenTSDBは適しています。バックエンドのHBaseによりスケーラブルで、タグ機能によりデータをレイヤーごとやノードごとに分けたり、統合できたりします。今回紹介した方法のように1つのサーバで動作させることも可能なので、ぜひ一度OpenTSDBを試してみてください。

次回は、URL共有やダッシュボード作成などのOpenTSDBの実践的な使用方法を説明します。SD

注4) http://opentsdb.net/tcollector.html

注5) https://github.com/OpenTSDB/tcollector/tree/master/collectors/0の下を見れば、どのソフトウェアに対応しているかが見られます。

短期集中連載

アプリケーション開発者がインフラ担当に!

Rettyのサービス拡大を支えた“たたき上げ”DevOps

第2回 ほどほどのセキュリティ対策と監視

実名ユーザたちによるお勧めからレストランを探せるグルメ系Webサービス「Retty」。急成長するサービスの裏側では、融通のきかない古いシステムから大規模システムへの移行という難題が立ちはだかっていました。それを乗り越えたのはインフラ経験なしのアプリケーションエンジニア。スマートなだけではすまされない、現場でのInfrastructure as Code実践を紹介してもらいます。

Writer 梅田 昌太(うめだ しょうた)　Retty㈱　チャーハン担当

できる範囲で最善をつくす

最近ログの収集やちょっとしたマイニング、開発者へのエスカレーションに凝っているRettyチャーハン担当の梅田です。

この原稿を書いているときはAWS(Amazon Web Services)のELB(Elastic Load Balancing)がアクセスログの出力をサポートしはじめて2週間ほど経っているので(筆者はサポート初日に設定しました)、この号が発売される頃にはログの有効活用ができていることを祈ってます。

前回[注1]はRettyのWebサーバのリファクタリングを中心に、「変化に対応できるように」するため既存のサーバ／インフラ構成を掘り起こしたことを中心に書きました。

今回は第2回ということで前回に引き続き「変化に対応できるように」するため、

・ほどほどに行うセキュリティ対策
・ほどほどに行う監視

について書いてみようと思います。

前回も書いたとおりRetty内にはWebサービスを提供するにあたっての、アンチパターンな設計がまだまだ残っています。その内容はアプリケーションの構成もインフラもあらゆるところに細々と残っています。こういった技術的負債は後々チャンスが到来したときに足かせとなるので、できる限り返済しておきたいものです。

しかし、これらの問題を対処するためにある日突然、

「今日から一斉にセキュリティ対策する！」
「今日から1ヵ月間リファクタリングだけやる！」

といったことは現実的ではありません。事実上不可能だと思っています(やっている会社さんもあるのかもしれませんが)。

悪く言えば日々の改善に追われているということなんですが、もう少し言うと、これらはユーザさんへ価値を届ける直接的な作業ではないので、日々の改善を止めてまで行うということはサービス運営者としてはできるだけ避けたいところです。なので、前述したとおり「ほどほどに」という前提のもと、対策を行うことが大事だと思います。

ただし、対策は行ってもリスクは残ります。そのリスクはしっかりとコミュニケーションで共有しておく必要があります。リスクをゼロにすることは不可能ですし、神経質にリスク管理ばかりを行うと本質的な作業の足かせになるからです。

Rettyのインフラ構成

これからRettyのセキュリティ対策について触れる前に、前回よりもう少し広くRettyのインフラ全体の構成を見てから説明したいと思い

注1) Software Design 2014年5月号

ます。

Rettyの現状は**図1**のようになります(個別の台数などは省いています)。すべてを1人で見ているわけではないですが、我ながらよく組めたもんだなと思っています。AWSならではだと思います。

大まかに説明するとサブドメイン含め、次のような3つのドメインをRoute53で振り分けています。

- retty.meはEC2-Classicの中にELB＋EC2＋RDS(Relational Database Service)＋サブシステムいろいろ
- news.retty.meはVPC(Virtual Private Cloud)
- owner.retty.meはElastic Beanstalk(PHP 5.5、MySQL5.6.13)＋Elastic MapReduce

こういった構成の中で、個別の問題に対応したことについて紹介します。

厄介な問題、セキュリティ

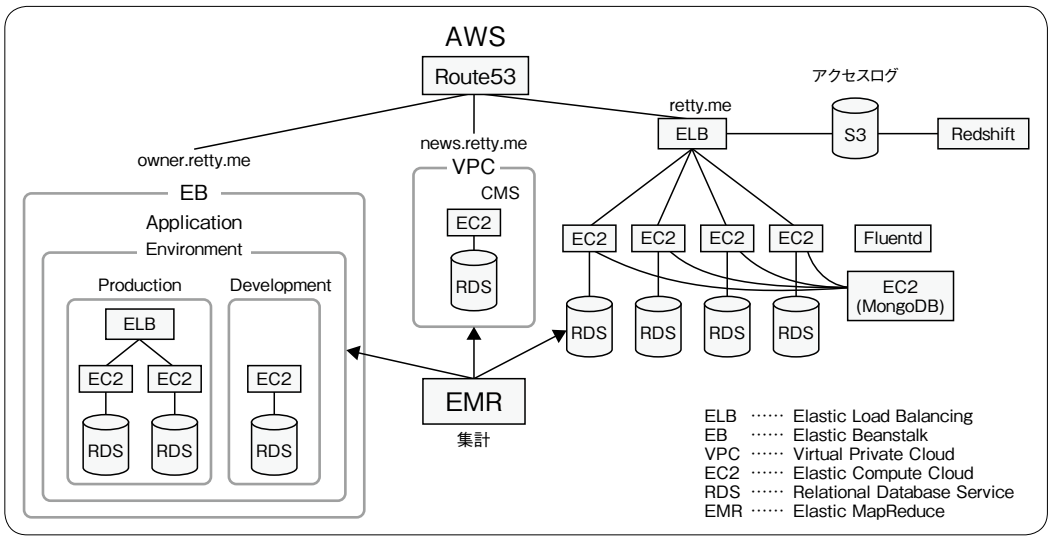
成長過程に入ったところでいろいろな意味でやっかいな問題としてあがるのがセキュリティです。多分「あるある」だと思います。

大前提として、セキュリティは非常に大事です。が、ないがしろにされがちで、正直筆者もあまりやりたい仕事ではありません。というのもセキュリティ対策というのは基本的に「不自由になる」ことしかなくて、「自由」になったり「便利」になることはありません。おまけに変更によるリスクが大きかったり、移行コストが高かったり、その割にプロダクトの価値自体が上がるわけではないのでますます厄介です。とはいえ「後悔先に立たず」なのがセキュリティ対策なのでやるしかないのです。

誤解を恐れずに言えば、旧環境のRettyはスタートアップサービスらしく全体的に「とりあえずポートを開けといた」的な設定が散見されます。で、これは容易に想像できると思うのですが、「開けるのは易(やす)し、閉じるのは難(かた)し」です。開いている口を閉じるということは、自分の把握してないジョブやサービスが突然動かなくなってしまう可能性があり、ユーザさんに不快な体験をさせてしまう可能性があります。また、ジョブがコケれば大事なデータがロストしてしまう可能性もあります。

これに対してはテクニカルな解がないのが悲しいのですが「セキュリティ強化に対して、メンバーと慎重にコミュニケーションを取りなが

▼図1　Rettyのインフラ構成図



ら意識を合わせるしかない」というのが結論です。ちなみにここで挙げているセキュリティとはAWSサービスへのアクセスコントロールがほとんどです(ブラウザからのWebアプリケーションアクセスやRDSへのアクセスを含みます)。PCの持ち出しとかそういった類いのものは別担当となります。

ということでいろいろと問題を抱えているセキュリティ問題への対策として、

「極力不自由を持ち込まない」
「導入コストを低く」
「手厚くサポート」

を念頭に置いて対策を行っています。

RettyのAWS上でのセキュリティの問題点

ここではretty.meの問題点について書きます。問題点は大まかに言うと3つありました。

①ロール(役割)が違うのにAWS上で同一のセキュリティグループが使い回されている
②VPCが組まれていない
③公開鍵の管理ができていない

①と②は問題点として若干重なるのですが、サービス発展時にインスタンスをロールごと(役割ごと)のセキュリティグループに分けていなかったのが背景としてあります。詳細なポリシーについては差し控えますが、**図2**のように開発環境も本番環境も同一の大きなセキュリティグループに全体が属していました[注2]。

こういうふうに1つのセキュリティグループになっていると、「このインスタンスはWebサーバを落としたから80番を閉じたい」「このサーバは踏み台からしかアクセスしないのでIPを制限したい」という要望に対して柔軟にopenもcloseもできません。

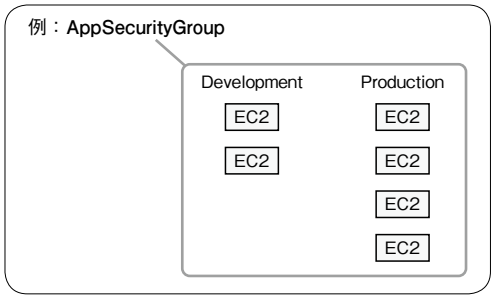
では、セキュリティグループを変更(分ければ)すれば良いのでしょうか？　残念ながらEC2-Classicを利用していると、インスタンスのセキュリティグループを後から変更することができません。

VPCを組むと「ローカルIPの固定」「セキュリティグループが変更できる」という嬉しい機能もつくので、本音はVPCに移行するのが望ましいと思っていました。しかし、VPC自体のノウハウがなかったことや、移行コストがかかるため(RDSなどアプリケーションサーバ以外の部分も検討しなくてはいけない)今回は見送って、

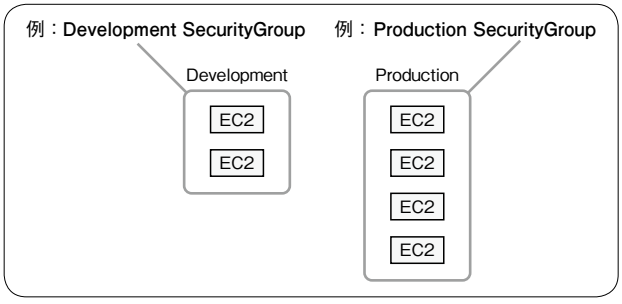
- セキュリティグループを割り振りなおす(導入コストを低く)
- オフィスに固定IPを割り振る(導入コストを低く)
- VPN環境を構築(極力不自由を持ち込まない。手厚くサポート)

注2) AWS上の「セキュリティグループ」というのは大まかに言うと、個別のインスタンスに対して「どういったポートを開けるのか？」「どういったインスタンスからの接続を許可するのか？」といったパラメータを設定して、インスタンスごとに適用できる機能です。詳しくはドキュメントを参照してください。

▼図2　改善前のRettyのセキュリティグループ



▼図3　改善後のRettyのセキュリティグループ



で対応することにしました。

大まかに言うと、開発環境用のAWSへのアクセス(sshや開発環境へのhttp)は社内IPからのみとしました(オフィス移転に伴って固定IPを取得してVPNを構築)。Rettyにはリモート開発者もいるので、その課題はVPNで対応することにしました。VPN環境はドキュメントをそろえてしっかりサポートします。ちなみにVPCへの移行はいずれ必要と考えているので、サブドメインサービス(news.retty.me)で小さく始めてみることにしました。

インスタンスを作りなおして新しいセキュリティグループに分ける

セキュリティグループを割り振りなおすために、インスタンスを一度廃棄して作りなおします。これをいちいち手作業でやっていたら手間がかかってしょうがないですが、前回記事に書いたWebサーバの入れ替え作業時に作ったレシピがあるので、これを実行して新しいインスタンスを作りました。起動も削除もとても楽ですね(Vagrant + shell)。

こうして**図3**のように開発環境用のセキュリティグループと本番環境用のセキュリティグループに分けました。その中身をもう少し詳しく書きます。

開発環境のセキュリティグループ(Development SecurityGroup、**図4**)を構築したときのポイントは、

- AWSへのアクセスは社内IPからのみとする
- 緊急対応と遠隔からの接続はVPNを用意する

これで守るポイントを社内アクセスポイントのみとすることができます。

一方、本番環境のセキュリティグループ(Production SecurityGroup、**図5**)では、EC2側へのアクセスを基本的に社内IPからのみとしました。違う点は、

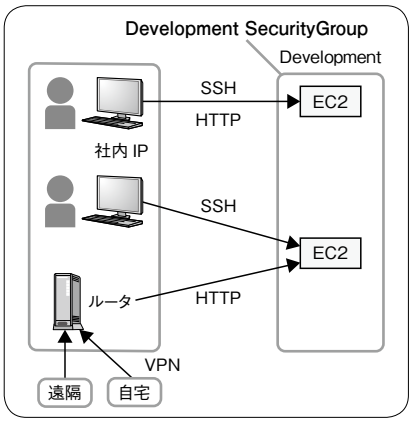
- EC2へのsshでのアクセスは踏み台サーバ(OpsServer)を通すこと
- WebサーバへのhttpリクエストはELBからのみとすること

このようにすることで、直接Webサーバへリクエストを投げることはさせないようにします。

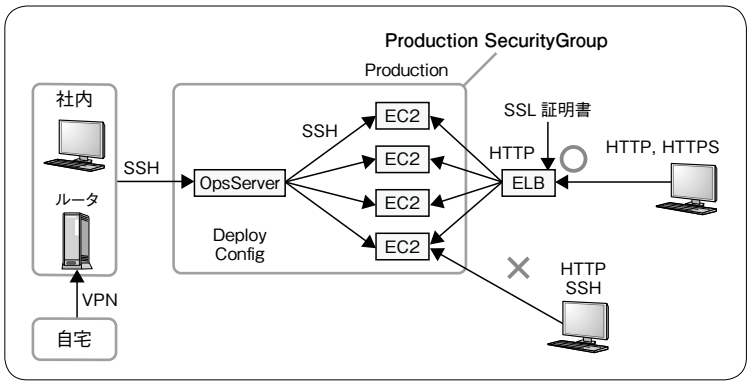
公開鍵の問題

Rettyでの公開鍵管理は何が問題だったかというと、不要になった公開鍵と必要な公開鍵がわかるようになっていないという点でした。なぜかというとRettyでは社員だけでなくインター

▼図4　開発環境(Development SecurityGroup)



▼図5　本番環境(Production SecurityGroup)



ン生にも積極的にソースを書いてもらっていて[注3]、作業するサーバにはそれぞれの公開鍵を設置します。しかしインターンという働き方の特性上、人の入れ替わりがそれなりの頻度で起こり、当然アクセスが不要になった場合は対象ユーザの公開鍵を削除する必要があります。

GitHubの公開鍵を取りなおして対応する

解決には、既存の鍵を整理するのではなく、公開鍵は全部捨てて鍵を入れなおすことにしました。知っている方には当たり前のことかもしれませんが、GitHubでは、「https://github.com/ユーザ名.keys」でそのユーザの公開鍵を参照することができます。ですのでここから公開鍵を取得してサーバに配置することにしました。

図6のような感じで鍵を追加するshellを用意しておくことで、GitHubでコラボレートしているユーザ名だけ管理しておけば、ユーザ名に基づいてすべての公開鍵を入れ替えることができます。

筆者はshellで行っていますが、CapistranoやFabricのようなデプロイツールを使うと、リモート側で実行するコマンドをRubyやPythonのようなプログラマブルなコードで管理でき、ロール分けも細かくできるでしょう。対象のロールが増えてきたら導入を検討してみると良いかもしれません。

セキュリティ対策のまとめ

以上が、簡単にではありますがほどほどに行ったセキュリティ対策の一部となります。最初に書いたとおり、基本的に「開発作業が楽になる」といった良いところはありません。「開いていた門を閉じただけ」なので、VPNの設定作業が必要だったり、sshの接続が面倒ならフォワードの設定をしたりする必要があるだけです。

こういった作業はペアワークで作業したりして丁寧に行わないと、「あー。やっぱりセキュリティ対策なんて面倒なだけ」と言われてしまいます。何かあってからでは遅いのですが、何か起こらないと価値が見いだしにくいのがとても厄介ですね（何も起こらないことが素晴らしいことだ。という啓蒙が必要なのでしょう）。

ほどほどの監視

何のために、何の監視を行うのか?

前回も若干触れましたが、RettyではもともとVPSやレンタルサーバの延長線上のアーキテクチャでクラスタが組まれていたので、統合的な監視が行われていませんでした。ということで筆者はRettyにアサイン（入社）してから「統合的な監視のアーキテクチャ」を考えるところからはじめました。

統合的な監視アーキテクチャというと大層に聞こえますが、具体的な作業としては「監視のツールを選定して設定をチューニングしている」だけのことです。

結論＆現状として、Rettyでは、

・おおまかなモニタリング→New Relic
・プロセスの監視→Nagios
・リソースの監視→CloudWatch

を使っています。

これらの監視ツールに関して具体的な話に入る前に、「監視」についてちょっと考えてみたいと思います。筆者はもともとアプリケーションエンジニアだったこともあり、これまでのキャリアでは監視に関して「インフラ担当が用意して

注3）https://www.wantedly.com/companies/retty

▼図6　公開鍵の追加コマンド例

```
curl https://github.com/ユーザ名.keys >> ~/.ssh/authorized_keys
```

くれたものを使う」というのが当然だと思っていました。おおざっぱに、エンジニアとしてアサインしたら、

・アラートのメーリングリストに入れてもらう
・Webベースのモニタリングツールを見られるようにしてもらう

というのが今までの認識でした。

アラートのメールが飛んできたら用意されていたNagiosやMRTG[注4]を見て、「あー。何かCPU食ってるな」とか「ディスクの容量がなくなってきたなー」などと監視ツールが知らせてくれることに対して考察し、原因がアプリケーションに起因するものであれば、ホットフィックスを行うまでが主な仕事でした。

しかし、Rettyにはそういった統合的な監視は導入されていなかったので、「何のために何を監視するのか？」を改めて考える必要がありました。結論から言うと「サービスダウンの前に障害を見つける」ためです(当たり前なんですが……)。ここで言う「障害」とは「サービスダウン」のことではなく、サーバが何か異常な振る舞いを見せている状態です。

当然ですが、

・プロセスは突然落ちることがあります
・プロセスは突然暴走することがあります
・ハードウェアリソースは突然枯渇することがあります

なので、単一の障害がサービスダウンにつながらないようするためには、スケーラブルなアーキテクチャにしておくことが非常に重要です。

この結論はこれまでの経験とは違うアプローチから出てきた結論です(理屈上ではわかっていた気はしますが)。というのも大規模サイトの開発経験は一応あるので、それなりのスケーラブルな設計、実装経験はありましたが、それは負荷対策としての思想が強かったのです。もちろんそれはそれで大事なのですが、スケーラブルであるということはシステム全体が疎結合であることが大前提になるので、単一の障害がサービスダウンにつながりににくくなるということに気づいたわけです。

要するに、とあるインスタンスのWebサーバプロセスが落ちたり、AWSの単一のゾーンが落ちたりといったことがあっても、サービスを存続できるしくみにしておく必要があります。書いていることはどれも頭ではわかっていたのですが、なかなか実感できなかったことでした。

グルメサイトで働いてるのに安心してランチに行けないストレス

Rettyのピークタイムは平日の12時〜13時頃と20時頃がピークです。ありがたいことにランチや夕食(呑み？)を探すのに使っていただけているのだと思います。

が、ということはOpsにとって最もハラハラする時間でもあります。アプリケーションサーバのプロセスが暴走したり、最悪落ちたり。データベースのコネクションが詰まったりしやすいのもこの時間です。自身こそが食事をこよなく愛するグルメサイトの社員としては、この時間に拘束されてしまうのは由々しき事態です。

まずはサービスを継続させ続ける

現在のRettyの障害に対する心構えは、障害を起こさないことよりも、障害をいち早く検知して復旧させることに重きを置いています。仮に、とあるサービスの障害対策フローが次のようなものだったとして、

①障害発生
②障害検知
③復旧作業
④復旧完了

この一連の時間を短くすることと、①〜③の間

注4) Multi Router Traffic Grapher：ネットワーク負荷をグラフ化して監視するツール。

は多少のパフォーマンス低下を許容しつつも、サービスを稼働させ続けることが必須になります。

A-Z配置でサービスを稼働させ続ける

これはテクニックではなく、AWSが推奨しているアーキテクチャです。いまさら説明するまでもないことですが、AWSはリージョン(例：東京)の中にアベイラビリティーゾーンという複数のそれぞれ独立したロケーションがあります。これによって片方のアベイラビリティーゾーンに障害が起こったときも、反対側でサービスを継続させることができます。ですので、インスタンスはゾーンを振り分けて同数ずつ置くのがセオリーになります(**図7**)。

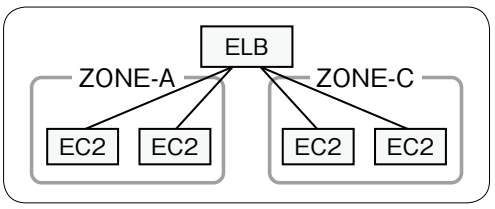
若干耐障害性とは話がそれますがELBはゾーンに対して均等にアクセスを割り振るので(詳しくはAWSのドキュメント参照)、同数のインスタンスを配置しておかないと均等にアクセスがさばけません。割り振る各ゾーンには均等にインスタンスを配置する必要があります。このように配置してホストそのものの障害に備えます。

検知と復旧を最短にする

次はプロセスの監視について対応します。

Rettyでは統合的な監視ツールは導入されていませんでした。これは繰り返しになりますが「まだ必要がなかった」からです。しかし、筆者が入社してからユーザ数が倍になっています。もう監視なしではサービスの安定稼働ができません。

▼図7　またがったゾーンに同数ずつ配置



監視ツールを導入するまではとりあえず「Rettyにつながらないです」という報告を受けてから、topやpsコマンド、mysqlコマンドなどのコマンドラインで調査して適切に対処していました。しかし、監視ツールがあれば「プロセスが落ちている」「リソースが枯渇している」といったことがすぐにわかるので、対応までの時間が大幅に削れます。ログを追って原因を調査するのは復旧した後です。

そこで筆者が選定したのがNew Relic[注5]とNagios[注6]とCloudWatch[注7]です。

New Relic

大人気の統合的なモニタリングツールです。大まかな監視であればハードウェアリソースもプロセスの監視も行ってくれます。ですが、個別のプロセス監視やリソース監視は微妙に速さや内容が不明瞭です。New Relicは全体的な傾向(サーバレスポンスタイムなど)を追うのに使っていて、サクっとモニタリングするには最適です。Amazon Linux 64bitで使いたい場合、オフィシャルに書いてあるコマンドをコピペするだけで始められます。基本的にはRPMのリポジトリを追加して、yumでインストールするだけで自動的にモニタリングが始まります。

Nagios

個別のプロセス監視とAWSのリソースのアラートには、筆者が使い慣れているNagiosを導入しました。mixi、DeNAといった大規模なサイトでも利用実績がある統合監視ツールです。モダンなものとしてはZabbixやSensuがあると思いますが、数十台の監視であれば十分対応できますし、レイヤーが薄い監視ツールなので自作のプラグインを書くことも容易です。細かいことは公式ドキュメントに書かれているので省きますが、Nagiosは統合監視と言ってもしくみ

注5) http://www.newrelic.com/
注6) http://www.nagios.org/
注7) https://aws.amazon.com/jp/cloudwatch/?nc1=h_l2_dm

は「設定した間隔で」「指定したコマンドを実行して」「戻り値を表示、閾値次第でアラート」というシンプルなものなので、PerlやRubyで簡単にプラグインを作れます。標準のプラグインも十分充実しているので筆者は、

・標準のプラグインでWebサーバとMySQL(RDS)を監視
・AWS CLI(Command Line Interface)とPerlを組み合わせてEC2とRDSのリソースを監視

しています。

CloudWatch

おもにEC2とRDSのリソース監視を行っています。Nagios側からもAWS CLIを通してAPI経由で監視を行っているのですが、やはり時系列にしたグラフで見たいと思うときがあると思います。そういったグラフの描画といったところはCloudWatch側に任せます。ちなみにリソースの監視をするといえばおもにCPU、Disk、Memoryだと思いますが、EC2のCloudWatchは標準ではDiskとMemoryモニタリングが付いていません。これらはプラグインのインストールで監視できようになります[注8]。

ドキュメントに書いてあるとおりにするだけでOKですが、大まかに書くと、適当なディレクトリにソースをダウンロードして、解凍してダウンロードディレクトリの中に入っているawscredsというファイルの中で「AWSAccessKeyId」と「AWSSecretKey」を設定すればインストール完了です。後はcronでCloudWatch側に定期的に情報を送ってやります。

```
*/5 * * * * ~/aws-scripts-mon/mon-put-instance-data.pl --mem-util --disk-space-util --disk-path=/ --from-cron
```

こんな感じでcronを設定してあげるとモニタリングできます。

注8) http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/mon-scripts-perl.html

監視のまとめ

監視を行うことによって障害対応が行いやすくなるだけでなく、ボトルネックになっているアプリケーションやクエリを発見する手助けにもなります。長めのスパン(1週間くらい)で「全体として」の傾向を見るのに「New Relic」。今すぐ対応する必要があるもの、障害の検知には「Nagios」。リソースの使用状況を時系列で見るのに「CloudWatch」という使い分けをしています。

統合的なツールを使うとどうしても「1つのツールですべてを見たい」となりがちですが、使い方をはっきりさせれば、すべてを1つのツールに絞らず適切にアウトソースしながら監視できると思います。

第2回のまとめ

いかがでしたでしょうか。まだまだ本来あるべき姿というのには程遠いですが、セキュリティの問題にしろ、監視の問題にしろ、ほどほどに対応するというのは非常に大事だと思います。できることから1つずつ、です。なので細目に許容しているリスク(セキュリティリスク、障害リスク)をメンバーでしっかり共有しておくことは非常に重要です。

また、セキュリティや監視のしくみはある程度大規模な組織に入るとはじめからいろんなものが当たり前のように用意されていることが多いです。それはそれで素晴らしいのですが、改めてゼロから考えてみると、今まで見えてなかった視点で物事を見られるので、ぜひ一度考えてみると良いと思います。

次回は「モダンな開発環境」へシフトして、ユーザさんに、より新しい価値を提供することについて書こう思います。SD

「Web標準技術」で行う Webアプリのパフォーマンス改善

短期集中連載

第2回 通信環境の改善で高速化を図る

Writer 川田 寛(かわだ ひろし)
NTTコムウェア㈱ 技術SE部
Web技術者コミュニティ「html5jエンタープライズ部」部長
URL furoshiki.hatenadiary.jp Twitter @kawada_hiroshi

今回は、プロトコルやキャッシュなど通信における種々の課題を解決する手段を紹介します。HTML5で注目を浴びているWebSocketやAppCacheも登場します。「名称は聞いたことがあるけど、詳しくは知らない」という人はこの機会にどんな機能なのか、その概要を理解しましょう。

HTML5はインフラも変えようとしている

パフォーマンスの観点で見れば、Web技術はインフラ系エンジニアにも理解が求められる技術です。HTML5の登場で、Webのインフラストラクチャとしての側面は、より強いものになりました。今回は、少し違う方向からパフォーマンスの改善方法を取り上げます。「ファイルの特性」から「通信の特性」に観点を変えてみます。アプリケーション開発者よりも、インフラエンジニアのほうが関心の高い内容となるでしょう。

プロトコルレベルでのパフォーマンス改善

XMLHttpRequestの普及とCOMETの登場

かつてのWebでは、「Webページの読み込み」だけが通信に求められました。Webは、HTMLドキュメントを読み込み、文書としてWebページを表示するという目的のために作られたのです。そこに、Netscape Communications社とシェア争いをしていたMicrosoft社が、Internet Explorer(以下、IE)でWeb技術をビューアからアプリケーションプラットフォームへと押し上げました。

サーバとの通信を変える決定的な事件は、1999年にリリースされた「Internet Explorer 5」と「Outlook Web Access 2000」です。Microsoftが提供していたWebメールのサービスは、さらなる改善のために、JavaScript上で非同期にデータを取得する機能が求められました。このニーズを埋めたのが、現在広く利用されている「XMLHttpRequest(通称XHR)」です。

XHRは、すでに読み込んだWebページ上のJavaScriptからサーバへHTTPリクエストを送れるAPIで、これを使うことでWebページの遷移なしにデータを取得できます。名前にXMLとありますが、XMLである必要はなく、自由なフォーマットでデータ通信が行えます。Google Mapsの登場で「Ajax」(XHRを使った動的なページ書き換え)が注目されます。そして、jQueryでより簡単にAjaxを実現できるようになったことで、さらに需要は高まり、XHRはユーザビリティを高めるうえで必須の技術となりました。

「サーバからクライアントへ情報を能動的に通知したい」というプッシュイベントに対するニーズが高まると、XHRはプロトコルの持つ機能の範疇を超え、無茶な使い方がされるようになりました。この無茶を体系化し、手法として確立させたものが「Comet」です。あとで紹介する「Polling」と区別するため、「Long Polling」

と呼んだりもします。

Cometは、HTTPリクエストのあと、クライアントへのプッシュイベントが発生するまでの間、HTTPレスポンスを返さずにコネクションを握り続けます。サーバはイベント発生の通知を受け取ったタイミングでHTTPレスポンスを返し、クライアントはすぐにまたHTTPリクエストを投げ、次の通知を待ちます(**図1**)。サーバは常に大量のステートレス通信用途のコネクションを抱え続けなくてはいけません。

「XHRによるPolling」と「Page Visibility」

Cometは技術的に正しくない、かなり無茶をしている技術です。そこで、サーバからの通知レベルくらいであれば、技術的に無茶をすることをやめて、擬似的に双方向通信を実現しようというアイデアも生まれます。「Polling」と呼ばれる方式です。

名前から察しがつくとは思いますが、Pollingは短い一定のタイミングでサーバにXHRで接続しにいくことで、擬似的にプッシュイベントを実現しようというアイデアになります。ステートレスな通信が適切なタイミングで切断されるため無理がありません。

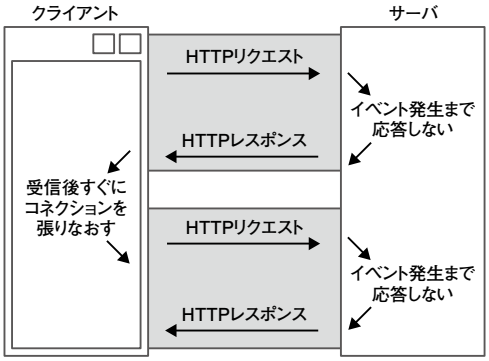
Pollingは技術的に正しく、またサーバミドルウェアも従来のHTTP1.Xを活用すれば良いので、既存のノウハウを活かせるかなり手堅い手段です。ただ、何の通知がない間も通信が発生するわけですから、効率はけっして良くありません。最近はスマートデバイスやノートPCなどのバッテリー稼働のハードウェアも多いので、無駄なエネルギーは極力使いたくないところでしょう。ここにちょっとした工夫があります。

Web標準「Page Visibility」は、Webページの表示／非表示、別のタブが開かれているなどを検知できるJavaScript APIです。スマートデバイス向けWindows OSで活用されるようなIE10以上や、最新のFirefoxやChrome、iOSのSafariでも動作するため、かなり実用的な技術と言えます。

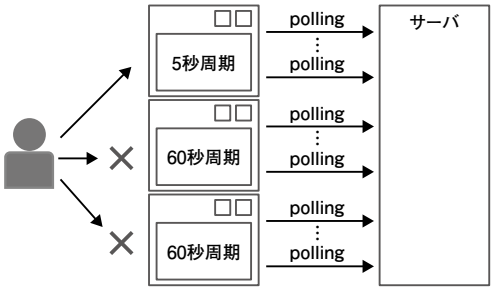
Pollingでは、このPage Visibilityを用いて、サーバに問い合わせる頻度を変えるのが一般的です(**図2**)。別のタブがアクティブなときにまで、ガンガン問い合わせを行うようなことはしません。バッテリーやネットワークリソースの節約で、わかりやすい定番の工夫と言えます。

リスト1は、Pollingによるプッシュ通知のシンプルな実装例です。「polling(①)」は、サーバへアクセスし、何か通知することがないかを確認する機能です。これを「adjustPollingCycle(②)」にて、Webページの表示／非表示を判定し、呼び出し間隔を指定しタイマーへセットしています。呼び出し間隔をセットするタイミン

▼図1　Comet



▼図2　Page Visibilityを用いたPolling



▼リスト1　Pollingの実装例

```
/* serverURLで指定したサーバから通知を受け取ると、callbackを呼び出す機能 */
function beginServerPush( serverURL, callback ) {

    var PERIOD_VISIBLE = 5000;      // 表示時のポーリング間隔(ミリ秒)
    var PERIOD_NOT_VISIBLE = 60000; // 非表示時のポーリング間隔(ミリ秒)

    var timer = 0; // タイマーオブジェクト
    var xhr = new XMLHttpRequest(); // XHRオブジェクト

    /* サーバへポーリングする機能を定義 */
    function polling() {
        xhr.onload = function() { // サーバから応答時の処理を指定
            if(xhr.status===200)  // HTTPステータスが200なら通知ありと判断
                callback(xhr.responseText);
        }
        xhr.open("GET", serverURL, true); // 接続オープン
        xhr.send(null); // サーバへアクセス
    }                                                              ①

    /* ポーリング間隔を切り替える機能を定義 */
    function adjustPollingCycle() {
        if( document.hidden ) {
            timer = setInterval(polling, PERIOD_NOT_VISIBLE); // 非表示時
        } else {
            timer = setInterval(polling, PERIOD_VISIBLE); // 表示時
        }
    }                                                              ②

    adjustPollingCycle(); // あらかじめポーリング間隔をセット          ③

    /* 表示・非表示の切り替わりを検知し、ポーリング間隔を変更する */
    document.addEventListener("visibilitychange", function() {
        clearTimeout(timer);  // タイマーのコールバックをクリアし
        adjustPollingCycle(); // ポーリング間隔をセットしなおす        ④
    });

}
```

グは、「機能自体の開始直後(③)」と「表示／非表示の切り替わり検知直後(④)」の2ヵ所です。

WebSocket

Pollingは、ステートレスな通信機構しか持たないHTTP1.Xを、擬似的にステートフルに扱うという強引なアプローチです。一定の間隔でサーバへアクセスして結果を取得しているのですから、リアルタイム性が低いのは明白です。

こうした問題を改善しようと、Web標準側では、HTTP1.Xのような従来の方法ではなく、専用のプロトコルを使おうという動きが生じました。その成果が「WebSocket」「ServerSent Events」です。本記事では、WebSocketを取り上げます。

リスト2はWebSocketの実装例です。Pollingでは一定の間隔でサーバにアクセスするため、その都度コネクションをオープンしていました。しかし、WebSocketは接続が切れないため、オープンは一度だけです。また、Pollingはサーバにアクセスしたときにしかサーバからの通知を受け取れませんでしたが、WebSocketではリアルタイムに通知を受け取れます。しかも、BLOB(Binary Large Object)型のストリーミングデータまで扱えるようにもなりました。とんでもない進歩です。

WebSocketは、ブラウザの対応機能に合わせてCometやPollingへフォールバックさせる

▼リスト2　WebSocketの実装例

```
/* serverURLで指定したサーバから通知を受け取ると、callbackを呼び出す機能 */
function beginServerPush( serverURL, callback ) {

    /* コンストラクタで、サーバ接続もまとめてオープンしてしまう */
    var websocket = new WebSocket( serverURL ); // WebSocketオブジェクト

    /* サーバからの通知を受け取った際の処理 */
    websocket.onmessage = function(e){
        callback(e.data);
    }

}
```

機能を有した、JavaScriptライブラリやフレームワークでラップして活用するのが一般的です。また、最近のサーバサイドのミドルウェアやフレームワークは、WebSocketをラップした新たなアーキテクチャ作りを模索し始めています。もはや、サーバサイドのトレンドの1つと言えるでしょう。

プロキシが超えられない、ミドルウェアが対応していないといった問題もありましたが、現在は改善が進み、インフラ全体で対応が追いつきつつあります。筆者の周りを見ていると、比較的制約の小さなイントラ向けのアプリケーションが先行して使い始め、徐々に広がっているという印象です。

ただ、本記事のメインテーマであるパフォーマンスの面では、テストツールが充実していないというのが難点に思えます。

KeepAliveとSPDY/HTTP2.0

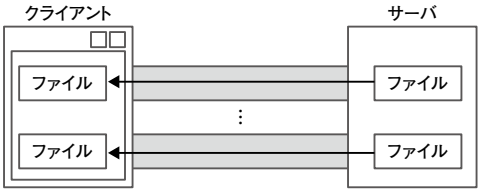
Webページの読み込みは、HTMLドキュメント内から参照された、JavaScript／CSS／画像などのファイルも同時に読み込む必要があります。Ajaxのように単一ではありません。最近のWebアプリは初期のWebで想定できないほどファイルサイズ／数が増えていますが、一方でWebを扱う環境／ハードウェアは多様化が進み、パフォーマンスの問題をより複雑にしています。こうした中、Webページの読み込みを改善するアプローチの1つとして、新たに作られたのが「SPDY/HTTP2.0」です。

過去のプロトコルから順を追って説明しましょう。HTTP1.0の時代は、1つのファイル取得に1つのコネクションを利用するという、原始的でシンプルなしくみでした(**図3**)。

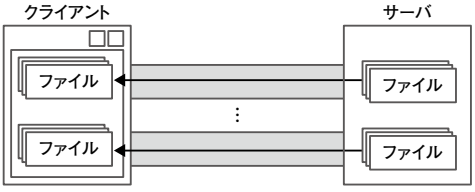
KeepAlive機能を有するHTTP1.1になると、1つのコネクションで複数回のファイル取得が行えるよう改善されます(**図4**)。ただ、この方法は通信負荷は下がりますが、コネクションが比較的長期に渡り握られることがあるため、サーバとしてもコネクションがFloodingするというリスクをはらんでいます。無闇に有効にできるものでもありません。

SPDYはHTTP1.Xからガラッと姿を変えます。SPDYはファイルをフレームという単位にバラして扱えるようにしたことで、1台のサーバあたりのコネクション数を1本にまで絞り、

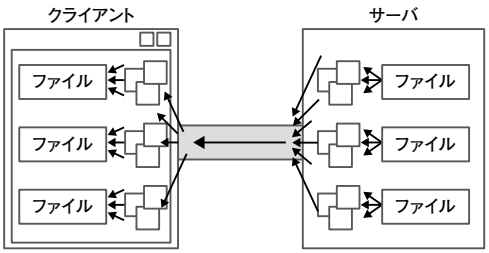
▼図3　HTTP1.0時代のファイル取得



▼図4　KeepAliveを利用したファイル取得



▼図5　SPDYを利用したファイル取得



それでいて従来のHTTPが持つ並列なファイル読み込みという機能性を損なわせないという、画期的なアプローチです(**図5**)。

しかし、これだけではただOSがやっていたことをミドルウェアがカバーしたにすぎません。当然ですが、もっといろんな機能を詰め込み、徹底した通信の改善を行っています。その1つとして、SPDYにはプッシュという機能があります。PollingやWebSocketのプッシュとは目的が異なり、今後リクエストされることがわかっているファイルをあらかじめ送信しておき、キャッシュさせるという機能です。通信の改善は、単純なプロトコルのしくみだけでは成り立たないため、SPDYではキャッシュのしくみにも踏み込んで改善を図っています。

SPDYで大きなパフォーマンス改善が期待できるかと問われると、筆者は「そんなに簡単なものではない」と答えています。残念ですが、Webページの読み込みは問題が複雑で、ミドルウェアの設定を変える程度でなんとかなるようなものではないのです。

たとえば、SPDYのコネクション数削減に目を向けてみましょう。従来のパフォーマンス改善といえば、サーバインフラへのアクセス集中を防ぐため、サーバを分散させるという対策が行われたはずです。インターネット向けサービスではCDN(Contents Delivery Network)による外部リソースを活用していることもあります。このようにサーバが複数にまたがると、コネクション数削減は意味を持たないでしょう。

古くからWebは強力なキャッシュ機能を持っており、ファイルの数が問題にならないこともあります。

そうなると、コネクションの1本化は、必ずしもメリットにはなり得ないでしょう。SPDYに限りませんが、プロトコルレベルでの改善は、サーバ負荷のスケールやアプリのアーキテクチャ側の対策があってこそ成立します。銀の弾丸はやはりここにもないようです。

キャッシュによるパフォーマンス改善

サーバ判定型のキャッシュ

SPDYの紹介の中でも触れましたが、キャッシュは相当昔から取り組まれている通信の改善技術です。プロトコルレベルの改善はたいへんですが、キャッシュはサーバのミドルウェアの設定だけでそれなりの改善が期待できます。ただ、キャッシュは古いファイルが参照されるという最悪の問題が起こらないよう、配慮することが求められます。プロトコルの改善に比べると、サーバ構成への配慮が小さい代わりに運用方法の工夫が求められ、極端なまでに悩みどころの違いを意識させられます。

まず、RFC2616で定義されているHTTPヘッダを用いたキャッシュのしくみについて紹介します。最初に断っておきますが、RFC2616のキャッシュは相互運用性がやや低めです。仕様として明確に決めていないものもあり、多くのブラウザがデファクトとしてこういう動きになる、ととらえるべきでしょう。ここではシンプルに理解するため、「サーバ判定型」と「ブラウザ判定型」の2つに分類して考えてみます。

最初に紹介するのは、「サーバ判定型」のキャッシュです。サーバ側にファイルの変更がないかを問い合わせて確認する方式になります。Apacheの場合、サーバ判定型はとくに設定を変更していなければ常に有効です。多くの場合、開発者は無意識のうちにこのパラメータの恩恵を受けており、トラブルもこのパラメータが要

因になることが多いでしょう。

サーバ判定型では、キャッシュの正当性を確認する際、ブラウザ側では何の判断も行いません。すべての判断は、一度サーバへ問い合わせを行い、サーバ内にて行われます。

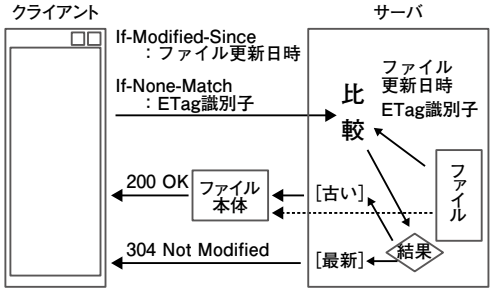
サーバでは、ブラウザから送られるHTTPリクエストヘッダ内の「If-Modified-Since」と「If-None-Match」の値を確認し、キャッシュの正当性を確認します。このIf-Modified-SinceとIf-None-Matchには、過去にサーバからファイルをダウンロードした際にHTTPレスポンスヘッダにセットされていた「Last-Modified(ファイル更新日時)」と「ETag(サーバによって設定される任意の識別子)」の値が、ブラウザによりそのままセットされています。

ブラウザからリクエストを受けると、サーバ内ではファイルを探してタイムスタンプなどを取得し、If-Modified-SinceとIf-None-Matchの値を確認します。値を比較し、異なれば「200 OK」を応答してファイルのデータが送られます。同じであれば、ブラウザのキャッシュは最新であると判断され、「304 Not Modified」を応答しファイルの本体は返しません(**図6**)。

Webサーバが1台であれば、Last-Modifiedだけで十分事足ります。しかし、複数のサーバでスケールする場合、ファイルのタイムスタンプが異なることがあるため、任意の識別子が指定できるETagのほうが有用です。

Apacheの場合はデフォルトで有効ですので、無効化するためにはmod_headersを利用することになります。**リスト3**が設定例です。

▼図6　サーバ判定型のキャッシュのしくみ



▼リスト3　Apacheでサーバ判定型キャッシュを無効にする

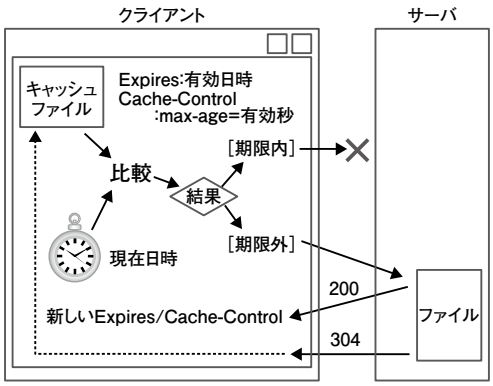
```
# Last-Modifiedを無効化
LoadModule headers_module modules/mod_ｱ
headers.so

RequestHeader unset If-Modified-Since
Header unset Last-Modified

# ETagを無効化
FileEtag None
```

ブラウザ判定型のキャッシュ

次に紹介するのは、「ブラウザ判定型」のキャッシュです。キャッシュされたファイルに変更がないかを、ブラウザ側で確認する方式になります。この方式では、基本的にキャッシュの正当性チェックのためにサーバへ問い合わせを行うことはしません。

ブラウザの内部では、キャッシュの有効期限を確認し、有効期限切れの状態、つまり「最新でない可能性がある」と判断されると、ブラウザはキャッシュを最新化しようとサーバへアクセスします(**図7**)。その際に、キャッシュは「サーバ判定型」のルールを適用します。変更がない場合は304で応答するため、やはり無駄なファイル読み込みは行われません。304応答時もHTTPレスポンスヘッダ内のルールは有効であるため、その後のキャッシュの正当性チェッ

▼図7　ブラウザ判定型のキャッシュのしくみ



▼リスト4　Apacheでブラウザ判定型キャッシュを有効にする

```
# 期限を1ヶ月に設定
LoadModule expires_module modules/mod_➥
expires.so
ExpiresActive On
ExpiresDefault "access plus 1 months" ←①
```

クでは、再び「ブラウザ判定型」のルールが適用されることになります。

ブラウザ判定型のキャッシュを制御するのは、HTTPレスポンスヘッダのうち、「Expires」「Cache-Control」の2つになります。この2つのパラメータは、セットととらえて良いでしょう。Apacheの「mod_expires」で設定を行うと、Apache側で自動的に計算され、両方が同時に有効になるためです。

例外的にChromeだけは、HTMLドキュメントやAjaxで取得するファイルにはこのルールを適用せず、常に「サーバ判定型」として動作します。IE、Firefox、Safariは「ブラウザ判定型」として動くため、相互運用性には注意を払わなくてはいけません。

また、ブラウザ判定型を適用している場合、動作テストに「更新ボタン」を使ってはいけません。アドレスバーにフォーカスを当て、Enterキーを叩くのがお作法です。更新ボタンでは、キャッシュは狙ったとおりの動作をしてくれません。動作テストなどでは、よくあるハマりどころです。

リスト4は、Apacheでキャッシュを有効にする例です。①の期間指定はApache側で計算され、ExpiresとCache-Control(max-age)の両方に自動設定されます。

有効にしたくない場合は、mod_headersで明示することをお勧めします(**リスト5**)。

AppCacheによるキャッシュ

最後に、HTML5より追加された「AppCache」によるキャッシュの価値について紹介します。

AppCacheは、サーバに接続が行えない環境

▼リスト5　Apacheでブラウザ判定型キャッシュを無効にする

```
# キャッシュを有効にしない
LoadModule headers_module modules/mod_➥
headers.so
Header set Cache-Control "max-age=0"
# ※ 過去の日付けを指定すること
Header set Expires "Mon, 26 Jul 1997
05:00:00 GMT"</pre>
```

下でも、キャッシュのみでアプリを動作させられます。このため、ブラウザで.NETやスマートデバイスのネイティブアプリのような、オフラインアプリケーションを作るための技術として注目を集めることが多いです。ただ、本記事はあくまでパフォーマンスがテーマですので、この点には深く触れません。RFC2616で扱ってきたようなキャッシュをアプリ側でより細かく制御できるというメリットに目を向けます。

AppCacheには、キャッシュを始める前、ダウンロード中、完了、サーバ側のファイルが更新されたなど、キャッシュを行う過程で生じるイベントを、JavaScript上で扱えます。モバイルの電波が入りにくいエリアや、ネットワークが細い環境下では、キャッシュのアップデートを後回しにしたいこともあるでしょう。AppCacheはこうしたニーズに応えられます。

図8では、キャッシュが古くなったと判断できた場合、勝手にサーバから取得するのでなく、一度ユーザにアップデートするかを問い合わせるフローとなっています。キャッシュの制御を、アプリの利用者に委ねているのです。

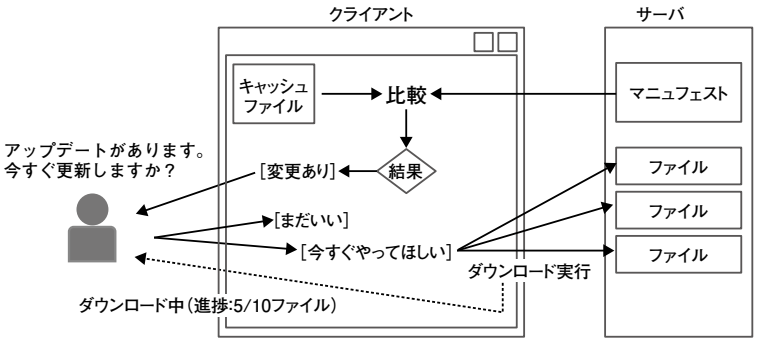
AppCacheはキャッシュする設定を、Apacheなどのミドルウェアに依存させません。キャッシュの設定ファイルを「マニュフェスト」と呼び、フォーマットまで細かく規定しています。キャッシュが動作していることを、サーバ側に意識させることはありません。

リスト6はマニュフェストの書き方の例です。これはApacheではなく、ブラウザ側が直接読み込む設定情報になります。マニュフェストファイルは、**リスト7**のようにHTMLドキュメン

ト上から参照します。

マニュフェストからキャッシュが古いことがわかり、ユーザへ確認するまでのJavaScript側の処理は、**リスト8**のとおりです。説明のために簡易化していますが、実際に使う際はupdatereadyイベントの発火時に、この処理を行うと良いでしょう。

▼図8　AppCacheによるキャッシュのしくみ



まだまだ続きます!

今回は通信周りへのアプローチについて紹介しましたが、いかがでしたでしょうか。ほとんどがインフラレイヤの話でした。たとえクライアントであっても、パフォーマンスはアプリ開発者だけでは十分な改善が期待できないことがご理解いただけたかと思います。アプリがパフォーマンスを悪化させる特徴的な機能を持つ場合、インフラエンジニアも巻き込んでいかないことには、最適な手段へたどりつけないのです。

さて、次回はいよいよ最終回です。パフォーマンスの問題を発見するためのWeb標準を紹介します。第1回で紹介した、Web Performance Working Groupの本業の部分になります。どうぞ、ご期待ください！SD

▼リスト6　マニュフェストの例

```
CACHE MANIFEST

# キャッシュするファイルの一覧
CACHE:
index.html
js/sample.js
css/sample.css

# オンラインで取得したいファイル
NETWORK:
*

# 失敗時に表示する代替ファイル
FALLBACK:
offline.html
```

▼リスト7　HTMLにおけるマニュフェストファイルの指定

```
<!DOCTYPE HTML>
<html manifest="appcache.manifest">
<head>
-- 以下略 --
```

▼リスト8　キャッシュが古い場合に、ユーザに確認してから最新化する機能

```
function checkCacheUpdate() {

    var appCache = window.applicationCache; // AppCacheオブジェクト取得

    // キャッシュが古いかを確認する
    if(appCache.status === appCache.UPDATEREADY){
        if (confirm("アップデートがあります。¥n今すぐ更新しますか？")) {
            appCache.swapCache(); // キャッシュを更新
            window.location.reload(); // Webページをリロード
        }
    }
}
```

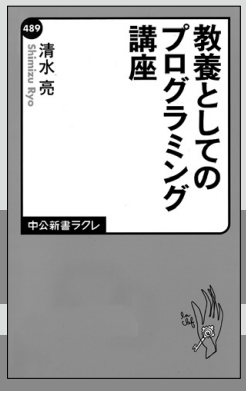
BOOK no.1 [第3版] Cisco LANスイッチ教科書

シスコシステムズ合同会社 基盤技術グループ 【著】
B5変形判、 400ページ／価格＝4,200円＋税／発行＝インプレスジャパン
ISBN＝978-4-8443-3564-1

本書は「LANとは何か」といった初歩的内容から始まるが、それでいて細部までしっかり解説をしてくれる。プロトコルの説明ならパケットのフォーマットまで、スイッチの説明ならその内部構造まで、解説を行う。スイッチの機能についてもネットワーク構成図だけでなく、Cisco社のスイッチでの設定例やコマンド実行例とともに解説される。ネットワーク機器は同じ機能でもメーカーによって設定方法や機能名に微妙な違いがある。本書はCisco製品固有の特徴についても言及してくれている。新人ネットワークエンジニアは、一般的な入門書よりも、このように製品に特化した本のほうが、現場ですぐに役立つ知識が得られそうだ。

BOOK no.2 教養としてのプログラミング講座

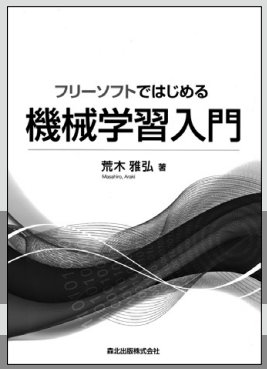
清水 亮 【著】
新書判、 192ページ／価格＝780円＋税／発行＝中央公論新社
ISBN＝978-4-12-150489-0



プログラミングをひとつの「教養」として学べる本。日常の作業をアルゴリズム化するプログラマの思考法、名簿作りにハッシュテーブルを使うといったテクニックなど、実生活に応用できる知識が豊富に紹介されている。「お使いの頼みかた」ひとつをとっても、伝え洩らさない、あいまいな表現を省く、条件を分岐させるなど、守るべきプログラミングの鉄則が多く存在するのである。また、コンピュータの歴史に関するコラムや、直観的に操作できる「moonblock」を使った、簡単なプログラミングの実習も設けられている。本書を読んで、自分の生活をプログラミングの観点から捉えなおしてみてはいかがだろうか。

BOOK no.3 フリーソフトではじめる機械学習入門

荒木 雅弘 【著】
菊判、 272ページ／価格＝3,600円＋税／発行＝森北出版
ISBN＝978-4-627-85211-2



フリーのデータマイニングソフト「weka」での実習を交えた、機械学習の入門書。各章に演習問題が設けられており、手を動かしながら勉強が進められる。昨今話題のビッグデータの分析には、データに正解がついているかどうか（教師あり／なし）、学習の手法の選択（識別、回帰、モデル推定、等）の判断が最も重要となる。データの収集から各手法について解説した本書は、機械学習を学び始める人にとっておすすめの教科書だ。高度な題材を扱っているため内容はやや難しく、数学と統計の知識を身につけてから読み始めたい。機械学習の結果が実際の分野にどんな影響を与えるかを考えながら読むと、理解が早まるだろう。

BOOK no.4 プログラミングPHP [第3版]

Kevin Tatroe, Peter MacIntyre, Rasmus Lerdrof 【著】 ／ 高木 正弘 【訳】
B5変形判、 416ページ／価格＝3,800円＋税／発行＝オライリー・ジャパン
ISBN＝978-4-87311-668-6

PHP開発者のラスマス・ラドルフらによって書かれた定番ガイドブック。言語の動作原理、構成要素から、拡張モジュールまで網羅した本書は、優れたPHPプログラマになるための必読の1冊となっている。

第3版では、「PHP5.4」をおもなバージョンとしてカバーし、データベースアクセス関連の記述が大幅に書き換えられた。PHPバージョン5から導入されたSQLiteについては、多くのサンプルコードとともに、非常にわかりやすく記述がされている。第2版で扱われた「PHPの拡張」に関する記述が省かれ、新たに「ウェブサービス」、「PHPのデバック」、「日付と時刻」の章が追加された。

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

本誌バックナンバー（紙版）はお近くの書店に注文されるか、下記のバックナンバー常備取り扱い店にてお求めいただけます。また、「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）や、e-hon（http://www.e-hon.ne.jp）にて、Webから注文し、ご自宅やお近くの書店へお届けするサービスもございます。

2014年5月号

第1特集
ネットワーク・ビギナー向け基礎講座
「ポート」と「ソケット」がわかれば TCP/IPネットワークがわかる！

第2特集
UNIX必須コマンドトレーニング
rmコマンドからcadaverまで基本を押さえる

一般記事
・Rettyのサービス拡大を支えた「たたき上げ」DevOps
・Webアプリのパフォーマンス改善 ほか

定価（本体1,220円+税）

2014年4月号

第1特集
<Java/JavaScript/PHP>言語別で考える
なぜMVCモデルは誤解されるのか？

第2特集
ネットワークエンジニア養成
ロードバランサの教科書

一般記事
・さらに踏み込む、Mac OS Xと仮想デスクトップ #2
・SIMのしくみ

定価（本体1,219円+税）

2014年3月号

第1特集
データベースの諸問題
RDBとNoSQL どちらを選びますか？

第2特集
ネットワークエンジニアのための
プロキシサーバの教科書

一般記事
・さらに踏み込む、Mac OS Xと仮想デスクトップ #1

定価（本体1,219円+税）

2014年2月号

第1特集
λ式からはじめませんか？
関数型プログラミング再入門

第2特集
目利きによるトレンド予測
2014年IT業界はどうなるのか？

一般記事
・会社組織を活性化するスパイス「コンパ」

定価（本体1,219円+税）

2014年1月号

第1特集
シェルがわかればシステムがわかる
あなたの好きなシェルは何ですか？

第2特集
未来を作るITインフラ
10ギガビットを実現するケーブリング技術

特別付録 & 一般記事
・法輪寺鎮守社電電宮 情報安全護符シール Ver.2
・ソーシャルゲームのDevOpsを支える技術（後編）

特別定価（本体1,314円+税）

2013年12月号

第1特集
SDN/OpenFlowの流れを総まとめ！
SDN/OpenFlowで幸せになれますか？

第2特集
下手でも好印象で効果絶大
エンジニアの伝わる図解術

一般記事
・LinuxとFreeBSDのファイルシステムの良い・悪いところをご存じですか？
・「Mirama」ほか

定価（本体1,219円+税）

Software Design　バックナンバー常備取り扱い店							
北海道	札幌市中央区	MARUZEN & ジュンク堂書店　札幌店	011-223-1911	神奈川県	川崎市高津区	文教堂書店　溝の口本店	044-812-0063
	札幌市中央区	紀伊國屋書店　札幌本店	011-231-2131	静岡県	静岡市葵区	戸田書店　静岡本店	054-205-6111
東京都	豊島区	ジュンク堂書店　池袋本店	03-5956-6111	愛知県	名古屋市中区	三洋堂書店　上前津店	052-251-8334
	新宿区	紀伊國屋書店　新宿本店	03-3354-0131	大阪府	大阪市北区	ジュンク堂書店　大阪本店	06-4799-1090
	渋谷区	紀伊國屋書店　新宿南店	03-5361-3315	兵庫県	神戸市中央区	ジュンク堂書店　三宮店	078-392-1001
	千代田区	書泉ブックタワー	03-5296-0051	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	千代田区	丸善　丸の内本店	03-5288-8881		広島市中区	丸善　広島店	082-504-6210
	中央区	八重洲ブックセンター本店	03-3281-1811	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322
	渋谷区	MARUZEN & ジュンク堂書店　渋谷店	03-5456-2111				

※店舗によってバックナンバー取り扱い期間などが異なります。在庫などは各書店にご確認ください。

デジタル版のお知らせ　DIGITAL

デジタル版Software Design 好評発売中！紙版で在庫切れのバックナンバーも購入できます

本誌デジタル版は「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）と、「雑誌オンライン.com」（http://www.zasshi-online.com/）で購入できます。最新号、バックナンバーとも1冊のみの購入はもちろん、定期購読も対応。1年間定期購読なら5%強の割引になります。デジタル版はPCのほかにiPad／iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく、読むことができます。

思考をカタチにするエディタの使い方

るびきち流 Emacs超入門

Writer | るびきち
twitter@rubikitch
http://d.hatena.ne.jp/rubikitch/
http://www.rubyist.net/~rubikitch/

第2回 スマートにEmacsを始めるための小さいけど重要なコツ

Emacsを使う最大のメリット、それは文字入力が一元化できることです。メールもファイル操作も、Twitterも、アプリケーションを切り替えることなく、スムースに操作できるというものです。しかし、ビギナーがそこまで使うことができるようになるには、手順を踏んで、スジ良く使い方を学ぶことが必要です。本稿を手がかりにしてください!

さぁ、Emacsを使ってみよう

前回はEmacsとは何なのかということを説明してきました。表面的にはテキストエディタの顔をしていますが、実際はEmacs Lispによりコントロールされている「世界」です。Emacs Lispを書けば、テキストエディタとしての挙動をカスタマイズできるのはもちろんのこと、Emacs上で動くアプリケーションをも作れます。メーラ、ファイラ、シェル、Twitterクライアント、ゲームなどがEmacs Lispで書かれています。また、Emacsは外部プログラムと連携するのがとても得意で、シェルを動かしたり、コンパイルエラー行やgrep検索結果にジャンプすることもお手のものです。

文字入力の一元化

Emacsを使うことにおけるユーザサイドの最大のメリットというのは、なんといっても「文字入力の一元化」です。もともとテキストエディタなのでプログラミングや文書作成にEmacsを使うことはもちろんできます。なにより嬉しいのが、Emacsアプリケーションと日常のプログラミングで操作を統一できることです。

Emacsアプリケーションは多岐にわたりますが、ほぼどれもが文字入力を伴います。メーラでメールを書いたり、IM(Instant Messenger)のメッセージを書いたり、ブラウザのフォームに入力することなど、ありとあらゆる文字入力をEmacsでまかなえます。もし目的を達成するEmacsアプリケーションがあれば、ほかのソフトを使わずにEmacsでやってしまう方法があります。そのEmacsアプリケーションの使い勝手はともかく、文字入力をする場面においてはいつものEmacsの操作で行えます。この安心感はとてつもなく大きいものです。

たとえば、Webメールではブラウザで用意されたインターフェースでメールを書くことができます。対してEmacs Lispで書かれたメーラを使えば、Emacsから出ることなくメールが書けます。メールの内容を入力するときも、高度なコマンドがたくさん使えるEmacsならばストレスフリーです。見慣れた背景色やフォントなので目にも優しいです。それこそが筆者がEmacsを溺愛している理由なのです。

それでは、文字入力一元化のメリットを享受するために、テキストエディタEmacsの基本的な概念を見ていきましょう。確かにEmacs Lispを書けば操作方法は好き勝手に変更できます。しかし根底にはEmacs流のやりかたというのがあり、それを無闇に変更するとひずみが出てきかねません。Emacs入門者は基本的な操作方法をきっちり学んでおきましょう。カスタマイズ

するのは基本ができてからで遅くありません。

Emacsでの効果的な編集方法を学んでおくことは投資効果が大きいです。筆者は20年近くEmacsを使っていますが、最初のころに覚えた操作方法は今でも使っています。数ヵ月や数年使っていれば無意識に手が動くレベルになってきます。つまり、Emacsのスキルというのは一度覚えれば長年使い続けることができます。わずか数年で時代遅れになってしまうほど移り変わりの激しいIT業界において、長年使い続けられるテキストエディタの基礎スキルというのは異色を放っています。

さあ、始めましょう

Emacsはシェルから「emacs」と引数をつけずに実行すれば立ち上がります。そして、終了は Ctrl + X の後に Ctrl + C (C-x C-c)を押してください。

本連載では入門者や初級者でもEmacsの魅力に触れていただけることをコンセプトとしています。そのため、しばらく中級者以上には釈迦に説法になりますがご容赦ください。さあ、始めましょう。

キー操作の表記法

Emacsの世界では、キー操作を記述する独特の記法が存在します。Emacsでは Ctrl や Alt を多用するため、なるべく簡潔な記法が求められてきたからです。

テキストエディタとしての最も基本的なコマンドはファイルを開くことですが、Ctrl + X の後に Ctrl + F を押します。これを「Ctrl+X Ctrl+F」と書くと長くなるので、Emacsでは縮めて「C-x C-f」と書きます。「C-」というのは Ctrl を押しながらを意味するEmacs記法です。

画面を逆スクロールするのは、Alt + v を押します。これもEmacs記法で「M-v」と書きます。「A-」ではなくて「M-」となっているのは歴史的理由です。かつてのキーボードにはメタキーというのが存在していて、現在では Alt キーに相当します。Emacsでメタキーと言えば Alt キーのことです。

Ctrl と Alt を同時に押すのは「C-M-」あるいは「M-C-」と書きます。たとえば、Ctrl+Alt+x は「C-M-x」となります。

このEmacs記法はEmacs関連の文書で使われているだけでなく、Emacs記法を解釈するLisp関数も存在します。つまり、Emacs Lisp内でEmacs記法を使ってキー割り当てを記述することもできます。

ファイル・バッファ・ミニバッファ

Emacsを使うということは、バッファを編集することにほかなりません。バッファとはテキストを入れる器のことです。ファイルを編集するということは、ファイル名に関連付けられたバッファを編集することそのものです。

このように、Emacsではファイルとバッファを切り離しているので、ファイル編集以外のことができます。ファイルと関連付けられていないバッファには、シェルコマンドの実行結果やヘルプを表示したり、メモやブラウザのフォーム入力などがあります。

ファイルを開くと、それに対応するバッファが作られます。複数のファイルを同時に開いて編集できますが、一度ファイルを開いてしまえば、バッファ名の先頭数文字を指定するだけで切り替えられます。すでに開かれているファイルをふたたび開こうとすれば、対応するバッファに切り替わります。

「ミニバッファ」という特別なバッファがあります。それは、Emacsがユーザに情報を尋ねるときに、プロンプトとともに画面下部に出てくる領域です。GUIだとダイアログボックスが使われるところですが、Emacsはシンプルです。C-x C-fでファイルを開くとき、ミニバッファでファイル名を聞いてきます。C-x bでバッファを切り替えるとき、ミニバッファでバッファ名

を聞いてきます。M-xの後はコマンド名を聞いてきます。

ファイル名やバッファ名を聞かれたとき、フルネームを入力する必要はありません。最初の数文字を入力したあとにTabを押せば足りない文字列を補ってくれます。それを「補完機能」といいます。補完候補が複数個ある場合は補完候補を教えてくれます。

コマンドの覚え方

Emacsには無数のコマンドが存在します。長年にわたってEmacsを使っている筆者でさえも、知らないコマンドはたくさんあります。

詰め込み教育に慣らされた日本人は、まず使い方をたくさん勉強してから使うような傾向があると思います。Emacsはたしかに複雑で多機能ですが、そこまで肩肘張ってやる必要はありません。なぜコマンドがたくさんあるかを理解すれば、必要なコマンドが自然と身に付いていきます。

逆説的な言い方ですが、ただ単にEmacsを使いたければコマンドなど覚える必要はありません。ファイルはツールバーやメニューから開けますし、カーソル移動はマウスや矢印キーも使えます。コピー&ペーストなどの基本的な編集操作もマウスから行えます。このようにごくごく普通のテキストエディタとしてEmacsは使えます。

最初の取っ掛かりとしては、マウスや矢印キーでもよいです。でも、わざわざ本記事を読んでEmacsに入門しようとしているあなたはそれで満足ですか？　ホームポジションから手を離して、マウスに手をやって、再び手をキーボードに戻すことにあなたは耐えられますか？　面倒だと思うはずです。面倒なことはなるべく避けてより効率的に操作したくなるものです。GUIでもマウスでコピー&ペーストするのは面倒なのでショートカットキーがありますよね。

そこで基本的なコマンドを覚えれば、Emacsはもっと使いやすくなります。面倒なことを回避するという明確な目的意識さえあれば、それで十分です。

カーソル移動も1文字ごとに動かす(C-b、C-f)以外に、行単位(C-p、C-n)・単語単位(M-b、M-f)で動かしたり、行頭・行末(C-a、C-e)、バッファ先頭・バッファ末尾(M-<、M->)に移動できます。なぜそれらのコマンドがあるのでしょうか？　1文字ごとに動かしていては、遅すぎるからです。目的地にスパッとカーソルを移動させたいからこそたくさんのコマンドが存在するのです。

このように多くのコマンドを知れば知るほどEmacsは使いやすくなってきます。コマンドを覚えなくても一応使えますが、覚えるにこしたことはありません。しばらくコマンドを使っていれば無意識に手が動くようになります。慣れるまでは不自由ですが、1年使えば、Emacsが手足の一部になった感覚になることでしょう。

覚えることが少ないことと、使いやすいということはまったく別です。覚えることが少ないと確かに簡単に使えるものの、しばらくしたら操作が面倒になってきます。使える操作が少なすぎるのは不便です。より便利に使うためにコマンドを覚えるのです。

多くのコマンドを知れば知るほどEmacsは使いやすくなりますが、普段使わないコマンドを無理矢理詰め込んで覚えるのもよくありません。使わないものはあっさり忘却曲線の彼方へ飛んでいってしまいます。ある操作をしたときに不便だと思ったその時こそ、よりよい方法・コマンドを覚えるチャンスです。

禁断の果実cua-mode

現在のGUI環境でのショートカットキーはだいたい統一されています。たとえば、C-cでコピー、C-xでカットみたいな感じになっていますね。そのため、使い慣れているいつものショートカットキーをEmacsで使いたいという欲求も

出てきます。

これを実現するのがcua-modeです。CUA(Common User Access)というのは、GUIでの操作を決めている規格であり、その規格に従うことでアプリケーションごとに操作方法を覚える必要をなくすものです。おかげさまでWindowsでもMacでもUnixでも同じ操作が使えます。CUAが制定される以前は操作体系が乱立していました。Emacsの歴史はCUAより古いので現代人から見て変則的なキーバインドになっています。

M-x cua-modeを実行すると、Emacs本来の操作体系に加え、CUAの操作体系が使えるようになります。

・C-z：undo
・C-x：cut
・C-c：copy
・C-v：paste

さらに、Shiftを押しながら矢印キーを押せばその部分を範囲選択できますし、範囲選択時に文字キーを打てば、その部分が置き換わります。つまりEmacsがよりいっそう「普通の」エディタとしてふるまうようになります。

しかし残念なことにEmacsとCUAは本質的に相容れない操作体系です。C-vは本来次ページにスクロールをするコマンドで、Emacsチュートリアルで最初に出てくるほど超初歩的なものですが、cua-modeではペーストに変更されてしまいます。EmacsにおいてC-xとC-cは2ストローク以上のコマンドのためにすでに使われています。そこで、cua-modeでは強引な方法でEmacsとCUAの操作体系を共存させています。

1.範囲が選択されているときのみカット・コピーを行う

2.C-xやC-cから始まり、範囲に対して作用するコマンドは次の方法で動作する
・0.2秒以内に次のキーを打つ
・C-x、C-cの代わりにShiftも押してC-S-x、C-S-cと打つ
・C-xの代わりにC-x C-x、C-cの代わりにC-c C-cと打つ

2のケースに該当するコマンドはさほど多くないので、cua-modeはなんとか使いものになるでしょう。しかし、「郷に入っては郷に従え」という言葉があります。あえてEmacsを選んだ以上、Emacs本来の方法から学ぶべきです。

EmacsにはCUAのショートカットキーをはるかに超えるコマンドがあり、高度な編集作業も楽々こなす力があります。あらゆる入力作業をEmacsで行うようになれば、CUAに執着する必要はなくなります。

cua-mode自体が多機能で便利な機能もありますが、Emacsの挙動を大幅に変更してしまいます。よって、初心者救済的な位置付けと思ってください。

チュートリアル

Emacsには最初から優れた日本語チュートリアルが存在します。実践形式のテキストになっているので、これを一通りやればEmacsの操作体系の概要が身に付きます。<f1> t(F1を押したあとにtを押す)を押せば始められます。

基本的なコマンドを効率的に学ぶには、チュートリアルをやりながら御自身でチートシートを作ることをお勧めします。コマンドは実際に手を動かして学ぶのが一番です。そのため、あえてコマンド一覧を本稿に載せませんでした。

もっとEmacsについて学びたいのであれば、筆者のEmacs専門メルマガ[注1]に登録してみませんか？　無制限メール相談権付きであなたを徹底サポートいたします。Happy Emacsing！SD

注1）http://www.mag2.com/m/0001373131.html

シェルスクリプトではじめるAWS入門

――ゼロから始めるAWS API

第3回　AWS利用環境の設定（前編）CloudTrail & IAM

Writer　波田野 裕一（はたの ひろかず）／運用設計ラボ　operation@office.operation-lab.co.jp

トピック

AWSには、サービス利用上のすべての権限を持つ「AWSアカウント」と、利用者認証やアクセス権限管理のサービスであるAWS Identity and Access Management（以下IAM）上に作成する「IAMユーザ」の2種類のアカウント／ユーザ管理機構があります。

これまではAWSマネジメントコンソール（Webインターフェース、以下「マネジメントコンソール」）やAWS SDK（Sytem Developer Kit）、AWS CLI（Command Line Interface）などを利用したすべての作業をAWSアカウントで行うことが可能でしたが、2014年4月21日以降はAWSアカウントのシークレットアクセスキーの取得ができなくなる旨の公式アナウンスが2014年3月に告知されました[注1]。

シークレットアクセスキーは、AWS CLIやAWS SDKなどでAPIにアクセスする際に必要となるものですので、マネジメントコンソールしか使わない場合は何ら影響ありません。また、AWS CLIやAWS SDKなどを利用している場合でも、すでにシークレットアクセスキーを取得している場合はただちに影響があるわけではありません。しかし、AWSアカウントは、支払い方法など契約情報の管理権限を持ち、アカウント自体の解約も可能であるなど、作業用として日常的に利用するには権限が強過ぎます。今後は契約管理はAWSアカウントで、AWSプロダクトの操作はIAMユーザでそれぞれ行うことが従来以上に一般的になると思われます。これを機会にIAMに親しんでしまいましょう。

注1）URL http://aws.typepad.com/aws_japan/2014/03/important-aws-account-key-change-coming-on-april-21-2014.html

今回の流れ

今回と次回の2回にわたって、AWSアカウントの作成と、AWS APIを快適に利用するうえでやっておくと良い下記の4つの作業について解説します。

1.AWS CloudTrail（APIのロギング）の設定

AWSアカウントを作成したら、まず最初にAWS APIに対する作業ログを保存するためにAWS CloudTrail（以下「CloudTrail」）の設定をします。

2.作業用IAMユーザの作成

次に、AWSプロダクトのAPIを操作するために、IAMユーザを作成し、そのユーザのアクセスキーIDとシークレットアクセスキーを取得します。

3.請求関連の設定

AWSでは月次の請求書をメールで受け取ったり、詳細な請求レポートをAmazon Simple

Storage Service（以降「S3」）に保存するしくみを提供しています。これらはデフォルトでは無効になっているので、アカウント作成直後に有効にしておきます。また、AWSでは請求金額が想定金額を超えたときにアラートを上げるしくみが用意されています。請求金額にビクビクしながら使うのは不安だと思いますので、併せてここで設定しておきましょう。

4. 多要素認証（MFA）の設定

AWSは多要素認証（Multi-Factor Authentication）に対応しています。MFAは、AWSのサービスにアクセスする際に、専用デバイスもしくはスマートフォンのソフトウェアなどにより生成されるワンタイムの認証コードを要求する認証方式で、なりすまし防止効果の向上が期待できます。AWSアカウントや権限の強いIAMユーザについてはなるべくMFAの設定を有効にしておくようにしましょう。

AWSアカウントの作成

最初に、新規にAWSアカウントを作成します。すでにAWSを利用中でAWSアカウントをお持ちの場合はそちらを利用していただいても良いですが、とくに理由がない場合は別途新規に作成することを勧めます（右下コラム参照）。

AWSアカウントの作成には次の3つが必要になります。登録作業前に用意しておくようにしましょう。

・メールアドレス（ログインIDになります）
・クレジットカード
・電話番号（身元確認のために電話がかかってきます）

最新のAWSアカウント作成手順については、公式ページ[注2]を参照してください。

おおまかな流れは下記のようになります。

ステップ1：サインイン／AWSアカウント作成画面
ステップ2：AWSログイン情報の入力
ステップ3：ユーザお問い合わせ情報の入力
ステップ4：クレジットカード情報の入力
ステップ5：自動音声電話による身元確認
ステップ6：AWSサポートプランの選択

注2）URL http://aws.amazon.com/jp/register-flow/

利用するリージョンについて

AWSでは、ある地域に配置されたデータセンターの集合体を「リージョン」と呼び、AWSユーザであれば、米国政府向けの特殊なリージョン（AWS GovCloud（US））を除く世界8ヵ所のリージョンを利用できます（2014年4月現在）。本連載では、基本的にバージニアリージョン（us-east-1）を利用します。これは下記が理由となります。

・全リージョンでバージニアリージョンがもっとも新サービスへの対応が早い
・コストが安い。Amazon EC2のt1.micro（Linux）インスタンスの場合、東京（0.027ドル／時）に比べてバージニアとオレゴン（0.020ドル／時）は約26%安くなる
・CloudTrailが提供されているのがバージニアとオレゴン（us-west-2）だけである（2014年4月中旬現在）

とくに、CloudTailについては複数の開発者でAPI利用する場合には欠かせないサービスと言えると思います。東京リージョンでの早期のリリースを期待したいと思います。

AWSアカウントの複数利用

AWSでは、複数のAWSアカウントの料金を特定のAWSアカウントに対して一括請求するサービス（Consolidated Billing）を提供しているほか、何らかのトラブルがあった場合にAWSアカウント単位で制限や停止が行われることから、AWSアカウントをプロジェクトや用途ごとに作成することが一般的に行われています。

サポートプランについては後日変更が可能ですので、アカウント作成時点では無料の“Basic”を選択しておいて問題ありません。AWS Trusted Advisor(AWSが提供する自動診断ツール)を利用したくなったり、有料レベルのサポートを受ける必要が発生したときにプランの変更をすれば通常は十分です。

CloudTrailの設定

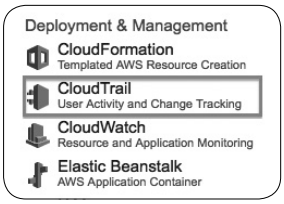
ログ取得の有効化

AWSアカウントの作成が完了したら、さっそくAPIの操作ログを保存するためにCloud Trailの設定をします。

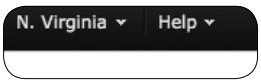
手順①

CloudTrailマネジメントコンソールにアクセスします(**図1**)。マネジメントコンソールのトップページの[Deployment & Management]の欄にリンクがあります。

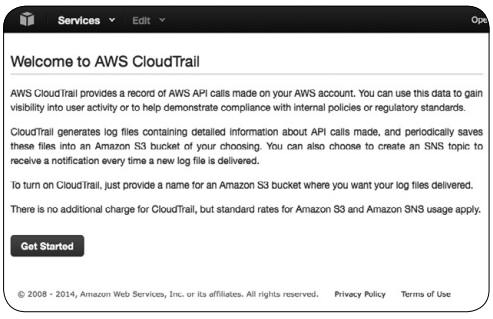
▼図1　マネジメントコンソール上のリンク(CloudTrail)



▼図2　バージニアリージョンであるか確認(画面右上)



▼図3　[Get Started]をクリック



手順②

右上のHelpの左隣にリージョンの表示があります。ここが“N.Virginia”になっていることを確認しておきましょう(**図2**)。

手順③

[Get Started]ボタンをクリックします(**図3**)。

手順④

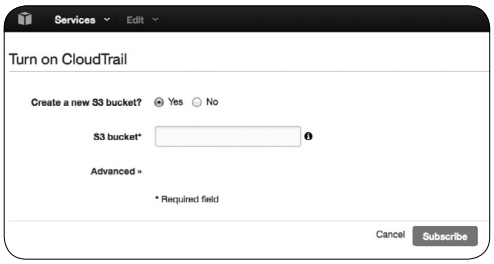
CloudTrailは、S3にログファイルを保存します。[S3 bucket]欄にAPI操作ログを保存するためのバケットの名前を入力してください。バケットの名前はS3全体でユニークである必要があるので、たとえば「組織名-trail」というような名称を入力します(**図4**)。

> **バケット**
>
> S3においてファイルやディレクトリを入れる箱のようなもので、一番大きな管理単位となるものです。日本語ではバケツと発音されることが一般的です。

手順⑤

[Subscribe]ボタンをクリックすると、Cloud Trail Configuration画面が表示されます(**図5**)。画面の右側に「Logging is on」と表示されてい

▼図4　S3バケット名の入力



▼図5　ロギングの開始

れば、これ以降AWS APIに対する操作ログがS3上のバケットに保存されていきます。

設定後しばらくして画面をリロードすると、[Logging is on]の下に「Last log file delivery:」という文字列と日時情報が表示されます（**図6**）。これは、先ほどのCloudTrailを有効にしたときの操作ログがさっそくS3に保存されたことを示しています。

現時点ではベータ版のサービス

CloudTrailは現時点でベータ版であり、操作ログの保存を保証していないことと、AWS APIへの操作をしてから実際にバケット上にログが保存されるまでにはタイムラグがあることに留意してください。

操作ログの確認

手順①

実際にS3にアクセスして操作ログを見てみましょう。S3マネジメントコンソールにアクセスします。トップページの[Storage & Content Delivery]の欄にリンクがあります（**図7**）。」

手順②

S3マネジメントコンソールのバケット一覧に先ほどのCloudTrail設定で指定したバケットが表示されているので、クリックします。現時点でほかにログがないはずですので、どんどん下層のディレクトリに移動していきましょう。

すると、“All Buckets /[バケット名]/AWSLogs/[アカウントID]/CloudTrail/[リージョン]/[年]/[月]/[日]”といったパスにたどり着きます。しばらくして画面をリロードすると、gz形式のファイルが一覧に表示されます。

▼図6　操作ログの最終保存時刻が表示される

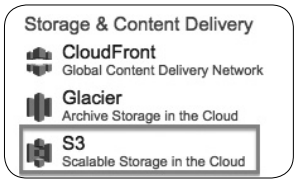
手順③

ファイルの先頭のチェックボックスにチェックを入れて、画面の上方にある[Actionsボタン]→[Download]の順に選択すると、ファイルをダウンロードするためのリンクが表示されるのでクリックしてダウンロードします。

手順④

ログファイルを開くと、ログデータはJSON形式になっており、先ほどのログ取得の有効化をしたときの操作について**リスト1**の情報が記録されていることがわかります。“eventSource”に操作対象となったAWSプロダクト名、“eventTime”に操作がされた日時（グリニッジ標準時：GMT）が記録されています。そしてこのログから、“userAgent”によりAWSマネジメントコンソールから操作が行われたこと、“eventName”によりロギング開始の操作が行われたこと、“userIdentity”により誰がその操作

▼図7　マネジメントコンソール上のリンク（S3）



▼リスト1　JSON（整形して一部抜粋）

```
{
  "Records" : [
    "eventSource" : "cloudtrail.amazonaws.com",
    "eventTime" : "2014-04-18T05:44:51Z",
    "userAgent" : "AWSConsole, aws-sdk-java/1.4.5 Linux/2.6.18-348.16.1.1123.21.fleetxen Java_HotSpot(TM)_64-Bit_Server_VM/24.45-b08",
    "eventName" : "StartLogging",
    "userIdentity" : {
    "accountId" : "XXXXXXXXXXXX",
    ……（以下省略）……
```

をしたか、などがわかるようになっています。

IAMユーザの作成

APIのログが取れるようになったら、次はAPIを実際に使えるようにするためにIAMユーザを作成します。今回は、AWS APIの動作を確認するため最も広い権限のIAMユーザを作成しますが、実運用環境では適切な範囲に限定した最小限の権限を付与するようにしてください。

IAM ユーザの作成

詳細はIAMの使用(ユーザガイド)の「最小限の特権を認める。」の項を参考にしてください(http://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/IAMBestPractices.html#grant-least-privilege)

手順①

IAMのマネジメントコンソールにアクセスします(**図8**)。マネジメントコンソールのトップページの[Deployment & Management]の欄にリンクがあります(先ほどのCloudTrailの下のほうにあります)。

手順②

[Create a New Group of Users]ボタンをクリックします(**図9**)。

手順③

最初のステップではグループ名を指定します。ここではたとえば管理者グループを意味するwheelという名前にします(**図10**)。

手順④

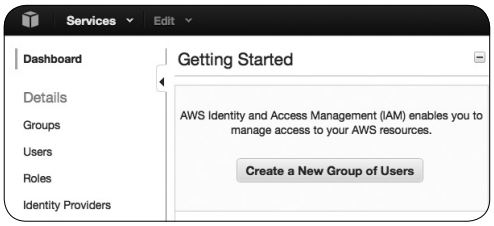
次に権限を指定します。ここでは、IAMユーザの作成権限を含む最も強い権限であるAdministrator権限を付与するので、[Administrator Access]の欄にある[Select]ボタンをクリックします(**図11**)。

手順⑤

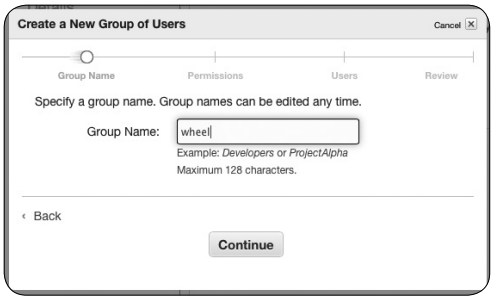
選択したポリシーの内容に従ったポリシードキュメントが表示されるので、[Continue]ボタンをクリックします。

▼図8　マネジメントコンソール上のリンク(IAM)

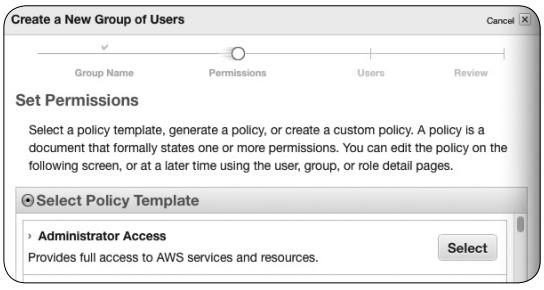
▼図9　[Create a New Group of Users]をクリック



▼図10　グループ名の指定



▼図11　権限の指定



手順⑥

作成する管理者ユーザの名前を入力します。ここではたとえばamzというユーザ名を入力し、[Continue]ボタンをクリックします(**図12**)。

手順⑦

確認画面が表示されるので、[Continue]ボタンをクリックします。

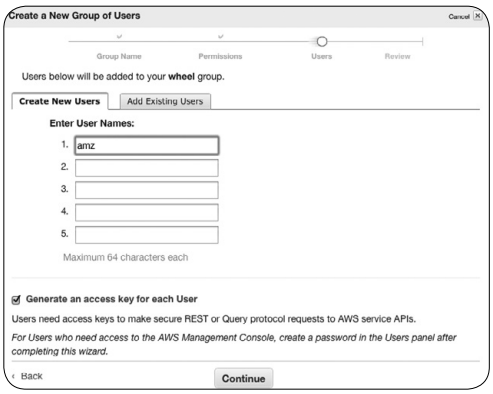
手順⑧

最後に、作成したユーザの認証情報を取得する画面が表示されるので、[Download Credentials]ボタンをクリックして、認証情報をダウンロードしてください(**図13**)。credentials.csvという名前のファイルがダウンロードフォルダに保存されます。

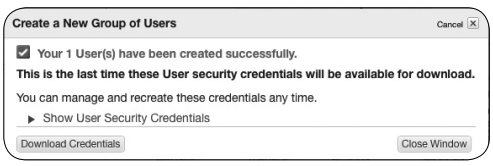
アクセスキーについて

アクセスキーの2つの情報のうち、アクセスキーIDはIAMの画面上で再確認ができますが、シークレットアクセスキーは、生成直後にダウンロードすることによってのみ取得ができます。

▼図12　管理者ユーザ名の入力



▼図13　ユーザ認証情報



もしダウンロードし忘れたり紛失した場合は、新しいアクセスキーを作成してダウンロードしなおす必要があります。なお、アクセスキーは1ユーザにつき最大2つまでしか保有できないので、不要になったアクセスキーは削除しておくようにしましょう。

今回のように最初に“Administrator Access”権限を持つIAMユーザを作成しておけば、そのあとはAPI経由で別のIAMユーザを作成できるようになります。この場合、IAMについてマネジメントコンソールで作業をしなくてはならないのは、AWSアカウント作成直後の1回だけということになります[注3]。IAMユーザのアクセスキーを取得したら、さっそく利用できるようにしましょう。

認証情報ファイルの作成

まず最初に、手動でAWS APIを叩くときに利用するために、シェルスクリプトから読み込める形式のファイルを作成します。ここでは、AWS CLIの流儀に従い、~/.awsというディレクトリを作成し、その下に置きます。

・コマンド:

```
% mkdir ~/.aws
% cat ~/Downloads/credentials.csv | grep ➥
-v ^User | awk 'BEGIN{FS=","}{print "aws_ ➥
access_key_id=" $2 "¥naws_secret_access_ ➥
key=" $3}' > ~/.aws/default.rc
```

次のような形式のファイルができあがります。

・~/.aws/default.rc

```
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=XXXXXXXXXXXXXXXXX ➥
XXXXXXXXXXXXXXXXXXXXXXX
```

続いてAWS CLI用の設定ファイルを作成します。

注3）ただし、このユーザは権限が強力過ぎるのでアクセスキーの管理には十分注意してください。

・コマンド:

```
% echo '[default]' > ~/.aws/config && ➥
cat ~/.aws/default.rc >> ~/.aws/config ➥
&& echo 'region=us-east-1' >> ~/.aws/ ➥
config
```

次のような形式のファイルができあがります。

・~/.aws/config

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=XXXXXXXXXXXXXXXXX➥
XXXXXXXXXXXXXXXXXXXX
region=us-east-1
```

これらのファイルが漏洩すると第三者に「なりすまし」をされてしまう可能性があるため、自分以外のユーザが閲覧できないようにディレクトリの権限を変更しておきましょう。

・コマンド:

```
% chmod 700 ~/.aws
```

はじめてのAWS APIアクセス

これでAWS CLIを使ってAWS APIにアクセスする準備が整いました。試しにS3にアクセスしてみましょう。

・コマンド:

```
% aws s3 ls
```

数秒程度で次のように、先ほど作成したS3バケットが表示されればOKです。

・実行結果:

```
2014-04-18 14:44:51 [組織名]-trail
```

もし、エラーが出る場合は、configファイル内で行末に余計なキャリッジリターン(^M)が存在しないか確認してみてください。

AWS CLIのコマンド補完

AWS CLIは、コマンドライン上でサブコマンドを補完する機能を提供しています。補完コマンドは、bashとtcshでは/usr/local/bin/aws_completer、zshでは/usr/local/bin/aws_zsh_completer.shを組み込むことで利用可能になります。

以下のコマンド例を、awsコマンドを使う前に実行するか、それぞれのシェルの設定ファイルに組み込んでおくようにしましょう。

・コマンド例(bashの場合)

```
$ complete -C aws_completer aws
```

・コマンド例(tcshの場合)

```
% complete aws 'p/*/`aws_completer`/'
```

・コマンド例(zshの場合)

```
$ source /usr/local/bin/aws_zsh_completer.sh
```

参考までに、自分の場合は昔からの.cshrcファイルを使っている関係で、下記のように設定しています。

・~/.cshrc

```
if ( ${?tcsh} ) then
  complete aws 'p/*/`aws_completer`/'
endif
```

組み込み後、サブコマンドの入力中にTABキーを押すとコマンド補完が行われます。候補一覧を表示するには、bashではTABキーの2度押し、tcshではCtrl+dを押してください。AWS CLIでの入力が非常に楽になることが実感できると思います。

次回は

今回の解説で、AWS APIを操作するための準備は整いました。次回は、引き続きマネジメントコンソール上で、請求に関連した各種設定事項、セキュリティ強度を向上させるための多要素認証(MFA)の設定について解説する予定です。SD

ひみつのLinux通信

作)くつなりょうすけ
@ryosuke927

第6回 ターミナルマルチプレクサの落とし穴

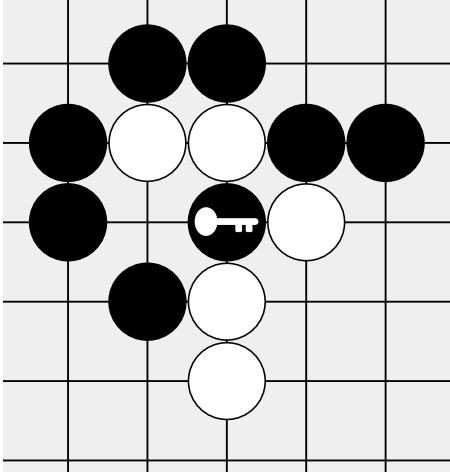
この3KのIT業界に新人なんか来るもんか！——って憤っていたら、新卒女子が配属されて怒りのやり場を失ったくつな先生のマンガが読めるのはSDだけ！

to be Continued

仮想端末管理ソフトを使ってますか？　見た目上、1端末に複数端末を起動できます。作業中、端末の異常終了や通信切断してもセッションのアタッチで終了前の環境が戻ります。数個のprefixキーとコマンドの組み合わせを覚えれば使えますよ。手元のtmuxからリモートホストに入り、そこでもtmuxを起動して「どうやって抜けよう……実行中のコマンドがあるのに……」となった場合はprefixキーを2回押し「d」すればリモートのセッションをデタッチできます。そんなこと悩んだことないですか？……そうですか。

セキュリティ実践の基本定石

すずきひろのぶ
suzuki.hironobu@gmail.com



みんなでもう一度見つめなおそう

【第十二回】信頼されるメールにするためには

今回のテーマは「信頼されるメール」です。メールの内容云々の話ではなく、形式的(機械的)に信頼できるメールとは何かを考えます。先日、筆者はとある銀行の法人向けインターネットバンキングから「不正送金被害が発生している」というメールを受信しました。しかし、それは自家製フィルタでフィッシングメールの可能性が高いと判断されました。その話題を取り上げます。

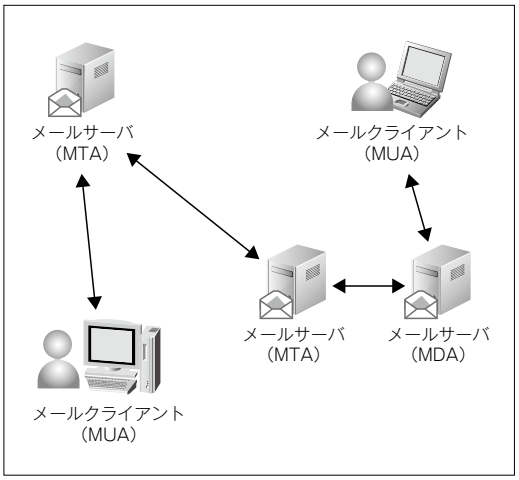
電子メールのシステム構成

すごく根本的な話題で申しわけないのですが、「電子メールがどう送られるのか」という基礎的な話題から入ります。電子メールを送付するシステム構成図は**図1**のようなものです。

メールユーザエージェント

ユーザはデスクトップPCあるいはノートPCで、メールを読み書きしています。ユーザの手元のPCからメールサーバに接続し、書いたり読んだりする

◆**図1　電子メール送受信における構成図**



MDAとMTAは、両方を兼ねたメールサーバが用意されていたり、別々に用意されていたりなど、サイトによってさまざまな環境の違いがある。

メールクライアントのことを「MUA(Mail User Agent)」と呼びます。ですが、もう日常的に使われているアプリケーションですので、普通に「メールクライアント」や「電子メールアプリケーション」と呼ぶ場合が多いでしょう。今やMUAと呼ぶのはエンジニアぐらいかもしれません。

ユーザの手元のマシン上で動作しているメールクライアントは、Mozilla Thunderbird やMicrosoft Outlook Express、はたまたEmacsのmh-eモードと多種多様です。Webブラウザでサーバ上にあるメールサービスを使う、いわゆるWebメールではGmailやYahoo!メール、大手インターネット情報会社が用意しているクラウド型環境、あるいはSquirrelMailなどを使って自ドメインでWebメールインターフェースを用意しているなど、こちらも多種多様です。ここでは「メールクライアント」と総称します。

メール転送／メール配送エージェント

インターネット上にあるメールサーバ間で電子メールを送受信するメールサーバのことを「メール転送エージェント」、または「MTA(Mail Transfer Agent)」と呼びます。インターネットに接続されているサーバの上で動いているMTAのソフトウェアは、Sendmail、Postfix、Exim、Qmail、Microsoft Exchange Serverなどがあります。

メールクライアントと接続し、メール転送エージェントに配送され保存されていたメールをメール

クライアントに送信する役割、あるいはメールクライアントから送られる電子メールを受け取り、メール転送エージェントに送る役割を受け持つのは「メール配送エージェント」、または「MDA（Mail Delivery Agent）」と呼びます。MDAとして使われているソフトウェアとしてはProcmail、Qpopper、UW（University of Washington）IMAPdなど、これも多数あります。

一般に、メールサーバと呼ぶときは、MTA/MDAを区別せず1つの（あるいは1セットの）サーバという認識で説明される場合が多いように思われます。ここでは、MTAとMDAを厳密に分類することはせず、「メールサーバ」と総称します。

SMTPの役割

SMTP（Simple Mail Transfer Protocol）は、インターネットで電子メールを転送するためのプロトコルです。歴史的にはSMTPの前にMail Transfer Protocol[注1]というプロトコルが提案され、そこから2年ほどの時間を経てSMTP[注2]が提案されます。そこからいろいろと改訂を重ね、2014年現在ではRFC 5321（2008）[注3]が参照されています。

SMTPプロトコルを使うことでインターネット間でメールをやりとりしますが、SMTPプロトコルを解釈できるプログラムを作れば、当然、電子メールを送りつけることができます。SMTPのポートは25/TCPです。

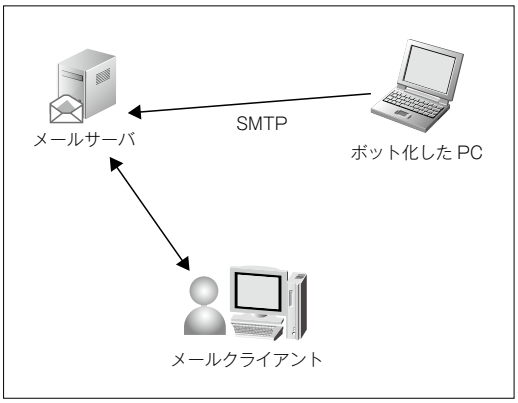
SMTPプロトコルを話すスパムプログラム

スパムプログラムやスパムボットは、まさにこのしくみを持っているプログラムで、メールサーバのSMTPポート（25/TCP）で待ち受けているMTAに接続し、SMTPプロトコルで大量の電子メールを送りつけます（**図2**）[注4]。

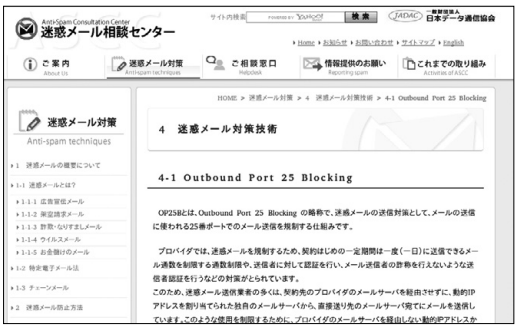
さて、ユーザが意図しなくても、マルウェアに感染しボットとなってしまったPCがたくさん存在する可能性（可能性というより、それが現実なのですが）があります。そのため、ISPや企業や学校では、ネットワーク内部から外部にスパムメールを送信させないために、外部のSMTPポートへの通信をブロックすることが、広く行われています（**図3**）。このことを「OP25B（Outbound Port 25 Blocking／25番ポートブロック）」と呼んでいます。

ISPがどのような具体的な対策をしているかは、「OCN迷惑メール対策」のWebページ[注5]を見ると、わかりやすいと思います。これによれば、送信規制

◆**図2　SMTPプロトコルでメールサーバにスパムを送りつける**



◆**図3　国内のOP25Bの取り組みを紹介するWebページ**



一般財団法人日本データ通信協会　迷惑メール対策技術
http://www.dekyo.or.jp/soudan/taisaku/4-1.html

注1）https://tools.ietf.org/html/rfc772
注2）https://tools.ietf.org/html/rfc821
注3）https://tools.ietf.org/html/rfc5321
注4）マルウェアに感染し、ボット化したPCが外部からコントロールを受け、スパムを大量に発信する状況に関しては、DDoSで使われるボットネットと基本的には同じ構造です。本連載第10回「根深くはびこるDDoS攻撃の脅威」（本誌2014年4月号）を読んでいただければ、より理解が深まるかと思います。
注5）http://service.ocn.ne.jp/mail/info/op25b/

は2007年7月より順次実施したとのことですので、かなり以前より実施されています。国内のISPではよく整備されていることがわかります。

グローバルで見た場合のスパムの状況

しかし、世界に目を向けると、整備されている環境ばかりではありません。スパムの現状分析に関しては、Kaspersky Labのサイト[注6]の内容が非常に役立つので、参考にしてみましょう。ここの分析レポートは毎月あがってくるので非常に助かっています。では、Kaspersky Labの2014年2月のレポートを参照してみます(**図4**)。

国別のスパムの割合を見てみると、上位5ヵ国で約67%を占めています。その上位5ヵ国の内訳は、中国約23%、米国約19%、韓国約13%、ロシア約7%、台湾約5%となっています。ちなみに日本は約2%です。

これを見てわかるように、中国、韓国、台湾の3ヵ国合計で約41%となっており、日本の周辺国はスパムを送出する国としては驚異的なレートになっています。また、ネットワーク的に非常に近い米国を含めると約60%に達します。つまり、日本はスパム発信国に囲まれているようなものです。一方、西ヨーロッパの地域は日本以上にスパム防止の対策が採られているようで、極めて少ない状況です。

スパムが多いということは、ボットネット経由でのスパム発信も数多くあるということを示唆しています。そして、フィッシングメールの発信基地としても使われる可能性が高い地域である、ということでもあります[注7]。

スパム発信のボットがたくさん存在し、多数のユーザが利用するメールサーバに大量のスパムを送り込むと、本来の電子メールの送受信に使う計算機のリソースが逼迫(ひっぱく)してしまうという現実的な問題が出てきます。

今どきの三種の神器

今どきの、スパムを防ぐために取り入れられている方法は、だいたい次の3つです。

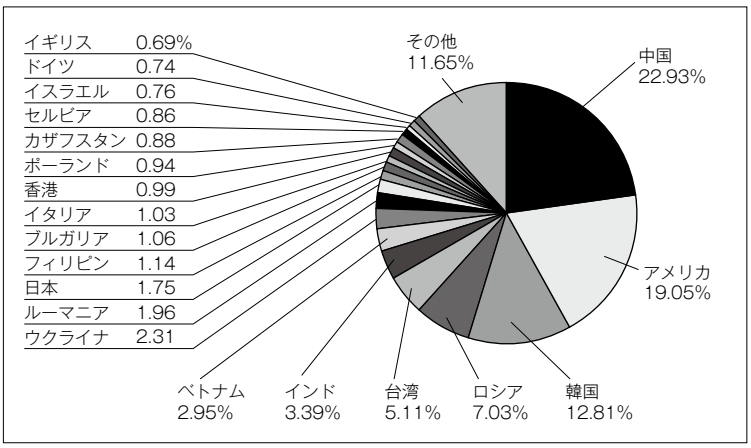
①DNSの逆引き
②SPFの設定
③DKIM-Signature

では、この3つの話題を取り上げていきます。

DNSの逆引き

前述したように、SMTPプロトコルでメールサーバのSMTPポートにアクセスすることで、個人のPCからでさえ、スパムを送り込めます。そこで、スパム対策として最初に現れたアイデアが、電子メールのヘッダにある接続元のホストの情報とIPアドレスからDNSを逆引きして、サイト名が一致

◆図4 国別スパム割合



出典："Spam report: February 2014"
http://www.securelist.com/en/analysis/204792328/Spam_report_February_2014

注6) www.securelist.com
注7) フィッシングに関しては、本連載第8回「真のフィッシング対策は『敵を知り、己を知る』ことから」(本誌2014年2月号)を参考にしていただけるとさらに理解が深まると思います。

しているかどうかを確認する方法です。

DNSにおいて、ドメイン名／ホスト名からIPアドレスを引く（検索する）のが「正引き」と呼ばれ、その反対にIPアドレスからドメイン名／ホスト名を引くのが「逆引き」と呼ばれています。

電子メールのヘッダには、接続元のホスト名とIPアドレスが記録されています。次の例は、筆者のところに届くIETF（Internet Engineering Task Force）のメーリングリストの電子メールのヘッダから抜粋した情報です。

```
Received: from mail.ietf.org ([4.31.198.44])
```

これをnslookupでチェックすると**図5**のようになります。このホスト名mail.ietf.orgの正引きと逆引きは矛盾がありません。よって、mail.ietf.orgは信頼性が高いと仮定しています。

いくつかのMTAでは、このように逆引きができないサイトからのSMTPポートへの接続を拒否する設定が可能です。あるいは、接続は拒否せず、SMTPで電子メールを受け取っても配送しない設定も可能です。実際にそのような運用をしていたメールサーバも多くあります。

筆者は見たことがないのですが、逆引きにDNSのMXレコード（メールサーバのホスト名を格納する）を使い、その接続元ホストがMXレコードに登録されていなければ拒否するといったスパムフィルタ機能を持ったものも世の中にはあるようです。

しかしながら、筆者の考えでは、この仮定はかなり怪しいです。逆引きができるサイトを信頼する／しないの問題ではなく、そもそも、「今どき、信頼性の確認にDNSの逆引きを使っていいものなのか」と思うからです。スパムを送ってくるホスト（実際は、一般家庭などからインターネットに接続している、いわゆるPCだと思いますが）は、逆引きができないパターンが多いという経験則みたいなことから、このDNSの逆引きが使われてきたのではないでしょうか。

実際にスパムの発信源となっている国は、逆引き設定がされていないものが多いというのは事実ですし、そのような状態では「逆引きできないマシン＝スパム発信」と考えるかもしれません。しかし、それは本質的ではありません。たまたまそのような現象になっているだけです。

先ほどのMXレコードに登録されていないということもそうです。今はバーチャルホスト全盛時代。1つのIPアドレスに複数のドメイン名／ホスト名を割り当てるのはざらです。筆者もそうしています。MXレコードは電子メールの宛先ドメイン名の部分をドメインで表記するときに、どのMTAに送るかの情報を与えるためのものです。特定のホストを指定してメールアドレスを記述する場合は、MXレコードを設定していなくても、原則として電子メールは届くのです。ですから、MXレコードに登録されていないから受け付けないというのもやり過ぎです。

たとえば、電子メールには送り主としてhironobu@h2np.netのメールアドレスがあり、送られてきたメールサーバがmail.h2np.netであれば、同一ドメインということで一貫性があります（**図6**）。これがもし、送り主がhironobu@example.comで、メールサーバがmail.h2np.netであったりすると、一貫

◆**図5　DNS逆引きによるチェック**

```
↓ DNS の正引き
% nslookup mail.ietf.org
（中略）
Name: mail.ietf.org
Address: 4.31.198.44

↓ DNS の逆引き
% nslookup 4.31.198.44
（中略）
44.198.31.4.in-addr.arpa name = mail.ietf.org.
（以下略）
```

◆**図6　MXレコードのチェック**

```
ドメイン名を指定し、メールサーバのホスト名などの情報が入っている MX レコードをチェックする
% nslookup -query=mx h2np.net
（中略）
h2np.net mail exchanger = 10 mail.h2np.net.
```

性はありません。なので、目安程度になることは確かです。

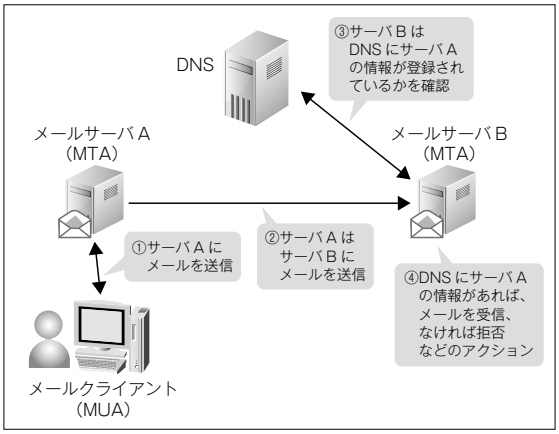
SPFの設定

これは送信者のドメインの詐称を防ぐ方法です。RFC 4408で仕様が決められています。DNSのSPF（Sender Policy Framework）レコードを設定し、それをメールサーバが使うことによって送信者ドメインの詐称を防ごうとします。

まず、DNSのSPFレコード（もしくはTXTレコード）には、そのドメイン名を使って送信して良いメールサーバのIPアドレスなどを指定しておきます。もちろんIPアドレスは複数指定することも可能です。

受信側のメールサーバは、相手メールサーバが接続してきた際に、（名乗ったドメイン名から）DNSのSPFレコード（もしくはTXTレコード）を引きにいきます（**図7**）。そこに登録されたIPアドレスの中に、接続してきたメールサーバのIPアドレスがあればメールの受信を行い、なければ拒否するといった動作をします。あらかじめ指定しておいたホスト以外から詐称して送信しようとしても、IPアドレスがマッチしないのでできません。これで送信元を詐称することは防げます。**図8**はietf.orgのSPFレコードをチェックしてみた例です。

◆**図7　DNSに問い合わせるモデル（SPFおよびDKIM-Signature）**



DKIM-Signature

現在、GmailやYahoo!メールなどの大手のサービスでは、送信したドメインを認証するためのしくみ「DomainKeys Identified Mail（DKIM）Signatures」[注8]を利用するようになってきました。これはRFC 4871[注9]とRFC 5672[注10]で定義されています。

DKIM-Signatureは電子メールのヘッダの中に含まれます。**リスト1**は先ほどのIETFのメーリングリストの電子メールにあったDKIM-Signatureです。

DKIM-Signatureはデジタル署名技術を使って、電子メールを保護します。検証のための公開鍵は、DNSに用意されています。受け取ったメールサーバはデジタル署名の検証を行い、正しいメールであるか否かの確認を行います。電子メールのヘッダ、そしてボディ（本文）も検証できる構造になっています。

公開鍵が登録してあるFQDN（ホスト名）は次のようになります。

```
<セレクタ>._domainkey.<ドメイン名>
```

◆**図8　SPFレコードのチェック**

```
$ nslookup -type=spf ietf.org
（中略）
ietf.org rdata_99 = "v=spf1 ip4:12.22.58.0/24 ip6:2001:1890:123a::/56
ip4:64.170.98.0/24 ip6:2001:1890:126c::/56 ip4:4.31.198.32/27
ip6:2001:1900:3001:0011::0/64 ip4:209.208.19.192/27
ip6:2607:f170:8000:1500::0/64 ip4:72.167.123.204 -all"
（以下略）
```

注8）www.dkim.org
注9）https://tools.ietf.org/html/rfc4871
注10）https://tools.ietf.org/html/rfc5672

リスト1では、s=ietf1、d=ietf.orgとなっていますので、FQDNはietf1._domainkey.ietf.orgとなります。これでチェックした結果が図9です。ここまで検証する能力があれば、送信元を詐称するのは（DKIMの脆弱性がない限り）無理だと言えます。

しかし、とても大切なことを忘れている

もう気がついたかもしれませんが、これは送り元のメールサーバ（MTA）が詐称されないようにするしくみです。DKIM-Signatureはメールの本文もデジタル署名する能力があるので、内容は改ざんされていない保証はありますが、それは送り元のドメインのメールサーバが正しい（存在が明らかな）場合だけです。

筆者に届いた、とある銀行の法人向けインターネットバンキングからの電子メールは、DKIM-Signatureが付いていましたが、その送られてきたドメインはクラウド業者のもので、こちらからわかることは、そのクラウド業者のPaaS環境にあるメールサーバだということのみです。

本当に銀行のものかどうかの区別はつけようがありません。しかも、「不正送金被害が発生しております」から始まり「詳しくは下記PDFをご覧ください」とあって、PDFのURLが書かれています。そのURLは確かに銀行のサイト上を指しています。しかし、Webサーバをクラックされて、PDFやFlashの脆弱性をついた感染型のマルウェアがしかけられているというのは、今日では日常茶飯事です。

GmailやYahoo!メールで名前を詐称して標的型攻撃でマルウェアを送ってくる場合、DKIM-Signatureは当然付いています。つまり、この銀行と標的型攻撃でマルウェアを送ってくる攻撃者とは、外形的には区別はつけられません。

「銀行が自らのドメインから電子メールを送るという前提で、SPF設定やDKIM-Signatureの設定をする」、さらに本文自体に確認性を持たせるならば、「S/MIME（Secure MIME）でもOpenPGPで自らの署名をつける」などしない限り、確認のしようがありません[注11]。

ちなみに、この銀行は日本でも最大級の銀行です。その銀行が、なぜこのように、セキュリティ面をまったく考慮していないのか、あるいは間違った理解をしているのか、筆者には理由がわかりません。このような電子メールを受け取り、深い憂慮（と、ため息）を覚えざるを得ませんでした。SD

◆リスト1　DKIM-Signatureの例

ドメイン名　セレクタ

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1;
t=1397940067; bh=OTHhmf927YP60zGq1wkkS0PID69Dl2609Xj1n4Slrhk=;
h=From:Subject:To:Reply-To:Date:Message-ID:MIME-Version:
Content-Type:Content-Transfer-Encoding:List-Id:List-Unsubscribe:
List-Archive:List-Post:List-Help:List-Subscribe:Sender;
b=vbMxeqNFQWslkMRyTV0g/Zp8o8F1nBUY9e9Jb0z0+0wfObRyrPHsRvfzDrlEnsvKp
6e/8+a8M66jeDcpNGcD71+XMNwDrA8tPtKvujuxxJgoZ0tCV8PkD3Z9VVMw6vFW4Ke
CcN+foMtUnTmVldqzN9wN6Kx016HLI4WaprGSiOY=
```

◆図9　DKIM-Signatureによるチェック

```
$ nslookup -type=txt ietf1._domainkey.ietf.org
（中略）
ietf1._domainkey.ietf.org text = "k=rsa\;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDNzNnjKTd5cczd2CDzHflCZuv1tMWYwd7zE+deoJ6s/fXR7/n9ZIBnDS5egt
7HAHjNjZrmjcoRlfSsNxRJvUQFyYvaU1BT1s8R+mkPgSOqZ4t9HqAVjiczn2B9+dbjdNN+S/zvSyMMuSCSJDKKAXhBpDeQTpeY7/
UdP9s6ws0yjQIDAQAB"
（以下略）
```

注11）しかし、本文に対するデジタル署名も、電子メールとテキスト内容との文字コードの違いなどがあり、すべての環境でうまく検証できるとは限りません。

ハイパーバイザの作り方

ちゃんと理解する仮想化技術

第20回 bhyveにおける仮想ディスクの実装

Writer 浅田 拓也（ASADA Takuya） Twitter @syuu1228

はじめに

前回の記事では、bhyveにおける仮想NICの実装について、TAPデバイスを用いたホストドライバの実現方法を例に挙げ解説しました。

今回は、bhyveにおける仮想ディスクの実装について解説していきます。

bhyveにおける仮想ディスクの実装

bhyveが、ゲストマシンに提供する仮想I/Oデバイスは、すべてユーザプロセスである/usr/sbin/bhyve上に実装されています（**図1**）。

bhyveは実機上のディスクコントローラと異なり、ホストOSのファイルシステム上のディスクイメージファイルに対してディスクI/Oを行います。

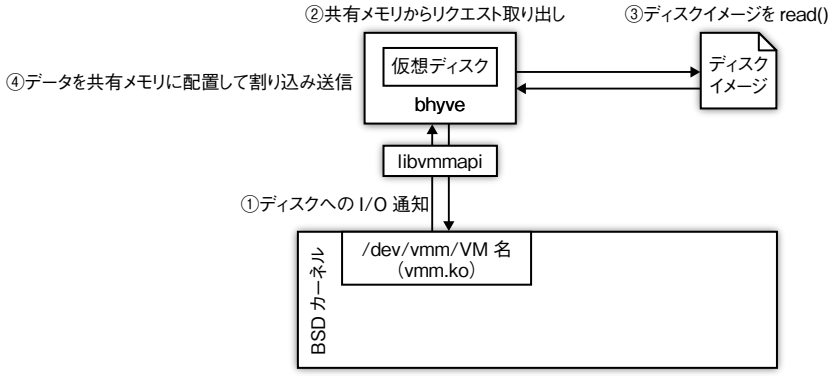
これを実現するために、/usr/sbin/bhyve上の仮想ディスクコントローラは、ゲストOSからのI/OリクエストをディスクイメージファイルへのファイルI/Oへ変換します。

以下に、ディスク読込み手順と全体図（**図1**）を示します。

書き込み処理では、リクエストと共にデータをリングバッファを用いて送りますが、それ以外は読み込みと同様です。

① ゲストOSはvirtio-blkドライバを用いて、共有メモリ上のリングバッファにI/Oリクエストを書き込む。そしてI/Oポートアクセスによってハイパーバイザにリクエスト送出を通知する。I/Oポートアクセスによって VMExitが発生し、CPUの制御がホストOSのvmm.koのコードに戻る。bhyveの仮想ディスクコントローラのエミュレーションは、ユーザランドで行われている。vmm.koはこのVMExitを受けてioctlをreturnし、/usr/sbin/

▼図1 ディスク読み込み手順



bhyveへ制御を移す
②ioctlのreturnを受け取った/usr/sbin/bhyveは、仮想ディスクコントローラの共有メモリ上のリングバッファからリクエストを取り出す
③②で取り出したリクエストをパースし、ディスクイメージファイルにread()を行う
④読み出したデータを、共有メモリ上のリングバッファに乗せ、ゲストOSに割り込みを送る
⑤ゲストOSは割り込みを受け、リングバッファからデータを読み出す

virtio-blkのしくみ

これまでに、準仮想化I/Oのしくみとして、virtioとVirtqueue、virtio-netについて解説してきました。ここでは、ブロックデバイスを準仮想化する、virtio-blkについて解説を行います。

virtio-netは受信キュー、送信キュー、コントロールキューの3つのVirtqueueからなっていましたが、virtio-blkでは単一のVirtqueueを用います。これは、ディスクコントローラの挙動がNICとは異なり、必ずOSからコマンド送信を行った後にデバイスからレスポンスが返るという順序になるためです[注1]。

ブロックI/Oのリクエストは連載第12回の「ゲスト→ホスト方向のデータ転送方法」で解説した手順で送信されます。

virtio-blkでは1つのブロックI/Oリクエストに対して、次のようにDescriptor[注2]群を使用します。1個目のDescriptorはstruct virtio_blk_outhdr（**表1**）を指します。この構造体にはリクエストの種類、リクエスト優先度、アクセス先オフセットを指定します。2～(n-1)個目以降のDescriptorはリクエストに使用するバッファを指します。リクエストがreadである場合は読み込み結果を入れる空きバッファを、writeである場合は書き込むデータを含むバッファを指定します。

バッファのアドレスは物理アドレス指定になるため、仮想アドレスで連続した領域でも物理的配置がバラバラな状態な場合があります。これをサポートするためにバッファ用Descriptorを複数に分けて確保できるようになっています。

struct virtio_blk_outhdrにはバッファ長のフィールドがありませんが、これはDescriptorのlenフィールドを用いてホストへ通知されます。n個目のDescriptorは1byteのステータスコード（**表2**）のアドレスを指します。このフィールドはホスト側がリクエストの実行結果を返すために使われます。

注1） NICではOSから何もリクエストを送らなくても、ネットワーク上のほかのノードからパケットが届くので、データが送られてきます。このため、必ずOSからリクエストを送ってから届くというような処理にはなりません。

注2） Virtqueue上でデータ転送そするための構造体。第12回のVirtqueueの項目を参照。

▼表1　struct virtio_blk_outhdr

type	member	description
u32	type	リクエストの種類（read=0x0, write=0x1, ident=0x8）
u32	ioprio	リクエスト優先度
u64	sector	セクタ番号（オフセット値）

▼表2　ステータスコード

value	name	description
0	OK	正常終了
1	IOERR	IOエラー
2	UNSUPP	サポートされないリクエスト

ディスクイメージへのI/O

/usr/sbin/bhyveはvirtio-blkを通じてゲストOSからディスクI/Oリクエストを受け取り、ディスクイメージへ読み書きを行います。bhyveが対応するディスクイメージはRAW形式のみなので、ディスクイメージへの読み書きはとても単純です。ゲストOSから指定されたオフセット値とバッファ長をそのまま用いてディスクイメージへ読み書きを行えばよいだけです[注3]。

それでは、このディスクイメージへのI/Oの部分についてbhyveのコードを実際に確認してみましょう。/usr/sbin/bhyveの仮想ディスクI/O処理のコードを**リスト1**に示します。

注3) QCOW2形式などのより複雑なフォーマットでは未使用領域を圧縮するため、ゲスト・ホスト間でオフセット値が一致しなくなり、またメタデータを持つ必要が出てくるのでRAWイメージと比較して複雑な実装になります。

▼リスト1　/usr/sbin/bhyveの仮想ディスクI/O処理

```
/* ゲストOSからI/O要求があった時に呼ばれる */
static void
pci_vtblk_proc(struct pci_vtblk_softc *sc, struct vqueue_info *vq)
{
  struct virtio_blk_hdr *vbh;
  uint8_t *status;
  int i, n;
  int err;
  int iolen;
  int writeop, type;
  off_t offset;
  struct iovec iov[VTBLK_MAXSEGS + 2];
  uint16_t flags[VTBLK_MAXSEGS + 2];
  /* iovに1リクエスト分のDescriptorを取り出し */
  n = vq_getchain(vq, iov, VTBLK_MAXSEGS + 2, flags);
          ……(中略)……
  /* 1つ目のDescriptorはstruct virtio_blk_outhdr */
  vbh = iov[0].iov_base;
  /* 最後のDescriptorはステータスコード */
  status = iov[--n].iov_base;
          ……(中略)……
  /* リクエストの種類 */
  type = vbh->vbh_type;
  writeop = (type == VBH_OP_WRITE);
  /* オフセットをsectorからbyteに変換 */
  offset = vbh->vbh_sector * DEV_BSIZE;
  /* バッファの合計長 */
  iolen = 0;
  for (i = 1; i < n; i++) {
          ……(中略)……
    iolen += iov[i].iov_len;
  }
          ……(中略)……
  switch (type) {
  /* WRITEならpwritev()でiovの配列で表されるバッファリストからディスクイメージへ書き込み */
  case VBH_OP_WRITE:
    err = pwritev(sc->vbsc_fd, iov + 1, i - 1, offset);
    break;
  /* READならpreadv()でディスクイメージからiovの配列で表されるバッファリストへ読み込み */
  case VBH_OP_READ:
    err = preadv(sc->vbsc_fd, iov + 1, i - 1, offset);
    break;
```

```
  /* IDENTなら仮想ディスクのidentifyを返す */
  case VBH_OP_IDENT:
    /* Assume a single buffer */
    strlcpy(iov[1].iov_base, sc->vbsc_ident,
        min(iov[1].iov_len, sizeof(sc->vbsc_ident)));
    err = 0;
    break;
  default:
    err = -ENOSYS;
    break;
  }
          ……(中略)……
  /* ステータスコードのアドレスにIOの結果を書き込む */
  if (err < 0) {
    if (err == -ENOSYS)
      *status = VTBLK_S_UNSUPP;
    else
      *status = VTBLK_S_IOERR;
  } else
    *status = VTBLK_S_OK;
          ……(中略)……
  /* ステータスコードを書き込んだことを通知 */
  vq_relchain(vq, 1);
}
```

まとめ

今回は仮想マシンのストレージデバイスについて解説しました。

次回は、仮想マシンのコンソールデバイスについて解説します。SD

Red Hat Enterprise Linuxを
極める・使いこなすヒント

.SPECs

ドット・スペックス

第2回 バージョニングとライフサイクル

前回はサブスクリプション契約と、契約によって得られるサポートについて説明しました。今回はRed Hat Enterprise LinuxとRPMのバージョンについて理解し、製品ライフサイクルを有効活用するための知識を深めましょう。

レッドハット(株)グローバルサービス本部プラットフォームソリューション統括部
ソリューションアーキテクト部長　藤田 稜 (ふじた りょう)

サポートと切っても切り離せないバージョン

Red Hat Enterprise Linux(以降、RHEL)にはバージョンがあります。読者諸氏の中には「何を今更」と思われる方もいらっしゃると思いますが、「RHELのバージョンとは何か?」を改めて考えると実は非常に難しい問題を含んでいます。このことはRHELに固有の問題ではなくソフトウェア一般に対して言えることなのですが、詳しく見てみましょう。

まずはRHELのバージョン番号について見てみます。執筆時点でのRHELの最新版は、6系は6.5、5系は5.10、4系は4.9[注1]です。ここでバージョン番号を"x.y"とすると、"x"は「メジャーバージョン」、"y"まで含めると「マイナーバージョン」となります。"RHEL6"はメジャーバージョン、"RHEL6.5"はマイナーバージョン、というわけです。

RHELの各メジャーバージョンにはサポートライフサイクル[注2]があり、米国での製品出荷日を起算点として、RHEL5/6では13年、RHEL3/4では10年のサポートが提供されます。ただし、この期間に渡って一律に同じレベルのサポートが提供されるわけではありません。

フェイズによって変化するサポート内容

サポートのレベルはProduction 1/2/3とExtended Life Phaseに分かれており、各レベルが提供される期間として、RHEL5と6では5.5/1/3.5/3年、RHEL3と4では4/1/2/3年が設定されています(**図1**)。

Production 1/2/3のサポート内容の違いは「新しいハードウェアへの対応」と「ソフトウェアの機能強化」の2点です。Production 1ではそのいずれも行われますが、Production 2ではCPUの動作クロックアップのような「限定的な新しいハードウェアへの対応」となり、「ソフトウェアの機能強化」は行われなくなります。さらにProduction 3では新しいハードウェアへの対応は仮想化を利用して行う[注3]こととなる

注1) このバージョン番号については命名ポリシーの変更が数回あり、古いRHELでは必ずしもこの呼び方ではない。たとえばRHEL 2.1は当初Red Hat Linux 2.1 Advanced Serverと呼ばれており、ライフサイクルの途中で"Red Hat Enterprise Linux 2.1"と変更された。また当時はマイナーバージョンを"Update"と呼んでおり、"Red Hat Enterprise Linux 2.1 Update 7"のように区別された。これを"RHEL 2.1.7"と呼ぶことはない。さらに一時期、"Service Pack"なる呼称が導入されたこともあり、米MicrosoftのWindowsと同様に"SP 1"のようなマイナーバージョンが付けられたこともある。

注2) 公式なサポートライフサイクルについては、次のURLを参照。https://access.redhat.com/site/support/policy/updates/errata/

注3) 仮想化ホストとして新しいRHELを採用し、古いRHELを仮想化ゲストとして動作させることで対応する、という意味。

ため、Production 3に入っているRHELを採用する際は注意が必要です。

Production 3の期間が終了するとExtended Life Phase(以降、ELP)となります。この期間に提供されるのは過去にリリースされたドキュメントへのアクセスだけとなり、すでにインストールされたREHLだけが対象となります。しかしながら、少なからぬユーザからできるだけ長く同じメジャーバージョンを利用したいという要望を受けて、レッドハットではすでにELPに入ったRHEL3/4に対してExtended Lifecycle Support(以降、ELS)を提供しています。サブスクリプション契約にELSを追加することで、重大度の高いセキュリティの脆弱性に対するエラータなどが提供されます。執筆時点では、ELSがRHEL5/6に対して提供されるかは未定となっています。

マイナーバージョンのライフサイクルとサポート

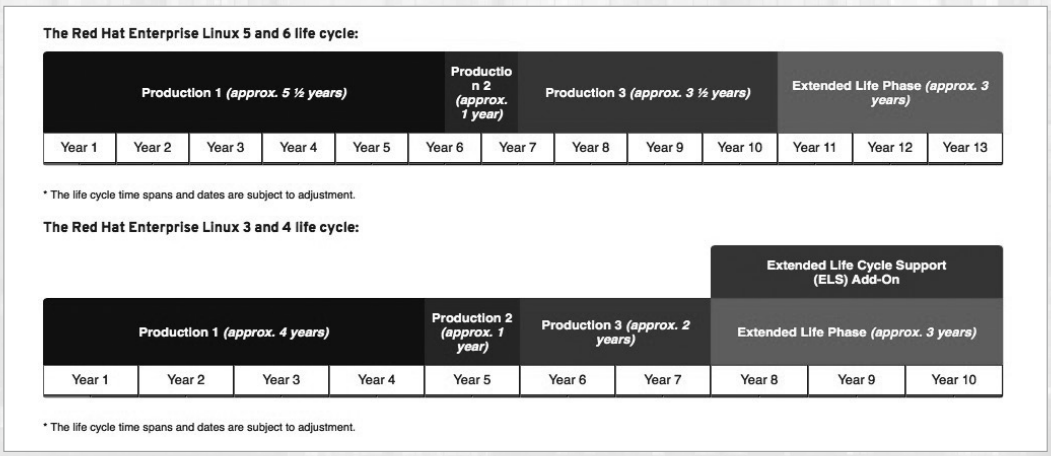
メジャーバージョンのサポートライフサイクルについての説明に続き、マイナーバージョンのライフサイクルについても紹介しましょう。

RHELのマイナーバージョンは基本的に6ヵ月ごとに更新されます。更新されると古いマイナーバージョンに対するエラータの提供は行われなくなります。たとえば、RHEL6.5の次のバージョン・6.6がリリースされると、6.5というマイナーバージョンに対するエラータの提供は終了する、ということです。ここで注意が必要なのは、RHEL6.6がリリースされたからといって、6.5のサポートが「すべて」終了するわけではないという点です。

マイナーバージョンが上がる際に更新されるRPMパッケージは全体の4分の1程度で、残りの4分の3は更新されません(**図2**)。マイナーバージョンのリリース時に例外なく更新されるのは実はkernelだけで[注4]、そのほか多くのRPMパッケージは更新されないことが多いのです。違う言い方をすると、更新されなかったRPMパッケージがRHELのどのマイナーバージョンに含まれているのかは、そのRPMパッケージのバージョンだけでは特定できない可能性が高いのです[注5]。

したがって、問題報告を受けたレッドハットのサポート窓口での回答は次のようになること

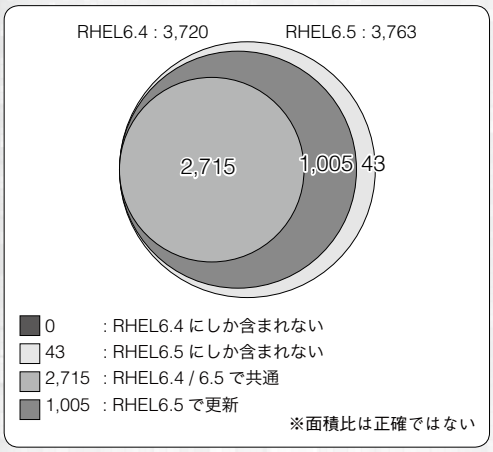
▼図1　各バージョンのライフサイクルとサポート



注4) kernelが必ず更新されるのは、kernelのRPMパッケージにデバイスドライバが入っており、新規ハードウェア対応のためにデバイスドライバが更新されるため。kernelに次いでほぼ毎回更新されるのはglibcだが、過去にマイナーバージョン間で更新されなかったことが1度だけある。

注5) サポート窓口では問題の原因究明のために、SOS(Son Of Sysreport)のレポートの取得をユーザに依頼することがあり、SOSレポートを参照することでマイナーバージョンの特定が可能。

▼図2　マイナーバージョン間のRPMパッケージの差分の例



が容易に想像できるはずです。

・未知の問題の場合→最新のバージョンでの修正
・既知の問題の場合→その問題が修正されているリリース済みエラータの案内

つまり、最新のマイナーバージョンでしかサポートを受けられないというのは間違った解釈で、上記のように場合分けしてサポート提供しているのが実際です。ただし、前回説明したハードウェア動作認定の観点からは、認定されているマイナーバージョン以降でしかハードウェアに起因する問題に関するサポートが提供されない、ということに注意が必要です。

マイナーバージョンの延長サポート

古いマイナーバージョンを利用していてもサポートを受けられるとわかっても、何か不具合に直面した際に、最新のマイナーバージョンに含まれるRPMパッケージ、とくにkernelを更新しないとならないのは、運用の現場にある方には負担が大きいかもしれません。そこでレッドハットはExtended Update Support(以降、EUS)というオプションを提供しています。このオプションを購入することで、特定のマイナーバージョンに対するエラータを最大で24ヵ月間受け取ることが可能になります(**図3**)。

ソフトウェアのエンジニアリングを考えると当然なのですが、6.5の次に6.6がリリースされたのであれば、さらに次のバージョンである6.7のリリースまで6.6をメンテナンスするのが最も合理的です。6.6や以降のバージョンで提供されたエラータを、6.5に向けて「バックポート」するのはコストのかかる作業なので、オプションとなっている点はご承知ください。

RPMパッケージの名称

マイナーバージョンの定義を説明する前に、RPMパッケージの命名法について説明しましょう。執筆時点で最新のRHEL6のkernelを例に挙げます。

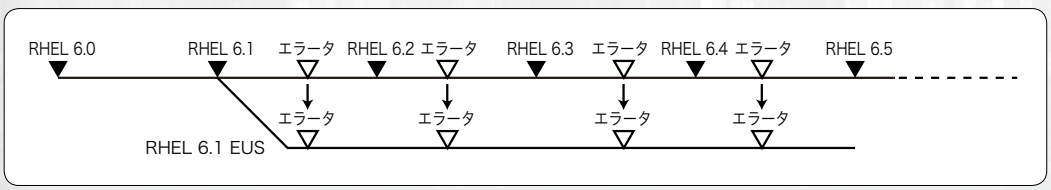
```
kernel-2.6.32-431.5.1.el6.x86_64.rpm
```

これは次のような書式[注6]になっています。

```
%{NAME}-%{VERSION}-%{RELEASE}%{ARCH}.rpm
```

ここで`%{VERSION}`はOSSコミュニティでの

▼図3　延長サポートの例



注6）rpmコマンドに--querytagsを付与して実行するとタグの一覧が参照できる。

バージョンを指し、**%{RELEASE}**はレッドハットでのリリースバージョンを指します。したがってこのkernelの場合、**kernel.org**でリリースした**2.6.32**をベースに、その後のパッチをレッドハットがバックポートして**431.5.1.el6**というバージョンとしてリリースしたもの、ということになります。**2.6.32**というkernelは3ヵ月に1度というkernelのリリース頻度からするとかなり古いものですが、レッドハットが選定した最新の機能やハードウェアにも対応しており、単純にkernel.orgの**2.6.33**や以降のバージョンとの新旧比較ができるものではないということです。この開発のフローについては次回の本連載で詳説します。

▼図4　RHEL4.8までのISOファイルしか用意されていない

マイナーバージョンとは？

RPMパッケージのバージョンについて説明しましたので、次にマイナーバージョンの実体について考えてみましょう。実はこれが冒頭に書いた「非常に難しい問題」なのです。

RHELをインストールする時には、ISOファイル、あるいはそれを焼いたメディアキットを用います。ではその中に「マイナーバージョンを定義するファイル（データ）」が含まれるのかというとそんなことはなく、インストーラはISOファイル／メディアに含まれるレポジトリから、指定されたRPMパッケージの名称だけでインストールするファイルを選択しています[注7]。つまり、特定のバージョンのRPMパッケージの組み合わせが特定のマイナーバージョンとなるわけではないということです。

実はマイナーバージョンの技術的に正確な定義とは「何年何月何日にリリースされたRPMパッケージの組み合わせ」なのです。実際に、同じマイナーバージョンのRHELで対応CPUアーキテクチャが異なる場合（例：x86_64とppc64など）でリリースが同日でなかったときに、含まれるRPMパッケージが異なることが過去にありました。

また、Production 3の開始と同時に最終マイナーバージョンがリリースされるのですが、この「最終マイナーバージョン」にはISOファイルが作成されません。しばしば「（RHEL4系最終マイナーバージョンの）RHEL4.9のインストーラが欲しいけれど、探しても見つからない」という質問を受けるのですが、それも当然で、そもそも最終マイナーバージョンのISOはリリースしていないのです（**図4**）。結論として、何をもってマイナーバージョンを定義できるかというと、日付をキーにするしか方法がないのです。

次回は

開発のフローを含むソフトウェアのエンジニアリングの観点から、どのようにすれば、安全で安定したRHELサーバを構築できるのか、初歩的な内容も含めて紹介します。SD

注7）comps.xmlを参照のこと。たとえばRHEL 6.5のx86_64版であれば、**repodata/6221039e7e3dabf7d538c76571d82aaf42b6292b8f6fe6cf56b8fcf1cff3d3ab-comps-rhel6-Server.xml**というファイル。

Be familiar with FreeBSD.

チャーリー・ルートからの手紙

第8回 ◇ 次世代仮想化基盤／ハイパーバイザ「bhyve」

◉著者プロフィール
後藤 大地（ごとう だいち）
BSDコンサルティング（株） 取締役／（有）オングス 代表取締役／FreeBSD committer
エンタープライズシステムの設計、開発、運用、保守からIT系ニュースの執筆、IT系雑誌や書籍における執筆まで幅広く手がける。カーネルのカスタマイズから業務に特化したディストリビューションの構築、自動デプロイ、ネットワークの構築など、FreeBSD/Linuxを含め対応。BSDコンサルティングでは企業レベルで求められるFreeBSDの要求に柔軟に対応。

FreeBSDに導入されたbhyve

FreeBSD 10.0-RELEASEからは新しい仮想化基盤／ハイパーバイザとしてbhyve（ビハイブ）が導入されました。NetAppがFreeBSDプロジェクトに寄贈したコードをベースに開発が進められている新しい仮想化基盤／ハイパーバイザです。既存の実装としてはXenやKVMなどの機能に相当します。

FreeBSDはストレージアプライアンスのオペレーティングシステムとして採用されることが多く、いくつかのストレージベンダはFreeBSDに仮想化機能を実装して自社プロダクトで活用しています。bhyveのベースとなったコードはそうした取り組みのひとつです。NetAppが活用している仮想化技術のうち、とくに基盤となる機能がコミットされています。

XenやKVMと比較した場合のbhyveの特徴は、まずBSDライセンスであること、そして後発のハイパーバイザだけあって構造の見通しがよいことです。Intelプロセッサの仮想化支援機能を前提にした実装になっており、構造がシンプルです。設計と実装に関しては本誌にて浅田拓也氏が連載「ハイパーバイザの作り方」で詳細を解説していますので、そちらをご覧ください。

今回はこの新しい仮想化基盤／ハイパーバイザでFreeBSDゲストを実行する方法を紹介します。

bhyveへFreeBSD 10をインストールしてみよう!

bhyveは、10系で段階的に機能の追加と性能の向上を実施します。10.0Rでは対応していませんが、最終的にはWindowsもサポートする予定です。第1弾となる10.0Rに取り込まれたのはbhyve、そのコントローラとゲストを仮想マシンに読み込むローダです。基本的にFreeBSDをゲストOSとして実行するためのしくみです。

次のリリースで、LinuxやOpenBSDといったほかのUnix系オペレーティングシステムのサポート、さらに次のリリースでWindowsといったBIOSエミュレーションを必要とするオペレーティングシステムのサポートといったことになるとみられます。ドライバ回りの改善も実施される見通しで、通信性能などの向上も期待されます。

bhyveにFreeBSDゲストをインストールする方法はいくつかありますが、bhyveの開発者であるNeel Natu氏が提供しているスクリプトを使って、**図1**のように作業する方法がシンプルでわかりやすいでしょう。

最後の`./vmrun.sh vm1`で仮想環境が起動します。**図2**の画面はターミナルでFreeBSDがブートしているような画面ですが、この画面がすでにbhyveで動作しているFreeBSDゲストが出力しているものです。

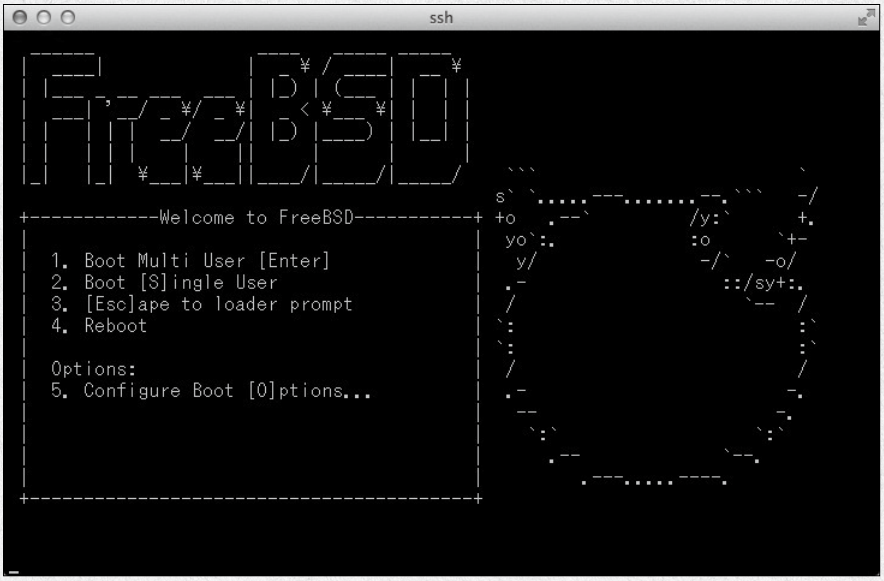
インストーラの起動の途中でターミナルタイプを入力するように聞かれますので、ここではvt100かxtermを入力しておきます（**図3**）。

各コマンドのうしろについている括弧書きの数字は、manコマンドで見ることができるマニュアルに記載されている章番号を表しています。

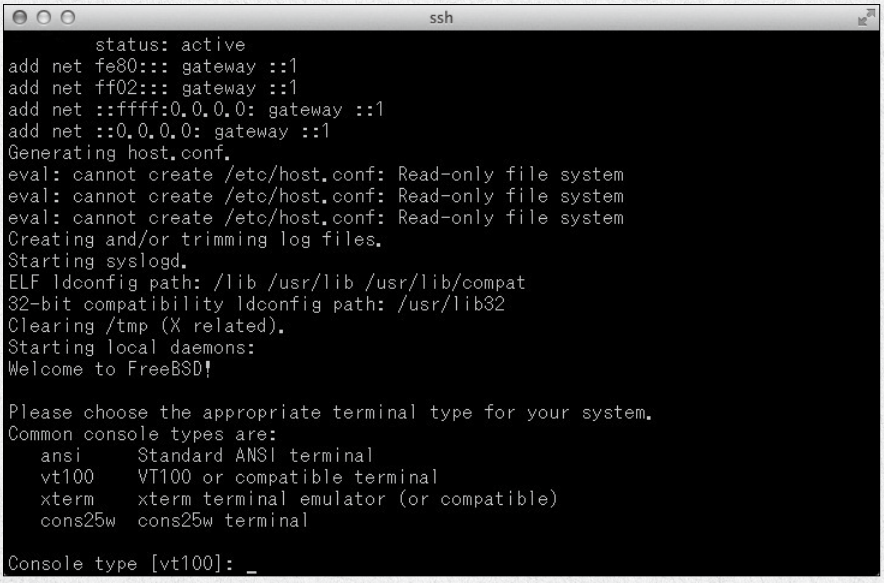
▼図1　FreeBSDゲストをインストールする方法

```
kldload vmm
kldload if_tap
ifconfig tap0 create
fetch http://people.freebsd.org/~neel/bhyve/vmrun.sh
chmod 600 ./vmrun.sh
fetch ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-amd64/10.0/FreeBSD-10.0-RELEASE-amd64-disc1.iso
mv *iso release.iso
./vmrun.sh vm1
```

▼図2　bhyveのゲストOSとして起動してくるFreeBSD 10.0-RELEASEのインストーラ



▼図3　インストーラの起動途中でターミナルタイプを聞かれるので、vt100やxtermなど使っている端末に適した値を入力



チャーリー・ルートからの手紙

あとはいつものようにインストール作業を進めるだけです(**図4**)。いつものインストール作業とちょっと違うのは、ストレージデバイスやネットワークインターフェースとして仮想環境向けのVirtIOデバイスドライバ(virtio_blk(4)、vtnet(4))が確認できるあたりです(**図5、6**)。

通常のインストールと1点だけ大きく異なるのは、システムを再起動する前にインストールした先の/etc/ttysファイルに**リスト1**のエントリを追加する必要があることです。インストーラを終了する前に、インタラクティブシェルのモードに入って編集作業を行います。インストーラの最後にシェルに入るか聞かれますので、このタイミングでシェルに入って作業します。

もし編集するのを忘れてしまったとしても、bhyveのゲストOSは仮想ディスクファイルにデータが書き込まれているだけですので、**図7**のようにホストで**mdconfig**(8)を使って仮想ディスクファイルマウントしてから編集するといったこともできます。

bhyveで動作しているゲストOSは、ホストOSから見ると**図8**のように、プロセスのように見えます。

10.1R、10.2R、11.0Rといった時代には周辺のツールやbhyveの提供する機能もより便利になるでしょうから、インストールの手順はさらに簡単になるでしょう。

▼図4　インストール作業は通常のインストールと同じ

▼図5　表示されるストレージデバイスがvtbdというVirtIO virtio_blk(4)ブロックドライバになっている

▼図6　ネットワークインターフェースはVirtIOイーサネットドライバvtnet(4)

▼リスト1　インストーラを再起動する前に/etc/ttysにこのエントリを追加する

```
console "/usr/libexec/getty std.9600"   vt100   on secure
```

▼図7　ホストから仮想ディスクイメージをマウントして編集するならこの方法

```
mdconfig -a -t vnode -f diskdev
mount /dev/md0 /mnt
vi /mnt/etc/ttys
sync
umount /mnt
mdconfig -d -u 0
```

▼図8　ホストからみるとゲストOSはbhyveという単一のプロセスとして見える

```
ssh
last pid: 13985;  load averages:  0.09,  0.12,  0.08    up 0+20:25:26  11:35:48
63 processes:  1 running, 62 sleeping
CPU:  0.0% user,  0.0% nice,  0.2% system,  0.0% interrupt, 99.7% idle
Mem: 134M Active, 2289M Inact, 2247M Wired, 22M Cache, 1643M Buf, 11G Free
ARC: 885M Total, 266M MFU, 228M MRU, 78K Anon, 13M Header, 379M Other
Swap: 2047M Total, 2047M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME    WCPU COMMAND
12911 root          4  20    0   539M 25444K vmidle  2   0:16   0.00% bhyve
  768 root          1  20    0   274M  3844K select  0   0:00   0.00% rpc.statd
 1180 daichi        1  20    0 86084K  6988K select  4   0:02   0.00% sshd
12877 daichi        1  20    0 86084K  6988K select  2   0:01   0.00% sshd
12875 root          1  33    0 86084K  6988K select  3   0:00   0.00% sshd
13963 root          1  33    0 86084K  6988K select  7   0:00   0.00% sshd
12690 daichi        1  20    0 86084K  6988K select  5   0:00   0.00% sshd
12688 root          1  27    0 86084K  6988K select  6   0:00   0.00% sshd
13965 daichi        1  20    0 86084K  6988K select  3   0:00   0.00% sshd
 1235 daichi        1  20    0 86084K  6980K select  3   0:01   0.00% sshd
 1178 root          1  49    0 86084K  6980K select  2   0:00   0.00% sshd
 1233 root          1  28    0 86084K  6980K select  7   0:00   0.00% sshd
  880 root          1  20    0 60816K  6312K select  1   0:00   0.00% sshd
12897 root          1  36    0 54352K  3704K select  7   0:00   0.00% sudo
 1184 daichi        1  20    0 49912K  5700K select  5   0:00   0.00% ssh-agent
 1239 daichi        1  20    0 49912K  5700K select  4   0:00   0.00% ssh-agent
13969 daichi        1  20    0 49912K  5688K select  7   0:00   0.00% ssh-agent
```

次世代の仮想化技術基盤：bhyve(8)とjail(8)

今後、FreeBSDの仮想化技術は**bhyve**(8)と**jail**(8)を二大基盤とすることになります。典型的なこの2つの仮想化技術の切り分けは、性能が必要となるケースではjail(8)、完全な仮想化やFreeBSDカーネル以外のオペレーティングシステムを動作させる場合にはbhyve(8)といった切り分けになるといえます。jail(8)はネットワークスタックの分離やリソース制御機能なども備えているため、bhyve(8)を使わずともjail(8)で完結できるケースも多く、どういったシステムを構築して運用するのか、アイディア次第でいろんなことができるようになりました。

XenやKVMといった仮想化技術は割り切りやすいので便利ですが、コンパートメント系の仮想化技術であるjail(8)やLXCと比べると遅いという問題があります。費用対効果の引き上げを考えると、XenやKVMのみでは訴求力が弱いといえます。今後は完全仮想化／準仮想化とコンパートメント系の仮想化技術を適材適所で切り分けて、費用対効果の高いシステム構築と運用が望まれることになるでしょう。

bhyveの採用事例：Linux on bhyveとFreeBSDという組み合わせ

現在bhyveが使われている事例ですと、商用ソフトウェアなど動作に特定のバージョンのLinuxディストリビューションが要求される環境をbhyveの上に構築し、それ以外はjail(8)などを活用している、といったものがあります。ベースはFreeBSDで統一して、Linuxが必要な部分だけbhyveを入れてLinuxを動かすというやり方です。Linuxは商用ソフトウェアが提供されているので、この組み合わせはロジカルな選択肢のひとつといえそうです。

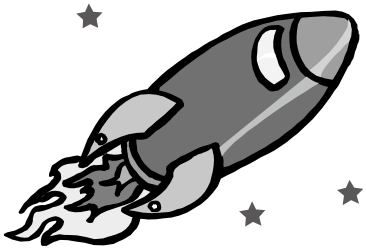
jail(8)の内部をLinuxディストリビューションにするというアプローチもありますが、この方法だとLinuxバイナリ互換機能で提供されていないLinuxカーネルのシステムコールが利用できないという不完全さがあります。bhyveでLinuxカーネルそのものを動作させることで、そういった問題を解決することができます。

bhyveはFreeBSDコミュニティやデベロッパの中に登場してからまだ日が浅く、活用事例や組み合わせのテクニックなどはまだ少ない状況です。今後さまざまな採用事例や活用方法、各種アイディア、スキルやテクニックなどが登場することになるとみられます。SD

15

Debian Developer　やまねひでき　henrich@debian.org

apt lineとexperimentalの関係



Debian Hot Topics

ホットトピックいろいろ

「Squeeze」LTS開始

2013年5月4日にDebian 7「Wheezy」がリリースされましたので、この号が出てしばらくするころ(5月31日)にDebian 6「Squeeze」のサポートはいったん終了を迎える予定です。「いったん」と書いたのは、前回案内したLTS(Long Term Support)が正式にアナウンスされたことによります[注1]。Squeeze-LTSの内容は次のとおりとなります。

- 2016年2月までの計5年間のサポート
- サポート対象アーキテクチャはi386とamd64のみ(つまり、NASなどで使われているarmelアーキテクチャはサポート対象外)
- Webベースのアプリケーションなどで5年間のサポートが難しく、対象外になるものがいくつか出てくる。どれがサポート対象外のソフトウェアなのか、という判別については別途ツールが提供される予定
- 今後、Debian 7「Wheezy」もLTSになるかどうかは、今回の結果次第
- 対応作業リソースが必要になるので、興味がある企業などはteam@security.debian.org宛に連絡をしてほしい

LTSが出たことは喜ばしいですが、Debian 6を使い続けなければならない強い理由がない限りは、適宜、Debian 7.xへアップグレードを実施するほうが無難です(そのための猶予ができた、というとらえ方もできますね)。現在は4月末にリリースされた7.5をご利用ください。

LTSの詳細は別途アナウンスが出る予定ですので、そちらを参照してください。

initシステム本決まり

initシステムについては、技術委員会より「systemdをデフォルトとして採用する」という勧告が出たのは先日お伝えしたとおりです。しかし、これに対して「Debianプロジェクトの全開発者での投票[注2]を行って、この勧告を受け入れるのかどうか、最終的な意志確認をしよう」という一部開発者らによる動きがありました(Debianプロジェクトは良くも悪くも開発者間での「平等」意識が高く、どちらかというとこのようなトップダウン方式の意思決定が好まれない、というのが如実に出た話だと思います)。

しかし、すでにオープンな長い議論を技術委員会がやった結果でしょうか、あまり賛成者を得られなかったようで、この投票自体の取り下

注1) URL https://lists.debian.org/debian-security-announce/2014/msg00082.html

注2) これを「一般決議(GR：General Resolution)」と呼びます。投票は電子投票になりますが、GnuPG鍵署名によって第三者による虚偽の投票を防ぐなど真正性が担保されています。その他の特徴として「選好投票(優先順位投票／ランキング投票)」と呼ばれる方式(コンドルセ方式)を採用しています。これは通常の選挙などで使われる方式と違い、単なる死票が少なくなるという利点があります。興味のある方は次のURLを参照してください。
URL http://ja.wikipedia.org/wiki/選好投票

げ表明が正式に出てきました。これで次のリリース「Jessie」でのsystemdの採用は本決まりです[注3]。あとは各パッケージ側でどの程度の作業が必要なのかが見えていないので、今後はそのあたりに注目したいと思います。

残されている問題はsystemd自体の安定した仕様と動作ですが、DHCPクライアントまでsystemd内で実装しようとしたり、カーネル側に問題を押しつけてLinus氏に悪態をつかれたりと、なかなかノープロブレムとはいかないようです。Debian 8「Jessie」のリリースまでに静かになってくれると良いのですが。

プロジェクトリーダー選挙

年1回の風物詩、プロジェクトリーダー選挙が行われました。今回は3名立候補うち1名辞退で、結局2名での争いとなり、現プロジェクトリーダーのLucas Nussbaumさんが再選を果たしました——つまり、あまり大きな方針転換などはないということです。

Rubyのデフォルトバージョンが2.0に

その再選したLucas Nussbaumさんがメンテナの1人であるRuby関連パッケージについてですが、デフォルトのバージョン(ruby-defaultパッケージが依存するrubyのバージョン)がようやく1.9.1から2.0へと引き上げられました。これに伴い、Rubyがらみのパッケージで、ビルドできないものがいくつも出ています。また、ruby1.8パッケージがSid(unstable)から削除されるターゲットとなり、整理が進んでいます。

過去、Ruby1.6時代に作られ、かろうじて1.8の時代までアップデートしていたソフトウェアが多くあります。今回、強制的にRuby2.0に引き上げられたことで、1.9.1あるいは2.0の時点でのRuby自体の変更に伴ってビルドや動作に問題が出ていたソフトウェアが、ここにきて炙り出されているかたちです。このまま対応が行われない古いソフトウェアは、Jessieから削除される可能性がありますのでご注意ください[注4]。

なお、さらにruby-defaultは2.0から2.1へのアップデートが予告されていますので、Debian 8「Jessie」ではruby2.1が標準として提供される見通しです。

Rubyがらみで問題のあるパッケージについては、Lucasさんが作成したUDD(Ultimate Debian Database)[注5]を使って簡単に確認ができますので、腕に自信のある方は修正にトライしてみてください。

OpenSSL「Heartbleed」問題

大騒ぎになったので、ご存じない方はいらっしゃらないとは思いますが、OpenSSLに破滅的な問題が見つかりました。些細なバグではあるのですが、リモートからサーバ側のメモリを読むことができてしまいます。これによって機密情報にアクセスが可能となり、場合によっては秘密鍵を盗まれて暗号化を無力化できてしまうという甚大なものです。

DebianではSqueezeまでのリリースは影響を受けず、Wheezy以降で、このバグに対応されたパッケージがリリースされました。とはいえ、それまでに作成した証明書は破棄し、パッケージをアップデートしたうえで再発行が必要になりますので忘れずに実行しましょう。

なお、今回のHeartbeat機能に起因するバグを受けてOpenBSDの開発者らが、動作を安全側に倒すようにしようとしたところ、メモリ管理自体が独自に作りこまれているうえにバグもあり、緩和策がまともに機能しませんでした。そんなことがあって「こんなにひどいコードはダメだ!」ということで大幅な整理を実施しています[注6]。

注3) kFreeBSDやHurdは対象外です。また、systemd以外のinitシステムを排除するわけではないので、その点は誤解ないようにしてください。あくまでも「Linuxカーネル採用アーキテクチャでのデフォルトのinitシステムを何にするのか」という点が明確になっただけです。

注4) できれば該当ソフトウェアの開発を引き継いでいただける方をご存じであれば、該当のバグ報告を転送するなどしていただけると助かります。

注5) URL http://goo.gl/7UWyZg

注6) URL http://www.libressl.org/

Debianも過去にOpenSSLに対してValgrind[注7]で発見した問題を修正しようとしてバグを作りこんでしまい、その結果、SSH鍵や証明書が脆弱な状態になるという問題を作ってしまいました。これも上記のOpenSSL側の特殊な実装を踏んでしまったことが発端ですので、この活動には注目していきたいと思います。

apt「ピン留め」補足

さて、ホットトピックから話題を転換してTipsの話をしましょう。前回は違うバージョンのリポジトリを使う設定を案内しました。これは、Debian以外のサードパーティ製のリポジトリを利用する場合にも応用できます。

たとえば、マルチメディア系のパッケージを独自提供している「deb-multimedia.org」というサイト[注8]があります。こちらにはDebianが提供しているのと同じパッケージ名で、ビルドオプションを変えて違う機能を有効にして提供しているパッケージがあります。そのままapt lineに追加すると、同名のパッケージはバージョン番号の関係上、Debianのパッケージではなくdeb-multimedia.orgが提供するものが問答無用でインストールされますが、あまり行儀がいいとは言えません。

deb-multimedia.orgリポジトリにあるパッケージは、明示的に指定した場合にだけインストールするという設定を行いたい場合は、

```
Package: *
Pin:  o=Unofficial Multimedia Packages
Pin-Priority: 100
```

というようにoriginをPinのキーとして指定して優先度を下げて/etc/apt/preferencesに記述します。

なお、この「ピン留め」作業について、experimentalだけは例外で、apt lineに記載されていても明示的にexperimentalを指定してインストールを実施しない限り、experimentalにあるバージョンは導入されません。よって、apt lineに追加しても、experimentalに関してはピン留めしてアップグレードを防ぐ設定は不要です。

experimentalは独立した存在

experimentalだけが例外なのは、experimentalに入るバージョンはその名前のとおり「実験」を想定して「隔離」を意図したものだからです。unstableのパッケージはtesting、そして最終的にはstableへマイグレーションをすることを前提としていますが、experimentalはその流れから隔絶されているという大きな違いがあります(**図1**)。

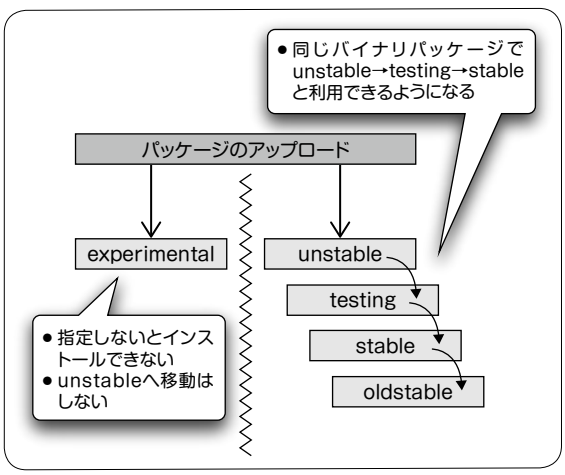
このexperimentalに入るバージョンは、

①非互換性が発生して、unstableへ投入する前に事前調整が必要になりそうな変更が含められている場合
②リリースマネジメント上、unstableに突っ込むとその後のリリースが手間になりそうな場合

のどちらかに該当します。

①についてはメンテナの裁量で判断されます。

▼図1　Debianのリリースの種類と流れ



注7）Linux用のメモリデバッグツール。
注8）旧ドメインdebian-multimedia.orgは商標問題のため破棄されて現在は第三者が取得していますので使わないように注意しましょう。

たとえば、大きなバージョン間のジャンプ(2.xから3.xというようにメジャーバージョンをまたぐなど)があり、さまざまな調整が必要な場合が挙げられます。こういった場合、既存のバージョン自体も(2.xxシリーズというように)それとは別にメンテナンスされており、unstableではそちらを採用し続けるという場合が多いようです[注9]。

②についてですが、リリース前の「フリーズ期間」に入ると、気軽に新しいバージョンや変更を加えたパッケージをunstableへアップロードすることが、ためらわれるようになります。それは「unblock」という気の重いやりとりがあるからです。

Debianではリリース前のある定められた時点で、パッケージのアップデートを凍結(フリーズ)し、その間で把握している重大な問題点(RC:Release Critical Bug、BTSでの重要度が高いもの)をすべて修正するというポリシーを持っています[注10]。リリース直前のtestingにあるパッケージに問題があった場合、それを修正したパッケージをunstableへアップロードします。フリーズ期間中は、ここで必ずリリースチームにアップデートの理由の説明を行い、併せて変更内容のパッチを提示する必要があります。このアップロードしたunstableのパッケージをtestingに移行するのを明示的に許可してもらう作業が「unblock」です。そして、その際には変更による副作用を極力なくすために、変更は「その問題の修正のみに限る」というルールがあります。

upstreamで新バージョンが出たら、通常であれば間を置かずにunstableにアップロードしますが、フリーズ中にtestingにあるバージョンにRCバグが見つかった場合には、先の「問題の修正のみに限る」というルールとぶつかってしまいます。

それを避けるため、通常フリーズ期間中はunstableにアップロードするのをぐっと我慢して、問題を避けてexperimentalへアップロードします。フリーズ期間中はDebian全体にあまり変化が少なくなる「凪」の時期ですが[注11]、メンテナによってはexperimentalへ精力的にアップデートを追加している場合があります。興味があるソフトウェアについてはexperimentalのものを利用するのも検討してみてください。

イベントの宣伝

6月14日に札幌で開かれる「オープンソースカンファレンス2014 Hokkaido」にDebian JP Projectとして参加します。展示やセミナー発表などを行いますので、札幌近郊にお住まいで、ちょっとでもDebian近辺に興味がある方は、ぜひ足を運んでいただければと思います。詳しくは同イベントのページ[注12]を参照してください。

また、その翌日6月15日に、同じく札幌で㈱インフィニットループの会議室をお借りしてイベント「Debian meetup Hokkaido 14.06」(仮)を開催します。こちらは勉強会というほど固くないユルいイベントをめざしています。まだ詳細は決まっていませんが、face to faceでいろいろな話ができるいい機会ですので、こちらも合わせて、参加を検討いただければと思います。情報はdebian-usersメーリングリストやTwitterの「@debianjp」などで、随時提供していく予定です。ご参加をお待ちしています。SD

注9) 正直、「とりあえず変更。誰か地雷踏む可能性高いから、とりあえずexperimentalに突っ込んでおこう」という適当なものもあります……。

注10) 実際のところは、「wheezy-ignore」などのタグを付けてリリースではあえて無視する、重要度を下げてRCと見なさないで処理する、ということもあります。このあたりの判断はリリースチームの判断に依存します。たとえばライセンス関連の例を挙げると、upstream側にてDebianで問題なく取り込めるようなライセンスに変更する予定があるが、事務手続きなどの関係からリリースには間に合わない……というような場合は、重要度が下げられます。同様にライセンス関連であっても、コアなパッケージでの問題で対処をするとなると大きな問題になり得る、という場合(昔あったのはglibc(GNU C Library)のNFS関連のコードのライセンスの問題。現在ですと、localeデータの扱いなど)もいったん無視というかたちが採られます。要は「リリースについては結構現実路線」ということです。

注11) 現在のところ、2014年11月5日以降予定。この2ヵ月ぐらい前からライブラリの大きな変更は許可されません。何か大きな変更が起きそうなものを要望する場合は、お早めにどうぞ。

注12) URL http://www.ospn.jp/osc2014-do/
ハッシュタグは「#osc14do」。

会社のカルチャー

藤田 稜
FUJITA Ryo
レッドハット㈱　グローバルサービス本部
プラットフォームソリューション統括部
ソリューションアーキテクト部長

Truth Happens

本連載第1回に書かせていただいたレッドハットの藤田です。恵比寿通信に書くのは2年ぶりなので前回(2012年3月号参照)は何を書いたのか読み返してみたところ、少しだけレッドハットのカルチャーに触れていたので、今回はその点についてもう少し詳しく紹介してみようと思います。

レッドハットにはいくつか標語のようなものがいくつかあります。「至誠に悖る勿かりしか」のような立派なもの[注1]ではないですが、"Truth Happens"はRed Hatが主催するセミナーなどでムービー[注2]を頻繁に用いていたので見たことがある方も多いと思います(**図1**)。このムービーが格好よかったので転職してきた、なんていう同僚もいたりするのでぜひ見てください。この"Truth Happens"で筆者が面白いと感じるのはそのアイロニーの利いた内容もですが、タイトルにhappenという動詞を用いている点です。happenを用いているということは「真実は(偶然に)起こる」ととらえていて、「何かの結果として必然に起こる」とはとらえられていないということと、筆者は解釈しています。ムービーは歴史をたどる形をとっていて「すでに起こったこと」を紹介しているので、結果論として確かに「真実」は起こっているけれども、それが必然であったか否かは判断していない、という立場をとっているように見えます。

◆　◆　◆

オープンソースソフトウェア(以降、OSS)に長年携わって多くのプロジェクトを筆者は見てきましたが、プロジェクトの成否が多くの「偶然」に依っているような気がします。

弊社を含めOSSに積極的に投資している企業が、あるプロジェクトのスポンサーをする場合、そのプロジェクトが成功すると見込んで投資をしますが、必ずしもうまくいくとは限りません。一方で企業による投資なしでも、多くの開発者やユーザが集まる中でプロジェクトの成功に欠かせないポジションに適切なタレントが当てはまると大きな成功につながることもあります。筆者にはこの状況をうまく説明する言葉が、"Truth Happens"であるように

▼図1 "Truth Happens"のムービースクリーンショット

注1) 旧海軍兵学校の五省。「言行に恥づる勿かりしか」「気力に缺(か)くる勿かりしか」「努力に憾(うら)み勿かりしか」「不精に亘(わた)る勿かりしか」と続く。提督諸氏ならご存じでしょう。

注2) https://www.youtube.com/watch?v=5EkkMfjetEY

思えるわけです。

Red Hat Way

筆者がレッドハットに合流した2005年時点では、レッドハットはグローバルで約700人の従業員を抱える程度の規模の企業でした。現在は従業員数が6,000人を超え、徐々に「大企業っぽさ」を増している傍らで、おもにエンジニアを中心として脈々と受け継がれている文化があります。それは、"Red Hat Way"、つまり「レッドハットのやり方」です。

レッドハットの創業は1993年で、当時から一貫してOSSのみでビジネスをしています。また、エンジニアを中心として会社が成長してきたため、OSSに携わるエンジニア達の考え方が深く染みこんでいるのは当然ともいえます。たとえば、Linuxカーネルの開発はおもにML（メーリングリスト）上で行われています。同様にレッドハット社内でも多くの情報の共有はMLで行われており、執筆時点で確認したところ3,000を超えるMLが社内に存在します。従業員数の半分に相当するMLがあるというのは、ちょっと珍しいのではないでしょうか？

もちろんこれだけのMLが存在するには理由があります。グローバルに開発エンジニアが分散しているため働いているタイムゾーンがバラバラであることはもちろん、参加者の時間を束縛する電話やTV会議と異なり、メールであれば受信者が好きな時に読み、回答することができるという、専ら合理性を優先するがゆえ、です。また、MLであれば数十人・数百人が情報を共有することができるので、コラボレーションツールとして最低限の機能を有し、余計な機能によるオーバーヘッドが少ない点も含め、今でも最も利用頻度の高いツールという位置づけにあります。もちろん容量の大きなファイルを共有するためのWebサイトも用意されていて、メールのプロトコルの仕様によって容量が大きくなってしまう添付ファイルは、そこに置くことが推奨されています。

少し"Red Hat Way"から外れましたので話を元に戻しましょう。MLを使って何をしているのかをマクロな視点から見ると、"Red Hat Way"を構成する最大の要素の1つがわかります。そう「コラボレーション」です。

OSSのコラボレーション

読者の中にはOSSのプロジェクトに参加している方もいると思いますが、「参加」とは何をすることでしょうか。「開発すること」はもちろんOSSへの貢献のうち最大のものですが、以下のようなアクティビティも「参加」と言えますし、さまざまなレベルのアクティビティが組み合わさることこそ、コラボレーションだと言えます。

- コードを書く
- コードをレビューする
- ドキュメントを書く
- 翻訳する
- イラストを描く
- バグを報告する
- 成果物を使う
- 使っていることを「宣伝」する

OSSのプロジェクトに参加するというと、何かとても敷居が高いように感じている読者もいるかもしれません。しかし、その成果物を使ってみて、使ってみたことやその感想をTwitterやFacebook、あるいはブログなどで書くだけでも、その成果物の利用を検討している「次のあなた」には役に立つ情報です。さらに、ひょっとするとコードを書いた人の目にとまってその人を勇気づけることになるかも知れず、決して「参加する」ことは敷居の高いことではないのです。

"Red Hat Way"、あなたもやってみませんか？

SD

UbuntuでMongoDBをJSON電卓に

おがさわらなるひこ　OGASAWARA Naruhiko
Mail ogasawara.naruhiko@miraitsystems.jp

今回は著名なNoSQLであるMongoDBをUbuntuに導入して、JSON電卓として使う方法について紹介します。

はじめに

MongoDBは米国MongoDB Inc.がおもに開発しているオープンソースのデータベースです。いわゆるNoSQLと呼ばれる種類のデータベースの中では知名度があるほうで、さまざまな特徴を持っていますが、「開発者フレンドリー」なところが筆者としては一番気に入っているところです。

一見データベースといえばサーバに導入してアレコレするものと考えがちですが、MongoDBは手元のマシンに入れて電卓的に使ってもけっこう便利ですので、この記事ではそういう内容を紹介しようと思います。

MongoDBとは

繰り返しになりますが、MongoDBとはいわゆるNoSQLと言われるデータベースの1つです。NoSQLというのはSQLを使わない、つまりRDBMSではないデータベースの総称で、RDBMSが不得意だったり性能を出しきるのが難しい場合において、相互補完的に使いましょう、という意味の"Not Only SQL"の略語ということになっています。

MongoDBは、RDBMSより機能をちょっと犠牲にして、その代わりに高い性能を確保しようという狙いで作られています[注1]（**図1**）。

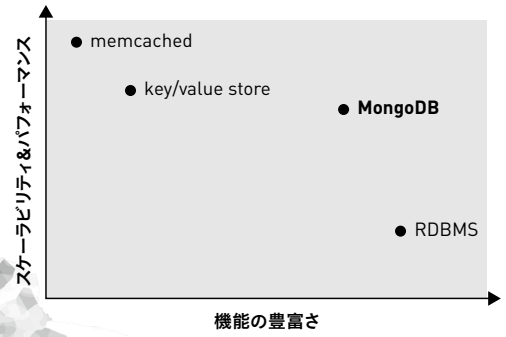
特徴をざっと挙げると、

- 高性能
- ドキュメント指向・動的スキーマによるアプリケーションフレンドリーさ
- レプリカセットによる柔軟な高可用性
- オートシャーディングによる容易な水平スケールアウト

という感じです。今回はとくにドキュメント指向に的を絞ります。

MongoDBは、ひとかたまりのデータは1つの「ドキュメント」として扱います。ドキュメントはJSON[注2]形式で表現されます。RDBMSと違ってJOINが存在しないので、あるドキュメントの中には複数の種類の情報が埋め込まれて（embedded）格納さ

図1　MongoDBの狙い



注1）出典：https://wiki.mongodb.com/pages/viewpage.action?pageId=20743144
注2）JavaScript Object Notation

れます。JSONは木構造ですので複雑なデータをカジュアルに表現できます(**図2**)。

MongoDBにおいては、ドキュメントの集まりを「コレクション」と呼びます。RDBMSのテーブルに対応しますが、テーブルは各列同じ形式であるのに対して、MongoDBの場合は同じコレクションの中にまったく形式が異なったドキュメントを放り込んだとしても問題ありません。それをどう扱うかはアプリケーションの自由です。アプリケーションを作りながらデータ設計できるところが良いところです。

MongoDBをUbuntuに導入する

MongoDBはUbuntuのUniverseリポジトリに入っているので、一番簡単な導入のしかたは単にapt-getでインストールするだけです。

```
$ sudo apt-get install mongodb
```

このコマンドで、クライアント(mongoコマンド)とサーバ(mongodなどなど)が両方入り、localhost:27017で単一構成のMongoDBが動くようになります。

MongoDBは進化が早いソフトウェアですので、公式な配布先からインストールしたいという気持ちもあるでしょう。MongoDB Inc.はUbuntu用にリポジトリを用意している[注3]ので、それをAPTに設定すれば簡単に最新版に追随できます(**図3**)。これも標準リポジトリからの導入と同じように、単一構成のMongoDBが動きます。

インストールできたら、ちゃんと動いているか確認しましょう。mongoコマンドを叩いて、**図4**のような画面が出たら成功です[注4]。Ctrl+Dを押して抜けましょう。

注3) http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/

注4) 「For interactive help,...」からプロンプト(>)までは、初回起動時にだけ表示されます。なお、お使いの環境が32ビット版だと警告がいろいろ出ますが、MongoDBはメモリを多く使うデータベースですのでサーバが32ビットだといろいろ制約があるのです。でも、今回紹介する電卓的利用ならそんなに困ることもないと思います。

図2 JSONによるドキュメント表現

• RDBMS:関係(表)

Entries

ID	TITLE	BODY	A_ID
1	もんごたん楽しい!	MongoDBいいですね!好きです!	1
…	…	…	…

Users

ID	NAME	EMAIL
1	Naruhiko Ogasawara	naruoga@example.com
2	Ichiro Tanaka	tanaka.ichiro@example.com
3	Taro Yamada	taroy@example.com

CommentsLink

ID	ARTICLE_ID	COMMENT_ID
1	1	1
2	1	1
…	…	…

Comments

ID	CONTENT	A_ID
1	ぼくもお気に入り!	2
2	使い所難しいですけどね。	3
…	…	…

• MongoDB:ドキュメント

```
{
  "_id" : ObjectId("52440666..."),
  "title" : "もんごたん楽しい!",
  "body" : "MongoDBいいですね!好きです!",
  "author" : "Naruhiko Ogasawara",
  "email" : "naruoga@example.com",
  "comment" : [
    { "author" : "Ichiro Tanaka",
      "email" : "tanaka.ichiro@example.com",
      "content" : "ぼくもお気に入り!" },
    { "author" : "Taro Yamada",
      "email" : "taroy@example.com",
      "content" : "使い所難しいですけどね。" }
  ]
}
```

↑ JSON
(JavaScript Object Notation)

図3 MongoDB Inc.の公式リポジトリよりインストール

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
$ echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
$ sudo apt-get update
$ sudo apt-get install mongodb-org
```

MongoDBを操作してみる

先ほどお試しで使ったmongoコマンドはMongoDBの対話型シェルです。RDBMSのように専用の問い合わせ言語を持たないMongoDBは、対話型シェルの言語としてJavaScriptを採用しています（エンジンとしてはV8を利用）。--helpサブコマンドでヘルプを見ると、mongoコマンドは次のようなパラメータを取ることがわかります。

```
mongo [<hostname>[:<port>]/][dbname]
```

デフォルトで、ホスト名はlocalhost、ポート番号は27017、データベース名はtestです。したがって引数なしでmongoコマンドを起動すると、インストールの章で説明したlocalhostで動いているMongoDBサーバに接続して、データベースtestにアクセスすることになります。

ドキュメントの挿入

まずは簡単に、データベースに値を挿入してみましょう。データベースはfoo、コレクションはbarとします（**図5**）。useコマンドは利用するデータベースの切り替え、db.<コレクション名>.insert()コマンドは指定したJSONオブジェクトによるデータをコレクションに挿入するものです。

これでデータベースfooのコレクションbarに3個ドキュメントができました。ポイントは、データベースやコレクションを「作る」操作が存在しないことです。useしてinsertするだけで、自動的にデータベースやコレクションは作成されます。これはmongoシェルに限らず、アプリケーションからでも同じです。また3個のドキュメントがまったく異なったスキーマを持っていることにも注意してください。

なお先ほども書いたとおりmongoシェルはJavaScriptですので、当然JavaScriptを使って適当なデータを挿入することもできます。試してみましょう。コレクションはbazにします（**図6**）。ちょっと時間はかかりますが、データベースに10万件のデータを簡単に挿入できました。

ドキュメントの読み込み（検索）

では、挿入したドキュメントをデータベースから取り出してみましょう。データベースの指定したコレクションから「とりあえず1個取り出す」コマンドfindOneを使ってみましょう（**図7**）。findOneは「それを見やすいように整形して表示する（pretty print）」機能も持っています。コレクションに格納されたドキュメントの中身をなんとなく把握したいときに便利です。

図5　データベースに値を設定する

```
// データベースfooを利用
> use foo
// 3個のドキュメントを挿入。スキーマがそれぞれ異なっているのに注意。
> db.bar.insert({name: "naruhiko", age: 43})
> db.bar.insert({name: "taro", hobbies: ["computer", "movie"]})
> db.bar.insert({name: "nobita", sex:"male"})
```

図4　mongoコマンドを最初に起動したときの画面

```
$ mongo
MongoDB shell version: 2.4.6
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
>
```

図7　findOneコマンドによりドキュメントを1つ取り出し

```
> db.bar.findOne()
{
        "_id" : ObjectId("5332a6d518f158574f030ce1"),
        "name" : "naruhiko",
        "age" : 43
}
```

図6　JavaScriptでデータベースに値を設定する

```
> for (var i = 0 ; i < 1000000 ; i++) {
  db.baz.insert({number: i, random: Math.random(), created_at: new Date()})
}
```

ここで気がつくのは、追加したときには存在しなかった"_id"というフィールドが自動的に追加されていることです。このフィールドは各ドキュメントを一意に表すもので、insertのときに存在しない場合は自動的に生成されます。

次に、コレクションbarから値を一気に取り出してみましょう(**図8**)。

findコマンドはfindOneコマンドと異なり、条件に一致するドキュメントをまとめてコレクションから取り出します。先ほど挿入したスキーマが異なる3個のドキュメントが一発で取り出せているのがわかります。

findOne/findとも引数を最大2つ取れます。1個目はquery、2個目はprojectionです。

queryは言葉のとおり「問い合わせ」で、シンプルなものとしては**図9**のような「あるフィールドの値を指定」するものです。非常に多彩なqueryが存在するので、くわしくはほかのドキュメント[注5]を見てください。

注5) MongoDBの公式ドキュメントReference > Operators > Query and Projection Operators (http://docs.mongodb.org/manual/reference/operator/query/)

projectionは取得したいフィールドを指定するものです。たとえば**図10**ではnameフィールドだけを指定しています。ただし、_idフィールドだけは常に表示されるので、それを抑制するには明示的に「0」を指定する必要があります。

queryとprojectionを組み合わせた例としては**図11**をご覧ください。

ドキュメントの編集と削除

当然ですが、一度挿入したドキュメントを書き換えたくなることはあります。そのためのごく一般的なコマンドがupdateです。たとえばbarコレクションのnaruhikoさんにhobbiesを追加する例が**図12**です。

updateコマンドは最大4個の引数を取ります。1つめは変更したい文書を特定するための問い合わせです。2つめはupdateオペレータと呼ばれ、ドキュメントをどう更新するかを指定するものです。これも非常に数多くあるので、別途ドキュメントを参考にし

図8　findコマンドによりドキュメントを一度に取り出し

```
> db.bar.find()
{ "_id" : ObjectId("5332a6d518f158574f030ce1"), "name" : "naruhiko", "age" : 43 }
{ "_id" : ObjectId("5332a6f118f158574f030ce2"), "name" : "taro", "hobbies" : [ "computer", "movie" ] }
{ "_id" : ObjectId("5336e3c78f02919610db28b1"), "name" : "nobita", "sex" : "male" }
```

図9　単純な問い合わせ条件によるfindの例

```
> db.bar.find({name:"naruhiko"})
{ "_id" : ObjectId("5332a6d518f158574f030ce1"), "name" : "naruhiko", "age" : 43 }
```

図10　findで必要なフィールドを指定して問い合わせた例

```
> db.bar.find({},{name:1,_id:0})
{ "name" : "naruhiko" }
{ "name" : "taro" }
{ "name" : "nobita" }
```

図11　findの問い合わせの追加の例

```
// hobbiesフィールドが存在するドキュメント
> db.bar.find({hobbies: {$exists: true}}, {_id:0})
{ "name" : "taro", "hobbies" : [ "computer", "movie" ] }

// nameフィールドが「n」で始まるドキュメント
> db.bar.find({name: {$regex: /n.*/}}, {_id:0})
{ "name" : "naruhiko", "age" : 43 }
{ "name" : "nobita", "sex" : "male" }

// hobbiesフィールドに「computer」があるドキュメント
> db.bar.find({hobbies:"computer"}, {_id:0})
{ "name" : "taro", "hobbies" : [ "computer", "movie" ] }

// numberフィールドが100未満でrandomフィールドが0.4以上0.5未満を満たすドキュメント
> db.baz.find({number:{$lt:100}, random: {$gte:0.4,$lt:0.5}}, {_id:0})
{ "number" : 4, "random" : 0.4927476807497442, "created_at" : ISODate("2014-03-29T15:37:04.899Z") }
{ "number" : 16, "random" : 0.42722524516284466, "created_at" : ISODate("2014-03-29T15:37:04.901Z") }
{ "number" : 19, "random" : 0.49802548601292074, "created_at" : ISODate("2014-03-29T15:37:04.901Z") }
```

てください[注6]。今回利用している$setオペレータは「ほかのフィールドには触れずに新たなフィールドを追加する」ものです。3～4つめの引数は今回は省略します。

ドキュメントを削除する操作ももちろん可能で、コマンドはremoveです。例を**図13**に挙げました。

以上、MongoDBのCRUD（Create、Read、Update、Delete）について簡単に説明しました。では、Ubuntuライフでの活用について考えてみましょう。

注6）MongoDBの公式ドキュメントReference > Operators > Update Operators (http://docs.mongodb.org/manual/reference/operator/update/)

図14　リスト1のJSONファイルをMongoDBに取り込んだ例

```
> use hoge
switched to db hoge
> db.fuga.find().pretty()    prettyな表示
{
        "_id" : ObjectId("533ac6e8e6716de39feabb46"),
        "name" : "naruhiko",
        "sex" : "male",
        "hobbies" : [
                "kayak",
                "computer"
        ],
        "emails" : [
                "naruhiko@example.com"
        ]
}
{
        "_id" : ObjectId("533ac6e8e6716de39feabb47"),
        "name" : "taro",
        "sex" : "male",
        "age" : 20,
        "hobbies" : [
                "computer",
                "cooking"
        ]
}
{
        "_id" : ObjectId("533ac6e8e6716de39feabb48"),
        "name" : "hanako",
        "sex" : "female",
        "hobbies" : [
                "cycling"
        ]
}
```

UbuntuでMongoDBを活用する

MongoDBではデータをほぼJSONで格納しています。そのため、何か手元にJSON形式のデータがあったらMongoDBに放り込んでささっと操作できます。こういう応用だとjqコマンドを使うことが多いですが、MongoDBを代わりに使ってもいいじゃない！　ということです。

例として、**リスト1**のようなJSONファイルを考えます（ファイル名はtest.jsonとします）。

これをMongoDBにインポートするにはmongoimportというコマンドを使います。細かな使い勝手はヘルプを見ていただくとして、データベースhogeのコレクションfugaにtest.jsonの中身を流し込むには次のようにすれば良いです。

```
$ mongoimport --drop --db hoge --collection fuga < test.json
```

さっそくmongoコマンドで開いてみて中身を覗いてみましょう。pretty指定をしているので見やすく表示されています（**図14**）。当然、検索などもごく普通に行えます。

もうちょっと凝った例をやってみましょう。「ricollab 郵便番号検索」というサイト[注7]があり、郵便番号の検索結果をJSONで引っ張りだすことができます。たとえば郵便番号285の情報をJSONで

注7）http://zip.ricollab.jp/

図12　updateコマンドの例

```
// nameがnaruhikoなドキュメントにhobbiesフィールドを追加
> db.bar.update({name:"naruhiko"},{$set: {hobbies: ["kayak", "computer"]}})
> db.bar.find()
{ "_id" : ObjectId("5332a6f118f158574f030ce2"), "name" : "taro", "hobbies" : [  "computer",  "movie" ] }
{ "_id" : ObjectId("5336e3c78f02919610db28b1"), "name" : "nobita", "sex" : "male" }
{ "_id" : ObjectId("5332a6d518f158574f030ce1"), "age" : 43, "hobbies" : [  "kayak",  "computer" ], "name" : "naruhiko" } // <- 追加されている
```

図13　removeコマンドの例

```
> db.baz.find({random:{$lt:0.001} }).count() // random < 0.001 の数
990
> db.baz.remove({random:{$lt:0.001} }) // random < 0.001 のドキュメントを全削除
> db.baz.find({random:{$lt:0.001} }).count() // 再度数を数えると
0 // 削除されているので0になる
```

引っ張りだしてみましょう。次のようなリクエストをcURLで投げれば、**図15**のようなJSONデータが得られます。

```
$ curl "http://zip.ricollab.jp/search?q=285&count=1000&type=json" > zips.json
```

ただし、このJSONデータは1つの塊の中にデータが詰め込まれているので、先ほどとmongoimportのやり方が異なります。具体的には --jsonArray というパラメータを付けます。このデータは後で加工したいので、仮に適当なコレクション（今回はtemp）に入れてしまいましょう。

```
$ mongoimport --drop --jsonArray --db zips --collection temp < zips.json
```

いったんMongoDBで受けてしまえば、単なるJSONデータなわけですから適当に加工すればいいわけです。このデータの場合「result」というキーに各郵便番号のデータが配列で格納されているので、それを取り出してJavaScriptのforEach文を使って新たなコレクションに突っ込めば良いです。

```
> db.temp.findOne().result.forEach(function(doc) {db.addr.insert(doc)})
```

得られた結果の最初の5個だけ覗いてみると**図16**のようになります。あとは好きなように問い合わせしたりソートしたりできるわけです。

おわりに

こんなわけで、一種のJSON電卓としてMongoDBを使うのは意外と便利だということがおわかりいただけでしょうか。

JSONでというと、Webサーバである Nginxがログ をJSONで吐く機能を持っています。こういうものの加工に使ったり、またもっと一歩進んだ応用として、fluentdでのログ収集なんかは今回紹介したノウハウの延長線上にあります。こちらについてもゆくゆくは紹介できればと思います。SD

図15　ricollab郵便番号検索から得たJSONデータ

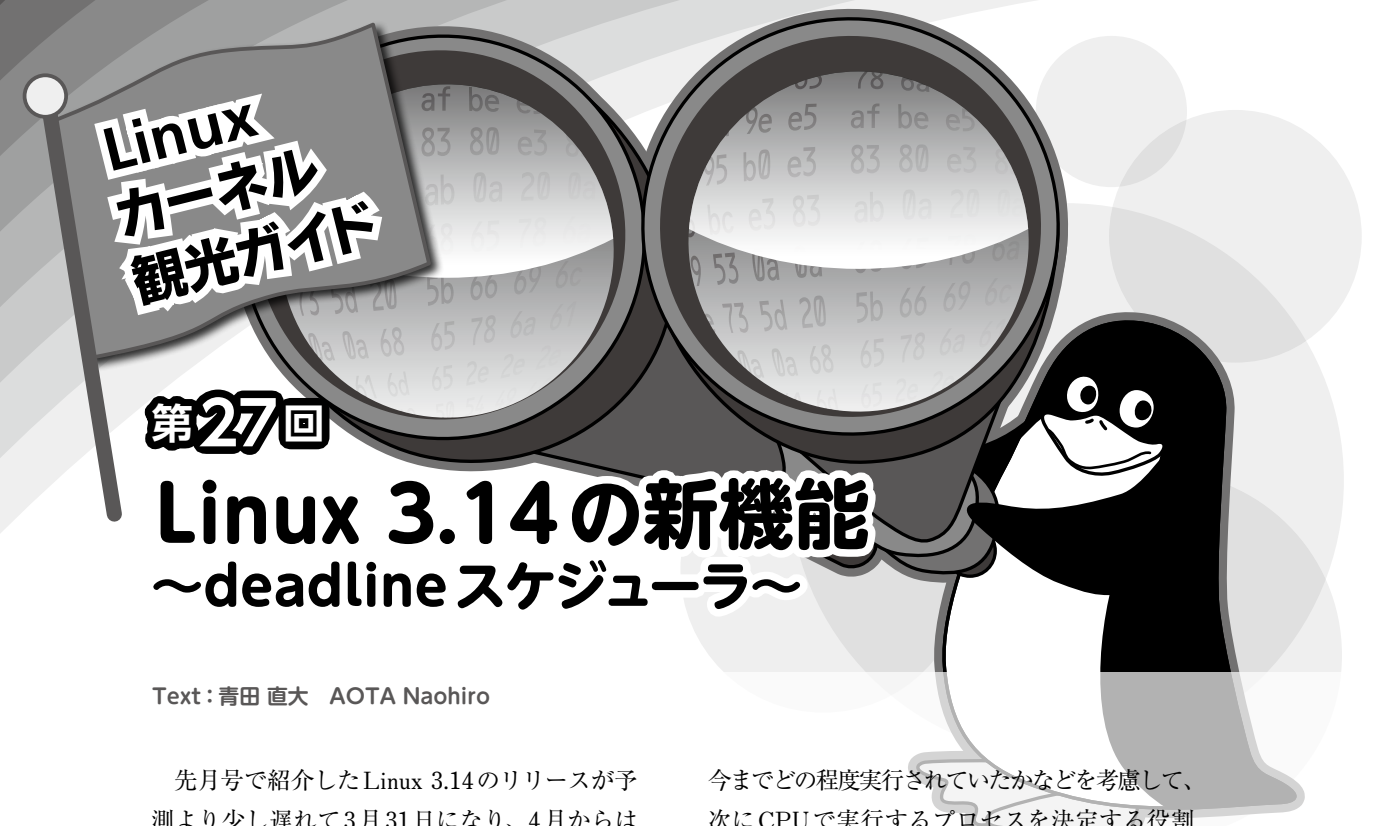
```
{
  "query": "285",
  "totalResults": 143,
  "itemsPerPage": 1000,
  "result": [
    {
      "zipcode": "2850000",
      "address": "千葉県佐倉市以下に掲載がない場合",
      "link": "http://zip.ricollab.jp/2850000"
    },
    {
      "zipcode": "2850001",
      "address": "千葉県佐倉市土浮",
      "link": "http://zip.ricollab.jp/2850001"
    },
    {
      "zipcode": "2850002",
      "address": "千葉県佐倉市萩山新田",
      "link": "http://zip.ricollab.jp/2850002"
    },
    ...
  ]
}
```

図16　加工して番号ごとに格納された郵便番号データ

```
> db.addr.find().limit(5).pretty()
{
        "_id" : ObjectId("533aced9c713c50b8ee46853"),
        "zipcode" : "2850000",
        "address" : "千葉県佐倉市以下に掲載がない場合",
        "link" : "http://zip.ricollab.jp/2850000"
}
{
        "_id" : ObjectId("533aced9c713c50b8ee46854"),
        "zipcode" : "2850001",
        "address" : "千葉県佐倉市土浮",
        "link" : "http://zip.ricollab.jp/2850001"
}
{
        "_id" : ObjectId("533aced9c713c50b8ee46855"),
        "zipcode" : "2850002",
        "address" : "千葉県佐倉市萩山新田",
        "link" : "http://zip.ricollab.jp/2850002"
}
{
        "_id" : ObjectId("533aced9c713c50b8ee46856"),
        "zipcode" : "2850003",
        "address" : "千葉県佐倉市飯野",
        "link" : "http://zip.ricollab.jp/2850003"
}
{
        "_id" : ObjectId("533aced9c713c50b8ee46857"),
        "zipcode" : "2850004",
        "address" : "千葉県佐倉市岩名",
        "link" : "http://zip.ricollab.jp/2850004"
}
```

リスト1　JSONファイルの例（test.json）

```
{name:"naruhiko", sex: "male", hobbies:["kayak", "computer"], emails: ["naruhiko@example.com"]}
{name:"taro", sex: "male", age: 20, hobbies: ["computer", "cooking"]}
{name:"hanako", sex: "female", hobbies: ["cycling"]}
```



第27回 Linux 3.14の新機能 ～deadlineスケジューラ～

Text：青田 直大　AOTA Naohiro

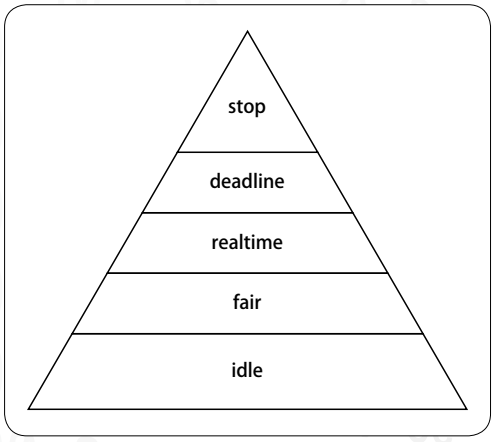
先月号で紹介したLinux 3.14のリリースが予測より少し遅れて3月31日になり、4月からは3.15の開発が始まっています。Linux 3.15にはすでにかなり多くのコミットがなされており、だいぶ大きなリリースとなるようです。3.15にどんな機能が入ってくるのか楽しみですね。今月はまだ3.15ではなくLinux 3.14の新機能からdeadlineスケジューラについて紹介します。

スケジューラ

スケジューラはカーネルのコアというべき機能です。スケジューラは優先度や、プロセスが今までどの程度実行されていたかなどを考慮して、次にCPUで実行するプロセスを決定する役割を果たします。

Linuxではそれぞれのプロセスが、次のいずれかのスケジューリングクラス(**図1**)に所属しています。このそれぞれのクラスではそれぞれ違ったスケジューリング方法が採られており、多様なスケジューリングを行うことを可能にしています。これらのクラスは完全に上下関係にあり、上のクラスで実行可能なプロセスがあれば必ずそのプロセスが実行され、下のクラスのプロセスは上のプロセスに実行可能なプロセスがなくなるまでは実行されません。

▼図1　スケジューリングクラス



- stop_sched_class
- dl_sched_class
- rt_sched_class
- fair_sched_class
- idle_sched_class

stop_sched_class

stop_sched_classは、最上位のスケジュールクラスです。ここに所属するプロセスは、ほかのどんなプロセスよりも優先されます。どんなプロセスの実行も割り込むことが可能で、逆にほかのどんなプロセスにも止められません。こ

のクラスはユーザ空間からは設定できず、カーネルのみが使用しています。カーネルでも動作中のCPUを停止し、そのCPUで実行中のプロセスをほかのCPUにマイグレーションする場合など限られた場面のみで使われています。

dl_sched_class

dl_sched_classは、Linux 3.14で追加された今回のメイントピックとなるスケジュールクラスです。事前に決められたデッドラインまでに、プロセスに設定された分だけのCPU時間を割り当てるように動作します。のちほど詳しく解説します。

rt_sched_class

rt_sched_classは、リアルタイムプロセス用のスケジュールクラスです。このクラスではプロセスの優先度が設定されています。優先度は、いわゆるnice値とは別のもので、優先度が高いプロセスは優先度の低いプロセスの実行を割り込みます。すなわち、優先度が3のプロセスが実行中であっても優先度が4以上のプロセスが実行可能になれば、そこで優先度が3のプロセスは停止し、優先度が上のプロセスが実行されます。

このクラスにはSCHED_FIFOとSCHED_RRという2つのスケジューリングポリシーがあり、どちらのポリシーであるかによってスケジューリングの挙動が変わります。

SCHED_FIFOは"First In, First Out"の名のとおり、(同じ優先度の中では)最初に実行可能になったプロセスが、プロセスの終了やI/O処理などで実行可能でなくなるか、sched_yield()関数を用いてプロセスがCPUを手放すまで実行を続けることになります。

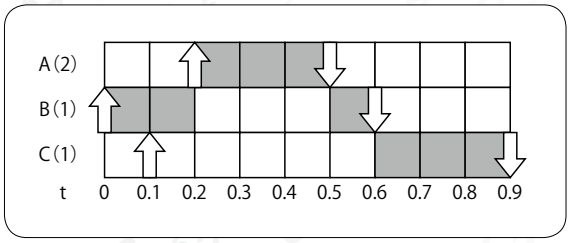
もう一方のSCHED_RRでは一定の期間(タイムスライス)ごとにラウンドロビンにプロセスを切り替えます。

SCHED_FIFOとSCHED_RRの動きを実際に見てみましょう。優先度2のプロセスA、優先度1のプロセスBとCを考えます。

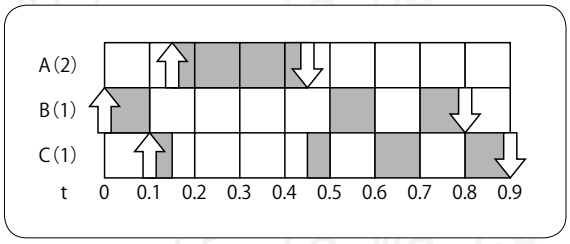
SCHED_FIFOでは、Aは時刻t=0.2秒に、Bはt=0に、Cはt=0.1秒に実行可能になるとします。上向きの矢印がプロセスが実行可能になったことを、網かけの部分がプロセスが実行されている時間を、下向きの矢印がプロセスの実行終了を示しています。1マスが0.1秒を表しています。**図2**では、t=0でBが立ち上がり、t=0.1でCが立ち上がります。BとCは同じ優先度ですのでBの実行は割り込まれず、Bが実行を続けます。しかし、t=0.2でAが立ち上がると、Aの優先度のほうがBよりも高いのでBの実行は割り込まれ、Aが実行されるようになります。t=0.5でAの実行が終了すると、再びBが実行され、Bがt=0.6で終了するとCが実行されるようになります。

次に、SCHED_FIFOの場合とほぼ同じ設定をSCHED_RRで動かした場合について見てみましょう(**図3**)。ここではタイムスライスは0.1秒とし、Aはt=0.15に、Bはt=0に、Cはt=0.1に実行可能になるとします。t=0.1でCが立ち上がると、Bはすでにタイムスライスを使いきっているので停止されCが実行されます。その後、t=0.15でAが立ち上がるとFIFOの場合と同様に、優先度の高いAに割り込まれAが実行され

▼図2 SCHED_FIFOの動き



▼図3 SCHED_RRの動き



ます。このときCがまだタイムスライスを使いきっていないので、t=0.45でAの実行が終われば残りのタイムスライスの分だけふたたびCがt=0.5まで実行されます。これ以降はBとCがそれぞれ0.1秒ずつ実行されていきます。

fair_sched_class

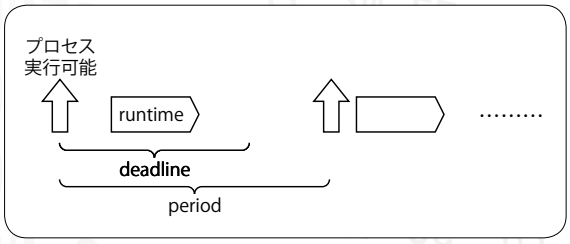
fair_sched_classはいわゆる「普通のプロセス」のためのスケジューリングクラスです。その名のとおりnice値を考慮しつつ、全プロセスが「公平」にCPU時間を使用できるようにスケジューリングを行うスケジューリングクラスとなっています。この実装にもいろいろとおもしろいところがあるのですが、今回は詳しくは扱いません。

このクラスではSCHED_NORMAL、SCHED_BATCH、SCHED_IDLEという3つのスケジューリングポリシーを使用できます。SCHED_BATCHは普通のプロセスよりも割り込まれるまでの期間が長めに設定されます。そのため普通のプロセスよりもキャッシュを活用しやすくなります。ただし、インタラクティブ性能は低くなります。名前のとおり、バッチ処理を行うプロセスに向いているポリシーということになります。SCHED_IDLEはnice値の19よりも実行されないポリシーとなります。このポリシーのプロセスは、CPUがアイドル状態のときだけ実行されるプロセスと考えることができます。

idle_sched_class

idle_sched_classは最下位のスケジュールクラスで、stop_sched_classと同じく、ユーザ空間から設定することのできない特殊なクラスです。このクラスは必ずほかに実行すべきものがないときに用いられるアイドルプロセスを返します。

▼図4　Earliest Deadline Firstのパラメータ



Earliest Deadline First

deadlineスケジューラではEarliest Deadline First(EDF)という方法で、次に動くべきプロセスを決定します。EDFでは各プロセスに“runtime”、“period”、“deadline”という3つのパラメータが設定されています(**図4**)。プロセスは“period”マイクロ秒ごとに“runtime”マイクロ秒実行できます。また、この実行される“runtime”マイクロ秒間は“period”から“deadline”マイクロ秒内にあるようにスケジュールされるようにします。具体的にはdeadlineが一番早いプロセスを選択してそのプロセスを実行します。

では、実際にEDFで3つのプロセスを動かしてみたときの様子を見てみましょう(**図5**)。プロセスAはruntime=1、period=deadline=3、プロセスBはruntime=1、period=deadline=4、プロセスCはruntime=2、period=deadline=7とし、t=0でA、B、Cすべてが実行可能になるとします。A、B、Cがそれぞれ処理するべきデータが時刻3、4、7ごとに訪れ、次のデータが着くまで前のデータを処理しなければいけない……といった状況になります。

時刻t=0ではA、B、Cのdeadlineがそれぞれ3、4、7であるのでAが実行されます。t=1でAの実行が終了すれば、実行可能なプロセスB、Cのうちdeadlineが早いBが実行されます。同様にt=2でCが実行されますが、t=3でAが実行可能になりAとCとではAのほうがdeadlineが早いのでAが実行されます。t=4でAの実行が終了するとCの実行が継続されます。以後同様にA、B、Cが実行されていきます。

さて、現実にはプロセスがこのように理想的に動作するとは限りません。EDFをそのままうまく動作させるには、runtimeを最悪の場合の実行時間よりも長く、periodを

処理すべきデータが来る間隔の中で最小に設定しなければいけませんが、実際にはこれは困難です。設定ミス／予測ミスがあった場合にEDFがどのような挙動になってしまうのかを見てみましょう。

以前とほぼ同じ設定ですが、Aが予測された実行時間1ではなく2ずつ動作したときを考えてみます(**図6**)。t=0で動作を開始したAはdeadlineが一番早いままですのでt=2まで動き続けます。t=8まではそのままうまく動き続けますが、t=8で実行を開始するAはt=10まで実行してしまい、設定されたdeadlineを守れなくなってしまいます。さらにそれが波及して、t=12ではBが、t=14ではCがdeadlineを守れなくなっています。1つのプロセスが事前の想定に合わない動作をするとこのようにほかのプロセスもdeadlineを守れなくなっていきます。

Constant Bandwidth Server

こういった現実の問題に対処するためにLinuxのdeadlineスケジューラでは、EDFのほかにConstant Bandwidth Server(CBS)を使用しています。CBSでは前述の3つのパラメータのほかに、"scheduling deadline"と"current runtime"という2つの変数を使用してプロセスの管理を行います。deadlineパラメータがプロセスが実行可能になってからの時間という意味で相対的であるのに対し、"scheduling deadline"はプロセスを実行させなければいけない絶対的なdeadlineを意味しています。また、current runtimeはプロセスが実行可能な残り時間を示しています。CBSは次のルールに基づいてプロセスのscheduling deadlineを決定します。

❶プロセス生成時はscheduling deadline、current runtimeは0に設定する
❷プロセスが実行可能になったときに「scheduling deadlineが現在時刻よりも前である」か**式1**を満たせば、

```
scheduling deadline = 現在時刻 + deadline
current runtime = runtime
```

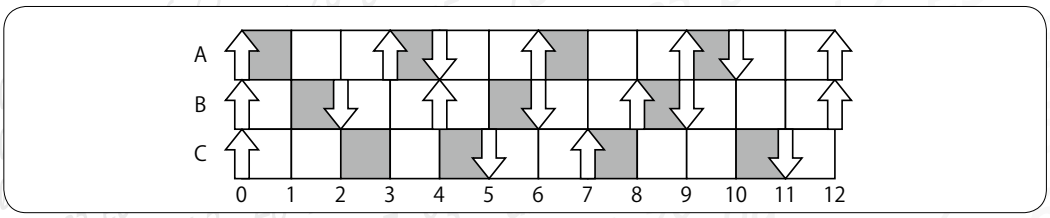
と設定する。

❸current runtimeはプロセスが実際に実行した分だけ減らされる
❹current runtimeが0以下になると、プロセスは"throttle"され"replenishment time"(＝現在のscheduling deadline)までこのプロセス

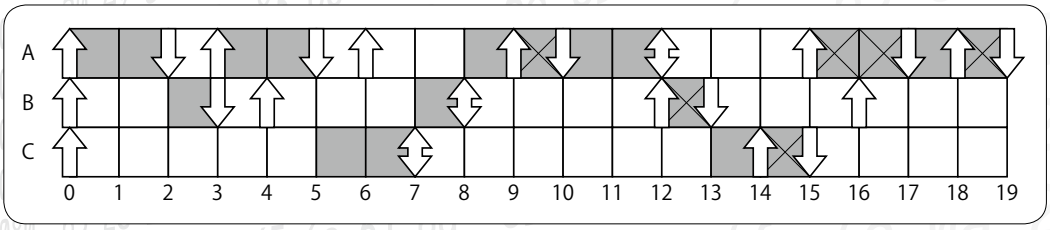
▼式1

$$\frac{\text{current runtime}}{\text{scheduling deadline} - \text{現在時刻}} > \frac{\text{runtime}}{\text{period}}$$

▼図5　EDFの動き



▼図6　deadlineの設定が間違った場合



は実行されなくなる

❺replenishment timeになるとcurrent runtimeが正になるまで次の式が実行される。

```
scheduling deadline = scheduling deadline
+ period
current runtime = current runtime + runtime
```

▼表1　初期化値

	CR	SD
A	1	3
B	1	4
C	2	7

▼表2　それぞれのパラメータ

	runtime	period	deadline
A	2	5	5
B	5	9	9

▼表3　CR、SDの時間変化

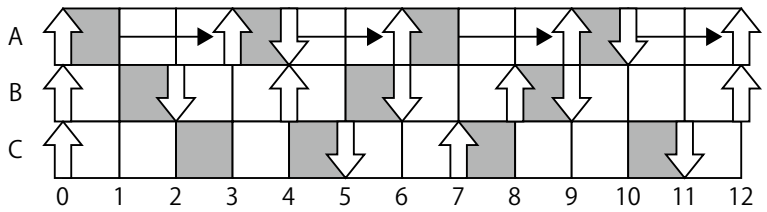
t	A: CR	A: SD	B: CR	B: SD
0	2	5	5	9
1	1	5	5	9
2	0	5	5	9
3	0	5	4	9
4	0	5	4	9
5	2	10	4	9
6	2	10	3	9
7	2	10	2	9
8	2	10	1	9
9	2	10	0	9

このルールを考えて、先ほどのAが設定されているruntimeである1よりも長く動く状況を見てみましょう(**図7**)。t=0でA、B、Cが実行可能になると2番のルールから、A、B、Cのcurrent runtime(**表1**のCR)とscheduling deadline(**表1**のSD)とがそれぞれ**表1**のように初期化されます。

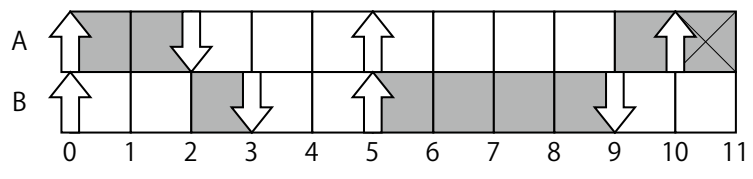
そして、scheduling deadlineが一番小さいAが実行されます。t=1では、4番5番のルールがあるため、Aがt=2まで実行しようとしてもスケジューラによってほかのプロセスの邪魔にならないようにreplenishment timeであるt=3まで停止されてしまいます。このように1つのプロセスの影響がほかのプロセスにまで及ばないようになっています。

もう1つ別の例でルール2の意味を考えてみましょう。ここではプロセスA、Bのruntime、period、deadlineをそれぞれ**表2**のように設定します。Aのほうはこれまでと同じく正確にperiodに定期的に実行可能になり、runtimeだけの時間をぴったり連続して使うプロセスとしますが、Bのほうを1期間だけ実行された後、2期間待ってから再び実行可能になるプロセスとします。そうするとAとBのcurrent runtime(**表3**のCR)とscheduling deadline(**表3**のSD)は、ルール2がなければ**表3**のように推移します。t=5からt=9の間までscheduling deadlineが小さいBが実行されてしまいますが、そのためにA

▼図7　CBSを使った場合の動き



▼図8　スリープするプロセスがいた場合



はdeadlineを逃してしまうことになります(**図8**)。

では、これがルール2が適用されるとどうなるでしょうか(**表4**)。t=5で**式1**の左辺が、**式2**であり、右辺が**式3**となり、式1が成立するのでルール2に従ってscheduling deadlineとcurrent runtimeが更新されます。するとt=5からt=7ではAが実行され、またBはt=7からt=12まで実行されるので、実行可能になってから"dealine"期間の間に、"runtime"だけの実行時間が保証されるスケジューリングが実現できていることがわかります。**式1**によって1期間あたりの対象プロセスへの割り当て率を超えずに、現在のdeadlineまでに残りのruntimeを実行できるのかどうかを見ているということになります(**図9**)。

deadlineの帯域幅

ここまでリアルタイムやdeadlineのスケジューラがほかのプロセスが何もないような前提で話してきましたが、実際にはこうしたプロセス以外にも普通のプロセスが同時に動作しているものです。最初に説明したように、deadlineやリアルタイムのプロセスが普通のプロセスよりも優先されますが、それにも上限があります。すなわちデフォルトでは最大でもCPU時間の95%までがdeadlineなどのプロセスに割り当てられるようにできています。この設定は/proc/sys/kernel/sched_rt_period_us と /proc/sys/kernel/sched_rt_runtime_usによって参照／変更できます。sched_rt_perioid_us分のsched_rt_runtime_usがdeadline、リアルタイム用の上限となります。以下の値であれば1000000でus＝1秒、950000でus＝0.95秒となっています。sched_rt_runtime_us = -1と設定すると制限なし、となります。

```
$ grep . /proc/sys/kernel/sched_rt_*
/proc/sys/kernel/sched_rt_period_us:1000000
/proc/sys/kernel/sched_rt_runtime_us:950000
```

まとめ

今回はLinuxのスケジューラの概要と、Linux 3.14で追加されたdeadlineスケジューラについて解説しました。SD

▼式2

$$\frac{\text{current runtime}}{\text{scheduling deadline} - \text{現在時刻}} = \frac{4}{9-5} = 1$$

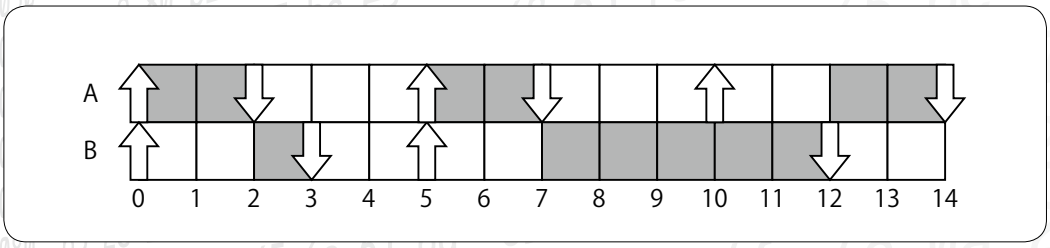
▼式3

$$\frac{\text{runtime}}{\text{period}} = \frac{5}{9}$$

▼表4　ルール2が適用された場合

t	A: CR	A: SD	B: CR	B: SD
5	2	10	5	14
6	1	10	5	14
7	0	10	5	14
8	0	10	4	14
9	0	10	3	14
10	2	15	2	14
11	2	15	1	14
12	2	15	0	14
13	1	15	0	14
14	0	15	0	14

▼図9　ルール2適用時の動き



June 2014

NO.32

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp

jusとコラボレーションしませんか?

jusは1年中絶え間なく行事を開催しているのですが、たまたまここ数ヵ月は行事がありません。そこで今回はイベントのレポートをお休みして、jusの活動を他団体とのかかわりという観点で振り返ってみます。また、記事の後半では、2014年度の会員募集と活動予定をお知らせします。

他団体とのコラボレーション

■昔のjusイベントは単独主催だった

筆者がjusの幹事に就任したのは1992年ですが、その時代のjusの行事はおおむね単独で主催していました。おもなイベントとしては、論文発表の場であるシンポジウム、UNIX関連機器やソフトウェアの展示会であるUNIX Fair、テーマ別のワークショップや月例の勉強会などです。これらをすべて幹事会で運営していましたが、当時は専属の事務局員がいたので、事務的な作業は事務局員による貢献が大きかったです。

■時代はコミュニティとの共同運営に

1999年と2001年に開催したオープンソースまつりのころから、おもにオープンソース系のソフトウェアを中心に数多くのユーザグループが活動を始め、jusの行事もそれらのコミュニティと共同で運営するものが増えていきました。現在の活動の中から、そのようなスタイルを採っているイベントをいくつか紹介します。

- **Lightweight Languageイベント**
  2003年から行っている、おもにプログラマ向けのイベントです。Perl、PHP、Python、Rubyのコミュニティとの共同開催という形でスタートし、その後はほかの言語コミュニティも巻き込んで、毎年8月にカンファレンスを開催しています。このイベントにおいては、jusはプログラム企画よりも運営面での貢献が大きく、チケット販売、会計、サーバ提供などを行っています
- **インターネットコンファレンス**
  1996年から続く、インターネット分野における論文発表の場です。当初は日本ソフトウェア科学会、WIDEプロジェクト、jusの3団体で主催していましたが、IT系の各種学会にもご協力いただき、毎年秋に開催地を変えながら開催を続けています。このイベントにおいては、jusは論文の査読やレポート作成で貢献しています
- **Internet Week**
  インターネットの管理／運用に携わる人々が集まるカンファレンスで、1997年から続いています。主催は日本ネットワークインフォメーションセンター（JPNIC）ですが、プログラムの企画はインターネット系の各種団体から集まった面々によるプログラム委員会が行っています。jusからも数名の幹事がプログラム委員会に参加し、セッションの立案や講師交渉において貢献しています
- **関西オープンフォーラム**
  おもに関西で活動するコミュニティや企業が集まるイベントで、通称KOFとして親しまれています。展示会、セミナー、ステージなど多様なプログラムを内包しているのが特徴の1つです。jusは2002年の開始時から共催という形でかかわり、実行委員の供出や開催費用の一部負担などを行ってきましたが、継続的な運営の見通しが立ったこ

とから、昨年より共催ではなく参加団体としての人的協力に切り替えました

■jusと一緒に何かやりませんか？

jusはさまざまな団体とコラボレーションしてきましたが、その中で感じたこととして、jusにはほかの団体と比べて次のような特徴があると思います。

- 商用UNIXの時代（20年以上前）からイベントを運営してきたので、イベント業者が手がけるものからボランティアベースのイベントまで、豊富な運営経験を持ち合わせています
- 時代の流れに応じてさまざまな技術分野とかかわってきたので、幅広いジャンルの技術者およびコミュニティとつながりがあります
- 毎年会計をまとめ、税務署にも申告しています。イベントなどで発生する金銭の扱いに苦労しているコミュニティも見受けられますが、jusは金銭の出入りは当然あるものと受け止めていて、まったく苦にしていません

このような特徴を考慮すると、複数のコミュニティが共同で行うイベントや会計処理が必要になりそうなイベントは、jusが力を発揮できる場と言えそうです。みなさんの中で、「コミュニティの活動範囲をもっと広げたい」「イベントで扱うお金の処理が面倒なんだけど……」という方は、jusと組めばうまくいくかもしれません。我々も従来の活動にとどまることなく、より多くの団体と交流し、盛り上げていきたいと考えています。jusと一緒に何かやりたいとお考えの方は、jus事務局までご相談ください。

E-mail：office@jus.or.jp　Twitter：@jus_pr

2014年度会員募集

jusの活動は会員が支えています。jusへの入会は随時受け付けていますが、6月に新年度を迎えるにあたり、とくに積極的に会員募集を行っています。jusの活動に賛同される方々の参加をお待ちしております。jus会員の種別、資格、年会費は**表1**のとおりです。

jus会員の特典としては、次のものがあります。

- 勉強会やワークショップなどの参加費割引
- イベント告知やレポートなどを載せた会員向けニュースレター
- 会員向け機関誌「/etc/wall」

2014年度の行事としては**表2**のものを予定しています。

jusへの入会をご希望の方は、jusのWebサイト（http://www.jus.or.jp/）に申込フォームを用意していますので、そちらからお申し込みください。また、不明な点がありましたら、jus事務局までお問い合わせください。SD

▼表1　会員の種別、資格、年会費

種別	資格	初年度年会費	次年度以降
個人会員	個人	7,000円	6,000円
学生会員	学生	4,500円	3,500円
法人会員	組織（営利団体）	95,000円	90,000円（1口あたり）
賛助会員	個人または組織（非営利団体）	45,000円	40,000円

▼表2　2014年度jus行事予定

イベント	時期	場所
定期総会	7月	東京都内
Lightweight Languageイベント	8月23日	日本科学未来館
インターネットコンファレンス2014	11月4～5日	広島市内（予定）
関西オープンフォーラム2014	11月7～8日	大阪南港ATC
Internet Week 2014	11月後半	東京都内（予定）
勉強会	随時	東京、大阪、名古屋など
研究会	年6回程度	全国各地
ワークショップ	年2回程度	未定
運用研究会	不定期	未定
その他協力イベント	オープンソースカンファレンス、Developers Summit、TechLIONなど	

Hack For Japan
エンジニアだからこそできる復興への一歩

●Hack For Japanスタッフ
鎌田 篤慎　KAMATA Shigenori
Twitter @4niruddha

第30回

Hack For Japan 3.11～3年のクロスオーバー振り返り（前編）

“東日本大震災に対し、自分たちの開発スキルを役立てたい”というエンジニアの声をもとに発足された「Hack For Japan」。今回は3年間の活動を振り返ったイベントの報告です。

3年間の活動を共有

2011年3月11日に発生した東日本大震災から3年が経ちました。自分たちの開発スキルを役立てたいという開発者の想いを形にし、これまでITの技術を中心にした復旧復興支援を行ってきたHack For Japanも3年目の節目を迎えました。3年間の活動を通じて、宮城県、岩手県、福島県などの被災各地や復興支援活動に携わってきたさまざまな人々と出会い、コミュニティとして大きな広がりを見せています。この連載でも何度かご紹介した福島県は、中通り、浜通り、会津など歴史的な土地柄で、これまではあまりIT開発者同士の交流も進んでいませんでしたが、震災を契機に復旧復興支援といった面から交流が深まっています。

Hack For Japanの活動に共感し、一緒に活動してきた各地の開発者達をインターネット中継でつなぎ、これまでの振り返りを行うイベント「Hack For Japan 3.11～3年のクロスオーバー振り返り」を開催しました。その様子を今月と来月にかけてレポートしたいと思います。

Hackを通じて各地に拡がった開発者達の輪

2014年3月11日、日本各地をGoogle+ハングアウトオンエアでインターネット越しに中継し、3年が経過した東日本大震災と各地の開発者が行ってきた活動について振り返りを実施しました（**図1**）。宮城県からは仙台と石巻の2会場から、岩手県からは釜石会場、福島県からは中通りの郡山、浜通りの南相馬、会津のそれぞれにある3会場から、そのほか、東京会場、大阪会場、そして、海外のシドニーから、それぞれ中継で各地で活動する開発者達をつなぎました。それぞれの地域の発表をお伝えします。

◆図1　イベントはGoogle+ハングアウトオンエアで行われた

震災から3年、被災地への黙祷

今回のイベントで各地の中継をつなぐファシリテーターは、宮城県を中心に活動し、Hack For Japanのメンバーでもある小泉勝志郎さんが務めました。まず各地とのGoogle+ハングアウトオンエアでの接続が確認できた後、小泉さんの合図と共に今回のイベント参加者全員で黙祷を捧げました。

今回の「クロスオーバー振り返り」では、1人あたり10分を目安に、各会場の代表者がこれまでの活動の振り返りを発表し、良かったところ、不味（まず）かったところを共有しました。これからの活動のヒント、各地の活動で得られたナレッジを得て、より効果的な復興に向けての活動につなげることが狙いです。また、各地で活動する人達の中で、まだ顔見知りではない人同士をつなげていくことで、今後の活

動のさらなる発展も考えていました。

継続的な町おこしで自立した復興を目指す

トップバッターはファシリテーターの小泉さんがいる仙台会場から。小泉さんの自己紹介と共に、震災当時の様子を振り返ってもらいました。震災当時、小泉さんは塩竈在住で自宅が津波被害にあい、当時は自宅で釣りができたという話を交えつつ、塩竈の防波堤のような形で津波被害が大きかった浦戸諸島の様子と、弟さんが浦戸諸島の漁業復興を目指すために始めたクラウドファンディング「うらと海の子再生プロジェクト[注1]」の紹介をしていただきました。このプロジェクトは全国からおよそ1億8千万円ほどの資金が集まって、国内のクラウドファンディングの中でも大きく成功したものとなり、浦戸諸島の漁業復興に大きく貢献されたそうです。

また、仙台でも過去に何度か開催されたHack For Japanのハッカソンを振り返り、ハッカソンののち継続して開発が行われたプロジェクトの少なさから、継続的な活動の必要性を感じられました。その実現を目指して、町おこしが継続的な取り組みとなるようオープンガバメントの推進を図っているCode for Japan[注2]と連携し、地方自治体との協調も視野に入れたCode for Shiogama[注3]を立ち上げました。ITの力、いわゆるシビックハックで塩竈の町を盛り上げようとされています。例として、塩竈の観光スポットを位置情報とセットでデータとして用意し、それを元にした観光アプリの開発や、いくつかのイベントの開催、また島全体をハックしてしまうという「島ソン」を企画中と紹介してくれました。

このように震災直後の復旧にお金が必要な段階での「うらと海の子再生プロジェクト」のような話から、自立した復興、町おこしにつながるようなCode for Shiogamaのような活動の段階に移りつつあります。また、過去の活動でプロジェクトが継続しなかった失敗から学び、持続的な活動となることを目指して、今では塩竈の「楽しさ」を伝え、これまで以上に多くの人々を定期的に現地に呼ぶことにつながる施策となるように意識されているとのことでした。

また、被災地を応援する活動としては、ボランティアのほかにも、こうした現地の活動をインターネットなどを通じて紹介するようなことも、十分に被災地のためになるというお話でした。塩竈の町の楽しさを伝えるUstreamの企画などもスタートしているとのことで、今後の小泉さんの活動にも注目し、読者の皆さんの知り合いの方達にも、ぜひご紹介いただければと思います。

復興と過疎化の問題を継続して考える

続いて、同じく宮城県塩竈市で震災復興団体「よみがえれ！塩竈[注4]」を運営する土見大介さんからの発表です。震災からの復興のフェーズが変っていく中で、活動自体の変化とこれからの展望を話していただきました。

先に紹介したとおり、塩竈市の津波被害が小さく済んだのは浦戸諸島が防波堤のような役割を果たしたからで、逆に言えば浦戸諸島全体が深刻な津波被害にあったということが言えます。また、小さな島が多く、津波によって損壊した建物や流されてきた瓦礫を運び出す重機の搬入なども進まず、復興に時間のかかる状況が続いていました。またそれとは別の問題として、震災以前から浦戸諸島から若者が離れ、そこに住む人々の高齢化が進んでいた現状がありました。

塩竈や浦戸諸島がそのような課題を乗り越え、本当の意味で復興を果たした際には、ここでの知見が震災からの復興だけでなく、他の過疎化が進む地方でも必ず役に立つと信じて活動されているということでした。

そうした想いで活動されている「よみがえれ！塩竈」ですが、団体としての立ち上がりは震災直後、避難所を回って安否情報をTwitter上で発信していた土見さんに呼応する形で人が集まりました。塩竈の人々、あるいは塩竈出身の人達が自然とインター

注1 http://www.uminoko-saisei.net/
注2 http://code4japan.org/
注3 https://www.facebook.com/CodeForShiogama
注4 http://yomigaereshiogama.jp/

ネット上で集まり、被災当初の復旧を支援する活動から、「塩竈でがんばる人達を応援する」という目的に変化していったそうです。ここでもやはりキーワードとなったのは“継続した活動”という点です。震災から3年が経ち、人々の関心が低下していくというのは、我々Hack For Japanでも感じているところですが、被災地で活動されている方々のほうが顕著に感じられているのかもしれません。そうしたこともあり、現在の「よみがえれ！塩竈」のメンバーは「できることを、できる人が、できる範囲で」をテーマに、各メンバーが無理のない範囲での活動を行うようにしているとのことです。

震災直後の安否確認、復旧後は塩竈の特産品の地方販売などで地元の生産物の認知向上と販売を手がけ、現在では塩竈のコミュニティ創出を支援する活動に変遷してきました。地元にゆるいつながりをもったコミュニティを形成し、地元を見つめ直す機会を作るために、気軽さや楽しさをより伝えていくことを目指しているそうです。こういった活動は、土見さんが目指している震災からの復興だけでなく、過疎化という課題を抱えている地方へのヒントがあるかもしれません。また、緩やかなコミュニティ作りは次なる災害が発生した際に、顔見知り同士であるという利点を活かした初動の早さや防災にもつながります。Hack For Japanでもこのような視点は今後の活動の参考となりました。

写真とITで家族の絆を再確認した山元町

次の発表は津波被害が大きかった山元町で、持ち主が不明になってしまった写真約75万枚をITを使って救済、返却するプロジェクト「思い出サルベージ[注5]」を運営している溝口佑爾さんからの発表となりました。

山元町は宮城県の東南端に位置し、太平洋沿岸部の街であったため、街の半分が津波による水害にあい、人口の4%の人が亡くなりました。津波被害を被った地域の中でも被害が甚大だったにもかかわらず、メディアで取り上げられることがほとんどなかった状況を憂い、インターネットの力を使って少しでも世間に伝えたいという思いから、山元町とのかかわりを持つようになったとのことでした。しかし、予想外のところから現在の被災写真のデジタル化・持ち主への返却といった活動につながっていきます。

溝口さんは当初、インターネットの力を使って山元町の現状を発信する目的で、現地にパソコンを設置して地元の人達に利用してもらおうと試みました。そうすると、お年寄りにはパソコンを十分に活用してもらうのが難しく、逆に子供達はゲームに夢中になってしまうなどといった課題がありました。そうした課題を解決すべく、パソコンでやってもらいたいことについて現地の人達に直接ヒアリングを重ねて行く過程で、「津波被害にあってしまった写真のデジタル化」という要望が数多くあがることに溝口さんは気付きます。津波被害にあった写真は著しい劣化により、表面が溶けて思い出が消えて行ってしまいます。そうなってしまう前に洗浄したり、デジタル化することで思い出の写真がなくなってしまわないように保存することが求められていました。それがきっかけで「思い出サルベージ」は設立されました。

そうした経緯で設立された「思い出サルベージ」ですが、デジタル化した写真を持ち主に返して行く中で、被災された方達にとって思い出の写真というものが切実なニーズであったことに気付かされたそうです。

津波被害が大きかった山元町では時間の経過にしたがって、被災された人達の会話が「生きていて良かったね」から「ご遺体が戻ってきて良かったね」に変わり、そして「写真が戻ってきて良かったね」に変わっていったそうです。津波によって奪われてしまったものがあまりにも大きいことを痛感するなかで、溝口さんは写真を返した男性から「これでようやく妻の遺影が作れる」といった言葉をかけていただいたこともあったそうです。

津波に家を流されてしまった方も多く、自分がどう生きてきたかといったアイデンティティが失われ

注5　http://jsis-bjk.cocolog-nifty.com/blog/

てしまった方々にとっては、戻ってきた写真を通して、そのアイデンティティを見つけることができたようでした。そのような光景を見るうちに、写真が人々の心の拠りどころであることがわかったとのことです。ITの力を使って持ち主に写真を返していく中で、見つからなかったご家族の写真を顔認識技術が見つけたり、自分の両親と間違って認識されてしまう過程で家族とのつながりを再認識するなど、テクノロジが1つの写真の意味合いにも広がりをもたらしてくれたことが数多くあったようでした。このように当初予想していたところから被災地の予想外の出来事の連続に対応していく中で、ITの活用できる領域、やり方に応じて見出せるITの可能性に気付いたとのことでした。

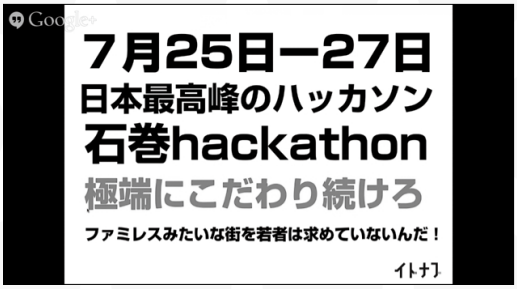
変化の連続の中で現地で求められていることを見つけ、それを拡げていく過程の中で、ITが持つ力を実感する瞬間はあります。ITの力ではできないこと、ITだからこそできることを知ることができる良い発表でした。

大人の背中を見て中高生が自ら育ち始めた石巻

宮城県からの最後の発表は、石巻で活動するイトナブ[注6]の古山隆幸さんからの発表となりました。冒頭、2014年3月11日の震災の時刻に石巻港に鳴り響く哀悼の警笛と共に、黙祷を捧げている古山さんの姿を映し出した動画から発表が始まりました。あの日を境に夢や希望、家族や友達など多くの想いのつぼみが失われてしまいましたが、その代わりに得た多くの人との出会い、そのつながりを今は大切に、そして太くしていこうと思いを新たにしたとのことでした。

イトナブは震災からの復興を“震災以前の元の石巻に戻す”ものにはしたくないという古山さんの思いから、10年後の2021年までに石巻から1,000人のIT技術者を輩出することを目標に活動しています。当初、イトナブとは「IT×学び×イノベーション×営む」の造語でしたが、参加する子供達が自発的にITを学ぶことを楽しみ、遊ぶかのようにしている様子から、最近では「IT×学び×イノベーション×営む×遊ぶ」の造語に変化したそうです。

◆図2　第3回 石巻hackathonの告知



イトナブの方針はその「学び方」に表れています。教育のカリキュラムや教科書、パソコンなどの機器を用意するだけではダメで、いかに子供達に自発的に学ぶ姿勢を持たせるか、という点をとくに意識しています。その姿勢を持ってもらうために、どのようなことを心がけているかという秘訣を発表してもらいました。それは、プロスポーツ選手に憧れを持つように、大人のIT技術者の背中に憧れを持ち、自ずとそうした大人達を目指して学んでいく。そのような環境を用意することを常に意識しているということでした。そのために古山さんは東京やそのほかの地方と石巻を駆け回り、多くのIT技術者達がイトナブで学ぶ子供達と関係を持てるように腐心されています。

また、大人の背中を見せて子供達を触発するだけではなく、イトナブにかかわる大人達もそうした子供達に触発されていくことが、過去に開催された2回の石巻hackathonから見えてきているようです。

そのような良いサイクルが回りはじめ、いま石巻ではIT技術を身につけ羽ばたきはじめた子供達がたくさんいます。そして、今夏も石巻Hackathonは開催されるとのことです（**図2**）。この記事をご覧の皆さまも新しい若い芽の息吹を感じ、大人の背中を見せに、そしてそこから刺激を受けて子供達と共に成長するために、2014年7月25日から27日の予定は空けておき、第3回 石巻hackathonに参加しましょう！ SD

注6　http://itnav.jp/

温故知新
IT むかしばなし

第34回

コモンバス

たけおかしょうぞう　TAKEOKA Shouzou

バックプレーン

現在のPCは、「マザーボード」という基板に、CPUチップやメモリ、I/Oデバイスが集約されています。ですが、昔のコンピュータは、1台のCPU(大型機の中央処理装置)であっても、複数の基板から成り立っていました。そして、それらはラックに収められ、ラックの背面部には、「バックプレーン」という、部品基板間の配線だけを載せた大きな基板などがありまし た。DEC 社(現 HP 社)のPDP-11より前は、バックプレーン上の配線は、互いに結ばれる基板の組み合わせごとに配線が異なり、基板を挿入する位置は決まっていました。

PDP-11は、メジャーな商用機として全面的に「コモンバス」方式を採用し、そのバックプレーンは、どのコネクタにも同じ信号線が同じように配置されました。PDP-11、VAX-11が採用したバスは、“Unibus”です。Unibusはコネクタ上の信号とそのタイミングが明らかにされ、いわゆるサードパーティもUnibus対応基板を製作でき、PDP-11が使用された計測や制御の分野でも各種I/Oボードが作られました。

8bit時代

1976年ごろには、8bitマイコンを汎用コンピュータとして使おうとしたものがあり、早期にコモンバス方式のバックプレーンを持つ筐体がいくつも開発されました。8080(Z80)系は、Altair 8800、IMSAI 8080 が「S-100バス」というものを採用し、6800系では、SWTPC 6800の「SS-50バス」が有名でした。

Intel 8080 CPUは、レジスタ数が多く、アセンブラ・ニーモニックもPDP-11に似せてあり、また、PDP-11と同じ「非同期バス」方式を採用していたので、PDP-11を好きな人ならば、S-100バスのようなコモンバス方式を容易に受け入れられたと思います。

S-100バスは、そもそもAltair 8800を作ったMITS社が考案したのですが、デファクトスタンダードとなり、後にはIEEE696-1983として規格化されました。S-100バスのために、CPUボード(CPUチップとバスドライバしか載っていない)、メモリボードを始めとして、さまざまなI/Oボードなどが作られ、市場を形成しました。日本でも、S-100バスボードを設計し販売していた会社は多く、関西でも「大阪ICM」とか、京都の「TAC」などが知られていました。

このように広範に使われたS-100バスですが、技術的にみると「どうしてこうなってるの???」という面が多く、コネクタのピン配置は無秩序で、電気的にもでたらめで、一般的に、「最初に基板を作るときの都合で、何も考えずに適当に配置したんじゃないの?」と、批判されました。

CPUメーカーであるIntelは、非常にまともな「Multibus(マルチバス、後にIEEE796となる)」というコモンバス方式のバスを作り、産業品質のボードを提供していました。S-100バスもMultibusも、8080CPUチップ(8228チップセット)のタイミングそのままでした。

その後、かなり経ってから、ザイログZ80のタイミングがそのままコモンバスになった「STDバス」もZ80好きには有名で、対応した産業用ボードが

作られました。S-100バスを始め、まともなバスに挿すボードは、すべて、バスをドライブするために、「バスドライバ(バスバッファとも言った)」(電気的に駆動能力が高いデバイス)を備えていました。

PC、パソコン

そのころ、AppleⅡも、マザーボード上にI/Oスロット(カードの挿さるコネクタ)を持っていました。このコネクタはほぼコモンバスだったのですが、一部にアドレスデコード済みの信号があり、コネクタごとに個別に異なる信号が供給されていました。AppleⅡのI/Oスロットは6502 CPUのタイミングです。

IBM-PCは、AppleⅡのコンセプトをほぼそのまま真似していたので、IBM-PCのI/Oスロットも同様でした。ちなみに、IBM-PCのI/Oスロット規格は、後に16bit化され「ISAバス」となりますが、それも8080由来の8086のタイミングがそのまま出たものです。

16bit時代

IntelのMultibusは、16bit時代(8086)もそのまま使用できるようになっていました。16bitの8086で、8080時代のMultibusボードがそのまま使用できるように、奇数番地(Little Endianの16bitの上位8bit)を1バイトアクセスするときは、そのデータをバスの下位8bitに持ってこなければならない、という恐ろしいバスプロトコルを定めてくれました。8086 CPUチップは、そういう仕様だったのでCPUボードは良いのですが、I/Oボードはすべてがデータバスの上位8bit(しかもIn/Outの双方向とも)を下位に持ってくるためのマルチプレクサが必要で、なかなかめんどうくさいものでした。しかし、8bit時代のI/Oをボードを使用できたり、8080や8088(8086の外部バス8bit版)のCPUボードもMultibusIIで使用できたので、低速CPU時代には重宝され、長く使用されました。

Motorolaは1980年ごろ、68000のリリースのころに、「VERSA bus」を出します。VERSA busは後に「VME bus」と名前を変え、ANSI/IEEE 1014-1987となります。VMEバスは、680x0シリーズとともに長く使われています。

32bit初期

Intelは32bit対応した「Multibus II(後にIEEE-1296)」を定めましたが、68020、80486やPowerPCなど、CPUが高速になり、これまでのコモンバスというものはなくなっていきます。メモリはメモリ専用の信号線に接続され、比較的低速なI/Oが、I/Oの共通バスに接続されます。

AppleのMacintoshⅡシリーズは、I/O共通バスに「NuBus(後にIEEE1196)」を採用しました。NuBusはMITのLispマシンあたりで使用され始め、商用Lispマシンである LMI社の"Lambda"という機械などに採用されていました。筆者は、Lispマシンが好きだったので、NuBusを知っていましたが、MacIIがそれを採用したときに「ええっ」と驚きました。その後、ビデオカードには、データの転送バンド幅が必要なので、専用にVLバス、AGPなどができました。

また、I/O共通バスであるISAの後釜を争って、MCA、VESA、EISAがしのぎを削りました。しかし、32bitの後期には、「PCIバス」が残りました。

Sun Sparcのワークステーションは筐体内のI/Oバスとして「SBus」を作り、「MBus」がCPUのコモンバス的なものを使用していました。

まとめ

現在は、一般的にはコモンバスは使用されなくなり、I/Oバスは「PCI Expressバス」が一般的であり、世代を順調に進めています。

ちなみに、PDP-11時代からPCIまでの上記の挙げたバスは、すべてデータはパラレルでクロックにあたる信号が別に存在していました。

PCI Expressは、現代的な高速性を実現するためデータはシリアルであり、クロック信号はデータと同じ線に重畳されています。信号が高速になり過ぎ、クロックとデータが別々の線では、それらのずれが補正できなくなっているからです。SD

Service

東陽テクニカとパステル・ネットワークス、IT機器ベンチマークサービス「@benchmark」提供開始

東陽テクニカ㈱とパステル・ネットワークス㈱は4月15日から共同事業として、会員制のIT機器ベンチマークテストサービス「@benchmark」を提供開始した。

同サービスは、個人ユーザやSIサービス会社などの、IT機器の的確な選定、効果的な導入を支援することを目的としたものである。登録会員の依頼に応じた実環境に近い設定で、検証を無料で実施、テスト結果を提供する。テスト結果は、全会員が無料で閲覧・再利用できるように、依頼主情報を非公開の形にしてWeb上で公開される。

その他、テスト結果の活用方法、ノウハウに関するコラムも公開され、会員同士が自由に情報交換できるコミュニティの場も提供される。

検証は、東陽テクニカ㈱の「電子技術センター」で行われ、約7～10日間で結果が得られる。会員登録料は月額50,000円（税別／最低利用期間：6ヵ月）または、年額550,000円（税別）となっており、初年度100会員・団体の利用を見込んでいるとのこと。

CONTACT
東陽テクニカ㈱
URL http://www.toyo.co.jp/
パステル・ネットワークス㈱
URL http://www.pastelnetworks.com/

Software

ビーブレイクシステムズ、ERPパッケージ「MA-EYES」に在庫管理機能を追加

ビーブレイクシステムズ㈱は、ERPパッケージ「MA-EYES」に在庫管理機能を追加し、4月10日から販売を開始した。

MA-EYESは、IT会社や広告会社などのサービス業系企業向けのERPパッケージである。販売管理、購買・経費管理作業や実績管理、債券債務管理、財務会計、分析・レポート、ワークフローなどの基幹業務を統合して管理できる。

今回のリリースでは、商品の登録や在庫照会、見積提出時の在庫の仮押さえや、受注時の引当・出庫、商品の仕入発注や検収・入庫、月末の棚卸業務などの、在庫管理に関する操作を、プロジェクト単位で行えるようになった。

ユーザの環境にインストールして利用する「一括導入版」は個別見積もりとなる。「SaaS版」は初期導入支援費用（要件定義費用＋開発導入費用）に加え、ユーザ数に応じたモジュール毎の利用料金が月額で発生する。たとえば、今回の新機能「在庫管理」モジュールは、1人あたりの月額利用料金が5,000円となっている。

CONTACT
ビーブレイクシステムズ㈱
URL http://www.bbreak.co.jp

Software

トレンドマイクロ、「ウイルスバスターモバイル」がiPhone/iPadに対応

トレンドマイクロ㈱は4月17日、スマートフォン／タブレット端末向けセキュリティソフトである「ウイルスバスターモバイル」の最新版を店頭で販売開始した（ダウンロード版は4月10日発売）。今回のリリースでは、Android OS、Kindle Fireシリーズに加え、iPhone/iPad（iOS）にも対応した。

このソフトウェアは、不正サイトや不正アプリの対策、盗難／紛失時の端末探索や、クラウドへのデータバックアップを主な機能として持つ（OS／デバイスによって提供される機能は異なる）。スマートフォンが普及し、モバイルユーザを標的にするサイバー攻撃が急増する中、8月末までに1,000万ユーザを目指す。

▼製品ラインナップと価格

製品ラインナップ		価格（税込）
POSA 版	1 年版	オープン価格
	2 年版	オープン価格
ダウンロード版	1 年版	3,065 円
	2 年版	5,637 円

CONTACT
トレンドマイクロ㈱
URL http://www.trendmicro.co.jp/jp/

Report

日本マイクロソフト、「Windows XP」のサポートが終了

日本マイクロソフト㈱は4月9日、Windows XPのサポート終了に関する記者説明会を行った。同社の最高技術責任者である加治佐俊一氏が記者の前に立った。

2001年10月の発売以来、約12年半のサポートが続けられてきたWindows XPであるが、2014年4月9日をもってセキュリティ更新プログラムの提供が終了となった。

▲日本マイクロソフト㈱ 業務執行役員 最高技術責任者 加治佐俊一氏

いまだ一定の利用者がいる当OSのサポート終了に対して記者からは、「利用者数の低減目標はあるか」「新たに見つかった脆弱性はどうするのか」などの質問が挙がった。これに対し加治佐氏は、「目標はもちろんゼロであるが、仮想マシンとしての利用などでまだまだ残っていくと思う。脆弱性については、更新プログラムは配布しないにしても、対策は呼びかけ続ける。だが、よりセキュアな最新OSへの移行を進めてほしい」と語った。

また説明会では、やむを得ず今後もWindows XPを利用するユーザに対して、インターネットからの切断、USBメモリなどの利用停止といった注意が喚起され、移行を検討しているユーザに対しては、データ移行ツールの無料配布と相談窓口が紹介された。

- **XP専用ファイナルパソコンデータ引っ越しeXPress（7月31日まで）**
  URL：http://www.microsoft.com/windows/ja-jp/xp/transfer-your-data.aspx
- **個人向け相談窓口（5月31日まで）**
  フリーダイヤル：0120-256-790
- **中小企業向け相談窓口（6月30日まで）**
  フリーダイヤル：0120-023-999

CONTACT 日本マイクロソフト株
URL http://www.microsoft.com/ja-jp/

Software

サイボウズ、「サイボウズOffice 10.1」提供開始

サイボウズ㈱ は、クラウド版およびパッケージ版の「サイボウズ Office」の最新バージョン「サイボウズ Office 10.1」を4月14日に提供開始した。

サイボウズOfficeは、日本企業向けの多機能グループウェアであり、2013年の中小企業シェアではトップを誇る（ノークリサーチ社）。業務に合わせたアプリケーションをユーザが自由に作成できる「カスタムアプリ」機能が特徴である。今回のアップデートで追加されたおもな機能は次のとおりである。

- **スケジュールの簡易登録機能、ファイル添付機能**
  トップページのスケジュールパーツなどから、アイコンをクリックして、画面遷移なしにスケジュールを登録できる。また、予定に対してファイルを複数添付できるようになった。
- **スケジュールの「iCalendar形式」書き出し機能（クラウド版のみの提供）**
  Google Calendarなど、ユーザ個人で利用しているカレンダーソフト／サービスに「サイボウズOffice」のスケジュールを重ねて表示できる。
- **カスタムアプリ機能における宛先指定機能**
  カスタムアプリ機能で宛先を指定してコメントできるようになった。必要なときに必要な人にだけ通知を送ることができる。また、インクリメンタルサーチにより、氏名を入力するだけでユーザ候補が表示される。

価格は、クラウド版は1ユーザあたり月額500円から、パッケージ版は10ユーザあたり月額63,800円から。クラウド版、パッケージ版それぞれに対して、「スタンダードコース」と、カスタムアプリ機能が付与された「プレミアムコース」が用意されている。

なお同社は、4月13日から9月30日までの期間限定で、クラウド版への移行、またはパッケージ版のバージョンアップを申し込んだユーザに対して、料金が約15% OFFになる「4915（至急行こう）キャンペーン」を実施している。

CONTACT サイボウズ㈱
URL http://cybozu.co.jp/

Service

シーズ、AWSマネージドサービス「ウィズクラウド」を提供開始

シーズ㈱は、Amazon Web Servicesをはじめとするクラウドサービスを利用してサーバ運用を行う新サービス「ウィズクラウド」の提供を開始した。レンタルサーバを使うような感覚でクラウドサービスを利用できることが特徴となっている。

サーバ性能の選定やクラウド上でのネットワーク設定など、ユーザ側が行わなければならないさまざまな作業を同社が行うことで、スムーズなクラウドサービスの活用が可能となる。また、従量課金制による利用料金の高騰を防ぐため、利用料金を定額としており、データ転送量の増減に伴う影響を抑えている。安定したクラウドサービスの運用を実現するための監視・保守サービスや、専任のエンジニアによる技術サポートも提供しており、障害発生時の対応のほか、サーバ性能の向上、複数台構成などの相談も受ける用意があるとのことだ。

クラウドサービスのより安定的な稼働、堅牢なセキュリティ対策を希望する利用者に対しては、クラウドサービス上で作成されたサーバを専任のエンジニアが、OSレベルで保守・運用を行うオプションサービス「マネージドプラス」が用意されている。

CONTACT シーズ㈱
URL http://www.seeds.ne.jp/

Hardware

ウエスタンデジタルジャパン、Thunderbolt対応のポータブルHDD「My Passport Pro」を発表

ウエスタンデジタルジャパン㈱は、Thunderbolt対応、HDD2台を搭載したポータブルストレージ「My Passport Pro」を4月23日に発売した。

製品本体にあらかじめ接続されているThunderboltケーブルから直接給電できるので、ACアダプタを必要とせず、最大233MB/s（2TBモデルでの実測値）のデータ移送速度によって快適なデータ編集やバックアップが可能となっている。また、設定変更が可能なRAIDモードを搭載しており、RAID0とRAID1の切り替えができる。アルマイト仕上げのアルミニウム製筐体は高い耐久性を持ち、大量のデジタルコンテンツ扱うようなユーザが、出先でコンテンツを編集・保存する際に重宝することだろう。

メーカー希望小売価格は、2TBモデルが43,800円（税別）、4TBモデルが58,800円（税別）となっている（ともに製品保証は3年）。

▲ My Passport Pro

CONTACT ウエスタンデジタルジャパン㈱
URL http://www.wdc.com/jp/

Hardware

日本IBM、次世代スケールアウト型サーバ「IBM Power System Sクラス」発表

日本IBM㈱は4月24日、IBMの次世代プロセッサ「POWER8」を搭載し、ビックデータ時代に合わせて開発されたスケールアウト型サーバ「IBM Power Systems Sクラス」を発表した。オープンな開発コミュニティ「OpenPOWER Foundation」の活動の成果を採用した業界初のサーバとなっている。搭載されているPOWER8は、メモリバンド幅やI/Oバンド幅を従来比2倍に強化し、ビッグデータの高速処理を可能にしている。

同社社長マーティン・イェッター氏は同製品について「ISP、MSPといった顧客をメインとし、ハイパフォーマンス、管理のしやすさを売りに、市場に挑む」という。

同製品は、Linux専用の「S812L」、「S822L」、Linuxに加えAIXとIBMiの複数OSに対応した「S814」、「S822」、「S824」の5モデルから構成されており、6月10日からの出荷を予定している。


▲ 日本IBM社長
マーティン・イェッター氏

CONTACT 日本IBM㈱
URL http://www.ibm.com/jp/ja/

Software

エンバカデロ・テクノロジーズ、C++でのAndroidアプリ開発が新たにサポート クロス開発環境の最新版「RAD Studio XE6」を発売

エンバカデロ・テクノロジーズは、ソフトウェア開発環境のスイート製品「RAD Studio」の最新版「RAD Studio XE6」の販売を4月16日より開始した。

今回のバージョンアップの目玉は、これまでDelphiでのみ実現されていたAndroidアプリのビジュアル開発が、C++でも行えるようになったこと。これによりDelphiとC++でクロス開発が可能になったプラットフォームは、Windows/Mac/iOS/Androidの4つとなった。同社の掲げるワンソース・マルチプラットフォームによる生産性向上が、一層推し進められる製品となっている。また、Windows XPからの移行対策として、これまでXP向けに作っていたアプリケーションをWindows 8.1のルック&フィールにアップデートする機能を用意した。

価格は、RAD Studio XE6 Professionalが208,000円、同Enterpriseが344,000円（いずれも新規購入、税別）。RAD Studio XE6に含まれるDelphi XE6、C++ Builder XE6を単体製品としても購入できる。なお、30日間無料で全機能が試せるトライアル版が同社のWebサイトからダウンロード可能。

CONTACT エンバカデロ・テクノロジーズ
URL http://www.embarcadero.com/jp

Hardware

TFFフルーク社、ネットワークテスター「OneTouch AT3 ネットワーク・アシスタント」を発売

㈱TFFフルーク社は4月22日、トラブルシューティング用ネットワーク・テスター「OneTouch AT3 ネットワーク・アシスタント」を提供開始した。

「OneTouch AT」シリーズは、有線、無線Wifi、光ファイバーでのケーブル試験など、すべてを1台でカバーできるオールインワンタイプのトラブルシューティングツールである。

新製品AT3の主な新機能として、スマートフォンでの遠隔トラブルチェックや、タブレットを使ったトラブルシューティングが行える「クラウド機能」、長期収集されたトレンドデータを遠隔からブラウザで閲覧できる「トレンド解析機能」、起動させるだけで自動的にリモート試験を稼働させ、解析結果を提供しつつクラウドへアップロードする「自動化テスト機能」が挙げられる。

価格は、780,000円（税別）からとなっている。

▲ One Touch AT3 ネットワーク・アシスタント

CONTACT ㈱TFFフルーク社
URL http://www.flukejp.com/

Software

マカフィー、次世代ファイアウォール「McAfee Next Generation Firewall」提供開始

McAfee Inc.の日本法人、マカフィー㈱は5月1日より、次世代ファイアウォール製品「McAfee Next Generation Firewall」を提供開始した。

同社では、2014年には従来のセキュリティ対策を回避する潜伏型攻撃、とくに高度な検知回避技法を有したものが増加すると予測し、検知回避防止機能を備えた同製品を開発した。同製品の主な特長としては次の3つがある。

- モジュール化された搭載機能を拡張性に優れた方法で提供することで、導入プロセスを簡素化できる
- 最大120Gbpsのファイアウォールスループットで動作する最大16ノードのアクティブなクラスタリングを提供できる
- 直観的に操作可能なコンソールから、デバイスを効率的に管理・監視できる

最小構成価格は720,000円～（税別）（管理サーバを含まない参考価格）となっている。

CONTACT マカフィー㈱
URL http://www.mcafee.com/japan/

Letters from Readers

『Software Design』PDF版、始めました!

すでにご存じの方もいらっしゃると思いますが、5月号から『Software Design』のPDF版の発売が始まっています。Twitterなどではすごい反響があり、みなさんのPDF提供の要望がいかに強かったか、あらためて知りました。「紙のほうが読みやすいけど、収納面を考えるとPDFのほうが便利だし、どちらがいいの?」とお悩みの方、両方購入してもいいんですよ!(そういう問題じゃない?)

2014年4月号について、たくさんのお便りをありがとうございました!

第1特集　なぜMVCモデルは誤解されるのか?

Webアプリケーションを開発する設計技法として広く浸透しているMVCモデル。しかしながら、本当に正しく理解できているでしょうか?　本特集ではJava、JavaScript、PHPのそれぞれの言語ごとにMVCモデルの実現方法を紹介しました。

Javaの記事でブラウザの役目はViewではないということに驚愕した。今まで信じていたことが、違っていたという衝撃。これからは認識を改めます。

東京都／tomato360さん

MVCだけで特集を組めるとは驚きです。ギュッとまとまっていて、リファレンスとして活用しやすく、重宝しそうです。

山梨県／shoz-fさん

MVCは今までJava開発で何度も使ってきましたが、開発者間で共通認識ができていないということを初めて知りました。新しいメンバーと開発を行うときには、単語だけでなく、言葉の意味についても認識のすり合せが大事ですね。

広島県／はっせんさん

MVCモデルを効率よく実現するためにフレームワークを利用することが多いです。本特集でもいくつかのフレームワークを紹介しましたが、そのせいか、「現場でもフレームワーク導入を検討したい」という声も多く寄せられました。

第2特集　ロードバランサの教科書

複数のサーバから構築されるWebシステムにおいて、負荷分散のためのロードバランサは必須の機能です。最新のハードウェア製品、今流行のソフトウェアベース／クラウドベースのロードバランサなど、その機能や導入方法について解説ました。

ソフトウェアベースのロードバランサのところは、家庭内のネットワークでいろいろ試して遊ぶのに参考になりそうでした。

石川県／Keiさん

ロードバランサに限らず、何でもソフトウェアで実現する時代ですね。

東京都／山下さん

今は組込み系の仕事をしているので、ネットワーク系の知識が得られてうれしい。

長崎県／JavaSplictさん

ソフトウェア、もしくはクラウドのロードバランサに興味を持った読者が多かったようです。IaaSなどで手軽にインフラ環境を構築できるようになったため、ロードバランサもクラウドで用意するというのは必然なのかもしれませんね。

特別企画　今すぐ知りたいSIMのしくみ

SIMという言葉は聞いたことはあっても、その目的やしくみはご存じないのではないでしょうか?　本記事では意外と知られていないSIMの歴史、機能、技術的なしくみについて解説しました。

SIMのお話はおもしろかったです。ただ、国内に限って言えば、SIMフリーの端末の種類が少なく、また、海外の展示会などで発表された魅力的な端末も、国内では国内法による規制(技適など)があり、簡単には使用できないのが辛いところです。海外情報を目にするたびに、「この端末ほしいな〜。でも国内では業社が扱わないだろうな〜」と思うことも度々です。個人で技適の適応を受けることは、難しいですからね。

山口県／A758さん

SuicaやおサイフケータイのICチップもSIMカードのように着脱できれば機種変更も簡単になると思います。SIMと一体化してほしいです。

大阪府／澤下さん

特集に匹敵するほど、反響が多かった記事でした。スマートフォンや携帯電話に備わっていて身近な存在なのに、そのしくみについては、あまり語られることがなかったからでしょう。この記事を読んだあとでは、端末を選ぶときの視点やSIMの使い方も変わってきますね。

短期集中連載　さらに踏み込む、Mac OS Xと仮想デスクトップ

Mac OS Xの仮想デスクトップ記事の第2回。X Window Systemを使ってウィンドウを共有する方法を取り上げました。

今後も続けてほしいです。X Window Systemの日本語入力なども取り扱ってほしかったです。

大阪府／山下さん

最近、Macに乗り換えたばかりなので、この手の連載は大歓迎。

千葉県／若山さん

X Windows Systemとの連携といった話が出てくると、UNIXベースであるMacとの相性の良さが出てきますね。仮想環境で複数OSを使っている方も多いでしょうから、ぜひ試してみてください。

フリートーク

毎号楽しみにしています。増税前に年間定期購読に切り替えました。

大阪府／きよさん

ありがとうございます！　年間定期購読だと割引になりますから、毎月購入するのと比べてずいぶんお得になったのではないでしょうか？　この機会に、ほか読者のみなさんも定期購読をご検討ください！　もちろん電子版（PDF版、Fujisan版）でもできますよ。

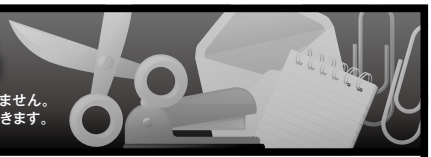
ついにXPのサポートが切れました。これから先、残ったXPのPC本体をどうするか悩ましいところですよ。

神奈川県／眞　泰志さん

編集部では、Windows PCも使っていますが、さすがにWindows XPはないと思っていたら、アシスタントさんが使っていたPCがXPでした。盲点でした。ちなみに担当の自宅にもXP時代に購入したPCがありますが、こちらはCentOSを入れて、すっかり実験機になっています。

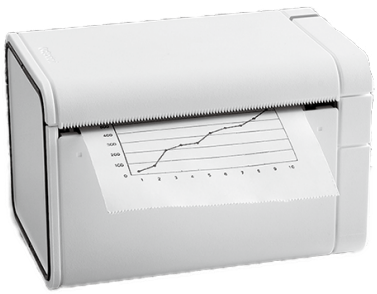
エンジニアの能率を高める一品

仕事の能率を高める道具は、ソフトウェアやスマートフォンのようなデジタルなものだけではありません。
このコーナーではエンジニアの能率アップ心をくすぐるアナログな文具、雑貨、小物を紹介していきます。



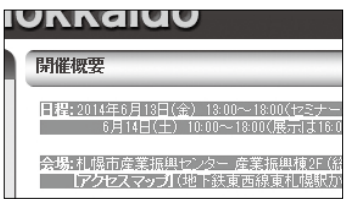
クリップ専用プリンター「ココドリ」

14,000円（税別）／㈱キングジム　http://www.kingjim.co.jp/



▶図2 選択したテキストだけが印刷される。印刷はモノクロ。

ココドリはPC画面の必要な部分だけを印刷できるプリンタです。画面の指定した範囲を画像として印刷する「キャプチャモード」と選択したテキストだけを印刷する「テキストモード」の2つの機能があります。試用して便利だったのは、意外にもテキストモードのほう。たとえば、ブラウザでイベントのWebサイトを見ているとします。その中の日程、会場などの情報だけをメモしたい場合は、その部分のテキストを選択して（**図1**）、Ctrl＋C→Pause と押下するだけでプレビューが表示され、印刷ボタンをクリックすれば印刷できます（**図2**）。小さなプリンタですので、デスク上に置いて手軽に選択→印刷とできるのがいいですね。ふせんタイプのロール紙を使えばシールのように貼れます。電子メールの署名部分だけを印刷すれば、そのまま郵便の宛名シールとして使えそうです。（読者プレゼントあります。P.20参照）



▲図1 Webサイトで必要な情報を選択する。

4月号のプレゼント当選者は、次の皆さまです

①EVOUNI Leather Arc Cover iPad mini Retina L37 ..... 福島県　國島京子様
②SD・microSD カードリーダ・ライタ SCR-SD03/BK..... 富山県　小林久壽雄様

★その他のプレゼントの当選者の発表は、発送をもって代えさせていただきます。ご了承ください。

次号予告

Software Design
July 2014

2014年7月号
定価（本体1,220円+税）
176ページ

6月18日発売

［第1特集］多機能・高速処理・高負荷対策

そろそろNginx移行を考えているあなたへ

Webサーバを題材に運用技術の基礎を押さえる

ApacheからNginxへ移行するメリットとは／Nginxを導入する理由／移行前チェックポイントの洗い出し／Nginxのインストール+コンフィグ／引っ越し本番!／移行後の「あるある」エラー・トラブルシューティング／クラウドへNginxの利用

［第2特集］知っているようで知らない

DHCPサーバの教科書

DHCP（Dynamic Host Configuration Protocol）サーバはネットワークの縁の下の力持ち。当たり前のようにIPが使える現在、DHCPの存在などいまさらと思われるかもしれませんが、案外知られていないしくみの解説から、応用的な事例まで一気に解説!

［一般記事］OpenSSLの脆弱性“Heartbleed”解説

※特集・記事内容は、予告なく変更される場合があります。あらかじめご容赦ください。

休載のお知らせ

「Androidエンジニアからの招待状」（第48回）は都合によりお休みいたします。

SD Staff Room

●当編集部に新人が配属されました。なんと22歳! 会社全体の平均年齢も大幅に変わりました。彼らの世代が楽しくなるようなコンテンツを作りたい。そんな気持ちで作った5月号は、おかげさまで在庫がはけてしまいました。技術を理解し知ることで、視野と可能性が広がる、そういう雑誌にします!（本）

●陽気が良くなってきて、休日は自転車で買い物に行くことが多くなった。自宅から半径10km以内の移動だが、普段のデスクワークで身体がなまっているので、少しずつ距離を延ばしたい。×虫ペダルのように、日に250kmは無理にしても、最終的には日に40kmくらいは走れるようになりたいなぁ。（幕）

●春。我が家にも新人がやってきました。前任者は10年以上勤務の働き者でしたが、消費税アップ前の駆け込みでついに選手交代。入れたものの重さを量って洗剤の目安にするとか、お風呂の残り湯を吸い上げてくれたりするエコっぽさが今風。容量も増えて洗濯物の山がなくなりました! 若いってイイネ（キ）

●読者アンケートの回答に「プレゼントの当選者を全員発表してほしい」という声があった。今は掲載枠が狭く2人しか載せていませんが、Letters From Readersを見直す機会があれば、全員掲載を検討したい。ただ、当選結果を細かく分析されると、当選しやすい品物などがわかってしまいそう。（よし）

●先日、初めて神保町に行きました。スポーツ用品店、ボードゲーム専門店など、趣味の町という感じ。なにより本屋だらけ! 世の中には本当に多くの本があるんだなと、再認識させられました。その1冊1冊に読者がいると思うと、感慨深いですね。神保町で会いましょう! そんな感じの週末でした。（な）

●我が家の近所に時間無制限でワイン飲み放題のお店があります。はじめて来店したとき閉店までいたからか、まだ2回しか行っていないのに店長さんに覚えられていてお得意様気分。開店して半年ばかりの穴場で、お財布にもやさしいので繁盛してほしいが、繁盛しすぎると気軽に行けなくなるしと複雑です。（ま）

ご案内
編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［FAX］
03-3513-6173
※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

［E-mail］
sd@gihyo.co.jp

Software Design
2014年6月号

発行日
2014年6月18日

●発行人
片岡 巌

●編集人
池本公平

●編集
金田富士男
菊池 猛
吉岡高弘
中田瑛人

●編集アシスタント
松本涼子

●広告
中島亮太
北川香織

●発行所
㈱技術評論社
編集部
TEL：03-3513-6170
販売促進部
TEL：03-3513-6150
広告企画部
TEL：03-3513-6165

●印刷
図書印刷㈱

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。