Special Feature 1
→Linuxネットワーク入門
Special Feature 2
→ドッグフーディング
Special Feature 3
→ブロックチェーン

Software Design

［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

May 2017

5

2017年5月18日発行
毎月1回18日発行
通巻385号
（発刊319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

新卒準備号
第2弾

先輩が教える
得ノウハウ

Special Feature 1

Linux入門

UNIX
ネットワーク
編

ネットワーク技術は
やっぱり難しいと
思っているあなたへ

Special Feature 2

サービス改善につなげる
ドッグフーディング
環境の作り方

Special Feature 3

ブロックチェーン
入門
いますぐわかりたい
Fintechの基本

Extra Feature

DDoS攻撃は防げないのか？
ニフティクラウド mobile backend（第2回）
ラムダ式から人工知能まで、今こそLispを学ぶ理由

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。

目次

新卒準備号 第2弾

Contents

Software Design

May. 2017

5

Special Feature 1 → 第1特集 | Page

目次

Contents

Software Design

May, 2017

Column

Development

OS/Network

[ロゴデザイン]
デザイン集合ゼブラ+坂井 哲也
[表紙デザイン]
藤井 耕志(Re:D Co.)
[表紙写真]
Kadmy / AdobeStock
[イラスト]
*高野 涼香
*フクモトミホ
[本文デザイン]
*安達 恵美子
*石田 昌治(マップス)
*岩井 栄子
*ごぼうデザイン事務所
*近藤 しのぶ
*SeaGrape
*轟木 亜紀子、阿保 裕美、
佐藤 みどり、徳田 久美
(トップスタジオデザイン室)
*伊勢 歩、横山 慎昌(BUCH+)
*藤井 耕志、萩村 美和(Re:D Co.)
*森井 一三

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。

BOOKS INFOMATION

情シス・IT担当者［必携］

システム発注から導入までを成功させる90の鉄則

田村昇平 著

A5判／256ページ
定価（本体2180円+税）

ISBN978-4-7741-8925-3

なぜシステムの発注～導入には失敗がつきまとうのでしょうか？筆者は「失敗の原因はユーザー企業の力量不足」と喝破します。ユーザー企業は、少なからず何らかのシステム導入を経験しているものです。であれば、経験はノウハウとして蓄積されているはずです。しかし、プロジェクトは失敗してしまいます。ノウハウに体系的なまとまりがないからです。
本書にはITコンサルタントという立場だからこそ知りえた筆者のノウハウが詰まっています。

Java本格入門

モダンスタイルによる基礎から
オブジェクト指向・
実用ライブラリまで

B5変形判／448ページ
定価（本体2980円+税）

ISBN978-4-7741-8909-3

谷本心、阪本雄一郎、岡田拓也、秋葉誠、村田賢一郎 著
Acroquest Technology株式会社 監修

誕生から20年を迎え、幅広い分野のプログラミングに欠かせないJavaの基礎から応用までをしっかり解説。Javaの最新仕様（Java 8）に基づく文法から、オブジェクト指向やデザインパターン、そしてビルド、ドキュメンテーション、品質への配慮などまで、現場の開発で避けては通れない話もきちんとおさえました。開発やトラブルシュート経験の豊富なアクロクエストテクノロジーのメンバーが、保守性、堅牢性、性能、開発効率などの観点をふまえて書き下ろした、Java開発者必携の1冊です。

Ruby on Rails 5 アプリケーションプログラミング

山田祥寛 著

ISBN978-4-7741-8883-6

B5変形判／608ページ　定価（本体3600円+税）

Ruby on Railsの定番解説書が、大幅改訂して最新バージョン5に対応！MVCに則ったWebアプリケーションフレームワークの最新版「Ruby on Rails 5」を対象に、Scaffolding機能から、ビュー／モデル／コントローラ開発、ルーティング、テスト、クライアントサイド開発まで、Railsの主要機能を徹底解説しています。クライアント開発で必要となるCoffeeScriptやSCSS、バージョン4以降の新機能であるActive JobやRails APIなどにも対応しているので、最新技術を取り入れたWebアプリケーション開発にも柔軟に対応できます！

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

言語のセオリーを徹底解説！

基礎から応用まで、必須知識を完全網羅！

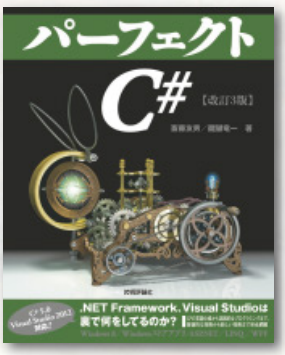
Rサポーターズ　著
B5変形判／672ページ
定価（本体3600円+税）
ISBN978-4-7741-8812-6

Pythonサポーターズ　著
B5変形判／464ページ
定価（本体3200円+税）
ISBN978-4-7741-5539-5

小川雄大、柄沢聡太郎
橋口誠　著
B5変形判／592ページ
定価（本体3600円+税）
ISBN978-4-7741-4437-5

Rubyサポーターズ　著
B5変形判／640ページ
定価（本体3200円+税）
ISBN978-4-7741-5879-2

すがわらまさのり、前島真一
近藤宇智朗、橋立友宏　著
B5変形判／432ページ
定価（本体2880円+税）
ISBN978-4-7741-6516-5



斎藤友男、醍醐竜一　著
B5変形判／608ページ
定価（本体3600円+税）
ISBN978-4-7741-5680-4

井上誠一郎、永井雅人　著
B5変形判／592ページ
定価（本体3200円+税）
ISBN978-4-7741-6685-8

井上誠一郎、槙俊明
上妻宜人、菊田洋一　著
B5変形判／592ページ
定価（本体3200円+税）
ISBN978-4-7741-8316-9

井上誠一郎、土江拓郎
浜辺将太　著
B5変形判／544ページ
定価（本体3200円+税）
ISBN978-4-7741-4813-7

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

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

Software Design plus

Software Design plusシリーズは、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』編集部が自信を持ってお届けする書籍シリーズです。

最新刊！

山本小太郎 著
B5変形判・176ページ
定価 2,480円（本体）+税
ISBN 978-4-7741-8885-0

養成読本編集部 編
B5判・144ページ
定価1,780円（本体）+税
ISBN 978-4-7741-8865-2

養成読本編集部 編
B5判・112ページ
定価2,180円（本体）+税
ISBN 978-4-7741-8894-2

香山哲司、小野寺匠 著
A5判・176ページ
定価 2,480円（本体）+税
ISBN 978-4-7741-8815-7

星野武史 著、花井志生 監修
A5判・256ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-8729-7

小川晃通 著
A5判・272ページ
定価 2,280円（本体）+税
ISBN 978-4-7741-8570-5

中井悦司 著
B5変形判・272ページ
定価 2,980円（本体）+税
ISBN 978-4-7741-8426-5

前橋和弥 著
B5変形判・304ページ
定価 2,680円（本体）+税
ISBN 978-4-7741-8188-2

五味弘 著
B5変形判・272ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-8035-9

神原健一 著
B5変形判・192ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-7749-6

養成読本編集部 編
B5判・192ページ
定価1,980円（本体）+税
ISBN 978-4-7741-8863-8

養成読本編集部 編
B5判・160ページ
定価2,180円（本体）+税
ISBN 978-4-7741-8895-9

養成読本編集部 編
B5判・240ページ
定価 1,980円（本体）+税
ISBN 978-4-7741-8877-5

養成読本編集部 編
B5判・168ページ
定価1,980円（本体）+税
ISBN 978-4-7741-8360-2

改訂3版　Linuxエンジニア養成読本
養成読本編集部 編
定価 2,080円+税　ISBN 978-4-7741-8385-5

改訂3版 サーバ／インフラエンジニア養成読本
養成読本編集部 編
定価 2,080円+税　ISBN 978-4-7741-8034-2

【改訂新版】サーバ構築の実例がわかるSamba［実践］入門
髙橋基信 著
定価 2,680円+税　ISBN 978-4-7741-8000-7

AWSエキスパート養成読本
養成読本編集部 編
定価1,980円+税　ISBN 978-4-7741-7992-6

サーバ／インフラエンジニア養成読本 DevOps編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-7993-3

［増補改訂版］クラウド時代のネットワーク技術 OpenFlow実践入門
高宮安仁、鈴木一哉、松井暢之、村木暢哉、山崎泰宏 著
定価 3,200円+税　ISBN 978-4-7741-7983-4

Unreal Engine&Unityエンジニア養成読本
養成読本編集部 編
定価 2,280円+税　ISBN 978-4-7741-7962-9

ソフトウェアエンジニアのためのITインフラ監視［実践］入門
斎藤祐一郎 著
定価 2,280円+税　ISBN 978-4-7741-7865-3

Unityエキスパート養成読本
養成読本編集部 編
定価 2,480円+税　ISBN 978-4-7741-7858-5

Docker 実践入門
中井悦司 著
定価 2,680円+税　ISBN 978-4-7741-7654-3

データサイエンティスト養成読本 機械学習入門編
養成読本編集部 編
定価 2,280円+税　ISBN 978-4-7741-7631-4

C#エンジニア養成読本
岩永信之、山田祥寛、井上章、伊藤伸裕、熊家賢治、神原淳史 著
定価 1,980円+税　ISBN 978-4-7741-7607-9

Dockerエキスパート養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-7441-9

AWK実践入門
中島雅弘、富永浩之、國信真吾、花川直己 著
定価 2,980円+税　ISBN 978-4-7741-7369-6

シェルプログラミング実用テクニック
上田隆一 著、USP研究所 監修
定価 2,980円+税　ISBN 978-4-7741-7344-3

技術評論社

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。

- 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）

第101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。

Adobe Sensei

Adobe製AI「Adobe Sensei」

「Adobe Sensei」は、米Adobe Systems社によるユーザエクスペリエンスにまつわる問題の解決に特化したソリューションです。機械学習や深層学習といったAI技術を活用した新しいフレームワークおよびインテリジェントサービス群として、2016年11月に発表されました。

Adobe Senseiの学習のベースとなるのは、Adobeが長年にわたって蓄積してきたコンテンツやデータアセット、デザインツール開発のノウハウ、顧客クリック数をはじめとするマーケティングデータなどです。これらの膨大なデータをもとに学習を行い、ユーザエクスペリエンスにおける複雑な課題に対する解決策の提供を目指します。

Adobe Senseiは単体の製品の名称ではなく、Adobeが提供する各種ツールやサービスのバックグラウンドを支える基盤として機能する技術とのことです。現在Adobeでは「Creative Cloud」「Experience Cloud（従来のMarketing Cloudを含む）」「Document Cloud」の3種類のクラウドサービスを展開しています。Adobeの発表によれば、Adobe Senseiはすでにこの3つのクラウドそれぞれで導入されており、それによって実現した機能の一例として次のようなものが挙げられています。

◎ Creative Cloud

- 指定した画像と類似する画像を検索する
- 写真やイラスト中の文字を認識し、似た特徴を持ったフォントを探し出す
- 画像から目、鼻、口といった顔のパーツを認識し、それを自然な形で変化させて表情を変えることなどができる
- 画像に含まれる領域やオブジェクトに対して、そのタイプに基づいたラベル付けを自動で行う

◎ Experience Cloud

- マーケティング投資を最適化するための顧客分析やタッチポイントの分析、データ分析の簡素化などを自動で行う
- キャンペーンの有効性を分析し、マーケティング投資の最適化をサポートする
- 顧客をインテリジェントにセグメント化することで、見込み客の特定の効率を高める
- コンテンツの提供先ターゲットを自動でパーソナライズする

◎ Document Cloud

- デジタルドキュメントのテキストの認識やトピックのモデリング、センチメント分析などを自動化する
- スキャンしたドキュメントの歪（ゆが）み補正や色調補正を自動で行う

AIによる予測をツールとして活用する

上記のようにAdobe Senseiが活用される範囲は極めて多岐にわたりますが、その本質は“ユーザが望んでいる結果・体験を予測して具現化する”ということです。言うまでもなく、これは機械学習・深層学習がもっとも得意とする分野です。

たとえば前述の「顔写真の表情を自然な形で変化させる」という処理は、もともとはクリエイターが繊細な編集技術を駆使して行っていたものでした。これに対してAdobe Senseiは、過去のノウハウや膨大な写真データの分析をもとにして「自然な表情に見える編集結果」を予測し、その理想の形を提案します。

Adobe Senseiが特徴的なのは、このような“予測”を、直接的なサービスではなくクリエイターやマーケッターを手助けするツールの一機能として提供し、その結果をユーザ自身が選択・調整できるようにしている点です。機械学習や深層学習の活用をツールの中に取り込み、より具体的な日々の作業に落とし込んでいくという試みは、ツールベンダーであるAdobeらしいアプローチとも言えます。

さらにAdobe Senseiの場合、分野が異なる3つのクラウドサービスの共通基盤として動作するという点も見逃せません。将来的には、「マーケティング分析から導き出された最適な広告デザインをデザイナに提案する」というような、分野をまたいだ予測によるインテリジェント機能が提供される可能性も十分に考えられるわけです。

Adobeでは今後もAdobe Senseiの性能の向上に努めるとともに、その適用範囲も広げていく予定だそうです。また、サードパーティのベンダーや開発者への提供も進めていくとのことです。SD

Adobe Sensei
http://www.adobe.com/jp/sensei.html

DIGITAL GADGET

vol.221

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

Interaction Awardから読みとる多様性

インタラクション（相互作用）とトリガー（きっかけ）

インタラクションは「相互作用」と訳され、2つ以上のモノがお互いに影響し合うことを指し示します。それぞれの作用にはトリガーと呼ばれるきっかけが存在します。人々が何かを好んで使い始めたり、何かにハマって使い続けたりするには、このトリガー（きっかけ）の先に、行動することで報酬を得たり目的を達成したりなどといった、次につなげる何かが必要です。トリガーには内的要因と外的要因があります。トリガーは何度も繰り返されることで習慣化され、何か面倒なこともわざわざ手間をかけるようになります。

トリガーにはさまざまな種類があり、ガジェットやサービスで広く受け入れられているものには、これらのトリガーが計算づくで組み込まれているか、もしくは製作者の意図なく無意識のうちに組み込まれている傾向があります。

- 人間のさまざまな欲を刺激するもの
- それを使った結果の満足を確約するもの
- 自分で勝手に理屈をつけて、使うことを正当化してしまうもの
- お買い得感、便利感があるもの
- 権威の象徴となるもの
- 自分のモノとなる所有欲を満たすもの
- 何かと関連づけて思い浮かび、忘れないもの
- どこかに所属したいという欲求を満たすもの
- 収集欲を満たすもの
- 限定、特別感があるもの
- 罪悪感を払拭するもの
- 好奇心や、それによって考えるきっかけが得られるもの

今回本稿で紹介するインタラクションアワード（Interaction Award）は、米国IxDA（Interaction Design Association）が主催するアワードです。多数の作品応募の中から優秀なものが各部門ごとに12作品ほど選出され、さらにその中から部門ごとに5作品ほどが最終候補として公開され、一般審査と専門家の審査により部門ごとに賞が決まります。

Interaction Award 2017 公式サイト
http://awards.ixda.org/

各賞は、次の6つのカテゴリに分けられています。

[Connecting：つながり]
人と人や、人とコミュニティ間のコミュニケーションを手助けする

[Disrupting：再構築]
新しい行動、用途、市場を作り出し、既存の製品やサービスを壊して新しい価値を生み出す

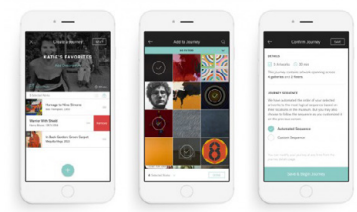
[Empowering：助力]
限界を超え、今までにできなかったような事柄をできるように仕向ける

[Engaging：魅了]
喜びや注目を集め、その事柄に意味をあたえる

[Expressing：表現]
自己表現や、創造性を手助けする

[Optimizing：最適化]
日々の活動を効率化する

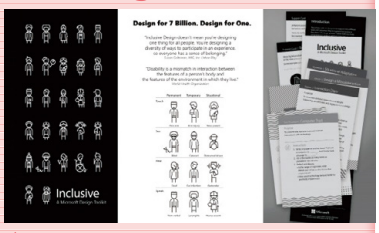
VRを活用した子供向け教育システム「Peer」



スマートフォンの中に展開される、終わらない博物展「Mia Journeys App」

※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います。

Connecting部門



pic.1 Microsoft Inclusive Toolkit

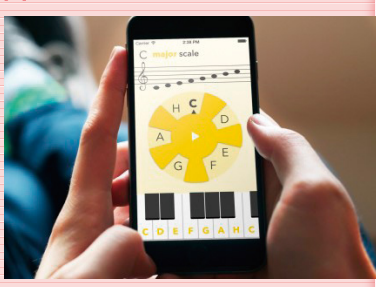
Expressing部門

pic.2 Together Radio

pic.3 Vochlea

Disrupting、Optimizing部門

pic.4 Kinetic



pic.5 Musiclock

今年の傾向と作品の評価

今年の傾向は、世の中の潮流どおり、単なるモノから、コトに移行してきていることが強く感じられます。また、今まで以上により大きな社会的な課題に目を向け、その解決に取り組んでいるものが多く、テクノロジと、インタラクティブ技術で世の中がより良い方向に向かっていることが実感されるアイデアがより高く評価される傾向にありました。

Connecting部門より

Microsoft Inclusive Toolkit
https://www.microsoft.com/en-us/Design/inclusive#toolkit

Microsoft Inclusive Toolkitは、Microsoftが提唱する、さまざまな状態・状況の人を想定して考えるためのツールキット。平易なアイコンとともに、さまざまな事情や状況の人々を網羅的に紹介しており、社会的サービスや、広く多くの人々に使われる公共サービスやアプリを作るときの参考になるツールキットです。たとえば、腕にケガをした人、車いすの人、赤ちゃんを抱えている人など、意識的に気にしないと気付かない、さまざまな環境の差異に気付くことができるようになります。このツールキットの活用で、たとえば、ケガで片手しか使えない人用に何かを考えることで、赤ちゃんを抱えている人や、ほかの条件の人々にも的確なサービスを提供できるようになるのです(**pic.1**)。

Expressing部門より

Together Radio
http://awards.ixda.org/entry/2017/together-radio-an-anonymous-support-group-network/

Together Radioはラジオ放送のDJ、ナレーターと、視聴者が手をつないでいるような感覚を得ることのできる共感ラジオです。安価な組み込みコンピュータであるArduinoで制御された温度の変化するシート、モーター、振動モーターからなり、ラジオから流れる放送を聴きながら、手を入れることで、離れたところ同士でも振動や暖かさがネット経由で転送され、感覚を共有することができます。現在はまだ試作品の段階です(**pic.2**)。

Vochlea
http://www.vochlea.co.uk/

Vochleaは音楽をプロトタイピングするためのデバイスです。体験者の声を素材に、リズムを静止したり、音程を合わせたり、楽器が演奏できない人であっても、イメージした音楽を作り上げることができます。ヒューマンビートボックス(口と声だけでリズムやメロディを奏でる)のデジタル版のような感じです(**pic.3**)。

Disrupting、Optimizing部門より

Kinetic
http://awards.ixda.org/entry/2017/kinetic/

Kineticは、労働者のケガ防止デバイスです。体を使った労働、たとえば倉庫業務や建設土木業など、身体を酷使するような労働業務において、このセンサーデバイスを腰ベルトに装着して働くことで、姿勢や動きなどの検知から、事故の予兆や、疲労度合いの配慮、仕事の分担への配慮ができるようになりました。実際にこのデバイスの装着と、そこからのデータ収集によって、ケガをする度合いが半分に減ったそうです(**pic.4**)。

Musiclock
http://www.mymusiclock.com/

Musiclockは複雑な音楽理論を知らずとも、手軽に音楽を生成することのできるアプリです。複雑な音楽理論や、音楽とはこうでなければいけないという固定概念なしに、自由に一般の人々が音楽を作り上げ、その出来上がった音楽が、ちゃんと従来の音楽理論に正しく当てはめられた聞き応えのある音楽になるのが特徴です。一流のJazzプレーヤでなくとも、アドリブ演奏ができるのです。言葉による説明だけではわかりにくいかもしれませんが、iOSアプリとして無料で提供

されているので、ぜひ試してみてください（pic.5）。

インタラクションの多様性。単なるモノだけではないインタラクション

アーサー・C・クラーク（共著マイクル・P．キュービー＝マクダウエル）のSF小説「トリガー」には、ある領域に入ると、すべての火薬が爆発してしまい、武器が無効化されるという防御兵器が登場してきます。

意図するトリガー、意図しないトリガーがあり、それによって開始したインタラクションも、何に対する操作や行動なのか、単なるモノを操作するだけではない、新しいインタラクションの価値や役目が今回のInteraction Awardから読みとることができます。

たぶん世の中は、私たちが思っている以上に多様性を持っています。従来は、大量生産の製品のもと、型に当てはめて使わなければいけなかった事柄も、コンピューティングパワーが安価になり、スマートフォンが一般化したおかげでパーソナライズやレコメンデーションの技術が進化し、人それぞれ、多様性を持ったニーズにも対応することができるようになってきたわけです。

また物理的に豊かになり、必要な物がすぐに手に入るようになったことで、「モノ」に固執するよりも、体験や経験といった「コト」がより大事にされるようになってきました。デジタルガジェットも、新しいもの好きの、それほど役に立たない嗜好品というよりも、より多様な人々に役立つ「コト」を提供するためのデバイスになってきているのかもしれません。

モノよりコトだと実感する顕著な例の1つが、音声や、声によるデジタルデバイスの操作が台頭してきたことです。最近では音声入力、音声操作も、結構なレベルで認識され、さまざまな事柄にも対応しはじめています。人間本来の振る舞いを考えると、考えたことを口に出して操作できるのであれば、それが一番楽なハズです。SD

Gadget 1

Pop Home Switch

http://www.logitech.com/en-us/product/pop-home-switch

スマートホーム向け物理ボタン

Pop Home Switchは、家のなかにある照明や家電製品など、スマートフォンで操作可能なIoTデバイスを、物理ボタンで操作できるようにするためのデバイスです。たとえば、部屋にスマートフォンを持っていない子供だけしかいない場合でも、複数の組み合わせをあらかじめ設定しておいたボタンを押すだけで実行させられます。設定は簡単で、家のWi-Fiネットワークを自動的にスキャンし、リモート操作に対応している装置を登録・設定します。複数のボタンをそれぞれ違う用途で使えます。

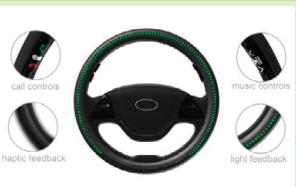
Gadget 2

The Smart Steering Wheel Cover

https://smartwheelusa.com/

ハイテクハンドル

The Smart Steering Wheel Coverは、車のハンドルに取り付けるだけで、さまざまなデジタル操作が可能になるハンドルカバーです。スマートフォンの電話の受話器の操作や、音楽アプリの操作が可能です。また振動やライトが光る反応で、急な加速を警告し、燃費の良い安全運転を指南してくれます。定価は199ドル、早期予約で149ドルで受付中です。発売時期は未定です。最新鋭の自動運転車でなくとも、ハンドルにデバイスを装着するだけで新機能が追加されるのが特徴です。



Gadget 3

Bots

http://www.kevingaunt.com/bots/

スマートホーム向け人工知能

Botsは、拡張可能なスマートホーム向けの人工知能です。高齢者が1人で暮らすことを援助します。家庭内の家電製品のコントロールを追加したり、交換してアップグレードしたりすることができます。機能が異なる18種類のボットによって構成され、部屋ごとに置かれたマイク&スピーカーで、対話型の操作が可能です。18種類のボットは、音声メモ、オンラインショッピング、占い、家電コントロールなどの機能を持ちます。現在はまだコンセプトモデルです。

Gadget 4

Post/Biotics

http://www.post-biotics.com/

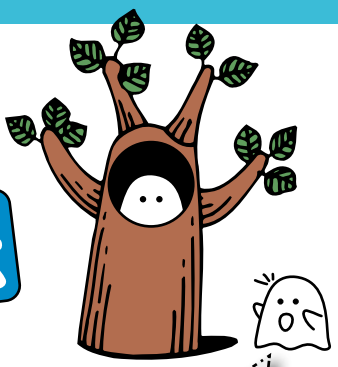
化学実験キット

Post/Bioticsは、子供達の学習用途、民間の研究用途として提供されている、小さな化学実験室キットです。菌類を培養したり、観察したり、土壌を観察したり、さまざまな用途に用いられます。主な用途として考えられているのは、身近にある植物、野菜、果物、キノコや土壌から新たに抗生物質を発見するという、手間も時間もかかる面倒な作業を、たくさんのアマチュア研究家が手分けして行おうというクラウドソーシング的研究です。80ポンド（約11,000円）で予約受付中です。

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

48

結城浩の再発見の発想法



ACK

ACKとは

ACK(アック)とは、通信において、**データが受信できる状態であることや、データが正しく受信できたことを表す応答**のことです。その性質上、受信者が送信者に対して返すものになります。ACKはAcknowledge(アクノリッジ、肯定応答)の略です。ACKの反対語はNAK(ナック、Negative Acknowlege、否定応答)で、相手への問い合わせはENQ(Enquiry)と言い、ASCIIコードの制御文字にも、ACK(0x06)、NAK(0x15)、ENQ(0x05)のように割り当てられています。

テレタイプ

19世紀から20世紀にかけて使われていたテレタイプという通信端末では、通信相手が現在データを受信できる状態かどうか問い合わせて通信しました。送信者は、データ送信前にENQを送ります。これは受信者へ「あなたは受信できる状態か」と問い合わせを行っていることになります。受信者は、受信できる状態ならACKを返し、受信できない状態ならNAKを返します。送信者は、ACKが返ってきたらデータを送信しますが、NAKが返ってきたら送信しません。

また、データが受信できるかどうかだけではなく、データが正常に受信したかどうかを示すときもACKが使われます。受信者は、受信したデータが壊れていないかどうかを調べ、正常に受信した場合はACKを返し、そうでないときにはNAKを返します。送信者は、受信者からの応答を調べ、NAKだったらデータの再送を行います。

テレタイプは現在ではほとんど使われていませんが、「**ACKを返す**」という概念は通信プロトコル中で(あるいは技術者の日常会話で)よく使われています。

TCP/IP

TCP/IPで通信を行うときには、接続の確立、データの送受信、そして接続の終了(要するにすべての状況)でACKに関する情報もやりとりされます。TCP/IPは信頼性のある(データの到達や順序を保証する)通信プロトコルで、信頼性を保つために、ACKと再送のしくみが使われます。

TCP/IPでクライアントがサーバと接続を確立するときには、「3ウェイ・ハンドシェイク」という方法を用います(**図1**)。

①クライアントは、サーバに対してSYN(Synchronize)を含むパケットを送ります。SYNは、サーバに対する接続要求を表します
②サーバは、①に答えてACKとSYNとを含むパケットを返します。ACKは、クライアントから送られてきたSYNに対する肯定応答を表し、SYNは、クライアントに対する接続要求を表します
③クライアントは、②に対して、ACKを含む

パケットを返します。ACKは、サーバから送られてきたSYNに対する肯定応答を表します

TCP/IPでは、クライアントとサーバでいきなり通信が始まるわけではなく、このように互いにSYNを送り合い、SYNに対するACKを相手から受け取って初めて通信が始まるのです。

TCP/IPで受信者は「正常に受信したかどうか(ACKフラグ)」だけではなく、「正常に受信したのはデータのどこまでか(ACK番号)」も返します。ACK番号によって送信者は「再送があり得るのはデータのどこからか」を判断できることになります。

TCP/IPでは、タイムアウトも用います。送信者は、定められた時間内にACKが来なかったら、経路の途中で何かが発生したと考えて再送を行います。

日常生活とACK

私たちの日常生活では、他人との多様なやりとりが発生しますので、ACKに相当するものはたくさんあります。

たとえば、誰かから説明を聞いているときに返す**うなずき**は、もっとも単純なACKと言えるでしょう。話し手の説明に対して聞き手がうなずくことで、話し手は「相手に説明が正しく伝わっているんだな」ということがわかります。うなずき以外でも、**相づち**や、わかっていますよという**表情**によって、聞き手から話し手に肯定的な応答を伝えることができます。

否定応答であるNAKもよく使われます。**首を傾げる動作**や、**眉をひそめる動作**などを使えば、聞き手から話し手に否定的な応答を伝えることができます。

これらは、通信プロトコル同様に、スムーズなコミュニケーションには欠かせません。話し手は、聞き手から送られてくるACKやNAKを見分けて、

- そのまま話題を先に進める
- もう一度説明を繰り返す
- さらにブレークダウンした説明を行う

などの判断をリアルタイムで行っていることになりますね。

もしも、会話をしている相手が無表情で無言なら、たいへん話しにくいでしょう。それは、相手に話が届いているかどうかの手がかり、すなわちACKやNAKが返ってこないからです。技術者同士の会話では、話し相手が黙りこんだときにしばしば「ACKは？」と尋ねたりします。

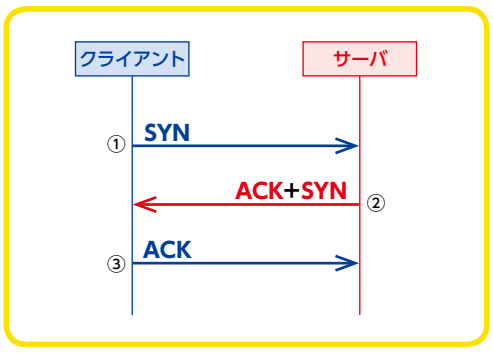
さて、作業者にメールで依頼をしたときには、作業者から「了解しました」という**返信メール**がやってくることを期待します。返信メールはまさにACKの役割を果たしていますね。「時間が取れないのでお引き受けできません」という返信はNAKに相当します。作業者からの返信を確認するのは大事です。依頼メールがスパムフォルダに入っていて「実は届いていなかった」という事故があり得るからです。しばらく返信が来ないので依頼メールをもう一度送るのは、TCP/IPでACKが返ってこないときに再送するのとまったく同じです。

◆ ◆ ◆

あなたの周りを見回して、コミュニケーションで、どんな種類のACKが使われているかを観察してみましょう。もしもそこで、ACKが返らなかったら、どんな問題が起きるでしょうか。

ぜひ、考えてみてください。SD

▼図1　TCP/IPの3ウェイ・ハンドシェイク



及川卓也の プロダクト開発の道しるべ

品質を高めるプロダクトマネージャーの仕事とは？

第7回

アジャイル開発におけるPMの役割

Author 及川 卓也（おいかわ たくや）
Twitter @takoratta

アジャイルソフトウェア開発

アジャイルソフトウェア開発（以下、アジャイル開発）は、目まぐるしく変わる要求や技術に柔軟に対応するためのさまざまなソフトウェア開発手法の総称です。本誌の読者には釈迦に説法かもしれませんが、ここであらためてその本質について考え、アジャイル開発におけるPM（Product Manager；プロダクトマネージャー）の役割について考えてみましょう。

アジャイル開発が生まれた背景には、ウォーターフォール開発に代表される従来のソフトウェア開発手法への反省があります。そもそもアジャイル（agile）は「素早い、機敏な、敏捷な、活発な」というような意味を持つ形容詞です。わざわざこのような形容をするのは、それまでのソフトウェア開発がその真逆の「遅く、緩慢で、鈍重で」あったことへのアンチテーゼです。

ソフトウェア開発・システム開発においては、その規模が大きくなればなるほど、必然的に組織横断的に関わる人が多くなります。しかし、それにもかかわらず、組織間のコミュニケーションやコラボレーションについて考えられることが少なく、本来ならば利害は一致したチームとして同じゴールを目指すべきものが、ときには対立を生むことさえあるような状況だったのが従来の開発手法でした。

このような反省からアジャイル開発は生まれました。アジャイル開発の精神はアジャイルソフトウェア開発宣言[注1]にまとめられています。

［アジャイルソフトウェア開発宣言］

私たちは、ソフトウェア開発の実践
あるいは実践を手助けをする活動を通じて、
よりよい開発方法を見つけだそうとしている。
この活動を通して、私たちは以下の価値に至った。

プロセスやツールよりも個人と対話を、
包括的なドキュメントよりも動くソフトウェアを、
契約交渉よりも顧客との協調を、
計画に従うことよりも変化への対応を、

価値とする。すなわち、左記のことがらに価値があることを認めながらも、私たちは右記のことがらにより価値をおく。

アジャイル開発の特徴は次のような点にあります[注2]。

- 反復と追加と進化を大事にする
- 顔を突き合わせての効果的なコミュニケーションを追求する
- 短期間でのフィードバックと素早い適応を実現する
- 品質を重んじる

注1）http://agilemanifesto.org/iso/ja/manifesto.html
注2）詳しくはアジャイル宣言の背後にある原則またはアジャイル12の原則と呼ばれる文書を参照してください。
http://agilemanifesto.org/iso/ja/principles.html

アジャイルの「素早い」という言葉に現れているように、アジャイル開発の肝は短い期間での反復（イテレーション）を通じて、常に進化し続けることにあります。短い反復に呼応するように、振り返りを通じての学習と対応、そしてそれに欠かせないコミュニケーションが、すべてのアジャイル手法の共通概念となります。

組織横断的なプロジェクトにおいてのコラボレーションが重視されることなど、アジャイル開発はプロダクトマネージメントが重要となる状況と近いものがあります。この連載でお伝えしているように、プロダクトマネージメントはエンジニアやデザイナー、品質管理を行うQA（Quality Assuarance）、サポート、マーケティングや広報、そして営業などのビジネスサイドまで含めたプロダクトチームをリードすることを通じて、プロダクトの成功を目指す仕事です。

実際、現代のPMが働くプロジェクトでアジャイル開発を用いていることは多くあります。では、アジャイル開発におけるPMの役割とは何なのでしょうか？

アジャイル開発とPM

従来までのソフトウェア開発手法と異なるアジャイル開発であっても、一種のプロジェクトマネージメントと考えることはできます。プロジェクトマネージメントという言葉に違和感を感じることはあるかもしれませんし、実際そのように解釈することに若干の無理があることは否めません。しかし、ここでは議論を単純化するために、アジャイル開発をプロジェクトマネージメントとしてとらえましょう。

そのように考えると、前回（本誌2017年4月号）の記事で書いたように、プロダクトマネージメントとプロジェクトマネージメントの違いが、そのままアジャイル開発とプロダクトマネージメントにも当てはまります。一番端的に理解できるのは、そのスコープでしょう。

アジャイル開発はプロジェクトが完了し、顧客に提供した時点で終了します。顧客の関与が従来型ソフトウェア開発手法よりも早い段階から頻繁に行われることにより、反復を通じて、品質と価値を高めることができますが、それでもプロジェクトという単位でアジャイル開発は終了します。

それに対し、プロダクトマネージメントはその製品が続く限り続きます。顧客に提供されたあとのフォローアップも、フィードバックをもとにした改善や次のバージョンの計画もプロダクトマネージメントの一環となります。

アジャイル開発とPMの相性の悪さ

とは言うものの、アジャイル開発においてPMは若干の相性の悪さみたいなものを感じることがあります。それは先ほど紹介したアジャイル憲章の中にも書かれているドキュメンテーションへの考え方です。

アジャイル憲章の中では「包括的なドキュメントよりも動くソフトウェアを」と書かれています。この考えはソフトウェア開発やソフトウェア工学のコミュニティの中でも、ドキュメント軽視の現れではないかと議論になることがあるようですが、テキスト化された情報を通じて、プロダクトを定義し、チームをまとめるというPMの典型的な役割とも相容れないものがあります。

この「包括的なドキュメントよりも動くソフトウェアを」という考えの意味するところは、包括的な（完全さを求める）ドキュメント作成は開発者の時間の使い方として必ずしも最善ではないということのようです。往々にして、そのようなドキュメントよりも動作しているソフトウェアのほうが多くを語り、アジャイル開発が適していると考えられるような迅速さを要求されるプロジェクトではドキュメント作成は時間の無駄であることも多いことから、この考えは出てきているようです。

そのように考えると、必ずしも「包括的なドキュメントよりも動くソフトウェアを」という

考えはPMの価値観や役割と矛盾するものではありません。変化の激しいプロダクト開発の中で、動くソフトウェアだけでは提供できないものをドキュメントの形でしっかりと記録していくものと考えれば良いでしょう。

また、反復を繰り返し、変化を遂げていくアジャイル開発を適用した**プロジェクト**においては、ドキュメントよりも動くソフトウェアを重視し、プロジェクトをまたがる、もっと長期の視野での**プロダクト**の普遍的な価値などはドキュメント化していくという考えも可能です。

いずれにしろ、ドキュメントは製品そのものではありません。また、ドキュメント化作業はともすると、その完成度をあげることが目的化してしまうこともあるので、このアジャイル憲章の考えは常に肝に銘じておく必要があります。

具体的には、この連載でも紹介したPRD（Product Requirements Document）を必要以上に重厚なものとならないように心がけ、状況に応じてはその軽量版を用意するなどすると良いでしょう。また、英語には1枚に要旨をまとめたもののことを表す“One Pager”という言葉がありますが、もしOne Pagerで済むならば、それで済ますなども良いと思います。GitHubのIssueにWhat、Goals、Non-Goals、Why、Reference、ToDoなどをテンプレートとして用意し、それを簡易PRD代わりにするというのもよく取られる手段です。

スクラム開発とPM

アジャイル開発の中でも、とくによく使われているのはスクラム開発ではないでしょうか。ナレッジマネジメントの大家である野中郁次郎氏の“The New New Product Development Game”というハーバード・ビジネス・レビューに掲載された記事で、新しい製品開発のあり方はラグビーにおけるスクラムのように、全員が一丸となってゴールを目指す必要があると提唱したことが名前の由来です。

記事の中に示されている**図**のType AやType Bでは、最初の計画時に関与した人がゴールである製品開発プロジェクトの完了時にはすでにいないために、伝言ゲームのように本来目指すものと異なるものができてしまうことや、それを防ぐためのコミュニケーションコストが大きくなることが問題となります。

Type Cが今日スクラム開発と言われているもののモデルで、全員が一丸となってゴールを目指すものです。

スクラム開発における開発チームは、まさにこのラグビーでスクラムを組むようなイメージとなります。同じゴールを目指すために、コミュニケーションも密にし、決められた行動様式を持ちながらも、状況に応じて柔軟に対応します。チームの人数は5名から9名程度とされています。

本誌の読者はすでにご存じのように、このチームの中で特別な役割を持つのが次の2名となります。

・プロダクトオーナー（PO）
・スクラムマスター（SM）

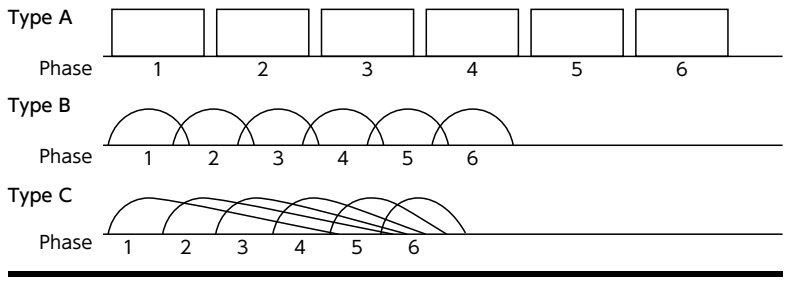
プロダクトオーナーは製品に関わる人を代表する立場であり、顧客の声を代弁することが期待されます。開発チームが正しい価値を顧客に提供することを推進するのがプロダクトオーナーになり、1つの開発チームには1人のプロダクトオーナーがいる必要があります。

スクラムマスターは開発チームがスクラムという開発手法を正しく適用する責任を持ち、チームのファシリテーションと障害対応をおもに行います。

このような形でプロダクトオーナーとスクラムマスターはスクラム手法を採用した開発チームの中で役割を分担していますが、原則的には同一人物がこの2つの役割を兼任してはならないとされています。

では、PMはどちらの役割を担えばよいでしょう。先ほどの役割の説明から恐らく自明だと思

▼図　スクラム開発の元となった野中郁次郎氏の論文の中で、全員が一丸となった開発を説明した模式図



出典：The New New Product Development Game, Harbard Business Review, 1986年1月
https://hbr.org/1986/01/the-new-new-product-development-game

いますが、プロダクトオーナーです。

PMとPO

プロダクトオーナーは顧客の要求をユーザーストーリーという形式で定義します。これはユーザ視点でプロダクトに期待するものを記述したもので、「ユーザは×××をできる」という形でシンプルにまとめます[注3]。このユーザーストーリーにストーリーポイントという見積もりを付け、プロダクトオーナーが決定した優先度と合わせて、掲載したものがプロダクトバックログになります。このプロダクトバックログから反復期間（イテレーション）ごとのタスクとして取り上げるユーザーストーリーを決め、イテレーションを何度も繰り返し、開発を進めていくのがスクラム開発の基本となります。

このように、ユーザーストーリーはスクラム開発のタスクの基本単位となり、これを用意するのがプロダクトオーナーの役割となります。さて、このユーザーストーリーとPRDはどのように使い分ければよいのでしょう。

いろいろな考え方がありますが、スクラム開発を適用したプロジェクトの場合はPRDではなく、ユーザーストーリー中心に進めていくのが良いとされているようです。これはすでに紹介した「包括的なドキュメントよりも動くソフトウェアを」というアジャイルの考えにもつながります。

いくら簡略化するとは言っても、PRDはどうしても重厚になりがちです。PRDをマスタープランとして用意するのは悪くはないですが、それよりもユーザーストーリーを中心にチームと会話していくことのほうが効率的です。PRDを用意するとしても、その中のユースケースの項目はユーザーストーリーで置き換えられるので省略し、それ以外の部分だけ記述するので十分でしょう。

これ以外にも、PMとプロダクトオーナーは類似点や相違点はありますが、人によって解釈は異なります。確実に言えるのは、PMは古くからある職種（だけれどもその定義はまちまち）であり、プロダクトオーナーはスクラム開発とともに一般的になった比較的新しい職種（だけれどもその定義は明確）であるということです。

ですので、その違いや職種にこだわるよりは、プロダクトの成功を目指すという共通のゴールに向けて、組織やプロジェクトにふさわしい役割をまっとうすることを考えるのが良いでしょう。SD

注3）よりユーザの視点を明確にするために、ユーザの役割とその目的を含む場合もあります。たとえば、楽天市場のようなECサイトではユーザは購買者と店舗側の両方が存在するので、「購買者としての私は同じ商品を価格順に眺めることができる。それにより最安値で購入が可能となる」という形式です。

Profile
ソフトウェアエンジニアとして社会人キャリアをスタートした後、MicrosoftやGoogleでプロダクトマネージャーやエンジニアリングマネージャーを経験。現在はプログラマーのための情報共有サービスQiitaのプロダクトマネージャーを務める。

宮原徹の

オープンソース放浪記

第15回 春だからこそ入門したいよね

宮原 徹（みやはら とおる） Twitter @tmiyahar 株式会社びぎねっと

毎度お馴染み 明星大学での開催です

オープンソースカンファレンスで規模の大きい開催は東京と京都ですが、東京は春秋2回の開催です。今回のOSC2017東京春も、毎回おなじみ東京都日野市にある明星大学のキャンパスをお借りして開催されました。

毎回、いろいろな人に「遠い！」と言われてしまうのですが、開催規模に対して安価に、かつ広いスペースを取ることができる場所がほとんどありません。都心からの距離を抜きにすれば、大学関係の方々にもたいへんご協力をいただいており、かつ今のところよい代替案がないこともあって、毎回明星大学のキャンパスをお借りして開催しているしだいです（**写真1**）。いつもありがとうございます。

▼写真1 当日早朝からの準備に備えて、高幡不動に泊まり込む弊社スタッフ。夜はホテルの部屋で「けものフレンズ」鑑賞会

OSSは「普通」になり過ぎてしまったのですか？

今回の開催では、初日の3月10日（金）に600名、3月11日（土）に700名、両日合わせて1,300名の来場者がありました。また、展示ブースも5教室、セミナーも数え切れないぐらい開催され、盛況の内に終了できました。

しかし一方で、来場者合計数は年々減少しているのも事実です。以前なら700名、800名の合計1,500名ぐらいは来るかな？と予測していましたが、今回はやや無難な数字に落ち着いてしまいました。

減少傾向の理由はいろいろとあるかと思いますが、一番大きな理由は「オープンソースが普通になった」ということが挙げられるのではないでしょうか。

OSCをスタートした10年以上前は、OSSの利用に対して懐疑的な意見も多かったので、活用事例や技術解説が多く求められていました。しかし、現在では業務システムでのOSS活用は当たり前ですし、事例や技術情報はネット上に溢（あふ）れています。わざわざ情報収集のためにOSCに来るインセンティブが働きにくいのは事実です。

しかし、OSCのような集まりに価値や魅力がなくなったわけではありません。実際、会場内は和やかに、かつ活気に溢れていましたし、初日の金曜日の夕方には、詰め襟学生服を着た高校生が連れだって来場してくれました。土曜日はお子さん連れの方が多数来場してくれていました（**写真2**）。

Twitter上でもたくさんの「すごーい！」「たーのしー！」が溢れています。ぜひ、ハッシュタグ#osc17tkで検索してみてください。この雰囲気を大事にしながら、今後も継続的にOSCを開催していきたいですね。「変わらないでいるためには変わらないといけない」ということで、いくつか新たな取り組みを始めていますので紹介したいと思います。

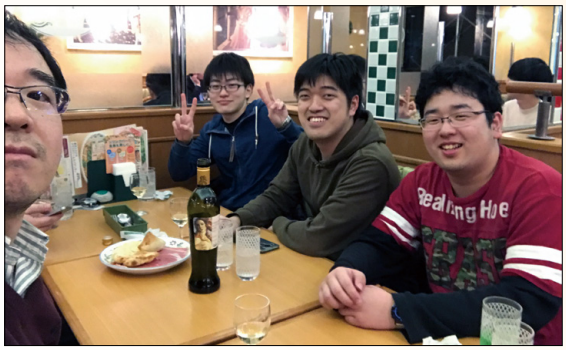
今、あらためて入門セミナーをやる意味

まず最初に紹介したい取り組みが、入門セミナーを企画開催したことです。開催したセミナーは以下のとおりです。

- コマンド入門
- Linuxインストール入門
- Linuxシステム管理入門
- ファイルサーバ入門

▼写真2 NTTデータの「Hinemos」のマスコットキャラ「もにた」と来場されたお子さん

▼写真3 恒例の学生スタッフとサイゼリヤで打ち上げ。今回もたくさんの学生スタッフが手伝ってくれました



どれもこれも、Linuxなどを日常的に使っている本誌読者には今さら感があるテーマかもしれません。しかし、まだまだこれからLinuxやOSSを使い始めたい、という初心者はいるものです。実際、各セミナーとも参加者は40名以上で教室は満席になっており、入門セミナーを目的にOSCに参加した、という人も多くいました。まだまだ、OSSの仲間に加わりたい人はたくさんいるようです。

今回の入門セミナーは、有志が講師を務めましたが、作成した資料はクリエイティブコモンズでライセンスして自由に再利用できるようにしました。さらに、今後の東京以外の地域でのOSCでも開催していく予定です。データベース入門など、そのほかのテーマの入門セミナーも順次そろえていこうと考えています。

継続的に開催してきているからこそ、振り返って、これからの人たちのための情報発信もしていかないといけないですね。

OSC学生スカラーシップ制度を始めます

もう1つの取り組みが、OSCに出展したい学生に対して交通費などを補助するOSC学生スカラーシップ制度です。これまでもたくさんの学生スタッフが運営を担ったり、OSCに出展している様子をこの連載で紹介しました(**写真3**)。

意欲のある学生に対して、ほかの地域のOSCへの出展を支援していきます。今回はこの制度の原資として、3年間で300万円まで使える予算を確保しました。予算を明確にしたことで、より積極的に支援が行えるようになるかと思います。全国各地の若者を発掘して、OSCで次々とデビューさせていきますよ！　ご期待ください。SD

Report

いつものように懇親会も盛大に

「OSCの半分は懇親会でできている」は誰かの名言ですが、今回も盛大に開催されました。今回は、さくらインターネットのエバンジェリストである横田真俊氏と2人で「あすなろブラザーズ」を結成して、「2017年5大ニュース」と題したかけあい漫才をしたり、毎度お馴染みお酒を持ち寄る「Bar root」を開店したりと大忙しでした。また、OSC2015東京秋(第4回)から出展していただいているサードウェアさんから20周年記念の紅白まんじゅうが来場者に配られたりして、OSCの夜は更けていくのでした。みんな、明日もあるんだからな！

◀ライトニングトークの司会をしてくれたIoT女子のみなさん。懇親会でもいろいろな人と交流してくれました

▼サードウェアの久保元治社長から紅白まんじゅうをいただきました。20周年おめでとうございます

ツボイのなんでもネットにつなげちまえ道場

第23回

Mongoose OSを使ってみる

Author 坪井 義浩（つぼい よしひろ） Mail ytsuboi@gmail.com @ytsuboi
協力：スイッチサイエンス

Mongoose OS

今回は、Mongoose OS[注1]（マングースオーエス）というオープンソースのIoT向けOSのことを聞きましたので、これを使ってみたいと思います。このソフトウェアは、アイルランドのCesantaという会社のプロダクトのようです。オープンソースということで、ソースがどこかと探してみたところ、GitHub[注2]でホスティングされていました。あとでよくMongoose OSのトップページを見てみると、GitHubへのリンクや、ライセンスについての説明が記載されています。ライセンスの説明をざっと読む限り、Mongoose OSはGPL v2で提供されていますが、商用利用の際にはコマーシャルライセンスを購入することも可能だというデュアルライセンスのプロダクトのようです。

GoogleのAndroid Things[注3]のように、Linux Kernelを使ったCortex-Aなどのプロセッサ向けのIoT向けOSが最近多く見られます。一方でMongoose OSは、これまで扱ってきたCortex-Mや、最近話題のESP8266やESP32といったマイコンで動かすためのOSです。

今回は、Mongoose OSのブログポスト[注4]に従い、Mongoose OSを使ってAWS IoTに接続できるボタンの実験をしてみます。AWS IoT[注5]は、Amazon Web Servicesが提供するサービスの1つで、IoTのエンドノードからMQTT、HTTPなどのプロトコルを使った接続、認証、メッセージ交換を行うためのものです。

ESP32

今回Mongoose OSを試用するにあたって、ESP32（**写真1**）を選択してみました。ESP32は最近話題のWi-FiとBluetoothインターフェース搭載のTensilica LX6のデュアルコアなマイコンです。Tensilicaというのは、ARMと同じようにIP（Intellectual Property：知的財産）、つまり半導体の設計情報を販売している会社です。今回、ESP32を使ってみたのは、このチップを搭載したESP-WROOM-32というモジュールや、このモジュールを搭載した開発ボードが発売されたてで話題だからです。

筆者はこのESP32-DevKitCを、秋月電子通商[注6]で購入しました。このように開発ボードも安価ですし、モジュールのESP-WROOM-32も

注1） https://mongoose-os.com
注2） https://github.com/cesanta/mongoose-os
注3） https://developer.android.com/things/index.html
注4） https://mongoose-os.com/blog/internet-button-on-esp8266-and-amazon-aws-iot-in-2-minutes/
注5） https://aws.amazon.com/jp/iot-platform/

注6） http://akizukidenshi.com/
通販コードM-11819、税込1,480円

▼写真1 ESP32-DevKitC

税込700円程度と安価に流通しており、気軽にIoTのプロトタイピングを行うことができます。

先ほど紹介したAWS IoTに接続できる機器の1つに、AWS IoTボタン(**写真2**)があります。しかしAWS IoTボタンは技適マークが付いていないために電波法令で定められている技術基準に適合するかどうかわからず、電波法違反になる場合があります。しかし、このESP-WROOM-32には技適マークがついており、国内で適法に使用できそうです。

開発環境の構築

さっそく、Mongoose OSの開発環境のセットアップを行いましょう。まず、ダウンロードページ[注7]にアクセスします。筆者はmacOS(Mac OS X)を利用しているので、ターミナルを立ち上げ、記載どおり次のコマンドを実行してインストールしました。

```
$ curl -fsSL https://mongoose-os.com/downloads/mos/install.sh | /bin/sh
```

インストールの進捗を眺めていると、libftdiというFTDI社のUSB-UARTブリッジのドライバがインストールされているのを見かけました。しかし、今回使っているESP32開発ボードで採用されているUSB-UARTブリッジのチップは、Silicon LaboratoriesのCP2102というチップです。そこで、同社のドライバ配布ページ[注8]にアクセスし、ドライバのインストールを行いました。これで、ESP32開発ボードをパソコンに接続すると、シリアルポートが認識されるようになるはずです。

Mongoose OSの開発環境と、USB-UARTブリッジのドライバのインストールを終えたので、ESP32開発ボードへのMongoose OSのインストールを行いましょう。まず、パソコンにUSBでESP32開発ボードを接続します。次に、先ほどのWebサイトに記載されていた次のコマンドを実行します。

```
$ ~/.mos/bin/mos
```

すると、**図1**のようにWebブラウザが立ち上がり、http://127.0.0.1:1992を開きました。

インストール

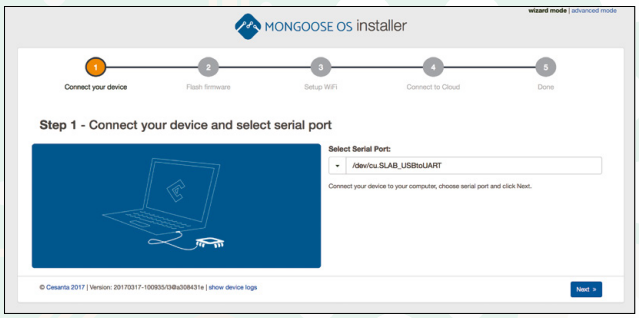
さて、インストーラが立ち上がったので、インストールを進めていきましょう。Step 1では、ESP32開発ボードのシリアルポートを選択します。これで、パソコンとESP32がコミュニケーションできるようになりました。そのままStep 2とStep 3に進み、ファームウェアのインストールとWi-Fiへの接続の設定を済ませます。Step 4では、クラウドへの接続方法を指定します(**図2**)。ここでは、AWS IoTを選択し、AWS IoTインテグレーションの設定に進みました。

注7) https://mongoose-os.com/software.html

注8) http://jp.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

▼写真2　AWS IoTボタン

▼図1　インストーラが立ち上がったところ

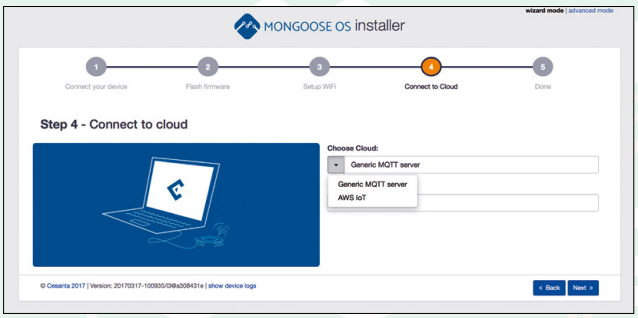


AWS IoTを使用するには、AWSのアカウントが必要です。また、Mongoose OSのAWS IoTインテグレーションを使用する際には、AWS側で作ったAccess Key IDとSecret Access KeyをMongoose OSのセットアップで入力する必要があります。これらは、AWS IAM(Identity and Access Management)というサービスを利用して認証を行うためのものです。ここではIAMに関する詳しい説明は省きますが、AWSのアカウントを作ったあと、IAMのユーザを作り、AWSのユーザガイド[注9]に従いAccess Key IDとSecret Access Keyを発行して入力をしました。

AWS IoTのリージョンは、どこでも差し支えはないと思いますが、筆者は東京である「ap-northeast-1」を選択しました(**図3**)。また、筆者はすでにAWS IoTを利用していたため、ポリシーを登録済みでしたが、ポリシーが1つ以上登録されている必要があるようです。

これでインストールは完了です。AWSを使い慣れていないと戸惑うことも多いかと思いますが、IAMのユーザの登録と、AWS IoTのポリシーの登録ができれば、とくに引っかかることはないかと思います。

▼図2　クラウドへの接続の設定



▼図3　AWS IoTへの接続の設定

▼図4　開発を行う画面

注9) http://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-getting-set-up.html#cli-signup

サンプルコード

インストールを終えると、「START PROTOTYPING」というボタンが表示されます。これをクリックすると、Mongoose OSの開発を行う画面(**図4**)に遷移します。ちなみに、次回以降コードを変更したい場合、ESP32開発ボードを接続した状態で、ターミナルで、

```
$ ~/.mos/bin/mos
```

と入力すると、ブラウザが立ち上がり、Mongoose OSのインストーラの画面が表示されます、ここでページの右上に「advanced mode」というリンクがありますので、ここをクリックすると、この画面に戻ってくることができます。

今回は、「Code examples」にある「button_mqtt.js」を使います。左ペインで「Code examples」を選択し、「button_mqtt.js」をクリック、コードが表示された状態で、上にあるオレンジのボタンをクリックします。すると、ESP32のinit.jsが書き換えられ、ESP32がリブートされます。

この状態でAWS IoTのコンソール[注10]にアクセスし、左ペインにある「Test」を選択します(**図5**)。すると「MQTT client」が表示されますので、「Subscription topic」の欄に「mOS/topic1」と入力し、「Subscribe to topic」というボタンをクリックしましょう。すると欄の左側のペインに「mOS/topic1」という行が追加されますので、これを選択します。

この状態で、ESP32開発ボード上にあるBootと書かれたスイッチ(**写真1**のUSBレセプタクルの上側)を押すと、ESP32からメッセージが発行されていることが確認できました(**図6**)。

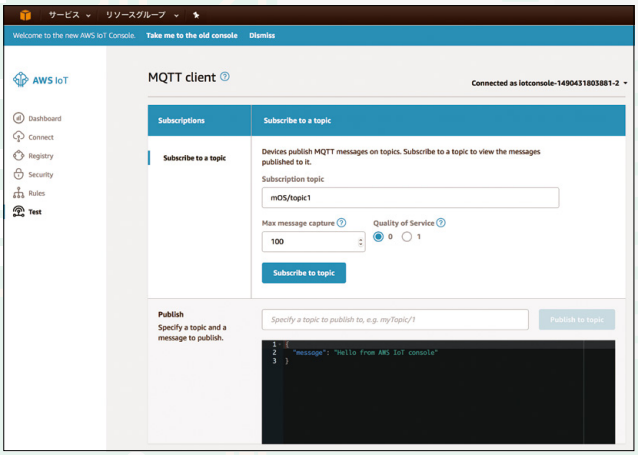
注10) https://ap-northeast-1.console.aws.amazon.com/iotv2/home

まとめ

今回は、Mongoose OSを使ってAWS IoTにメッセージを発行し、AWS IoTコンソールでこれを購読してその確認をしました。ここではMQTTというプロトコルが使われています(**図7**)。

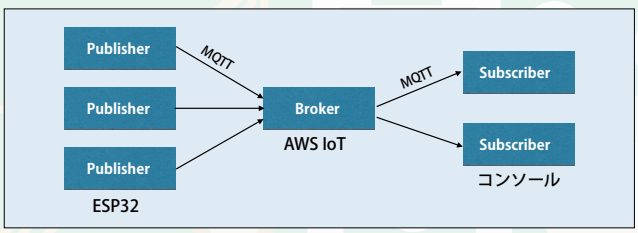
AWS IoTでは、発行されたメッセージをルールとして使い、ほかのサービスと連携をさせることもできます。たとえば、ESP32からAWS IoTにメッセージを送り、それをAWS Lambdaに送って何か処理をさせるといったことができます。次回はこれを実際に試して、自作AWS IoTボタンを実現してみましょう。SD

▼図5　AWS IoTのコンソール



▼図6　コンソールでメッセージを購読

▼図7　MQTTの概略図



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」にアクセスし、アンケートにご協力ください（アンケートに回答するにはgihyo.jpへのお名前と住所のアカウント登録が必要となります）。ご希望のプレゼント番号を記入いただいた方の中から抽選でプレゼントを差し上げます。締め切りは**2017年5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、ご容赦ください。

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。アンケートの回答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。記入いただいた個人情報は、作業終了後に責任を持って破棄いたします。

01 タイプライター風 交換用キートップ「DN-914671」 2名

レトロなタイプライター風キートップセット（英語104キー）です。見た目だけにこだわらず、ひとつひとつのキーが指にフィットする形に若干くぼんでいるので、打鍵感も良好。対象キーボードは、チェリーキーなどのメカニカルキーボード（英語87キー推奨）、お勧めは同社発売の「DN-914217」です。

※キーボード本体は付属しません。

提供元 ドスパラ上海問屋 http://www.donya.jp

02 寝るまでスマホ 3名

寝る直前までスマホを使っていたい人のための、仰向けや横向きでも使えるアーム＆ホルダーです。アームは約40㎝と余裕のある長さ、ホルダーは360度回転でき、好みの位置にスマホを固定できます。6.4インチ以下のスマホで使用できます。

※スマホは付属しません。

提供元 フォースメディア https://www.forcemedia.co.jp

03 Acronis True Image 2017 New Generation（5台1年版） 5名

Windows、MacのHDDのフルイメージ・データをローカルやクラウドへ高速にバックアップできるソフト。前バージョンから、ランサムウェア対策機能などが搭載されました。5台までのPCにインストールできます（1TBのクラウドストレージ付き）。

提供元 アクロニス http://www.acronis.com

04 エンジニアになりたい君へ

森實 敏彦 著

IT業界の内情や会社選びのポイントを詳しく紹介しつつ、一流エンジニアになるためのキャリア形成術、心構えなどを紹介する1冊です。就活を控えた学生、転職を考えている新人・若手にお勧め。

提供元 幻冬舎メディアコンサルティング http://www.gentosha-mc.com 2名

05 ITエンジニアのためのデータベース再入門

真野 正 著

データベース利用におけるアンチパターンを暴き出し、基礎理論を再確認しながら、DB設計やSQLの最適化、運用の改善策を解説していきます。読者対象は、初級から中級のアプリケーションエンジニア。

提供元 リックテレコム http://www.ric.co.jp 2名

06 パーフェクトR

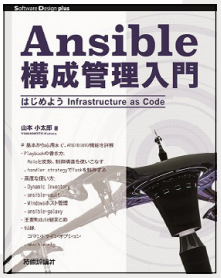
Rサポーターズ 著

R言語の仕様をはじめ、データハンドリングやデータ可視化など基本的な操作から、クラスタリング、クラス分類・回帰、時系列回帰などのデータ分析、Webアプリケーション化まで解説した1冊。

提供元 技術評論社 http://gihyo.jp 2名

07 Ansible 構成管理入門

山本 小太郎 著



構成管理ツール「Ansible」について、インストールから丁寧に解説する初心者向けの入門書です。入門にとどまらず、Play Bookの高速化やWindowsホストの管理方法など実践的なノウハウも紹介。

提供元 技術評論社 http://gihyo.jp 2名

第1特集

新人歓迎企画【第2弾】 先輩が教える得ノウハウ

Linux入門

【UNIXネットワーク編】

イラスト：高野 涼香

ネットワーク技術はやっぱり難しいと思っているあなたへ

新人歓迎企画第1弾【OSの基本操作編】に続き、第2弾の今回は【UNIXネットワーク編】です。

今回も先輩社員が新人社員に教えるストーリーで、「通信プロトコル」「ネットワークコマンド」「ルーティング」「ファイルサーバ」「DNS」などをキーワードに、コンピュータがどのように通信しているのか、インターネットはどのように成り立っているのかを解説します。ネットワークに苦手意識を持つ新人の方は本特集に沿って、基礎知識を押さえ、実際にコマンドを打ち、設定ファイルを書くことで、ネットワーク技術を自分のものにしてください。

第1章 コンピュータはどうやって通信するのか?

Author 五十嵐 綾(いがらし あや) Twitter @ladicle

登場人物

ボブ(先輩)
アリスのトレーナーで入社5年目の頼れるインフラエンジニア。

アリス(新人)
4月に入社したばかりの明るい新入社員。基本情報処理技術者試験は合格したものの実践はこれから。

Webページはどうやって表示されるのか?

研修のため、アリスは先輩のボブと一緒に日本のオフィスからアメリカにあるWebサーバのインストールをしていました。今はブラウザからWebページが正常に表示されるかを確認しているようです(**図1**)。

これで完了だよ。Webブラウザから確認してみよう。

URLを入力して……。先輩、Hello Worldが表示されました!

正しくインストールできているね。インストールも終わったことだし確認テストをしよう。今Webページを表示させたけれど、どのような通信が行われて表示されたかわかるかな?

えっと、ブラウザからURLを入力したので「Webページを開く」というリクエストが送られたと思います。今回はアメリカにあるサーバにインストールしたので、リクエストはインターネットを通ってアメリカのサーバまで届くはずですよね。そのあとは、サーバからWebページのデータが送り返されて、そのデータがブラウザに表示されたんじゃないでしょうか。

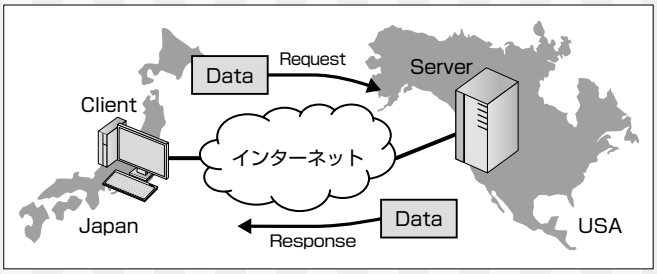
正解。ではもう少し詳しい質問をしよう。そのインターネットとはどんなものかわかるかな?

うーん。もやっとしたイメージはあるんですけど、言葉で説明するのは難しいです……。

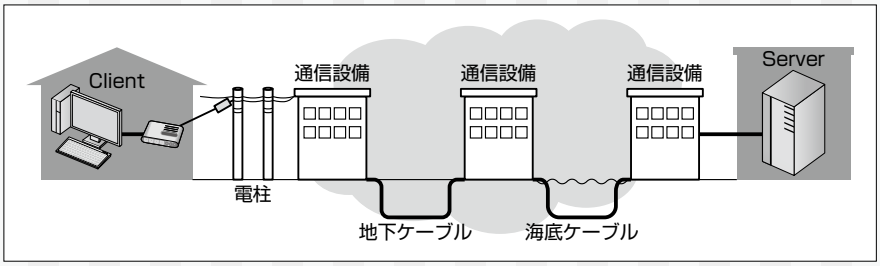
ちょっと難しかったかな。でも大丈夫。次の研修を終えたら答えられるようになるはずだよ。さっそくだけど、そもそもインターネットは何ものなのか説明しよう。

はい! お願いします。

▼図1 ブラウザからWebサーバへアクセスする



▼図2　国外にあるサーバへつながる回線のイメージ



そもそもインターネットとは？

インターネット（the Internet）とは、複数のネットワークがつなぎ合わさったものです。インターネットはよく雲のようなイメージで表現されますが、実際にはこれらのネットワークは物理的なケーブルでつながっています。身近な光回線を考えてみましょう（**図2**）。自宅や会社で見たことがある方もいるかと思いますが、コンピュータにはネットワークインターフェースカード（NIC）と呼ばれる通信ケーブルを接続するための口があります。このNICはデータを電気信号に変換し、ケーブルを介して信号を送信する役割を持っています。コンピュータからつながったケーブルから出た信号は光回線終端装置を通って電気信号から光信号に変換されて、建物の外へ出ていきます。外に出たケーブルは、さまざまな設備で処理されながら電柱を通り、地下を通って目的地にたどり着くのです。また、目的地が国外にあるときは海の底に敷かれている海底ケーブルを通り、海を越えていきます。


なるほど。私が今見ているこのサイトもさまざまなケーブルや設備を経て物理的につながっていたんですね。でも、世界中つながっているとなると設備とか管理がたいへんそうだなぁ。先輩、インターネットは誰が作っているんでしょうか？


良い質問だね。次はインターネットを作っている組織について説明しよう。

インターネットは、1つの組織が管理しているのではなく複数の組織によって作られています。有名どころでは、NTTドコモ[注1]などの携帯電話事業者やOCN[注2]などのインターネットサービスプロバイダ（ISP）、Google[注3]などの大きなコンテンツ提供事業者やインターネットエクスチェンジ（IX）などが挙げられます。IXは聞きなれないかと思いますが、これはインターネットを作っている組織間をつなげるサービスのことです。インターネットは物理的につながっていると説明しましたが、すべての組織を直接つなげることは物理的にもコスト的にもたいへんです。そこで、ほかの組織と接続するための中継ポイントを提供するIXが生まれました。日本ではJPNAP[注4]やJPIX[注5]がこのサービスを提供しています。

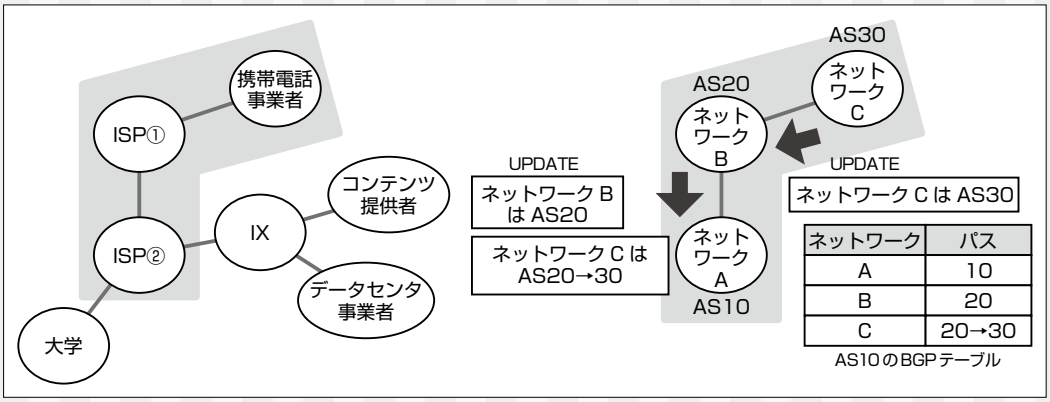

インターネットが1つの組織でないとすると、どの組織を通れば目的地にたどり着くのかどうやってわかるんでしょうか？


では、組織を識別するAS番号と、どの組織が目的地につながっているかを管理するBGPについて説明しよう。

インターネットを構成する組織のネットワークは自律システム（AS：Autonomous System）

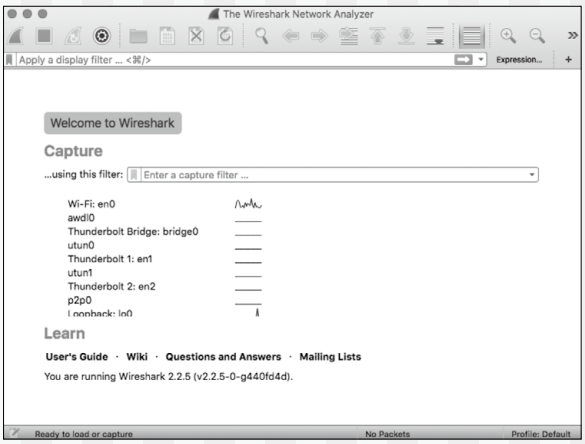
注1）URL https://www.nttdocomo.co.jp/
注2）URL http://www.ntt.com/personal/ocn.html
注3）URL https://www.google.com
注4）URL http://www.mfeed.co.jp/service/jpnap.html
注5）URL http://www.jpix.ad.jp/

▼図3 （左）インターネットを構成するAS例 、（右）BGPで経路情報を交換するAS



と呼ばれ、それぞれを識別するためにAS番号[注6]が振られています（**図3**）。この番号は、インターネット上でユニークな番号で、ICNN[注7]という国際的な非営利組織によって管理されています。AS番号によって組織のネットワークを識別することはわかりました。しかし、これだけだとどのASへデータを送れば目的地にたどりつくのかわかりません。どのように経路情報を手に入れているのでしょうか？　AS間の経路制御には、必ずBGPというプロトコル[注8]が使われています。BGPは、**図3右**のようにAS番号やネットワーク情報などを送り合うことで経路情報のテーブルを作っていくもので、IETF[注9]というインターネットに必要な技術を標準化する組織によって発行されました。また、この仕様はRFCという形式で公開されています。

▼図4 Wiresharkのスタート画面



Wiresharkで通信の中身を見てみよう


一気に説明してしまったけど大丈夫かな？話だけだとイメージができないと思うので、次は実際に手を動かして通信の中身を見てみよう。

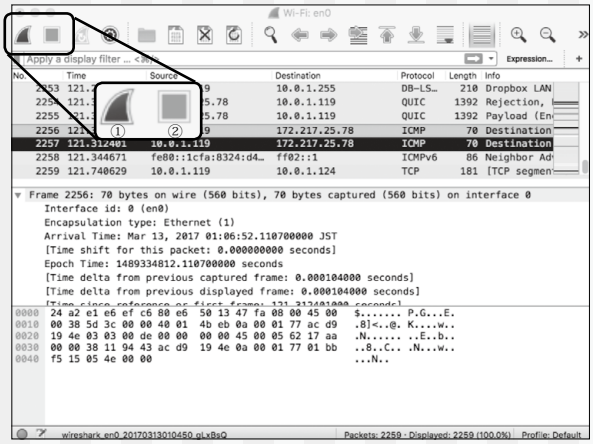

えっ。通信の内容が見れるんですか！


Wiresharkというツールで簡単に見ることができるよ。

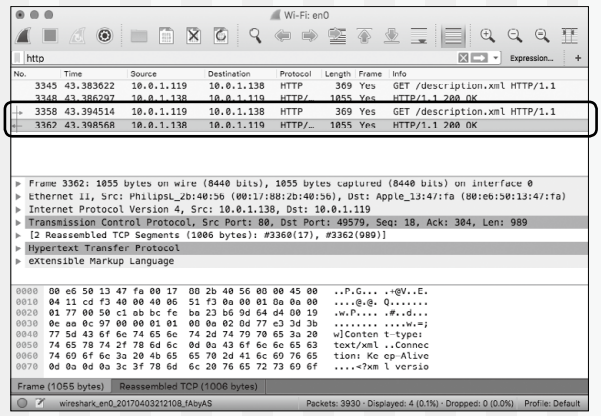
Wireshark[注10]は、コンピュータが接続しているネットワーク内を流れるパケットを取得し、解析するためのツールです。パケットとは、ネットワークを流れるデータの単位のことをいいます。この中には、画像や音楽など実際に送りたいデータに加えて宛先などの制御情報も含まれています。

注6）用途ごとに複数のAS番号を持っている組織もある。
注7）URL https://www.icann.org/
注8）ある処理を共通化し、正確に実行できるように定めた手順書のようなもの。
注9）URL http://www.ietf.org/
注10）URL https://www.wireshark.org/

▼図5　パケットのキャプチャ画面



▼図6　HTTPパケットのフィルタリング画面



この画面は、指定したNICを流れるパケットの一覧画面です。一番上のアイコンバーにはよく使う機能がまとまっています。一番左の①のボタンを押すとキャプチャが開始され、隣の②ボタンを押すと停止します。その下の細い部分は検索バーで、表示するパケットをフィルタリングすることができます。カラフルなウィンドウはパケットの一覧で、見やすいようにプロトコルごとに色が分かれています。その下のウィンドウは選択中のパケットの解析内容が表示されています。

つづいて、パケットのフィルタ機能を使ってWebサイトへアクセスするときのパケットだけを表示させてみましょう。検索バーにhttpと入力してください。**図6**のように、Webサイトとやりとりしているパケットだけを表示できます。HTTPというのはHyper Text Transfer Protocolの略で、Webブラウザなどのクライアントから Webサーバとデータをやりとりするためのプロトコルです。また、No.3358と3362のパケットには矢印が書かれています。この矢印は、クライアントからのHTTPリクエスト（→）と、該当するレスポンス（←）を表しています。これをふまえてNo.3358と3362のInfoカラムを読むと、description.xmlをGETするというリクエストが送られ、200 OK[注13]というレスポンスが返ってきたことがわかります。

さっそくツールをインストールしてみましょう。ダウンロードページ[注11]から自分のOSにあったインストーラをダウンロードしてください。ダウンロードが完了したら、あとはインストーラの指示どおりに進めていくだけです。Mac OSでWiresharkを起動すると、**図4**のような[注12]画面が表示されるかと思います。この画面では、コンピュータから利用できるNICの一覧と通信量のグラフを見ることができます。では、先頭のNIC（ここではen0）をクリックして、パケットを取得し始めましょう。

図5のような画面に切り替わりましたか？

スタック状の通信プロトコル

OSI参照モデルって聞いたことがあるかな？

注11） URL https://www.wireshark.org/download.html（現在の最新バージョンはv2.2.5）

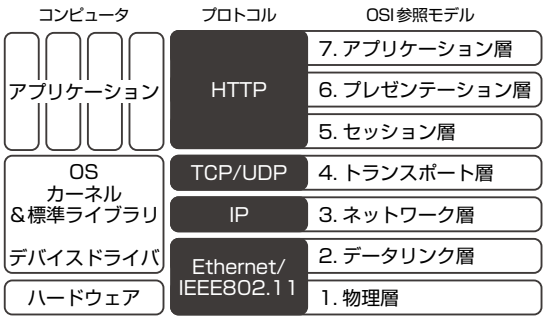
注12） 環境によってNICの数や名前は違います。

注13） 200 OKの意味については、あとで詳しく説明します。


大学の授業で聞いた覚えがあります。たしか……通信プロトコルを7つの層に分けることで、パーソナルコンピュータや携帯とかの違いを吸収しやすくしたものですよね？


そのとおり。では、HTTPがどの層にあたるのか見てみよう。

▼図7　HTTP通信とOSI参照モデル



OSI参照モデル[注14]は国際標準化機構（ISO）によって決められたもので、通信プロトコルを7つの層（レイヤ）に分けて定義しています。このように層を分けることによって各層のプロトコルはほかの層について考える必要がなく、コンピュータの種類の違いなどに対応しやすいというメリットがあります。しかし、**図7**のように、必ずしも1つの層に対して1つのプロトコルが対応するわけではありません。そのため、TCP/IPに特化したTCP/IPモデルや単に階層構造を表すプロトコルスタックと呼ばれることもあります。

物理層・データリンク層の役割

それでは、各層について下から順に説明していきましょう。物理層からデータリンク層は、1つのネットワークに接続されているコンピュータ間でデータをやりとりする方法を定めたものです。通信ケーブルやNICなどのハードウェアと、それらを制御するためのデバイスドライバと呼ばれるソフトウェアから構成されています。この層に該当する通信プロトコルには、Ethernet[注15]やIEEE802.11などがあります。Ethernetは、有線接続のLocal Area Network（LAN）[注16]でよく使われているもので、これはアイ・トリプル・イー（IEEE）と呼ばれるアメリカの電気・情報工学の学会によって定められた規格です。一般的にLANケーブルと呼ばれているものは、Ethernet規格のケーブルと考えて問題ありません。また、IEEE 802.11もIEEEによって定められた規格で、無線LANによく使われています。この名前はあまり知られていませんが、Wi-Fiは聞いたことのある方が多いのではないでしょうか？　Wi-FiはIEEE802.11の1つで、コンピュータ同士が互いに通信できることを保証したもののことをいいます。

ちなみに、携帯電話網の1層と2層では基地局と無線通信を行うためのプロトコルを使用し、さらに2層と3層の間にはモバイルIPと呼ばれるプロトコルが存在しています。モバイルIPは、移動して携帯端末が接続するネットワークが変わっても、同じIPアドレス[注17]を使い続けられるように制御するためのプロトコルです。これによって、携帯でもパーソナルコンピュータでも3層以上は同じプロトコルで同じようにWebサイトを見ることができます。

ネットワーク層の役割

次のネットワーク層は、複数のネットワークに接続されているコンピュータ間でデータをやりとりする方法を定めたものです。おもに、

注14）厳密に覚える必要はない。しかしネットワークについて会話するときに頻出する（とくに3と4層）。

注15）Ethernetの標準化名称はIEEE802.3。

注16）インターネットなど、LANをつなぎ合わせたネットワークをWide Area Network（WAN）と呼ぶ。

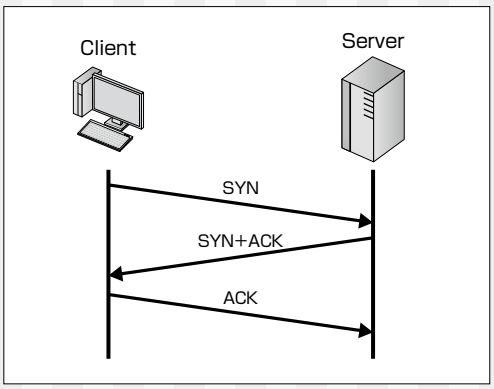
注17）次の節（P.24）で解説しますが、コンピュータの住所のようなもの。

OSのカーネルや標準ライブラリによって制御されています。カーネルとは、CPUやメモリやディスクなどのリソースを制御するためのソフトウェアのことです。同じOS上であれば、アプリケーションから同じ方法でネットワークやストレージなどのハードウェアを利用できるように抽象化しています。また標準ライブラリは、カーネルの機能を呼び出すためのシステムコールなど、アプリケーションが共通して使うようなプログラムを集めたものです。この層に該当する通信プロトコルには、Internet Protocol (IP) などがあります。

トランスポート層の役割

トランスポート層は、ネットワーク層と同じくカーネルや標準ライブラリによって制御されています。この層に該当する通信プロトコルには、Transport Connection Protocol (TCP) やUser Datagram Protocol (UDP) があります。TCPは、パケットの送信エラー時に再送するというエラー制御機能や、バラバラに送られたパケットを順番どおりに並べかえるソート機能を持っています。これらの機能を使うためには、コネクションと呼ばれる論理的な接続をサーバとクライアント間で確立する必要があります。この手順を3ウェイ・ハンドシェイクと呼び、接続要求のSYNと、承認のACKメッセージをやりとりします(**図8**)。もう1つのUDPは、TCPと対照的にコネクションを作らず、エラー制御もありません。そう聞くと欠点しかないように思えますが、複雑な手順がないのでTCPに比べて通信速度が速いという利点があります。そのため、速さや滑らかさが優先される動画配信などに利用されています。

▼図8　TCPと3ウェイ・ハンドシェイク



セッション層・プレゼンテーション層・アプリケーション層

セッション層からアプリケーション層は、その名のとおりアプリケーションによって構成されています。本題のHTTPは、この層に分類されます。ほかにも、前節で説明したBGPもここにあたります。また、はじめに他の層のことは考えなくても良いと説明しましたが、HTTPの仕様では基本的にTCP/IPがネットワーク層とトランスポート層のプロトコルとして定められています。

パケットはどこへ届くのか


EthernetとTCP/IPとHTTPと……プロトコルが組み合わさって通信ができるんですね。

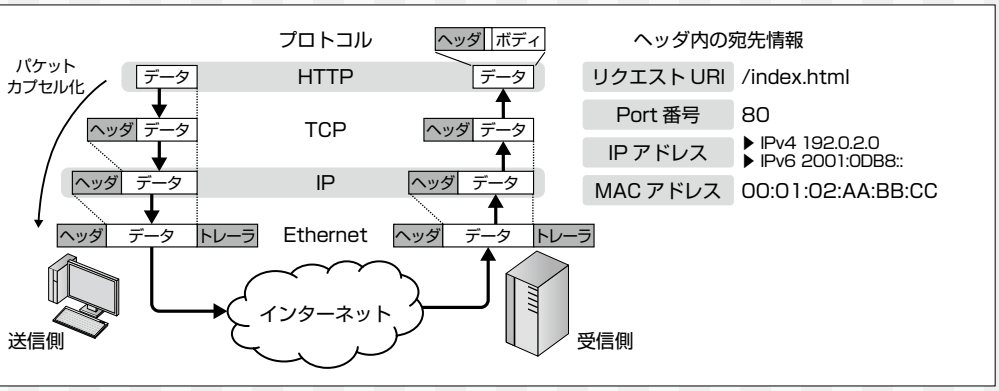

そのとおり。一気に説明してしまったけど、ここら辺の知識は徐々に覚えていけば大丈夫だよ。


う〜ん。パケットが宛先情報を持っているのはわかったんですけど、宛先ってどんな情報なんでしょうか?


宛先といっても種類があるんだ。それでは、パケットの構造と宛先を見てみようか。

前節でパケットは付加情報を持っていると説明しました。この情報というのは、OSIの各層で使われているプロトコルがパケットを制御するために使われています。付加情報には名前があり、先頭に付ける情報をヘッダ、末尾に付ける情報をトレーラと呼びます。**図9**は、パケットのカプセル化の流れを表しています。パケッ

▼図9　パケットのカプセル化と宛先



トのカプセル化とは、送信するときは上の層から順にデータに対してヘッダやトレーラを付加し、受信時は下の層から順にこれらを取り除いていくことを言います。これでデータがカプセル化される流れはわかりました。では、この制御情報にはどのようなものが含まれているのでしょうか？

MACアドレスとは何か

パケットのヘッダには、Ethernetのほかのパケットとの区切りを知るための情報や、TCPのパケットの順番を示す情報など、さまざまな情報が付加されています。その中の1つに、宛先があります。もちろん、それぞれの層が同じ情報を持っているのではなく、異なる範囲の宛先情報を持っています。では、Ethernetから順に見ていきましょう。Ethernetはコンピュータの物理的な住所を示すMACアドレスを持っています。MACアドレスは、48ビット[注18]長のアドレスです。長さをビットで数えていたことからわかるとおり、このアドレスは0と1で表現されます。しかし、この値を人が覚えるのは困難です。そこで、人が見るときは8ビットに区切った数列を16進数へ変換し、コロンで繋いだ値を使っています。

IPアドレスは論理的な住所

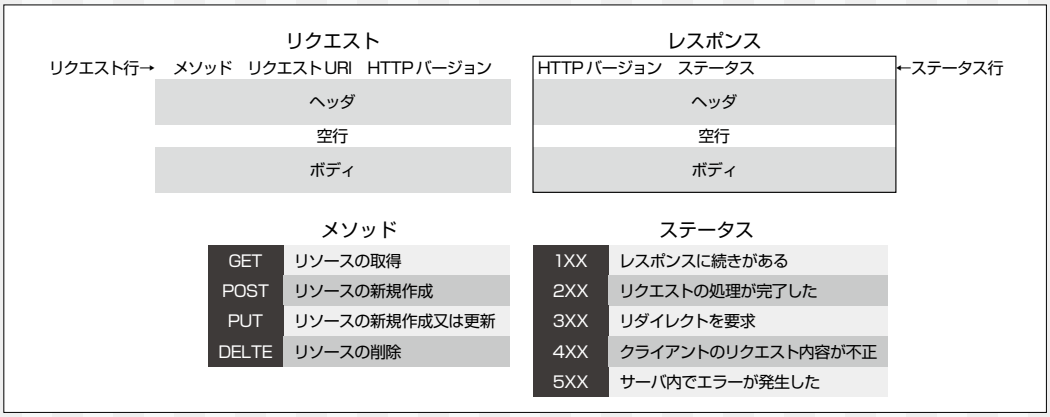
次はIPアドレスです。先ほどは物理的なアドレスでしたが、IPアドレスはコンピュータの論理的な住所を示します。このIPアドレスには2つのバージョンがあります。もともとは、32ビットの長さでアドレスを表現するIPv4だけが使われていましたが、インターネットの発展にともない42億9,496万7,296個[注19]のIPアドレスだけでは足りなくなってしまいました。そこで、この問題を解決するために考えられたプロトコルがIPv6です。IPv6は、IPアドレスの長さが128ビットで表現されているため、より多くのアドレスを割り当てることができます。IPv4だと2の32乗個のアドレスが存在していると説明しましたが、そのすべてがインターネットから接続できるアドレスではありません。IPアドレスには、大まかにインターネット上でユニークなアドレスであるグローバルアドレス[注20]と、インターネット上に公開する必要のない端末に割り振るローカルネットワーク用のプライベートアドレスの2つに分類できます。また、IPアドレスもMACアドレス同様、人に優しい表現方法があります。IPv4では8ビットずつ区切った数列を10進数に変換し、これをドットでつなぎ合わせます。IPv6では16ビッ

注18）1ビットはコンピュータが扱うデータの最小単位で、2進数の1桁を表す。

注19）2の32乗。

注20）AS番号と同じくICNNによって管理されている。

▼図10　HTTPのデータ構造



トずつ区切った数列を16進数に変換し、コロンでつなぎ合わせる方法です。

TCPとポート番号

続いてTCPのポート番号です。これは、コンピュータ内のアプリケーションを識別する番号のことを言います。全部で65,536個ありますが、0から1023番はウェルノウンポート番号[注21]と呼ばれ特定のアプリケーションが割り振られています。ちなみに、HTTPには80番ポートが指定されています。しかし、これ以外の番号を使えないわけではありません。8080でも9090でも好きなポート番号でWebアプリケーションを立ち上げることができます。ではなぜ推奨ポートが決まっているのでしょうか?

あるサーバのWebサイトにアクセスしたい場合を考えてみると、サーバの多くは80番に割り当てられているので、80番ポートにアクセスすれば閲覧できる可能性が高いわけです。しかし、これがバラバラの値だとするとどの番号にアクセスすべきかわからなくなってしまいます。このように、アプリケーションを公開したり、アクセスしたりするのに便利なため、推奨ポートが決まっているのです。

最後は、リクエストURIです。これはWebサーバの中のリソースを指定するためのもので す。**図9**のようにパスだけを指定したり、URIをフルで指定することもできます。


MACアドレスに、IPアドレス、ポート番号、リクエストURIと……あれ? URLがない?


URLは、通信の宛先として直接使われないんだよ。どこで使われているかというと、事前にURLからIPアドレスに変換したものを宛先として使っているんだ。ちょうど良いのでそもそもURLとは何か、そしてアドレス変換に使われているDNSについては、本特集の第5章で勉強しようね。

もう一度パケットの中身を読もう


なるほど。HTTPヘッダのリクエストURIでページが指定できるようになっていたんですね。そういえば、Wiresharkでパケットを見たときには、ほかにもいろいろな情報が出てきましたが、あれは何だったんでしょうか?


HTTPの仕様について説明したあと、もう一度パケットを見てみよう。

HTTPのデータ構造

HTTPとは、Hyper Text Transfer Protocol

注21) URL https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

の略でWebサーバとクライアントがHyper Text Markup Language（HTML）や付随する画像や音声などのデータをやりとりするための通信プロトコルです。HTTP/1.1[注22]の仕様はRFC 2616によって定義されています。**図10**は、HTTPリクエストとレスポンスのデータ構造を示しています。1行目の内容は違いますが、そのあとはヘッダと空行、ボディと、どちらも4つの区画に分かれていることがわかると思います。では、最初の1行目を中心に見ていきましょう。まずはリクエスト行です。メソッドのあと、スペースを開けてリクエストURI、HTTPのバージョンが続いています。メソッドはクライアントからサーバに対して求める処理の内容のことで、よく使われるGETやPOST、PUT、DELETEを**図10**にまとめています。ステータス行はHTTPバージョンのあとスペースを挟んでステータスが続いています。ステータスコードは100桁ごとに同じような意味を持つグループに分かれています。先頭行以外にもヘッダやボディがありますが、ここでは紹介しきれないのでぜひRFC 2616を読んでみてください。

HTTPのストリーム

それでは、復習をかねてもう一度WiresharkでHTTPパケットを読んでみましょう。まずは前回と同じようにフィルタ機能を使ってHTTPのパケットだけを表示します。次に解析したいパケットを選択した状態でメニューバーの［Analyze］→［Follow］→［HTTP Stream］をクリックすると、**図11**のような画面が表示されると思います。ここでは、解析済みのHTTPパケットをまとめて見ることができます。さっそくリクエスト行を見てみましょう。メソッドはGETで/desctiption.xmlが指定されていることから、サーバに対して/description.xmlの情報取得を要求していることがわかります。今

▼図11　HTTPのストリーム画面

回はGETリクエストを投げているのでヘッダで終了していますが、POSTリクエストの場合は1行の空行を挟んでボディが続きます。もう1つのレスポンスデータを見てみると、ステータス行のステータスは200 OKになっています。よって、このリクエストは成功していることがわかります。そのあとのヘッダではボディがXMLであることを示すContent-typeなどが定義され、1行の空行のあとにボディの内容が含まれています。

Webサーバのしくみ


前回よりパケットの中身が理解できるようになりました！


それはよかった。では最後に、Linuxのウェブサーバのしくみについて知ろう。

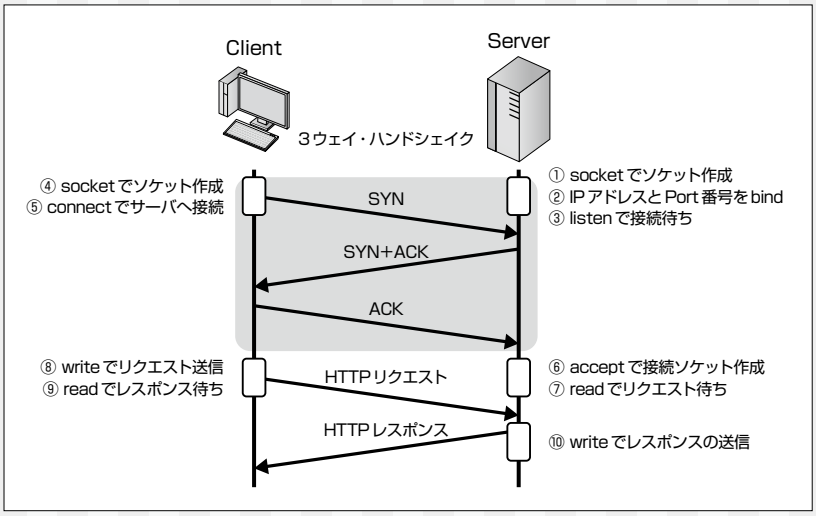
コンテキストスイッチでプロセス切り替え

Webサーバのしくみを理解するには、OSの動きを知る必要があります。まずは、アプリケーションを実行するときの話から始めましょう。

コンピュータ上のアプリケーションは、OSによってプロセスという実体として実行されて

注22）現在では、セッション層の通信を効率化したHTTP/2がRFC 7540としても策定された。

▼図12　HTTP通信の流れとソケットの動き



います。CPU[注23]の1つの処理装置が一度に実行できるのは1つのプロセスだけです。最近では処理装置が1つだけのCPUは珍しい[注24]ですが、そうでない時代にもネットサーフィンしながらエディタを開いたりと、複数のアプリケーションを動かすことができました。いったいなぜでしょうか？　これは、プロセスを短い時間に区切り、区切ったプロセスを次々と切り替えて実行していくことによって、人の目には同時に実行しているように見せているためです。このように実行するプロセスが切り替えられるように実行プロセスの状態を保存したり、次プロセスを読み込むこんだりすることをコンテキストスイッチと言います。

UNIXではすべてファイルで管理する

通信を理解するのにもう1つ重要な概念がファイルです。Linuxでは、通常ファイルだけではなく、ディレクトリやディスプレイやキーボードなどの装置を管理するデバイスファイルなどがファイルとして管理されています。このようにさまざまなものをファイルとして抽象化することによって、統一された操作が可能なのです。ソケットもファイルの1つです。これはプロセス間でデータをやりとりするためのファイルで、コンピュータ通信には欠かせないものです。というのも、このプロセス間というのは同じコンピュータ上でも、インターネットを介した別のコンピュータ上にも対応しているためです。また、作成時に通信の範囲と通信方法を選択できます。通信の範囲は、同じコンピュータ内で通信するためのUNIXドメイン[注25]と、IPv4プロトコルを使うためのIPv4ドメイン、IPv6プロトコルを使うためのIPv6ドメインを指定できます。種類としては、UDPかTCPを指定できます。

HTTPの通信はTCP/IP上でどう行われるのか

一通り用語の説明が終わったところで、HTTPのベースとなるTCP/IP通信を標準ライブラリ[注26]でどのように制御するのかを見ていきましょう。**図12**のように、サーバ側はIPドメインのTCPソケットを作成します。その後IPアドレスとポート番号をbindで紐付け、

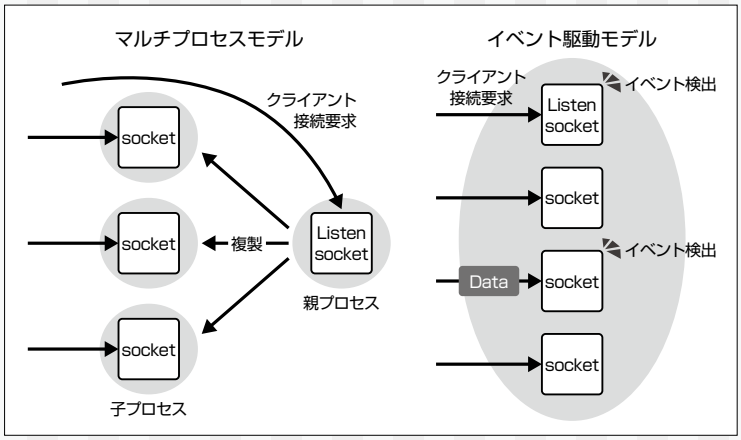
注23) 命令列を順に読み込み、解釈実行するコンピュータの中心的な処理装置のこと。

注24) 最近のCPUは、マルチコアやハイパースレッディングなどによって処理装置を複数持つ。

注25) プロトコルではない。

注26) スタック状の通信プロトコルのところで説明したよく使う機能集のこと。

▼図13　2つのWebサーバアーキテクチャ



listenを実行してリクエストを聞き受ける状態にします。クライアント側もIPドメインのTCPソケットを作成し、サーバのIPアドレスとポートにconnectで接続します。3ウェイ・ハンドシェイクが完了したあとにサーバ側でacceptを実行すると、クライアントと接続した新たなソケットを作成します。そのあとはreadを実行してクライアントからHTTPリクエストが送信されるまで待ち、受信したらそのレスポンスを返します。ここで注意したいのは、readを実行するとクライアントからのリクエストを受信するまで待ってしまうことです。もちろんこの間はほかのクライアントと通信できません。しかし、Webサーバは一度に複数のクライアントと通信する必要があります。いったいどのようにこの問題を解決しているのでしょうか？

この問題を解決する方法として、**図13**のマルチプロセスモデルとイベント駆動モデルについて説明します。マルチプロセスモデルは、その名のとおり複数のプロセスごとにソケットを持ち、プロセスの数だけ並行にクライアントを処理できるようにしたシンプルなモデルです。しかし、プロセスの生成やコンテキストスイッチに時間がかかるというデメリットもあります。そのため、クライアントと接続するたびにプロセスを複製するのではなく、接続可能数まで事前にプロセスを複製[注27]することがあります。対するイベント駆動モデルは、1つのプロセスが複数のソケットを管理するモデルです。readを使って「クライアントからのリクエストを待つ」というではなく、epollなどを使ってI/O処理[注28]を止めずに、クライアントの接続要求やHTTPリクエストというようなイベント検出時だけ処理を行う方法です。これ以外にもスレッドを使う方法やハイブリッド型などいろいろありますが、ベースの考え方は前述の2つとあまり変わりません。

終わりに


最後は初めてのことがいっぱい出てきて難しかったです。でも、Webサーバが大量のリクエストを捌(さば)くための方法がいくつか存在していることがわかりました。


うん、それだけわかっていれば十分だよ。今回は表面的な説明しかできなかったので、気になるところがあれば深掘りしていくと良いかな。


はい、頑張ります！

SD

注27）プレ・フォーク型とも言う。
注28）情報を入力したり、出力したりすることを言う。

第2章 ネットワークコマンドってなんですか? 知っておきたい7つのコマンド

Author 黒崎 優太（くろさき ゆうた） ㈱サイバーエージェント Dynalyst開発チーム
Twitter @kuro_m88

登場人物

渋谷（先輩）
Webアプリのインフラを担当。社歴は浅いけれど見た目はベテラン。実力もあって一目置かれている。

服部（新人）
大学卒業したての新入社員。教科書的な知識はあるけれど、ネットワークコマンドを打ったことなどはない。

ネットワークコマンドを学ぶ理由

先輩、研修用のLinux PCでgihyo.jpにアクセスしようとしたのですが、つながりませんでした。「切り分け」という作業が必要ということは習ったのですが、具体的に何をすれば良いのかわからず……。

どれどれ、僕のマシンだとアクセスできているから、とりあえず名前解決ができているか見てみよう。ほら、やっぱり。DNSの設定を見直してごらん。

本当だ、僕の設定が間違っていたようです。どうやって調べたんですか？

ちょうどいい機会だから、便利なコマンドをいくつか紹介するね。

PCからWebサイトにアクセスできなかったことを新人から相談された先輩は、コマンドを叩いてすぐに問題を特定してしまいました。開発・運用の現場では、ネットワーク経由で操作したり、サービスを提供したりすることが多いでしょうから、ネットワーク系のコマンドを知っておくと理解も深まりますし、トラブルがあったときに非常に役立ちます。もちろん、TCP/IPやDNS、HTTPといった概念を知っていなければ、切り分けるための要素さえわかりませんが、これらを実際に手を動かしながら学んでいくうえでも、ネットワーク系のコマンドはあなたの理解の助けとなってくれるはずです。

先輩は新人のPCを使って、「名前解決」ができるかどうかを調べたようです。名前解決というのは、Webサイト（ここではgihyo.jp）にアクセスするためにはどのIPアドレスに対してリクエストを送れば良いのか、を調べることですが、これには**dig**コマンドを使います（**図1**）。正しく名前解決ができていれば、ANSWER SECTIONという項目にgihyo.jpのIPアドレスが表示されるはずです。

このように、ネットワーク系のコマンドを知っていればサーバやネットワークの状態をすぐに調べられます。本章ではネットワーク系のよく使うコマンドを、その利用事例とともに紹介していきます。コマンドオプションなどについては詳しく説明しませんが、実際に自分で使うときに調べてみると良いと思います。

なお、操作する環境はUbuntu 16.04.2 LTSで統一しています。

▼図1　digコマンドの実行結果

```
$ dig gihyo.jp

; <<>> DiG 9.10.3-P4-Ubuntu <<>> gihyo.jp
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1808
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;gihyo.jp.                      IN      A

;; ANSWER SECTION:
gihyo.jp.               210     IN      A       160.16.113.252
                                gihyo.jpのIPアドレスが表示される↑
;; Query time: 36 msec
;; SERVER: 10.84.149.1#53(10.84.149.1)
;; WHEN: Sun Mar 19 05:41:43 UTC 2017
;; MSG SIZE  rcvd: 53
```

自分のIPアドレスを知るには?

まずは自分のIPアドレスを表示してみようか。

それなら知っています！　**ipconfig**ですよね？

惜しい！　Windowsだとそうなんだけど、Linuxだと**ifconfig**なんだよ。

ネットワークインターフェースの状態を調べるにはifconfigコマンドが便利です。ifconfigコマンドを打って表示された結果が**図2**です。

この出力結果の見方を説明していきます。eth0（①）とlo（④）と書いてあるのがネットワークインターフェース名です。この場合、eth0が、操作しているマシンのネットワークインターフェースです。ネットワークインターフェースが複数個存在する場合は、ほかにもeth1、eth2……などと表示されているかもしれません[注1]。

loというインターフェースは「ループバックインターフェース」という特別なインターフェースで、自分自身と通信するために利用されます。

eth0の項目を見ていくと、inet addr（②）とinet6 addr（③）と書かれた項目があります。ここに書かれているのがこのマシンのIPアドレスです。このマシンではIPv4アドレスとIPv6アドレスが設定されていて、IPv4アドレスは10.84.149.185で、IPv6アドレスはfe80::216:3eff:fed3:e9c5とfd62:2320:ef1b:6ec6:216:3eff:fed3:e9c5の2つが設定されているようです[注2]。

ほかにも、インターフェースごとの統計情報（パケット数や通信量など）も表示されています。また、コマンドオプションを付加することで、情報を表示するだけではなく設定を変更することもできます。

注1） OSのバージョンやマシンが動いている環境によっては、eno1、ens1、enp1s1などといった命名規則の場合もある。
URL https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Understanding_the_Predictable_Network_Interface_Device_Names.html

注2） IPv6には、IPv4にはない概念があり、この例では1つ目がリンクローカルアドレス、2つ目がユニークローカルアドレスとして設定されている。本章ではIPv6について解説しないが、気になる人は調べてみよう。

▼図2　ifconfigコマンドの実行結果

```
$ ifconfig
①eth0      Link encap:Ethernet  HWaddr 00:16:3e:d3:e9:c5
②          inet addr:10.84.149.185  Bcast:10.84.149.255  Mask:255.255.255.0
③          inet6 addr: fe80::216:3eff:fed3:e9c5/64 Scope:Link
           inet6 addr: fd62:2320:ef1b:6ec6:216:3eff:fed3:e9c5/64 Scope:Global
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:341842 errors:0 dropped:0 overruns:0 frame:0
           TX packets:173954 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:517012402 (517.0 MB)  TX bytes:11559953 (11.5 MB)

④lo        Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
           RX packets:10 errors:0 dropped:0 overruns:0 frame:0
           TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1
           RX bytes:930 (930.0 B)  TX bytes:930 (930.0 B)
```

どうやってパケットが届いているのか調べてみよう

届いているかどうかを知る

宛先のIPアドレス、自分のIPアドレスが調べられるようになったら、今度はその間のネットワークについて知りたくならない？

パケットの通り道ってことですよね？気になります！

では基本中の基本、**ping**コマンドから！

pingコマンドは、クライアントが送ったパケットを相手にそのまま返送してもらうだけ、というとてもシンプルなものですが、疎通確認や通信の遅延時間の測定によく使われます。gihyo.jpはpingに対して応答しないようになっているので、代わりにwww.example.comにpingを送ってみます（**図3**）。

-cオプションでpingを送る回数が指定できます。指定しなかった場合はpingを送り続けるので、何回かpingが送られたことが確認できたら Ctrl +c で中断しましょう。こちらが送ったpingに対して応答が返ってきたことがわかりますね。

time=に書かれている数字は、パケットが往復するのにかかった時間を表していて、RTT（ラウンドトリップタイム）と言います。使っているネットワークが不安定だった場合は、pingを送り続けてコマンドの出力の様子を見ていると、ときどきタイムアウトして応答がないという、いわゆる「パケットロス」が見られたり、パケットが帰ってくるまでの時間が急に遅くなったり

▼図3　pingコマンドの実行結果

```
$ ping -c 3 www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=49 time=99.2 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=49 time=117 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=49 time=99.9 ms

--- www.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 99.206/105.695/117.907/8.648 ms
```

▼図4　tracerouteコマンドの実行結果

```
$ traceroute -n www.example.com
traceroute to www.example.com (93.184.216.34), 30 hops max, 60 byte packets
1  10.84.149.1  0.028 ms  0.013 ms  0.010 ms
2  100.64.0.82  1.714 ms  1.747 ms  1.733 ms
3  198.51.100.22  1.893 ms  1.928 ms  2.006 ms
4  203.0.113.71  2.579 ms  2.606 ms  2.641 ms
5  58.138.105.49  3.300 ms 3.260 ms  4.213 ms
6  129.250.198.177  96.283 ms  95.302 ms 99.911 ms
7  129.250.6.157  98.817 ms  101.789 ms  101.944 ms
8   93.184.216.34  99.662 ms 102.743 ms 133.093 ms
```

することがあります。通信が不安定だな、と思うことがあった場合の調査にもpingコマンドは重宝されます。

オプションで指定することによってpingパケットを送信する頻度を高くしたり、pingパケットのサイズを大きくしたりできますが、大きいサイズのパケットを大量に送ることは宛先のサーバの負荷になったり、途中経路のネットワークに攻撃をすることになるなど迷惑をかけてしまうので、必要以上に大量のパケットを送らないように気を付けましょう。

道筋を知る


インターネットっていろんなルータを通って目的地までたどり着くんですよね？


tracerouteコマンドを使うとパケットが通る道筋を知ることができるよ。

パケットがどんな経路を通っているのかを知るために、tracerouteというコマンドがあります。お使いのマシンにtracerouteコマンドが入っていない場合は、インストールしてください[注3]。www.example.comへtracerouteした結果が**図4**です。出力を簡略化するために、**-n**オプションを付加して出力されるIPアドレスの逆引きを行わないようにしています。

セキュリティ上の都合などで、tracerouteするのに必要なパケットがフィルタされていると、「* * *」と表示されてうまく表示されない場合があるので、その場合は宛先のホストやネットワーク環境を変えて試してください。

左端に1〜8の数字が表示されていますが、これは「ホップ数」を表していて、tracerouteコマンドを実行したマシンから宛先に向かって経由したルータの順に番号が振られています。**図4**だと、10.84.149.1を経由して、100.64.0.82、198.51.100.22、203.0.113.71……とたどって、目的の93.184.216.34に到達していることがわかります。8番目はルータではなく宛先ですので、「手元のマシンからwww.example.comまで7ホップで到達できた」と言うことができます。

tracerouteの出力結果に書かれているRTTからなんとなく推測できるかもしれませんが、筆者は東京でtracerouteコマンドを実行し、www.example.comのサーバはアメリカにあるために、100ms前後で宛先のサーバから応答が

COLUMN　tracerouteの結果がすべてではない

インターネット上では、ある地点からある地点まで到達するには複数とおりの道があるといったことが多く、tracerouteコマンドからわかる経路は、その時点で選ばれた経路を見ているにすぎません。また、ここで知ることができた経路は往路だけであり、復路はまた違った経路をたどっていることもあります。復路を確かめるためには相手から自分のネットワークに対してtracerouteをしてもらい、その結果を共有してもらわなければ知ることはできません。

注3）ubuntuなら、apt-get install tracerouteなど。

返ってきたようです。途中の経路のRTTに注目してみると、5ホップ目で数msだったのが6ホップ目で100ms前後と、RTTが大きく違っていることがわかります。きっと、5ホップ目と6ホップ目の間に太平洋を渡る海底ケーブルがあるのでしょう。tracerouteコマンドだけでこんなことまで推測できてしまうのです。

ルータまでの経路を知る


パケットの道筋はわかりましたが、これは誰が決めているんですか?


おもしろいところに気がついたね。「ルーティング」っていう概念を知れば、きっとすんなり理解できるはずだよ。

インターネットはおおざっぱに言うと、ネットワークとネットワークをパケットを転送することによって相互につなぐ「ルータ」という機器によって構成されています。このあたりの話は第3章で解説されますので、まずは今回操作しているマシンがどのようにしてルータまで到達しているのか見てみましょう。

先ほどの**図4**を見ると、パケットがインターネットへ出るまでの経路がわかります。まず最初に、1ホップ目でIPアドレスが10.84.149.1のルータを経由していることがわかります。今回のネットワークでは出口は1ヵ所で、基本的にこのルータにさえパケットを転送すれば、操作したマシンがいるネットワークから外に出られるので、このルータは「デフォルトゲートウェイ」と呼ばれることもあります。確認するために**route**コマンドを使ってみましょう(**図5**)。

この出力結果の2行目がデフォルトゲートウェイの向き先を表しています。デフォルトの宛先(0.0.0.0)に通信するには、ネットワークインターフェースeth0から10.84.149.1に向けてパケッ

CentOS 7ではifconfigが使えない!?

紹介したばかりのifconfig、routeコマンドですが、CentOS 7では標準では使えません。使えるようにするためには、ifconfigコマンドなどが入っている「net-tools」というパッケージをインストールする必要があります。「最初から入っていないなんて不便だな」と思った方もいらっしゃるかと思いますが、変わりに「iproute2」というパッケージが入っていて、ifconfigや**netstat**コマンドの変わりに、**ip**や**ss**といったコマンドが用いられます。

net-toolsは近年、ネットワーク機能追加への対応が不十分になってきたということもあり、新しいiproute2への移行が推奨されています。とはいえユーザ側の実情としては、まだnet-toolsのコマンドのほうが広く知られていて使う機会も多いでしょうから、本章はnet-toolsのコマンドを前提として紹介しています。net-toolsのコマンドとiproute2のコマンドの対応例を**表A**にまとめたので、iproute2のコマンドが使える環境では今のうちから慣れておくと移行しやすいのではないかと思います。

ちなみに、Ubuntu16.04にはnet-toolsもiproute2も両方インストールされていますが、Ubuntu 17.04 minimalからはデフォルトではインストールされない方向で議論が進んでおり[注A]、Ubuntuでは今後net-toolsからiproute2への移行が進んでいくと思われます。

▼表A　net-toolsとiproute2のコマンドの対応一覧例

net-tools	iproute2
arp	ip neighbor
ifconfig	ip address、ip link、ip -statistics address
iptunnel	ip tunnel
iwconfig	iw
nameif	ip link、ifrename
netstat	ss、ip route
route	ip route

注A)「Proposal: removing net-tools from ubuntu-minimal in 17.04」
URL https://lists.ubuntu.com/archives/ubuntu-devel/2017-January/039643.html

▼図5　routeコマンドの実行結果

```
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.84.149.1     0.0.0.0         UG    0      0        0 eth0
10.84.149.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

トを転送するというルールが書かれています。

パケットの流れを眺めてみよう

tracerouteコマンドはどうやってパケットが通る道筋を調べているんですか？

tcpdumpというコマンドを使えばどんなパケットが流れているか見られるから、中身をのぞいてみようか。

パケットキャプチャってやつですね！やってみたいです。

tcpdumpというコマンドをインストールすれば[注4]、パケットキャプチャができます。tcpdumpコマンドを実行してから、先ほど実行したtracerouteコマンドを別の端末からもう一度実行してみました（**図6**）。出力を簡略化するために-nオプションを付加して出力されるIPアドレスの逆引きを行わないようにし、**-i**オプションでネットワークインターフェースeth0に流れるパケットのみキャプチャするようにしています。

出力は左端から、パケットが送受信されたタイムスタンプ、送信元IPアドレス、ポート、送信先IPアドレス、ポート、パケットの概要となっています。この結果から、tracerouteコマンドがどのようにしてパケットの通り道を表示しているのかを探ってみましょう。

まず①で、www.example.comの名前解決をしており、その応答パケットが帰ってきています。

次に②で、宛先である93.184.216.34の適当なポート番号に対して、パケットのTTLを1から1ずつ増やしながらUDPパケットを送信しています。パケットにはTTL（Time To Live）という概念があり、ルータを経由するご

注4）ubuntuなら、apt-get install tcpdumpなど。

▼図6　tcpdumpコマンドの実行結果

```
$ sudo tcpdump -n -i eth0  tcpdumpには管理者権限が必要なのでsudoを前置
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:54:45.621524 IP 10.84.149.185.33812 > 10.84.149.1.53: 47690+ A? www.example.com. (33)   ①
10:54:45.621730 IP 10.84.149.1.53 > 10.84.149.185.33812: 47690 1/0/0 A 93.184.216.34 (49)  ①
10:54:45.623238 IP 10.84.149.185.37237 > 93.184.216.34.33434: UDP, length 32               ②
10:54:45.623368 IP 10.84.149.185.55653 > 93.184.216.34.33435: UDP, length 32               ②
10:54:45.623414 IP 10.84.149.185.46555 > 93.184.216.34.33436: UDP, length 32               ②
……省略……
10:54:45.623982 IP 10.84.149.1 > 10.84.149.185: ICMP time exceeded in-transit, length 68   ③
10:54:45.626080 IP 100.64.0.82 > 10.84.149.185: ICMP time exceeded in-transit, length 68   ③
10:54:45.627080 IP 198.51.100.22 > 10.84.149.185: ICMP time exceeded in-transit, length 36 ③
……省略……
10:54:45.724498 IP 93.184.216.34 > 10.84.149.185: ICMP 93.184.216.34 udp port 33449 ➡
unreachable, length 68                                                                      ④
10:54:45.729116 IP 93.184.216.34 > 10.84.149.185: ICMP 93.184.216.34 udp port 33451 ➡
unreachable, length 68                                                                      ④
10:54:45.730385 IP 93.184.216.34 > 10.84.149.185: ICMP 93.184.216.34 udp port 33454 ➡
unreachable, length 68                                                                      ④
……省略……
```

とにTTLが1ずつ減少し、宛先に到達する前にTTLが0になると③のように途中経路のルータからICMP time exceededというメッセージが送られてきます。TTLを意図的に小さくしたパケットを送ることで途中経路のルータの存在を知ることができる、というしくみです。**図6**ではTTLの値は表示されていませんが、tcpdumpに**-v**オプションを付加することで表示されます。

最後に、④でTTLが十分に大きくなったため、宛先のサーバからudp port xxxxx unreachableというメッセージが返ってくることで、宛先まで到達できたということがわかります。

今回はtracerouteコマンド以外のパケットが飛んでいない環境でパケットキャプチャをしましたが、実際に使う場合は調査したいアプリケーション／宛先以外の通信のパケットが一緒に表示されて、見づらい場合も多いです。そのような場合は、tracerouteコマンドのオプションで送信元、宛先、ポート、プロトコルなどで出力するパケットをフィルタすることができるので、フィルタを組み合わせて目的のパケットだけをうまく絞って表示できるように工夫すると良いかもしれません。

tcpdumpのほかにも、1章で紹介したWiresharkという[注5]GUIで使えるネットワークプロトコルアナライザがあるので、そちらの利用も併せて検討してみると良いでしょう。

curlでピザを注文してみよう!

便利なコマンドがいろいろあるんですね。ところで少しお腹が空いてきました。

ピザでも注文しようか。そうだ、**curl**というHTTPなどのリクエストを送信できるコマンドがあるからブラウザを使わずに注文してみてね。それができたらおごってあげるよ。

が、がんばります……。

curlコマンドはHTTP、HTTPS、FTP、FTPS、SCP、IMAP、POP3をはじめ、さまざまなプロトコルをサポートしている便利なコマンドで、オプションもかなり豊富です。サーバなどブラウザが使えない環境でファイルをダウンロードしたり、Webアプリケーション開発でHTTPのリクエスト／レスポンスの内容を詳しく見たりするときによく使います。もちろん、WebアプリケーションやHTTPのしくみを理解するうえでもたいへん役に立つコマンドです。

curlコマンドでgihyo.jpにアクセスしてみましょう(**図7**)。**-v**オプションを使うとこちら側(クライアント)が送ったリクエストと、それに対するサーバの応答がすべて表示されます。「>」で始まる行がこちらから送信したHTTPリクエストのヘッダで、「<」で始まる行はレスポンスのHTTPヘッダです。<!DOCTYPE html>からはHTTPのレスポンスの内容(HTML)が続いています。

よく使うオプションは**-O**(大文字)で、レスポンスを標準出力に出力する代わりにファイルに保存してくれます。ブラウザが使えないサーバ上でファイルをダウンロードするときに便利です。**図8**のようにコマンドを打つと、gihyojp_logo.pngというファイルがダウンロードされるはずです。

-oオプション(小文字)を使うと、保存するときにファイル名を指定できます。命名規則を指定して連番でファイルをダウンロードすることもできます。**図9**とすると、image-1.jpg〜image-10.jpgまでの10個のファイルが連番で取得されます。規則的に命名されているファイルを一括でダウンロードしたい場合に便利そうですね。

curlはHTTPのPOSTリクエストを送信することもできて、**-X POST**というオプションを使うことでPOSTリクエストになります。

注5) URL https://www.wireshark.org

同時に-Fオプションを使うと、ローカルにあるファイルをアップロードしたり、--dataオプションを使うと、Webページのフォームに値を入力して送信することと同じことができたりします。

今回、新人は先輩からブラウザを使わずにWebサイトからピザを注文してみようという無茶振り(?)を受けましたが、curlコマンドを駆使して、

①ピザの注文サイトのHTMLを取得
②HTMLの内容を読んでピザの注文フォームのURLを取得
③ピザの注文フォームの内容を解読
④フォームに入力するのに必要なパラメータ(氏名、届け先住所、商品名など)をセットして送信

とすれば、その無茶振りに応えることができるでしょう。

一度手順が確立できれば、curlコマンドとそのほかのシェルコマンドを組み合わせて、コマンドを1回叩くだけでピザを自動で注文できるシェルスクリプトが作れてしまうかもしれませんね(笑)。SD

▼図7 curlコマンドの実行結果

```
$ curl -v gihyo.jp
* Rebuilt URL to: gihyo.jp/
*   Trying 160.16.113.252...
* Connected to gihyo.jp (160.16.113.252) port 80 (#0)
> GET / HTTP/1.1
> Host: gihyo.jp
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Sun, 19 Mar 2017 08:41:26 GMT
< Content-Type: text/html; charset=UTF-8
< X-FRAME-OPTIONS: SAMEORIGIN
< Set-Cookie: SN4dc8937382ea6=BFezfNaMKrWSrTsY1f%2Cpi9TDI2a; path=/; HttpOnly
< Set-Cookie: ac89ff871769306caui02875976dfa1c7f1b50c0=0; expires=Sun, 19-Mar-2017 ➡
09:41:25 GMT; Max-Age=3600; path=/
<
<!DOCTYPE html>
<html xmlns:og="http://opengraphprotocol.org/schema/" xmlns:fb="http://www.facebook.com/➡
2008/fbml" xml:lang="ja" lang="ja
">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>トップページ | gihyo.jp … 技術評論社</title>
<meta name="description" content="技術評論社提供のIT関連コンテンツサイト" />
……省略……
```

▼図8 -Oオプションで、画像をダウンロード

```
$ curl -O "http://image.gihyo.co.jp/assets/templates/gihyojp2007/image/gihyojp_logo.png"
```

▼図9 -oオプションで、連番の画像をダウンロード

```
$ curl -o "http://www.example.com/image-[1-10].png"
```

第3章 僕もルーティングできたほうがいいですか? 概念を押さえて、実環境での設定へ

Author 中西 建登(なかにし けんと) 電気通信大学
Twitter @whywaita

登場人物

立花(先輩)
教育熱心で、頼りがいのある先輩エンジニア。社内システムの運用を務める。

中田(新人)
大学卒業したての新入社員。IPアドレスやルーティングなど、ネットワークのしくみは現在勉強中。

サーバへの疎通が取れない!?

新人。我が社には検証環境として、共有の仮想マシンホストサーバがある。もし仮想マシンでスペックが足りなくなってきたら、そこから新しくサーバを立ててみるといい。このURLで表示されるWeb画面から作成できるからな。

わかりました、さっそく作ってみます! あれ? アクセスできません……。

ん? ああ、そういえば新人は検証用のネットワークに接続していたな。それだと、ネットワークが違うから接続できないんだ。

世の中には、多くの「ネットワーク」と呼ばれるものが存在しています。大きいものであれば国や企業が管理するものもありますし、小さいものであればインターネットがつながっている家庭にもネットワークがあるでしょう。

みなさんがインターネットを使うとき、必ずどこかのネットワークに所属しています。スマートフォン、ラップトップ、サーバ、どれを扱う場合にも、この事実は変わりません。そして、それぞれのネットワークが相互に通信するためには、「どのネットワークにどのようなマシンが存在しているか」の道のりを知っている存在が必要です。この存在を「ルータ」と呼び、ルータに道のりを教えてもらうことを「ルーティング」と呼びます。

ルーティングの基礎概念

ネットワークが違うと、通信ができないんですか?

今のままだと、な。とりあえず、今の設定を確認してみよう。

自分のルーティング設定を確認するには、**ip**コマンドを用いて確認できます(**図1**)。このコマンドで表示されるものを、「ルーティングテーブル」と呼びます。その内容から、この

▼図1 ipコマンドを用いてルーティングテーブルを確認する

```
$ ip route show
default via 192.168.100.1 dev eth0
192.168.100.0/24 dev eth0  proto kernel  scope link  src 192.168.100.101
```

Linuxマシンがどのようなルーティングポリシーを持っているかを確認できます。

2行目の表示から、通常は「192.168.100.1」というIPアドレスにルーティングする設定になっていることがわかります。このようなIPアドレスを「デフォルトゲートウェイ」、略して「デフォゲ」と呼んだりします。

3行目の表示から、「192.168.100.0/24」のネットワークにはルータを経由せず、直接接続されていることがわかります。この「/24」はサブネットマスクと呼ばれる表記で、ネットワークの区切りを指定する際に用いられるものです。24という数字は、24ビットであることを表しています。

1章にて、IPアドレスは32ビット（8ビットの数値4つを.で区切ったもの）で表されると説明しましたが、「192.168.100.0/24」というIPアドレスレンジ注1においては、先頭24ビットを「ネットワーク部」、後半の8ビットを「ホスト部」と呼びます。ネットワーク部が同じであれば、同じ1つのネットワークに所属していることになります。今回のネットワークですと、「192.168.100.0〜192.168.100.255」までのIPアドレスが、1つのネットワークに所属していることになります。

「192.168.100.0」は「ネットワークアドレス」、「192.168.100.255」は「ブロードキャストアドレス」と呼ばれるIPアドレスとなるため、このネットワークでは、実際には254個のIPアドレスが利用できます。

サブネットマスクはネットワークを設計する際に、どのぐらいの数のIPアドレスが必要かを考えて設定します。1つのネットワークに200台ほど接続する可能性があるのなら「/24」に設定しますし、（実際にはあまり存在しませんが）65,000台ほど接続したいのであれば「/16」のサブネットマスクを設定し、ネットワークを区切ります。

注1）IPアドレスの範囲のこと。

ルーティングは誰がするの?


あれ？　すでに設定されてますね……。僕、設定した記憶がないんですが？


それはDHCPによって自動で設定されたからだな。


自動で？


そうだ。ルーティングの状況などは接続するネットワークによって違うし、それをいちいちネットワークに接続するたびに設定するのは面倒だろう？　だから、ネットワークにつないだら自動的にIPアドレスやデフォルトゲートウェイを設定するDHCPというプロトコルが存

「/31」のネットワークは存在する?

ネットワークの先頭のアドレスはネットワークアドレス、最後のアドレスはブロードキャストアドレスとなり、実際には利用できないIPアドレスであると書きました。では、「/31」のネットワークというのは存在するのでしょうか。

「/31」は1ビットしかホスト部がなく、「192.168.100.0/31」のネットワークは192.168.100.0、192.168.100.1の2つのIPアドレスしか利用できません。ネットワークアドレスとブロードキャストアドレスのことを考えると通信ができないように思えます。実際にサブネットマスクが運用されはじめてからしばらくは通信が行えず、もし1対1の通信を行いたい場合は「/30」のネットワークを利用していました。

ですが、IPアドレスを節約したいという欲求から、「/31」のネットワークを利用できるようにするRFC3021注Aが提案されており、実際に通信できるように実装されている例もいくつか存在しています。

注A）URL https://tools.ietf.org/html/rfc3021

▼図2　tracerouteコマンドを用いて、「gihyo.jp」までのルーティングを確認する

```
$ traceroute gihyo.jp
traceroute to gihyo.jp (160.16.113.252), 30 hops max, 60 byte packets
1  192.168.100.1 (192.168.100.1)  1.824 ms  1.821 ms  1.818 ms ←デフォルトゲートウェイ
2  192.168.1.1 (192.168.1.1)  1.820 ms  1.821 ms  1.818 ms
3  tok2hzbiz1.vectant.ne.jp (163.139.100.83)  4.258 ms  4.552 ms  4.438 ms
4  163.139.152.209 (163.139.152.209)  4.190 ms  6.033 ms  5.547 ms
5  163-139-68-133.rv.vectant.ne.jp (163.139.68.133)  4.241 ms  6.903 ms  4.092 ms
6  ae28.core2.nihonbashi.vectant.ne.jp (163.139.130.157)  4.604 ms
    ae28.core1.nihonbashi.vectant.ne.jp (163.139.130.153)  4.461 ms  5.638 ms
7  ae1.peer2.nihonbashi.vectant.ne.jp (163.139.128.238)  4.345 ms
    163.139.128.234 (163.139.128.234)  4.525 ms
    ae1.peer2.nihonbashi.vectant.ne.jp (163.139.128.238)  7.074 ms
8  as9370.ix.jpix.ad.jp (210.171.224.113)  6.860 ms  5.301 ms  4.733 ms
9  tkgrt1s-ort3.bb.sakura.ad.jp (157.17.130.98)  5.132 ms
    tkwrt1s-ort3-2.bb.sakura.ad.jp (157.17.130.174)  9.090 ms
    tkwrt1s-ort3.bb.sakura.ad.jp (157.17.130.102)  5.503 ms
10  tkgrt1b-wrt1s.bb.sakura.ad.jp (157.17.130.2)  11.908 ms
    tkgrt1b-wrt1s-2.bb.sakura.ad.jp (157.17.130.186)  15.100 ms
    tkgrt1b-wrt1s.bb.sakura.ad.jp (157.17.130.2)  18.327 ms
11  tkgrt20e-grt1b.bb.sakura.ad.jp (157.17.132.82)  5.465 ms
    tkgrt19e-grt2b.bb.sakura.ad.jp (157.17.132.102)  4.946 ms  6.347 ms
……省略……
```

在するんだ。

DHCPはDynamic Host Configuration Protocolの略称であり、このプロトコルを利用することで、IPアドレスを自動的に配布できます。また一般的には、利用できるIPアドレスのほかにも、デフォルトゲートウェイのIPアドレスやDNSサーバのIPアドレスも一緒に配布されます。これにより、自分が接続したいネットワークに参加するだけで、IPアドレスとデフォルトゲートウェイが設定され、ルーティング情報などの情報を知らなくても設定ができます。

DHCPによってデフォルトゲートウェイを知ったマシンは、ほかのルーティング設定がなされていない場合、ほぼすべてのパケットをデフォルトゲートウェイに設定されたルータに送信します。では、デフォルトゲートウェイに設定されているルータはどのようにルーティングするのでしょうか?

ルータには、そのさらに上にルータが存在しており、上位のルータに対してルーティングを行います。上位のルータは、そのまた上位のルータに、その上位のルータはそのまた上位の……と、繰り返し繰り返しルーティングを行います。最終的に、最も上位のルータにたどり着き、通信先のサーバがどのネットワークに存在しているかを調べ、正しくルーティングが行われます。ですが、実際にはすべてのパケット[注2]が最も上位のルータにたどり着いている訳ではありません。世界中のパケットがすべて最上位のルータに到達するとしたら、その量は膨大なものになるでしょう。

パケット量を抑えるために、途中のルータ同士でお互いのルーティング情報を交換し合ったり、すべてのルーティング情報(フルルートと呼びます)を途中のルータが保持したりして、全体のパケット量を抑えるようにしています。

普段何気なく利用しているインターネットも、いくつものルータが相互に正しく設定されていることによって、始めて利用できるようになっているのです。またそれぞれのルータは、自分の上位のルータだけを意識すれば良いため、処理は非常に簡潔です。

実際に、自分のマシンからどのようにルーティングされているかを調べるには、第2章で紹介した**traceroute**コマンドを用います(**図2**)。この結果から、デフォルトゲートウェイであっ

注2) パケットについては本特集第1章を参照のこと。

た「192.168.100.1」の上位ルータとして、「192.168.1.1」というルータが存在していることがわかります。192.168.1.1のルータはさらに上位のルータに問い合わせを行い、これを繰り返しています。このように多くのルータが接続されているのです。

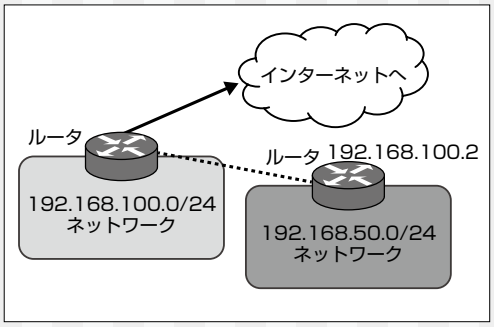
ルーティングを実際に設定する


なるほど。少しルーティングのことがわかってきました。


よし。じゃあ、仮想サーバを立ち上げるためのネットワークに接続するために、ルーティング設定をしてみよう。

図3は新人と先輩が所属する会社のネットワーク図です。現在新人は「192.168.100.0/24」のネットワークに接続していますが、そのルータとは別に「192.168.50.0/24」のネットワークを管理しているルータがあります。仮に、このルータのIPアドレスを「192.168.100.2」とします。この場合、「192.168.50.0/24」のネットワークについては、「192.168.100.2」のルータに聞けば良いはずです。

▼図3　社内のネットワーク図



実際に変更を行う前に、「192.168.100.2」のルータに疎通するのかどうかを確認してみると良いでしょう。ルータやPCなどと疎通するかどうかを確認するには、第2章でも紹介した**ping**コマンドを用います（**図4**）。疎通する場合、**図4**のように「0% packet loss」と表示されるでしょう。もし疎通しない場合は、「100% packet loss」と表示されます。ルータまで疎通するものの、到達するまでに何らかの要因でパケットが欠けてしまった場合は、ほかの値を示すこともあります。

ルーティングの設定を変更する場合は、

▼図4　設定するルータに疎通するのかを確認する

```
$ ping -c 3 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=255 time=0.430 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=255 time=0.569 ms
64 bytes from 192.168.100.2: icmp_seq=3 ttl=255 time=0.561 ms

--- 192.168.100.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.430/0.520/0.569/0.063 ms
```

▼図5　設定ファイル/etc/network/interfaceにルーティング設定を追加する

```
$ cat /etc/network/interface
auto lo
iface lo inet loopback

allow-hotplug eth0
auto eth0
iface eth0 inet dhcp

ルーティングを追加する
up route add -net 192.168.50.0 netmask 255.255.255.0 gw 192.168.100.2
```

Ubuntuであれば設定ファイル**/etc/network/interface**を編集することで変更できます(**図5**)。この最後の行が、ルーティングを追加する設定です。「アドレスが192.168.50.0で、サブネットマスクが255.255.255.0(= /24)であるネットワークのルータは、192.168.100.2である」という設定を入力しています。

設定を入力したあとに、設定を反映させるために再起動を行います。Ubuntuの場合は次のようなコマンドを実行します。

```
$ sudo systemctl restart networking
```

正常に再起動ができた場合、実際にルーティングが増えていることが確認できるはずです。再度ipコマンドを入力し、現在のルーティングを確認してみましょう(**図6**)。「192.168.50.0/24」のルーティングについては、「192.168.100.2」に聞きにいく、という設定が追加されました。

ルーティングのテスト――正しくルーティングしていますか?

やった! 仮想マシンを作成するためのWeb画面が表示できました。

ルーティングを正しく設定することで、うまく通信を行うことができる。ただ、ネットワークを再起動してしまうと、最悪の場合サーバには疎通が取れなくなってしまう。だから、コマンドを使ってルーティングをテストする方法も覚えておこう。

設定ファイルに記載するほかにも、ルーティング設定を変更する方法はいくつか存在します。コマンドを使って実際にルーティング設定を変更してみましょう。これにも**図7**のようなipコマンドを利用します。このようにコマンドを用いてルーティングテーブルを操作しても、設定ファイルを変更したときと同様にルーティングが設定され、パケットは正しくルーティングされるようになります。

コマンドを用いてルーティングテーブルを設定した場合、再起動してしまうとその設定は消失してしまいます。そのため、もし追加したルーティング設定を永続化させたい場合は、設定ファイルにルーティング設定を書くことが必要となります。

設定ファイルに書き込んでネットワークを再起動する場合、ネットワークインターフェースも再起動されるため、どうしても瞬断が発生してしまいます。1回だけならば、そのネットワークを利用するシステムや人に対しての影響は少ないかもしれませんが、何度かトライアンドエラーを繰り返す場合は、影響が大きくなってしまうでしょう。そのため、まずはコマンドを用いて正しいルーティングが行えるのかを確認してから設定ファイルに書き込み、ネットワークを再起動することをお勧めします。

コマンドを用いれば、手軽に追加と削除がで

▼図6 現在のルーティングを再度確認する

```
$ ip route show
default via 192.168.100.1 dev eth0
192.168.50.0/24 via 192.168.100.2 dev eth0
192.168.100.0/24 dev eth0  proto kernel  scope link  src 192.168.100.101
```

▼図7 コマンドを用いてルーティングテーブルを操作する

```
ルーティングの追加
$ ip route add 192.168.50.0/24 dev eth0 via 192.168.100.2

ルーティングの削除
$ ip route del 192.168.50.0/24 dev eth0 via 192.168.100.2
```

▼図8 「zebra.conf」と「ospfd.conf」をコピー

```
$ sudo cp /usr/share/doc/quagga/examples/zebra.conf.sample /etc/quagga/zebra.conf
$ sudo cp /usr/share/doc/quagga/examples/ospfd.conf.sample /etc/quagga/ospfd.conf
```

き、間違っていた場合にも気軽に修正できます。もし誤って、もともと存在しているルーティング設定を変更してしまった場合、サーバの再起動を行えば正しいルーティングテーブルに戻るはずです。

ルーティングを進めるアプリの紹介と応用

ルーティングすごいです！　でも、毎回設定するのは面倒なような……。

た、確かにそうだな（まあ、普通はルータにルーティングが書いてあるんだけどな……）。じゃあ、動的にルーティングを設定できるようにしてみようか。ほかのポートにそういう検証ができるネットワークがあるから、移動してたしかめてみよう。

前述の設定によるルーティングは「静的ルーティング（Static Routing）」と呼ばれます。これに対して、「動的ルーティング（Dynamic Routing）」と呼ばれるものがあります。

動的ルーティングはRIP（Routing Information Protocol）、OSPF（Open Shortest Path First）、BGP（Border Gateway Protocol）などのプロトコルを用いて、複数台のルータやPC間でルーティングを共有し、片方の機器でルーティングに変更が起こった場合に自動で設定変更を通知し、ルーティングテーブルを動的に変更します。

動的ルーティングを設定しておくことにより、ルーティング設定を共有できたり、多くのルータ設定を1台の設定だけで済ませることができるなどの恩恵を得られます。大きなネットワーク間では多くの場合、静的ルーティングは行わず、おもに動的ルーティングを利用します。

今回は、UNIX上で動的なルーティング設定を手軽に行えるQuagga[注3]を用いて、OSPFで動的ルーティングを実現してみましょう。Ubuntuの場合、次のコマンドを用いてインストールすることができます。

```
$ sudo apt install -y quagga
```

今回はQuagga version 0.99.24.1を利用します。Ubuntu系であれば、インストール後に設定ファイルのサンプルが**/usr/share/doc/quagga/examples/**配下にできますので、必要な設定ファイルをコピーしましょう。今回は「zebra.conf」と「ospfd.conf」が該当します（**図8**）。**/etc/quagga/daemons**の設定ファイルを変更し、zebraとospfdを有効化、Quaggaを再起動しましょう。

注3）URL http://www.nongnu.org/quagga/

▼図9 ospfdにログインし、OSPFの設定を行う

```
$ telnet localhost 2604
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.24.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
ospfd> enable
ospfd# configure terminal
ospfd(config)# router ospf
ospfd(config-router)# redistribute connected
ospfd(config-router)# network 192.168.53.100.101/24 area 0
```

▼図10　動的ルーティングが設定されているか確認する

```
$ ip route show
default via 192.168.100.1 dev eth0
192.168.50.0/24 via 192.168.100.2 dev eth0 proto zebra  metric 1
192.168.100.0/24 dev eth0  proto kernel  scope link  src 192.168.100.101
```

```
$ sudo cat /etc/quagga/daemons
……省略……
zebra=yes ←noから変更
……省略……
ospfd=yes ←noから変更
……省略……
$ systemctl restart quagga
```

そのあと、ospfdに接続し、ospfの設定を入力します（**図9**）。**telnet**コマンドの詳細やその接続先の説明については割愛しますので、実際に設定を行う際には各コマンドの動作を調べたうえで設定を行ってください。

同じネットワークにOSPFの設定が行われているルータなどが存在した場合、動的ルーティングの設定が追加されます。先ほども利用したルーティングテーブルを確認するコマンドを用いて、実際にルーティングがOSPFによって伝達されているのかを確認できます（**図10**）。「proto zebra」を含むルーティング設定が追加されていることが確認できます。このルーティングは、OSPFを用いて動的にルーティングが追加されたルーティング設定です。この場合、もしOSPFでつながっているルータなどが、新たにほかのルーティングを追加したりした場合、手元のマシンに手を加えることなく、ルーティングテーブルが動的に変更されます。

もし静的ルーティングで設定を行っていた場合、新しいネットワークが追加されるたびに手元のマシンの設定を変更する必要がありますが、OSPFなどの動的ルーティングを用いて通信していれば、ネットワークが追加された際に動的にルーティングが追加されます。SD

0.0.0.0/0のルーティングはどうなる?

ルーティングを設定する際、ホスト部のビットでルータの行き先を決めるという話を書きました。では、「0.0.0.0/0」というのはどういった行き先を表しているでしょうか?

答えは「すべてのパケットの行き先」です。UNIXにおけるルーティングテーブルでは「default」と表示されることもありますが、「0.0.0.0/0」と表示される実装も存在しています。

これを悪用したツールとして、PoisonTap[注B]と呼ばれるものがあります。PoisonTapをインストールしたデバイスをPCに接続すると、デバイス内で動作しているDHCPサーバが動作し、「0.0.0.0/1」のネットワークからIPアドレスを払い出し、そのルータを自分（接続したデバイス）であると設定させます。「0.0.0.0/1」というのは、DHCPで実際に払い出せる最も大きなネットワークです。この場合、すべてのパケットは接続したデバイスを経由しようとしてしまいます。正しいルータを経由せずにインターネットなどに出ようとしてしまうことで、正常なルーティングが行われないだけでなく、デバイス内に悪質なバックドアなどがしかけられており、デバイスを通過したパケットの中を盗み見たりする設定が行われていた場合、情報流出にもつながる恐れがあります。

もしOSPFなどを用いて動的に設定を行っていたりした場合、その影響は大きくなります。0.0.0.0/0の設定を頻繁に変更することは少ないと思いますが、十分に注意を払ってください。

注B）URL https://samy.pl/poisontap

第4章

LinuxがWindowsサーバに変身? ファイルサーバを立ててみよう!

Author 髙橋 基信(たかはし もとのぶ)
Twitter @damemonyo

登場人物

高緑(先輩)
シブイ感じの木訥な男。服装に頓着がなく作業着を着ている。

宮本(新人)
無理矢理作業着を着ることになった少し天然な雰囲気の新卒女子。

Windowsとファイル共有しよう

うーん、WindowsとLinuxとの間でファイルのやりとりするのって、何かめんどくさいですね。

もともとの生い立ちが違うからなぁ……。よし! Samba(サンバ)でファイル共有できるように設定するか。

えっ? 先輩踊るんですか?

……。

Linuxサーバに関わる業務を行う場合でも、会社で支給されるPCはWindowsで、Windows上で作成したファイルをLinuxサーバにSCPなど[注1]を使ってファイル転送して利用するケースが多いと思います。ただ、ファイル転送を行うためにWindowsにプログラムのインストールが必要となるほか、頻繁にファイルの修正を行うような場合、その都度ファイル転送を行うのは、ともすると誤ったファイルを転送してしまうなどの作業ミスを引き起こしかねません。

Linux上にSamba(サンバ)というソフトウェアをインストールすることで、Linuxとのファイル共有が実現します。Sambaを使うことで、Windowsのエクスプローラなどを使ってLinux上に直接ファイルをコピーしたり、Excelなどから Linux上のファイルを直接編集したりすることが可能になります。

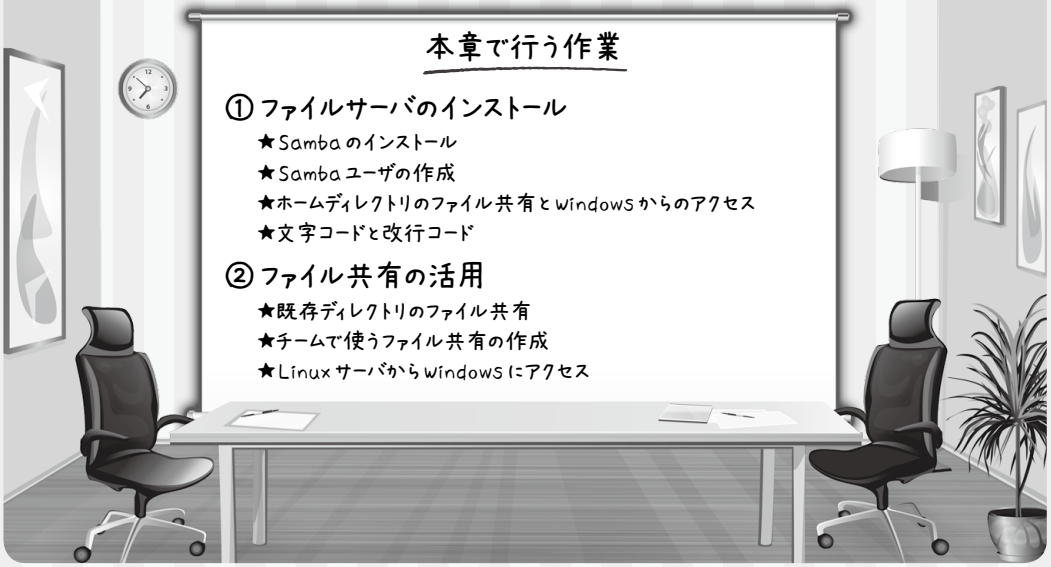
本記事では、Red Hat Enterprise LinuxやCentOSといったRed Hat系ディストリビューション(以下Red Hat系)と、Debian GNU/LinuxやUbuntuといったDebian系ディストリビューション(以下Ubuntu系)を例に、簡単な設定手順を説明します。

SambaとSMB

えっと、ネットで検索したら、LinuxにはNFSっていうファイル共有のしくみがあるみたいなんですが、何でわざわざ「Samba」をインストールするんですか?

ファイル共有にはいくつかのプロトコルがあるんだ。Windowsのファイル共有は

注1) SCPはSecure Copy Protocolの略で、もともとはLinuxを始めとするUNIX系サーバ間で安全にファイル転送を行うためのしくみです。Windows標準ではサポートされていないため、利用するにはWinScpやTeraTermなどのツールを別途インストールする必要があります。
なお、昔ながらのシステムでは、まだFTPという機能を使ってファイル転送しているかもしれません。FTPについてはWindows標準でもftpというコマンドが用意されていますが、使い勝手がよくないため、結局別途ツールをインストールして使っていることが多いでしょう。



SMB[注2]（エスエムビー）というプロトコル、LinuxではNFS（エヌエフエス）というプロトコルが昔から使われてきている……。


そういえば、DropboxやGoogleドライブとか、インターネット上のファイル共有サービスってブラウザからアクセスできるからHTTPですよね。全部HTTPに統一すれば簡単だと思ったんですけど……。


（無邪気に鋭いツッコミをしてくるな。）ええと、SMBやNFSは、最初からファイル共有を目的に作られたプロトコルだから、それぞれWindowsやLinuxのアクセス権や認証といった機能をサポートしているんだ。たとえばNFSを使えば、パーミッションやルート権限[注3]といった、Linux固有の概念もサポートできる。

LinuxとWindowsでは、ファイルの操作やアクセス権について異なる方法で管理が行われていることもあり、ファイル共有についてもNFSとSMBという異なるプロトコル（手順）が標準となっています[注4]。ちなみにMac OS X（macOS）では、最近でこそSMBが標準で使われるようになりましたが、以前はNFS、SMBとも異なるAFPというプロトコルが標準でした。

このように標準が異なるため、橋渡しをするソフトウェアが必要になります。Sambaはまさにこの役割を担うソフトウェアで、Sambaを実行することで、LinuxサーバがSMBプロトコルを使ってWindowsとやりとりできるようになります。

Sambaのインストール


では、さっそくインストールしてみますね。って、インストールだけならyumやaptを使えばいいって教わりましたし、簡単ですよね。


それだけで終わりじゃないぞ、ファイアウォールにSELinux、そういえばIPアドレスは固定にしたか？


あわわ……。

それでは、さっそくSambaをインストール

注2）歴史的経緯でCIFS（シフス）と呼ばれることもあります。またSMBプロトコルのことを「サンバ」と呼ぶ人もいますので、仕事の話で「サンバ」という言葉が出てきたら、SMBとSambaどちらを指しているのか、気を付けるようにしましょう。

注3）パーミッションやルート権限については2017年4月号の第1特集 第3章で詳しく説明しています。

注4）Ubuntu Desktopのように、最近はSMBを標準的に扱えるLinuxディストリビューションも増えてきました。

してみましょう。Red Hat系、Ubuntu系ともに、Sambaはsambaという名前のパッケージになっています。Red Hat系であればrootでログインのうえ、**yum**コマンドを使ってインストールしてください。

```
# yum install samba
```

Ubuntu系であれば**apt-get**コマンドを使ってインストールします[注5]。

```
$ sudo apt-get install samba
```

なお、サーバとして動作させる場合はIPアドレスは固定化したほうが良いでしょう。あらかじめ設定を行っておいてください[注6]。

Ubuntu系であれば、これだけでSambaがインストールされてただちに起動するとともに、システム起動時に自動的に起動する設定も行われます[注7]。一方Red Hat系ではインストールだけが行われ、Sambaの起動もシステム起動時の自動起動の設定も行われません。Red Hat系では次のコマンドを入力して、システム起動時にSambaが自動的に起動する設定を行います。Sambaを構成するサービスはsmbとnmbという2つに分かれているため、各サービスについてコマンドを入力します。

```
# systemctl enable smb
# systemctl enable nmb
```

自動起動を止めたい場合は、enableの部分をdisableに変えてコマンドを入力します。

さらに、Red Hat系ではデフォルトでファイアウォールが有効になっていて、Sambaに必要な通信（ポート）を遮断していますので、次のようにしてポートを開放します。

注5） yumやapt-getコマンドの詳細については2017年4月号の第1特集 第2章で詳しく説明しています。

注6） IPアドレスを固定化する設定の詳細については2017年4月号の第1特集 第4章で詳しく説明しています。

注7） デフォルト設定のままでSambaが起動してしまいますので、セキュリティ的な理由などで好ましくない場合は、後述するSambaの停止コマンドを使ってSambaをただちに停止させてください。また、自動起動を止めたい場合や再開したい場合は、Red Hat系と同じくsystemctl enable（disable）コマンドで制御します。

ルート権限でのコマンド実行

ここで説明する大半のコマンドはルート権限（管理者権限）で実行する必要があります。

Red Hat系やDebianでは、通常rootユーザでログインするか、一般ユーザでログインしたうえでsuコマンドなどでrootユーザにユーザを変更することでこれを行います。

一方Ubuntuでは、デフォルトでrootユーザはログイン禁止となっており、一般ユーザのまま、sudoコマンドに続き、ルート権限を必要とするコマンドを入力することが推奨されています。

このため、本記事でコマンド実行例を示す際、Red Hat系の場合はrootユーザ、Ubuntu系の場合は一般ユーザによるsudoコマンドを使った実行を前提として表記します。またRed Hat系、Ubuntu系に共通したコマンド実行例については、rootユーザによる実行例を示します。

慣用的にrootユーザでコマンドを実行する場合は、

```
#
```

というプロンプト表記で、一般ユーザでコマンドを実行する場合は、

```
$
```

というプロンプト表記を使いますので、本記事でもそれにならっています。

▼図1　psコマンドによるSamba起動の確認

```
# ps -aef| grep mbd
root       985     1  0 10:49 ?        00:00:00 /usr/sbin/nmbd
root      1282     1  0 10:49 ?        00:00:00 /usr/sbin/smbd
root      1288  1282  0 10:49 ?        00:00:00 /usr/sbin/smbd
root      1289  1282  0 10:49 ?        00:00:00 /usr/sbin/smbd
root      1340  1282  0 10:49 ?        00:00:00 /usr/sbin/smbd
root      2199  2180  0 10:55 pts/0    00:00:00 grep --color=auto mbd
```

```
# firewall-cmd --add-service=samba
# firewall-cmd --add-service=samba ➡
--permanent
```

これにより必要なポートが開放され、外部からのアクセスが可能になります[注8]。

システムを起動したままSambaだけ起動させたい場合は、Red Hat系、Ubuntu系ともに次のコマンドを入力します[注9]。

```
# systemctl start nmbd
# systemctl start smbd
```

Red Hat系Linuxの場合、ここまでの設定を順に行っただけではSambaは起動していませんので、Sambaを起動させるにはシステムを再起動するか、このコマンドを入力してください。なお、停止させたい場合は、同様にstartをstopに変えて入力します。

Sambaを構成するプロセスはsmbdおよびnmbdという名称になります[注10]。Sambaを起動させたら、**ps -aef**コマンドや**top**コマンドで、smbdやnmbdプロセスが起動していることを確認してみましょう。**ps**コマンドの実行例を**図1**に示します。

このように、「mbd」という文字列でフィルタすることで、smbdとnmbdだけを抽出することができます。**systemctl**コマンドでSambaを操作するときに、smbとnmbを個別に操作する

注8) 厳密には1行目がただちにポートを開放するコマンド、2行目が再起動後にポートを開放するためのコマンドになります。

注9) Ubuntu 14.04ではsystemctlではなくinitctlコマンド、Debian 7以前ではservice start sambaコマンド、RHEL6およびCentOS 6ではservice start smbコマンドを使います。詳細な説明は割愛します。

注10) このほか、環境や構成によってはwinbinddおよびsambaという名称のプロセスが起動されることもありますが、ここでは扱いませんので説明を割愛します。

SELinuxについて

Red Hat系ではSELinuxというセキュリティ機能の存在も意識する必要があります。本記事ではSELinuxは有効にした状態を前提として説明しますが、厳格なアクセス制御を実現する機能のため、セキュリティが強固になる半面、トラブルの要因になることも多い機能ですので、業務サーバでは無効にすることが多いと思います。SELinuxを無効にするには、次のように/etc/selinux/configファイル中のSELINUX行をdisable[注A]にして再起動してください。

SELinuxの無効化設定

```
……(省略)……
#     disabled - No SELinux policy is loaded.
SELINUX=disabled  ←この行のenableをdisableを変更する
# SELINUXTYPE= can take one of these two values:
……(省略)……
```

注A) permissiveでもかまいません。permissiveはSELinuxによりアクセスが拒否される事態が発生した際に、警告をログに出力するだけでアクセス自体は許可するモードです。

必要があるのは、このsmbdとnmbdの起動や停止が個別制御となっているためです。

Sambaユーザの作成


Sambaもインストールできたし、Windowsからアクセスしてみよっと。


まてまて、まだユーザを作ってないだろ？


え、作ってますけど、ほら……。


いや、Linux上でユーザを作るだけじゃダメで、SambaユーザというSamba独自のユーザも作らないとだめなんだ。

Sambaでは、Windowsからアクセスさせたい各Linuxユーザごとに、対応するSambaユーザを作成し、Linuxユーザとは別にパスワードを設定する必要があります。Linuxユーザのパスワードとは別管理になってしまいますので、ちょっと面倒なところです[注11]。

ユーザの作成や削除は、**図2**のように**pdbedit**コマンドで行います。

ユーザを削除する場合は**-a**オプションの代わりに**-x**オプションを指定します。なお指定するユーザ（ここではmonyo）は、事前に対応するLinuxユーザを作成しておく必要があります。

パスワードはLinuxユーザと同じものにしたほうがわかりやすいでしょう。ただし、Linuxユーザのパスワードとは別管理のため、Linuxユーザのパスワードを変更してもSambaユーザのパスワードは変更されません。逆もまた然りです。

作成済みのSambaユーザのパスワードを変更する場合は**smbpasswd**というコマンドを使います。**passwd**コマンドと同様に、ルート権限があるユーザはユーザ名を指定することで任意のSambaユーザのパスワードを変更できます。一般のユーザは自身のパスワードしか変更できず、またデフォルトでは5文字以下の長さのパスワードは設定できないようになっています。

注11）パスワードを同期させたり、もしくはActive Directory上のユーザのパスワードを参照させたりすることもできるのですが、今回は割愛します。

▼図2　Sambaユーザの作成

```
# pdbedit -a monyo
new password:  ←パスワードを入力
retype new password:  ←パスワードを再度入力
Unix username:        monyo
……（省略）……
```

ホームディレクトリのファイル共有とWindowsからのアクセス


ユーザも作成したし、まずはホームディレクトリのファイル共有にアクセスしてみるか。


あの、ホームディレクトリって……。


あのな、Linuxにログインすると最初に表示されるディレクトリがあるだろ、そこが自分のホームディレクトリってこと。

準備が長くなりましたが、WindowsからSambaが動作しているLinuxサーバにアクセスしてみましょう。ここでは簡単な例として、自分のホームディレクトリをファイル共有して、そこにアクセスする設定をしてみます。

Red Hat系の場合、Sambaの設定的にはデフォルトでホームディレクトリのファイル共有は有効になっていますが、次のコマンドでホームディレクトリの共有をSELinux的に有効化する必要があります。

```
# setsebool samba_enable_home_dirs on
```

Sambaをまだ起動していない場合は、前述したsystemctl startコマンドを使ってSambaを起動してください。

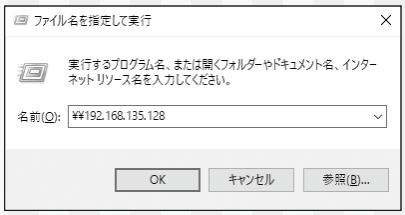
Ubuntu系の場合は、デフォルトでホームディ

レクトリのファイル共有が無効になっていますので、2017年4月号の第1特集で説明したviエディタなどでSambaの設定ファイル/etc/samba/smb.confを**リスト1**のように修正してファイル共有を有効化する必要があります。

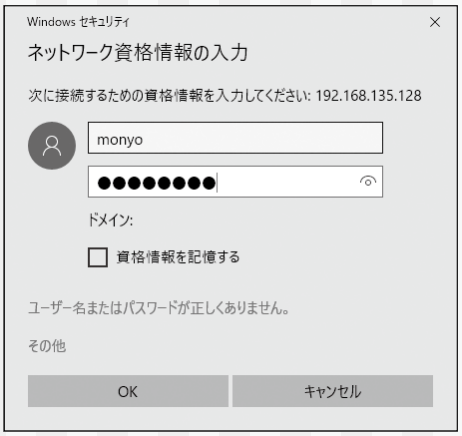
ファイルの修正後、設定を反映させるために前述したsystemctl stopコマンドを使ってSambaを一度停止後、再度起動させます[注12]。

注12) Sambaを再起動せずに設定ファイルを再読み込みさせる方法もありますが、ここでは覚えるコマンドをなるべく少なくするために割愛します。

▼図3　Sambaサーバの IPアドレスを入力



▼図4　ユーザ名の入力



さて、いよいよWindowsからのアクセスです。⊞+Rなどで「ファイル名を指定して実行」ウィンドウを表示させ、そこに**図3**のように「¥¥」に続けてSambaサーバのIPアドレス（ここでは192.168.135.128）を入力してOKを押します[注13]。

うまく設定ができていれば、**図4**のようにユーザ名とパスワードを確認するウィンドウが表示されますので、先ほどpdbeditコマンドで設定したユーザ名とパスワードを入力すると、**図5**のようにユーザ名の共有フォルダが表示されます。さらにクリックすると**図6**のようにホームディレクトリ内のファイルがWindowsから「ふつうに」参照できます。

Windowsからメモ帳などでテキストファイルなどを作成してLinux側で参照してみると、

注13) エクスプローラのアドレスバーなどに入力してもかまいません。また、環境によってはIPアドレスではなく、Sambaサーバのホスト名を入力してもアクセスできますが、トラブル発生時の切り分けが難しいため、本記事は説明を割愛します。

▼図5　ユーザ名の共有フォルダが表示された

▼リスト1　ホームディレクトリのファイル共有を有効化する（Ubuntu系）

```
…… (省略) ……
# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
;[homes]
;   comment = Home Directories
;   browseable = no

# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
;   read only = yes
```

この3行の先頭の;を削除する（ホームディレクトリのファイル共有を有効化）

この行の先頭の;を削除し、yesをnoに変更する（ファイル共有への書き込みを許可）

「とりあえず」書き込んだ内容が反映されていることを確認できます。

文字コードと改行コード


あの、Windowsから作ったファイルなんですが、Linuxから見るとなんか変です！

どれどれ、なるほどこれは文字コードと改行コードの問題だな。正しく設定すれば直るよ。


え、どう設定すればいいんですか？


えっと、うーん（いわれてみると正しい設定ってどうすれば……）。とりあえず英語にしておけ！


あっ、はい！（なんか悪いこと言っちゃったのかなぁ……）

COLUMN 隠し属性とドット（.）ファイル

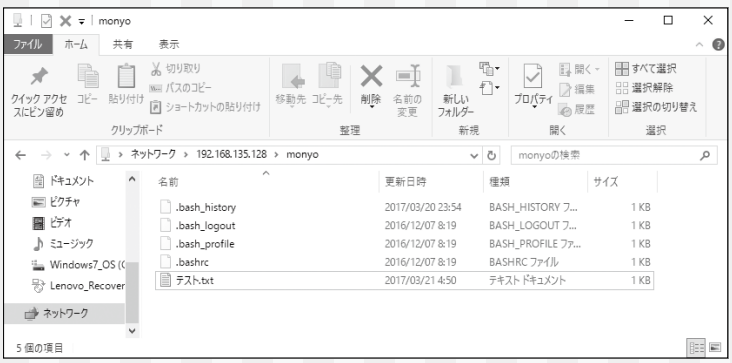
図6では.bashrcなどのファイルが表示されていますが、Sambaのデフォルト設定ではファイル名がドットから始まるファイルは「隠しファイル」という扱いになるため、そのままではエクスプローラで表示されません。**図6**では、**図7**のように「ファイル名拡張子」、「隠しファイル」を表示するように設定を変更しています。

Linuxは標準で多言語に対応していますが、逆にいうとデフォルトでは日本語に特化した設定が行われていないため、日本語を扱う際には注意が必要です。たとえば**図6**で表示されている「テスト.txt」というファイルですが、Linux側の設定が正しくないと、Linux上では**図8**のようにファイル名が「文字化け」して表示されてしまいます。これは日本語の情報を格納する際に使われる「文字コード」にはいくつかの種類があり、格納時に使われた種類を正しく指定しないと日本語の情報をうまく認識できなくなってしまうためです。

Linux上のロケールと呼ばれる設定や、TeraTermなどのツールの設定などを適切に変更すれば表示できるようになりますが、初心者が理解するのは難しいため、基本的にWindowsとLinuxとで共有するファイルのファイル名は英数記号にしておくことをお勧めします。

文字コードはファイルの内容にも影響しますので、Windowsから書き込んだ

▼図6　ホームディレクトリ内のファイルが表示された



▼図7　隠しファイルと拡張子を表示させる設定

テキストファイルをLinux上のviエディタなどで開いた場合にも、同様の問題が発生することがあります。そのため、WindowsでLinux用の設定ファイルを作成してLinuxに転送するといった場合は、指示がない限り、安易に日本語のコメントなどを使わないことをお勧めします。

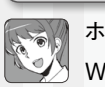
もう1つ面倒な問題として改行コードの問題があります。たとえばviエディタで、

```
test1
test2
```

という内容のファイルをtest.txtという名前で作成し、これをWindowsのメモ帳から参照すると**図9**のように1行につながってしまったように表示されてしまうはずです。

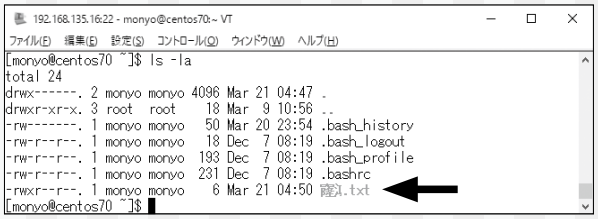
これは、改行コード(「改行」を意味する文字の文字コード)がWindowsとLinuxとで異なっているため、Linuxで作成したファイルの「改行」をWindowsのメモ帳がうまく認識できないのが理由です。Sambaはファイルの内容には関与しませんので、Linux側のエディタなどでWindowsの改行コードでファイルを作成するか、Windows側でLinuxの改行に対応したツールを使う必要があります。Linuxの改行コードに対応したWindowsのツールはいろいろありますので、基本的にはそれらを使うのが良いでしょう[注14]。

既存ディレクトリのファイル共有

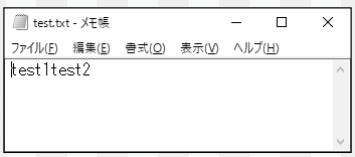


ホームディレクトリをファイル共有して、WindowsとLinuxでファイル共有できるようになって便利になったんですが、できればLinuxの作業ディレクトリを直接ファイル共有し

▼図8　文字化けの例。図6で表示されているディレクトリをロケールがen_US.UTF-8のLinux上で表示したところ。「テスト.txt」というファイル名が文字化けしている



▼図9　Linuxの改行コードを使ったテキストファイルをWindowsのメモ帳で開いた例



てWindowsから見たいんですが……。

よし。じゃあホームディレクトリ以外のディレクトリをファイル共有してみるか。

Sambaではlinux上の任意のディレクトリをファイル共有することができます。たとえば「/var/www/html」を「www」という共有名で読み取り専用で共有するには、Samba的にはSambaの設定ファイル/etc/samba/smb.confの末尾に次のような設定を追加したうえで、Sambaをいったん停止後、再起動させるだけです。

```
任意のディレクトリをファイル共有する
[www]
  path = /var/www/html
```

ファイル共有を作成するには、まずSambaの設定ファイルに[ ]で囲った行を追加します。[ ]で囲った中の文字列はファイル共有名としてWindowsから認識されます。以降の行は、次の[ ]で囲った行に到達するまでの間の行が、そのファイル共有固有の設定としてSambaから認識されます。また先頭が「#」や「;」の行はコメント行となりますので、前述した**リスト1**のように、ファイル共有の説明など、任意の文

注14)「エディタ 改行コード UNIX」といったキーワードで検索すると対応ツールや変換方法など、さまざまな情報が表示されますので、詳しくはそちらを参照してください。

字列を設定することができます。

SELinuxを有効にしている場合は、これに加えてSELinux的にSambaによるファイル共有を許可する必要があります。いくつかの方法がありますが、

```
# setsebool samba_export_all_ro on
```

を実行して、SELinux的にどのディレクトリについても読み取り専用によるファイル共有を許可する設定を行うのがよいでしょう[注15]。

なお、この設定を行っただけではファイル共有に書き込みを行うことはできません。既存のディレクトリを書き込み可能な状態にしてファイル共有を行うには、パーミッションまわりの設定や利用形態に応じた考慮がLinux、Samba双方に必要となり、一筋縄ではいかないため本記事では説明を割愛します。興味のある方はぜひチャレンジしてみてください。

チームで使うファイル共有の作成


よし、ファイルを添付してチームメンバーにメールっ……と。あっファイルが大き過ぎてメールに添付できないです。どうしましょう？


この際だからチーム専用のファイル共有を作ってみるか。Sambaだったら手軽にファイル共有を作れるしな。


えっ、でも書き込み可能なファイル共有って難しいって……。

普通の会社であれば、社内のファイルサーバでファイルを共有する環境が整っていると思います。とはいえ、容量の問題や、開発用にネットワークが分離されているといった事情で社内のファイルサーバを使えないケースも多いと思います。このようなときにSambaを活用することで、簡単にプロジェクト用のファイルサーバを構築することができます。なお、本記事では紹介を割愛しますが、Sambaは本格的なファイルサーバとしての利用を想定した高度な機能を有しています。興味のある方はぜひいろいろ調べてみてください。

ここでは/disk/shareというディレクトリを新規に作成のうえ[注16]、projectという共有名で共有し、proj-rwグループに所属するユーザが読み書き可能、proj-roグループに所属するユーザが読み取り専用な設定を行います。

まずはproj-roとproj-rwグループをたとえば次のようにして作成します。

```
# groupadd proj-ro
# groupadd proj-rw
```

なお、グループについては既存のグループを使ってもかまいません。その場合は以降の記事のグループ名を適宜読み替えてください。

続いてLinux上で次のようにして/disk/shareディレクトリを作成のうえ、proj-rwグループがLinux上で書き込み可能な設定を行います。

```
# mkdir -p /disk/share
# chgrp proj-rw /disk/share
# chmod g+w /disk/share
# ls -l /disk
↑shareディレクトリの設定を確認
……（省略）……
drwxrw-r-x  2 root proj-rw 6 Mar 21 14:43 share
……（省略）……
```

SELinuxを有効にしている場合は、これに加えて次のコマンドを実行して、SELinux的にSamba経由でのディレクトリへのアクセスを許可します。

注15）/var配下のファイルにはSELinux的にさまざまな設定が行われていますので、後述するchconコマンドでファイルの設定を変更してはいけません。

注16）前述したとおり、既存のディレクトリをファイル共有するうえでは利用状況に応じて適切な設定を行う必要がありますので、ここでは新規に作成することを前提としています。

```
# chcon -t samba_share_t /disk/share
```

引き続き、/etc/samba/smb.conf末尾に**リスト2**の設定を付加します。

最後に、プロジェクトのメンバーをユーザとして登録します。user1というユーザを作成のうえ、proj-rwグループのメンバーとする例を次に示しますので、必要なユーザを適宜追加してください。

```
# useradd user1   ←user1ユーザの作成
# usermod -G proj-rw -a user1
↑user1ユーザをproj-rwグループのメンバーに追加
# pdbedit -a user1
↑user1ユーザに対応するSambaユーザを作成
```

必要なユーザを作成したら、Sambaの停止、起動を行うことでSambaの設定は完了です[注17]。

後はWindowsから、作成したユーザのいずれかとしてアクセスすることで、shareという名前のファイル共有が参照でき、ユーザの所属するグループに応じて読み取りもしくは読み書きが可能となっているはずです。

もちろん、Linux上から共有内のファイルを参照したり、場合によっては書き込んだりすることもできます。ただし、Linux上からの書き込みの場合、コラムで説明したパーミッションやファイルの所有グループの設定は行われないため、自分でLinuxのパーミッションや所有グループを適宜設定して対応する必要があるなど注意が必要です[注18]。とくにLinuxを知らない人

注17) ユーザの作成、削除はSambaの起動中に適宜行うことが可能です。

注18) chmod g+sコマンドによりsetgidビットを設定することで、作成したファイルやディレクトリの所有グループを上位ディレクトリのものに設定することは可能ですが、パーミッションについてはumask値を変更するか、都度chmod g+rwコマンドを実行して所有グループに対する書き込みアクセスを許可する必要があります。
また、SELinuxを有効にしている場合、ファイル共有外からファイルを「移動」させる場合は明示的にsamba_share_tラベルを付与する必要があります。これが煩雑な場合は、setsebool samba_export_all_rw onコマンドを実行して、SELinux的にSambaで共有するすべてのファイル共有が読み書き可能となる設定を行うことも可能です。

▼リスト2　プロジェクト用のファイル共有

```
[share]
  path = /disk/share

  valid users = @proj-ro @proj-rw
  write list  = @proj-rw

  force group = proj-rw
  force create mode    = 664
  force directory mode = 775
```

リスト2の設定の詳細

LinuxやSambaの知識がある方向けに、**リスト2**の設定について簡単に説明しておきます。

valid usersはファイル共有にアクセス可能なユーザやグループを指定するパラメータで、write listはファイル共有に書き込み可能なユーザやグループを指定するパラメータです。これらのパラメータの設定により、

- proj-rwグループに所属するユーザはファイル共有に書き込みが可能
- proj-roグループに所属するユーザはファイル共有に読み取りが可能
- そのほかのユーザはファイル共有にアクセスできない

という設定を実現しています。なお、@proj-rwのように名称の先頭に@を付加することで続く名称がグループ名として認識されます。

force groupはファイル共有内に書き込んだファイルの所有グループを強制的に設定するパラメータ で、force create modeとforce directory modeは、各々ファイル共有内に書き込んだファイル、ディレクトリのパーミッションを強制的に設定するパラメータです。これらの設定により、

- ファイルやディレクトリの所有グループが常にproj-rwとなる
- 所有グループに対する書き込みが常に許可される

という設定が行われるため、proj-rwグループに所属するユーザ同士が同じファイルに書き込むことが可能となります。

向けにSambaでファイル共有を提供する場合は、無用のトラブルを避ける意味でLinuxからのファイル書き込みは避けたほうが無難です。

誌面の関係もあり、駆け足での説明となってしまいましたが、**リスト2**の設定はSambaのファイル共有の基本となる設定です。この設定を理解したうえで、Sambaのさまざまな設定を付加していくことで、要件に応じたより実践的なファイル共有の設定が可能です。ぜひチャレンジしてみてください。

(おまけ) Linuxサーバから Windowsにアクセス

うー、先輩！　支店のWindowsサーバのファイル共有にファイルを置きたいんですけど、Linuxからだとどうしようもなくて……。やっぱり支店の人にお願いするしかないでしょうか？

smbclientってコマンドを使えば、Linuxからファイルのアップロードができるぞ。

さすが先輩！　あれっ、そんなコマンドないみたいです？

どれどれ……、なるほど、まずはインストールからだな。

SambaにはWindowsのクライアント機能を司る**smbclient**というコマンドが含まれています。smbclientはRed Hat系の場合はsamba-clientパッケージに、Ubuntu系の場合はsmbclientパッケージに含まれていますので、Samba本体とは別にインストールが必要です。

インストールされていたら、**図10**のように、WindowsサーバのIPアドレスと共有名およびユーザ名を指定することで、Windowsサーバのファイル共有にアクセスすることができます。

図10では192.168.1.14というIPアドレスのWindowsサーバ[注19]のtemp共有フォルダにユーザmonyoとしてアクセスしています。

適切なパスワードを入力することで、ログインに成功して「smb: ¥>」というプロンプトが表示されます。ここでFTPに類似した各種コマンドを入力することによりファイル操作を行うことができます。

さまざまなコマンドがありますが、ディレクトリを移動するための**cd**コマンドと、ファイルの送受信を行う**get**と**put**コマンドが使えれば、最低限の操作は行えるでしょう。なお、**-c**オプションを使うことでファイルの送受信を自動的に行うこともできます。興味のある方は、各種情報を検索してみてください。SD

注19) 適切な名前解決が行われていれば、¥¥win10-pc¥tempのように名前で指定してもかまいません。

▼図10　smbclientコマンドによるWindowsサーバへのアクセス

```
$ smbclient //192.168.1.14/temp -U monyo
Enter monyo's password:  ←Windowsサーバ上のユーザmonyoのパスワード
Domain=[ADDOM1] OS=[Windows 10 Pro 10240] Server=[Windows 10 Pro 6.3]
smb: ¥> dir
  .                                   D        0  Tue Apr  4 01:21:19 2017
  ..                                  D        0  Tue Apr  4 01:21:19 2017
  LOCAL1.TXT                          A        7  Tue Apr  4 01:23:40 2017
  テスト1.TXT                         A        7  Tue Apr  4 01:21:19 2017

                65022 blocks of size 2097152. 58468 blocks available
smb: ¥> get テスト1.TXT  ←ファイルのダウンロード
getting file ¥テスト1.TXT of size 7 as テスト1.TXT (0.1 KiloBytes/sec) (average 0.1 ➡
KiloBytes/sec)
smb: ¥> put samba.txt サンバ.txt  ←ファイルのアップロード
puting file samba.txt as ¥サンバ.txt (0.3 kb/s) (average 0.3 kb/s)
smb: ¥> quit
$ ls  ←ダウンロードしたファイルの確認
テスト1.TXT  samba.txt
```

第5章 DNSって何ですか? 自分のサーバでDNSを設定してみよう!

Author 尾崎 勝義（おざき かつよし）（株）日本レジストリサービス（JPRS）システム部
Author 平林 有理（ひらばやし ゆうり）（株）日本レジストリサービス（JPRS）システム部
Author 久保田 秀（くぼた しゅう）（株）日本レジストリサービス（JPRS）システム部

登場人物

林田（先輩）
サーバ運用部門の先輩社員。サーバの構築から運用保守まで何でもこなす。髭がトレードマーク。

星野（新人）
大学卒業したての新入社員。DNSという言葉は聞いたことがあるが、そのしくみや、どんなサーバがどう動いているかはよくわかっていない。

身近なところで動いているDNS

今日はこれから、DNSについて勉強しよう。

先輩、DNSという言葉は聞いたことがあります。たまにSNSで「DNSが動かない」とか「DNSがおかしい」と言って、困っている人を見かけます。

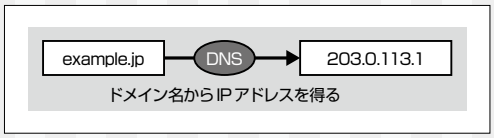
そうだね。そしてそういう人はたいてい「ネットにつながらない」とも言っている。でもどうして、DNSがおかしいとネットにつながらなくなってしまうんだろうか。

第1章で説明したように、インターネットでは通信相手をIPアドレスで指定します。IPアドレスは192.0.2.1や2001:db8::1といった、10進数や16進数で表される番号（数字の羅列）です。

しかし、数字の羅列は人間にとって覚えにくく、使いにくいという問題があります。そのため、インターネットではDNS（Domain Name System）という、名前（ドメイン名）とIPアドレスを対応付けるためのしくみが動いています（**図1**）。

DNSによって、ユーザはIPアドレスに替え、より覚えやすく使いやすいドメイン名で通信相手を指定できるようになります。また、何らかの理由で通信相手のIPアドレスが変更された場合にも、DNSの対応付けを変更することで、同じドメイン名を使い続けることができます。

▼図1　DNSによるドメイン名とIPアドレスの対応付け



ドメイン名はWebサイトのURLや電子メールアドレスなど、インターネットのさまざまなサービスで使われています。そのため、もしDNSがうまく動かないと、それらのサービスすべてに、致命的な影響を及ぼすことになります。

DNSの歴史

DNSがうまく動かないと、いろいろなものに影響が出るんですね。ところでDNSって、いつ頃から使われているんですか？

DNSは、インターネットが研究者の間に広まり始めた、30年ぐらい前からずっと使われ続けているんだ。

DNSって、私が生まれる前から使われているんですか！

驚いたかい？　なぜDNSが作られたかを知るには、その当時のインターネットの状況を、少しだけ勉強する必要がある。DNSが作られたころの状況を、簡単に振り返ってみよう。

名前とIPアドレスを対応づけるためのしくみは、DNSが作られる以前から存在していました。それが「HOSTS.TXT方式」です。HOSTS.TXT方式では**リスト1**のようなIPアドレスと名前の対応表（ファイル）を準備しておき、それによって名前とIPアドレスの対応付けを行います。現在でも多くのオペレーティングシステム（OS）に、HOSTS.TXTに相当するファイルが存在しています[注1]。

DNSが作られる以前は米国のSRI-NICという組織が、インターネット[注2]全体のHOSTS.TXTを集中管理していました。インターネットに接続した各組織はSRI-NICのサーバからHOSTS.TXTを入手し、自分のコンピュータに導入することで、通信相手を名前で指定できるようになりました（**図2**）。

しかし、ネットワークそのものの成長により、

- 多数の情報を、HOSTS.TXTに重複なく登録管理するための手間の増大
- HOSTS.TXTを公開するサーバへのアクセス集中による負荷の増大
- HOSTS.TXTを最新版に更新するための、利用者の負担の増大

などの問題が顕在化し、1980年代にはすでに将来的な破たんが予想されていました。

そのため、これらの問題を解決するための新しい方式が開発されました。それがDNSです。最初のDNSの仕様が1983年に発表されたあと、1987年に改定され、現在のDNSとなっています。

つまり、現在のDNSができてから、今年でちょうど30年を迎えたことになります。DNSにはその後もさまざまな改良が加えられ、現在もインターネットの重要な基盤技術の1つとして運用が続けられています。

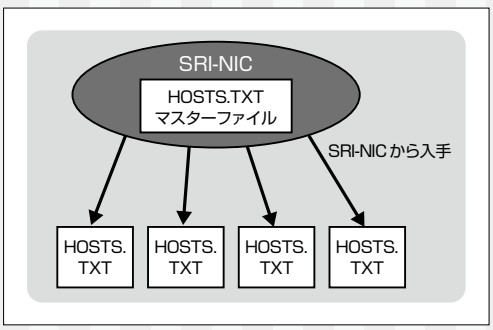
なお、HOSTS.TXT方式とDNSは、いずれもIPアドレスと名前を対応づけるための方式の1つであり、目的は同様であるという点が重要です。

注1）UNIX系のOSであれば**/etc/hosts**、Windowsであれば**%WINDIR%¥System32¥drivers¥etc¥hosts**が存在する。
注2）正確には、インターネットの前身であるARPANET（アーパネット）。

▼リスト1　HOSTS.TXT方式による対応表の例（Linuxの/etc/hostsファイル）

```
198.51.100.1      computer1
2001:db8::1       computer2
```

▼図2　SRI-NICによる集中管理



DNSの構造

自分自身がもっと成長できるようにするため、HOSTS.TXT方式に替わる新しいしくみが必要になったんですね。

そうだね。そして、DNSではHOSTS.TXT方式の問題を解決するため、「ドメイン名」と「権限の委任」という、2つの考え方が導入されたんだ。これがDNSの基本構造を決めているんだよ。

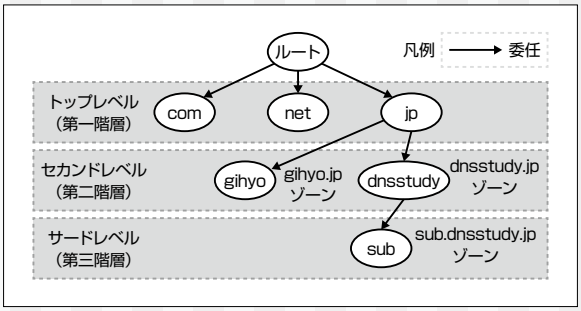
DNSではHOSTS.TXT方式の問題を解決するため、階層的な名前（ドメイン名）と管理権限の委任という、2つの考えが導入されました。

階層的な名前（ドメイン名）の導入

異なった組織で同じ名前が使われないようにするため、名前に階層構造を持たせた、ドメイン名というしくみが導入されました。ドメイン名のしくみでは、名前の起点となるルートの直下に一意な識別子（ラベル）を割り当てます。ここで割り当てられるラベルには、たとえばcomやnet、jpなどがあり、これらをTLD（Top Level Domain）と呼びます。続いて、TLDのラベルの左に組織を表すための一意なラベルをドットでつなげ、それぞれの組織に割り当てます。たとえばgihyo.jpといった具合です。**図3**に表すとおり、このような階層構造をもたせることで組織ごとに一意な名前を作ることができます。

ドメイン名は英数字とハイフンによって表現[注3]され、たとえば、**リスト2**のようなものがあります[注4]。

▼図3　ドメイン名の階層構造



▼リスト2　ドメイン名の例

```
gihyo.jp
3rd.dnsstudy.jp
whois.nic.jprs
xn--lhr645fjve.jp
iana.org
```

管理権限の委任というしくみ

HOSTS.TXT方式では、特定の1組織がすべての名前を集中管理していることが問題となりました。DNSでは、ドメイン名のある階層以下の管理権限を別の組織に任せる（委任する）ことで、この問題を解決します。

DNSでは、各管理組織が管理する単位をゾーンと呼びます。たとえば、jpゾーンは㈱日本レジストリサービス（JPRS）が管理しています。JPRSはjpゾーンに登録されるラベルが一意になるように管理します。

しかし、その配下（サブドメイン）、たとえばgihyo.jpドメインは㈱技術評論社（以下、技術評論社）が管理します。 これは、gihyo.jpゾーンの管理を技術評論社に委任しているためです[注5]。逆に言えば、gihyo.jpゾーンを管理する技術評論社はそのサブドメインとしてsub.gihyo.jpゾーンを、委任元であるJPRSの許可や承認を得ることなく、作ることができます。

DNSではこのような委任のしくみによって、管理の一極集中を防いでいます。

DNSの構成要素と名前解決の流れ

管理権限の委任による分散管理って、会社のしくみとちょっと似てますよね。社内のいろんな部署が協調して動くことで、会社全体がうまく動くみたいな。

いいところに気がついたね。会社のような組織では、それぞれが自分の役割をきちんと果たすことが大事だよね。それはDNSも同じなんだ。これから、DNSにはどんな構成要素があって、実際の名前解決がどのように行われているか、その流れについて説明するよ。

DNSには役割に応じ、名前情報を提供する

注3）現在では英数字以外の文字を用いた国際化ドメイン名（IDN）も使用できる。国際化ドメイン名は、DNSではASCII文字に変換した形で扱われる。

注4）「xn--lhr645fjve.jp」は「総務省.jp」のASCII互換表現。

注5）管理上の理由により、サブドメインを作成するが委任はしないということも可能。たとえば、属性型JPドメイン名ではco.jp、tokyo.jpなどのセカンドレベルドメインは委任せず、jpゾーンで管理している。

権威DNSサーバと、名前解決を実行するフルリゾルバーの2種類のサーバが存在します。権威DNSサーバは、管理対象のゾーンにおけるIPアドレスとドメイン名の対応付けや、自分が委任しているゾーンの委任情報を管理します。

フルリゾルバーは、DNSクライアントからの名前解決要求を受け付け、ルートサーバから順にDNSの階層構造をたどって名前解決を実行し、結果をDNSクライアントに返します。

また、DNSクライアントとフルリゾルバーの間にDNSの問い合わせ・応答を中継する、DNSプロキシが入ることがあります。各家庭に設置されるホームルータの多くは、DNSプロキシの機能を備えています。

フルリゾルバーは名前解決の際に得られた情報をキャッシュすることで、名前解決にかかる応答時間と権威DNSサーバへの問い合わせ回数を減らします。そのため、フルリゾルバーはキャッシュDNSサーバと呼ばれることもあります[注6]。

実際に名前解決が行われる際の流れについて、sub.dnsstudy.jpのIPアドレスを検索する場合で説明します（**図4**）。

① DNSクライアントがフルリゾルバーに対し、sub.dnsstudy.jpのIPアドレスを問い合わせる
② 名前解決要求を受け取ったフルリゾルバーはルートサーバに、sub.dnsstudy.jpのIPアドレスを問い合わせる
③ ルートサーバは、jpについては委任情報だけを知っているため、jpの委任情報（JP DNSサーバの一覧とIPアドレス）を応答する
④ ③の応答を受け取ったフルリゾルバーは、委任先のJP DNSサーバにsub.dnsstudy.jpのIPアドレスを問い合わせる
⑤ JP DNSサーバは、dnsstudy.jpについては委任情報だけを知っているため、dnsstudy.jpの委任情報（権威サーバの一覧とIPアドレス）を応答する
⑥ ⑤の応答を受け取ったフルリゾルバーは、委任先のdnsstudy.jpの権威DNSサーバにsub.dnsstudy.jpのIPアドレスを問い合わせる
⑦ dnsstudy.jpの権威DNSサーバは、sub.dnsstudy.jpについては委任情報だけを知っているため、sub.dnsstudy.jpの委任情報（権威DNSサーバの一覧とIPアドレス）を応答する
⑧ ⑦の応答を受け取ったフルリゾルバーは、委任先のsub.dnsstudy.jpの権威DNSサーバにsub.dnsstudy.jpのIPアドレスを問い合わせる
⑨ sub.dnsstudy.jpの権威DNSサーバは、IPアドレスをフルリゾルバーに応答する

注6）DNSの仕様を定義しているRFC 1034/1035には、キャッシュDNSサーバという記載はない。そのため、本稿でも「フルリゾルバー」を使用している。

DNSに関する用語の不統一について

DNSでは、同一の対象を指しているにもかかわらず複数の用語が存在する場合があります。権威DNSサーバとコンテンツサーバ、フルリゾルバーとキャッシュDNSサーバといった具合です。いくつかの用語を知っておき、それらの用語が何を指し示しているのかということを知っておくことが、DNSを理解するための近道となります。

フルリゾルバーが権威DNSサーバに問い合わせる名前・型は常に同じ

名前解決の流れで説明したとおり、フルリゾルバーが名前解決を行う際には、権威DNSサーバとの間で反復的な問い合わせを行います。この際、フルリゾルバーはDNSクライアントやDNSプロキシから受け取った問い合わせの内容をそのまま使用します。そのため、各問い合わせにおいて権威DNSサーバに送られる名前・型は、常に同一です[注A]。

注A）送信する権威DNSサーバに応じて問い合わせ内容を変化させる方法もあるが、本稿では説明を省略する。

⑩ フルリゾルバーは、名前解決の結果をDNSクライアントに応答する

このような流れにより名前解決が行われ、ドメイン名からIPアドレスを得ることができるのです。

DNSソフトウェア・サービスの紹介


DNSサーバには役割の違うものが2種類あるんですね。ということは、サーバのソフトウェアも2種類必要なんですか?


ソフトウェアを2種類使う場合もあるし、同じソフトウェアが両方の機能を持っている場合もある。最近はDNSをサービスとして提供しているところもあるね。これから、よく使われているDNSソフトウェアやサービスをいくつか紹介しよう。

2種類のDNSサーバ

前節で説明したように、DNSサーバは名前情報を提供する権威DNSサーバと名前解決を実行するフルリゾルバーに大別できます。そのため、DNSの構築、運用にあたってはこれらが提供する機能をきちんと区別することが重要です。この節では権威DNSサーバ、フルリゾルバーについて、どのようなソフトウェア・サービスがあるのかを紹介します。

おもなDNSソフトウェア

表1にオープンソースソフトウェア(OSS)として開発されている著名なDNSサーバソフトウェアの比較表を示します。

BIND

米国のISC(Internet Systems Consortium, Inc.)によって開発されているDNSサーバです。多くのUNIX系OSにおいて標準のDNSソフトウェアとなっており、ルートサーバとしても運用実績があります。BINDはDNSプロトコルのリファレンス実装という位置付けで開発されており、標準化されたDNSの機能の多くを備えているという特徴があります。しかしながら、機能の豊富さに起因する設計の複雑さや、権威DNSサーバとフルリゾルバーを1つのプログラムで兼用しているといった設計上の理由から、しばしば致命的な脆弱性が報告されています。

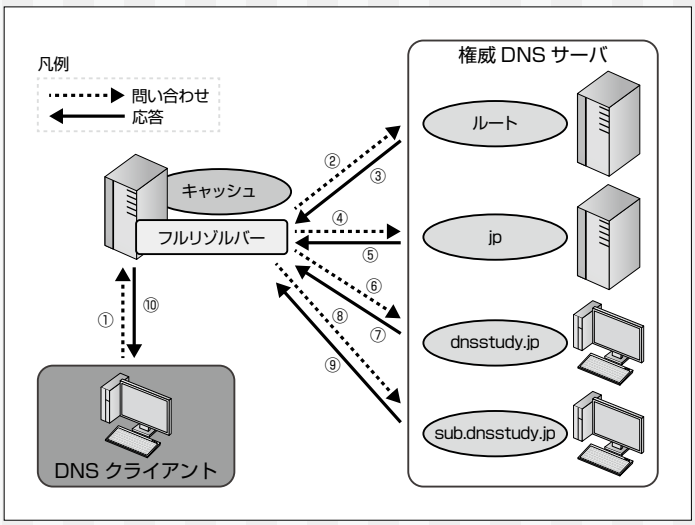
NSD

オランダのNLnet Labsによって開発されている権威DNSサーバです。権威DNSサーバの機能のみが簡潔に実装されており、BINDと比較し、セキュアでパフォーマンスが高いという特徴があります。NSDもBINDと同様、ルートサーバでの運用実績があります。

Unbound

NSDと同じく、NLnet Labsによって開発さ

▼図4 DNSによる名前解決の流れ



れているフルリゾルバーです。Unboundという名前が示すように、BINDの代替とすることを目指して開発されました。DNSSEC検証にも対応しており、フルリゾルバーとして十分な機能を持っています。

PowerDNS Authoritative Server

オランダのPowerDNS.COM BVによって開発が行われている権威DNSサーバです。ゾーンデータをMySQL、PostgreSQLなどのバックエンドのDBに格納する形で運用できるという特徴を持っています。運用実績として.mn、.mp、.tkなどのTLDや、wikipedia.orgの権威DNSサーバなどがあります[注7]。

PowerDNS Recursor

PowerDNS Authoritative Serverと同じく、PowerDNS.COM BVによって開発されているフルリゾルバーです。Luaスクリプトによるフィルタリングや名前解決処理の拡張が可能で、バージョン4.0からDNSSEC検証に対応しています。

注7） https://ds9a.nl/powerdns-denic.pdf

Knot DNS

チェコ（.cz）のccTLDレジストリであるCZ.NIC（CZ.NIC z. s. p. o）によって開発されている権威DNSサーバです。NSDと同様、権威DNSサーバの機能に特化した形で高いパフォーマンスを実現しており、CZ.NICが運用している.czはもちろん、.dk、.clといったTLDでも運用されており、ルートサーバでの運用実績もあります[注8]。

Knot Resolver

Knot DNSと同じくCZ.NICによって開発されており、2016年に正式リリースされたフルリゾルバーです。ソフトウェアのコア部分は小さく、効率を重視した作りとなっていますが、機能拡張のために用意されたAPIで、DNSアプリケーションファイアウォールや、Webインターフェースなど、多くのモダンな機能が実装されています。

注8） https://archive.fosdem.org/2015/schedule/event/knot_dns/attachments/slides/719/export/events/attachments/knot_dns/slides/719/knot_dns_fosdem_2015.pdf

▼表1　DNSサーバソフトウェアの比較（2017年3月31日現在）

		BIND	NSD	Unbound	PowerDNS Authoritative Server	PowerDNS Recursor	Knot DNS	Knot Resolver
開発元		ISC	NLnet Labs		PowerDNS.COM BV		CZ.NIC Labs	
最新バージョン		9.11.0-P3	4.1.15	1.6.1	4.0.3	4.0.4	2.4.2	1.2.4
権威DNSサーバ機能の提供		○	○	—	○	—	○	—
フルリゾルバー機能の提供		○	—	○	—	○	—	○
DNSSEC対応		○	○	○	○	○	○	○
脆弱性の発生頻度								
※CVE識別番号の西暦別件数	2012	8	2	1	1	1	0	—
	2013	4	0	0	0	0	0	—
	2014	5	0	1	0	2	0	—
	2015	9	0	0	4	2	0	—
	2016	12	1	0	8	3	1	0
	2017	1	0	0	0	0	0	0

ソフトウェア・サービスの選択におけるポイント


いろんなDNSソフトウェアがあるんですね。


そうだね。でも実は、オープンソースのDNSソフトウェアの種類が充実してきたのは2000年代以降なんだ。今はいろいろなDNSソフトウェアから選べるようになっているね。

このようにDNSソフトウェアにはさまざまなものがありますが、どのような観点でDNSサーバソフトウェアを選択すべきでしょうか。今回挙げたソフトウェアはどれもDNSサーバとして必要十分な機能を備えており、初学者がDNSを学ぶうえでは、どれを選択しても問題ないといえます。しかし、実際の運用にあたっては、それぞれの運用ケースに合致した機能を備えているかどうかといった機能面での評価と、脆弱性の発生頻度やパフォーマンスといった非機能面での評価が必要です。昨今のハードウェアの性能向上により、通常運用におけるパフォーマンスに関しては多くの場合、あまり問題にならないようになってきています。その一方で、脆弱性対応のコストは無視できるものではなく、システム構成の規模に比例して、リスクと作業工数が膨れ上がります。それぞれのメリット、デメリットをふまえたうえで、適切なDNSソフトウェアを選択したいものです。

DNSサービス


DNSをサービスとして提供しているところも増えてきた。有名なのはGoogle Public DNSかな。8.8.8.8という、Google Public DNSのサービス用IPアドレスを聞いたことがあるいんじゃないかな。あと、Amazon.comがAmazon Web Services（AWS）の1つとして、Route 53というサービスを提供しているね。ちなみに53というのはDNSが使っているポート番号だよ。

――先輩の説明も興が乗ってきたようです。

DNSをクラウドサービスとして提供するベンダも存在します。Webサーバやメールサーバなどと同様、DNSサーバも一度構築したらそれで終わりではなく、継続的にメンテナンスしていく必要があります。DNSは障害発生時の影響が大きく、脆弱性対応や負荷・問い合わせ量のモニタリングなど、日々の運用コストもけっして小さくありません。

このため、権威DNSサーバ機能をサービスとして提供するAmazon Route 53やGoogle Cloud DNSなど、広域分散されたマネージドDNSサービスを利用するといった方法も、運用コストの低減やDDoS攻撃耐性の強化といった観点からは、考慮の対象となり得ると言えるでしょう。

フルリゾルバーについても同様にパブリックDNSと呼ばれるサービスが存在し、Google Public DNS、OpenDNS、Norton ConnectSafeなどがあります。これらは、高度なフルリゾルバーの機能を提供したり、悪意あるWebサイトへの名前解決をフィルタしたり、といった付

COLUMN 多様性（ダイバーシティ）の確保

サービスの可用性向上のため、運用におけるDNSソフトウェア・サービスに、多様性を持たせることを検討するとよいでしょう。複数のDNSソフトウェアやサービスを併用することで、特定のソフトウェアやサービスに脆弱性が発見されたりサービスダウンが発生したりした場合の、利用者に対するサービスの継続性向上が期待できます[注B]。ただし、デメリットとして、設定ファイルや、サービス間の連携部分について考慮が必要となり、構築、運用におけるコスト増が見込まれることは考慮しておく必要があります。

注B）2016年10月に発生したDNSサービスプロバイダ大手の米Dynに対するDDoS攻撃の際、複数のDNSサービスを併用していたWebサイトではサービスを継続できていた旨が報告されている。

加価値を持たせた形で提供されており、いくつかのサービスは無償で利用可能となっています。

BINDによる権威DNSサーバの構築


先輩、講義よりもそろそろサーバ構築がしたいです。


ごめんごめん。ちょっと話が長くなっちゃったかな。じゃあ、これから権威DNSサーバを試しに作ってみよう。


はい！

今回の構築対象

今回の構築対象は社内で使用しているドメイン名である「dnsstudy.jp」のサブドメイン「sub.dnsstudy.jp」を管理する権威DNSサーバとなります[注9]。DNSの構成図内では①にあたるサーバとなります（**図5**）。

また今回の構築では、OSとしてUbuntu Server 16.04.2 LTS、権威DNSサーバとしてBINDを使用することとします。全体的な作業の流れは次のとおりです。

(1) ソフトウェア（BIND）のインストール
(2) sub.dnsstudy.jpのゾーンファイルの作成
(3) BINDの設定ファイルの編集と確認
(4) BINDの起動と動作確認
(5) 親ゾーンからの委任の設定

また、(5) では**図5**の②にあたる権威DNSサーバの設定を実施することになりますが、こちらについては誌面掲載の都合上、一部の設定例を紹介する程度に留めています。

(1) BINDのインストール

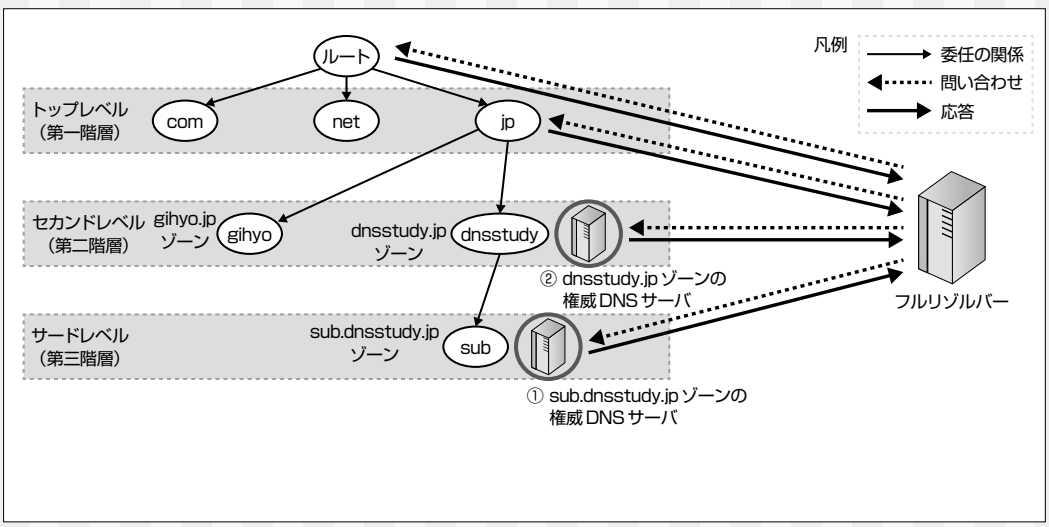
Ubuntu 16.04.2 LTSにおいてBINDはパッケージとして提供されているため、aptコマンドでインストールを実施します。なお、本稿でのコマンドプロンプト表記は、#がroot（スーパーユーザ権限）、$が一般ユーザ権限での実行で表すこととします。

```
# apt -y install bind9
```

原稿執筆時点でインストールされるBINDのバージョンはBIND 9.10.3-P4-Ubuntuです。また、インストールの際にBINDのプロセスが使用するbindユーザとbindグループが自動的

注9）本構築で使用するドメイン名およびIPアドレスは、本稿執筆のためのサンプルである。

▼図5　DNSの構成図



に追加されます。

(2) ゾーンファイルの作成

続いてsub.dnsstudy.jpのゾーンファイルを作成します。今回、作成するsub.dnsstudy.jpゾーンの内容は**表2**のとおりです。このゾーンを管理する権威DNSサーバの情報とゾーンに追加するWebサーバの情報を記載しています。

この設計内容を基に作成したゾーンファイルが**リスト3**のsub.dnsstudy.jpゾーンです。

1行めのTTLはTime To Liveに由来し、フルリゾルバーやDNSクライアントがこのゾーンの各リソースレコードをキャッシュに保持してもよい時間となります。リソースレコードとは、DNSで保持されるそれぞれのデータのことです。リソースレコードには保持する内容によってタイプが指定され、3カラムめのSOA、NS、Aがそのタイプとなります。SOAはStart Of Authorityに由来し、管理権限を持つゾーンの開始を意味します。ns.sub.dnsstudy.jpはそのゾーンのオリジナルデータを管理する権威DNSサーバのホスト名、root.sub.dnsstudy.jpはゾーンの管理者の連絡先(root@sub.dnsstudy.jpの@を.に置き換えた文字列)を示します。NSはName Serverに由来し、そのゾーンを管理する権威DNSサーバのホスト名を指定するリソースレコードです。AはAddressに由来し、ホスト名とそのIPv4アドレスを指定します[注10]。

なお、SOAリソースレコードやAリソースレコードなどでホスト名を指定する際、そのホスト名の末尾に「.(ドット)」を付与しない場合には、相対ドメイン名で記述されたとみなされ、管理対象のゾーンのドメイン名(この場合はsub.dnsstudy.jp)が付与されます。つまり"ns"という表記は"ns.sub.dnsstudy.jp"と解釈されます。一方「.(ドット)」を付与した場合には、そのホスト名は絶対ドメイン名で記述されたとみなされ、このような補完はされません。

前述のsub.dnsstudy.jpゾーンをファイル名「db.sub.dnsstudy.jp」として、適切なパーミッションであるかを確認します。

```
# ls -l /etc/bind/db.sub.dnsstudy.jp
-rw-r----- 1 root bind 238 Mar 10 07:03 /etc/bind/db.sub.dnsstudy.jp
```

(3) 設定ファイルの編集と確認

続いてBINDを権威DNSサーバとして動作させるため、BINDの設定ファイル(named.conf)を編集します。apt経由でBINDをインス

注10) IPv6アドレスを指定する場合は、AAAAリソースレコードを使用する。

▼表2　sub.dnsstudy.jpゾーンの内容

ドメイン名	sub.dnsstudy.jp	
権威DNSサーバ	ns.sub.dnsstudy.jp	203.178.129.30
Webサーバ	www.sub.dnsstudy.jp	203.178.129.30

▼リスト3　sub.dnsstudy.jpゾーン

```
$TTL    86400
@       IN      SOA     ns.sub.dnsstudy.jp. root.sub.dnsstudy.jp. (
                        2017041801      ; シリアル番号
                        10800           ; ゾーンのリフレッシュ間隔(秒)
                        900             ; ゾーンのリフレッシュのリトライ間隔(秒)
                        1814400         ; ゾーンの有効期間(秒)
                        900 )           ; ネガティブキャッシュの維持期間(秒)
;
        IN      NS      ns.sub.dnsstudy.jp.
ns      IN      A       203.178.129.30
www     IN      A       203.178.129.30
```

▼リスト4 /etc/bind/named.confの内容

```
options {
※   listen-on port 53 { 203.178.129.30; };  ←BINDをIPv4アドレス203.178.129.30、53番ポート上で動作させる
     listen-on-v6 { none; };  ←IPv6アドレスではBINDを動作させない
     directory       "/var/cache/bind";  ←BINDが使用するdirectoryを指定する
※   allow-query     { any; };  ←すべてのIPアドレスからの問い合わせを許可する
※   allow-query-cache { none; };  ←すべてのIPアドレスへの、キャッシュ内容の応答を拒否する
※   recursion no;  ←フルリゾルバーの機能を無効化する
※   allow-recursion { none; };  ←すべてのIPアドレスにフルリゾルバー機能を提供しない

     pid-file "/var/run/named/named.pid";
     session-keyfile "/var/run/named/session.key";
};

// リモート制御に関する設定
controls {
     inet 127.0.0.1 port 953 allow { localhost; };
     inet ::1 port 953 allow { localhost; };
};

// sub.dnsstudy.jpゾーンに関する設定
     zone "sub.dnsstudy.jp" IN {
※   type master;  ←マスタの権威DNSサーバとして動作させる
※   file "/etc/bind/db.sub.dnsstudy.jp";  ←ゾーンファイル名を指定する
};
```

トールした場合は設定ファイルがすでに作成されていますので、そちらはいったん退避しておきます。

```
# cp -p /etc/bind/named.conf /etc/bind/named.conf.org
# vi /etc/bind/named.conf
```

編集後のnamed.confの内容を**リスト4**に示します。BINDを権威DNSサーバとして動作させる際に、とくに注意が必要な設定に※印を付けています。不必要なフルリゾルバーの機能を無効化し、外部からのすべての問い合わせに対して応答するように設定することがポイントです。

named.confファイルの編集後、記述内容を確認するためにnamed-checkconfコマンドを実行します[注11]。実行結果にエラーが含まれていなければ編集は完了です。

```
# named-checkconf -z /etc/bind/named.conf
zone sub.dnsstudy.jp/IN: loaded serial 2017041801
```

(4)BINDの起動と動作確認

次のコマンドでBINDを起動します。

```
# systemctl start bind9
```

COLUMN オープンリゾルバーの危険性

適切な管理をされておらず、インターネット上の任意の相手からの名前解決要求を受け付け、処理してしまう状態のDNSサーバをオープンリゾルバーといいます。オープンリゾルバーは放置しておくと、DNS反射攻撃という攻撃手法でDDoS攻撃の踏み台として悪用されてしまう可能性があります。オープンリゾルバーにならないため、フルリゾルバーでは適切なアクセス制限を、アクセス制限を実施できない権威DNSサーバではフルリゾルバーの機能の無効化を実施しておく必要があります。

注11) named-checkconfに-zオプションをつけることで、named.confで指定されたsub.dnsstudy.jpゾーンの内容も確認している。

続いて、**rndc**コマンドを使用してBINDのプロセスの状態を確認します[注12]。**rndc**はローカルホストおよびリモートホストからBINDのプロセスを制御するためのコマンドラインツールです。

```
# rndc status
version: BIND 9.10.3-P4-Ubuntu
<id:ebd72b3>

……(省略)……

server is up and running
```

実行結果として表示された“**server is up and running**”というメッセージで、BINDのプロセスが実行中であることがわかります。

続いて**netstat**コマンドでBINDのnamedプロセスが、指定したIPアドレス(203.178.129.30)のTCPおよびUDPの53番ポートと、ループバックアドレスの953番ポートで動作していることを確認します(**図6**)。

最後にdigコマンドでwww.sub.dnsstudy.jpのAリソースレコードが引けるかを確認します[注13]。

digコマンドの応答内容に正しいAリソースレコードが表示され、**flags**に**"aa" (Authoriative Answer)**が含まれていることを確

注12) rndcがnamedプロセスと安全に通信するための共有鍵の作成・設定が必要になる。BINDをaptでインストールした場合、共有鍵が自動作成・設定される。

注13) +norecは、フルリゾルバーが権威DNSサーバに送信する問い合わせと同じ形式の問い合わせを送信するためのオプション。権威DNSサーバへの動作確認の際には、+norecを付けるとより確実。

▼図6　netstatコマンドでBINDのnamedプロセスを確認

```
# netstat -lnutp
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
tcp        0      0 203.178.129.30:53       0.0.0.0:*               LISTEN      3027/named
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      3027/named
tcp6       0      0 ::1:953                 :::*                    LISTEN      3027/named
udp        0      0 203.178.129.30:53       0.0.0.0:*                           3027/named
```

▼図7　digコマンドの動作確認

```
$ dig @203.178.129.30 www.sub.dnsstudy.jp A +norec

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @203.178.129.30 www.sub.dnsstudy.jp A +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7316
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.sub.dnsstudy.jp.           IN      A

;; ANSWER SECTION:
www.sub.dnsstudy.jp.    86400   IN      A       203.178.129.30

;; AUTHORITY SECTION:
sub.dnsstudy.jp.        86400   IN      NS      ns.sub.dnsstudy.jp.

;; Query time: 0 msec
;; SERVER: 203.178.129.30#53(203.178.129.30)
;; WHEN: Fri Mar 10 11:30:43 UTC 2017
;; MSG SIZE  rcvd: 84
```

認します[注14]。

(5) 親ゾーンからの委任の設定

ここまでの設定で、権威DNSサーバ単体での設定は完了となります。しかし、実際に名前解決ができるようにするためには、追加の作業を実施する必要があります。www.sub.dnsstudy.jpを名前解決する場合、本章の前半部分で解説したとおりルートゾーンからの木構造をたどって今回構築した権威DNSサーバにたどり着けるようにする必要があります。そのため、権威DNSサーバの構築後に親ゾーン(dnsstudy.jp)の権威DNSサーバにサブドメインのNSリソースレコードと、NSリソースレコードに対応するAリソースレコード(もしくはAAAAレコード)を追加する必要があります(委任情報の追加)。**図8**は親ゾーンのゾーンファイルへの追加部分の設定例です。

親ゾーンへの登録を適切に実施することにより、構築した権威DNSサーバが、DNSという分散システムの一部として組み込まれ、実際の名前解決でフルリゾルバーからの問い合わせを受ける状態となります。

また、実際にJPドメイン名などでドメイン名の新規登録を行い、そのドメイン名に対応する権威DNSサーバを構築した場合、その作業内容が少し違ったものになります。この場合はドメイン名の登録代行事業者が提供する手段を使用して、親ゾーンにそのドメイン名のネームサーバ情報を登録することになります。SD

注14) 権威DNSサーバは自分が管理するゾーンを応答する際、応答にAA(権威を持つ応答を示すフラグ)をセットする。

▼図8 親ゾーンのゾーンファイルへの追加

```
……(省略)……
sub.dnsstudy.jp.        IN      NS      ns.sub.dnsstudy.jp.
ns.sub.dnsstudy.jp.     IN      A       203.178.129.30
……(省略)……
```

エンジニアになりたい君へ

森實 敏彦 著
四六判／202ページ
1,400円＋税
幻冬舎メディアコンサルティング
ISBN＝978-4-344-91093-5

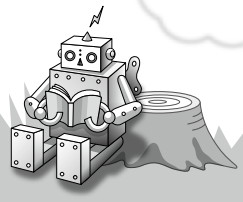
「総合エンジニアリング企業」㈱タマディックの森實敏彦社長の著書。多くの企業とエンジニアを見てきた自身の経験を元に、エンジニアとしての第一歩を踏み出す前に知っておくべきポイントを紹介している。想定しているエンジニアは、機械系、電気･電子系、組み込み系、IT系だが、各論ではなく大きなエンジニアとしての枠で語られている。まずは自分の適性を踏まえたうえで、理想と現実を見極める。大きな会社はエンジニア領域の仕事をアウトソーシングしている場合も多いので、その会社ではどんなエンジニアの仕事があるのか内情を調べて後悔しないように選択する。そして就職後に、エンジニアとして一流になるためにどのように心がければ良いか。などの助言の数々が説かれている。

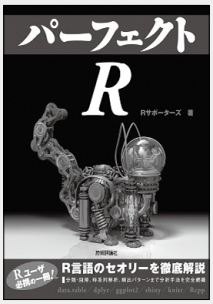
ITエンジニアのためのデータベース再入門

真野 正 著
A5判／200ページ
2,200円＋税
リックテレコム
ISBN＝978-4-86594-025-1

3部構成の本書は、「課題編」でDBMSに起因するトラブル、陥りがちなDBの誤った使い方を示し、「理論編」でリレーショナルモデル、RDBMSのアーキテクチャという基礎理論を押さえ、「解決編」でDB設計とSQLの最適化、運用改善策について解説する構成だ。アンチパターンを示しながらその解決策を探るという構成はよくみられるものだが、本書ではその間に「理論編」をはさむところに特徴がある。また、まえがきに、“「基本さえおさえていれば、こうはならないだろう」と思うケースがたいへん多い”とあり、基礎知識を備えておくことでアンチパターンを避ける、というのがメインテーマであるようだ。タイトルに「再入門」とあるとおり、基本のキから解説している本ではないので、DBや開発全般の前提知識はあったほうが良いだろう。

SD BOOK REVIEW

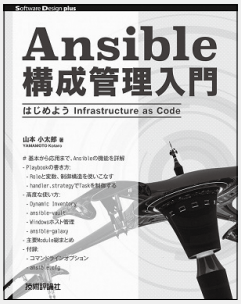




パーフェクトR

Rサポーターズ 著
B5変形判／672ページ
3,600円＋税
技術評論社
ISBN＝978-4-7741-8812-6

本書は統計解析に特化したプログラミング言語Rの解説書である。R言語の仕様をはじめ、データ処理、データ分析、可視化、開発のPartに分かれ、幅広いテーマを扱っている。それぞれのテーマは672ページもあるのでしっかりと解説されている。ほかのプログラミング言語の経験があれば、Rの入門書として問題なく読みこなせるだろう。本書のコードは入門者に向けてベーシックに書かれているかと思いきや、ggplot2、dplyrなどのモダンなパッケージを利用して解説されているのもポイントだ。Rの処理は遅いと言われることが多い。本書の24章で解説されているRcppパッケージを利用すれば、その問題も解消できるだろう。Rcppについての資料はまだ少なく、この部分だけでも本書の価値はありそうだ。



Ansible 構成管理入門

山本 小太郎 著
B5変形判／176ページ
2,480円＋税
技術評論社
ISBN＝978-4-7741-8885-0

ChefやPuppet、Ansibleといった構成管理ツール、クラウドサービス、テスト･デプロイツールを連携させ、継続的インテグレーション／デリバリーを実現することが最近のトレンドである。

本書では第一歩として、Ansibleの使い方をマスターすることを目的としている。Ansibleを手元の環境で動かすためのVirtualBox/Vagrantのインストールと初期設定から始め、Playbookの書き方、おもなモジュールの使い方、複雑な処理を行うPlaybookの作り方などを丹念に解説していく。入門的な内容にとどまらず、Playbookの高速化やWindowsホストの管理方法、モジュールの自作方法など実践的なノウハウも紹介しており、入門を終えた方にも役立つだろう。

第2特集

サイボウズ流 サービス改善につなげるドッグフーディング環境の作り方

Author 岡田 勇樹（おかだ ゆうき）
Twitter @y_okady
サイボウズ㈱

自社製品を社員自ら使って改善点を見つける試みをドッグフーディングと言います。言うは易しですが、ただ使うだけでは「思ったほど改善点が出なかった」なんて結果になります。長年ドッグフーディングに取り組んできたサイボウズは、コストやリスクを最小限にし、効果を最大限に引き出すノウハウを持っています。そのいくつかを紹介しましょう。

はじめに

サイボウズ大阪オフィスで、Webアプリケーションエンジニア兼マネージャーをやっている岡田です。最近は人材マネジメントをおもに担当していますが、以前は「kintone（キントーン）」の開発チームリーダーを担当していました。kintoneは2011年にリリースされたクラウドサービスで、利用者は自分たちに合った業務アプリをノンプログラミングで作成できます。

2013年には、チームが作業を進める際に必要なやりとりを集約するスペース機能や社員一人一人がアイデアを気軽に投稿できるピープル機能などのコミュニケーション機能（**図1**）が強化されました。筆者はこのコミュニケーション機能を開発すべく、2012年にkintone開発チームに参加しました。

サイボウズはkintone以外にも「サイボウズ Office」や「サイボウズ ガルーン」などの製品を提供してお

▼図1　kintoneのコミュニケーション機能

り、どの製品も昔から社内で実際の業務に利用されています。そうすることで、製品開発チームは社内の利用者からたくさんのフィードバックをもらえます。筆者がkintone開発チームに参加したとき、kintoneもほかの製品と同じように社内で業務利用されていました。コミュニケーション機能も同じように社内利用してもらうつもりでしたが、従来の社内利用には問題点もありました。

それは、社内利用開始から本番環境更新までの期間が短く、本番環境更新までに取り込めるフィードバックが少ない点です(**図2-(A)**)。限られた時間で取り込めるフィードバックは、重要度の高い不具合の改修が中心になります。軽微な不具合が改修されないままリリースされ、後に社外の利用者から同じ不具合を指摘されることもしばしばあります。

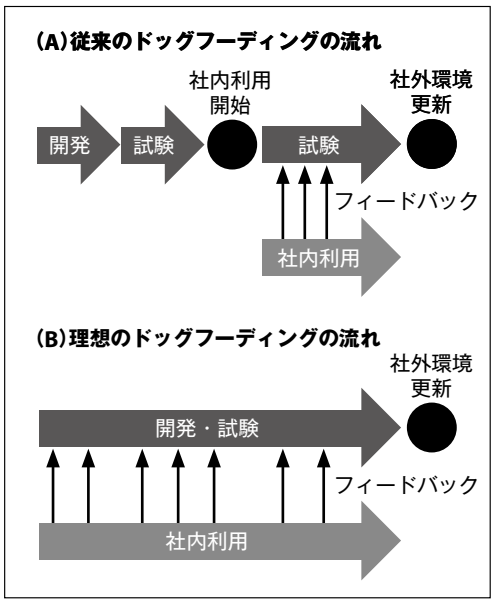
社内利用開始を早めれば多くのフィードバックに対応できます(**図2-(B)**)。しかし、致命的な不具合が存在する製品を業務利用するわけにはいきません。そのため、社内利用開始までに試験を実施して一定の品質を保証する必要があり、どうしても社内利用開始が遅くなってしまうのです。

kintoneのコミュニケーション機能の開発では、品質向上だけでなく機能の進化につながるフィードバックをどんどん取り込める開発プロセスを実現したいと考えました。従来のウォーターフォール型開発から脱却し、アジャイル開発にチャレンジしていかなければという思いもありました。そこで筆者が注力したのが、kintoneのドッグフーディング環境の構築と活用でした。

ドッグフーディングに欠かせない5つのポイント

ドッグフーディングとは、自社製品の改善を目的に社員が日常的に製品を利用することです。不具合や使い勝手の悪い機能などの問題点を社員に発見してもらい、フィードバックと修正を繰り返すことで品質向上やUX(User Experience：ユーザ体験)改善の効果が期待できます。海外では“Eating your own dog food”や“dogfooding”と呼ばれており、自社製品を改善する手法の1つとして広く利用されています。

▼図2　従来のドッグフーディングの流れと理想のドッグフーディングの流れ



ドッグフーディングを効果的に実施して自社製品の改善につなげるためには、ドッグフーディング環境の構築と活用に関する次の5つのポイントが欠かせません。

①製品の最新版を提供し続ける
②多くの社員に使ってもらう
③継続的に使い続けてもらう
④たくさんのフィードバックをもらう
⑤すばやくフィードバックを取り込む

サイボウズの従来の社内利用もドッグフーディングと言えます。実際の業務に利用していたため、多くの社員に継続的に使い続けてもらい(②と③)、たくさんのフィードバックをもらうこと(④)はできていました。しかし、製品の最新版を提供し続けること(①)、すばやくフィードバックを取り込むこと(⑤)ができておらず、製品の十分な改善にはつながっていませんでした。

一方、kintoneのコミュニケーション機能の改善を目指したドッグフーディング環境は、今では製品の改善に欠かせないものとなりました。社員からのフィードバックに対して製品開発チームが改善案を検討し、最短で当日中に製品を修正してドッグフーディング環境を更新できるまでに成長しました。運用開始からしばらくの間はしくみ作りと社員の利用促進に苦労しましたが、しくみが整備されて社員の利用が進めばあとはフィードバックと修正を繰り返すだけです。

以降の節では、ドッグフーディング環境の構築と活用に関する5つのポイントについて、サイボウズでの事例を交えて詳細を述べていきます。

更新作業の自動化で最新版を提供し続けよう

手作業の限界

ドッグフーディング環境を初めて構築する際、サーバの設定やアーカイブの適用を手動で実行する方が多いのではないでしょうか。もちろん初めから自動化に取り組む必要はありませんが、製品の最新版を提供し続けるためには何度も繰り返し実行する必要のある手順ですので、手作業が多ければ多いほど作業工数は積み上がっていきます。

kintoneのドッグフーディング環境も初めは手作業で環境を更新していました。担当者を決めて進めていたのですが、手作業だとミスも起こりますし、担当者が不在だと更新したくてもできないなど、作業工数以外の問題も発生しました。データのバックアップを取っておらず、運用開始から1ヵ月でデータがすべて消えてしまうというトラブルもありました。

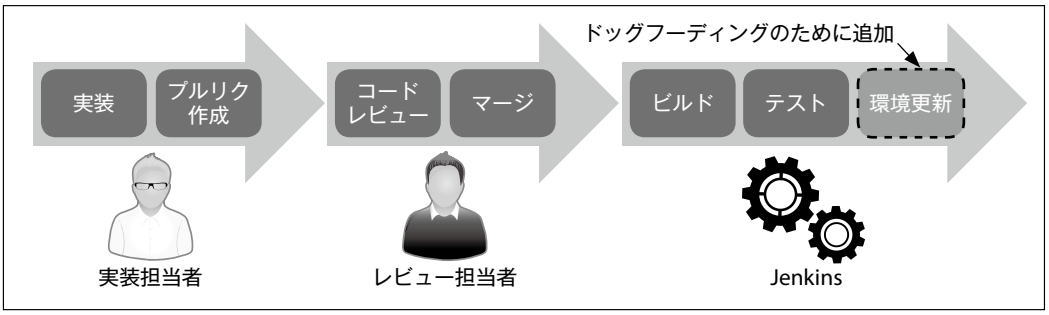
このままでは製品の最新版を提供し続けるのは困難と判断し、自動化に取り組むことにしました。バックアップやアーカイブ更新など、さまざまな作業を自動化しました。まずは手動で実行していたコマンドをスクリプトで自動化することで、手動やcronで簡単に実行できるようになりました。そこまでできるようになるとさらに欲が出てきて、新しい機能が実装されたらすぐにドッグフーディング環境も更新されてほしいと考えるようになりました。そこで取り組んだのが、継続的インテグレーションでのドッグフーディング環境更新です。

継続的インテグレーション

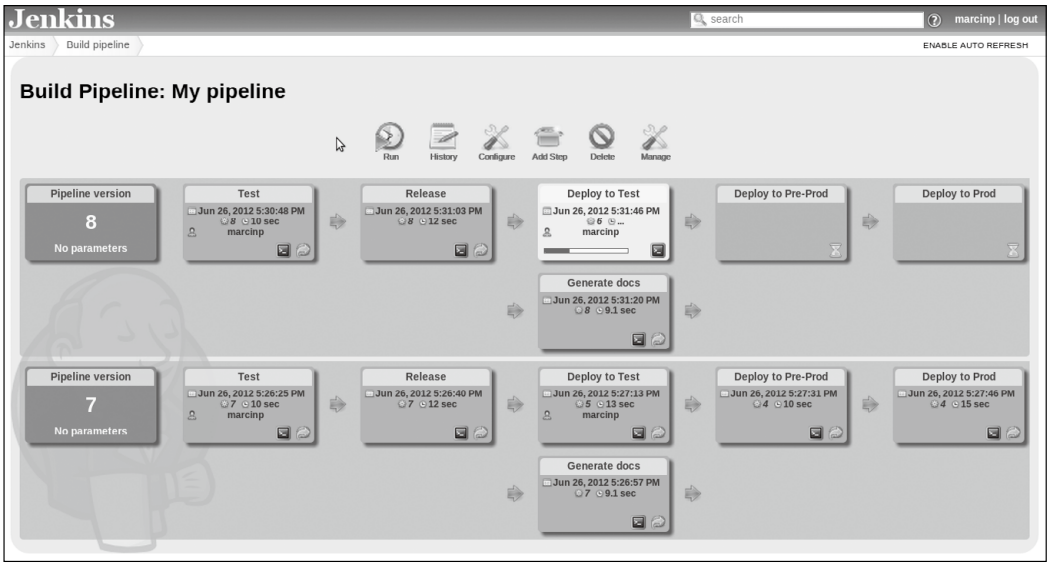
製品開発チームにとって、ビルドやテストの自動化を実現する継続的インテグレーションは欠かせないものとなっています。kintone開発チームでは、もともとGitHubとJenkinsを利用した継続的インテグレーションを導入していました。GitHubのプルリクエストがメインブランチにマージされるとJenkinsのジョブが実行されるしくみとなっており、多いときで1日に10回以上実行されることもあります(**図3**)。

ビルドのジョブが成功したら次はテストのジョブを実行するといった設定は、Jenkins Pipeline Pluginで実現しています。処理がどこまで進ん

▼図3　kintone開発チームの継続的インテグレーション



▼図4　Jenkins Pipeline Plugin



でいるか、どこで失敗したかなどを確認できるため、Jenkinsに慣れていないユーザでも簡単に状況を把握できます(**図4**)。なお、Jenkins 2ではパイプライン機能が標準搭載されています。

kintone開発チームでは、このパイプラインの最後の工程にドッグフーディング環境更新用のジョブを追加することにしました。このジョブではデプロイからデータマイグレーションまで実行され、完了したら利用者がすぐに最新版を使い始められるようになっています。

なお、テストのジョブではユニットテストに加えてSeleniumによるブラウザテストも実行しており、約1,500パターンの操作が期待どおり動作することを確認しています。ユニットテストやAPIテストも合わせると、合計10,000パターン以上のテストを実行しています。このように大量のテストに成功していることを理由に、デプロイ後の動作確認も不要としました。こうして、プルリクエストをマージしてからいっさいの手作業なしで製品の最新版を安心して利用できるようになりました。

更新作業を自動化してみて

継続的インテグレーションを用いて更新作業を自動化することにより、製品の最新版を提供し続けられるようになりました。

実際にやってみると、新たに追加した機能を試してもらえるだけでなく、それまで正常に動

COLUMN

継続的デリバリー

ドッグフーディング環境へのデプロイまで自動化できているということは、継続的インテグレーションだけでなく継続的デリバリーまで実現できているのでは？と思われる方もいらっしゃるかもしれません。しかし、継続的デリバリーでは、テスト、ビルド、非本番環境へのデプロイの自動化に加えて、本番環境へのリリース準備が整っている必要があります。たとえば、ボタン1つで本番環境にデプロイできる状態になっていれば、継続的デリバリーが実現できていると言えるでしょう。

サイボウズではドッグフーディング環境と本番環境は似て非なるもので、継続的デリバリーの実現はまだ道半ばです。本番環境では今日も温かみのある人手デプロイが行われています。

作していた既存機能に不具合が生じるいわゆるデグレードの早期発見にも役立つことがわかりました。また、本番環境に適用する前にドッグフーディング環境でデプロイやインフラのテストも行えます。

このように、ドッグフーディング環境の更新作業自動化は、インフラからアプリケーションまで製品全体の改善につながることがわかりました。

ユーザ管理の自動化で多くの人に使ってもらおう

ユーザアカウント管理の必要性

ドッグフーディング環境を実際の業務で利用する場合、ほかの正式な社内システムと同様にしっかりした運用管理が欠かせません。業務で利用する以上システムは正常に稼働し続けなければなりませんし、全社員が利用可能な状態を保つ必要があります。ユーザアカウント管理もその1つで、社員の入社や休職、退職の際に適切に対処する必要があります。

こういった作業は社内システムの運用管理部門が担当することが多いかと思いますが、サイボウズの従来のドッグフーディング環境の運用管理も運用管理部門が担当していました。製品開発チームは製品アーカイブと更新手順を運用管理部門に伝えるだけで、ドッグフーディング環境の運用管理はすべてお任せ状態でした。

一方で、kintoneのドッグフーディング環境は製品開発チームが運用管理を担当しています。好きなタイミングで環境を更新したり、試験的な機能を組み込んだりするためには、製品開発チームの管理下に置いておくほうが手っ取り早いのです。しかし、そうなってくるとユーザアカウント管理も製品開発チームが担当する必要があります。正式な社内システムとは違ってドッグフーディング環境で全社員のユーザアカウントは必須ではありませんが、より多くの社員に使ってもらうためには全社員が利用できる状態を保つようにしましょう。

手動管理は新入社員を仲間外れにする

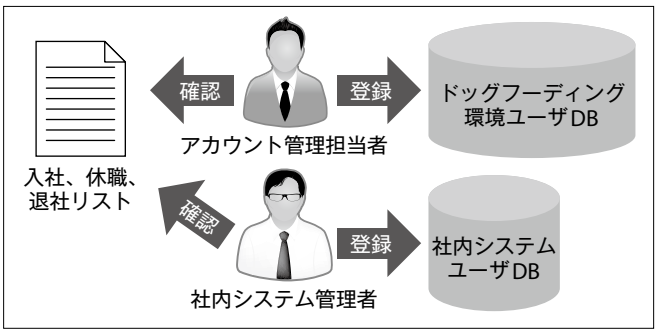
製品開発チームがドッグフーディング環境のデータベースを操作できたり、CSVなどでのユーザアカウント一括登録機能が搭載されていたりすれば、最初に全社員のユーザアカウントを一括で登録する手間はそれほど気になりません。

しかし、毎月全社向けに告知される入社や休職、退職の情報を確認し、手作業でユーザアカウントを管理するのはとても手間のかかる作業です。手間はかかりますが、より多くの社員に利用してもらうためには欠かせない作業なので、kintoneのドッグフーディング環境では毎月手作業で管理するようにしました(**図5**)。

ユーザアカウントの管理は筆者が一手に引き受けていたのですが、面倒だったので徐々に後回しにするようになっていきました。一度後回しにすると次回の更新内容が倍増するため、さらに面倒になってほとんどやらなくなってしまいました。

しかし、4月に新入社員が一気に入社した際、それまで更新していなかった分も含めて50名以上の社員がドッグフーディング環境を利用できていない状態であることに気づきました。更新を怠った結果、いつの間にか昔からいる社員しか使えないドッグフーディング環境

▼図5　手作業でのユーザアカウント管理



ができあがってしまっていたのです。とりあえず手作業でユーザアカウントを追加しましたが、今後も手作業で管理し続けるのは無理があると感じ、自動化を決意しました。

社内システムとの連携

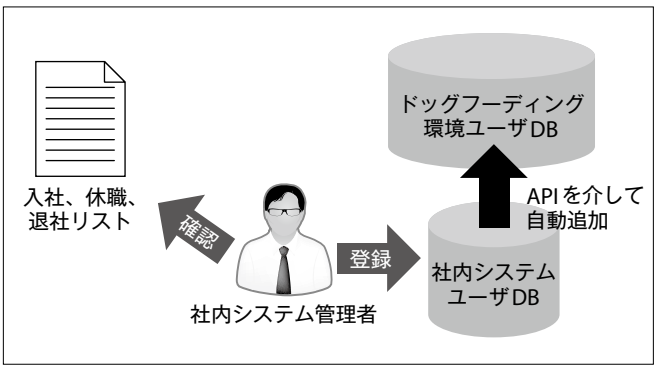
サイボウズでは幸いにも、社内システムに登録されているユーザアカウント情報(ログイン名や氏名など)を、APIを介して一括で取得できます。また、ドッグフーディング対象の製品にはユーザアカウントを一括で取得・追加するAPIが用意されています。これらのAPIを利用して、社内システムとドッグフーディング環境のユーザアカウントの差分を定期的にチェックし、ドッグフーディング環境に存在しないユーザアカウントを自動的に追加するようにしました(**図6**)。ユーザアカウントの削除については、誤って削除してしまわないよう差分を目視で確認したうえで、手動で削除しています。

ユーザアカウント情報を一括で取得するAPIが社内システムに搭載されていない場合は、社内システムを管理する部門に依頼してユーザアカウント情報を定期的にCSVファイルに書き出してもらうと良いかもしれません。複数の製品開発チームがそれぞれドッグフーディング環境を構築する場合、ユーザアカウント情報を再利用できるメリットもあります。

ユーザアカウント管理を自動化してみて

新入社員が入ったとき、ドッグフーディング環境の存在を伝えるだけですぐに使ってもらえるようになりました。また、ユーザアカウント情報の追加や削除など、普段あまり触る機会のない機能をドッグフーディングできるという副次的な効果も生まれました。より多くの社員に使ってもらうために、手作業の手間を省いて自動化に力を注ぎましょう。

▼図6　ユーザアカウントの自動追加



手間や心理的障壁を除いて継続的に使ってもらおう

ログインさせたら負け

ドッグフーディングにおいて、できる限り利用者の手間を取り除くことは重要です。利用者が自ら利用したいと思って選んだ製品であればそれほど重要でないかもしれませんが、ドッグフーディングではほんの少しの手間が利用を妨げる可能性は十分にあります。

ログイン処理も手間の1つで、休憩がてらドッグフーディング環境でも覗(のぞ)いてみようかと思ったときにログイン画面が表示されると「やっぱり後でいいや」と思われてしまうかもしれません。パスワードがわからず放置されることもあるかもしれません。製品開発チームとしては、パスワードがわからないという問い合わせに対応するのもたいへんです。

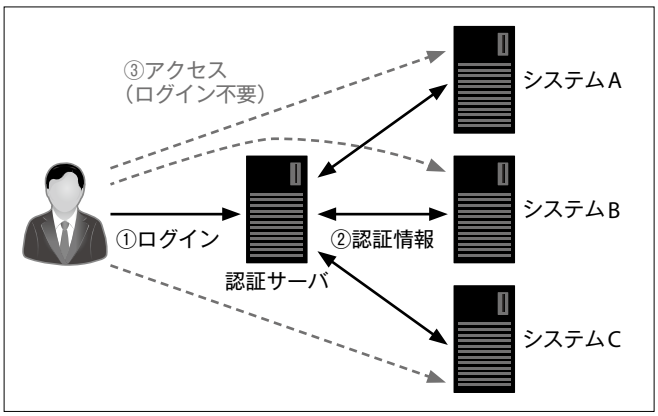
社外からもアクセス可能なドッグフーディング環境であればセキュリティに気を配る必要がありますが、社内からしかアクセスできない環境であればできるだけ利用者にログインさせないようにしましょう。もっともスマートな解決方法は、社内システムとのシングルサインオンです(**図7**)。シングルサインオン製品を導入済みであれば、それを利用するのが一番の近道です。

kintoneにはSAML認証(コラム「SAML認証を用いたシングルサインオン」を参照)を用いた

シングルサインオン機能が搭載されており、社内システムとのシングルサインオンによってログイン不要となっています。しかし、この機能が搭載されたのはドッグフーディング環境の運用開始から1年以上が経ったころで、それまではログインセッションの有効期間を長くして凌いでいました。ログインセッションの有効期間を長くすることにより、ドッグフーディング環境を頻繁に利用するユーザの手間を取り除くことはできますが、たまにしか利用しないユーザには効果がありません。実際、シングルサインオン機能の搭載前は利用者の大半が製品開発チームのメンバーでしたが、搭載後に多くの社員が継続的に使い続けてくれるようになりました。

暫定的な対処と根本的な対処の両方を実践してみて、ドッグフーディング環境の利用を広めるうえで利用者にログインさせないことの重要性を感じました。

▼図7　シングルサインオン



書き込みやすい雰囲気作り

ログインの手間を取り除いて多くの社員がドッグフーディング環境を閲覧してくれるようになっても、自社製品の改善を実現するにはまだ十分ではありません。製品には閲覧系の機能もあれば、書き込み系や管理系の機能もあります。とくに書き込み系の機能は使い勝手の向上が重要で、不具合を生みやすいところでもあります。多くの社員に閲覧系だけでなく書き込み系の機能も使ってもらえるようになると、ドッグフーディングによる自社製品の改善はさらに加速します。

kintoneのドッグフーディング環境では、長い間製品開発チームの書き込みが中心でした。製品開発チームの書き込みに対してほかの社員が「いいね」を付けてくれることが多く、閲覧はしてくれていたようです。しかし、製品開発チーム以外の社員の書き込みを目にすることは稀でした。製品開発チームの書き込みだけだとどうしても製品開発チームが想定した範囲の使い方しかされないため、予期せぬ不具合や思いもよらない使われ方を発見することはできません。

製品開発チーム以外の社員に書き込みをお願いしてもなかなか書き込んでくれない原因の1つに、心理的障壁が高いことが考えられます。書き込んでみたいと思っても、「製品開発チーム以外の社員が誰も書き込んでおらず目立ってしまいそうで書き込みづらい」「ほかの社員がどん

COLUMN

SAML認証を用いたシングルサインオン

SAMLとはSecurity Assertion Markup Languageの略で、異なるセキュリティドメイン間で認証情報を連携するためのXMLベースの標準仕様です。たとえば、社内ネットワークに存在するActive Directory Federation Services（ADFS）などの認証サーバの認証情報を使って、第三者のクラウドサービスに安全にシングルサインオンできるようになります。ユーザは認証サーバに一度ログインするだけです。

サイボウズ社内にはkintoneのドッグフーディング環境以外にも多くの社内システムが存在しますが、SAML認証を用いたシングルサインオンによってログインの手間がかからないようになっています。

な反応をするのか不安」など、ドッグフーディング環境に限らず、誰しも新しいサービスを利用する際は構えてしまうものです。

そういった心理的障壁を取り除くためには、書き込みやすい雰囲気を提供し、安心感を与えることが大切です。製品開発チームが内輪で盛り上がっているだけの環境にほかの社員が書き込むのは勇気がいります。一方でSNSのようにみんなが好きなことを書き込める環境だと、安心して書き込めるのではないでしょうか。

kintoneのドッグフーディング環境でも、製品開発チームのメンバーが好きなことを書き込むようになってから、ほかの社員も書き込んでくれるようになりました。好きなことと言っても一応仕事なので、製品開発の裏話や会社周辺のランチ情報など、ほかの社員に知ってもらいたい情報や役に立つ情報を書き込むようにしています。製品開発チームがこのような書き込みをすることで、ほかの社員も安心していろんなことを書き込んでくれるようになりました。

簡単にフィードバックできるしくみ

ドッグフーディングのゴールは製品の改善です。製品開発チームが製品のどの部分を改善するかを決定するうえで、利用者からのフィードバックは必要不可欠な情報です。より多くのフィードバックを得ることが、より多くの改善につながります。ドッグフーディング環境を継続的に使い続けてもらうのと同様に、手間や心理的障壁を軽減することでより多くのフィードバックを期待できます。

利用者からたくさんのフィードバックをもらうためには、簡単にフィードバックできるしくみが必要です。フィードバックする方法がわからなかったり、わかっていても手間のかかる方法だったりすると、利用者が問題点に気づいても製品開発チームまで伝わらない可能性があります。

サイボウズでは昔から製品開発チームがフィードバック登録フォームを用意して、利用者がいつでも登録できるようにしています。しかし、登録フォームがどこにあるかを知らないと登録のしようがありません。昔から在籍している社員は登録フォームのありかを知っていますが、新入社員は知りません。製品を使い慣れていない新入社員の意見はたいへん貴重なので、できるだけフィードバックしてもらえるようにしたいところです。実はサイボウズでもこの問題はまだ解決できておらず、登録フォームがどこにあるかわからないといった声をしばしば耳にします。

利用者にとってもっとも簡単なのは、製品の問題点を見つけたときにその場でフィードバックできることです。これを実現するための手段

COLUMN

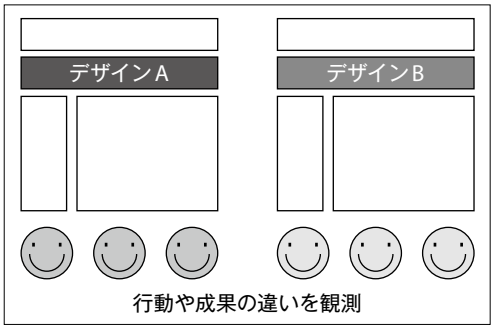
ドッグフーディング環境の運用管理

正式な社内システムは運用管理部門がきちんと管理してくれますが、運用管理のプロではない製品開発チームが正式な社内システムと同じようにドッグフーディング環境をきちんと管理するのは困難です。ドッグフーディング環境上のデータはいつ消えてなくなるかわかりませんし、製品がいつ利用できなくなるかもわかりません。バックアップや冗長化を整備しておくことも可能ですが、あくまで目的は製品の改善です。製品開発チームがドッグフーディング環境の運用に必要以上に手間をかけて、製品を改善する時間がなくなってしまっては本末転倒です。

利用者には、消えると困るデータをドッグフーディング環境に置かないこと、止まると困る業務をドッグフーディング環境でやらないことを理解してもらうことが重要です。業務を効率化するための正式な社内システムとは目的もサービスレベルも異なることを利用者に理解してもらい、トラブルなく使えるドッグフーディング環境を目指しましょう。

▼図8　ブラウザ拡張を活用したフィードバックのしくみ

▼図9　A/Bテスト



として、ドッグフーディング環境でのみフィードバック登録フォームへのリンクを表示する機能を製品に搭載したり、Webアプリケーションであればブラウザ拡張を開発して利用者に配布したりするなどが考えられます(**図8**)。誰でも簡単にフィードバックできるように、製品開発チームは登録フォームへの導線をしっかりと意識しましょう。

また、A/Bテストのように利用者が明示的にフィードバックしなくても自動で情報を収集できるしくみも効果的です。A/BテストとはWebサイトやインターネット広告でよく用いられる手法で、異なる2パターンのデザインを用意してユーザに利用してもらい、行動や成果の違いを観測してどちらのパターンが優れているかを判断する手法です(**図9**)。

A/Bテストのように自動で情報を収集できるしくみは、ドッグフーディング環境だけでなく本番環境でも活用できます。Webサイトやインターネット広告にとどまらず、kintoneのようなビジネス向けクラウドサービスでも、このようなしくみは今後さらに広まっていくでしょう。

新機能を自動で紹介するしくみ

更新作業の自動化により製品の変更点をすぐに試せるようになっても、実際に試してもらうためにはまずその変更点に気づいてもらう必要があります。社外向けにはリリースノートを公表するのが一般的ですが、社内向けにも同様のものが必要となります。

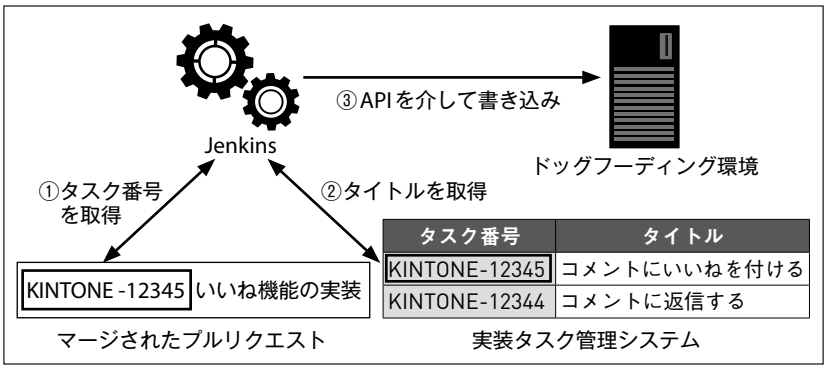
サイボウズでも従来のドッグフーディング環境を更新する際、変更点をまとめたものを社内向けに公表しています。しかし、kintoneのドッグフーディング環境のように日々製品が更新される場合、都度変更点を公表するのはたいへんです。一方、製品の早期改善を実現するためにはなるべく早く使ってもらうこと、すなわち、なるべく早く変更点を伝えることが大切です。

そこでkintoneのドッグフーディング環境では、継続的インテグレーションを用いて変更点の案内を自動化することにしました(**図10**)。まず、実装担当者がGitHubでプルリクエストを作成する際、プルリクエストのタイトルに実装タスク管理システム[注1]のタスク番号を書いておきます。タスク番号はGitHub Issuesの課題番号のようなものです。

kintoneには情報取得APIが搭載されており、実装タスク管理システムに対してそのAPIを実行すると指定されたタスク番号に応じた実装タスクのタイトルを取得できます。実装タスクのタイトルは「コメントにいいねを付ける」のように、変更点がわかる内容を登録しておきます。その後、プルリクエストがマージされてドッグフーディング環境更新用のジョブが実行される際、プルリクエストのタイトル(**図10-①**)から実装タスクのタイトル(**図10-②**)を取得し、製

注1) 実装タスク管理システムは、kintoneの業務アプリ作成機能を用いて構築しています。ちなみに、少し話がややこしくなるのですが、このkintoneは社員全員が実際の業務で利用している環境で、ドッグフーディング環境とは別に正式な社内システムとして運用されています。

▼図10　プルリクエストと実装タスク管理システムの連携



品のAPIを介してドッグフーディング環境に変更点を書き込みます(**図10-③**、**図11**)。このように、ドッグフーディング環境が更新されるたびに利用者が更新内容を確認できるしくみを構築しています。

お披露目会でドッグフーディング環境を案内

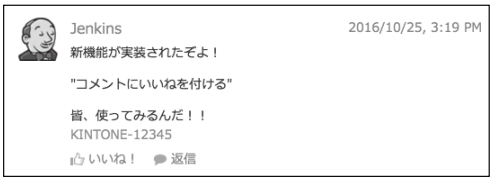
サイボウズでは、営業やマーケティングなどの販売系の社員に製品の新機能をお披露目する会があります。新機能は社外にアピールするネタになりますし、短時間で新機能について知ることができるため、多くの社員が積極的に参加してくれます。このようなお披露目会でドッグフーディング環境を見せると、ドッグフーディング環境の利用促進につながります。

お披露目会ではたくさんの質問や意見が出ますが、口頭ではうまく説明できない場合や、時間の都合で十分に説明できない場合があります。そんなときは「ドッグフーディング環境で新機能を触ってみてください」と案内しましょう。普段ドッグフーディング環境を利用しない人を巻き込むには、こういったアナログなアプローチが効果的です。

開発プロセスを見直してすばやくフィードバックを取り込もう

ドッグフーディング環境を最大限に活用するためには、得られたフィードバックをすばやく取り込むことが重要です。これはドッグフーディングに限った話ではありませんが、製品の最新版を提供し続けることができるドッグフーディング環境ではより効果的です。

▼図11　kintoneのドッグフーディング環境に自動的に書き込まれた変更点



近年広く採用されているスクラム開発は、フィードバックをすばやく取り込める開発プロセスとなっています。一方、最初に立てた計画に沿って順次開発を進めるウォーターフォール型開発では、フィードバックの取り込みなどの割り込みは計画の変更やスケジュールの遅れにつながります。

これまでに紹介したドッグフーディング環境の構築と活用に関する4つのポイントはスクラム開発でもウォーターフォール開発でも実現可能です。しかし、最後の1つであるフィードバックのすばやい取り込みを実現できるかどうかは開発プロセスに大きく左右されます。ドッグフーディングを活用して製品を改善し続けるのであれば、開発プロセスの見直しが必要になるかもしれません。簡単に変えられるものではありませんが、より良い製品開発のためにぜひチャレンジしてみてください。

フィードバックをすばやく取り込める体制が

整ったらあとは優先度を付けて対応するだけですが、フィードバックの中には優先度の低いものも存在します。ドッグフーディングが活用されればされるほどたくさんのフィードバックが集まり、対応されないフィードバックもたくさん出てきます。利用者にとっては、せっかくフィードバックしてもなかなか取り込んでもらえなかったら、徐々にフィードバックするモチベーションが下がってきます。フィードバックをくれた社員と丁寧にコミュニケーションすることを心がけて、これからもフィードバックし続けようと思ってもらえるようにしましょう。

ドッグフーディング成功事例

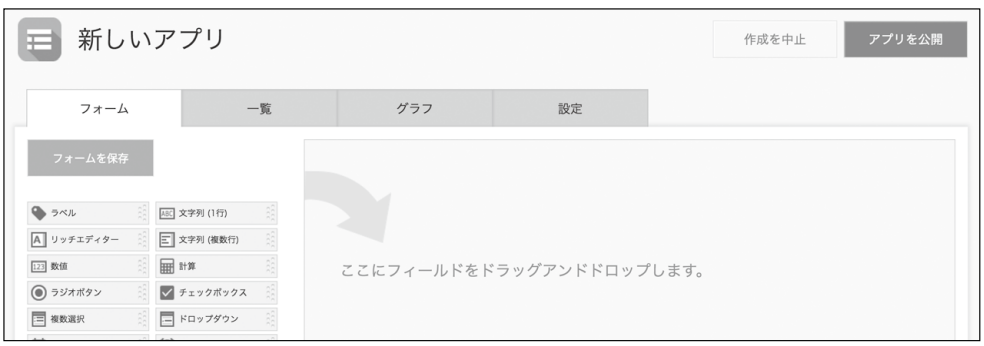
kintoneのドッグフーディングを通じて、これまでに本当に多くの改善を積み重ねることができました。その中でも、kintoneのデザインをリニューアルするプロジェクトにおいて大きな効果が得られたので少し紹介します(**図12**)。

デザインリニューアルのプロジェクトは2014年春にスタートし、同年11月に初版がリリースされました。開発期間は非常に短く、とにかくスピードが求められるプロジェクトでした。デザイナーが用意したモックアップをベースにエンジニアが実装するプロセスで、実装したデザインは随時ドッグフーディング環境に反映されました。

モックアップに対してレビューやフィードバックを繰り返してしっかりブラッシュアップしたうえで自信を持って実装を始めたはずだったのですが、いざ実装してドッグフーディング環境に適用してみると、利用者から非常に多くのフィードバックがありました。その数、2ヵ月間で100件以上です。実際に触ってみないと発見するのが難しい画面スクロールやアニメーションに対するフィードバックだけではなく、文字や背景・線の色、ボタンの位置や大きさ、余白の大きさ、フォントの種類など、モックアップの時点でなぜ発見できなかったのかと思うものもたくさんありました。

ドッグフーディングを通じて見つかった100件

▼図12　デザインリニューアル前(上)と、リニューアル後(下)のkintone



以上の問題点のうち、最終的に40件以上に対応して初版のリリースにこぎつけました。対応し切れなかったものもたくさんあり、お客様に不便を強いることになってしまったのは心苦しく感じますが、もしドッグフーディングがなかったらと思うとぞっとします。ドッグフーディングをやっていて良かった、たくさん問題点が見つかって良かった、心からそう思った瞬間でした。

ドッグフーディングのさらなる活用

製品を実際に利用してもらってフィードバックを得ることがドッグフーディングの主目的ですが、kintoneのドッグフーディング環境は利用者の操作記録の収集や実験的な機能の動作確認にも活用されています。その事例を2つ紹介します。

検索機能の操作記録収集

まず1つ目は、検索機能の操作記録収集です。利用者がどういった単語を検索し、kintone内のどのデータが検索結果画面に表示され、利用者がどの検索結果をクリックしたかを収集しています。そして、収集した操作記録を検索機能の改善に役立てています。

こういった操作記録は、お客様の環境で収集できないのはもちろんのこと、正式な社内システムとして運用されているkintoneでも関係者外秘情報を扱うため収集できません。

kintoneのドッグフーディング環境では見られて困るデータを登録しないよう利用者に周知しており、こういった操作記録の収集が可能となっています。

アーキテクチャ刷新時の動作確認

2つ目は、バックエンドのアーキテクチャ刷新に伴う動作確認です。kintoneの中核を担うしくみを置き換えるもので、品質の担保は極めて重要です。そのため長期間のロードテストが必要となりますが、正式な社内システムとして運用されているkintoneは普段の業務で利用しているため、不具合の内容によっては業務が止まってしまうリスクがあります。

そこで、品質が安定していない段階からドッグフーディング環境で長期間に渡って動作確認を実施し、不具合があれば改修するといった開発プロセスを採用することにしました。もしドッグフーディング環境がなかったら、置き換え作業と試験とロードテストを直列で実施する必要があり、リリースが大幅に遅れてしまいます。ドッグフーディング環境のおかげで置き換え作業と試験とロードテストを並行で進めることが可能となり、早期リリースを実現できています。

終わりに

製品の最新版を提供し続け、多くの社員に継続的に使い続けてもらい、得られたたくさんのフィードバックをすばやく取り込んで製品を改善することがドッグフーディングには重要であると述べてきましたが、これらの活動には開発プロセスや企業文化、対象製品の特性などさまざまな要素が影響します。ドッグフーディングを成功させて製品を改善するためには、開発プロセスを見直したり、全社員がドッグフーディング環境を触ることが認められたりといった、そんな風土を作り出すことが求められるかもしれません。

一方で、ドッグフーディングを成功させるための取り組みは、製品だけでなく開発プロセスや企業文化の改善にもつながる可能性を秘めているとも言えます。サイボウズでも、ドッグフーディングのおかげで開発プロセスの改善やチームワークの向上を実現できました。ドッグフーディング環境の構築や活用にはたくさんの苦労がありましたが、今やサイボウズにとって必要不可欠な存在です。

最後に、今回紹介した内容が、これからドッグフーディングに取り組まれるみなさんやドッグフーディングの構築・活用にお悩みのみなさんの一助となれば幸いです。SD

第3特集

いまから学ぶブロックチェーンのしくみ

ブロックチェーンの構造と機能、次なる展開

Author 嶋田 大輔（しまだ だいすけ） ㈱オルトプラス Twitter @cimadai
竹田 光孝（たけだ みつたか） ㈱オルトプラス Twitter @MitsutakaTakeda

ブロックチェーンとは何か？

身近になっているブロックチェーン

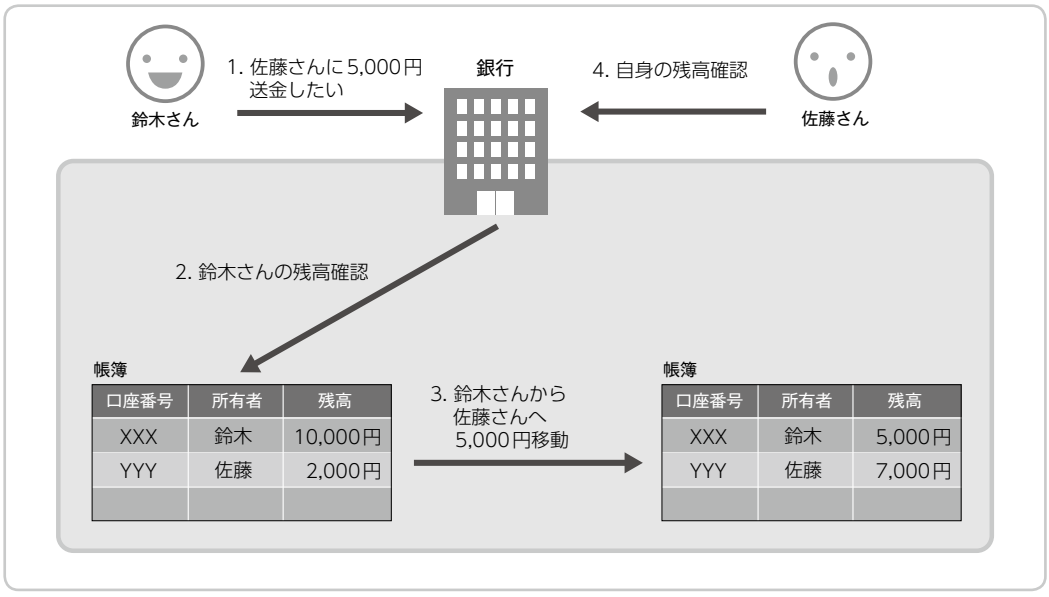
仮想通貨またはビットコインという言葉を聞いたことはあるが、ビットコインの基盤技術ブロックチェーンについてはあまりよくわからないという方は多いかと思います。仮想通貨というと難しい話のように聞こえてしまいますが、その基盤技術ブロックチェーンはハッシュ関数や電子署名など普段のWeb開発でも使われる身近な技術の上に構築された技術です。

本稿ではブロックチェーンの成り立ち、技術の概要を説明します。

ブロックチェーンのなりたち

ブロックチェーンの始まりは、2008年にSatoshi Nakamotoという人物が発表した仮想通貨ビットコインについての論文に遡（さかのぼ）ります。この論文の中でSatoshi Nakamotoは、金融機関のような信頼できる仲介者がいなくても自由に送金するために必要な技術要素と、それらを組み合わせた送金システムを提案しました。この「信頼がなくても取引を可能にするしくみ」がブロックチェーンと現在呼ばれているものです。どのように「信頼」を技術的なもので置き換えられるのか、ということを説明する前に「信頼」と

▼図1　銀行を通じた送金と信用



は何かを、通貨の送金という例で考えます。

たとえば、鈴木さんが佐藤さんから本を購入をするとき、その代金として鈴木さんは佐藤さんに5,000円を支払うとします。距離的に近い場所に住んでいれば現金が一番手軽そうですが、遠距離の場合は銀行振り込みが一般的な送金手段です。この例では鈴木さんの口座から佐藤さんの口座に5,000円を振り込みます。このとき、鈴木さんの口座の残高が5,000円減り、佐藤さんの口座の残高が5,000円増えますが、もちろん実際に現金を送っているわけではなく電子的に処理が行われるだけです(**図1**)。

さてこの銀行口座での送金の際、鈴木さんが本当に5,000円を持っているのか(口座残高が5,000円以上であるのか)、また、本当に鈴木さんの口座から5,000円が引き落とされ佐藤さんの口座に5,000円が振り込まれているのか、送金を行っている鈴木さん、佐藤さんには確認できません。

送金の過程で何かしら不備や不正があった場合、お金を受けとった佐藤さんは5,000円を使えなくなる可能性があり、大問題になります。金融機関は不備や不正がないことを確認して送金処理を行い、鈴木さんや佐藤さんは金融機関が送金の正しさを確認してくれることを「信頼」することで口座を通じた送金が可能になります。もし鈴木さんか佐藤さんの一方が金融機関を信頼せず、金融機関に口座を開設しなければ口座を通じた送金は不可能です。

このように金融機関という信頼できる仲介者がいなくても、何かしらの取引は可能なのでしょうか。これを可能にするしくみがブロックチェーンです。

ブロックチェーンの特徴

◆ 分散型P2P

上記の例では金融機関への信頼が取引を成立させるための重要な鍵でした。このような信頼できる仲介者がいない場合、取引の正当性は取引の参加者が各自検証できなければなりません。

ビットコインでは取引の参加者が分散型P2Pネットワークを形成し、ネットワーク全体で取引の検証を行います。分散型P2Pネットワークという名前には、ノード(ネットワークへの参加者)が地理的に分散しているため、ノード間で情報の共有が瞬時に行われないという意味の「分散型」と、各ノードがサーバなどを介さずほかのノードと直接通信する「P2P(peer to peer)」という意味が込められています。分散型P2Pネットワーク上で取引を検証するためには、ノード間で情報を共有する必要があります。上記送金のケースでは送金が正しく行われるか検証するために、誰がいくら持っているのかという情報をネットワークに参加している全ノードで共有する必要があります。

不特定多数が参加するネットワークでは、情報を共有する際にその情報が改ざんされる可能性はないか、異なるノードで違う結果にたどりついたときにどのように結果について合意するか、について考慮する必要があります。

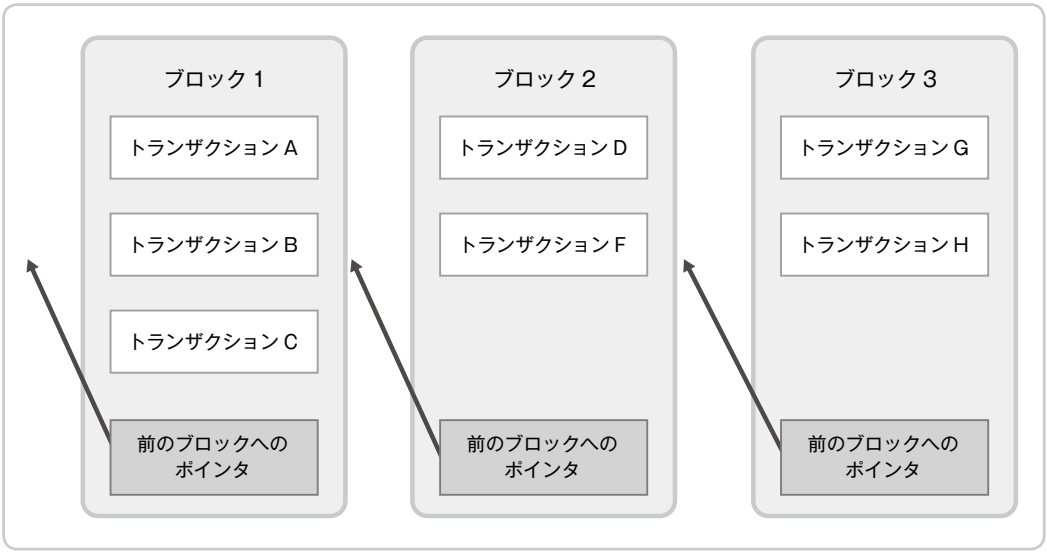
◆ ブロックの鎖(チェーン)

仮想通貨という応用から、どのようなデータ構造が必要か考えてみましょう。

まずは誰が誰にいくら送金したのかという取引の履歴を保持する必要があります。一般的にブロックチェーンでは、この取引のことをトランザクションと呼び、複数のトランザクションをひとまとめにしたものをブロックと呼びます。

個別の取引履歴を記録しているだけでは、同じお金が二重に使用されてしまう「ダブル・スペンディング」という問題が生じます。たとえば鈴木さんの口座には1,000円残高があるとします。1つのノードで鈴木さんが1,000円を佐藤さんに支払おうとします。また同時に別のノードで鈴木さんが高橋さんに1,000円支払おうとします。各ノードでは鈴木さんの残高は1,000円と認識しているため、佐藤さんへの支払いも高橋さんへの支払いも正しいものとして処理してしまいます。鈴木さんは1,000円しか持って

▼図2 ブロックの鎖



いなかったのに1,000円を二重に使用して2,000円使えたことになってしまいます。

この問題を回避するためには各取引を順序付けする必要があります。たとえば佐藤さんへの支払いが高橋さんへの支払いより「先」に起きたとした場合、高橋さんへの支払いは残高不足のため実行されません。ブロックチェーンでは、ブロックを時系列に並べ、各ブロックが1つ前のブロックへのポインタを持ちます(**図2**)。このようにブロックが一列に連なった構造がブロックチェーンの名前の由来です。

ブロックチェーンのしくみ

ブロックチェーンは、ビットコインを作り上げる過程で、その基盤を支える技術として考案されました。ビットコインはパブリックに公開されており、多くのユーザに利用されているにもかかわらず2009年の稼働開始から現在に至るまで、一度も停止することなくその価値を提供し続けています。こうした堅牢なシステムを作り上げるために必要な要素技術として次の4つが挙げられます。

- 暗号技術……トランザクションやブロックといった共有するべき情報が正しく、改ざんされていないことを保証するために利用される
- 合意形成……ブロックチェーンへの参加者がどのようなルールで共有するべき情報を正しいものとして認識するかを決めるために利用される
- 分散台帳……全参加者が同じデータを共有し、すべての参加者が対等なノードとして相互に情報を共有しあうことで耐障害性の高いシステムを作る
- スマートコントラクト……ブロックチェーン上でやりとりに応じたアプリケーションコードを実行することで不正利用や中央集権の必要性を排除する

ブロックチェーンにとって欠かせない2つの暗号技術

◆ 暗号学的ハッシュ関数

皆さんも普段プログラミングをしていてMD5(Message Digest Algorithm 5)やSHA-1、SHA-256(SHAはSecure Hash Algorithmの略)などを利用することも多いと思いますが、それこそが暗号学的ハッシュ関数(以降ハッシュ関数)

Column 「SHA-1は危険か？」

暗号学的ハッシュ関数では、異なるデータは異なるハッシュ値になるので十分な強度があれば安全なものです。動作は一方向に限定されており、一度出力されたハッシュ値からは元の値を求めることはできません。SHA-1は1995年に現在利用されているものが発表され、その安全性からHTTPS（TLS、SSL）やファイルのハッシュ値比較、Gitの管理などあらゆるシーンで広く利用されてきました。

しかし2005年以降、SHA-1に対して意図的に任意のハッシュ値を算出できるという脆弱性が指摘されており、各社のブラウザなどではSHA-1を利用した証明書のサポートを切るなどしてきました。そして2017年2月23日にはとうとう、GoogleとCWIによって同一のSHA-1ハッシュ値を持つファイルを生成することに成功したと発表されました。この発表の中では、このSHA-1衝突攻撃の詳細を90日後に公開すると言っており、現在SHA-1を利用しているコンテンツではSHA-256への移行を早急に求められています。

もし、これからハッシュ関数を利用しようと考えているのであれば、SHA-256以上の利用を推奨します。

です。ハッシュ関数は入力されたデータの長さに関係なく、それぞれ決まった長さのハッシュ値を出力します。

たとえば手もとのMacのコマンドラインで次のように入力することで手軽にSHA-256のハッシュ値を得ることができます。

```
$ echo "gihyo" | openssl sha256
(stdin)= 632cdcbd69210fd...a7c0caf794c5a73
$ echo "gihya" | openssl sha256
(stdin)= 90ef5341972e616...d9827e914a22147
```

この例では、「gihyo」という5文字の単語のSHA-256ハッシュ値と、末尾を「o」から「a」に変更した際のSHA-256ハッシュ値をそれぞれ出力しています。「gihyo」と「gihya」はたった一文字しか違いがないのですが、出力結果は大きく変わっています。このようにハッシュ関数は入力のデータが1バイトでも異なっていればまったく異なる結果を出力するという特徴をもっています。この特徴を利用して送受信するトランザクションデータや、それらが格納されるブロック全体のデータのハッシュ値をそれ自体のデータ構造に含めることで、改ざんが行われていないことを保証するのに利用しています。

◆ デジタル署名

普段の生活の中でファイルや文書の送受信について、とくに深く考えずに実行している方は多いと思います。しかし、送られたファイルが本当に正しい送信者によって送られたものかどうか保証できないかもしれません。

たとえば、鈴木さんが佐藤さんに対して重要な資料を送る場合、悪意のある誰かが鈴木さんになりすまして資料を送るかもしれませんし、鈴木さんが送った資料を途中で盗み、改ざんしてから佐藤さんに送るかもしれません。このときに佐藤さんは受け取ったファイルが「鈴木さんから送られた正しいものである」ということをどのようにして確認したら良いでしょうか。

こうした課題に対して解決策を提示してくれるのがデジタル署名です。上記の例のように、多くの場合でファイルや文書を送る際に送られた側の受信者としては、送信者が正当な送信者であり、文書自体も正当なものであるということを認証する必要があります。

デジタル署名では、鍵ペア（秘密鍵と公開鍵）を利用して署名生成と署名検証を行うことによってこれらの要求を満たしています。

秘密鍵は署名鍵とも呼ばれ、署名を行う人のみが保持するものです。一方、公開鍵は検証鍵とも呼ばれ、誰でも入手可能な状態で公開するものになります。

一般的に、秘密鍵と公開鍵には次の特徴的な

関係があります。

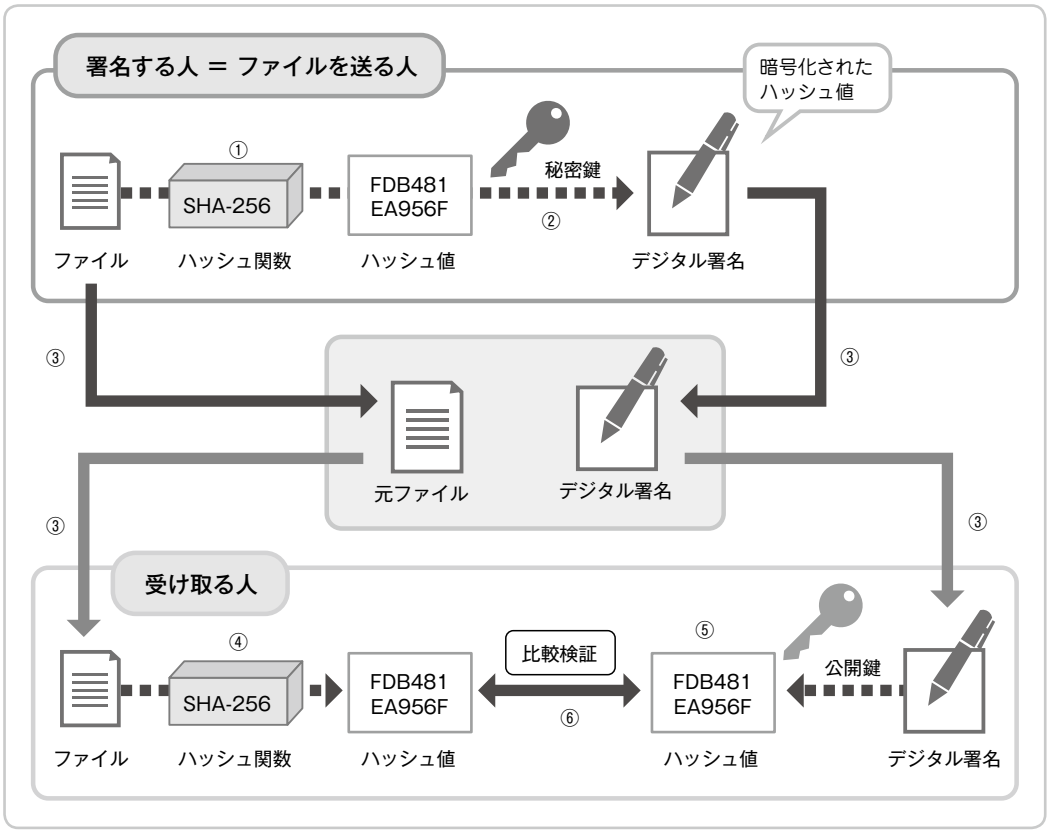
・秘密鍵で暗号化したデータは公開鍵でしか復号できない
・公開鍵で暗号化したデータは秘密鍵でしか復号できない

公開鍵による暗号化は「ネットショップでクレジットカード情報を送る」「ログインパスワードを送る」などのケースで利用されます。一方の秘密鍵による暗号化は、秘密鍵が暗号化を行う人しかもっていないということを利用して「誰が暗号化をしたか」ということを証明する手段(＝デジタル署名)に用いられています。

鈴木さんが佐藤さんに文書を送るケースを例としてデジタル署名の流れを①から⑦に示します(**図3**)。

①鈴木さん(署名する人＝秘密鍵を持つ人)が原本ファイルのハッシュ値を計算する
②鈴木さんの秘密鍵によってハッシュ値を暗号化し、これを署名とする
③鈴木さんから原本ファイルと署名を合わせて佐藤さんに送信する
④佐藤さんは受信したファイルからハッシュ値を計算する
⑤佐藤さんは受信した署名を鈴木さんの秘密鍵に対応する公開鍵で復号し、原本ファイルに対するハッシュ値を得る
⑥佐藤さんは④で計算した「受信したファイルのハッシュ値」と⑤で復号した「原本ファイルのハッシュ値」を比較する
⑦④と⑤が等しければそのファイルは確かに鈴木さんが送ったもので、改ざんなく受信できたことが保証される

▼図3　デジタル署名の流れ



このように、デジタル署名ではハッシュ関数と鍵ペアによる署名を合わせて利用し、データの送信者の認証と内容の完全性の証明を同時に行っています。

ブロックチェーンにおいては前項のハッシュ関数と本項のデジタル署名をどちらも利用しており、各トランザクションの正真性や合意形成に関するメッセージの発信元・内容の完全性の証明などさまざまな用途で利用しています。

合意形成

これまでにすでに触れたように、ブロックチェーンはそれぞれのブロックが直列に並んだデータ構造をしています。参加するノードすべてにおいて、同じ情報を持ったブロックが同じ順番で並ぶ必要があります。このブロックの並びを決めるための手法として、ブロックチェーンではさまざまな合意形成アルゴリズムが用いられています(**表1**)。

◆ PoWの動作

PoWはビットコインで利用されているアルゴリズムで、一般的にはマイニングと言われる行為で行われていることです。PoWではブロックに含まれるトランザクション情報および前ブロックのハッシュ値に乱数(ナンス)を加えてハッシュ値を計算していきます。計算されたハッシュ値があらかじめ定められたしきい値よりも小さい値になるまで乱数を変更し、繰り返し計算します。条件に合うハッシュ値が見つかったら、そのブロックを有効なブロックとして参加者に発信し、承認してもらいます。

承認を行う側としては、通知されたブロックに含まれる乱数と各情報のハッシュ値を一度だけ計算し、本当に条件に合うかどうかを確かめるということをします。このときに行われる計算処理は一度だけであり、マイニングで行う膨大な計算に比べて非常に短い時間で検算できるという非対称性があることが特徴です。

ビットコインでは、事前に定められた条件に合うブロックを生成できた場合に生成者に対して報酬としてBTCが付与されます。現在は1ブロック生成あたり得られる報酬は12.5BTCですので個人の報酬としては非常に高額なものでしょう[注1]。ビットコインではこのような報酬がモチベーションとなってハッシュ計算の競争が行われています。

余談ではありますが、ビットコインのマイニングにはSha256dというアルゴリズムが用いられており、このアルゴリズムに最適化されたASIC(エーシック)(Application Specific Integrated Circuit)という専用回路を用いることで高速にマイニングすることが可能です。ハッシュを計算する速

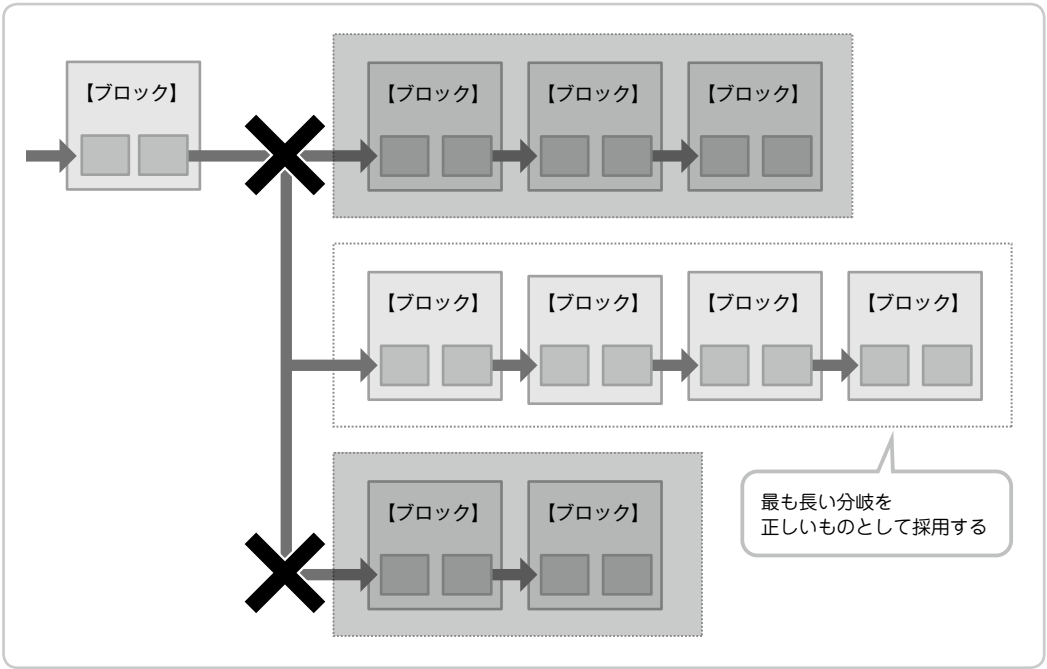
注1) 執筆時点では1BTC＝14万円台なので、12.5BTCは175万円相当

▼表1　代表的な合意形成アルゴリズム

合意形成アルゴリズム	おもな採用ブロックチェーン	利用シーン	合意方法
PoW(Proof of Work)	ビットコイン	パブリック型	ブロックのハッシュ値が一定の基準を満たすようなナンスを、非常に多くの計算を行って探す。計算力が高いほどブロック生成の確率が上がる
PoS(Proof of Stake)	Peercoin	パブリック型	当該ブロックチェーンで扱うコインの所持量による確率調整を行う。コイン所持量が多いほどブロック生成の確率が上がる
PoI(Proof of Importance)	NEM	パブリック型	参加者の中で、仮想通貨の所持量・取引量・取引相手の重要度などを指標とする重要度算出を行う。重要度が高いほどブロック生成の確率が上がる
PBFT(Practical Byzantine Fault Tolerance)	Hyperledger	プライベート型	既知の特定のノード間で発生し得るビザンチン障害を解決する。多くの計算は不要で比較的高速に合意できる

※参加ノードに制限のないものを「パブリック型」、制限のあるものを「プライベート型」と分類

▼図4　最長チェーン採用



度を表す指標としてハッシュレート(ハッシュ生成数／秒)というものが用いられるのですが、2017年3月時点でビットコイン全体のハッシュレートは3,252PH/s(ペタハッシュ／秒)となっており、非常に高い計算力があることがわかります。比較として、Intel Core i7 5820KのCPUでマイニングした場合のハッシュレートがおよそ10MH/sですので、個人がCPUでマイニングした場合にブロックを生成できる確率は3,250億分の1になります。隕石に当たる確率が100億分の1と言われているので、現実的にCPUでは到底太刀打ちできないことはわかるでしょう。

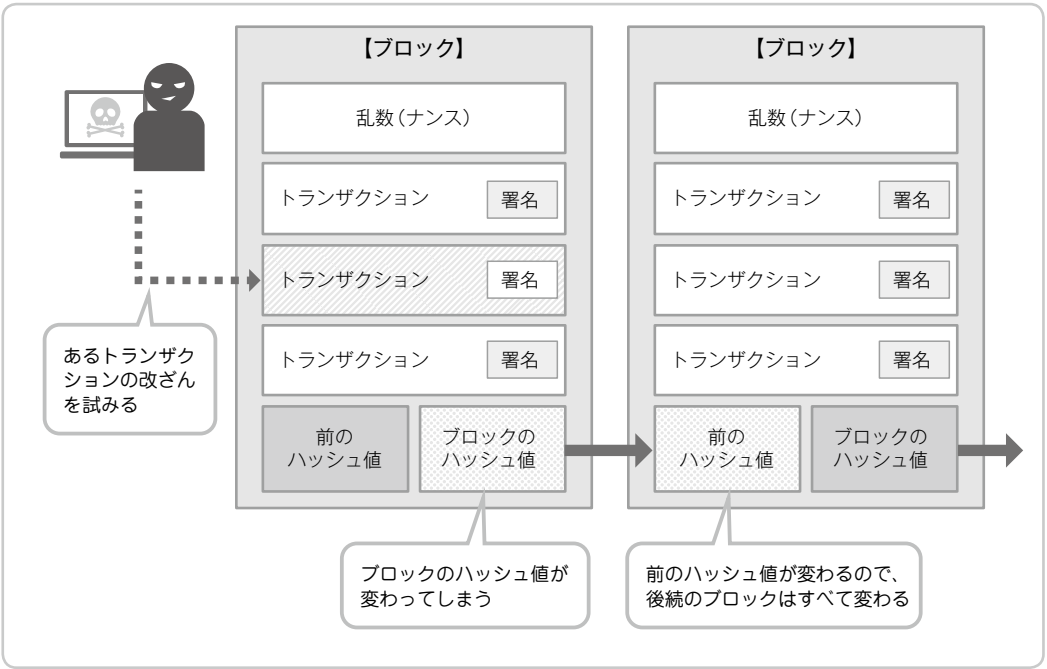
ビットコインではブロックの同時発見があり得るので、各地で同時発見された場合にチェーンの分岐が発生します。このような場合には、「最も長いチェーンを持つ分岐が最も大きな計算資源が投下されたものである」という考えに従って、最も長いチェーンを持つものを採用するという方式を取っています(**図4**)。

最も長いチェーンを採用するというアプローチには、チェーンの改ざんに対しても効果があります。たとえば悪意のあるユーザが、あるブロックに含まれるトランザクションの内容を書き換えようとしているケースを考えます(**図5**)。

この場合、トランザクションの内容を書き換えますので、ブロック全体のハッシュ値も変わってしまいます。ブロック全体のハッシュ値が変わると以前満たしていた条件(ハッシュ値がしきい値以下)を満たさなくなってしまいますので、ふたたび乱数を変更しながら適切なハッシュ値を見つけ出します。

このブロックがチェーンの先頭以外の場合は次のブロックの要素としてこのブロックのハッシュ値が利用されていますので、次のブロックのハッシュ値も書き換える必要があります。そうするとブロックの先頭まで延々とハッシュ値の再計算とハッシュ値書き換えをしなくてはならず、これが先頭ブロックまで続いていきます。改ざんしたいと思ったブロックから先頭までの成長速度(攻撃者の計算資源)が正規のチェーンの成長速度(ビットコイン全体の計算資源)を超

▼図5　トランザクション改ざん



えない限り攻撃は成功しないことから、チェーンの改ざんに対して耐性があると言われています。

合意形成アルゴリズムの違い

◆ 性能特性

ビットコインでは最も長いチェーンが最もコストがかかっているものとするPoW（Proof of Work）によってノード間で情報の合意を行っていることを紹介しましたが、プライベート・ブロックチェーンでは参加ノードの数を制限できることから、Paxos[注2]などの従来からよく知られた合意アルゴリズムを利用することが多いです。

パブリックとプライベート・ブロックチェーンの合意形成アルゴリズムを比較すると、パブリック・ブロックチェーンが採用している合意形成アルゴリズムは、参加ノード数の増加に対してトランザクションの承認速度が影響を受けにくい性質がありますが、トランザクション処理のスループットは高くありません。

一方、プライベート・ブロックチェーンではノード数の増加はトランザクションの承認速度を下げますが、決められたノード数の範囲内では高いスループットを保つという傾向があります。

◆ ファイナリティ

パブリック・ブロックチェーンとプライベート・ブロックチェーンの合意アルゴリズムでは、ファイナリティと呼ばれる性質に関しても違いがあります。ファイナリティとは、一度実行された取引の結果が覆らないことを意味します。この性質は現実世界の取引を行う場合には重要な要素と言えます。

参加ノードに制限のないパブリック・ブロックチェーンでは、ファイナリティを担保することが難しいです。たとえば、ブロックチェーン上に鈴木さんの残高が1,000円と記録されていたとします。“鈴木さんから佐藤さんへの送金600円（取引SS）”と“鈴木さんから高橋さんへの送金700円（取引ST）”が同時に別ノードにリクエストされたとします（SS：鈴木→佐藤、

注2）URL https://ja.wikipedia.org/wiki/Paxosアルゴリズム

ST：鈴木→高橋の略）。まだどちらの取引もブロックチェーン上に記録されていないためリクエストを受けたノードはどちらの取引も鈴木さんには十分な残高があり問題ないものとみなします。2つの取引が両方とも受け入れられてしまうと鈴木さんの残高はマイナス300円になってしまい、システム全体としては負の残高という不正な状態になってしまいます。

この問題を解決するには、【取引SS】か【取引ST】のどちらが“先に起きた”かを決定する必要があります。【取引SS】が【取引ST】より先に起きたとしたなら、【取引SS】の終了時点で鈴木さんの残高は400円となり、【取引ST】は残高不足で不正な送金としてブロックチェーン上には記録されません。逆に【取引ST】が【取引SS】より先に起きたとされた場合は【取引SS】が不正な送金になります。

パブリック・ブロックチェーンでは、各ノードは自分が保持している状態をもとに取引の検証を行います。取引の検証が終わった時点で、検証が終了したことをネットワーク内のノードに対してブロード・キャストします。検証の終了通知を受け取ったノードは検証の正しさを確認して自分の状態を更新します。このノードは新しい取引を検証するときは、この更新された状態をもとに取引を検証します。上記のように別々のノードで異なる取引が検証されて、終了がネットワークに通知された場合、終了通知を受け取ったノードは自分の状態をどちらかの取引結果で更新しなければいけません。このとき終了通知を受け取ったノードは各検証結果のチェーンの長さが長いほうを優先して状態を更新します。

チェーンの長さによって優先された取引はブロックチェーンに記録されます。優先されなかった取引について、ノードは優先された取引の結果を反映した状態をもとに再検証します。再検証の結果、取引が正当であれば、ネットワーク内に再度ブロードキャストされほかのノードによる受け入れを待ちます。また、取引が不正であれば、その取引のリクエストは失敗します。

上記の例に戻ると【取引SS】のチェーンの長さが【取引ST】よりも長く、【取引SS】がブロックチェーン上に記録されたとします。その結果、最新の鈴木さんの残高は1,000円から600円減って400円になります。その状態で取引STを検証すると残高不足のため【取引ST】は不正な取引となって失敗します。

【取引ST】の検証の状況だけを考えると残高1,000円の時点で検証したときは正当ということで取引が実行（ネットワーク内へのブロードキャスト）されましたが、そのあと、別の取引が実行され、そちらの取引が優先されたために残高が400円になり、【取引ST】は不正な取引になりました。並行して実行されている取引が2つしかないとわかっていれば、片方の取引が終わってから、もう一方の取引を実行すれば取り消される可能性をなくすことができます。しかし、不特定多数のノードが参加しているパブリック・ブロックチェーンでは、並行して実行されている取引がいくつあるかはわかりません。もしかしたら【取引SS】や【取引ST】以外の取引も並行して実行されていて両方の取引が取り消される可能性もあります。このようにパブリック・ブロックチェーンでは取引の実行結果が取り消されないというファイナリティという性質を担保できません。

一方、参加ノードが限定されているプライベート・ブロックチェーンではPaxosなどの合意アルゴリズムで取引を確定できます。たとえば、全ノードで多数決を行い【取引ST】を採用するか【取引SS】を採用するか決定するようなイメージです。採用された取引はチェーン上に記録されて、それ以降取り消されることはありません。多数決やその結果を全ノードで共有できるのは参加者が限定されているからです。

分散台帳

これまで見てきたようにハッシュ関数とデジタル署名、合意形成アルゴリズムによって1つ

の分散型P2Pネットワークに接続するノード間において、トランザクションおよびそれらの集合体であるブロックの正しさ・ブロックの並びを合意して、中央の巨大なデータベースに頼ることなく同じデータを全ノードで安全かつ確実に共有しています。

この「中央管理者を必要とせずに互いに信頼しないノード間においてデータベースを直接安全に共有できる」という特徴こそが分散台帳であり、ブロックチェーンであると言えます。

スマートコントラクト

ビットコインにおいては、コインの移転情報を全参加者で共有するというしくみになっていました。その上にほかの通貨の情報や、車・不動産の所有権といったものをトークン化してビットコインのやりとりに付随してこれらを授受しようとするカラードコインというアイデアが出てきました。

しかし、あくまでビットコインの既存のしくみをそのまま使うので、扱える情報量に限りがあったり、やりとりにビットコインが必要だったりと制約がありました。

そこで、Ethereum（イーサリアム）[注3]がスマートコントラクトという概念を導入したブロックチェーンとして公開され、その後複数のブロックチェーンにスマートコントラクトが導入されています。

スマートコントラクトを利用するとどういうことができるのでしょうか。よく用いられる例ですが、「自動販売機にコインを入れたら商品が出てくる」という一連の流れもスマートコントラクトです。このようにある条件下で決まった動作が行われるものをスマートコントラクトと呼んでいます。

本誌読者の皆さんは普段からプログラミングをしていると思いますが、スマートコントラクトは一言で言えば「ブロックチェーン上で動作する」という特徴を持っただけの普通のプログラムだと言えます。スマートコントラクトはブロックチェーンごとに呼び名が違ったりするのですが、実体としてはJavaScript（ライクな言語）やGolang、Java、C#など普通のプログラミング言語で書くことができるプログラムで、ブロックチェーンの状態や格納されているデータを読み書きするためのものです。

ただし、「ブロックチェーン上で動作する」という点において注意しなくてはいけないことがあります。 この「ブロックチェーン上で動作する」という言葉からどのような動作を想像するでしょうか？　ブロックチェーン全体につき1つの共通した処理機構がある、あるいは参加者の中の誰かが代表してプログラムを実行をする、といったイメージでしょうか。実はどちらも違い、全ノードで同じプログラムを実行して答え合わせをする、という動作になっています。

ブロックチェーンでは全ノードが同じデータを共有する、ということはすでに説明した内容ですが、これはスマートコントラクトにおいても同様ということです。

一般的なプログラムでは乱数を利用したり、外部のデータソースから値を取得したりして処理を行うといったことはよく行われていますが、このような不確定性のある動作は、同じデータを共有するという性質を守らなくてはならないスマートコントラクトにおいてはできず、決定論的にすべての処理が進まなくてはならないのです。

実行のたびに乱数を利用したい場合や外部サーバと通信をしたい場合は、オラクル（信託者）と呼ばれる信用できる第3のエンティティがそれらを行ってから、取得した値をスマートコントラクトに渡すという方法があります。この場合、ブロックチェーンはせっかくの分散システムですので、オラクルが単一障害点にならないように注意しなくてはなりません。

注3）URL https://www.ethereum.org

ブロックチェーンの今とこれから

ブロックチェーンをとりまく環境

過去、アプリケーション開発の主戦場がオンプレミスからクラウドに移行してきたのと同様に、ブロックチェーンではすでにいくつかのクラウド環境が提供されています。

おもなサービスとしては、IBMが主導して開発を進めるHyperledgerのクラウドサービスである「IBM Bluemix Blockchain」やMicrosoftが提供するEthereumのクラウドサービスである「Microsoft Azure Blockchain as a Service (BaaS)」、さくらインターネットがテックビューロと共同で提供しているNEMベースであるmijinのクラウドサービス「mijinクラウドチェーンβ」などがあります。いずれのサービスも、期間の多少はありますがある程度は無償で利用できますので、すぐにブロックチェーンを始めてみたいという場合にはこういったクラウドサービスを利用することを検討してもいいでしょう。

今後こうした動きがさらに活発になり、ブロックチェーンを取り巻く開発環境がより充実してくるのではないかと思います。

技術と法律

このようにブロックチェーンの開発を行う環境が成熟しつつある中で、仮想通貨の側面においては残念ながらマネーロンダリングに利用されたり犯罪に関連して利用されたりすることも増えてきました。

そこで、2016年5月に改正資金決済法(通称、仮想通貨法)が成立しました。この法律が施行されると仮想通貨は「決済手段に使える財産的価値である」と定義され、(従来の小切手やプリペイドカードと同じく)仮想通貨を法定通貨と交換する場合には非課税とすることになります。

また、仮想通貨と法定通貨の交換を行う業者は登録制となり、資本要件や業務体制における情報管理・定期的な監査など細かくルールが定められています。

ブロックチェーンの可能性

ここまでブロックチェーンの成り立ちから、その構成技術までを説明してきました。最後にブロックチェーンが今後どのような分野で応用が期待されているのか紹介したいと思います。

まず、第一の応用分野はビットコインに代表される仮想通貨です。既存の金融機関を通じた送金は複雑ないくつものステップを経る必要があり送金手数料が高く、手続きにかかる時間が長くなりがちです。ブロックチェーンを利用した仮想通貨では、このような仲介者を必要としない取引が可能になり手数料の低減や送金時間の短縮が期待されています。日本のようにインフラが発達していてコンビニなどで手軽に送金ができると仮想通貨の恩恵を感じづらいかもしれませんが、外国人労働者が自国の家族に国際送金をする場合、上記のメリットを受けられる可能性があります。2015年の時点で、全世界では仕送りの金額が約60兆円と大きな市場です。

次に期待される応用分野は文書の公証です。改ざん困難性を利用して、ブロックチェーン上に文書、または文書のハッシュ値を保存することで、その文書がある時期に存在し文書の中身が改ざんされていないことを証明できます。エストニアでは、婚姻届け、遺書、土地の登記文書などの文書をブロックチェーン上に保存することで、従来の公証の代わりにしようという試みを始めました。また同国では、個人の医療履歴へのアクセスにブロックチェーンを基盤とした認証技術を導入しました。

このほかにもブロックチェーンは資源・製品の追跡、金融商品の取引の自動化など幅広い応用が考えられています。

技術もさることながら、それに合わせて法律も変わりつつあり、とくにFintechに関しては各社注力をしている分野で、2017年も大いに盛り上がるものと考えています。SD

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。

2017年4月号

第1特集
基礎から学べば役に立つ!
目的から考える! 実作業から学ぶLinux入門（OS操作編）

第2特集
AWS LambdaとMicrosoft Azure Functionsで体験
はじめてのサーバーレスアーキテクチャ

一般記事
・mBaaSのしくみ紹介

定価（本体1,220円+税）

2017年3月号

第1特集
マーケティング&サービス向上に役立つ
ログ＆データ分析基盤入門

第2特集
CIでインフラ開発を効率化
Jenkins＋Gerrit＋Ansible＋Serverspecで基盤構築

第3特集
どうなってる? なりすましメール対策
DKIMとホワイトリストによる安心の可視化

定価（本体1,220円+税）

2017年2月号

第1特集
なぜプログラマの役に立つのか
いまはじめるDocker

第2特集
Linuxファイルシステムの教科書
Ext3、Ext4、XFS、F2FS、Btrfsの特徴と進化

第3特集
エンジニアが採用できない会社と評価されないエンジニア（後編）

定価（本体1,220円+税）

2017年1月号

第1特集
bash書き初め
シェル30本ノック

第2特集
機械学習をどう学ぶべきか?
数学とコンピュータのつなぎ方

第3特集
エンジニアが採用できない会社と評価されないエンジニア（前編）

定価（本体1,220円+税）

2016年12月号

第1特集
NoSQLの教科書
MongoDB、Couchbase、Redis、MySQLでNoSQL!

第2特集
文字コード攻略マニュアル
HTML・Java・Ruby・MySQLのハマりどころ

特別企画
先人たちの知恵と足跡に学ぶ
温故知新 ITむかしばなしSP

定価（本体1,220円+税）

2016年11月号

第1特集
クラウドコンピューティングのしくみ
AWS・Azure・SoftLayer・Heroku・さくらのクラウド

第2特集
レガシーコード改善実践録
サイボウズ流バグゼロまでの道のり

一般記事
・［次世代言語］Elixirの実力を知る【前編】
・Jamesのセキュリティレッスン

定価（本体1,220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
	千代田区	書泉ブックタワー	03-5296-0051	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	丸善　丸の内本店	03-5288-8881	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	中央区	八重洲ブックセンター本店	03-3281-1811		広島市中区	丸善　広島店	082-504-6210
	渋谷区	MARUZEN＆ジュンク堂書店　渋谷店	03-5456-2111	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322

※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）、「雑誌オンライン.com」（http://www.zasshi-online.com/）、「Gihyo Digital Publishing」（https://gihyo.jp/dp）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。

DDoS攻撃！　そのときクラウド事業者は!?

クラウド事業者が考えるDDoS攻撃への対策と対処方法

Author 山田 修司（やまだ しゅうじ）
さくらインターネット㈱

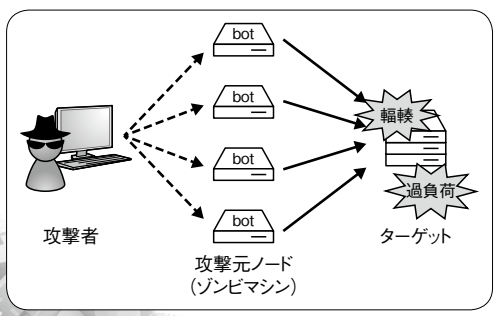
はじめに

近年では、テクノロジの進歩とともにインターネットに接続されるデバイスが増加の一途をたどっています。一方で、特定のWebサービスや設備を狙ったDDoS攻撃（Distributed Denial of Service attack）による被害も年々増加しています。

DDoS攻撃は、不特定多数のマシンを攻撃元とする、分散型のサービス妨害攻撃です（**図1**）。攻撃元として利用されるマシンは、脆弱性などを利用されてトロイの木馬などのbotを埋め込まれた機器（俗称としてゾンビマシンと呼ばれます）です。ゾンビマシンは攻撃者からの指令を受けてDDoS攻撃を開始します。

DDoS攻撃の攻撃元として利用されるゾンビマシンはおよそ数百から数千台にも及びます。DDoS攻撃の攻撃目標となってしまったターゲットでは、大量のパケットを送り込まれることを起因とするインターネット接続回線の輻輳（ふくそう）、あるいはサーバまたはデータベースなどの処理能力を大幅に上回る多大な負荷が発生するなどして、一時的あるいは長期的なサービス停止が発生してしまいます。

▼図1　DDoS攻撃のしくみ



もし、IaaSクラウド上で稼働中の仮想インスタンスやアプリケーション、あるいはVPSや専用サーバなどの従来型のサーバ環境がDDoS攻撃による被害を受けてしまったときには、いったいどのような対処ができるのでしょうか。事前に何か対策しておくべきことはないのか。もし仮にDDoS攻撃を受けてしまったときにはクラウド事業者ではどのような対応がなされているのか……。この記事では、近年におけるDDoS攻撃の特徴とクラウド事業者側の対応内容をふまえながら、シンプルなWebサーバを構成例として見立てて、DDoS攻撃の基礎知識と対策方法などのテクニックについて説明していきます。

DDoS攻撃を取り巻く現状

インターネットと接続される多種多様なデバイスは歳月の経過とともに指数的な性能向上を続けています。また現在では世界中のさまざまな場所で、より手軽で安価にインターネット接続デバイスを入手できるようになりました。

今後は、モノのインターネット（Internet of Things, IoT）の進展によって、インターネットに接続されるデバイスの数は大きく伸びること

が予想され、インターネット上を流れるトラフィックの種類も大きな変化を迎えるとも言われています。

しかしながら、脆弱な初期パスワードなどが設定された状態でインターネットに直接接続されているデバイスや、重大な脆弱性が修正されないままインターネット上に放置されてしまったデバイスの数も急増しています。

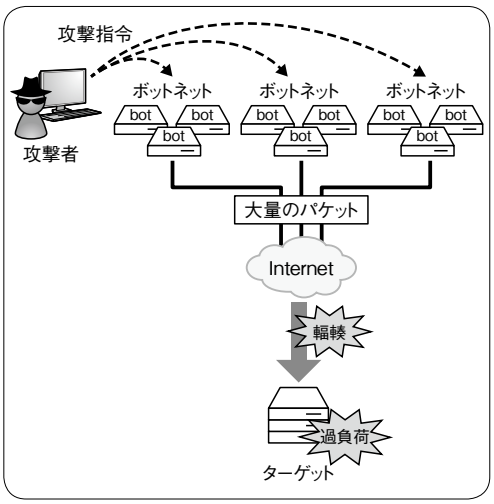
これらのような放置デバイスは、悪意ある第三者の手にかかってしまうと、あっけなく簡単にクラック(乗っ取り行為)されてしまいます。クラックされてしまったデバイスは、サイバー犯罪者による攻撃の踏み台などに利用されるボットネットの一部に組み込まれるなどして、さまざまなサイバー犯罪の攻撃拠点の1つとして悪用されてしまいます。

近年における DDoS攻撃の傾向

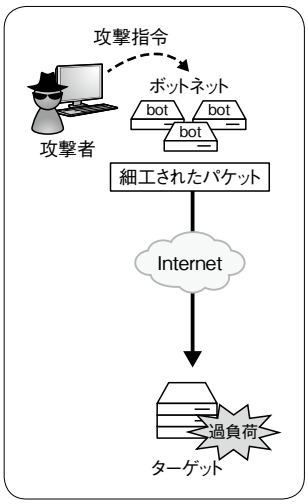
DDoS攻撃は、「過負荷を与える攻撃(大量のパケットを送りつける攻撃)」(**図2**)と、「例外処理されない攻撃(脆弱性を突く攻撃)」(**図3**)の2種類に大きく分けることができます。それぞれの特徴は**表1**のとおりです。近年では、過負荷を与えるタイプのDDoS攻撃の発生件数や規模が増加しています。

これは、現在のインターネット上でサービス停止を目的とする攻撃をしかけることを考えた場合、ターゲットの詳細を入念にスキャン

▼図2　過負荷を与える攻撃(大量のパケットを送りつける攻撃)



▼図3　例外処理されない攻撃(脆弱性を突く攻撃)



▼表1　DDoS攻撃の大まかな特徴

	過負荷を与える攻撃	例外処理されない攻撃
特徴	大量のパケットを送りつけてリソースを枯渇させる攻撃	OSやアプリケーションの脆弱性を突く攻撃
攻撃手法	UDP Flood、ICMP Flood、DNSリフレクション、NTPリフレクション、SSDPリフレクション	Slow攻撃、HTTP GET Flood、ゼロデイDDoS攻撃
攻撃規模	大きい(数Gbps～数十Gbps)	小さい(Mbps規模)
主なサービス停止原因	回線輻輳、ファイアウォールやサーバへの過負荷	サーバ、あるいはデータベースなどへの過負荷

Column　ボットネット

ボットネット(Botnet)は、トロイの木馬のようなバックドアを埋め込まれるなどして、悪意ある第三者からの指令を受けて操られてしまうマシンの総称です。ボットネットは、およそ数百台から数千台のゾンビマシンで構成されますが、数万台もの規模で構成されることもあります。これらのゾンビマシンは、DDoS攻撃における攻撃元として利用されてしまうほか、不正アクセスや不正送金のような悪質なサイバー犯罪の温床にもなっています。

したうえで脆弱性を突くタイプの攻撃を巧みにしかけるよりは、ターゲットが直接接続されているインターネット接続回線そのものを輻輳させてしまったほうが、より短時間でターゲットをサービス停止状態に陥れやすいことが理由の1つとして考えられます。

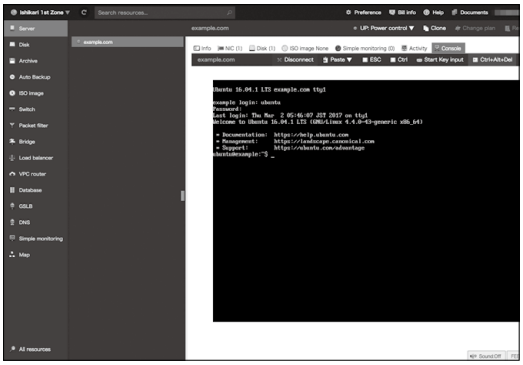
DDoS攻撃を受けてしまったら……

国内のインターネット上で現在稼働しているWebサービスのインターネット接続回線の帯域は、100Mbpsまたは1,000Mbps回線であることが一般的です。どちらにせよ、数Gbpsから数十Gbps規模のDDoS攻撃を受けてしまった場合には、ほとんどのWebサービスではDDoS攻撃の発生とほぼ同時にインターネット接続回線の輻輳が発生してしまいます。その状況から何も手を打てないということになってしまうと、DDoS攻撃が終息するまでの間、数時間あるいは数日間にも及ぶサービス停止が発生することもあります。

仮にサーバのインターネット接続回線が輻輳してしまった場合、緊急用やSSHログイン用として予備のインターネット接続回線が用意されていなければ、サーバにリモートログインしてオペレーションするのも困難な状況に陥ってしまいます。さらに、インターネット接続回線のトラフィックグラフも用意されていない環境だとしたら、いったい何が起きたのか本当に把握できなくなります。

IaaSクラウドなどを含む最近のホスティングサービスでは、サービスのコントロールパネル画面上からリモートコンソール(**図4**)によるサーバ操作が可能なサービスや、サーバのトラフィックグラフがあらかじめ用意されているサービスがあります。また、物理サーバのホスティングサービスでもIPMI搭載仕様のサーバ(「IPMI」のコラム参照)であれば、IPMIリモートコンソール経由でのログインやサーバ操作が可能なサービスもあります。

▼図4　リモートコンソール画面のスクリーンショット



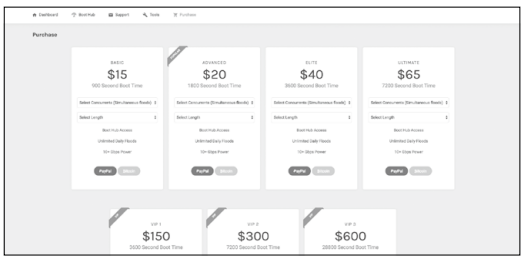
何も準備がない状態からDDoS攻撃は回避できるのか?

DDoS攻撃の攻撃パケットは、ターゲットの「IPアドレス」あるいは「ドメイン名」に向けて送信されます。よって、DDoS攻撃がターゲットのIPアドレスをターゲットにしている場合、攻

Column　DDoSサービス

海外では、料金を支払うだけで任意のIPアドレスやドメインに対するDDoS攻撃の実行が可能なDDoSサービスや、ボットネットのレンタルサービスのようなものまで出現しています(**図A**)。このようなサービスを悪用すると、誰でも簡単に特定のサイトをサービス停止に陥れるような大規模なDDoS攻撃を実行できてしまいます(**※当然ですが、このようなサービスを攻撃に利用することは法律で禁止されています**)。

▼図A　DDoSサービスサイトのスクリーンショット



撃対象となってしまったIPアドレスを変更することで一時的に回避できる可能性があります。

やや泥臭い手法ではありますが、ここでは「IPアドレスの変更」を例としてシミュレーションしてみましょう。

IPアドレスの変更作業

対象サーバのIPアドレスを変更するためには、あらかじめクリアされていなければならない前提条件がいくつかあります。ここではおおまかに2つ挙げます。

≫複数のグローバルIPアドレスを取得していること：
最近のクラウドサービスなどでは、1ユーザアカウントあたりで取得可能な固定IPv4アドレスの数が限定されていることがあります。また、従来型のホスティングサービス（VPS、専用サーバなど）では、サーバに紐付けられたIPアドレスはマシンに対して固定されているサービス仕様が一般的であり、IPスプーフィング防止などの観点から、マシンのIPアドレス変更が認められていないケースがほとんどなので注意が必要です。

≫対象IPアドレスのDNSレコード切り替えが迅速にできること：
IPアドレス変更をする場合、対象IPアドレスと紐付くDNSのレコードも変更する必要のあるケースがほとんどです。このとき、変更対象となるレコードのTTL（time to live）が86400（24時間）として設定された状態で運用されている場合、最新のレコード変更が反映されるのは最長で24時間後になってしまう可能性があります。

上記の前提条件を満たした場合でも、同一ネットワークセグメント内にあるほかのサーバのIPアドレスと重複したIPアドレスをマシンに設定してしまうと、ネットワーク通信ができなくなってしまうなどの二次障害を引き起こすことがあります。

本番環境上でIPアドレス変更作業を迅速かつ正確に実施するならば、念入りなシミュレーションとレビューがなされた作業計画が用意されていることが望ましいです。

DDoS攻撃回避策のしくみと効果

何も準備がない状態からDDoS攻撃を回避することは難しいと言わざるを得ません。障害対応や緊急作業にさほど慣れていない場合、あるいはそのような作業に慣れていたとしても、人的なオペレーションミスを起因とする不具合や障害を引き起こす可能性が高いです。

ここでは、あらかじめ準備しておくことで、DDoS攻撃に対して限定的に有効とされる予防策をいくつか挙げます。

CDN

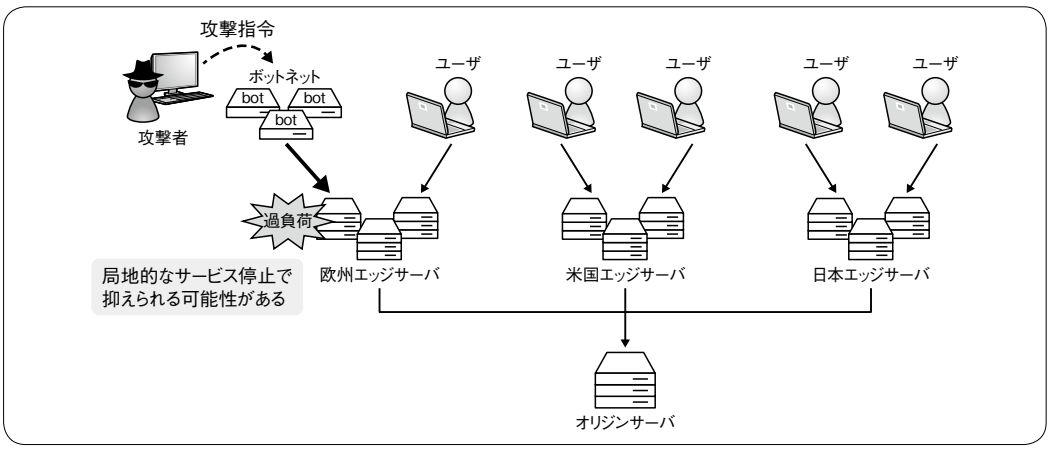
CDN（Contents Delivery Network）は、エンドユーザがリクエストしたコンテンツをWebサーバ（origin、オリジンサーバ）から取得し、一時的にそのコピーをCDNのエッジサーバ上にキャッシュとして保持することで、次回以降のリクエストにはキャッシュをエンドユーザに配信するしくみです。

CDNは大量アクセスに対する負荷分散を想定したサービスではありますが、副次的な作用と

Column IPMI

IPMI（Intelligent Platform Management Interface）は、サーバ本体の管理機能を提供するインターフェースです。サーバ本体とは物理的に独立したモジュールで動いているため、インターネット接続回線もサーバ本体とは物理的に別回線を利用していることが多く、サーバの電源がOFFの状態でも電源ケーブルから通電していてIPMI側のインターネット接続回線が正常に接続されている状態であれば、サーバ本体の状況確認や電源操作、リモートコンソール操作などに利用することが可能です。

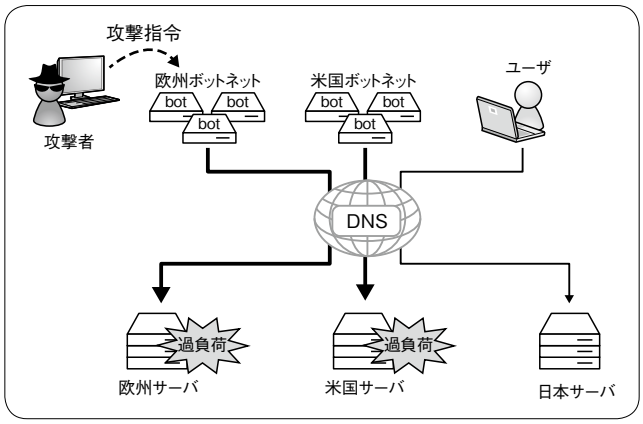
▼図5　CDNのしくみとDDoSへの効果



してFlood攻撃(**表1**の××Flood系の攻撃)や、Slowポストによる応答遅延攻撃などのDDoS攻撃にも耐性を発揮します(**図5**)。

インターネットと関連する事業を主力事業として営んでいる企業であっても、資金繰りなどのさまざまな懐事情の都合もあって、サービスのインフラ整備などに割り当てられる予算や経費というのは厳しく絞られているところも多いと思います。しかし、大量の画像や動画などのコンテンツ配信事業を営んでいる場合では、CDNを利用することでインターネット接続回線の転送量や回線帯域を大幅に削減できることが見込めることから、従量課金制の回線契約を利用中の場合は回線コストの削減、あるいは定額制の回線契約でも契約内容をダウングレードすることで、CDNの利用料金を相殺できるケースも少なくありません。

▼図6　GeoDNSを使った回避策



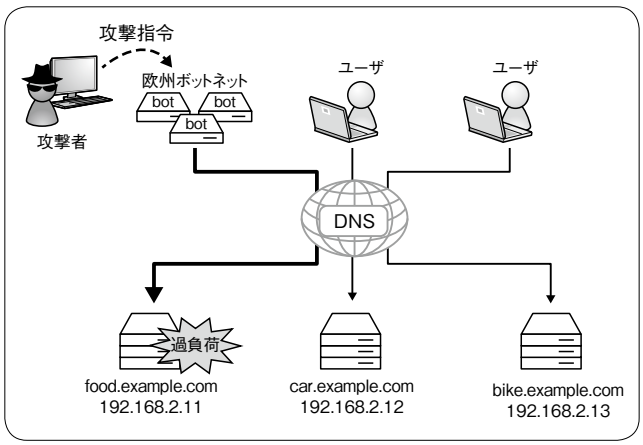
GeoDNS

GeoDNSを利用することで、DDoS攻撃の発信元から地理的に近い場所にあるサーバにDDoS攻撃を吸い込ませてしまうという手法もあります(**図6**)。

GeoDNSは、リクエスト送信元からのDNS名前解決のときに、リクエスト送信元と地理的に近い場所にあるサーバのIPアドレスと紐付けられているAレコードを返すことで、広域的な負荷分散を実現するしくみです。

この場合、本番環境同様に構成したサーバをあらかじめさまざまな地域で稼働させておく必要があります。あるいは、従量課金制のインスタンスを借用しておいて、DDoS攻撃が発生したときだけスポット的にインスタンスを起動するような手法もあります。しかし、複数台のサーバを設置しておくための準備や手間とコストがかかるというデメリットがあり、運用の手間を考えると、商用サービスの運営においてはあまり現実的な手法ではありません。

▼図7　ドメイン名を分割することによる抑制策



ドメイン名の分割

運営するコンテンツをあらかじめサブドメイン単位に分割する設計を許容できる場合には、1サブドメイン／1サーバ／1グローバルIPアドレスとする設計にすることで、DDoS攻撃のターゲットにされたときに影響範囲を1サブドメインのみに抑制し、ほかのサブドメインへの影響を最小限に食い止める手法があります(**図7**)。

しかし、この手法にも欠点があります。1つのコンテンツサイトを複数のサブドメインに分割して運営する場合、大手検索サイトの順位アルゴリズムしだいでは1つの大きなコンテンツサイトとして認識されず、それぞれまったく別々のコンテンツサイトであると認識されてしまい、検索サイトにおいて順位を大きく下げてしまうことがあります。SEO対策を気にしなければならない商用サービスにおいては、この手法の実施には慎重を期す必要があります。

大規模DDoS攻撃には役立ちにくいしくみ

ファイアウォール、IPS/IDS

IPS/IDS(Intrusion Prevention System/Intrusion Detection System)は、サーバやインターネット接続回線の通信を監視する機器です。これはおもにパターンマッチングで不審な通信の検出に成功したときに、あらかじめ連携設定されたファイアウォールに対して該当する通信をフィルタリングするポリシーを適用するアクションを起こすことで、不正な通信やDoS攻撃などからサービスを守るときに有効なシステムです。

しかしながら、DDoS攻撃に利用される攻撃パケットのほとんどは、インターネット上で一般的に利用されている正規のパケット形式でもあるため、単純なパターンマッチングによる検出が可能とされる攻撃の種類は一部のFlood攻撃(Syn Flood、UDP Flood、ICMP Flood)に限定されます。

また、多くのパケットをリアルタイム解析しなければならないことから、突発的な攻撃に迅速にアクションすることを得意とはしません。インターネット接続回線上を流れるパケット数が増加するとともに、指数的に解析処理にも時間を要してしまうことがあります。

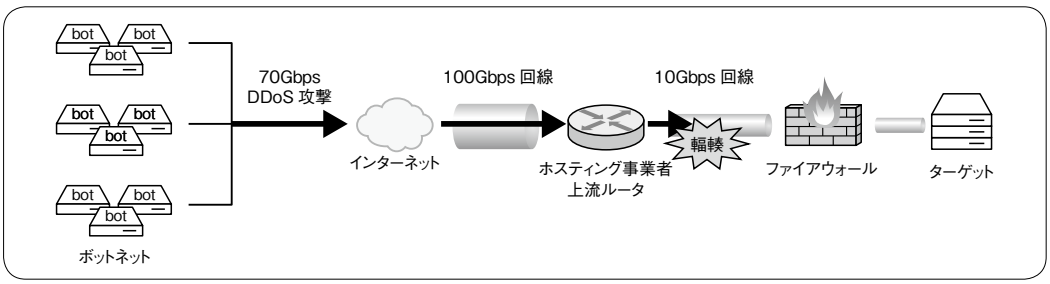
インターネット接続回線の輻輳を目的とする規模のDDoS攻撃を想定した場合、多くのケースにおいて、ファイアウォールやIPS/IDSに到達するよりも前の地点でインターネット接続回線が遮断されてしまいます(**図8**)。よって、大規模なDDoS攻撃に対しては効果を発揮することができません。

クラウド事業者の対応

DDoS攻撃が発生したとき、クラウド事業者ではどのような対応がなされているのでしょうか。クラウド事業者などと連携することで、DDoS攻撃に対して有効な手を講じることはできるのでしょうか。

ここでは、いくつかのクラウド事業者の代表的な対応をご紹介します。実際には、攻撃規模やパターンなどによって対応内容は多岐にわた

▼図8　ファイアウォールなどの設置位置と実際に回線が輻輳してしまう地点



ります。また、事業者によっても対応内容は異なります。事業者のヘルプページやドキュメントなどに対応内容が細かく掲載されていることもありますので、気になったらチェックしてみましょう。とくに何も掲載されていないようであれば、顧客サポート窓口などに一度問い合わせしてみてもよいでしょう。

ブラックホール・ルーティング

クラウド事業者が管理するインターネット接続回線が輻輳するほどの大規模なDDoS攻撃が生じたとき、クラウド事業者はブラックホール・ルーティングを実施することで「攻撃を受けている対象IPアドレス宛のすべての通信パケット」を廃棄します(**図9**)。「攻撃を受けている対象IPアドレス宛のすべての通信パケット」をネットワークの上流で廃棄することで、ブラックホール・ルーティングを実施した位置よりも下流にあるインターネット接続回線を保護するための手法です。

ブラックホール・ルーティングが実施された場合、DDoS攻撃を送りつけられたターゲットは結果的にすべての通信ができなくなります。そうすると、ブラックホール・ルーティングが解除されるまでの間は、DDoS攻撃が成功したも同然の状況に陥ってしまいます。

事業者の対応ポリシーしだいではありますが、一度設定されてしまったブラックホール・ルーティングというのはすぐには解除されません。DDoS攻撃が終息したからといってすぐにブラックホールを解除してしまうと、DDoS攻撃が間もなく再発するケースが多く、事業者が管理するインターネット接続回線のどこかの地点で回線輻輳によるネットワーク障害が発生、あるいは再発する恐れが懸念されるためです。

もし、自身が管理するサーバがクラウド事業者にブラックホール・ルーティングされてしまった場合、DDoS攻撃が終息してからしばらくの間までか、もしくはクラウド事業者側でDDoS攻撃のみをフィルタリングするような対応が可能でなければ、ブラックホール・ルーティングが解除されることはありません。

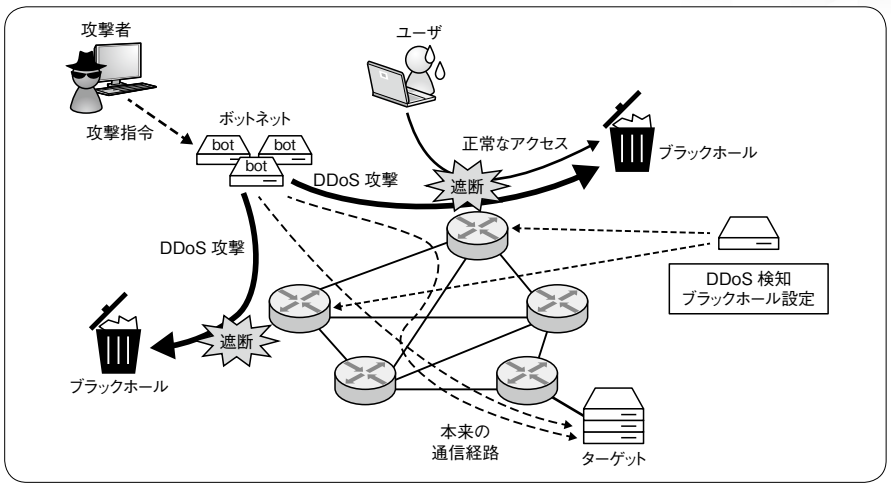
ACLによる対応

クラウド事業者が管理する上流ネットワーク機器において、ACL(Access Control List)によるパケットフィルタリングをすることは可能なのだから、攻撃元IPアドレスからの攻撃パケットだけをフィルタすることで、正常な通信のみを通過させるような対応も可能なのではないかと考えることもあると思います。

しかし、ルータのようなネットワーク機器に実装されているACL機能は、CPUに負担を与えやすいソフトウェア処理を利用してパケットの照合がされることも多く、一定以上の行を持つACL設定を該当インターフェースに適用するとネットワーク機器のCPUに異常な過負荷が発生してしまい、未知の不具合やネットワーク障害を引き起こしてしまうことがあります。

DDoS攻撃における攻撃パケットの送信元となるIPアドレスの件数は平均的に数百から数千件にも及ぶため、手動によるACL設定はそもそ

▼図9　ブラックホール・ルーティング



も難しいという理由も挙げられます。

さらにこの対応で難しいのは、クラウド事業者が提供するインターネット接続回線が輻輳するような危機が差し迫った状況下でなければ、クラウド事業者側の判断で勝手に特定の通信を遮断するような措置を実施することは法律的にできないという事情もあります。正当な理由なく、特定の通信を遮断するような行為は、法律で原則禁止とされている通信の秘密を侵害する行為に該当してしまいます(「ブロッキング」のコラム参照)。

よって、クラウド事業者側で特定の攻撃パケットのみをフィルタリングできるようなケースはかなり稀(まれ)であるか、特別なインフラ構成で構築されているケースでしか実施することができないことがほとんどです。現在時点において、この手法をクラウド事業者側で実施できるケースは限定されています。

SDNによる取り組み

SDN(Software Defined Network)はコントローラと呼ばれるマシンに搭載されるソフトウェアを使ったパケット制御をコードで定義することによって、通信回線上を流れるパケットを柔軟にスイッチングあるいはルーティング処理などできるようにする技術です。これはプログラマブルに仮想的なネットワークを構築するための技術でもありますが、比較的大容量な通信回

Column　ブロッキング

通信の盗聴や検閲、あるいは改ざん行為については、通信の秘密を侵害する行為に相当するため、日本国内の法律によって原則禁止されています。クラウド事業者は、提供するサービス事業を安定的に提供できなくなる状況に相当すると判断されるほどの危機的状況にならなければ、通信のフィルタリングなどの措置を実施することができません。これはDDoS攻撃でも例外ではありません。

特定の通信を遮断することは検閲行為に該当するため、正当な理由またはやむを得ない理由なく攻撃パケットを遮断するようなことはできません。よって、原則としては、事業者が管理するインターネット接続回線のどこかが輻輳寸前に陥るような状況になるまでは、事業者側は通信の遮断などの対応を実施することができません。

ただし、サービス約款などにおいて、特定の通信をブロッキングすることについて当事者間での合意がなされている場合に限っては、あらかじめ特定の通信をブロッキングするなどの対応が可能とされています(例：OP25Bなど)。

線を流れるパケットでもほぼリアルタイムに制御することができるという特徴を持つため、SDNのテクノロジを応用することで、インターネット接続回線上を流れるパケットをリアルタイム解析したうえでDDoS攻撃を半自動的に遮断するような取り組みも一部では行われています。

DDoSは攻撃側が圧倒的に優位的な立場であることは間違いありません。攻撃側は低いコストで大量の攻撃パケットを生成することができます。しかしながら、攻撃側にもさまざまな制約があるため、攻撃パケットを無尽蔵に生成できるわけではありません。

DDoS対策を徹底するならば、防御側には高度な技術力や高コストな機材、あるいはDDoS対策サービスなどを組み合わせて運用する必要があります。また、かなり大規模なDDoS攻撃を想定した定期的な訓練まで必要になってきてしまいます。

あらかじめさまざまなDDoS対策を複合的に施しておくことで被害を最小限に食い止めることは可能です。しかし、攻撃のしくみ上の原則としては攻撃発生源のより近く、より上流で食い止めることができる手段であるほど対策としての効果は絶大であり、クラウド事業者のユーザが利用できるDDoS対策は、対応範囲やその効果が限定されてしまうというのも実情です。

悩ましいところですが、ここで見方を変えてみましょう。ほとんどのサーバでは、サーバの稼働を始めてからサーバを廃止するまでのサイクルを終える間にDDoS攻撃を送りつけられるようなことはありません。適切な管理がなされているサーバにおいて、連日のようにDDoS攻撃を受けるというケースはとても稀です。

仮にDDoS攻撃のターゲットになってしまったとしても、ほとんどのDDoS攻撃は発生から24時間以内に自然終息します。本格的なDDoS対策にかかる各種コストなどを十分に検討したうえで、「攻撃が終息するまで、まずはじっと我慢する」とすることも全然アリです。

DDoS攻撃に狙われている！と思ったら、実際にはリフレクションDDoSの踏み台になっていたというケースも少なくはありません。誰もがDDoS攻撃の被害者となりうる可能性はありますが、自身の管理下にあるマシンが悪意ある第三者に踏み台として利用されてしまうことで、意図せずに加害者側に回ってしまう危険性についても認識しておかなければなりません。iptables設定やトラフィック監視、パスワード管理やセキュリティアップデートなど、サーバ管理の基本のイロハをしっかり守ることも大切です。SD

Column　パスワード

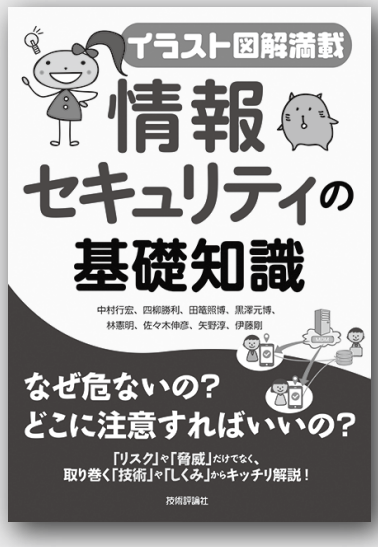
テスト環境、あるいは一時的なものだからといって簡単なパスワードを設定した状態でインターネット上において稼働させてしまった場合、SSHブルートフォースアタックの被害を受けるなどして数日のうちにあっけなくクラックされてしまいます。

以前はFTPサーバやレンタルサーバ、VPSやクラウドのように手軽な環境ほど簡単なパスワードが設定されがちでしたが、最近ではSSHサーバを起動しているDockerコンテナに脆弱なSSHパスワードを設定してしまうケースが急増しています。

パスワードを使用しなければならない場合は、なるべく強固なパスワードを利用するか、SSHログインであれば公開鍵認証方式を利用するほうがより安全です。また、fail2banのような防御ツールを導入することでSSHブルートフォースアタックによる攻撃頻度を抑制することができます。

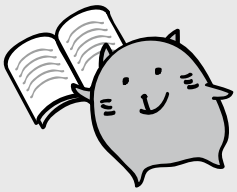
また、レアケースではありますが、機器を廃棄する際にパスワード情報や各種設定情報の消去作業を怠ると、廃棄したはずの機器が中古ショップなどに流通してしまうなどして第三者の手に渡ってしまった場合には、会社内で利用しているパスワード情報や各種認証情報などが外部に流出してしまうこともあります。ご注意ください。

技術評論社



イラスト図解満載

情報セキュリティの基礎知識



情報漏えいやサイバー攻撃など情報セキュリティに関するニュースが毎日のように流れています。また、日常的にPCやスマートフォンでインターネットに接している我々は、他人事のように思えません。ただし、技術的な背景やしくみは年々複雑化し、「なぜ危ないのか?」「どこに注意すればいいのか?」すらわかりづらくなってきました。
そこで、本書では、情報セキュリティを複雑に構成する個々の要素技術をイラスト図解でやさしく解説しました。安全かつ快適なIT生活を送るためにも活用してください。

中村行宏、四柳勝利、田篭照博、黒澤元博、林憲明、佐々木伸彦、矢野淳、伊藤剛　著
A5判／288ページ
定価（本体1,980円+税）
ISBN 978-4-7741-8807-2

大好評発売中！

こんな方におすすめ

- 情報セキュリティに初めて学ぼうとする方
- なんとなく知っているけれど、きちんと学びたい方

技術評論社

軽量・高速モバイルデータベース

Realm（レルム）入門

Realm（レルム）はSQLiteやCore Dataの代替となるモバイルデータベースで、メモリ効率が良く、高速に動作することから多くのモバイルアプリ開発者に注目されています。本書はRealmをテーマにした国内初の技術書で、モバイルデータベースの基本的なことからリファレンスや利用する際の注意点／Tipsに加え、Swift 3をベースにしたiOSアプリ開発のサンプルソースを盛り込みながら解説していきます。

菅原 祐、Realm 岸川克己　監修
B5変形判／288ページ
定価（本体2,880円+税）
ISBN 978-4-7741-8848-5

大好評発売中！

こんな方におすすめ

- iOSアプリ開発でデータベースをまったく使ったことがない方
- SQLiteやCore Dataをよりシンプルな形に置き換えたい方
- Realmを使ったことはあるが、より実践的なアプリ開発での使用方法を知りたい方

特別企画

スマホアプリ開発をもっと便利に簡単に!

ニフティクラウド mobile backend mBaaSのしくみ紹介

第2回 クラウドサービスで運用不要! すべておまかせシステムの舞台裏

Part1 ニフティクラウドmobile backendの遍歴と裏側、すべて語ります!

● Author 中津川 篤司(なかつがわ あつし)

開発の舞台裏インタビュー

ニフティクラウドmobile backend(以下NCMB)がサービスを開始したのが2013年9月で、それから3年半ほど経過しました。当然ですがWebサービスは一度ローンチしたら終わりではなく、継続的なメンテナンスやアップデートが行われます。NCMBについても同様で、リリース当時とは大きく様変わりしています。とくにNCMBがターゲットとしているアプリ市場は年々大きく進化しており、それに合わせてNCMBも進化しています。

今回はそうした遍歴やこれまでにあった苦労した点などを実際の開発者であるニフティ㈱の野田 雄也氏にインタビューしました(本インタビューは2017年2月末時点のものです)。

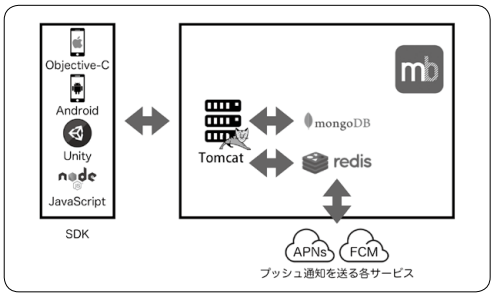
▼写真1 ニフティ㈱ 野田 雄也氏

原点から語る、NCMBのしくみ

――現状のアーキテクチャについて教えてください

サーバサイドの言語はJavaで、Tomcatを採用しています。バックエンドはMongoDBやRedisを使っています。管理画面についてはAngularJSを採用しています。最初のローンチ時点ではRedisは使っておらず、途中から利用開始しました(詳細は後述)。あとは各SDK

▼図1 NCMBのアーキテクチャ



(iOS/Android/JavaScript/Unity)があり、各言語で開発されています。すべて自社のIaaSであるニフティクラウド上に構築されています(**図1**)。

開発体制としてはサーバサイドは7名(外部委託含む)、管理画面は2〜3名となっています。SDKについては全体で4名程度となっています。

ソフトウェア開発体制の変化

――サービス提供し始めてから大きく変わった点はありますか？

1つめとしてはMongoDBのバージョンアップがあります。MongoDBは世界中で広く使われているドキュメントデータベースですが、運用が非常にたいへんなことでも知られています。実際、ローンチ当初で使っていた2.1系では運用でとても苦労させられました。たとえば2.1系ではレコード単位のロックができず、コレクション(RDBMSでいうところのテーブル相当)やデータベース単位でのロックが発生します。このせいでパフォーマンスが激しく低下する時期がありました。

バージョン3系でドキュメントロックに対応したことで大幅にパフォーマンスが向上しています。今後3.4系にアップデートする計画もありますが、すでに数万のアプリを運用している中とあって調査に時間がかかっている状態です。内部ではRDBMSに移行するかという話も出ているのですが、MongoDBでは同じカラムであっても型の異なるデータが入れられてしまうので、すでにそういったデータが存在する中では移行は困難と思っています。現在では4つのレプリカセット、計12台のMongoDBサーバで運用しています。

2つめとしてRedisの採用があります。もともとセッションやプッシュ通知のキューを管理するのにMongoDBを使っていましたが、Redisに乗り換えました。MongoDBではレプリカへのデータ伝搬が遅く、その結果としてプライマリの負荷が高くなる傾向がありました。Redisを使うことでそうした問題が出なくなっています。移行前に比べるとiOSの配信速度は約5倍まで向上しています。処理するワーカーとしてもiOS向け2台、Android向け2台で運用しています。

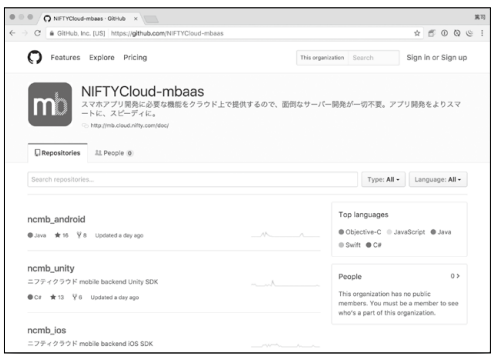
最後に各SDKをオープンソースにしたことです。iOS/Android/Unity/JavaScriptのSDKすべてがオープンソースとしてGitHub上で公開されています(**図2**)。実はニフティとしてオープンソース・ソフトウェアを提供する文化は当時はありませんでした。そのため社内でどういったメリットやリスクがあるのかを慎重に議論しました。結果としてデメリットはとくに感じられず、メリットが多かったと感じています。何よりコードを見る目が増えたことで、バグを利用者が発見して報告してくれたりします。内部の開発者としてもちゃんとしたものを作ろうと気を引き締める効果もありますし、SDKだけで足りない場合は利用者自らが一部を書き換えて使っているケースもあります。サポートとしてもSDKのコードが見られないことで、アプリの問題なのかSDKの問題なのか切り分けられないことがありました。そうした問題がオープンソース化によって解決しています。

NCMBが今後どのような展開をしていくのか？

――改善したいポイントはありますか？

NCMBでは各データに対してACL(権限)が

▼図2　NCMBのGitHubリポジトリ

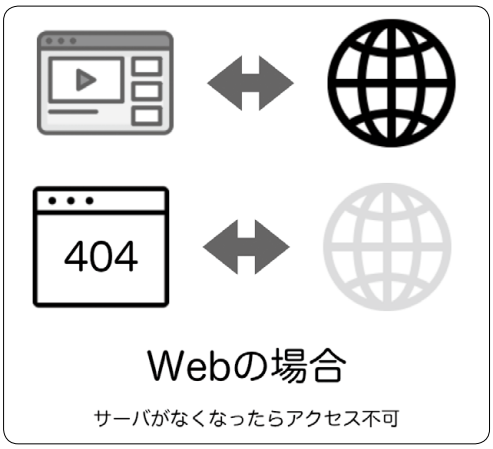


設定できます。これはオブジェクトの形式になっており、さらに検索条件として必ず含まれるものになります。たとえば全員アクセス可能なデータであっても、全体に対する読み取り権限が設定されているデータといった検索条件が設定されます。ACLはオブジェクトであるために有効なインデックスが設定しづらく、検索結果が大量の場合や、シンプルな検索条件であった場合にパフォーマンスを改善しづらいことがあります。ここは今後対応したいと考えています。

また、アプリならではと思うのですが、アプリを削除したり、退会したりしていてもアプリからのアクセスは続きます。Webサービスであればサーバを落としてしまえばアクセスはこないのですが(**図3**)、アプリはすでに配布されていて、各端末にインストールされているからです(**図4**)。その結果、アクセスのあったアプリキーの存在チェックや、シグネチャの検証(アクセスの妥当性チェック)を毎回行わなければなりません。この負荷が無視できないくらい大きくなっています。

NCMBは共用サービスですので、一部のアプリがAPIのコネクションを食いつぶしてしまうことがあります。そうなるとほかのアプリに対しても影響が出るという問題があります。それを防ぐため、アプリ単位での同時接続数について上限を設けようかと考えています。

▼図3　Webの場合



NCMBを運用していてわかったことですが、この同時接続数が一番の問題になります。先日サービスを停止してしまったParse.comも、途中で料金体系を接続数単位に変更しています。同時接続数を引き上げると料金も上がるしくみです。恐らく私たちが今感じている問題と同じ原因だったのではないかと予測しています。解決策としてはひとつひとつのコネクションを速く処理してレスポンスを返していくことにあります。それによって全体の負荷も下がっていきます。

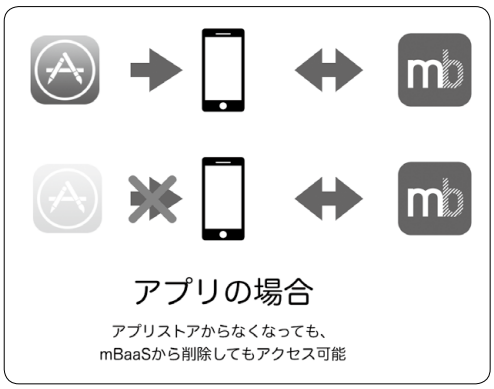
NCMBの運用と現実

――運用体制はどうなっていますか？

開発陣については前述のとおり、全体で13～14名となっています。あとはサポート体制として7名くらいで行っています。これは有償の利用者に対して1対1でのメールサポートを行うものです。ここ1年でユーザ数が5倍くらいになっており、それに伴ってサポートへの問い合わせも3倍になっています。ドキュメントは整備していますが、それでもカバーし切れていない部分が存在します。

NCMBに限らずクラウド系サービスは、利用者から見れば内部がブラックボックスになっています。そのためセキュリティやパフォーマンスなど、外からではわからない部分に対する問い合わせも増えています。この対応には、開発

▼図4　アプリの場合



ガイドラインを公開して、よくある質問についてはそちらを参照してもらえるようにしています。

mBaaSは開発者のコードしだいでパフォーマンスが変わってくるサービスです。そのためサポートに来るメールも千差万別で、IaaSのようにわかりやすくありません。その意味ではサービス提供側に求められるサポートレベルが高いサービスだと感じています。

――これまででたいへんだったことはありますか？

基本的にたいへんなのは利用者の増加や、想定外の利用法に対するものになります。NCMBは汎用的な作りになっていますので、実際にどう使うかはアプリ開発者に依存しています。そのため時々想定外の利用法によって負荷が急激に上がることがあります。そういった経験はこれまでの運用の中で1つずつ解決してきました。もともと事例がParse.comくらいしかない市場でしたので、どう使われて、どれくらい負荷があるのかといった指標がありませんでした。その意味では暗中模索しながらの運用だったと言えます。

また、とくにゲームアプリにおいてはヒットするかどうかでトラフィックが大きく変わることがあります。もともとの契約の5倍以上なんてこともあります。それでも想定されている利用法だったらいいのですが、実装方法によっては問題になります。

mBaaSへの特化で切り拓くマーケット

――今後についてはいかがでしょうか？

当たり前ですが、mBaaSを利用する方はサーバサイドについてのリテラシーは高くない場合が多いです。また、アプリを初めて作り始めたという方も数多くいらっしゃいます。そうした方でも安心して使ってもらえるプラットフォームになれるとすばらしいと思います。翻(ひるがえ)って、大規模に利用するユーザの存在もあります。ニフティクラウドのリソースを無制限に投入していけば解決できることも多いので、初級者と大規模ユーザの二軸を大切にしていきたいと考えています。

現状を見ると、NCMBはmBaaSに特化したサービスとしては世界最大手のレベルになってきていると感じます。まわりを見てもNCMBと正面から競合するようなサービスは生き残っていません。一方で、GoogleやAWS、MicrosoftなどがmBaaSのようなサービスをリリースしてきています。今後はそういった巨人企業がライバルになっていくでしょう。

2015年くらいまでは時代の先取り感が強くて、苦労したところもありますが、2016年以降伸びが継続しています。今後も開発や改善を進めて皆さんのアプリ開発に役立っていく存在でありたいと思います。

Part2 アプリ作成の実際

● Author 小山 哲志(こやま てつじ)

MCMBでアプリを作ろう！

このパートでは「Ncmboard」というシンプルな掲示板アプリを作成していきます。プラットフォームにはAndroid、言語はJavaを選びました。なおソースコード一式は筆者のGitHubで公開しています[注1]。執筆時時点のAndroid Studio最新版(バージョン2.3)でのビルドと動作を確認しています。

NCMB Web管理画面での設定

アプリ（プロジェクト）の新規作成

まずはNCMBのWeb管理画面にログインします[注2]。アカウントを持っていない場合は@

注1） https://github.com/koyhoge/ncmboard

注2） http://mb.cloud.nifty.com/signup.htm

nifty IDが必要です。こちらも持っていない場合は新規登録画面から無料登録ができます。

NCMBに初めてログインする場合は、利用規約の確認画面があります。確認して先に進んでください。NCMB管理画面では、いわゆるプロジェクトのことを「アプリ」と呼びます。最初はアプリがまだ1つも作られていないので、アプリ作成画面になります(**図5**)。アプリ名の入力欄にそのアプリを表す名称を入れます。今回は「Ncmboard」としましょう。新規作成するとアプリ作成を確認できる画面が表示されます(**図6**)。画面の中ほどに、APIキーとして「アプリケーションキー」と「クライアントキー」という2つの文字列が表示されています。この2つはNCMBサービスにアクセスするために、作成するAndroidアプリに埋め込むことになります。

プッシュ通知などの設定

今回のアプリではプッシュ通知の受信を行いますので、その設定を管理画面で行います。管理画面の右上にある[アプリ設定]ボタンから[プッシュ通知]タブを選択し、一番上の[プッシュ通知の許可]の項目を[許可する]に変更して[保存する]ボタンを押します(**図7**)。また[Androidプッシュ通知]のAPIキーには、Firebase Cloud Messaging(FCM)のFCMトークンを入力します。Firebaseの管理画面で取得してください。詳しくはNCMBで簡単なチュートリアルがありますので参考にしてください[注3]。

▼図5　ログイン直後の新規作成画面

▼図6　アプリ作成後の画面

ユーザの作成

今回のアプリではユーザがログインして掲示板にメッセージを書き込むシンプルなものですが、ユーザの登録機能は持たないので、ログインできるユーザをあらかじめ作っておきます。

NCMB管理画面の左端のメニューから[会員管理]を選択し、上部の[+新しい会員▼]ボタンから[新しい会員の新規作成]を選びます(**図8**)。グレーで[(undefined)]となっているuserName

注3) チュートリアル(Android):mobile backendとFCMの連携に必要な設定(http://mb.cloud.nifty.com/doc/current/tutorial/push_setup_android.html)

▼図7　プッシュ通知設定

▼図8　会員管理画面

▼図9　ユーザ作成

フィールドをダブルクリックして入力モードに入り、ユーザ名を入力して Enter を押します。同様にpasswordフィールドも入力するとレコードが保存されます。objectIdフィールドに自動的に値が入って、無事に保存されたことがわかります。この記事ではユーザ名を「testuser01」、パスワードを「testtest」としてユーザを作ったことにしておきます(**図9**)。

アプリ開発の実際

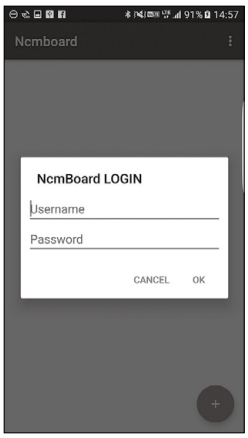
次は、アプリの基本的な挙動を見てみましょう。アプリを起動すると、まずログインパネルが表れます(**図10**)。ログインパネルに先に作っておいたユーザ名とパスワード：testuser01、testtestを入力して[OK]をタップします(**図11**)。メッセージリスト画面が表示されますが、最初はまだメッセージがありません(**図12**)。右下の[＋]のFloating Action Buttonを押すと、メッセージの投稿画面が開きます(**図13**)。何かメッセージを入力して[POST]ボタンを押します(**図14**)。メッセージリスト画面に先ほど入力した内容が表示されます(**図15**)。ふたたびメッセージを投稿するとメッセージリストに追加されます(**図16**)。

NCMBの機能を使ってユーザログインを管理しているので、アプリを起動しなおして、別のユーザでログインしてメッセージを投稿しても、それぞれのユーザ名が表示されます。

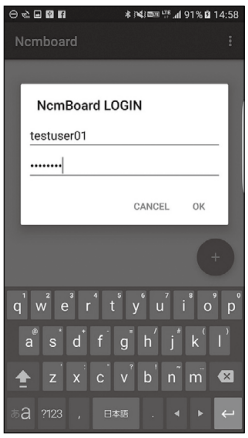
SDKの初期化

ここからプログラムの解説に入ります。まずは管理画面に表示されたAPIキーを指定して、SDKを初期化します。

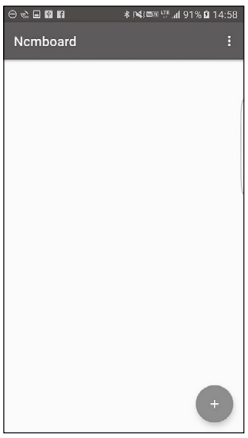
▼図10　ログイン画面表示



▼図11　ユーザ名とパスワードの入力



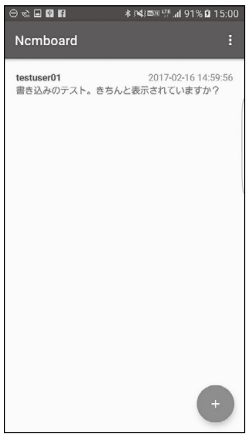
▼図12　まだメッセージが表示されない



▼図13　メッセージ投稿画面

▼図14　メッセージの入力

▼図15　メッセージが表示される



▼図16　図15の状態にさらに追加される

MainActivity.javaのonCreate()メソッドでこの処理をしています。app_keyとclient_keyは実際のAPIキーに書き換えてください。この2つのキーが外部に流出すると、ほかの人がNCMBのアプリデータに自由にアクセスできてしまいますので、扱いにはくれぐれも注意してください。

・SDKの初期化部分

```
// initialize NCMB
String app_key = "xxxxxxxx";
String client_key = "xxxxxxxx";
NCMB.initialize(this, app_key, client_key);
```

ユーザログイン

次に行われるのはダイアログを表示してのユーザログイン処理です。これはMainActivityのshowLoginDialog()メソッドで行われています。画面上のEditTextに入力された文字列からユーザ名とパスワードを取得したあとは、**リスト1**の処理を行います。NCMBUserクラスのloginInBackground()メソッドを呼び出して、そのコールバックに渡された例外がnullかどうかでログインが成功/失敗を判断します。ログインが成功した場合は、コールバックに渡されるNCMBUserインスタンスncmbUserがログインユーザ情報を保持しているので、それをMainActivityのインスタンス変数currentUserに保持しておきます。

▼リスト1　Javaによるユーザログインプログラムの例

```
NCMBUser.loginInBackground(username, password, ➥
new LoginCallback() {
    @Override
    public void done(NCMBUser ncmbUser, ➥
NCMBException e) {
        if (e != null) {
            // Login failed
            Toast.makeText(MainActivity.this,
                    "Invalid username or passwrd",
                    Toast.LENGTH_LONG)
                    .show();
            showLoginDialog();
        } else {
            // Login successed
            currentUser = ncmbUser;
        }
    }
});
```

データストアにメッセージ保存

NCMBのデータストアは、クラスと呼ばれる識別子ごとに、任意のフィールドを指定してデータを保存できる機能です。RDBでいえばクラスがテーブルに、レコードが行、フィールドが列にそれぞれ相当します。いわゆるKVS(Key-Value Store)ですので、あらかじめフィールドの定義を決めておく必要はなく、1レコード中にkeyとvalueの組み合わせはいくつでも保存できます。

それぞれvalueは型を持っていて、次の型を使うことができます。

・文字列
・配列
・数値
・日付
・真偽値
・オブジェクト
・緯度経度

メッセージ投稿画面に入力された文字列は、appendMessage()メソッドでデータストアに保存されることになります。**リスト2**のデータストアにレコードを保存するためには、まずデータストアのクラス名を指定してNCMBObjectインスタンスをnewします。コード中のNCMB_CLASSNAME_MESSAGE定数は"messages"と定義されているので、Javaのコードとしては、

```
NCMBObject messageObj = new ➥
NCMBObject("messages");
```

となります。

次にNCMBOject.put()メソッドを使ってkeyとvalueをセットします。今回はユーザID、ユーザ名、投稿メッセージをそれぞれセットしますが、いずれも文字列になります。

```
messageObj.put("userId", currentUser.
getObjectId());
messageObj.put("userName", currentUser.
getUserName());
messageObj.put("message", item.
getMessage());
```

最後に組み立てたNCMBObjectのsaveInBackground()メソッドを呼び出して、保存完了です。成功したら、力技ですがすべてのメッセージを読み込んで再表示しています(**リスト2**)。

NCMB管理画面で確認すると、投稿したメッセージがmessagesクラスに登録されていることがわかります(**図17**)。

データストアからメッセージ読み込む

データストアからレコードを一気に読み込むには、NCMBQueryクラスを使います。Queryという名のごとく本来は特定の条件のものを検索するための役割ですが、何も指定しないと与えられたクラスのレコードを全件取得します(**リスト3**)。

NCMBQueryの検索はNCMBObjectに限らず多くのものを対象とするので、Javaのジェネリクス機能を使って汎用化されています。

まずは仮型引数にNCMBObjectを使ってNCMBQueryインスタンスを作ります。

```
NCMBQuery<NCMBObject> query = new
NCMBQuery<>(NCMB_CLASSNAME_MESSAGES);
```

NCMBQuery.whereEqualTo()などのメソッドを使って次のように条件を絞り込むこともできますが、今回は全件取得なので使用しません。

```
query.whereEqualTo("key", "value");
```

NCMBQuery.findInBackground()メソッドで検索とデータの取得を行います。コールバックにはList<NCMBObject>が返されるので、そこからオブジェクトを1つずつ取り出してメッセージリストに追加します。

```
query.findInBackground(new
FindCallback<NCMBObject>() {
    @Override
    public void done(List<NCMBObject>
list, NCMBException e) {
        if (e != null) {
```

▼図17　NCMB管理画面で確認する

▼リスト2　saveInBackground()のサンプル

```
protected void appendMessage(MessageItem item) {
    final MessageItem tmpItem = item;

    NCMBObject messageObj = new NCMBObject
(NCMB_CLASSNAME_MESSAGES);
    messageObj.put("userId", currentUser.
getObjectId());
    messageObj.put("userName", currentUser.
getUserName());
    messageObj.put("message", item.getMessage());

    messageObj.saveInBackground(new DoneCallback() {
        @Override
        public void done(NCMBException e) {
            if (e != null) {
                Toast.makeText(MainActivity.this,
                        "Message post failed",
                        Toast.LENGTH_LONG)
                        .show();
                return;
            }
            // update all messages
            loadMessages();
        }
    });
}
```

プッシュ通知の受信

最後に今回のアプリでプッシュ通知を受け取れるようにしてみましょう。まずはプッシュ通知を受け取るためのパーミッションの設定と、レシーバー、サービスの登録をAndroidManifest.xmlに記述します。manifestタグの要素として必要なパーミッションを記述します(**リスト4**)。

次にapplicationタグの要素としてレシーバーとサービスを登録します(**リスト5**)。

同じくapplicationタグの要素として、通知に表示されるアイコンや、タップしたときに表示されるアクティビティを設定します(**リスト6**)。

次に実際のコードを見ていきます。

アプリの起動時には配信端末情報をNCMBに登録します。これはInstallation(インスタレーション)と呼ばれ、データストアのinstallationクラスに保存されます。installationを登録するには、FCMの送信者IDを指定します。これはFirebaseの管理画面に表示されているものを使います(**リスト7**)。

アプリを起動したあとでNCMB管理画面の[データストア]→[installation]を見ると、その端末がプッシュ配信対象として登録されていることがわかります。

installationは登録時にさまざまな付加情報を付けることができて、それをもとにプッシュ通知の配信端末を絞り込むことができます。

それではNCMB管理画面から実際にプッシュ通知を送ってみましょう。

管理画面の右端メニューの[プッシュ通知]から[+新しいプッシュ通知]ボタンを押すと、プッシュ通知の情報登録画面になります。そこでタイトルとメッセージを入力し、[Android端末に配信する]にチェックを入れると、installationに登録されていれば「N端末に向けて送信されます」と配信数が表示されます(**図18**)。

そこで[プッシュ通知を作成する]ボタンを押すと、通知がキューに登録され、しばらくのあとに端末に配信されます。Android端末の通知バーからプッシュが通知されたことが確認できます(**図19**)。

今回はプッシュ通知を管理画面から送信しましたが、もちろん送信用のAPI

▼リスト3　NCMBQueryクラスを用いてメッセージを読み込む

```
protected void loadMessages() {

    NCMBQuery<NCMBObject> query = new NCMBQuery<>(NCMB_CLASSNAME_MESSAGES);
    query.findInBackground(new FindCallback<NCMBObject>() {
        @Override
        public void done(List<NCMBObject> list, NCMBException e) {
            if (e != null) {
                Toast.makeText(MainActivity.this,
                        "Failed loading messages",
                        Toast.LENGTH_LONG)
                        .show();
            } else {
                List<MessageItem> tmpMessages = new ArrayList<MessageItem>();
                for (NCMBObject obj : list) {
                    MessageItem item = new MessageItem();
                    item.setUserName(obj.getString("userName"));
                    item.setUserId(obj.getString("userId"));
                    item.setMessage(obj.getString("message"));
                    item.setTimestamp(obj.getUpdateDate());

                    tmpMessages.add(item);
                }

                // update messages
                messages.clear();
                messages.addAll(tmpMessages);
                messageAdapter.notifyDataSetChanged();
            }
        }
    });
}
```

▼リスト4　AndroidManifest.xmlの例(その1)

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<permission android:name="パッケージ名.permission.C2D_MESSAGE" android:protectionLevel="signature" />
<uses-permission android:name="パッケージ名.permission.C2D_MESSAGE" />
```

▼リスト5　AndroidManifest.xmlの例（その2）

```
<receiver
    android:name="com.nifty.cloud.mb.core.NCMBGcmReceiver"
    android:exported="true"
    android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE"/>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="パッケージ名"/>
    </intent-filter>
</receiver>
<service
    android:name="com.nifty.cloud.mb.core.NCMBGcmListenerService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE"/>
    </intent-filter>
</service>
```

▼リスト6　AndroidManifest.xmlの例（その3）

```
<meta-data android:name="openPushStartActivity" android:value=".MainActivity"/>
<meta-data android:name="smallIcon" android:resource="@mipmap/ic_launcher"/>
<meta-data android:name="notificationOverlap" android:value="0"/>
```

▼リスト7　installationクラスへのFCMの送信者ID指定例

```
final NCMBInstallation installation = NCMBInstallation.getCurrentInstallation();
String fcm_sender_id = "xxxxxxxx";
installation.getRegistrationIdInBackground(fcm_sender_id, new DoneCallback() {
```

▼図18　プッシュ通知の登録画面

▼図19　届いたプッシュ通知

も用意されているので、プログラムから自動的に送ることもできます。

終わりに

機能的にはほんのさわりだけの紹介になってしまいましたが、NCMBを利用することでスマホアプリの開発に必要な基本機能を、簡単に使えるようになります。この優位性を本稿で理解いただけたのではないでしょうか。NCMBには、ほかにも便利な機能がたくさんあります。興味が湧きましたら、ぜひ無料アカウントを作成して試してください。SD

人工知能時代のLispのススメ

～ラムダ式からLispの作り方まで

第1回

なぜ今Lispなのか? Lispはここがすごい!

Author 五味 弘(ごみ ひろし)　沖電気工業㈱

Lispは、半世紀以上前に生まれた古いプログラミング言語です。そのLispをなぜ今、取り上げるのでしょうか。それはLispには現在でも、他言語にない斬新なものがあるからです。そしてLispは今でも多くの言語に大きな影響を与えて続けています。今回はLispの古くて新しい側面を見ていきます。

なぜ今Lispなのか?

Lispは1958年に米国のジョン・マッカーシーらによって作られた世界最古の記号処理用言語です。1954年に数値計算用として作られた世界最初の高水準言語FORTRANから少し遅れて誕生しました。この古いLispをなぜ今、取り上げるのでしょうか。新しい言語が次から次へと生まれているこの時代に、この古いLispを学ぶ価値はあるのでしょうか。

もちろん、答えは「T」(TはLispで真を意味する)です！　そしてLispが今も生き続けている理由は、ほかの言語にはない特徴があり、今でも多くの言語に影響を与えているからです。それが現代のIoTやクラウド、人工知能の時代に再評価されています。

Lispはここが新しい!

半世紀以上前の古いLispが持つ新しいものとは何でしょうか。「プログラミング言語には2種類あり、Lispとそれ以外の言語である」と言われることがありますが、Lispは見かけも機能も、ほかの言語と大きく違っています(**図1**)。

1 数値計算でなく記号処理

FORTRANやC、Javaなどの言語は数値計算を主な目的として作られてきましたが、Lispは最初から記号処理を目的に作られました。ここでの記号(シンボル)は単なる文字列でなく、各種の属性を持っているデータで、自然言語処理などの多くの分野で使われています。Lispはこのシンボルとリストを基本として、記号列を`(I hava a pen)`のようなリストで表します(**図2**)。このようにLispは、その誕生のときからほかの言語と違う記号処理という目的で作られました。

▼図1　Lispはここが違う!

2 関数型プログラミングのラムダ式

Lispは関数型プログラミング言語の元祖で、関数型の基本であるラムダ計算とその表記である「ラムダ式」が言語仕様として最初からあります(ラムダ式については、後述します)。また関数型言語で使う再帰関数も、Lispが最初に実装しました。

今流行しているIoTやクラウドシステムでは並列処理が重要であり、副作用[注1]がない関数型プログラミングはこの並列処理に有効です。この利点が評価され、最近ではラムダ式が関数型言語だけでなくJavaやC#、C++のような一般の言語にも採用されています。

3 データとプログラムが同じ表記

Lispは動的な機能を持った言語です。プログラムとデータが同じ形式で表現され、プログラムもデータであるため、プログラムの実行中にデータだけでなくプログラムも柔軟に変更できます。プログラムとデータが同じ形式で表現できるのはほかの言語では類を見ない一大特徴です。これについては後述します。

4 インタプリタの元祖

Lispはインタプリタ言語の元祖で、プログラムを完成させなくても部分的に柔軟に逐次実行できます。Lispが会話型言語と呼ばれるのは、このインタプリタがあるからです。インタプリタは今ではいろいろな言語に採用され珍しくなくなりました。もちろん、Lispにはコンパイラもあります。

5 型宣言が不要

さらにLispでは型宣言する必要はなく、型宣言の煩わしさからプログラマを解放します。これはシンボルの値にはどんな型でも格納できるしかけがあるからです。またこれはJavaScriptのvariant変数などにも伝承されています。

6 強力なマクロ

そしてLispには強力なマクロ機能があり、マクロ定義やリードマクロを使って縦横無尽にLispを拡張できます。このLispのマクロは、C言語のように単に書き換えをするマクロでなく、プログラムを生成するプログラミング(これをメタプログラミングと呼んでいます)ができる強力なマクロです。

7 煩(わずら)わしいメモリ管理から解放するGC

今はもう当たり前になったGC[注2]機能ですが、このGCはLispによって最初に採用されたものです。GCのおかげでプログラマはメモリ管理の煩わしさから解放されました。そしてこの成果がJavaなどの多くの言語に受け継がれています。

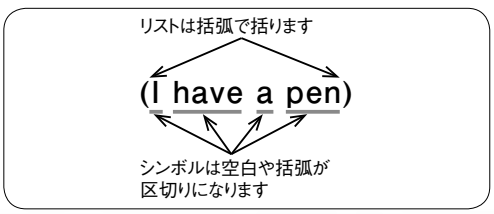
8 構文がシンプル

ほかの言語では、加算演算子「+」の表記は中置記法で「1 + 2」のように書き、関数は前置記法でfoo(1, 2)のように書きますが、Lispではどちらも前置記法で(+ 1 2)と(foo 1 2)のように統一されています。でも(* (+ 1 2) (+ 3 4))のように括弧だらけになってしまいますが。

Lispには呪文(おまじない)も必要ではありません。たとえばJavaではメインメソッド関数は「public static void main(String[] args)」のような呪文を唱える必要がありますが、

注1) Side Effect。関数の本来の作用は引数を入力して値を返すことです。これ以外の作用が副作用で、たとえば、関数適用の前後で変数の値を変えるなどは副作用です。なお、副作用の例も含めて関数型プログラミングについては後述します。

▼図2　リストとシンボル



注2) Garbage Collection：ガーベッジコレクション。GCとは、プログラムから使われずに不要になったオブジェクト(これをガーベッジ(ゴミ)と呼びます)をプログラマが明示的に解放するのではなく、言語処理系が自動的にガーベッジを見つけ、その領域を再利用できるように回収することです。

Lispでは必要ありません。Lispの関数car(カー)やcdr(クダー)も、ほかからみれば呪文のように聞こえるかもしれませんが。

以上のようにLispの特徴を紹介してきましたが、すでにほかの言語にこれらの特徴が採用されて、今では当たり前になった技術もあります。しかし記号処理向きということと、プログラムとデータが同じ形式ということ、強力なマクロ、構文のシンプルさはほかの言語にまだまだ引けを取りません。Lispの独擅場です。

しかしこのような機能を実装するには、Lispが開発された当時としては大きなマシンパワーが必要になり、半世紀前のコンピュータではその力を発揮することはできませんでした。つまり早過ぎた天才だったのです。コンピュータが進歩して、やっと時代がLispに追いついてきたのです。

Lispの昔と今

1 Lispの歴史

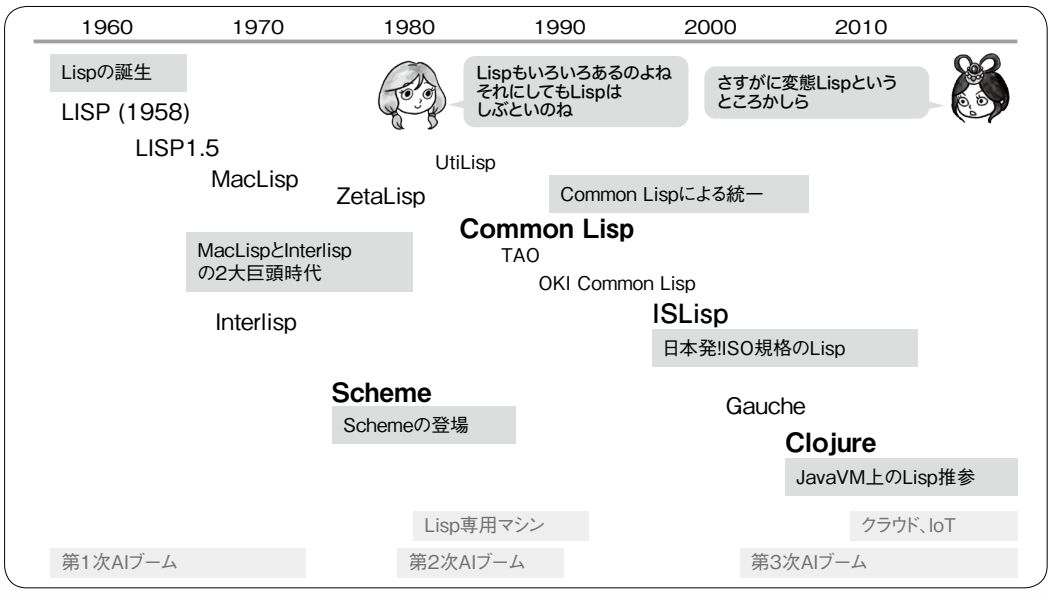
Lispの歴史を**図3**に紹介します。ここでは詳しくは書きませんが、1958年に誕生したLispはMacLisp[注3]とInterlisp[注4]の2大巨頭時代からCommon Lisp[注5]による統一を迎えました。またScheme[注6]も少し前に誕生しました。その後Common Lispの巨大さを反省してISLisp[注7]などの小さなLispも誕生し、またJava VM上で動作するClojure[注8]も誕生しました。なお、この記事で紹介するLispプログラムはCommon Lispで記述するようにしています。

2 スクリプト言語としてのLisp

さらにLispはプログラミング言語単独でなく、Lispの持つ柔軟で動的な機能を活かして、ソフ

注3) 1960年代後半にMITで開発されたミニコン上のLisp処理系。
注4) 1960年代後半にBBNとスタンフォード大で開発されたミニコン上のLisp。
注5) ガイ・L・スティールJr(GLS)らがMacLispを中心にInterlispも加えて仕様策定したLispで今の主流。1994年にANSI規格になり、その処理系は多く実装され、たとえば、CLISPなどがあります。
注6) 1970年代にGLSらによって仕様策定されたLisp。小さな言語仕様で先進的な機能を多く持ち、現在でも多く使われています。
注7) 1990年代に日本の伊藤貴康氏や湯浅太一氏が中心となって策定したISO規格のLisp。Common Lispの核言語として小さな仕様となっています。
注8) 2000年代に開発されたJava VM上で動作するLisp。Javaの豊富なライブラリが使えます。

▼図3 Lispの歴史



トウェアシステムのスクリプト言語としても実装されています。古くはEmacsのEmacs Lispから、AUTOCADのAutoLisp、InterleafのInterleaf lisp、変わったところでは掃除機ロボットのルンバもLispで書かれています。

❸ Lispに影響を受けた言語たち

Lispは**図4**に示すように多くの言語に影響を与えています。そしてLispはこれからも多くの言語に影響を与え、しぶとく生き続けていくでしょう。

Lispはリスト処理

❶ Lispの生きる道はリスト処理

Lispでは**図2**で紹介した`(I have a pen)`のようなリストが基本的なデータ構造です。**図5**にこのリストの実装例を紹介します。そして何と言ってもLispはLISt Processing(リスト処理)から名付けられたように、リスト処理を中心にしたプログラミング言語です。このシンボルのリストを使って記号処理プログラムを書くための言語がLispなのです。

❷ Lispの形式 ー Lispは括弧だらけ

Lispでは実行することを評価(eval)と呼びます。Lispでリスト`(+ x y)`を評価すると、リストの先頭の「`+`」は関数として扱い、続く`x`と`y`はその関数の実引数となります。`(+ x y)`のように関数を前に書きますので、前置形式(prefix notation)と呼ばれています。一方、ほかの言語では「`x + y`」と書き、これは「`+`」を中に書きますので、中置記法(infix notation)または代数記法(algebra notation)と呼ばれています。Lisp以外の言語では「`+`」などの演算子は中置記法で、関数は`six(x)`のように前置記法になります。一方、Lispではすべて`(+ x y)`のような前置記法になります。この結果、`(+ (* x y) (* z u))`のような括弧だらけになりますが。これもLispの特徴というか、性癖みたいなものです。

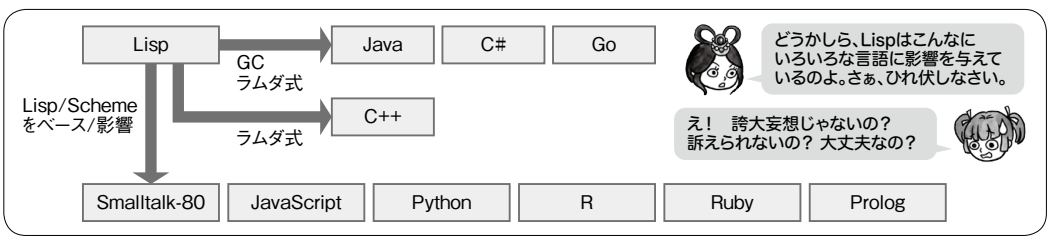
❸ リストをデータとして扱うquote

リストをデータとして扱うときは先頭の要素が関数として評価されては困りますので、リストをデータとしてそのまま返すquoteがあります。たとえば`(quote (1 2 3))`は、引数の`(1 2 3)`を評価せずにそのまま返します。また`(quote (1 2 3))`の短縮形として`'(1 2 3)`と書くことができます。

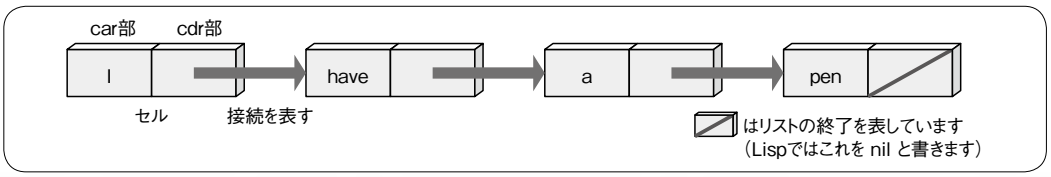
❹ リスト関数

Lispには非常に多彩なリスト処理用の関数が用意されています。そのほんの一部を**表1**に、そ

▼図4　Lispから影響を受けたプログラム言語



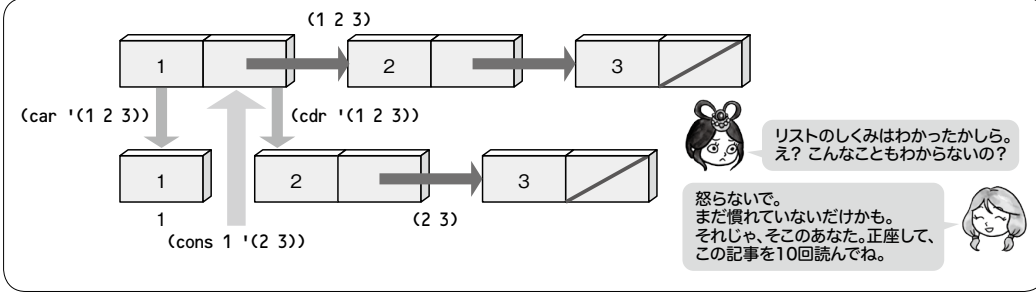
▼図5　リストの実装例



▼表1　Lispのリスト関数（一部）

動作	入力	結果
リストの先頭の要素を取り出す	`(car '(1 2 3))`	`1`
リストの先頭の要素を除いたリストを取り出す	`(cdr '(1 2 3))`	`(2 3)`
リストの先頭に要素を追加する	`(cons 1 '(2 3))`	`(1 2 3)`
リストを連結する	`(append '(1 2 3) '(4 5 6))`	`(1 2 3 4 5 6)`
与えられた要素からなるリストを生成する	`(list 1 2 3)`	`(1 2 3)`
	`(cons 1 (cons 2 (cons 3 nil)))`と同じ	

▼図6　リストのデータ構造とcar、cdr、cons



してリストのデータ構造の実装例を図6に示します。

図5でも紹介したnilはリストの終端を表す以外にも真理値の偽や、空リストを表すのにも使われます。

図6にあるように`(cons (car list) (cdr list))`がlistになることを確認してみてください。つまり、consとcar、cdrは反対の動作（逆関数）になっています。また`(cons 1 (cons 2 (cons 3 nil)))`→`(1 2 3)`となり、`(list 1 2 3)`→`(1 2 3)`はconsの簡略形になります。もちろんconsをこのように連続で書くのは不便ですので、普通はlistを使います。

Lispは記号処理、シンボルがすべてを扱う

❶ シンボルとは

Lispは記号処理用言語です。記号（シンボル）とはたとえば「`GOMI`」のようなものです。シンボルはいろいろな情報を持っているデータで、シンボルの`GOMI`の名前は文字列"`GOMI`"を持っていて、またシンボルは値を持つことができます。値を持つことから、ほかの言語の変数と同じ働きをします。

さらにシンボルは関数を持つことができます。これもほかの言語の関数と同じ働きになります。そしてシンボルは属性（プロパティ）とパッケージ（名前空間）も持ちます。ほかの言語の変数や関数と大きく違うことは、シンボルがデータであり、実行中に生成したり削除したりできることです。

❷ シンボルのネーミング

Lispのシンボルはほかの言語の変数よりもネーミング規則が自由で、以下のようなものはすべてシンボルとして使えます（例：`1+2`、`-1.0e`、`*`、`$`）。またシンボルに空白を入れたいときや小文字を入れたいときはエスケープ文字（\や|）で行います。

・例

```
'abc\ def → |ABC DEF|
'|abc def|→|abc def|
```

❸ シンボルの評価

シンボルを評価すると、その値が返りますが、シンボルそのものを返したいときはquoteを使

います。(quote gomi)や'gomiとすれば、シンボルgomiがそのまま値として返ります。

4 シンボルのインターン

多くのLispでは小文字で入力しても処理系のシンボルを管理するシンボルテーブル(図7参照)内部では大文字で登録されます。これを大文字インターンと呼んでいます。小文字で入力したシンボルgomiがいきなりGOMIのようになり、驚くでしょう。

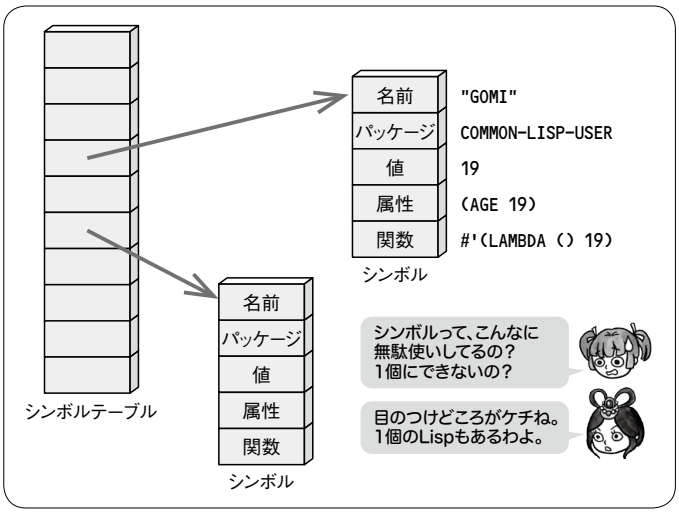
ここでインターン(intern)の説明をします。インターンとは言語処理系でシンボルテーブルにシンボルを登録することを言います。同じ名前のシンボルがすでにシンボルテーブルに登録されていれば、新たに登録せずに、既存のシンボルを返します。つまりシンボルテーブルの登録時に、シンボルの唯一性(同じ名前のシンボルは唯一であること)を保証します。これによりアドレスの等価性のみをチェックする高速な比較関数eqでシンボルの比較ができます。ちなみにJavaではメソッド関数のinternを文字列に対して実行すると文字列の唯一性が保証されるので==で比較できます。

5 シンボル関数

Lispの代表的なシンボル関数を表2に示します。またシンボルテーブルとシンボルの実装例を図7に示します。シンボルGOMIには値だけでなく、名前やパッケージ、属性(プロパティ)、関数も格納していることがわかります。

表2に(setf (foo x) y)のようなプログラ

▼図7　シンボルテーブルとシンボルの構造(実装例)



▼表2　Lispのシンボル関数(一部)

動作	入力	結果	備考
シンボルの名前を参照	(symbol-name 'gomi)	"GOMI"	大文字でインターンされていることに注意
シンボルのパッケージを参照	(symbol-package 'gomi)	#<PACKAGE COMMON-LISP-USER>	パッケージ
シンボルの値を代入	(setf (symbol-value 'gomi) 19)	19	
	(setf gomi 19)		シンボルは変数として使えます(setqでも可能)
シンボルの値を参照	(symbol-value 'gomi) または gomi	19	
シンボルの属性をセット	(setf (get 'gomi 'age) 19)	19	またはsymbol-plistでも可。GOMIの年齢属性は19
シンボルの属性を参照	(get 'gomi 'age)	19	またはsymbol-plistでも可
シンボルの関数をセット	(setf (symbol-function 'gomi) #'(lambda () 19))	#<FUNCTION :LAMBDA NIL 19>	
シンボルの関数を参照	(symbol-function 'gomi)	#<FUNCTION :LAMBDA NIL 19>	ひとつ上の関数セット後、実行した結果
シンボルの関数を実行	(funcall (symbol-function 'gomi))	19	または(gomi)

ムが出てきますが、これは`(foo x)`の左辺値(値を格納できる場所)に右辺値yの(その場所に格納する値)を代入するものです。たとえば、`(setf (symbol-value 'gomi) 19)`はシンボルgomiの値を格納する左辺値(場所)に19の値を代入することになります。これは`(setq gomi 19)`や`(setf gomi 19)`としても同じ結果になります。

一方C言語などにはシンボル型はなく、ソースプログラムにある変数(例：`int x;`)をプログラムの実行中に生成したり削除したりすることはできません。同じことをしたいときはCプログラムの文字列を生成し、それをファイルに書き込んで、そのファイルをコンパイルし、ダイナミックリンクしなければなりません。これはあまりにも面倒です。

「プログラム=データ」という世界

Lispの最大の特徴は何と言っても、プログラムとデータが同じ表現で、同じデータ型として扱えることです。ほかの言語ではソースプログラムとデータはまったく別次元のもので、それが同じというのは理解できないかもしれません。たとえば、Javaであれば、`void add(int x, int y){return x + y;}`というプログラムと`1, 2, 3`のデータが同じであるということです。

このようにプログラムもデータもリストで表されるために、リストを操作する関数は対象がデータであってもプログラムであってもまったく同じように実行できます。このプログラム＝データという特徴が後述する動的言語の源泉になり、Lispが柔軟な言語である理由です。

ちなみにJavaなどの仮想マシン上で動作する言語は、データとプログラムは同じバイトコード(コンパイル後の中間コード)で表されます。これは機械語と同じ感覚で「プログラム＝データ」として扱うことができ、Javaではリフレクションの機能でバイトコードレベルでの動的機能が一応備わっています。

Lispの基本はラムダ式

Lispの動作の基本はラムダ計算で、その表記であるラムダ式(Lispではラムダリスト)が用意されています。このラムダ式は再帰関数とともに関数型プログラミングの基本で、関数型言語だけでなく、JavaやC#、C++の一般的な言語でも最近導入されています。

ラムダ式の目的は、関数を定義した環境のまま、その関数を別の場所でいつでも実行できるようにすることです。ここで環境とはローカル変数(Lispではシンボル)とその値の集まりを言います。つまり、このラムダ式を一言で言うと「定義時の環境を持った匿名関数」です。

たとえば、Javaでは内部クラスは環境を持つので、内部クラスの匿名関数はすでにラムダ式と同じことができていました。Javaで導入されたラムダ式はこの内部クラスの匿名関数を使いやすくしたものになります。一方、Cの関数ポインタは関数をデータとしていつでも実行できますが環境は持ちません。

このラムダ式を用いて、関数をほかの関数の引数や返す値にする(これを高階関数と呼びます)ことができ、また関数の評価順序を変えたりできます。これがLispの柔軟さの源泉です。でも理解するのが面倒になる原因にもなっています。楽があれば苦があります。いえ、苦があれば楽になります。このラムダ式については次回に解説します。

動的で柔軟な世界、すべては無常

Lispの特徴として、動的(ダイナミック)なことがあります。動的とはプログラムの実行中にいろいろと変えることができることです。たとえば、実行中に関数の再定義や、クラスの再定義、変数(シンボル)自身の変更ができます。

次にシンボルの生成や関数の再定義などの動的なプログラミングを見ていきます(**図8**)。誌

面の都合で詳しくは説明しませんが、この手のハックができるのがLispです。まさにハッカー推薦の言語です。

このような動的機能が完備されていますので、難しいアルゴリズムの実装や人工知能、言語実装などでプログラミングするときに、各種の実験が簡単に行えます。まさにプロトタイピングと実験に向いている言語です。しかし一方でバグをデバッグするときには、この動的機能により複雑怪奇になるという面もあり、たいへんなことになるのは公然の秘密です。

また、この柔軟さと引き換えにプログラムの了解性と実行効率、デバッグを犠牲にしている面もほんの少しあるのは秘密です。

そしてLispのこれからは?

ここまでLispの古くて新しいいろいろな面とLispの歴史を振り返り、現在のLispとLispに影響を受けた言語を見てきました。

このように今でも新しい特徴を持つLispですが、これからはどのようになっていき、どのように使われていくでしょうか。今はIoTやビッグデータ、クラウドコンピューティングの隆盛により関数型プログラミングと柔軟で動的な言語が必要とされ、人工知能が花開く時代で記号処理の需要も増えています。

Lispの思想は今の時代には必要で、Lispに影響を受けた言語はこれからも重要な位置を占めているでしょう。これからLispを知り、学ぶことは必要になります。何よりもLispはマニアックでハッカー好みです。

次回はLispで語りたいこととして、関数型プログラミングや人工知能、オブジェクト指向について見ていきます。そのあと、Lispのプログラミングとして、再帰プログラミングやラムダ式などを紹介します。SD

▼図8　Lispの動的機能の例

・シンボルの生成とインターン

```
(intern "HIROSHI") → HIROSHI; NIL
```

入力文字列"HIROSHI"を名前とするシンボルを(既存でなければ生成し)、現状のパッケージのシンボルにします。インターンとは特定のパッケージに属するシンボルとしてシンボルテーブルに登録することです。またCommon Lispは多値を返すことができ、internは副値として、既存のシンボルだったかどうかを返します。今回の副値はNILであり、これはシンボルが既存にはなく、新たに生成したことになります。

・シンボルのアンインターンと削除

```
(unintern 'hiroshi) → T
```

シンボルをアンインターン(シンボルテーブルから削除)し、シンボルを削除します。

・関数定義

```
(defun fact (n) (if (<= n 1) 1 (* n (fact (1- n))))) → FACT
```

関数の新規定義や再定義は実行中に動的に行うことができます。

・関数定義を代入

```
(setf (symbol-function 'add) #'+) → #<SYSTEM-FUNCTION +>
```

シンボルaddの関数定義をシステム関数の「+」に変更します。前節のラムダリストも関数定義として代入できます。#<SYSTEM-FUNCTION +>はシステム定義の関数「+」であることを表しています。

```
(eval '(function (lambda (x y) (+ x y)))) → #<FUNCTION :LAMBDA (X Y) (+ X Y)>
```

上記で代入する関数「+」は静的定義のものですが、次は動的に関数を生成して代入してみます。

```
(setf (symbol-function 'add) 上記の式) → #<FUNCTION :LAMBDA (X Y) (+ X Y)>
(add 10 20) → 30
```

これは以下でも同じになります。

```
(eval '(defun add (x y) (+ x y))) → ADD
(add 100 200) → 300
```

RDBアンチパターン

PostgreSQLとMySQLの失敗と対策

Author 曽根 壮大（そね たけとも） ㈱はてな Twitter @soudai1025

第1回 データベースの迷宮

今月からはじまる本連載では、筆者が開発の現場で遭遇したデータベースにおける失敗と、そうならないためのベストプラクティスを紹介します。第1回は、設計に多くの問題抱えたデータベースを引き継ぐことになった新人エンジニアが主人公。

連載のはじめに

この連載では筆者が業務系、Web系のシステム開発を通じて見てきた「RDBアンチパターン」について話していきたいと思います。

「データベースの寿命はアプリケーションよりも長い」が筆者の持論です。なぜならば、データベースはサービス開発当初から存在することが一般的で、アプリケーションコードのようにリプレースされることは稀（まれ）だからです。また複数のサービスから参照されることも多々あり、その場合、最初に使われていたサービスが終了しても、データベースはほかのサービスとともに運用され続けます。

このように、データベースは長く付き合っていかねばならない相手であり、開発者はその特性ゆえの問題にぶつかることがあります。そういった、開発の現場で実際に起こっている、発生しやすいリレーショナルデータベース（RDB）全般の問題をRDBアンチパターンとして紹介していきます。

RDBは広く使われている反面、次のような注意事項があります。

- データベースの停止はサービスの停止を伴うことが多いため、メンテナンスしにくい
- データは常に増え続け、リファクタリングしにくい
- サービスの中核を担うため、変更による影響が大きい

このように、RDBのアンチパターンはアプリケーションのアンチパターン以上にダメージが大きいのです。

そして厄介なことに、RDBの問題はある日を境に顕在化するということが多いです。当初は良かれと思った設計が、あとになって問題を引き起こすこともままあります。しかし、誰かが経験したその問題をパターン化して共有しておくことで、初めのうちからその問題を避けられます。そういった点から、RDBアンチパターンを紹介することはたいへん有意義であると考えます。

本連載ではRDBアンチパターンを通して問題を提起し、多くの方に周知していただくことで問題を未然に防ぎ、現在の問題と戦っていくための1つの答えを提供できればと思います。

データベースの迷宮

今回はデータベースの不適切な名前付けや、構造が紐解けない設計について説明します。プログラミングでは度々話題になる名前付けやクラス設計ですが、データベースでも同じくとても大切です。しかし現場では次のような話をよく耳にします。

・memo1、memo2、memo3……と無限に続く、何が入っているのかわからないカラム
・「hoge_data」のタイピングミスで「hoge_date」になり、意味が変わってしまっているカラム
・中に入っている値の意味がわからないカラム
・外部キー制約がなく、リレーションシップがまったくわからないテーブル

これらのような例は笑い話ではなく、現場に散見されます。そのようなアンチパターンをここで紹介します。なおこのアンチパターンの事前知識として、RDBで設定できるおもな制約を**表1**にまとめています。

事の始まり

とある会社での、自社サービス開発中の営業担当者とエンジニアの会話。

営業：ごめんね。ここの項目1個増やしといてくれる？
エンジニア：○○の項目ですね、わかりました。
営業：あとこれ、削除ボタンも必要ね。
エンジニア：わかりました。
――3日後――
営業：あれ？　ここの項目足りないよ？
エンジニア：○○はありますよ。
営業：○○っていったら△△も普通一緒でしょ。タバコといったら灰皿も一緒に持ってくるでしょ？
エンジニア：あー……すみません、対応します(いや俺タバコ吸わんからわからんし1個って言ってたじゃん……)。
営業：これ、削除ボタン押したら管理画面からも消えるんだけど？
エンジニア：え？　当たり前じゃないですか。
営業：あー困る困る。削除はユーザさんから見えなくしてほしいだけで、管理画面では見える必要があるの。
エンジニア：……わかりました(削除じゃねーじゃん！)。
営業：それ、土曜日のイベントまでにリリースしたいからよろしく。
エンジニア：はい……(今、金曜日の19時だけど？)。

5年後のある日、新人エンジニア(以下A)は、この会社の自社サービスを改修していた。

A：よし、このdelete_flagをselectして……あれ？　エラーがでた……。

よく見てみると、カラム名はdelete_flagではなくdelete_falgだった。

A:もぅ！　名前くらいちゃんと付けてよね……。この場合は1が立ってると削除済みでいいのかな？　GROUP BYしてみよう。

▼表1　RDBにおける制約

制約の種類	説明
PRIMARY KEY制約	重複とNULLがなく、そのテーブルで一意な行であることを確定させる
NOT NULL制約	NULLがないことを確定させる
UNIQUE制約	その値がテーブルで一意であることを確定させる(NULLは許容される)
CHECK制約	指定した条件の値のみが保存されていることを確定させる
DEFAULT制約	値が指定されないときに保存される値を決める。それにより初期値を確定させる
FOREIGN KEY制約(外部キー制約)	別テーブルの主キーと参照整合性が保たれていることを確定させる

▼図1　delete_falgの中身をみると……

```
demo=# SELECT delete_falg AS delete_flag FROM users GROUP BY delete_flag
 delete_flag
-------------
           1
           2
           0
           9
           99
           NULL
(6 行)
```

▼図2　先輩が教えてくれたdelete_flagの仕様

```
0: 未削除
1: 削除済み
2: 管理者による強制削除
9: 抹消
99: よくわからない
NULL: バグで入る
```

そこには驚きの結果が(**図1**)。

A：2に9に……NULL？

腑に落ちないAさんは、さっそく先輩エンジニアのSさんに聞いてみることにしました。

S：これはずいぶん前に退職したエンジニアが1人で作ったアプリケーションなんだ。コードを読んだ感じだと、こんな感じかな(**図2**)。……そうだ！　Aさんがこのシステムの改修をやってよ。よろしくね！
A：……わかりました。

こうして、Aさんの長い旅が始まったのであった。

読み解けない苦しみ

さっそくAさんはツールを利用して、このシステムのER図を自動作成した。そこには外部キー制約がまったく設定されておらず、数百個のテーブルが並ぶだけであった。泣く泣くAさんはひとつひとつのテーブルの中身を確認し、リレーションシップを紐付けていくことに。そこでとある悩みにぶつかったのだった。

A：あれ？　このテーブル、detail_idってあるけどcustomer_detail_idなのかuser_detail_idなのかわかんない……。外部キー制約がないうえに名前で判断できないからコードを読まなきゃ……。

こうして、Aさんの工数はドンドン増えていった。

A：うーん、コードを読んでも、商品statusにはドキュメントもstatusマスタもないから、中の値が何を指してるかわからないやつがいる。

読めば読むほど難解になっていくデータベース構造。

A：テーブルがetc、etc2、etc3、yobi1、yobi2とあって、何に使われてるのかわかんない。コードをgrepすると使われ方が統一されてないし、yobi2は使われてない。でもDBにはデータがあるし……。
A：あぁ、このdetailテーブル、keyカラムの名前に紐付いてvalueカラムの値が変わってる。keyがageのときは年齢だし、keyがgenderのときは性別。コード上のカラム名をgrepしただけじゃ読み解けない……。

調べれば調べるほど難解になっていくデータベースのパズル。変更が怖くなってきたAさん。

この例の問題点

冒頭でも話したとおり、このような話は珍しい話ではなく、長期間、継続的に開発されたプロジェクトや、短い納期で開発された場合などに散見されます。大きな問題点としては、Aさんが直面していたように、次のような点があります。

- 不適切な名前では、データベースのテーブルの関連性や意図が理解できない
- リレーショナルモデルに基づいた設計をしていないと、既存の便利なツールを利用できない
- 保存されたデータが正しいかどうかが判断できない
- どのようなデータを保存し、どのようなデータを取り出せば良いかわからない

このようにデータベースとデータを読み解けないことで、改修が非常に難しくなります。場合によっては「バグなのか仕様なのか」の判断さえ難しくなり、挙動としては不適切なため、結局バグとなってしまいます。

delete_flagの例ですと、削除のコードを読んで1が削除なのか0が削除なのか、はたまた9が削除なのかを調べる必要があります。また、ブラックボックス化したデータベースに対する改修は影響範囲が読めないため安易に変更できません。しかしこのようなデータベース構造でも、そのサービスがビジネスを支えている以上、メンテナンスや改修が必要とされることが多々あります。

どうすれば良かったのか

では、どのようにすればこのような問題を解決できたのでしょうか。

読者の中には、エンジニアの素養よりもまず営業が悪いと言う方もいるかもしれません。もちろん、短納期かつ不適切な手順で仕様変更を強いることは、エンジニアを大きく消耗させますし、「まずは動くものを作ること」を優先してしまいます。この営業の立ち振る舞いやこのようなプロジェクトの進め方は是正されて然るべきです。しかし、「まずは動くものを作ること」の本質はとても大切なことですし、必ずしも否定的にとらえる必要はありません。当たり前だと言われるかもしれませんが、大切なことは、

動くものを作るときに適切に作る

ということです。

たとえば、delete_flagの命名が間違っていたことは途中で気づけた、または初期であれば直せたはずです。delete_flagの値にNULLや9が入っていた問題はCHECK制約を利用していれば防げた問題です。

また、もし本当に削除のflagを表すのであれば、PostgreSQLの場合はboolean型を使ってdeletedなどの名前を利用するのが最近の主流です。現場で稀に、外部キー制約やCHECK制約をかけずに「アプリケーション側でバリデーションすれば良い」と言う人もいますが、CHECK制約が守る対象は「アプリケーションのバグ」も含みますし、DDL[注1]からそのカラムの持つ意味を担保することも含みます。

「etc、etc2、etc3……」と続くテーブルについては、etc1ではなくetcという名前から「当初はetcしか作る予定ではなかった」ことが推測できます。つまりetc自体は正しい設計だった可能性が高いのです。何らかの仕様変更の際に、項目追加としてetc2を追加したのならば、そのカラム追加が不適切だった可能性があります。たとえばその時点でetcが複数個できるのであれば、hogeテーブルとhoge_etcテーブルに分けることも可能だったかもしれません。場合によってはmemoという別の名前カラムだったかもしれません。

データベースはよく積み木に例えられます。データの追加やカラムの追加のしかたで、次の

注1）Data Definition Language：データ定義言語。

変更に大きく影響がでます。etc2を作るときにhoge_etcと別テーブルにしておき、hoge_idをkeyとしていれば、etc3やetc4は不要なカラムになっていたでしょう。etc2を作ったことで、次のetc3、etc4を生み出すきっかけになってしまったのです。

この問題で考えなければならないことに、

何らかのやむを得ない理由から、将来に課題が残る方法を採用してしまったこと

があります。最近では技術的負債と言われたりします。

今回の例ですと、delete_flagや外部キー制約レス設計などが該当します。これらがなぜ技術的負債なのかについては、今後の連載で深掘りして説明していきますので今回は割愛しますが、このような「技術的負債」を「返済していく」ことも大切です。

繰り返しになりますが、データベースの寿命はアプリケーションよりも長いですし、場合によっては複数のアプリケーションから1つのデータベースが接続されることもあります。そのため技術的負債が積み上がり出すと改修しづらく、また次の負債を生みやすいのです。そうなる前に、早め早めに技術的負債を返済していきましょう。

RDB TIPS MySQLとCHECK制約

本編では、CHECK制約の重要性に触れました。しかし、MySQLにはCHECK制約がありません。しかも、CHECK制約を作るSQL自体はエラーになりません(**図A**)。これはとても注意すべきMySQLの仕様の1つですので、読者のみなさんもご注意ください。

▼図A　MySQLにCHECK制約の機能はないが、CHECK制約を作るSQLはエラーにならない

```
mysql> CREATE TABLE scores (
    ->   score INT NOT NULL,
    ->   CHECK ( score BETWEEN 1 AND 100 )
    -> ) ENGINE = InnoDB;
Query OK, 0 rows affected (0.03 sec)  ←エラーにならずテーブルができる

mysql> INSERT INTO scores (score) VALUES (1000);
Query OK, 1 row affected (0.00 sec)  ←エラーにならず保存される

mysql> SELECT * FROM scores;
+-------+
| score |
+-------+
|  1000 |
+-------+
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE scores;
+--------+-----------------------------------------------------------------------------+
| Table  | Create Table
|
+--------+-----------------------------------------------------------------------------+
| scores | CREATE TABLE `scores` (
  `score` int(11) NOT NULL  CHECK句が無視されている
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+--------+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

リファクタリングの例

たとえば、カラム名の変更が怖くて難しい場合などは、次のような手順があります。

❶変更後の名前のカラムを、新しい名前を付けて追加で作る
例）delete_flagを追加する
❷ ❶で作ったカラムは、トリガーを利用して変更前のデータと同じになるようにする
例）古いカラム名の「delete_falg」のINSERTやUPDATEのactionに対してトリガーを定義し、新しいカラム名のdelete_flagを同じデータにする
❸サービス単位やモデル単位で順に、参照や更新を追加したカラムに設定しなおす
例）参照・更新をdelete_flagに設定する
❹切り替えが完了して動作が問題なければトリガーと古いカラムをDROPする
例）トリガーとdelete_falgをDROPする

この手順は名著『データベースリファクタリング』[注2]で紹介されている手順です。このように、RDBの機能を利用して少しずつ変更する方法があります。

etcの例ですと、hoge_etcをetc2やetc3のように見えるviewを用意し、まずは更新から切り替えていくなどの方法もあります。

これまでの内容をまとめると次のとおりです。

・テーブルやレコードの中身がわかる適切な名前を付ける
・外部キー制約やCHECK制約を利用してデータを適切に防ぐ
・リレーショナルモデルに基づいた設計を心がける
・何らかの理由で課題の残る設計をした場合、早めに改修する

注2）スコット・W・アンブラー、ピラモド・サダラージ 著、梅澤 真史 ほか 訳、ピアソンエデュケーション、2008

このアンチパターンのポイント

今回紹介したデータベースの迷宮のアンチパターンは、ある日突然発生するというものではありません。少しずつ少しずつ蝕(むしば)んでいきます。しかし問題が顕在化するときは、今回の例のように数年後に新しい担当者に変わった際などに顕在化し、その対応に四苦八苦することになります。

1つの名前の付け間違いという些細なことでも、それがetc2のように次の問題を生む原因になります。また、外部キー制約やCHECK制約などは、開発チームの文化の問題が起因ということも少なくありません。このような問題は、その日には発生しない反面、解決には長い時間を要します。そのため、小さなところからコツコツと改善することが、とてもとても大切です。

これは割れ窓理論の、「建物の窓が壊れているのを放置すると、誰も注意を払っていないという象徴になり、やがてほかの窓もまもなくすべて壊される」と同様で、「データベースのオブジェクトに不適切な名前を付けて放置すると、誰も注意を払っていないという象徴になり、やがてデータベースそのものもすべて壊される」ことになります。

これらの問題を事前に防ぐコツは、

わかりづらい設計や名前はデータベースの破綻の始まり

と、常日頃から細心の注意を払うことです。

次回のRDBアンチパターン

今回のRDBアンチパターンはいかがでしたでしょうか？　次回はリレーショナルモデルでは難しい履歴データにまつわるお話となります。読者のみなさんも経験したことのある、背筋が凍るお話があるかも！？　次回の「失われた事実」もお楽しみに！ SD

RDB性能トラブルバスターズ奮闘記

原案 生島 勘富(いくしま さだよし) info@g1sys.co.jp ㈱ジーワンシステム
構成・文 開米 瑞浩(かいまい みずひろ) イラスト フクモトミホ

第15回 O/Rマッパを使っていいとき／悪いとき

O/Rマッパを使うべきか／使わないべきか、これはエンジニアの間でよく議論になるテーマです。明確な答えが出るものではありませんが、使うにしろ使わないにしろ、O/Rマッパは何のためにあるのか、代わりにどこに問題があるのかは、きちんと理解しておきたいものです。今回は大道君がズバリ、生島氏に尋ねてみました。

登場人物紹介

生島氏
DBコンサルタント。性能トラブルの応援のため大道君の会社に来た。

大道君
浪速システムズの新米エンジニア。素直さとヤル気が取り柄。

五代氏
大道君の上司。プロジェクトリーダーでもある。

O/Rマッパで起きがちなN＋1問題とは

大阪を中心に20年間システム開発に携わったあと、現在は東京で仕事をしている「SQLの伝道師」ことジーワンシステムの生島です。

「O/Rマッパって、どうですか？」

と、大道君がある日唐突に尋ねてきました。O/Rマッパ(以下、ORM)とは、RDBに使われるSQLと手続き型プログラミング言語との間のギャップ、いわゆるインピーダンス・ミスマッチと呼ばれる問題を解消するために手続き型(オブジェクト指向)言語側に作る、高機能なDBアクセス・フレームワークの一種です。

「どうですか」とだけ聞かれてもわからないので質問の背景を聞くと、最近、協力会社でのDB性能トラブルシューティングに駆り出されたところ、DBアクセスには基本的にORMを使用していて、いわゆるN＋1問題を起こしていたということです。

N＋1問題というのは、当連載でも何度か触れた「ぐるぐる系」SQLをORMが発行してしまう現象で、ORMがらみの性能トラブル原因の代表格です。例としては**リスト1**のようなものがあります。usersテーブルを取得するために1回、そこに含まれるuserごとにgroupsテーブルを取

▼リスト1　N＋1問題を起こすO/Rマッパを使ったコード

・Ruby on RailsのActiveRecordでN＋1問題を起こすサンプル

```
users = User.all()                    ← select * from usersを1回発行
for user in users
    print user.group.name             ← select * from groups where id =(userのid)をuser件数分(これがN回)発行
end
```

合計N+1回

・N＋1問題への対策バージョン(eager loading方式)

```
users = User.includes("groups").all()  ← select * from usersを1回発行してから、select * from groups where id in (...)を1回発行
for user in users
    print user.group.name              ← ループの中ではselect文は発行されない
end
```

得するためにN回で、合計N＋1回のSQLが発行されてしまうことからこの名前があります。この問題はプログラマが気づかないうちに起こりがちで、性能への影響も大きいことから有名であり、その分、対策も知られてきました。

「ええ、原因がわかったので対応はできましたけど、そのプログラマさんと話していてどうも違和感があったんです」

「何があったんや？」

「RDBとSQLをよく知らないらしくて、それに対策を教えたときの様子がなんだかコピペ(コピー＆ペースト)的で……」

「ああ、コピペ的ね……」

「こういうふうにすればいいですよ、とサンプルコード見せると、『あ、そうですか』とそのまんまコピペしてそれだけだったんですよ。お礼は言うんですけど、質問がなくて。普通はどうしてこれで改善されるんですか？と理由を聞きますよね？」

「普通はそうやね～」

「なので、聞かれていないけど理由も説明しました。でもわかってくれたような気がしません」

「そら、わかってないやろな～」

残念ながらコピペプログラマはそこかしこに存在しているので、珍しいこととは言えません。

O/Rマッパを使っていいとき／悪いとき

ORMについてはギャップ解消の救世主的な声もある一方で批判派も多く、私は基本的に批判派ですが全面否定するつもりはありません。私見でORMを使っていいときと悪いときを整理すると次のように考えています。

使っていいとき

- データ構造が簡単(テーブル数が10程度)
- NoSQLに移行する可能性がかなり高い
- 性能要求が低い
- SQLの教育がどうしても無理
- 主キーを使った単純な更新処理(追加／変更／削除)

使うべきでないとき

- 上記の各項目を満たさないもの
- 「SQLが理解できない」ことを理由にORMに頼ること

SQLはプログラミング言語の中で最も簡単

そもそもSQLがどのような性格のものなのかを**図1**にまとめました。ソフトウェアは何らかの用途のために作ります。「業務ドメイン固有データ」というのは業務ユーザが書くもので、たとえばExcelのシート、Wordの文書、Photoshopの画像のようなものがそれにあたります。そのデータを処理するアプリケーションや、そのためのライブラリはプログラマが書くもので、一般の手続き型言語を使って作ります。一方、RDBMSは「表形式(実際は関係モデルですが)のデータを操作する」という限定された用途に絞って使われるもので、SQL文は「多様な業務」の側で言えば「業務ドメイン固有データ」と「アプリケーション」にまたがったぐらいの位置づけになります。

ここで注目したいのは、「SQLはプログラマではない業務ユーザでも書ける」ということで

▼O/Rマッパの導入には賛否両論ある

す。一般のプログラミング言語は何でもできるポテンシャルを持っていますが、その分、表形式データの扱いはSQLよりも煩雑です。一方、SQLは「表形式データ」を扱うのに特化していて、「こういうデータがほしい」と、データへの要求をSQL文として書けば、実際にどこからどういう手順でデータを集めて処理するかという「アルゴリズム」はRDBMSが考えてやってくれます。その結果、

SQLはプログラミング言語の中でも最も簡単なもの

になっています。これが重要なところで、SQLを習得するのが難しいのならば「SQLが理解できない」ことを理由にORMに頼ってもいいと思いますが、本来簡単な言語なのですからそれを理解できない、というのはそれこそ理解できません。

実際、先日私は「営業マンも全員がSQLをバリバリ使いこなす」という会社を取材してきました。

- 営業さんまで、社員全員がSQLを使う「越境型組織」ができるまでの3＋1のポイント（リブセンス）
  https://www.slideshare.net/livesense/150225-sql-foreveryone-45695818

元技術者というわけでもない、完全に文系の営業でも普通に理解できるようになるのがSQLです。私が開いているSQL講座でも、普段Excelで仕事をしている事務職オペレータでも3日もあればSQLによるデータ操作は一通りできるようになります。それをプログラマが「わかりません」というのはオカシイのではないでしょうか。

SQLをわかったうえで、リザルトセット（DBから返ってきた検索結果）の、オブジェクト（アプリケーション側で処理をするための変数）への変換やSQL文生成ジェネレータとしての簡便さを活かすためにORMを使うというのであれば理解できますが、そうではないケースをよく見かけます。

「そうなんですよ。あの人はどう見てもわかっていない感じでしたから……」と大道君。「**リスト1**の修正点も、その修正でSQLがどう変わるのかは理解していないはずです。単にこういうおまじないをしとけば大丈夫、と人に聞いたからコピペしている感じでした」

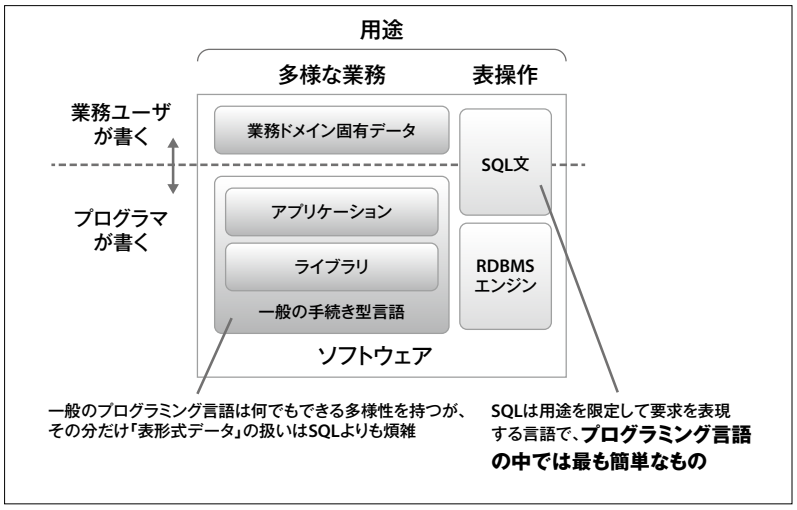
「わかっていないと、意味もわからんで使うおまじないになってしまうんよね」

「こうしてみると、**リスト1**って修正前も後もRubyコード上のループ構造のパターンはほぼ同じですよね。これじゃあ、これでどうしてN＋1問題が解決するのか、ピンとこないんじゃないかなあ……」

「同感！」

そこがまさにSQL初心者がORMを使うことへの違和感です。SQLを隠蔽（いんぺい）し過ぎると、結局SQLの本来の守備範囲である「表形式のデータを操作する」という感覚を身につけることが難しくなるように思いま

▼図1　プログラミング言語の守備範囲



す。それは長い目で見たときにIT技術者としてのスキル向上を阻害するのではないでしょうか。

インピーダンス・ミスマッチとは?

そもそも、インピーダンス・ミスマッチとはどんな問題なのでしょうか？

手続き型言語は多様なデータ構造に対する多様な処理（アルゴリズム）を表現することに向いた言語で、SQLは表形式のデータ構造に対する共通の処理（集合操作）を表現することに向いた言語です。この両者には「言語仕様」レベルでもギャップがあります（**図2**のAの部分）。たとえばリザルトセットとオブジェクトの間で変換が必要です。一方、そもそも主に想定しているデータ構造と処理パターンが違うというギャップもあります（**図2**のBの部分）。真のインピーダンス・ミスマッチはこのB部分の基本的な発想の違いです。ORMはAのギャップを解消することはできますが、それによってSQLの隠蔽を進めると普段SQLを使わないことになるため、Bのギャップは逆に拡大してしまうのではないでしょうか。

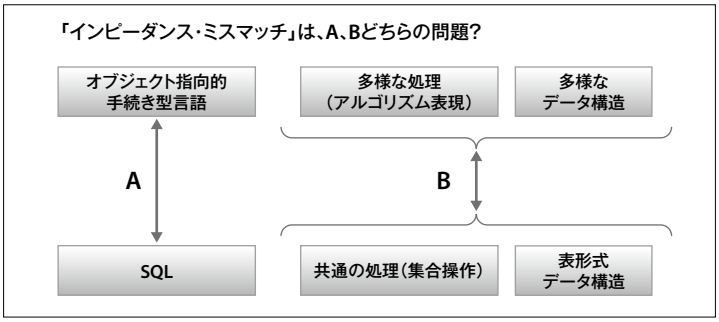
SQLは何を表現しているのか?

そもそもSQLは何を表現しているのでしょう？これを簡単にまとめると**図3**のようになります。

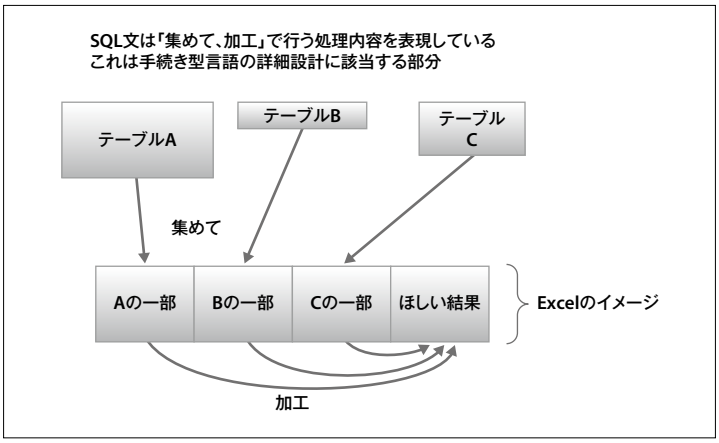
テーブルA、B、Cのように複数のテーブルから、関連のある必要な部分を切り出して集め、それを加工し、最終的にほしい結果を含めて1つの表にまとめるのがSQLのSELECT操作の本質です。ほしい結果を含む「1つの表」は前回（当連載第14回）で書いたようにExcelの表を作って考えるのが最もイメージしやすいものであり、SQL文自体はそこに至る「集めて、加工する」操作を表現しています。

通常のプログラミング言語は**図1、2**に記したように「多様なデータ構造に対する多様なアルゴリズムを記述」することが可能であり、その性質によって**図1**でいうライブラリからアプリケーションまでの幅広い記述能力を持ちますが、その分、扱いが面倒です。それに対してSQLは表形式データのマネジメントに限定して、アルゴリズムではなく「ほしいもの（要求）」を表現するだけで、それを実現する「アルゴリズム」はRDBMSが代わりに決定してくれる、実に簡単に使えるしくみなのです。

▼図2　インピーダンス・ミスマッチの正体は?



▼図3　SQLは何を表現しているのか?



SQLベースで開発すると設計書類を減らせる

そのため、SQLをベースに開発を行うと、必要な設計書類も本来は手続き型言語をベースにしたときに比べて大幅に削減できます。

図4にその比較表を示しました。「要件定義」の段階では両者同じになりますが、ロジックを手続き型言語で記述する場合には基本設計、詳細設計で膨大な書類が必要になり、さらに実装段階でもプログラミング言語のソースコードを大量に書かなければなりません。前回でも紹介しましたが、私の経験では要件定義で十数ページだった文書が基本設計・詳細設計で百ページを超え、実装されたコードはJavaで2万行に達したことがあります。その同じ機能をSQLで作りなおしたときには、数枚のExcelシートで基本設計が済み、詳細設計は3つのSQL文で済んだため、何十倍もの工数削減になり、かつ、性能もざっと100倍になりました。

結局、SQLならばDBエンジンが代わりに作ってくれる「実行計画」に該当する部分を、手続き型言語では自分で書かなければいけないわけです。とくに困るのは、それに慣れてしまうと、SQLを使うときも手続きの感覚で実行計画のようなSQLの使い方をしてしまうこと。つまりそれが「ぐるぐる系」だったり、「場合分けの多用」だったりします。

SQLを理解してORMを使うなら問題はないが

お弁当屋さんでフロントが注文を受けて厨房(ちゅうぼう)が作るケースで例えましょう。フロントがアプリケーション側、厨房がDB側に相当します。この店が「餃子ランチ10食、酢豚ランチ10食」の注文を受けたとします。

SQLの発想ならその1行の注文書を厨房に渡してその2種類・20食を作り分け、全部できたところでフロントに送りますが、手続き型発想だと「餃子ランチ10」「酢豚ランチ10」と2回に分けて注文を出したり、「餃子ランチ1」の注文を10回、「酢豚ランチ1」の注文を10回出したり、あるいは「餃子10、酢豚10、飯20、小鉢20」のようにパーツごとに注文を出してフロント側でそれをランチとして組み立てたりといったことをやってしまいます。このときに必要なのが

▼手続き型言語でDBを扱うのはこんなイメージ?

▼図4　開発で必要な設計書類

	手続き型言語	SQL
要件定義	・求める仕様を言葉と図表などで表現したもの	
基本設計	・I/O関連図 ・画面イメージ ・計算式　など	・Excelのイメージ ・API仕様書
詳細設計	・参照／更新するテーブル、カラムの決定 ・具体的なアルゴリズム	・SQL文
実装	・プログラムソース ・(PHP、Java、Rubyなど)	・SQLの実行計画

※両方に必要なテーブル定義、ER図などは省略

「ループ処理を使うアルゴリズム」で、手続き型言語ではこれを詳細設計に書いたうえでプログラミング言語で実装します。しかし、SQLならそれはDBが作る実行計画なので人間が書く必要はありません。

厨房が「複数の食材を集め、調理して1つのランチを組み上げる」作業はつまり、DBで言えば「複数のテーブルから関連するデータを集めて1つの請求書を作る」作業です。この種の仕事は本来DBのほうが得意ですが、そのためのSQL文は複雑なものになります。しかし、ORMは複雑なSQLを作るのには向いていませんので、ORMに頼った開発をしていると簡単なSQLですべてを済ませてしまい、性能も出ないしSQLへの理解も上がらず、無用なトラブルを引き起こす結果を招くのです。

もちろん、SQLをきちんとわかった者がORMを使うなら適材適所の使い分けができますが、現実には「DB側のほうが得意なロジックをアプリケーション側で処理することを助長する」傾向のほうが目立ちます。

ORMは実行計画もどきのSQLを助長しやすい

「結局こういうことなんですかね……」と大道君が**図5**を書きました。

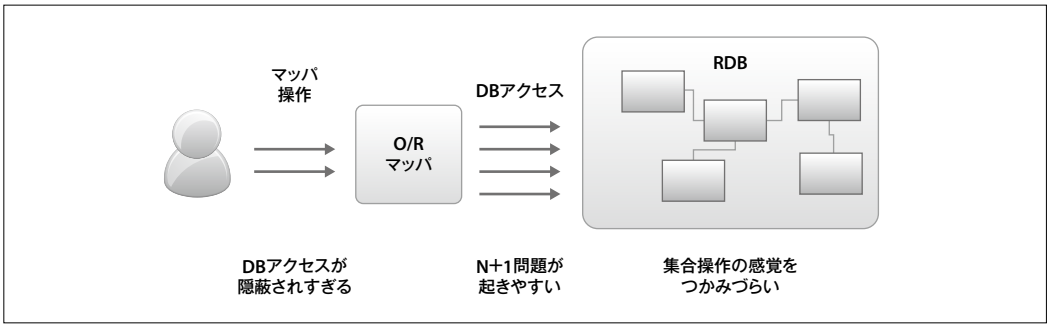
「ORMはDBアクセスを隠蔽してくれる便利なしくみですけど、隠蔽し過ぎるといつどこでDBが呼ばれるのかも読み取りづらくなるため、N＋1問題が起きやすい。しかも、隠蔽しているもんだから、RDBの基本である集合操作の感覚もつかみづらい。そうすると発想が手続き型のままだから、SQL初心者がORMを使って書くコードは手続き型でもないしSQL的でもない中途半端なものになって、RDBの真価を発揮できないしメンテナンスもしづらいものになってしまう……ということでしょうか？」

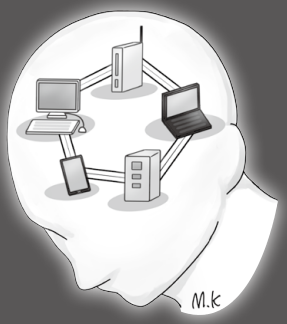
「そういうこっちゃ。インピーダンス・ミスマッチと言っても、解消すべきはコードの書き方のミスマッチじゃなくて、頭の中の発想のギャップなんよ。そこに目をつぶって『SQLが理解できない』ことを理由にORMに頼るのは、その問題を固定化するようなもので、とてもお勧めできないね」

とはいえ、ORMの代わりに「文字列をベタベタ結合してSQLを動的生成する」というORM以前のよくあるやり方に戻るのはそれこそ面倒ですし、当連載第8回[注1]で触れたようにSQLインジェクションも誘発します。代わりに第9回[注2]や前回で触れた、ストアドプロシージャを使い、DB担当を分けて分離開発を進める「APIファースト開発」がお勧めです。APIファースト開発については随時勉強会[注3]も開いていますので、興味のある方はぜひおいでください。SD

注1）本誌2016年10月号。
注2）本誌2016年11月号。
注3）https://api-first.connpass.com/

▼図5　ORMの欠点





仮想化の知識、再点検しませんか?
使って考える仮想化技術

本連載は「仮想化を使う中で問題の解決を行いつつ、残される課題を整理する」ことをテーマに、小さな仮想化環境の構築・運用からはじめてそのしくみを学び、現実的なネットワーク環境への実践、そして問題点・課題を考えます。仮想化環境を扱うエンジニアに必要な知識を身につけてください。

Author
笠野 英松(Mat Kasano)
オフィス ネットワーク・メンター
URL http://www.network-mentor.com/indexj.html

第12回 仮想環境の運用管理(1)

連載後半「仮想ネットワーク環境で使ってみよう〜現実的な使い方」の6回目です。

今回は、今までの連載の中で見て、考えてきた、いろいろな例のまとめとして、仮想環境全体や仮想マシンの運用管理の課題を考えていきます。

仮想環境における個々の仮想マシンは実際の環境では、個人単位の利用ではなく、部署・部門などの単位で利用することが多いと考えられます。したがって、その運用管理も1人、あるいは、1ヵ所で簡単にできるとは思えません。

そこで、ホスト物理システムと仮想マシンを分けて、それらの運用管理の担当や内容について考えてみましょう。

運用管理のポイント

最初に考えるべきポイントは、対象となる「ホスト物理システム」「仮想環境全体」「個々の仮想マシン」のさまざまな運用管理に対して、次の2つがあります(**図1**)。

1つは、運用管理を“担当する人間”です。つまり、運用管理者を1つの部署(たとえば、運用管理部門)で行うのか、仮想マシンを利用する人間(あるいは部門の人間)が行うのか、ということです。

もう1つは、運用管理を“どこで行うのか”ということです。ローカル、つまり、ホスト物理システムで行うか、リモートから行うのか、の2つが考えられます。

運用管理に必要な技術と担当者

まず、前者の運用管理を担当する人間(の技術)について考えてみましょう(**図2**)。

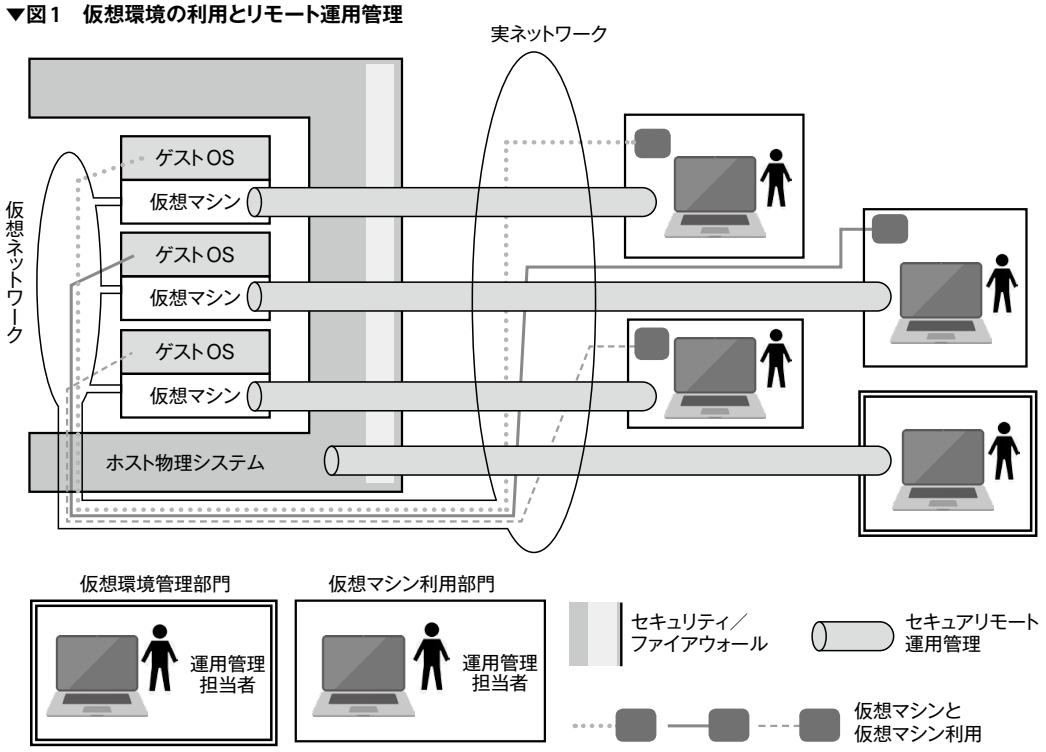
仮想化の実際の環境では、その管理処理の面で、また、利用・運用管理処理の面で複雑・多様な技術の必要性が増してきています。

たとえば、仮想化インフラや仮想マシンではOS/ネットワークとして、少なくとも、汎用UNIXやUNIX互換OS、そしてWindows(サーバ、クライアント)が使用される可能性があるので、担当する技術者もこれらのOSおよびそのネットワーク技術についてのかなりの知識が必要となります。

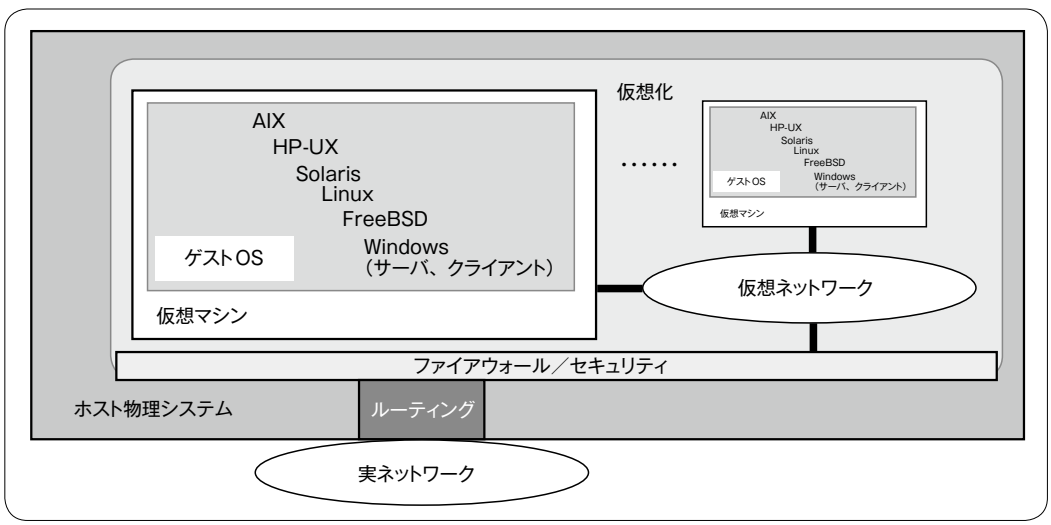
しかし、1人の技術者がこれらすべてに精通していないとすれば、AIXに1人、HP-UXに1人、Solarisに1人、Linuxに1人、FreeBSDに1人、そしてWindowsに1人、などのように個々のOS担当管理技術者が最低限必要になります。そうすると、仮想環境の管理チーム全体としてはかなりの人的リソースを食ってしまいます。

また、仮想環境を維持・運営するためには、その構築設定・運用管理はもちろん、仮想ネットワーキング/ルーティングの技術や、仮想化インフラと仮想マシンへの利用・管理システム以外からのアクセスを防止するためのファイア

▼図1　仮想環境の利用とリモート運用管理



▼図2　仮想環境の技術



ウォールを含むセキュア管理を徹底するしくみが必要になります。

さらに、仮想マシン自体の運用管理はもちろん、初期化やシステム停止・再起動などの「擬似」ハードウェア制御のしくみやシステム異常終了時の「ごみ」のリセットを含む仮想マシンの復元処理なども必要です。

以上のような仮想環境の維持・運営や仮想ネットワーキング／ルーティング、セキュア管理、仮想マシン自体の運用管理、ハードウェア制御、

仮想マシンの復元処理などに手操作で対応するとなれば、こちらもかなりの人的リソースを割かねばなりません。商用仮想環境では、個々のネットワーク・システムを詳細に運用管理するために高度に対応した、高価な商用の仮想環境やネットワークの統合運用管理システムで自動化しています。

一方、中小規模の仮想環境では、管理者任せにシステム化されたり、“パワーユーザ的”あるいは“趣味程度”の管理環境であったり、実・仮想を区別するのではない一般的なネットワークを対象とした汎用のネットワーク統合運用管理に頼っているにすぎないように見えます。

その原因は、コスト削減に偏重した仮想化環境の利用や、仮想ネットワークと実ネットワークを含む全体ネットワークの運用管理にばかり目がいっていること、また、個々の、ホスト物理システムや仮想環境、仮想マシンなどの統合運用管理・制御管理の面で問題点・課題が整理されておらず、置き去りにされていることによると思われます。

そうすると、ホスト物理システムや仮想環境全体はともかく、個々の仮想マシンだけは「そのシステムに精通しているはずの」仮想マシン利用者にその運用管理作業を任せるほうが、コスト(管理人件費)的にも技術的にも合理的であるように思えます。

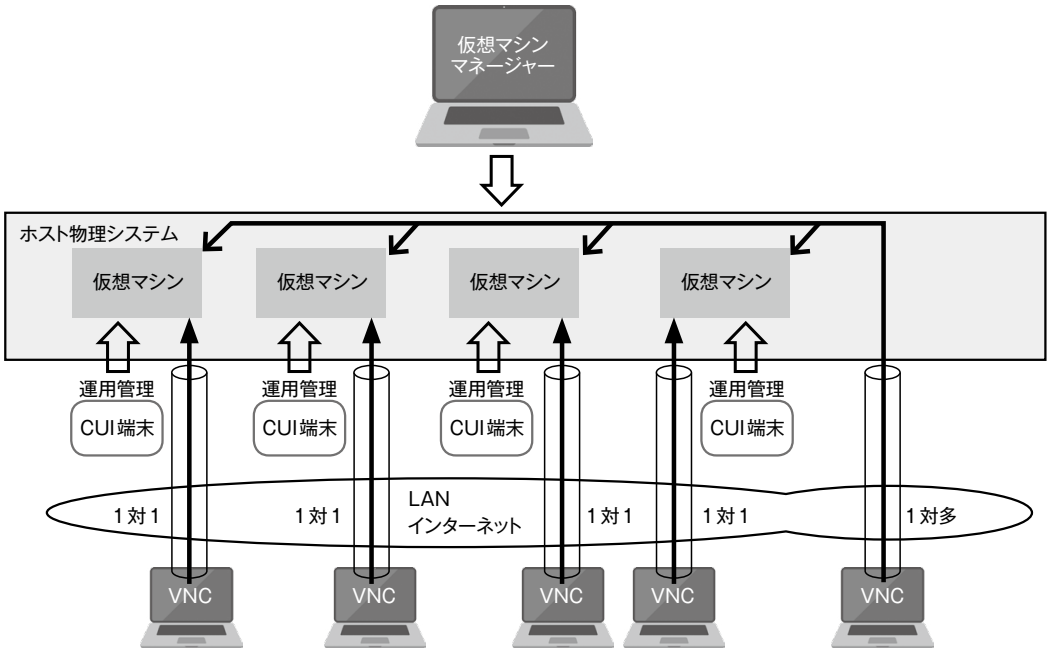
もちろん、仮想マシン運用管理作業のすべてを仮想マシン利用者に任せることは困難かもしれませんが、その可能性、実際には、可能性の範囲を見極めることが大事です。

ローカル管理かリモート管理か

次に後者の、ローカル管理かリモート管理か、を考えます(**図3**)。仮想マシンの数が増えて、前者のように「仮想マシン運用管理の個々の(利用者による)管理」を行おうとすると、複数あるいは多数の管理者が、入れ替わり立ち替わりローカルで操作することは現実的ではありません。もちろん、非同期端末を複数つなぐことでこれを回避することは可能ですが、いちいち利用者部門の場所からホスト物理システムの近くにやってきて運用管理操作するのはたいへんです。

したがって、リモート管理が適当ではないか

▼図3　仮想マシンのローカル／リモート運用管理



ということになります。

ただし、ここで注意すべきは、リモートからの運用管理のためにアクセスする接続についての最大限の信頼性――最大限のセキュリティ――を確保しなければならないということです。ローカルで集中的に管理している場合にはローカルという場所自体で一定のセキュリティが保たれていますが、リモートとなるとセキュリティはないのも同然です。

そこで、これに対するセキュリティ強化対策をどう取るか、が課題になります。

運用管理作業項目

運用管理の作業項目については、連載の第1回(2016年5月号)「仮想化の現状を見てみよう／仮想化環境の運用管理」で最初に概説していますが、具体的な項目は記事末にある**図4**のようなものです。

(1)(一般的な)システム運用監視作業

この項目の中で中心は「自動化」です。この自動化については、ホスト物理システムや仮想マシンのインストールが対象で、連載の第7回(2016年12月号)「ホストシステムと仮想環境の構築／自動インストール」で具体例を示しています。

(2)(一般的な)ネットワーク管理

これらの項目は仮想環境においても適用されます。

(3)仮想化管理

これについては連載第1回の「仮想化環境の運用管理と図3(仮想化管理の位置づけ)」で概説しています。

運用管理の要件

次に、リモートおよびセキュリティ、自動化、ホストと仮想マシンの運用管理の統合化と分散化、という中で現実の処理の要件を考えていきましょう。

統合化と分散化

どの機能を分散化し、どの機能を統合化するかは、まず、どのような運用管理機能があるか、そして、その作業を管理担当者(とくに、利用部門の運用管理者)が可能か／すべきか、を明確にすることが出発点です。

仮想マシン個々か全仮想マシンか

先述のように、仮想マシンの運用管理はできるだけ仮想マシン利用者に任せることが真のコスト(人件費)削減、技術合理化につながります。また逆に、全仮想マシンの運用管理を1人に集中することも可能にしておくと、運用管理の柔軟性が保たれます(**図3**)。

リモート運用管理

仮想マシンの運用管理はリモートから行いますが、仮想環境(ホスト物理システム)もリモートから可能にしておくことで、その担当者の運用管理作業の場所が広がります。

また、このリモート管理のためにvirt-managerをはじめ、virshなどKVM付属の専用ツールがありますが(次回で解説予定)、それらをそのまま使用することは一部のKVM専門技術者以外には馴染みが薄く、使いづらいものです。したがって、使い慣れたWebブラウザを通すことが前提条件です。

一方で、先述のように、こうしたリモート運用管理のセキュリティを保持し、強化することは至上命題となります。

仮想マシンの運用管理機能や分担

仮想マシンの運用管理機能は、ホスト物理システム管理者が可能な／処理すべき機能、利用者が可能な／処理すべき機能、利用者運用管理の標準最低限の機能、オプション機能、不可能

な機能、などに分類されます。

ホスト物理システム周辺装置の共用

ホスト物理システムの物理周辺装置、ネットワークインターフェース、ディスクストレージやUSBデバイス、などは共用ですので、その排他的制御を行うしくみも必要になります。

とくに、仮想マシン上のゲストOSインストールで共用するisoファイルの格納ストレージ、仮想マシンで追加するためのディスクストレージ/パーティション、USBメモリなどのリムーバブルデバイス、の仮想マシンへの連携のしくみなどは重要です。

仮想環境(ホスト物理システム)と仮想マシンの障害管理

トラブルシューティング処理として、事前、検出・検知、回避、解決、の処理・対策が必要です。具体的には、たとえば、仮想マシンのバックアップ/リストア、仮想マシンの状態監視、仮想マシン異常トラブル回避・対応、などです。

構成/セキュリティ/アカウント/ディレクトリ情報管理

仮想マシン/ゲストOSとリンクした、仮想マシンのMACアドレスやIPアドレス、リモート接続VNCポート番号や仮想マシン利用者の情報(運用管理アカウント名/パスワードなど)やゲストOSの情報、さらにはセキュア接続の証明書/キーなど、個々の仮想マシン情報データベースとして保持し、かつ、仮想マシン利用者が設定、変更・編集可能なようにしておくことも重要です。

コスト矛盾

以上のようなしくみや要件、機能などに対応する作業・処理を人的にこなそうとすると、かなりの「専門技術者」が必要になります。これは、コスト削減のために導入した仮想化がコスト増(人的コスト)をもたらすということになってしまいます。

逆に、「人的にこなす」のではなく「高度な運用管理システム」で自動対応しようとすると、中小規模の仮想化においては、「不相応な」コスト増に陥ってしまいます。

いずれにせよ、つまるところ、デッドロックです。

次回予告

今回は紙幅の都合でここまでの説明になります。次回は引き続き仮想環境の運用管理(2)として、各種管理ツールの利用方針などを解説します。

次回で説明する運用管理のしくみは、KVMで標準的に用意されている詳細なインフラやインターフェースとしての機能・技術です。これらはKVMに精通した技術者が知っている(知っているべき)もので、一般的な(KVM以外の)技術者には縁遠いものです。そのため、このまま使うとすれば、専門技術者の必要性がさらに増すと思われます。

そうすると、運用管理者の数はさらに増すのでコスト(人的コスト)の矛盾(コストを下げるための仮想環境の運用管理が、逆にコストを上げることになる)がますます大きくなります。

次回はそのあたりまで踏み込んでいきます。

SD

連載各回では、読者皆さんからの簡単な「ひとくち質問(QA)コーナー」や「何かこんなこともしてほしい要望(トライリクエスト)コーナー」を設けて「双方向連載」にできればと思っていますので、質問や要望をお寄せください。

宛先:sd@gihyo.co.jp
件名に、[仮想化連載]とつけてください

▼図4　仮想化環境の運用管理

(1)（一般的な）システム運用監視作業

①通常業務／ルーティン

日常業務	自動化
環境整備	物理的、組織的な環境整備
保守	明確な責任体制のもとでの保守
教育	技術習得などの定型化された教育指導

②障害対応

事前	稼動監視、性能監視
事後	検知・分離・原因解析・修正・復旧

(2)（一般的な）ネットワーク管理

①構成管理（Configuration Management）

・ネットワーク資源の構成の変更や、データベース内に格納される構成情報の管理、資源の動作状態の制御など

②障害管理（Fault Management）

・ネットワーク資源の障害を検知／検出し、障害情報からその原因を分析／解決した後に正常状態への回復を行う

③性能管理（Performance Management）

・ネットワーク資源の性能や効率の統計的管理、テスト、測定、接続性・機能性のチェックなど

④セキュリティ管理（Security Management）

・ネットワークおよびその資源への不正アクセス対策、対災害、信頼性対策など

⑤アカウント管理（Accounting Management）

・ネットワークおよびその資源の利用した記録を元にした情報収集や記録、統計的管理、課金

⑥ディレクトリ管理（Directory Management）

・ネットワーク資源の名前やアドレスなどの属性を相互関連付けた資源管理

(3)仮想化管理（連載第1回の図3「仮想化管理の位置づけ」参照）

一般的な仮想化管理は「仮想化インフラと仮想マシンの、個々およびそれぞれの間の、性能や障害、リソースなどの運用管理」であるが、実際には、下記のような4つの運用管理がある。

①階層化管理

・仮想化インフラと仮想マシン、仮想マシン接続端末、そして、仮想マシン利用者という4階層の縦（上下）および横（階層内）の運用管理

②仮想ネットワーク管理

・仮想マシン間ネットワーク管理、ホストマシン・仮想マシン間、および、実・仮想ネットワーク間、の連携ネットワーク管理

③セキュリティ管理

・実・仮想ネットワーク間、そして、VMM（Virtual Machine Monitor、仮想マシンモニタ）とVM（Virtual Machine、仮想マシン）間のアクセス制御

④仮想化環境と仮想マシンの運用管理

・仮想環境の運用管理=管理者、仮想マシン上のシステム（サーバ、デスクトップ）の運用管理および利用=「利用者自身運用管理」で効率化

※「**利用者自身運用管理**」：利用者自身による利用仮想マシンの運用管理。

コミュニティメンバーが伝える Androidで広がるエンジニアの愉しみ

presented by
Japan Android Group
http://www.android-group.jp/

第14回 最新Androidのイマ~MWCから見るAndroidとOとコミュニティ~

Androidは世界で出荷される約9割のスマートフォンに搭載される標準OSです※。そのため、多くのAndroidアプリが開発され続けており、そして多くのエンジニアが活躍しています。Androidで広がる新しい技術に魅了されたエンジニアが集うコミュニティもあり、そこでは自分が愉しむための技術を見つけては発信しています。その技術の一幕をここで紹介します。

※Gartner Worldwide Smartphone Sales to End Users by Operating System in 3Q16

嶋 是一(しま よしかず)
NPO法人
日本Androidの会
理事長

唯一無二の環境 Android

Androidは世界の9割近いスマートフォンに搭載され、まさにスマートフォンの標準OSとなっています。Android上で動くアプリケーション(アプリ)を開発したならば、実に世界の9割の人の手の中で、自分が開発したプログラムを動かすことができます。これほど多くの端末で動作させることができるのは、パソコンOS、組込みOSなど、これまでのいかなるプラットフォームにもなかった環境です。プログラムでアプリを開発する側としては、たいへんにありがたい、人類最大のアプリを動作させる環境だといえます(ちょっとおおげさすぎますか?)。

そのAndroidも進化しています。直近では、3月22日にAndroidの最新バージョンである「Android O(オー)」のデベロッパプレビュー1版(DP1)がリリースされました。

また、Androidの本体以上に進化が早いのは、Android周辺の技術です。たとえば、IoTに関する技術や、AI、VR/AR/MR、ドローン、スマートウォッチなどなど、このような新しい技術は、必ずどこかでAndroidの助けを借りて発展しています。なぜなら、新しいイノベーションは、世の中のスマートフォンを介して、広く普及させたり、多くの人に知らせたり、開発者を増やしたりしているからです。スマートフォンが介在した時点で、まさにその9割はAndroidなのですから、まるでAndroidは新技術のゆりかご(または孵化装置)のような役割を担っています。

たとえば、AIの技術自体はAndroidと関係ありません。しかし、Androidで取得したデータを利用することはあります。また、クラウドで学習させたモデルを再びAndroid上に持ってきて、学習結果をAndroidで利用する(本誌2017年2月号本連載参考)、つまりAIの結果をAndroid上で活用することもあります。このような使い方が広がり、開発する人も増えてくると、さまざまな人のアイデアで新しいAIの使い方が生み出されます。まさに、機械学習がAndroid上で育まれていくこととなります。

しかし、そのような新しい技術情報を、常に自分がアンテナを張ってキャッチするには、たいへんな労力が必要となります。個人でできる範囲には限界があります。そういうとき役立つのが、コミュニティです。コミュニティ活動は有志の人が集まって、勉強会を開催したり、開発イベントで集まったりしている活動となります。コミュニティ構成メンバーの中には旺盛な興味を持って新しい調査と開発を常に続けている人たちがいます。一緒に活動することにより、最新情報がどんどん入ってきます。しかも、情報が入ってきたと同時に、それが「良いものか」「悪いものか」を判断する目利き力も鍛えられることとなります。これは会社の企業内活動でも、

個人の趣味活動においても、とても役立つものとなります。そういう、自分のアンテナ磨きとして活用するコミュニティは、この変化の速いご時世に「適(かな)った」活動となっています。

MWC2017に見るAndroid状況

Androidが搭載されるスマートフォンも多く展示される、世界最大のモバイルの展示会「MWC(Mobile World Congress)」が毎年2月にスペインのバルセロナで開催されています。今年は2月27日から開催されました。毎年世界中の通信キャリア(電話会社)、そしてスマートフォンや通信装置などを扱うメーカー、それらでサービスを運用するサービサーなどが集結しており、その年のモバイル事情を予測するためにも、重要なイベントとして注目されています(**写真1**)。

今年のMWC2017では、モバイル通信方式「LTE」「4G」の次世代通信方式である「5G」が展示会の中心です。

当然ですが5G通信を利用するには、スマートフォンが欠かせません。そのとき使われるOSの多くはAndroidであり、その上のアプリを用いて5Gのサービスを使うこととなります。

その陰で、Androidはスマートフォンに限らない端末側の開発OSとして広がっているのを多く確認できました。SONYが発表したタッチ可能なプロジェクタ「Xperia Touch」(**写真2**)はOSにAndroidを搭載しており、来場者を驚かせていました。

今回の展示会は「これでもか」というくらいの新型のVRゴーグルが、世界中のメーカー、世界中のキャリアから出展されていました。ゴーグルの表示部にスマートフォンを入れて利用する「モバイル型VRゴーグル」だけでなく、スマートフォンも外部PCも必要なく、ゴーグル単体で動作する「オールインワン型ゴーグル」が数多く発表されていました。そして実はそれらもAndroidベースで作られているものがほとんど(**写真3**)。しかも、ほぼスマートフォンと同じ「Qualcommのハード+Android+アプリ」の構成で作られているものが目立っていました。

そのためVRの上で動作する「VRアプリケーション」は、Androidアプリとして配信されています。いよいよ、Androidアプリ開発者の出番ということです。

今回のVR展示でおもしろかったのが体感デバイスです。VRゴーグルを着けて見ているだけだと、右にカーブした映像を見ても、体は静止したままなので違和感を感じます。展示されていた「コックピットのような筐体」に座ると、遊園地のアトラクションのように、椅子が360度ぐるぐる回ります(**写真4**)。これで視覚だけでない体験ができるようなります。まさにこの様相が、遊園地のアトラクションそのもの。これらの技術もAndroidでできたVRゴーグルの延長にあると思うと、興味深いものがあります。

▼写真1　MWC2017入り口

▼写真2　AndroidのXperia Touch

▼写真3　Android VRゴーグル

▼写真4　SAMSUNGの360度回転できる体感筐体

MWC2017の会場にはAndroid展示ブースはないのですが「Android村」と名付けられた広い休憩スペースが設けられています。休憩スペースだけになかなかメディアでは紹介されないのですが、そこを訪れた人は、Googlerとハイタッチ(**写真5**)すれば無料で飲み物がもらえたり、スタンプラリーをするとグッズをもらえたりと、アミューズメントパーク感たっぷりです。たとえば「パートナーウォーク」というコーナーでは、Androidが搭載されているメーカーの展示場所が示されており(**写真6**)、そこに行くとピンとシールがもらえ、集めることでスタンプラリーが行えます。90種もあるスタンプをコンプリートするとグッズがもらえるといった「Androidをキーワードにお祭りをする」イベントに、多くの人が参加して、ちょっと技術とは違うAndroidを楽しんでいました。

▼写真5　ハイタッチ

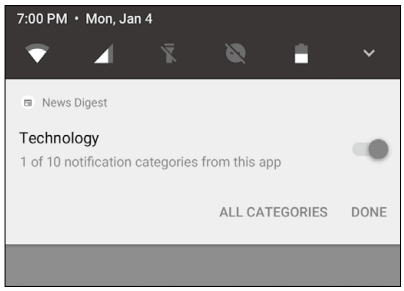
▼写真6　会場のピン配布場所

DP1に見るAndroid状況

Androidの進化はバージョンアップで行われます。先月3月22日にAndroidの最新バージョンが発表されました。今回のバージョンは「O」です。ゼロではありません。「オー」です。Androidのバージョンにはお菓子の名前が付くのが慣例です。しかもそのお菓子の名前の先頭文字は、アルファベットの順番で決められています。前回のバージョンが「Nougat」というお菓子で、先頭文字が「N」でしたので、今回の先頭文字は「O」というわけです。

ただ、お菓子の名前が付くのは正式バージョンになったときですので、今回発表された「デベロッパプレビュー(DP)」にはまだついていません。そのため、DPはすぐさま市販のAndroid端末で使えるというものではなく、PC上と一部のNexus/Pixelでのみ動作させることができます。DPとは開発者向けの事前評価というもので、アプリケーションを開発している人が、事前に新しいAndroidでも不具合なく動作できるかどうかを、検証するためにGoogleがリリースしています。このあとDP2、DP3、DP4までリリースし、そののち、今年のQ3に正式リリースとなる予定です。そ

▼図1　新しくなったノーティフィケーション



のときにバージョン名が決まるのでしょう。オレオ、オレンジピール、オカキ、お餅、お菓子(?!)、何になるのか楽しみです。

今回のバージョンアップでは**図1**にあるような、画面上部の通知(ノーティフィケーション)が変更されています。また、フォアグランドで動作するアプリケーションを優先に動かし、バックグラウンドはできる限り省電力のために動かさないよう、ブロードキャストインテントが受信できなくなっています。動かなくなるアプリケーションが多数発生しそうなので、気になる方は早めに試して、対処するのがよいでしょう。

コミュニティに見るAndroid状況

繰り返しになりますが、最新の技術情報を入手するアンテナとして、コミュニティで活動するのは良い方法です。国内には複数のAndroidのコミュニティが活動しており、筆者が運営している日本Androidの会もその1つです。今後本連載でコミュニティメンバーがリレー方式で執筆し、コミュニティイベントだけでなく、もっと多くの人とAndroid技術を共有させてもらいたいと思っています。

1人で開発するよりは、多くの人と接しながら開発したほうが、新しいアイデアも出てくるでしょう。イベントを開催するときに、自分の作品をアピールできる機会も増えるでしょう。やはり「新しいおもしろい技術」を知ることは、開発するための「モチベーション」を上げることになります。それがコミュニティで活動するための糧となります。日本Androidの会では、5

▼図2　ABC2017Spring

月28日土曜日にAndroidの祭典である「Android Bazaar and Conferene 2017 Spring」(ABC 2017Spring)を開催します[注1]。今回は「MR/VR、AI、IoT時代のユーザ体験、現在、過去、未来」というテーマで、東海大学高輪校舎(品川駅徒歩)に集います(**図2**)。

開催当日に来場者として来ていただけるとうれしいですし、ぜひコミュニティに参加するきっかけとして、ABCの実行委員スタッフとして参加してみてはいかがでしょうか。

このようなコミュニティ活動を続けていくコツは、参加するのは無料奉仕だと思わないことです。参加するからには、活動した結果の何か(楽しさ、知識、親睦等々)持ち帰るものを見つけることです。知識豊富で開発能力が秀でている人でも、一方的に情報を発信するだけであれば、やはり長続きしません。発信した、何らかの対価を見つけて、持ち帰るモノコトを見つけられる人のほうが長続きされるようです。

Androidの今後

Androidはさまざまな技術とともに、多くの人々の手で多様性を保ったまま進化を続けています。これはオープンソースであることも一因しています。今後もスマートフォン以外のデバイスに多数搭載されると思います。筆者もこの発展の一助を担い、多くの人にAndroidの開発にチャレンジしてもらえるとうれしいです。SD

注1) http://abc.android-group.jp/2017s/

一歩進んだ使い方のためのイロハ

Vimの細道

第18回

mattn
twitter:@mattn_jp

GVimを知る

今回はGUI版のVimである「GVim」を取り上げます。GVimの概要、できること・できないことを紹介したあとは、GVimの見た目を自分好みにカスタマイズするための設定方法を解説します。Vim初心者の方は、まずはこのGUI版から始めてみてはいかがでしょうか。

Vimというと、多くの方は端末上で動作するCUI(Character User Interface)のものを想像すると思いますが、VimはGUI(Graphical User Interface)版の「GVim」も提供しています。

GUIだからといって、操作感が大きく異なるものではありません。等幅のキャラクタの集合で作られたインターフェースで、見た目はCUIと同じです。メニューやツールバーだけでなくフォントもきれいにレンダリングされ、color schemeもフルカラーで表示されます[注1]。

当然のことながら、GVimにはCUI版のVimにはない機能がいくつかあります。今回はこのGVimを使ううえでのメリットと注意点をいくつか紹介したいと思います。

GVim入門

GVimのしくみ

Vimの実装では、「物理的なキー入力を受け付ける部分」「キーイベントに対する処理」「その処理結果をレンダリングする部分」が明確に分離されています。GVimは、CUI版のVimと同様に起動時にvimrcを読み込み、そのあとGVim専用の初期化ファイルgvimrcを読み込んでGUIを表示します。CUI版のVimはpty(疑似端末)から読み取ったキー入力をキーシーケンス処理に流し込み、map(キーマッピング)などを解釈します。GVimも同様に、WindowsのAPIやGTKといったGUI部品から得た仮想キーコードを端末のキーコードに置き換え、端末と同じような動作を行います。ですので、GVimの動作はCUI版のVimの動作とほとんど変わりません。

GVimの各種実装

GVimは、OSやGUI実装別にいろいろなバージョンが存在します(**表1**)。以前はKDE向けに実装されたkvimやyzisというforkも存在しましたが、最新のVimには追従できていません。Linuxにおいてはたくさんの GUI実装がありますが、今でもメンテナンスされているのは現状GTK2、GTK3、Athena、Motifくらいになります。オフィシャルでサポートしているPhoton GUIは、最近ではほぼメンテナンスされていません。各Linuxディストリビューションで提供されるGUI版のVimは、おおよそ次の名称と

▼表1　Gvimのバージョン

OS	GUI実装
Windows	Win32 API
Linux	GTK2、GTK3、Athena、NeXtaw、Motif、Photon
macOS	MacVim

注1) 最近の端末では、CUIのVimを使う場合に限ってTrue Colorで表示することもできます。

なっています(ここではUbuntuでの名称)。

・vim-gnome
・vim-athena
・vim-gtk
・vim-gtk3

vim-gtkはGTK2版のVim/GVimを指します。実はvim-gnomeとvim-gtkには本質的な違いはありません。vim-gnomeはGNOMEライブラリに依存し、vim-gtkはGTK2に依存しています。kubuntuなど、非GNOME環境を使われているのであれば、vim-gtkを選んだほうがインストールされる依存物が少なくなるはずです。

GVimでできないこと

GVimはGUIで動作するので、当然のことながら端末の中では動作しません。ですので、X11 Forwarding[注2]のようなGUIを転送する機能がない環境では、リモートにログインしてGVimを使用することはできません。

また、一般的なGUIアプリケーションで当然のように認識できるはずのキーが、GVimでは認識できません。たとえばGVimでは、CUI版のVimと同様に、Ctrl-;をマップすることはできません。GUIのイベントとして発生させることはできるのですが、マップできないのにはちゃんと理由があります。

GVimはCUI版のVimの動作をエミュレートしています。端末の文字入力でShiftはASCII入力コードの6ビット目を落とすキーにアサインされています。つまり、a(0x61)はA(0x41)というキーになります。また、CtrlはASCII入力コードの7ビット目を落とすキーにアサインされています。ここで、A(0x41)にCtrlを組み合わせると0x01、よってヘッディング開始になり、またH(0x48)にCtrlを合わせると0x08、つまりBackspaceになります。一般的な端末でCtrl-Hをタイプすると、バックスペースと同じ働きをするのはこれが理由です。

;(0x3B)はもともと7ビット目が0ですので、Ctrlを合わせても何も変わりません。ですので、一般的な端末エミュレータではCtrl-;は使えません。これと同じ理由で、GVimでもCtrl-;は扱うことができません。

GVimでできること

GVimの利点はやはり見た目の美しさではないでしょうか。メニューやツールバーだけでなくフォントもきれいにレンダリングされます。フルカラーのcolorschemeを使えるので、コードのシンタックスハイライトもきれいに表示されます。筆者の場合はメニューやツールバーを無効にすることで少しでも速く起動するようにカスタマイズしていますが、これらの機能をユーザが好きなようにカスタマイズできるのも、GVimの特色と言えるでしょう。

GVimのカスタマイズ

フォントの設定

GVimはCUI版のVimとは異なり、Vim専用のフォントが設定できます。前述のように、GUI関連の設定はgvimrcで行います。gvimrcのファイル名は、UNIXの場合は$HOME/.gvimrc、Windowsの場合は%USERPROFILE%_gvimrcになります。

```
set guifont=Ricty Diminished Discord:h11
```

これはWindowsの例ですが、LinuxのGTK2/GTK3版ではPango/fontconfigというライブラリを使ったフォント指定になります。次は、GTK2/GTK3でのフォント設定です。

```
set guifont=Ricty\ Diminished\ Discord 11
```

また:set guifont=*を実行すると、GUIのフォント選択画面を使ってフォントを選択できます。これをgvimrcに反映したい場合は、gvimrc

注2) X Window Systemの描画をssh経由で転送する機能(X11はX Window Systemのバージョン11を表す)。

を開いてset guifont=まで入力したあと、Ctrl-rとタイプし、続けて=&guifontを入力すると、現在のguifontの値が入力されます。その際、空白を含むフォント名は\（バックスラッシュとスペース）でエスケープする必要があります。あとは保存すれば、次回の起動からそのフォントが有効になります。

もし、ASCII文字とマルチバイト文字のフォントを個別に設定したい場合は、guifontwideを使用します。このオプションが空の場合は、guifontと同じフォントでマルチバイト文字が描画されます。

なお、フォントの行間が狭く感じる場合は、linespaceオプションで行間を調整できます。

```
set linespace=1
```

スクロールバーやツールバーのカスタマイズ

GUIに関するカスタマイズはguioptionsオプションに設定します。

```
set guioptions+=a
set guioptions+=A
```

のように、+=でほしい物だけ追加することもできます。

◆a：オートセレクト

このフラグを足すと、ビジュアルモードが開始されたり選択した領域が変更された際に、選択されたテキストが自動で*レジスタに格納されます。*レジスタはOSのクリップボードと連携しているため、テキストを選択したと同時に内容がクリップボードに格納されるようになります。

◆A：オートセレクト

aのオートセレクトに似ていますが、モードに関係なく、選択した領域やCtrl+Shiftとマウスでテキスト選択した領域がクリップボードに格納されます。

◆P：＋レジスタ

aのオートセレクトと同様に動作しますが、*レジスタではなく＋レジスタが使われます。

＋レジスタは、X11 Selection[注3]専用です。WindowsやmacOSでは*レジスタと同じものになります。

◆c、v：確認ダイアログ

cフラグを付与すると、ファイルの保存確認などのダイアログ表示に、GUIではなく文字による候補選択を使うようになります。vフラグを付与すると、GUIの確認ダイアログでボタンが縦に並ぶようになります。

◆e：Tab UI

VimのタブがGUIで表示されます。

◆f：Fork

通常、シェルからvim -gを実行するとGVimが起動し、シェルから切り離されます。しかし、GVimをシェルと連携して扱うために、切り離さずに実行したい場合もあります。vim -gfを実行するとシェルから切り離されずにGVimを起動できます。guioptionsにfフラグを足すとそれと同様に、シェルからgvimコマンドを実行した場合でもシェルから切り離されずに実行されます。これはたとえば、$ls | gvim -のように標準入力を編集する際に便利です。

◆i：アイコン表示

Vimのアイコンを表示します（一般的なウィンドウマネージャでは左上に表示されます）。

◆m：メニュー表示

メニューを表示します。このフラグは次に述べるMフラグに影響しますが、$VIMRUNTIME/menu.vimが読み込まれているならば、GVim起動後も変更できます。

注3）X Window Systemの持つクリップボード機能。

なおメニューは、リスト1のように自分で新しい項目を追加することもできます。ここでは、「vimrcを開く項目」と「vimrcを再読み込みする項目」を設定しています。

▼リスト1　メニューに新しい項目を追加する設定（gvimrc）

```
menu Commnads.Edit\ \.vimrc      :e $MYVIMRC<cr>
menu Commnads.-sep-              :
menu Commnads.\.vimrcを再読み込み :so $MYVIMRC<cr>
```

◆M：$VIMRUNTIME/menu.vimの読み込み無効化

メニューを表示しないのであれば、$VIMRUNTIME/menu.vimを読み込む必要はありません。このフラグを付けてメニューを読み込まなくするか、次の変数を設定してメニューの読み込みを無効化することができます。

```
let g:did_install_default_menus = 1
```

◆g：メニューのグレー表示

たとえば、ペーストする文字列がないときには、メニューのペーストコマンドはグレー表示されるべきです。このフラグを付けると、状態に応じたメニューの無効化が行われます。必要ないという方は、このフラグを抜いてしまってもかまいません。

◆t：メニューの切り離し

Vimのメニューは、メニュー内に表示される破線をクリックすることで切り離しができます（**図1**）。この切り離しを有効にするにはtフラグを付与します。

◆T：ツールバーを表示

このフラグを付与するとツールバーが表示されます。実はGVimのツールバーでは、アイコンを変更できます[注4]。

◆r、R、l、L、b、h：スクロールバー

水平垂直スクロールバーの表示、およびスクロールバーの幅の制御を行います。

・r：ウィンドウの右側に縦スクロールバーを表示
・R：縦分割されたウィンドウの右側にスクロールバーを表示
・l：ウィンドウの左側に縦スクロールバーを表示
・L：縦分割されたウィンドウの左側にスクロールバーを表示
・b：水平スクロールバーを表示。サイズは一番長い行に依存
・h：水平スクロールバーのサイズを、現在カーソルがある行の長さとなるように制御。これにより計算量が減る

▼図1　メニューの切り離し

まとめ

今回はGVimの解説を行いました。TrueColorを発色できてフォントが変更できるGVimは、設定しだいではきれいなユーザインターフェースを実現できるため、デスクトップ環境で使っていると、「そのエディタなんですか？」と言われることもたまにあります。良い設定ができあがったらぜひ公開して、皆と共有しましょう。今回紹介し切れなかったカラースキームなどについては、今後また紹介していきたいと思います。

SD

注4）URL http://mattn.kaoriya.net/software/vim/20130909212224.htm

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://rubikitch.com/

最終回 Emacsの学び方とエンジニアとしての成長

35回続いた本連載ですが、今回で最終回となります。Emacsの次の学びにつながるよう、今回はEmacsの信頼できる情報源を紹介します。またエンジニアとって必要不可欠な「英語」の学び方についても紹介します。筆者の「学び」に関する熱意を感じてください。

今月で最終回

ども、るびきちです。今回で連載「るびきち流Emacs超入門」は最終回となります。これまでついてきてくださり、誠にありがとうございます。あなたのEmacs力が少しでも向上すれば、筆者として最高の喜びです。本連載の草稿は筆者のサイトに追々全文掲載しますので、復習にお役立てください。

最終回にふさわしく、今回は、

・Emacsの有力な情報源
・Emacsで英語を楽しく読む方法
・Emacsは不朽であること

をお伝えします。

Emacs優良情報源3種

Emacsの最新情報を追い掛けるにあたって、有用な情報源を3つ挙げます。

最新版Emacs日本語マニュアル

1年ほど前まで、Emacsの日本語マニュアルは前世紀に翻訳されたものしかありませんでした。Emacs初心者が基本的な操作を覚える程度にはまだ存在価値はありましたが、情報はとても古く、実用性に乏しかったです。

ところが2016年、ayatakesi氏によりEmacs 24.5、25.1のマニュアルが翻訳されました[注1]。Emacs 25.2は25.1のバグフィクス版ですので、25.1のマニュアルでも十分に通用します。Emacsについて深く学ぶには必携です。

Emacs Advent Calendar

毎年12月1～24日に行われる技術系Advent Calendarがあります。年一の大イベントであるため、執筆者の気合の入りようはものすごいです。Emacs Advent Calendarは2009年から始まりました。「Emacs備忘録」というサイト[注2]に2009～2016年版がまとめられています。

Emacs News by Sacha Chua

世界有数のEmacs女子、Sacha Chua氏は毎週Emacsの動向をウォッチするEmacs Newsをブログに投稿しています(**図1**)[注3]。おもに次のカテゴリに分類されたリンク集です。

・Navigation
・Org Mode
・Configuration
・Coding
・Emacs Lisp

注1) URL https://ayatakesi.github.io/
注2) URL http://wp.hebon.net/emacs/?tag=advent-calendar
注3) URL http://sachachua.com/blog/category/geek/emacs/emacs-news/

・Emacs development
・Other
・New Packages

Emacsの最新情報を追い掛けたければ、ここさえ見ておけば問題ないです。新しい使い方から開発状況まで把握できます。彼女はとくにorg-modeのヘビーユーザだけに、org-modeに対するアンテナには目をみはるものがあります。

英語をEmacsで快適に読む

話は変わって、Emacsは英語を快適に読むのに適した環境です。たとえ苦手であったとしても、心理・システム双方のアプローチで克服できます。

一次情報が鉄則

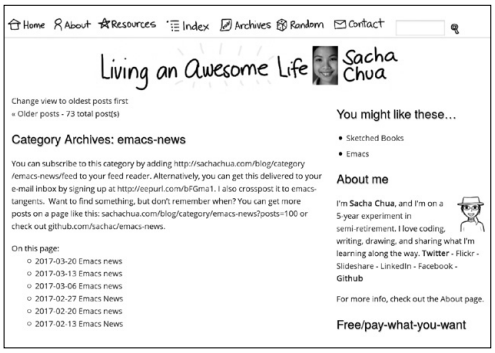
エンジニアにとって、英語は不可欠です。英語は事実上世界共通語であり、最新情報を得るためには英語の情報に触れることが不可避です。

Emacsについてもあてはまっています。有志が翻訳してくださった日本語マニュアルはとてもありがたいですが、往々にして情報が古いです。前述のとおり、最新版のEmacs日本語マニュアルが登場しましたが、それ以前の日本語マニュアルはEmacs21.3が対象と、10年以上前のものでした。そのため、最新のEmacsを体系的に学ぶには書籍か英語マニュアルを参照するしかありませんでした。

書籍やネットは、現在にはあてはまらない古い情報であることも多々あります。正確な情報を得るには一次情報に触れるのが鉄則です。Emacsに限らず、ソフトウェアの一次情報はやはり英語マニュアルとソースコードです。一次情報を避けていては、成長しません。

恥ずかしながら、筆者もつい最近まで英語が苦手でした。しかし、筆者は心理的アプローチとシステム的アプローチを併用して苦手を克服しました。

▼図1　Emacs News



心理的アプローチ

こう言われると驚くかもしれませんが、世の中に最初から「苦手」と意味がついているものは存在しません。「苦手意識」という言葉が象徴するように、「苦手」とはあなたが勝手に作り出した思い込みに過ぎません。確かに英語が苦手ならば、他人よりも英語を読解するまでに時間がかかるでしょう。

けれども、こうは考えられないでしょうか？

あなたは間違いなく、英語を習い始めたころよりは英語を読み書きでき、成長しています。苦手意識の原因は、他人と比較しているところにあります。比較の対象は過去の自分であるべきです。過去の自分よりも成長していると実感できれば、自然と苦手意識は消えていきます。

そのためには、習慣的に英語に向き合うようにする必要があります。最初は3分でもかまいません。毎日英語に向き合うようにすれば、継続できる自分に自信が出てきて、楽しくなってきます。実際に筆者はfishシェルのマニュアル全訳[注4]を毎日自分に課すことによって、苦手意識を消しました。

システム的アプローチ

これで終わりだと、単なる精神論になってしまいますが、Emacs使いはここからが違います！

筆者は「どうすれば英語を楽々読めるようにな

注4）URL http://fish.rubikitch.com/

▼図2　横長で書かれた改行のない英文

```
Emacs is the extensible, customizable, self-documenting real-time display editor.
```

▼図3　改行をいれて読みやすく

```
Emacs is
the extensible,
customizable,
self-documenting
real-time display editor.
```

るか」を考えました。英語が理解しづらい原因を突き止め、Emacsで解決する方法を編み出しました。

英語力が低い人は、一度に理解できる単語の“塊”の数が少なく、理解するのに時間がかかります。理解している間に目線が前後して、横長で書かれた英語の海に視線が迷子になってしまうのです。横長で書かれた少ない改行の文章が読みづらいことも、英語の理解を妨げます。

そこで、マウスを使い、理解できる範囲で改行を入れるアイデアを思いつきました。たとえば、**図2**ならば、**図3**のように改行を入れていきます。改行によって目線が迷子になることもなく、マウスクリックがリズムをもたらし、英語を読むのが楽しくなってくるはずです。Web上の英文ならば、Emacs内蔵Webブラウザ`M-x eww`で英語に集中できます。

詳しくは筆者のブログ[注5]を参照してください。

Emacsは永久に不滅です

Emacsそのものは40年、GNU Emacsは30年を超える歴史を持ち、今なお活発に開発が続けられています。平成の世の中は流れがとても速く、ましてやIT業界ではそれが顕著です。その中で30年もの間、継続的にメンテナンスされていることは、驚愕に値します。現存する古代文明とでも言えるのではないでしょうか。

Emacsはやりたいことがすぐに具現化できるすばらしいソフトウェアです。その一因として、拡張言語にLispという稀有（けう）な言語を選択したことがあります。Lispは単純な構造でありながら、いろいろな考え方のプログラミングが実現できます。「プログラマはLispを学べ」という昔ながらの格言があります。Lispの考え方は多くの現代プログラミング言語に取り入れられ、Lispを学ぶことで発想力が豊かになります。

とはいえ、実務でLispを使うことはほとんどないのが現状です。隙間時間にCommon LispやSchemeを学ぼうにも、日常に忙殺されると時間・体力・気力を捻出するのに苦労します。

でも、Emacs Lispはどうでしょうか？

学べば学ぶほどEmacsへの理解が深まり、より快適な環境へと進化させることができ、生産性が向上するのであれば、学ぶモチベーションになるはずです。

生産性の高い人は、改良グセがあるため、常により良い方法を模索しています。Emacs、そしてEmacs Lispには理想を具現化する力があります。筆者も「こうすればうまくいきそうだ」と思ったら、サッとEmacs Lispプログラムを書いています。長年改良を繰り返してきた筆者のEmacsは仕事、そして人生の大黒柱になっています。

あなたもご自分のEmacsを子育て感覚で育てていってください。あなたの成長に比例して、Emacsも成長していきます。Emacsを立派な「娘」に育ててください。

Emacsは永久に不滅です！

さらにEmacsを学ぶには

最後に宣伝させてください。

筆者のサイト「新生日刊Emacs」ではEmacsの

注5）URL http://emacs.rubikitch.com/english-reading/

情報をほぼ毎日配信しています。またネット塾「るびきち塾」を運営し、メルマガ「Emacsの鬼るびきちのココだけの話」を毎週土曜日に発行しています。Emacsに関する話題を中心に、読者のリクエストに応じて発行しています。

るびきち塾では、入塾者からの個別メール相談を無制限に受け付けています。Emacsの設定がわからない・うまく動かないなどのトラブルにテキパキ対応しています。それだけでなく、あらゆる分野の人生相談にも知識の貯蔵庫をフル活用して真摯に向き合って、感謝されています。

お互いに真剣になるために、「るびきち塾」は月々527円の有料サービスとなっています。初月無料ですので、筆者から直接学びたいのであれば、気楽に登録してください。

→http://emacs.rubikitch.com/juku/

Emacs Lispについては、2016年12月からまったくの初心者でもEmacs Lispを実践を通じて理解できるようにひとつひとつ解説しています。2016年12月からのバックナンバーを購入していただけると、着実にEmacs Lisp力が付くことでしょう。

連載の終わりに

筆者は自分を成長させるために、「学び」に膨大な時間と資金を投入しています。心理学、脳科学、自己啓発、食事法、サプリメント、トレーニング、文章術、パフォーマンスアップ、世の中の原理原則など多岐に渡ります。情報を音声化してICレコーダーで速聴し、起きている間はずっと学びの時間にするくらいの気概で生きています。学びに溢れた生活は楽しくてたまりません。

最後まで読んでいただき、ありがとうございます。SD

書いて覚える Swift 入門

第25回 Anything is nothing

Author 小飼弾（こがい だん）　twitter @dankogai

Swift on FreeBSD

本題に入る前にニュースを1つ。実はFreeBSDでもSwiftは動きます。ports（FreeBSDのパッケージ管理システム）にlang/swiftが登場していたことはFreeBSD Journal 2016年11/12月号[注1]にも取り上げられていたのですが、正直きちんとメンテナンスされているとは言えず、本記事執筆現在でもビルドエラーで止まってしまいます。

しかしビルドエラーの内容を精査してみると、問題はSwiftのソースコードではなく、ビルドが依存しているツールがアップデートされているのにそれにあわせてportsが更新されていないのが原因と判明したので、その点を微修正してビルドした後パッケージ化したものを公開しました[注2]。

同パッケージはもともとevalpark[注3]のためにビルドしたものです。evalparkというのは任意のシェルスクリプトをフルセットのFreeBSD 11で実行してJSONで返すというすごいWeb APIで、これだけでまるまる別記事を書けてしまうのですが、それはともかく同サービスを使うとWeb上でSwift on FreeBSDを実行できます（**図1**）。

FreeBSD上で実行している証拠に、次のコードはちゃんと"I'm running on FreeBSD"と出

▼図1　evalpark

```
#!/usr/local/bin/swift
extension Int {
  var fizzbuzz:String {
    switch(self % 3, self % 5) {
      case (0, 0): return "FizzBuzz"
      case (0, _): return "Fizz"
      case (_, 0): return "Buzz"
      default:     return String(self)
    }
  }
}
_=(1...30).map{ print($0.fizzbuzz) }
```

run　clear

```
droppings
    /tmp/org.llvm.clang.65534/ModuleCache/2OST40E2VCKYP/SwiftShims-27RP82Q7FPK
    /tmp/org.llvm.clang.65534/ModuleCache/2OST40E2VCKYP
    /tmp/org.llvm.clang.65534/ModuleCache
    /tmp/org.llvm.clang.65534
elapsed
    0.164241790771484
error
park
    park7
path
    /friends/2017/03/20/8e23dec0189f44b8feb03ffb254feeaf36b33df0ec
status
    -1
stderr
stdout
    1
    2
    Fizz
    4
    Buzz
    Fizz
    7
    8
    Fizz
    Buzz
    11
    Fizz
    13
    14
    FizzBuzz
    16
    17
    Fizz
    19
    Buzz
    Fizz
    22
    23
    Fizz
    Buzz
    26
    Fizz
    28
    29
    FizzBuzz
```

注1）URL https://www.freebsdfoundation.org/past-issues/programming-languages/
注2）URL http://blog.livedoor.jp/dankogai/archives/52022072.html
注3）URL https://eval.dan.co.jp/

力します。

```
#!/usr/local/bin/swift
var os = "(mac|i|tv|watch)OS"
#if os(FreeBSD)
os = "FreeBSD"
#elseif os(Linux)
os = "Linux"
#endif
print("I'm running on \(os).")
```

Swiftのバージョンは1世代前の2.2.1ですが、本連載のコードはほとんどそのまま動くので、読者のみなさんもお気軽にお試しください。

Anything is nothing

それでは本題。前号の終わりで筆者は「Anyは避けるべきです。少なくとも、動的言語の変数のように使うべきでありません」と述べましたが、なぜなのでしょうか?

一言で言えば、「Anyには何でも入るが何もできないから」ということになります。次のとおり、確かに何でも入ります。

```
var a:Any
a = true
a = 42
a = 42.195
a = "Everything"
a = [true, 42, 42.195, "Everything"]
a = ["answer":a]
```

しかし、そのままでは何も使えません。たとえば最後の状態でa["answer"]としても、出てくるのは[true, 42, 42.195, "Everything"]ではなく`error: type 'Any' has no subscript members`というエラーだけです。「期待どおり」の結果を出してもらうには、(a as! [String:Any])["answer"]と型を指定しなければならないのです。これでは型を省略したことにはなりませんよね。

メモリ消費量の点からも、Anyは避けるべきです。C言語のvoid *やObjective-Cのidとは異なり、Anyには本来の値に加えて型情報も収納されています。それゆえ安全なのですが、それゆえ余計な情報も抱え込んでいることになります。

```
MemoryLayout.size(ofValue: 0)           // 8
MemoryLayout.size(ofValue: "")          // 24
MemoryLayout.size(ofValue: [0])         // 8
MemoryLayout.size(ofValue: ["":0])      // 8
MemoryLayout.size(ofValue: 0 as Any)    // 32
```

それでもAnyという「無能な万能型」がわざわざ用意されているのには理由があります。たとえばSwift 3 の Foundation の JSONSerialization.jsonObjectの戻り値はAnyですが、これによりBoolもDoubleもStringもArrayもDictionaryもすべてカバーできます。が、その分何を取り出すにも`as`しなければならずとても不便です。

JSON型を作ってみる

そんなときはどうすればよいか。もうおわかりですね。型を作ってしまえばよいのです。JSONならばこんな感じですか。

```
enum JSON {
    case JSNull
    case JSBool(Bool)
    case JSNumber(Double)
    case JSString(String)
    case JSArray([JSON])
    case JSObject([String:JSON])
}
```

こんな感じで初期化できます。

```
let json0:JSON = .JSObject([
    "null":.JSNull,
    "bool":.JSBool(false),
    "number":.JSNumber(0),
    "string":.JSString(""),
    "array":.JSArray([
        .JSBool(true),
        .JSString("string"),
        .JSNumber(42.195)
        ]),
    "object":.JSObject([:])
])
```

が、こんなの筆者だって使いたくありません。Any?から初期化できるようにしましょう。Anyではなく、Any?であるのはJSON(nil)でJSON.JSNullを返したいから。

```
extension JSON {
    init?(_ any:Any?) {
        switch any {
        case nil:
            self = .JSNull
        case let number as Double:
            self = .JSNumber(number)
        case let int as Int:
            self = .JSNumber(Double(int))
        case let bool as Bool:
            self = .JSBool(bool)
        case let string as String:
            self = .JSString(String(string))
        case let array as [Any?]:
            self = .JSArray(
                array.map{ JSON($0)! }
            )
        case let dictionary as [String:Any?]:
            var object = [String:JSON]()
            for (k, v) in dictionary {
                object[k] = JSON(v)!
            }
            self = .JSObject(object)
        default:
            return nil
        }
    }
}
```

これで、

```
let json1 = JSON([
    "null":nil,
    "bool":false,
    "number":0,
    "string":"",
    "array":[true,"string",42.195],
    "object":[:]
] as [String:Any?])
```

という具合にずいぶんとスッキリしました。Any?で初期化できるようになったので、前述のJSONSerialization.jsonObjectを使えば文字列からも初期化できます。やってみましょう。

```
import Foundation
extension JSON {
    init?(_ s:String) {
        guard let nsd = s.data(
            using:String.Encoding.utf8
            )
            else { return nil }
        guard let any
            = try? JSONSerialization
                .jsonObject(with:nsd)
            else { return nil }
        self.init(any)
    }
}
```

実行結果は次のとおり。

```
let json2 = JSON("{\"number\":0,\"null\":null,
\"object\":{},\"array\":[true,\"string\",42.19
5],\"bool\":false,\"string\":\"\"}")
```

確かにできました。しかしこうしてできたJSON型の変数をprintすると、

```
JSObject(["number": JSON.JSNumber(0.0),
"null": JSON.JSNull, "object": JSON.
JSObject([:]), "array": JSON.JSArray([JSON.
JSBool(true), JSON.JSString("string"), JSON.
JSNumber(42.195)]), "bool": JSON.
JSBool(false), "string": JSON.JSString("")])
```

……という具合で、見づらいうえに使えません。逆変換もサポートしましょう。

```
extension JSON : CustomStringConvertible {
    var description:String {
        switch self {
        case .JSNull:
            return "null"
        case let .JSBool(b):
            return b.description
        case let .JSNumber(n):
            return n.description
        case let .JSString(s):
            return s.debugDescription
```

※次ページに続く

※前ページから続き

```
        case let .JSArray(a):
            return "["
                + a.map{ $0.description }
                   .joined(separator:",")
                + "]"

        case let .JSObject(o):
            var ds = [String]()
            for (k, v) in o {
                ds.append(
                    0k.debugDescription
                         + ":"
                         + v.description
                 )
            }
            return "{"
                + ds.joined(separator:",")
                + "}"
        }
    }
}
```

これでJSONと文字列の相互変換はバッチリです。が、まだまだ使いづらい。値も取り出せるようにしましょう(**リスト1**)。

ここまでくれば、

```
json2["array"][2].asNumber! - 0.195 == 42
// true
```

という具合にJavaScriptのJSONにさほどひけをとらない使い心地になっています。さらに==を定義してEquatableプロトコルに準拠したり、forループに直接かけられるようにしたりしていけば、立派なJSONモジュールができあがることでしょう。

次号予告

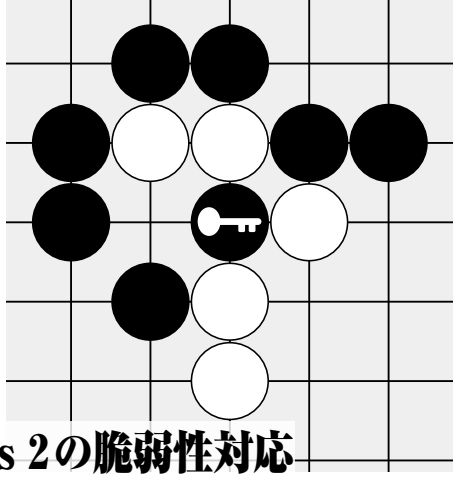
ところで前回はもう1つ謎がありました。前回作ったNestedArrayはなぜ、IntだけではなくdoubleやStringでもいけるのでしょう。次回はそれを可能にしている総称型(generics)に焦点をあてます。SD

▼リスト1　JSON型のサンプルに改良を加える

```
extension JSON {
    var isNull:Bool? {
        switch self {
            case .JSNull:       return true
            default:            return false
        }
    }
    var asBool:Bool? {
        switch self {
        case let .JSBool(b):    return b
        default:                return nil
        }
    }
    var asNumber:Double? {
        switch self {
        case let .JSNumber(n):  return n
        default:                return nil
        }
    }
    var asString:String? {
        switch self {
        case let .JSString(n):  return n
        default:                return nil
        }
    }
    var asArray:[JSON]? {
        switch self {
        case let .JSArray(a):   return a
        default:                return nil
        }
    }
    var asObject:[String:JSON]? {
        switch self {
        case let .JSObject(o):  return o
        default:                return nil
        }
    }
    subscript(i:Int)->JSON {
        switch self {
        case let .JSArray(a):
            return i < a.count ? a[i] : .JSNull
        default:
            return .JSNull
        }
    }
    subscript(s:String)->JSON {
        switch self {
        case let .JSObject(d):
            return d[s]!
        default:
            return .JSNull
        }
    }
}
```

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

【第四三回】告知が間に合わなかった!? Struts 2の脆弱性対応

今回は、3月10日に公開されたApache Struts 2の任意のコードが実行可能な脆弱性と、それに伴って発生した「都税クレジットカードお支払サイト」などの情報漏洩(ろうえい)事件について取り上げます。

Apache Struts 2

Apache Struts 2はJavaのWebアプリケーションフレームワークです。開発はApache Software FoundationのApache Struts Projectにより進められており、Apache License 2.0のライセンスで公開されているオープンソースです。Javaを使ってWebアプリケーションを作るときのフレームワークとしてかなり広く使われています。

Apache Strutsには、2000年にリリースされて2008年でリリース終了(EoL：End of Life)になったApache Struts 1と、2006年にリリースされて現在もリリースが続いているApache Struts 2があります(以下Struts 2)。

任意のコードが実行可能な脆弱性

2017年3月10日(米国時間)に、National Vulnerability Database(NVD)にCVE-2017-5638[注1]が掲載され、同時[注2]に国内のJapan Vulnerability Notes(JVN)にもJVNDB-2017-001621[注3]として掲載されました。いずれも「Apache Struts 2 には、任意のコードが実行可能な脆弱性が存在する」という内容です。

リモートから任意のコードが実行できる脆弱性を持つのは、次のバージョンです。

- Apache Struts 2.3.5～2.3.31
- Apache Struts 2.5～2.5.10

原稿執筆時に確認した脆弱性の危険性を表すCVSSは、CVSS v2の評価法で10.0、CVSS v3の評価法でも9.8であり、最悪な脆弱性だということを示しています。なお、オリジナルの脆弱性情報が公開されたのちに攻撃コードが出回ったために、CVSSはオリジナル公開時から改定されています。

この脆弱性が対応されたStruts 2.3.32と2.5.11は、3月7日にすでにリリースされていました。

実証コードの出現

3月7日に、Exploit DatabaseのサイトにCVE-2017-5638の脆弱性を使って任意のコードを実行させる実証コード(PoC：Proof of Concept)が公開されました。脆弱性対応版のリリース日と同日なので実質的にゼロデイ攻撃です。

このExploit Databaseに掲載されているPythonコードの一部を見てみると、任意のコマンドを実行する部分は**リスト1**のようになっています。

簡単に言ってしまえば、WindowsでもLinuxでも

注1) https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-5638
注2) 時差の関係で前日の9日付けになります。
注3) http://jvndb.jvn.jp/ja/contents/2017/JVNDB-2017-001621.html

シェルのコマンドラインから与えることのできるコマンドであれば、このコードを使って実行できます。以下、LinuxサーバでStruts 2が動作していることを前提に説明します。

ここで与えられたコマンドは、Struts 2が動作するときの実行権限と同じ権限で実行されます。任意のコマンドが動作しますから、外部からRAT(Remote Access Tool)をダウンロードして動かし、リモートからアクセスするなど簡単にできます。あるいはネットワーク関連のコマンドが一通り入っているならば、それを使って、外部からインタラクティブにシェルのコマンドラインを使えます。

そのため、データベースにアクセスするための情報(たとえばパスワードなど)を含んでいる設定ファイルにアクセスできてしまいます。あとはコマンドラインからデータベースにアクセスしたり、データベースをまるごとバックアップするような形でダンプできたりしてしまいます。

この場合、ルータやファイアウォールでアウトバウンド方向の通信制限(Egress Filtering)をかけているならば、RATは使えないため、侵入後の行動を制限させられる可能性もあります。しかし、現実問題としてそこまで厳しく制限をかけると、通常のメンテナンスでも支障が出るので、内部から外部に出ていく通信をあまり厳しく制限するケースは多くないと思います。

なお、Egress Filteringに興味があって、SANS Institute[注5]のEgress Filtering FAQのドキュメントを読んでいない方は、ぜひ読んでみてください。次のURLで入手できます。

- Egress Filtering FAQ
  https://www.sans.org/reading-room/whitepapers/firewalls/egress-filtering-faq-1059

都税クレジットカードお支払サイト情報漏洩事件

2017年3月10日(日本時間)に東京都主税局は「『都税クレジットカードお支払サイト』における不正アクセスについて」という報道発表資料を公表しました[注6]。

東京都は都税をクレジットカードで支払えるサイト(https://zei.tokyo/、以下zei.tokyo)を運用しています。zei.tokyoは指定代理納付者であるトヨタ

◆リスト1　CVE-2017-5638を使って任意のコードを実行させる実証コード(一部抜粋)[注4]

```
def exploit(url, cmd):
    payload = "%{(#_='multipart/form-data')."
    payload += "(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)."
    payload += "(#_memberAccess?"
    payload += "(#_memberAccess=#dm):"
    payload += "((#container=#context['com.opensymphony.xwork2.ActionContext.container']))."
    payload += "(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class))."
    payload += "(#ognlUtil.getExcludedPackageNames().clear())."
    payload += "(#ognlUtil.getExcludedClasses().clear())."
    payload += "(#context.setMemberAccess(#dm)))."
    payload += "(#cmd='%s')." % cmd
    payload += "(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win')))."
    payload += "(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd}))."
    payload += "(#p=new java.lang.ProcessBuilder(#cmds))."
    payload += "(#p.redirectErrorStream(true)).(#process=#p.start())."
    payload += "(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream()))."
    payload += "(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros))."
    payload += "(#ros.flush())}"
```

注4)　出典：https://www.exploit-db.com/exploits/41570/　作者：Vex Woo

注5)　政府、企業、団体間における研究、およびそれらに従事する人々のITセキュリティ教育を目的として、1989年に設立された組織。本部は米国ワシントンDCにあります。

注6)　http://www.metro.tokyo.jp/tosei/hodohappyo/press/2017/03/13/02.html

ファイナンス㈱を経由し、再委託業者のGMOペイメントゲートウェイ㈱[注7]が運営していました。

3月9日に、独立行政法人情報処理推進機構（IPA）からソフトウェアの脆弱性に関する注意喚起が出たので、指定代理納付者において影響調査を開始し、3月10日に指定代理納付者より不正アクセスおよび情報流出の可能性について都に報告があったとのことです。

ここから先は、zei.tokyoを管理していたGMOペイメントゲートウェイが公表した「不正アクセスに関するご報告と情報流出のお詫び」[注8]を参考に状況を確認してみます。

この報告には、Struts 2の脆弱性を使われたことが明示されています。「都税クレジットカードお支払サイト」だけではなく、同じく運用を請け負っている独立行政法人住宅金融支援機構の「団信特約料クレジットカード払い」のサイトも同じく侵入されたと説明しています。流出した可能性のある情報は**表1**のとおりです。

外部からRATを経由してデータベースにアクセスした可能性としてこの数字を示しているようです。状況としては、「団信特約料クレジットカード払い」サイトのほうが「都税クレジットカードお支払サイト」よりも件数は少ないですが、第三者がクレジットカードをオンラインで使うのに十分な情報を流出させているという点でより深刻です。さらに言えばセキュリティコードがサイト側に保存されている点で、Struts 2の脆弱性以前の問題としてデータベース設計の段階でたいへん問題があると筆者は指摘したいと思います。

一方、「都税クレジットカードお支払サイト」のほうは、クレジットカード情報とメールアドレスだけという妙に中途半端なデータベースのアクセスのしかたであることは気になります。

報告書にある調査の経過を見るかぎりでは、GMOペイメントゲートウェイ側で行った脆弱性の調査は、国内で脆弱性の告知が出るのと同日の3月9日に行っています。攻撃コードが出回ったのは、脆弱性対応バージョンが出たのと同日の3月7日。現状、国内では、JVNの告知が出たタイミングでその情報をもとに脆弱性に対処するというのがスタンダードな流れです。GMOペイメントゲートウェイの調査は平均的なものだったと言えるでしょう。

国内で被害があいつぐ

3月10日付けでJETRO（日本貿易振興機構）も同様の被害があったことを公表しました[注9]。こちらは3月8日に侵入された形跡があったと述べられています。PoCが公表されて1、2日といった極めて短い時間で攻撃が発生したということになります。

日経BP社のITproに掲載された「猛威振るうStruts2脆弱性への攻撃、どうすれば防げたか」の記事[注10]には、東京都主税局、住宅金融支援機構、

◆**表1　流出した可能性のある情報**

サイト名	情報	件数
都税クレジットカードお支払サイト	① クレジットカード番号、クレジットカード有効期限	61,661件
	② ①に加え、メールアドレス	614,629件
団信特約料クレジッカード払い	① クレジットカード番号、クレジットカード有効期限、セキュリティコード、カード払い申込日、住所、氏名、電話番号、生年月日	622件
	② ①に加え、メールアドレス、加入月	27,661件
	③ ①に加え、メールアドレス	5,569件
	④ ①に加え、加入月	9,688件

注7）念のためzei.tokyoのSSL証明書を確認すると、取得している組織もGMO Payment Gateway, Inc.でした。
注8）https://corp.gmo-pg.com/newsroom/pdf/170310_gmo_pg_ir_kaiji.pdf
注9）https://www.jetro.go.jp/news/announcement/2017/694a96ddf47971f2.html
注10）http://itpro.nikkeibp.co.jp/atcl/column/14/346926/032100893/

JETROに続いて、工業所有権情報・研修館、日本郵便、沖縄電力、ニッポン放送などのサイトがStruts 2の脆弱性により侵入されたことが紹介されています。

CVE-2017-5638への対応はできたか

Apache Struts 1および2には、これまでに何度も、外部から任意のコードを実行できる危険度の高い脆弱性が発見されています。前回の本連載で取り上げたWordPressもそうでしたが、脆弱性が対応されたバージョンがリリースされたのち、バージョンアップが広がったタイミングで、そのリリースの本来の目的、すなわち脆弱性を告知するケースが多くなってきています。

ある程度の技術力があれば、リリースされたソフトウェアの差分をチェックすれば、何のためにアップデートしたのかがわかります。つまり、攻撃者側は脆弱性情報告知など必要ない、というわけです。一方で、利用者の多くはしくみをわかって使っているわけではないですし、コードを読むわけでもありません。このような状況では、脆弱性だという認識が遅れる利用者側が、攻撃者よりも時間的に遅れてスタートすることになります。

先ほどのITproの記事には、3月7日午後7時過ぎに、金融ISAC[注11]の会員の間では脆弱性の問題共有が始まっていて、翌日の朝9時には全体に共有できていたということが書かれていました。PoCが公開されたという情報を見つけての行動かと思います。金融系企業でも多くのStruts 2を使っていると思いますが、被害が見られないのはこれらの迅速な情報共有のためだったようです。

FLOSS(Free/Libre and Open Source Software)のソフトウェアをうまく使うためには、横のつながり(=コミュニティ)が重要な役割を果たすと言います。今回は金融ISACがApache Strutsのコミュニティとしてうまく機能していたのではないかと思います。一方で、金融ISACから外れてしまっている国内ユーザは、コミュニティが弱いためか、うまく情報が伝わっていなかったのかもしれません。

Webのトラフィックでインバウンド方向(外部から内部への方向)からのトラフィックの内容をすべてスキャンし、コマンド、プログラム、スクリプトとみなせるものは全部ブロックするようなしくみがあるといいのかもしれません(すでにあるのかもしれませんが、筆者の知る範囲では見当たりません)。

ACL(Access Control List)の機能を使えば、wgetといったダウンロードに使われるコマンドは、Apacheの動いている権限からは使えないようにすることが可能です。この方法ではまだ完全ではないにしても、少しはこの手の攻撃を抑止する可能性は高まるはずです。

あと最後に考えられるのは、PoCが公開された時点で、JPCERT/CCやIPAが緊急告知するという方法です。これならもう少し早く対応できたのではないかとも考えられます。しかし、現実にはPoCが公開されても、それをJPCERT/CCやIPAが実証実験をして確認できて、さらにその影響を評価してからになるので、3月7日(日本時間では6日)にPoCが公開されても、情報公開日はやはり3月9日(のビジネスアワー)になります。実際の3月10日に公開した結果と、ほぼ同じ結果となるでしょう。

最後に

WordPress 4.7および4.7.1のときもそうでしたが、脆弱性公表アナウンスと攻撃が始まる時間の間隔が非常に短くなっています。脆弱性に対応したバージョンをリリースしたのちに、この脆弱性を使った攻撃コード(PoC)が作られているので、攻撃者が先に脆弱性を知っているゼロデイ攻撃ではありません。この連載が始まったころは、こんなことが次々に起こるという状況はありませんでした。脆弱性対応と、脆弱性を使った攻撃の関係は新しい段階に入り、我々も新しい定石を考えなければいけないのかもしれません。SD

注11) 国内の金融機関が加盟し、セキュリティに関する情報の共有および分析を行うための組織。

46

Debian Developer　やまねひでき　henrich@debian.org

SHA-1コリジョン問題の余波

Debian Hot Topics

Debian 9“Stretch”の進捗ですが、バグが閉じられては、新しいバグが登録され……という感じで数字の上ではほとんど変わりがありません。辛い期間が続きます。

debian-installerはGUIが標準に

Debianのインストーラ「debian-installer」がデフォルトで、GUI(Graphical User Interface)で動作するようになりました(これまでは明示的に選択が必要でした)。

とはいえ、GUIになったからといって劇的に変わるようなことは何もなく、CUI(Character User Interface)の内容がほぼそのままGUIになっているだけです(**図1**)。これは、もともとGUIが用意されたのが「CUIでは表示できない結合文字の言語に対応するため」という理由だからです。

「インストーラがGUIじゃないから」という理由だけで拒否するような食わず嫌いの人が減るのを期待する一方、もうちょっと使いやすくしてはもらえないものか……という思いも否定できません。ですが、そのあたりはDebian 10“Buster”のころに期待しましょう。

▼図1　GUIになったdebian-installer

SHA-1で署名したパッケージはリジェクト

SHA-1の危険性が明らかに

セキュリティの分野では、データが偽造されていないこと(同一性)の確認に使われるハッシュ関数というものがあります。このハッシュ関数は、同じ値を入力するとまったく同じ値を出力しますが、ほんの1文字でも違うデータを入力するとまったく違う値を吐き出すという特性を持ちます。

アルゴリズムとしてはMD5やSHA-1、SHA-2[注1]などがありますが、MD5やSHA-1はすでに前述の特性にもかかわらず、異なったデータを入力しても同一の値(コリジョン)を吐き出させることが可能な手

注1) SHA-2として分類される中に、さらにSHA-224、SHA-256、SHA-512などのバリエーションがあります。

法(攻撃)が発見、あるいは示唆されています。

ハッシュ関数を使ったもっともメジャーなものとしては、WebサイトのSSL/TLS証明書があります。過去にはMD5がSHA-1に置き換えられ、現在はSHA-1からSHA-2への移行が推奨されており、SHA-1は破られる危険性が高まっているとして、各Webブラウザで危険だと警告を出す、あるいはページの表示を拒否するようになっています。

とはいえ、MD5について攻撃方法が指摘されたのは1996年ですが[注2]、長い間、誰も対策を取りませんでした。そして、研究者らが実際に約1時間で攻撃に成功したのを発表したのが2004年、ようやくChromeがMD5を使ったSSL証明書を拒否するようになったのは2011年です。「危険性を指摘するだけではダメで、本当に攻撃可能なことを公表しないと誰も動かない」のが実情なのですね。

SHA-1についてもMD5よりはマシなものの同じ状況で、実際にCA Security Councilという業界団体は、2014年にGoogleに対してSHA-2移行を急がせないよう、抗議をしていました[注3]。

そして先日、実際にSHA-1で違うデータから同じ値を意図的に作る手法が可能になったことを、Googleとオランダの研究機関CWI Instituteが発表しました[注4]。GoogleにはMD5での教訓があったのは間違いなさそうです。これによって、SHA-1のTLS証明書での利用停止は「待ったなし」となることが確実でしょう。

Debianプロジェクトも危険性を認識

Debianプロジェクトも、SHA-1アルゴリズムでの署名に対する危険性を認識し、SHA-1またはRIPE-MD/160アルゴリズムを使ってGPG署名されたパッケージのアップロードを、サーバ側で拒否するようになりました[注5]。これで偽のSHA-1署名を使った危険なパッケージがアップロードされてしまう危険はなくなりました。

今年のアップロード44,189個中137個(0.3%)がSHA-1を使ったものだったそうで、作業者には影響はないようです。RIPE-MD/160が使われていた形跡はないようですが、同様に危険性が高いことを理由に拒否対象とされています。

ちなみに、DebianではほかにもパッケージリポジトリのReleaseファイル[注6]でハッシュ関数が使われていますが、こちらはすでにMD5、SHA-1、SHA-256をそれぞれ使って記述されるようになっています。まだSHA-256に対する有効な攻撃手法は見つかっていませんし、たとえ見つかったとしても、利用している3つのハッシュ関数を同時に攻撃するような攻撃方法は難しいと思われます。

ほかのOSSの状況

ほかのOSS関連では、Gitのコミットが SHA-1ハッシュを使っているということで、元Debian開発者のJoey Hessさんから質問が上がりました。

もともとの開発者であるLinus Torvalsさんや現在のメンテナの濱野純さんらと長い長い議論が行われましたが[注7]、Linusさんいわく「そもそも暗号署名とコンテンツを識別するためのハッシュ値の生成では意味合いが大きく違う。すでに問題の緩和策のパッチも書かれているし、ほかのハッシュアルゴリズムへ移行するけれども既存リポジトリは破壊されない。そして移行は今すぐには行われない」[注8]とのことで、現状はほかのアルゴリズムも使えるようにと改修が始

注2) "How to Break MD5 and Other Hash Functions" URL http://deb.li/keyj
注3) URL http://deb.li/iOtf2
注4) URL https://shattered.io/
注5) URL http://deb.li/3sbti
注6) URL http://ftp.jp.debian.org/debian/dists/stretch/Release
注7) URL http://deb.li/37pDR
注8) URL https://plus.google.com/+LinusTorvalds/posts/7tp2gYWQugL

まっているようです。

他方、同じVCS(バージョン管理システム)でも、Subversionでは、WebKitの開発者らが試しにコリジョンを起こしたPDFをリポジトリに突っ込んでみたところ異常状態になるなどしたようです[注9]。そのような問題が見つかってはいるものの、修正はまだリリースされていません。

LTSスポンサーの増加など

スポンサーの資金協力のもとで実施されているDebian LTS[注10]ですが、シルバースポンサーが1社、ブロンズスポンサーが2社増加して割り当て可能なサポート作業時間が増えているようです[注11]。

DebConf17のスポンサー

また、この夏にカナダ・モントリオールで開かれるDebConf17のスポンサーも順調に集まっているようです[注12]。地元モントリオール企業のSavoir-faire Linux社をはじめ、Steam OSを開発しているValve社や、各種OSSの開発/コンサルティングをワールドワイドで展開しているCollabora社などがすでに名乗りをあげています。SD

注9) チェックアウトができなくなったり、リポジトリの監視botが皆エラーを吐く状態になったりしたそうです。しかし、実運用中のリポジトリにいきなり突っ込むのは「軽率じゃないの?」という気がしますね……。
URL https://bugs.webkit.org/show_bug.cgi?id=168774

注10) LTSはLong Term Supportの略。Debianの場合は旧安定版(oldstable)のサポートが終了してから2年間、つまり安定版のサポート3年と合わせると、合計で5年間のサポートが行われます。 URL http://deb.li/eo4D

注11) 個人的には、プラチナスポンサーの㈱東芝は原子力発電事業部門の問題があるので、「継続していただけるのだろうか?」と懸念しているところです……。杞憂だと良いのですが。

注12) URL http://deb.li/3atzz

Ubuntu 17.04とそのフレーバーの変更点

Ubuntu Japanese Team　あわしろいくや

今回は、4月13日にリリースされたUbuntu 17.04とそのフレーバーの変更点をお知らせします。

地味な変更にとどまる

17.04では、Unity 8に移行など派手なものもなく、また目立った新機能もないため、地味な変更にとどまったと言えます。

そんな中で、意欲的な試みとして（LVMを除くという制限はあるものの）スワップパーティションがスワップファイルに変更されました。これまでは、本体内蔵メモリ32GB、SSD 250GBのPCにUbuntuをインストールした場合、SSD上に32GBのスワップパーティションが作成されていました。しかし、これは明らかに過剰です。全メモリをスワップすることはほとんどなく、ただでさえ少ないSSD容量を浪費していました。

17.04からはHDD/SSDなどハードドライブの空き容量の5%、あるいは2GBのどちらか少ないほうのスワップファイルが自動的に作られるようになります。先の例ですと、250GBのSSDに2GBのスワップファイルが作られるため、残りの30GB分多くルートパーティションに割り当てることができるようになります。スワップファイルは固定サイズですが、サイズ変更もそれほど難しいわけではないので、2GB以上のスワップ領域が必要な場合でも対処できます。

ほかにも意欲的な試みとして、ローカルDNSリゾルバがdnsmasqからsystemd-resolvedに変更になりました。ユーザにはとくに違いはわからないはずですが、このためにNetwork Managerを16.10の1.2.6から1.4.4にアップグレードするなど、相当の準備期間を経てようやくここで変更されました。

ネットワーク上、あるいはUSB接続で対応したプリンタが存在する場合、必要な設定を自動的に行うことができるようにもなりました。

忘れてはいけないのが、新しいフレーバーが追加されたことです。では、詳しく見ていきましょう。

17.04概要

リリース日とコードネーム

Ubuntu 17.04は4月13日にリリースされました。本誌の発売直前のため、本記事の執筆はベータ版の段階で行っています。コードネームは“Zesty Zapus”で、「ピリっとしたトビハツカネズミ」という意味です。頭文字がアルファベット順もついにZに到達し、次はどうなるのかが注目されます。

共通の変更点

GNOME関連パッケージは3.20と3.22の混合で、後者に統一されることはありませんでした。具体的にはファイル（Nautilus）やGNOME端末は3.20のままです。Ubuntu GNOMEとUbuntu Budgieに関連するパッケージは、GNOME 3.24にアップデートされています。

カーネルのバージョンは4.10で、X.Orgは1.19.3、グラフィックスライブラリであるMesaは17.0.2にな

る見込みです。とくにMesaは16.10では12.0.6だったため、大幅なバージョンアップとなりました。新しいハードウェアで3D関連のAPI（OpenGL、OpenGL ES、OpenCL、OpenMAX、VDPAU、VA API、XvMC、Vulkan）を使用したい場合にはうれしい変更です。なおX.Orgは不具合が見つかっているため、1.18.4でリリースされる可能性はあります。

冒頭にも書いたとおり、新規インストールの場合はスワップパーティションではなくスワップファイルが作成されるようになりました。スワップファイルはfallocateコマンドで作成でき、ルートパーティションの直下に置かれます。具体的には“/swapfile”です。これを/etc/fstabでマウントしています。よってスワップファイルのサイズを変更したい場合はfallocateコマンドでファイルを作成し、スワップをオフにしてから“/swapfile”を置き換え、ふたたびスワップをオンにするだけという手軽さです。

ローカルDNSリゾルバは、以前からパッケージ自体（パッケージ名はdnsmasq-base）はインストールされていたものの、有効になっていませんでした。前述のとおりユーザからは違いがわかりにくいですが、自動生成される/etc/resolv.confは別物ですので、覗（のぞ）いてみるといいかもしれません。少なくとも筆者には驚きがありました。

プリンタの自動設定はCUPSの最新版で実装されています。IPP Everywhere、Apple AirPrint、あるいはIPP-over-USB対応プリンタであれば恩恵に預かれる可能性があります。設定は不要で、対応プリンタを認識すれば自動追加されます（**図1**）。仕様としては、cups-browsedデーモンを終了すると設定は削除され、起動すると自動的に追加します。筆者が所有しているプリンタでは**図1**のとおりうまく動作しましたが、再起動を繰り返すとまれにドライバを正しく検出せずにエラーが印刷されることがありました。

図1　筆者のプリンタで自動設定した様子

LibreOfficeのバージョンは、5.3になっています。詳細は本連載第83回（2017年3月号）をご覧ください。Mozcのバージョンは16.10の2.17.2116.102から2.19.2623.102にアップデートしています。各種設定ツールがQt4から5に移行しています。

Ubuntuの変更点

Unityのバージョンは7.15で、16.10のマイナーアップデートです。Unity 8セッションがオプションでインストールされるのも同じです。

Ubuntu（デスクトップ）だけで、サーバや各種フレーバーには影響されないのですが、ついにnet-toolsパッケージがデフォルトではインストールされなくなり、**表1**のコマンドが使用できなくなりました。もちろん追加インストールすることはできるのですが、これを機にipコマンドなど、より新しいものを習得しましょう。

本稿の執筆時点では試せていないのですが、17.04のリポジトリにあるVirtualBoxのGuest Additionsをインストールすると Unity 8が動作するようになるとのことです。今までもKVMでUnity 8を試すことができていますが、VirtualBoxだとより手軽に試せるようになるので便利です。

Kubuntuの変更点

現在KDEは、KDE Software Compilation（SC）として3つに分割してリリースを行っています。Plasmaが5.9.4、Frameworksが5.31.0、Applicationsが16.12.3というバージョンです。おおむね最新と言っ

表1　net-toolsパッケージがインストールされなくなったことにより使えなくなったコマンド

slattach	plipconfig	ipmaddr
rarp	mii-tool	route
iptunnel	ifconfig	nameif
netstat	arp	

ていいですし、またリリースまでにアップデートされる可能性もあります。16.10と比較するとかなり新しくなっています。

本誌2016年12月号の本連載第80回でKubuntu 16.10についても取り上げ、その中でドキュメントビューアであるOkularの動向について紹介しました。17.04ではデフォルトでインストールされるように戻ったばかりか、KDE Frameworks 5へのポーティングが完了しています。Okularは高性能なドキュメントビューアですので、Kubuntuユーザにはうれしいニュースです。

Xubuntuの変更点

XubuntuはSGT Puzzles Collectionというゲームが追加されました(**図2**)。その名のとおり多数のパズルゲームを遊ぶことができます。

17.04でもやはり壁紙が変更になっていますが、その程度の違いしかありません。ファイルマネージャであるThunarは1.6.11というバグ修正バージョンになっていますが、すでに16.04 LTSでも16.10でも提供されています。

Lubuntuの変更点

LubuntuはLXQtに移行するという話もありましたが、現在ではずいぶんとトーンダウンしています。LXQt版のイメージを配布するという話もありましたが、やはり17.04でも見送られています。

Lubuntuとしての変更点は、lxhotkeyパッケージが追加されたことです(**図3**)。これでさまざまなホットキーの確認や変更を行うことができます。

リポジトリのLXQtパッケージは0.11.0にアップデートしており、これをインストールすることもできますが、現段階でLXQtを使用したい場合はほかのLinuxディストリビューションを選択するほうがいいでしょう。

Ubuntu GNOMEの変更点

GNOME関連パッケージは、Ubuntuその他と共通のパッケージは3.20あるいは3.22になっていますが、Ubuntu GNOME(や後述するUbuntu Budgie向け)は3.24になっています。すなわちGNOME Shellやそれに必要なパッケージは3.24になっています。

パッケージの構成も大きく変更になっています。ついにメーラーやスケジューラーなどの機能を備えたEvolutionがインストールされなくなりました。また、意図は不明ですがWaylandセッションもデフォルトインストールから外れています。

Flatpakに対応するためのパッケージも追加されました。Flatpakは明瞭な説明が難しいのですが、

図2　SGT Puzzles Collectionには多数のゲームが含まれている

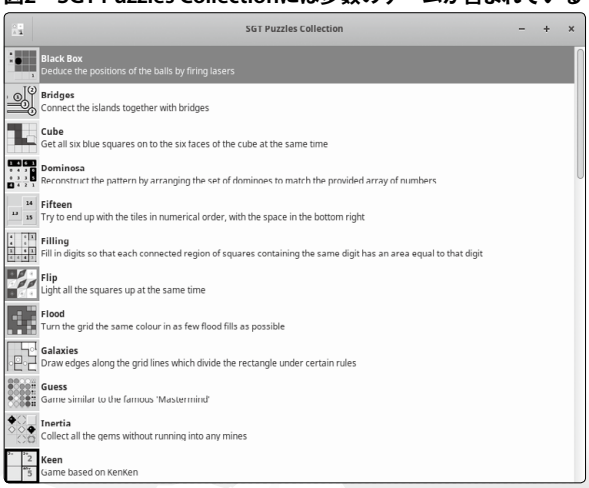
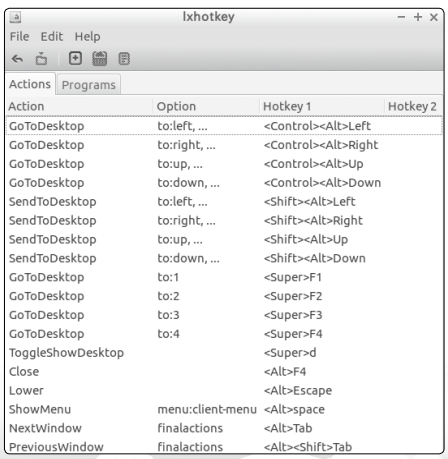


図3　lxhotkeyでホットキーの変更ができるものの、作りはやや大味だ



snapのようなもので、デスクトップに特化した新しいパッケージの配布と管理のしくみです。

Ubuntu MATEの変更点

Ubuntu MATEのMATEデスクトップ環境は1.18にアップデートしています。というよりも、MATE 1.18は17.04にスケジュールを合わせてリリースしたように見えます。もともとUbuntuのリリースが4月と10月になったのはGNOMEが3月と9月のリリースで、これに合わせたものだったのですが、Unityに移行してからは必ずしも最新のGNOMEにこだわらなくなりました。一方、GNOME 2.xからフォークしたMATEは主要開発者がCanonicalに就職し、3月と9月にリリースされるようになりました。MATEはリリース時期を明言しているわけではないので、今のところ1.16と1.18のみであり、今後も同じかどうかはわからないとは付記しておきます。

MATEはその性格上、新バージョンでたくさんの機能を追加するようなことはしません。よって、1.18の大きな変更点はGTK+ 3の完全サポートとGTK+ 2関連コードの削除です。あとは新しいライブラリへの対応や、個別アプリケーションの機能追加などがあります。増減したパッケージも多いのですが、特記すべきは設定ツールであるdconf-editorが削除されたこと、電卓アプリがMATE Calculatorになったことくらいでしょう。

MATEが気に入っている方は、まずはUbuntu MATEをまっ先に検討すべきと言える状況になってきました。

Ubuntu Budgie

Ubuntu Budgie[注1]は17.04から追加された新しいフレーバーです。どこから派生したわけでもなく、独自に開発されたSolus[注2]というLinuxディストリビューションがあるのですが、BudgieはそのSolusのために開発されたデスクトップシェルです。現在はそのSolus以外でも採用されており、そのうちの1つがこのUbuntu Budgieというわけです。

なぜBudgieはデスクトップ環境ではなくデスクトップシェルなのかというと、GNOMEスタック、より具体的にはGNOMEコントロールセンターとGNOME設定デーモンを必須としているからで、GNOME Shellの代わりにBudgieを採用していると考えればわかりやすいです。ただし、今後は変更の予定とのことです[注3]。

本題に戻りましょう。インストール後、ログインすると大きく3つのパートに分かれています。左上の●はBudgieメニューで、Xubuntuと同じような伝統的なメニューです。左中央にあるのがPlankというドックです。とくにBudgie用というわけではなく汎用に開発されているもので、Ubuntu MATEでも設定によっては使用されます。右上をクリックすると表示されるのがRavenというメニュー（**図4**）で、Budgieの大きな特徴になっています。

WebブラウザがChromium、メディアプレーヤがGNOME MPV、端末がTerminix（現Tilix）を採用しているなど、ほかのUbuntuフレーバーとの違いもけっこうあります。

インプットメソッドはデフォルトでIBusになりますが、そのままの設定だと使いづらいので**図5**のコマンドを実行してください。

注1） https://ubuntubudgie.org/
注2） https://solus-project.com/
注3） https://budgie-desktop.org/2017/01/25/kicking-off-budgie-11

図4　右側に表示されているのがRavenメニュー

これでIBusのステータスをインジケーターで表示するようになり、しかもそれが白で見やすくなり、Mozcへの切り替えをCtrl+スペースキーで行えるようになります。

もちろんFcitxも使用できます。[言語サポート]の[キーボード入力に使うIMシステム]を[fcitx]にしてログアウトし、再ログインします。コマンドでは「`im-config -n fcitx`」を実行し、やはりログアウトして再ログインしてください。

Ubuntu Kylin

Ubuntu Kylinは中国向けのフレーバーですので、日本語で使用する場合は積極的にインストールする選択肢には入らないのですが、17.04では興味深い動きがあったので取り上げることにしました。これまでデスクトップシェルはUbuntuと同じくUnity 7でしたが、17.04からはUKUIという専用のデスクトップ環境を開発し、採用することにしたようです。とはいえ一から開発したわけではなく、大きくはMATEデスクトップ環境の必要なコンポーネントのみをforkし、独自開発しています。これによりUnity 7よりは必要なハードウェアスペックが引き下げられるのと、Unity 7の今後に振り回されることがなくなったのがメリットといえます。逆にデメリットは自前で開発するリソースを確保しなければいけないことです。なぜMATEの開発に協力するわけではなく、forkという道を歩むことにしたのかは興味深いところです。いずれにせよそれだけのリソースがあるのはうらやましい限りです。

翻訳の問題

17.04のリリースサイクルで、とある方がWeb翻訳サービスの結果をLaunchpadやそのほか多数のアプリケーションの翻訳として貼り付けていたことが発覚しました。Web翻訳サービスの利用許諾はさまざまですが、少なくともLaunchpadの翻訳サービスは3条項BSDライセンスである必要があります。両者に互換性がある場合は問題ないのですが、多くの場合ありません。さらに問題なのは具体的にどのWeb翻訳サービスを使用したのか明言せず、かつそれが複数だったため、もはやどうなっているのか誰にも把握できない状態であったということが明らかになりました。

影響範囲は広範囲ですが、その方の協力が得られないため、いまだに全貌が明らかになっていません。Ubuntuも例外ではなく、これを受け主としてUbuntu Japanese TeamとUbuntu Japanese Translatorsのメンバにより IRCミーティングが行われました。

Launchpadで翻訳できるパッケージを精査した結果、さほどの分量がないことがわかりました。そこで、翻訳の削除者と再翻訳者を別の人にするというクリーンルーム方式を取ることになり、削除作業は完了しました。あとは17.04の締め切りに合わせて再翻訳が行われました。

Ubuntuとしてはそれでいいのですが、影響を受けるフレーバーやデスクトップ環境もあります。具体的にはUbuntu MATEとUbuntu Budgieで、これらはアップストリームでの対応が必要になります。Ubuntu MATEでは、すでに完了していることを確認しています。またLinux Mintでも甚大な影響があり、翻訳の削除作業が行われています。いずれも1ヵ月や2ヵ月で終わる作業ではありませんが、少なくともUbuntu Budgieの翻訳は17.04リリース時点のものと、今後ではまったく違うものになることが予想されます。救いはより高い精度の翻訳になることが見込まれることでしょうか。

この件に関しては来月以降に、詳細に紹介しようと考えています。SD

図5　IBusの設定を修正する

```
$ gsettings set org.freedesktop.ibus.panel show-icon-on-systray true
$ gsettings set org.freedesktop.ibus.panel xkb-icon-rgba '#ffffff'
$ gsettings set org.freedesktop.ibus.general.hotkey triggers "['<Control>space']"
```

Unix コマンドライン探検隊
Shellを知ればOSを理解できる

Author 中島 雅弘(なかじま まさひろ) ㈱アーヴァイン・システムズ

システムを操作するうえで不可欠な対話によるコマンドライン操作を紹介します。bashでのスクリーンエディタライクな編集と伝統的なhistory操作の2つを修得しましょう。

第13回 コマンドライン操作今昔

今日のコマンドライン編集

sttyふたたび ―Set the options for a terminal device interface

連載第7回(2016年11月号)に少しだけ登場したsttyは、端末のオプションを設定したり、現在の状態を確認したりするコマンドです。stty -aで、シグナルの送信、コマンドラインの編集、端末の操作など、キーボードに割り当てられたコマンドが表示されます。軽視されがちですが重要ですので、復習しておきましょう。

Ubuntuでのstty -aの例
```
$ stty -a↵
intr = ^C; quit = ^\; erase = ^?; kill = ↵
^U; eof = ^D; eol = M-^?; eol2 = M-^?; ↵
swtch = <undef>;
start = ^Q; stop = ^S; susp = ^Z; rprnt = ↵
^R; werase = ^W; lnext = ^V; discard = ^O;
...略...
```

シェルの出力に現れる「^X」という表記は、一部の例外を除き、Ctrl+X(Controlキーを押しながらXキーを押す)という操作を表しています。

ここでは、表示された情報の中で、コマンドライン編集だけに注目します。^?はDELキー[注1]と同じ意味です。カーソルのあるポジションを1文字削除します。^U:killは、カーソルの位置から行頭までを削除します。^W:weraseは、カーソルのある位置の前の単語を削除します。

コマンドライン操作の基本は、このsttyで確認できる機能です。行単位での入力しかできないbourne shellでは、間違えて入力したコマンドを編集する手立てが、Ctrl+H/BS[注2][前の文字を1文字消す]、Ctrl+W[前の単語を消す]、Ctrl+U[入力行を取り消す]、と限られていました。これら基本操作だけでは、間違いに気づいて長いコマンドラインを入力しなおさなければならないときはたいへんですね。

STEP UP! ttyコマンド―print the file name of the terminal connected to standard input― ユーザのターミナル名を表示する

sttyのsを取った、ttyというコマンドがあります。マルチユーザ・マルチタスクのUnixでは、複数のユーザがシステムへ接続して作業できます。ttyは、現在の標準入出力先に結び付いている端末[注3]を表示するコマンドです。

これまでの連載を読まれている方の中には、ちょっとおかしな点に気づいた方もいるかもしれません。sttyもttyも、説明の英文のどこにもttyに対応する文字がないのです。ターミナル(端末)は、古くはTeleTYpe(テレタイプ)を

注1) MacでDELはFn+deleteキー。

注2) BS=バックスペース(Backspace)。Macではdeleteキー。

注3) システム的には、入出力装置と対応したデバイスドライバの名前。

接続していました。ttyという名称は、この当時から使っていた名称の名残です。テレタイプとは、印刷電信機、つまり電気的なタイプライターで、電話線でつながった先に打った文字を印字する装置のことです。第3回(2016年7月号)でのシグナルの解説で、本来は回線の切断時に送られるシグナル(SIGHUP)があることを紹介しました。ttyという名称も、そうした名残です。ということで、sttyも"Set the options for a TeleTYpe device interface"となりますね。

```
Ubuntuでの例
$ tty
/dev/pts/21
macOSでの例
$ tty
/dev/ttys000
```

macOSは、直球のttyという名称が割り当てられていますが、Ubuntuではptsという名のデバイスドライバ[注4]が割り当てられています。ptsのptはpseude[注5] tty＝擬似端末のことです。物理的な端末装置ではなくて、ソフトウェアによる端末が接続されていると、このドライバが割り当てられます。System V系Unixでの命名で、ptmxのスレーブがptsになります。BSD系のシステムでは、ptya、ptybとかが使われていました。

近年、物理端末をつないでいるケースはあまりありませんが、UbuntuなどLinuxの標準GUI環境では、Ctrl＋Alt＋(F1～F6)で、tty1～tty6の(擬似？)物理端末に接続できます。ttyからは、Alt＋F7でGUIへ復帰します。GUIの動作が不安定なときなど、重宝する技ですので覚えておきましょう。

注4) デバイスドライバは、/devの下に配置されています。
注5) スードーと発音。

▼表1 bashのemacsモードでの移動操作

入力	操作内容
Ctrl＋A	行の先頭に移動
Ctrl＋E	行の終わりに移動
Ctrl＋B／←	カーソルを1文字手前に戻す
Ctrl＋F／→	カーソルを1文字先に進める
Esc B	1つ前の単語の先頭
Esc F	1つ後ろの単語の先頭
Ctrl＋]	続けて入力した文字がコマンドライン中に現れるところまでカーソルを移動

※「Esc B」という表記は、ESCキーを押して離したあと、Bキーを押す操作を表しています。

bashのemacsモード

bashでは、bourne shellやcshと異なり、スクリーンエディタのようにカーソルを移動してコマンドライン編集ができます。viモードとemacsモードがあり、デフォルトはemacsモードです。編集コマンドの入力が少なくて済むのが売りのviですが、bashのviモードは、コマンドライン編集ではかえって入力が煩雑になってしまう傾向があり、あまり人気がありません。ここではemacsモードのみ取り上げます。

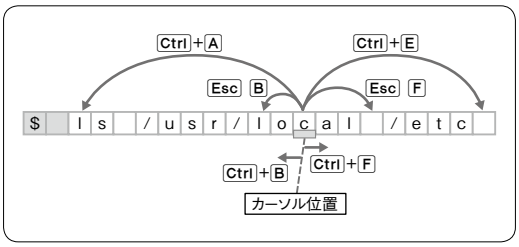
```
コマンドライン編集方式のモードを指定する(片方をセットすると片方はオフになる)
$ set -o emacs   emacsモードにする
$ set -o vi      viモードにする
```

まずは、基本のカーソル(入力する位置)移動を修得しましょう。基本のカーソル移動は**表1**の操作です。Ctrl＋]は、長い行を入力していて、行中の特定の位置に移動したいとき[注6]便利に使えます(**図1**)。

bashは、改行を入力するごとに、行をコマンドとして実行を試みます。そして、入力された行は履歴(history)として記憶されます。この履歴の中を移動するには、Ctrl＋P／↑[1つ前の履歴]、Ctrl＋N／↓[1つ後ろの履歴]、

注6) viでのfコマンド。

▼図1 emacsモードでのカーソル移動例1



Esc <[履歴の先頭]、Esc >[履歴の終わり]、を使います(**図2**)。

みなさんも実際に、コマンドラインにコントロールコマンドを入力してみてください。

emacsの操作コマンドの多くはB＝Backward、E＝End、P＝Previous、N＝Next、F＝Forwardなど単語の頭文字か、そのアルファベットが連想させるものが割り当てられています。とくにカーソルの移動は、viとずいぶん異なりますね。一方いずれにおいても、矢印(アロー)キーは、カーソル移動の中心的な役割ではありません。アローキーやPage-Up/Downキーなどは、手をホームポジションから大きく移動しなければなりませんし、そもそもこれらのキーがないキーボードもあります。コントロール、エスケープ、アルファベット、数字、記号で操作を完結できるように慣れておくと、異なる環境で操作するときも、作業効率が低下しないで済みます。いくつかのコマンド(とくにゲーム)には、US配列のキーボードを想定したものもあります。US配列での作業は効率が高く、筆者は好んで使っています。システム管理業務では、いろいろなシステムを操作する機会がありますので、日本語配列とUS配列の両方を使えるようになりましょう。

またエディタについても、viとemacsの基本操作は修得しておくべきです。初心者がコマンドラインで作業していて、ひょんなことからエディタが起動して、終了のしかたがわからないというのを見かけます。最低viの終了のEsc :q!と、emacsの終了のCtrl+X Ctrl+Cの2つは、必須の呪文です。infoコマンドでドキュメントを読むときもemacsの操作です。

さらに上級者は、インストールされていない、インストールしてはいけない、端末が遅いなどの理由で、スクリーンエディタが使えない過酷な環境では、ラインエディタ(exやed)を使います。テキスト処理でも応用が利きますので、修得しておいて損はありません。

次は、行を編集しましょう。よく使う操作は**表2**のとおりです。入力したキーとその動作のイメージは**図3**のようになります。

表2の操作で†が記されている処理は、削除時にkillバッファに内容が保存されます。これを貼り付けるには、Ctrl+Y[切り取りバッファ(†の付いた削除アクション)の内容を貼り付ける]、を使います(**図4**)。

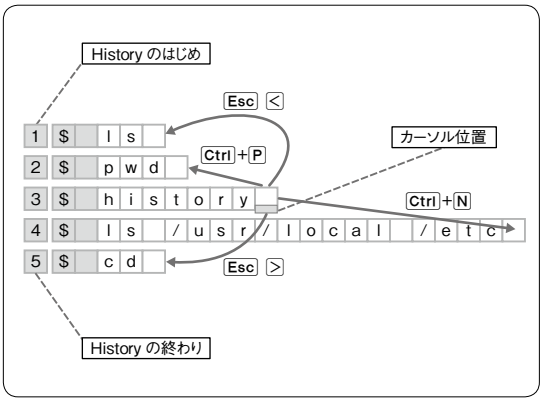
文字や単語を前後入れ替える操作として、Ctrl+T[カーソル位置とその前の1文字を入れ替え、入れ替え後カーソルは1つ前方に進む]、Esc T[カーソル位置とその前の単語を入れ替え、入れ替え後カーソルは1つ前方に進む]、もよく使います。

たくさんの履歴の中から、目的の行を見つけるのはたいへんです。このときは、Ctrl+R[コマンド履歴を逆順に検索する]、を使いましょう。Ctrl+Rのあとに探したい文字列を入力していくと、現在行からインクリメンタルサーチ(1文字入力するごとに、直近の候補を表示する)できます。

ほかにも役立つ操作はたくさんありますが、いくつかを**表3**に挙げておきます。

キーボードによっては、エスケープキーが見当たらない、などということがあります。そのときは、Ctrl+[を使いましょう。Escと同じ意味です。同様に、returnキーやEnterキーが見つからない!?……ということはないと思いますが、Ctrl+JとCtrl+Mは改行と同じ意味です。

▼図2　emacsモードでのカーソル移動例2

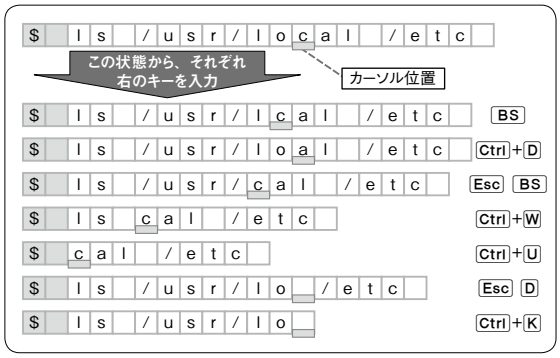


▼表2　bashのemacsモードでの編集操作

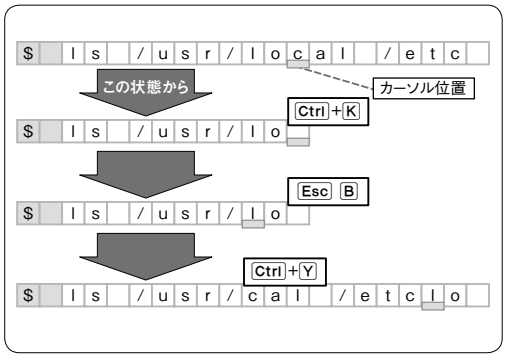
入力	操作内容
Ctrl+H／BS	カーソル位置の1文字後(左側)を消す
Ctrl+D／DEL	カーソル位置の文字を消す
Esc Ctrl+H／Esc BS	カーソル位置から後方(左側)の1単語を削除†
Ctrl+W	カーソル位置から後方(左側)のスペースまで削除†
Esc D	カーソル位置から前方(右側)の1単語を削除†
Ctrl+K	カーソル位置から行末まで削除†
Ctrl+U	カーソル位置から行の先頭まで削除†

※ BSはMacではdelete。DELはMacではFn＋delete。Ctrl＋W、Ctrl＋Uはsttyの節を参照。

▼図3　編集



▼図4　カット&ペースト



▼表3　そのほかの操作

入力	操作内容
Ctrl＋L	画面を消去して、現在行を表示する
Ctrl＋O	現在行を入力して、次の履歴行（なければ入力行）を表示する
Ctrl＋V	次の文字を文字として入力（コントロール文字などを入力したいとき）
Esc C	カーソルの後ろの単語の先頭の1文字を大文字にする
Esc U	カーソルの後ろの単語を大文字にする
Esc L	カーソルの後ろの単語を小文字にする
Esc .／Esc _	前のコマンドラインの最後の単語をカーソルの後ろに挿入する

▼表4　bashのemacsモードでの補完機能1

入力	操作内容
Tab	よろしく補完する
Esc ?	補完候補を一覧で表示する

▼表5　bashのemacsモードでの補完機能2

入力	操作内容
Esc ／	ファイル名として補完する
Ctrl＋X ／	ファイル名補完候補を一覧で表示する
Esc ~	ユーザ名として補完する
Ctrl＋X ~	ユーザ名補完候補を一覧で表示する
Esc $	変数名として補完する
Ctrl＋X $	変数補完候補を一覧で表示する
Esc @	ホスト名（/etc/hostsに登録されているなど、システムが知っているものに限る）として補完する
Ctrl＋X @	ホスト名補完候補を一覧で表示する
Esc !	コマンドとして補完する
Ctrl＋X !	コマンド補完候補を一覧で表示する

emacsモードでの補完機能

これがあるからbash、emacsは普及したと言っても過言ではないのが、**表4**の補完（Completion）機能です。Tabキー1回で補完できないときは、もう一度Tabを入力するとEsc ?と同じ効果が得られます。コマンドラインの文脈によって、ファイル名、ユーザ名、変数名、ホスト名、コマンドが補完されます。

補完する対象が明確な場合は、対象に合わせて**表5**のように入力すると誤った候補が出にくくなります。

STEP UP!　BIND―bindキーの割り当てを確認する

bashには、ここに挙げた以外にもたくさんの機能があります。どのキーが何の機能に割り当たっているかは、**bind**で（一応）確認できますが、出力行が多過ぎるので絞り込みが必要です。

```
$ bind -P↵
 すごい量出てきて、実用的ではない
$ bind -p↵
 -Pよりましだが、すごい量出てきて、実用的ではない
$ bind -p | egrep -va "(^#|^$|self-insert)"↵
 コントロール文字が入っているので-aを指定している
```

sttyで確認したキー割り当てとemacsモード[注7]で、衝突がある場合はどうなるでしょうか。試しにstty intr ^A（割り込みシグナルを^Aに割り当てる）としてみましょう。

```
$ stty intr ^A↵
          ↑^Vを入力してから^Aを入力する
$ stty -a↵   intrが^Aになっていることを確認
 ...略...
$ bind -p | egrep \\C-a↵
"\C-a": beginning-of-line
$ abcde^A
$       ^Aを入力すると、入力処理が中断されることが確認できた
```

注7）emacsモードやviモードでの編集は、readline(3)を使って処理しています。

Ctrl＋Aで入力行の先頭には移動しなくなりました。

伝統的なコマンドライン編集

History

コマンドラインをスクリーンエディタのようには操作できなかった時期に活躍したのが、historyコマンドです。emacsモードのキー操作で説明したbashの実行履歴を、コマンドによって扱う方法です。historyは、cshに由来するしくみです。

```
履歴を参照する
$ history↵
...略...
$ history 5↵  最新の5行
  524  ssh ubuntu0
  525  vim bestpractice
  526  ls
  527  git st
  528  history 5
```

過去の履歴をどれだけ保存するかは、シェル変数で指定できます。

```
% set history=11   cshの場合
$ HISTORYSIZE=10  bashの場合
```

履歴の参照には、!記号を使います(**表6**)。

履歴の実行の直後に:pを付ければ、コマンドを実行せずに、参照できます(**表7**)。

コマンド行中の特定の引数を指定することもできます。**表8**と**図5**を確認して、例を見てみましょう。

```
$ echo !!:0
$ !ls:2-$   範囲も使える。この場合2番めの引数から最後の引数という意味
```

履歴編集のために、csh由来のシェル変数に対する修飾子の一部、:h[パス名の先頭部分、最後の'/'の前まで]、:t[パス名の末尾部分、最後の'/'の後ろ]、:e[拡張子部分]、:r[拡張子を取り除いたパス]、が使えます(**図6**)。viでのexモードのように指定します(**表9**)。

```
$ ls -l   編集対象のコマンド
$ ^-l^-FC   -lオプションを-FCオプションに置き換える
$ !!:s/-FC/-l/   同様に、-FCを-lに置き換える操作

$ echo "abcde"↵
abcde
$ echo !!:gs/e/E/   前のコマンドライン中、すべてのeをEに置換
echo Echo "abcdE"
Echo abcdE
```

前の置換パターンを使って同じ変換をすることができます。&は、前回置換したときにマッチしたパターンに置き換えられます。!!:&、!str:&などとして使えます。

```
^での置換で&を利用する例
$ cp ab.c xyz.c↵
cp: ab.c: No such file or directory
$ ^xyz^&.backup↵   xyzをxyz.backupにする
cp ab.c xyz.backup.c
cp: ab.c: No such file or directory
```

▼表6　履歴の実行

コマンド	実行内容
!!	直前に実行したコマンド行
!str	strで始まる、直近のコマンド行
!#	hisotory番号＃番のコマンド行
!-#	＃個前のコマンド行
!?str?	strを含む直近のコマンド行

▼表7　履歴の参照(一部)

コマンド	実行内容
!!:p	直前のコマンドを実行せず、表示する。以降の編集対象にする
!str:p	strで始まる直近のコマンドを実行せず、表示する。以降の編集対象にする
!?str?:p	strを含む直近のコマンドを実行せず、表示する。以降の編集対象にする

▼表8　引数指定子

書式	内容
!*	引数全体
!^	最初の引数!!:1と同じ
!$	最後の引数

▼図5　引数指定子!!:#での、#の数の対応

```
コマンド   ls -l /bin | more
引数番号   0  1   2   3  4
```

▼図6　シェル変数に使う修飾子を使ったhistoryの例

```
$ ls Git/SD2016/12/*.txt↵   まずは、lsを実行
Git/SD2016/12/appendix.txt                    Git/SD2016/12/yogorecchimatta.txt
Git/SD2016/12/unix_command_explorer_12.txt
$ ls !$:h↵   最後の引数のhead部分だけ取り出す
ls Git/SD2016/12
Basic Computer Games 日本語版.pdf             tetris-bsd.jpg
Legacy8080_StarTrek_no25-20140725_manual.pdf trek.me.gz
appendix.txt                                  unix_command_explorer_12.txt
bangban.sh                                    yogorecchimatta.txt
test_int_limit.bash
$ echo !ls:$:t↵   lsで始まる最新の履歴のtail部分
echo 12
12
```

▼表9　履歴の編集

コマンド	実行内容
!!:s/abc/def/	直近のコマンド行中の最初のabcをdefに置き換える
^abc^def	（上と同じ）
!str:s/abc/def/	strで始まる直近のコマンド行中の最初のabcをdefに置き換える
!!:gs/abc/def/	直近のコマンド行中のすべてのabcをdefに置き換える
!str:gs/abc/def/	strで始まる直近のコマンド行中のすべてのabcをdefに置き換える

置換は、元が空文字だと、一番最近に使った変換パターンが使われます。

Fix Command—fc

fcは、historyを拡張したような内部コマンドです。viなどの対話型エディタが起動して、コマンドラインを編集できます。よほど長くて複雑なワンライナーを書いているときなどにしか出番はないかもしれませんが……。

コマンドライン編集が活躍するところ

コマンドライン編集が活躍するところは明らかですね。シェルと対話していれば、必ず活用するはずなのが今回の記事の内容です。historyは、古典的な編集のしくみですが、実行したコマンドラインや引数を記号で短く表記できるというしくみは、bashのコマンドライン編集機能とも競合するようなものではなく、うまく組み合わせて使えば作業がはかどります。

STEP UP! 履歴や補完、コマンドライン編集をする際の注意

プログラミングやデータの加工作業をしているときには、今回の記事で紹介した編集機能を駆使すると、とても作業がはかどり便利です。しかし、本番環境上などで、クリティカルなオペレーション作業をしているときは注意しなければなりません。不注意に履歴を使って、本来アクセスしてはいけないところへアクセスして、サービスにダメージを与えてしまったという事案をよく目撃します。

クリティカルな環境では、オペレーションルールとして、サービスに影響のある環境では、すべての作業記録を**script**コマンドなどを使って取るなどに加え、history操作は原則使わないなどの徹底が必要かもしれません。安全に作業することと、作業効率のバランスを取る施策について、オペレーションチームで、話し合ってみてはいかがでしょうか。

次回について

次回は、コマンドの実行や連結など、bashのしくみに深く踏み込みます。SD

今回の確認コマンド

【manで調べるもの（括弧内はセクション番号）】
stty(1), tty(1), ttyname(3), bash(1), csh(1), script(1), readline(3)（macOSのmanにはない）
【以下はbashのhelpコマンドを使って確認】
history, fc

Linux カーネル観光ガイド

第61回 Linux 4.4の機能
仮想PS/2デバイスなどを実装できるuserio

Text：青田 直大　AOTA Naohiro

Linux 4.11-rc3が3月19日にリリースされています。今のところ、Linux 4.11は比較的、更新内容が少ないように思われます。とはいえ、新しいシステムコール**statx**が導入されているなど、おもしろそうな機能もあります。

今回は、Linux 4.4の中から、ユーザランドで仮想PS/2デバイスなどを実装できるuserioについて紹介します。

userioとは

userioは、仮想的なシリアルデバイスをユーザランドで実装するための機能です。シリアルデバイスの例として、PS/2マウスや、ラップトップのタッチパッドなどがあります。こうしたデバイスをアプリケーションで使うためには、当然ながらデバイスドライバの開発が必要となります。しかしながら、これらの中には、実機が手に入れにくいものもあります。そうしたデバイスでドライバにバグが発生したとき、開発者の手元にその実機がなければ、デバッグは困難になります。userioを使うことで、そういったデバイスをユーザランドでエミュレートし、ドライバ開発・デバッグを容易にできます。

userioで仮想マウスを実装

さっそくuserioを使ってみましょう。今回は、シンプルなPS/2マウスをソフトウェア(**リスト1**)で実装しています。userioの操作は、**/dev/userio**に読み書きをすることで行います。書き込みの構造は**struct userio_cmd**(**リスト1**の①)で定義されています。この構造体の**type**にコマンドの種類、**data**にコマンドへの引数を渡します。

/dev/userioを開くと(②)、まずポートの接続とデバイスの登録を行います。ポートの接続は、**USERIO_CMD_SET_PORT_TYPE**コマンドを使います(③)。このコマンドはポートの種類を引数にとります。ここでは**SERIO_8042**とPS/2コントローラのポートに設定します(④)。ここで取り得る値は、**/usr/include/linux/serio.h**に定義されています。

ポートを設定したあとは、仮想デバイスを接続します。デバイスの接続は**USERIO_CMD_REGISTER**コマンドで行います(⑤)。このコマンドは、引数をとりません。

ここまででカーネルのPS/2ドライバには、新しいデバイスが接続されたように見えます。

接続されたデバイスの種類を調べるために、ドライバが種々のコマンドをデバイスに送信してきます。**recv()**関数(⑥)にあるように、**/dev/userio**を**read()**することで、ドライバが送ってきたコマンドがわかります。プログラムの残りの部分(whileループの中)では、簡単なPS/2マウスであるかのように、コマンドに返信していくことになります。

これらのコマンドの受信はrecv関数で行われます。この関数では、**/dev/userio**から最高でMAXBUF(=16)バイト分のデータを読んで、その内容を表示しています。コマンドへの返信は、先ほどと同様に**structuserio_cmd**の構造を**/dev/userio**に書いて返します。**type**は**USERIO_CMD_SEND_INTERRUPT**(⑦)になり、**data**は送信したいバイト数になります。

ここからは実行結果を見ながら、説明を進めていきましょう。プログラムを実行すると、ど

▼リスト1　userioでPS/2マウスを実装

```
#include <errno.h>
#include <fcntl.h>
#include <linux/serio.h>
#include <linux/userio.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>

#define ACK 0xfa
#define PS2MOUSE 0x00

#define GETID 0xf2
#define RESET 0xf6
#define SETRATE 0xf3
#define SETRES 0xe8
#define ENABLE 0xf4
#define DISABLE 0xf5

unsigned char state(int relx, int rely) {
  return (1 << 3) | (relx < 0 ? 1 << 4 : 0) | (rely < 0 ? 1 << 5 : 0);
}

int recv(int fd, char *buf) {
  const int MAXBUF = 16;
  int n;
  if ((n = read(fd, buf, MAXBUF)) < 0) {
    perror("read");
    return 1;
  }
  printf("-> ");
  for (int i = 0; i < n; i++)
    printf("%02x ", buf[i] & 0xff);
  printf("\n");
  return 0;
}

int send(int fd, unsigned char p) {
  struct userio_cmd cmd;  ……①
```

次ページにつづく ↗

▼リスト1　userioでPS/2マウスを実装（続き）

```
  cmd.type = USERIO_CMD_SEND_INTERRUPT;  ……⑦
  cmd.data = p;
  if (write(fd, &cmd, sizeof(cmd)) != sizeof(cmd)) {
    perror("INTERRUPT");
    return 1;
  }
  return 0;
}

pid_t start_reporting(int fd, int rate) {
  pid_t pid;
  if ((pid = fork()) == 0) {
    for (;;) {
      int relx = rand() % 3 - 1, ➥
rely = rand() % 3 - 1;
      send(fd, state(relx, rely));
      send(fd, relx & 0xff);
      send(fd, rely & 0xff);
      usleep(1000000 / rate);
    }
  }
  return pid;
}

int main(int argc, char const *argv[]) {
  struct userio_cmd cmd;  ……①

  srand(time(NULL));

  int fd = open("/dev/userio", O_RDWR);  ……②
  if (fd < 0) {
    perror("open");
    return 1;
  }

  // PS/2 ポートの設定
  cmd.type = USERIO_CMD_SET_PORT_TYPE;  ……③
  cmd.data = SERIO_8042;  ……④
  if (write(fd, &cmd, sizeof(cmd)) != sizeof(cmd)) {
    perror("SET_PORT_TYPE");
    return 1;
  }

  // デバイスの接続
  cmd.type = USERIO_CMD_REGISTER;  ……⑤
  cmd.data = 0;
  if (write(fd, &cmd, sizeof(cmd)) != sizeof(cmd)) {
    perror("REGISTER");
    return 1;
  }

  int done = 0;
  int rate = 100;
  pid_t pid = 0;
  while (!done) {
```

次段につづく ↗

```
    char buf[16];
    recv(fd, buf);  ……⑥
    switch (buf[0] & 0xff) {
    case GETID:
      send(fd, ACK);  ……⑧
      send(fd, PS2MOUSE);  ……⑨
      printf("GETID\n");
      break;
    case SETRATE:
      send(fd, ACK);
      recv(fd, buf);
      rate = buf[0] & 0xff;
      printf("SETRATE: rate = %d\n", rate);
      break;
    case SETRES:
      send(fd, ACK);
      recv(fd, buf);
      printf("SETRES: res = %d\n", 25 <<➥
 ((int)buf[0] & 0xff));
      break;
    case RESET:
      send(fd, ACK);
      printf("RESET\n");
      if (pid) {
        kill(pid, SIGTERM);
        wait(NULL);
      }
      pid = 0;
      break;
    case ENABLE:
      send(fd, ACK);
      printf("ENABLE\n");
      if (!pid)
        pid = start_reporting(fd, rate);
      break;
    case DISABLE:
      send(fd, ACK);
      printf("DISABLE\n");
      if (pid) {
        kill(pid, SIGTERM);
        wait(NULL);
      }
      pid = 0;
      break;
    default:
      printf("UNKNOWN\n");
      break;
    }
  }

  close(fd);
  return 0;
}
```

▼図1　プログラムの実行結果

```
$ sudo ./userio
# デバイスの識別（from keyboard driver）
-> f2
GETID  ……❶
# デバイスの識別（from mouse driver）
-> f2
GETID  ……❶
-> f6  ……❷
RESET
-> f3
-> 0a
# マウス拡張の検出
# Kensington ThinkingMouseの検出
...中略...
# synapticsの検出
-> e8
-> 00
SETRES: res = 25
-> e8
-> 00
SETRES: res = 25  ……❸
-> e8
-> 00
SETRES: res = 25
-> e8
-> 00
SETRES: res = 25
-> e9  ……❹
UNKNOWN
# ここで0x47を送れば、synapticsと認識
-> f6
RESET
```

次段につづく ↗

```
# GenPS/2の検出
...中略...
# logitechの検出
...中略...
# trackpointの検出
-> e1
UNKNOWN
# ここで0x01を返せば、trackpointと認識される
# FSPPS/2の検出
...中略...
-> f4
ENABLE
-> f6
RESET
-> ff
UNKNOWN
# IntelliMouse/Explorerの検出
...中略...
# 拡張の検出終わり
# 通常のマウスとして認識。デフォルトのパラメータを設定
-> f3  ……❺
-> 64
SETRATE: rate = 100
-> e8  ……❺
-> 03
SETRES: res = 200
-> e6  ……❻
UNKNOWN
# scalingの設定。今回は無視
-> f4  ……❼
ENABLE
```

のようなコマンドが送られてきたかを“-> <byte>”の行に続いて、そのコマンドの種類を出力させています(図1)。

PS/2デバイスが接続されると、ドライバはまずそのデバイスの種類を識別しようとします。識別にはGETID(0xf2)コマンドが使われます。プログラムではGETIDコマンドに対して、まずACK(0xfa)を送り返し(⑧)、次にPS/2マウスであることを示す0x00を送ります(⑨)。ちなみにここで、何も送り返さないとキーボードと認識される、というようにデバイス種別に合わせて0から2バイトの情報を送り返します。プログラム実行後に、2連続でGETIDが送られてきていますが(図1中の❶)、これは1つ目がキーボードドライバから、2つ目がマウスのドライバからのコマンドだと思われます。

デバイスが識別できると、ドライバはRESET(0xf6)を送り(❷)、各種パラメータをデフォルトに戻し、マウスの移動報告の割り込みを止めます。プログラムでは、ACKを返し、情報レポートのプロセスを止めるだけで、とくにパラメータを変えるなどはしていません。

マウスだということがわかると、ドライバはマウスの拡張機能の検出を行います。数バイトを送受信するPS/2のインターフェース上では、一度でどの種類の拡張マウスであるかを識別す

るのは困難です。そのため、各種拡張の検出が連続して行われます。

拡張検出の例としてsynapticsを見てみましょう。synapticsは、SETRESコマンドとGET INFO(0xe9)コマンドを使って、拡張の検出を行います。具体的には、4回連続でSETRESで解像度を25dpiに設定(❸)したあとに、GET INFO(出力中はUNKNOWN)を送ります(❹)。これに対して、デバイスが0x47を返してくると、ドライバはこのデバイスをsynapticsだと認識します。今回のプログラムでは、GETINFOを無視しているので、そのままほかの拡張機能の検出に進みます。

すべての拡張検出に失敗すれば、ドライバはマウスを拡張なしのマウスと認識して、パラメータの設定を行います。設定にはSETRATE(0xf3)コマンドと、SETRES(0xe8)コマンドを使います(❺)。どのコマンドも引数に1バイトをとります。SETRATEは、マウスの状態報告の頻度を設定するコマンドです。引数は1秒間に何回報告を上げるかを示します。SETRESは、マウス移動の解像度をdpiで設定します。

このプログラムでは、レートに関しては保存していますが、解像度のほうは表示するだけでとくに使っていません。また、SETSCALE11(0xe6)というコマンドも送られてきますが、ここでは無視しています(❻)。

パラメータの設定が終わると、ドライバはENABLE(0xf4)コマンドを送ってきます。これによってマウスから定期的に割り込みが入り、マウスの移動情報が伝えられるようになります(❼)。プログラムでは、`start_reporting`関数で、子プロセスを作ってレポートをさせています。

移動情報の報告は3バイト1組で送られます。1つ目のバイトは、マウスの移動量の正負や、ボタンのクリックなどを示すフラグで構成されています。残りの2つのバイトは、それぞれX軸Y軸方向のマウスの移動量を示します。プログラムでは、1回のレポートごとにX軸Y軸方向にそれぞれ-1から1までランダムに動かしています。

プログラムが動き、通常のマウスとして認識されると、マウスがランダムに動き出します。このとき、dmesgが/sys/class/input下を見ることで、確かにプログラムで作った仮想マウスが"PS/2 Generic Mouse"として認識されていることが確認できます(**図2**)。

まとめ

今回はuserlandで、PS/2マウスなどのserioデバイスをエミュレーションするためのフレームワークであるuserioについて紹介しました。サンプルプログラムでは、PS/2マウスをランダムに動かしてみました。

これだけではXレベルでやればいいので、あまりおもしろくはないですが(Waylandでも、そのまま動くのはちょっとおもしろい点ですが)、ps2emu[注1]というPS/2のコマンドを記録して、再生するプログラムもあります。これを使って、実デバイスのあるマシンで記録したコマンドを、別デバイスで再生すれば、実デバイスなしでの開発が可能になるというわけです。SD

注1) https://github.com/Lyude/ps2emu

▼図2　仮想マウスの認識を確認する

```
$ dmesg
...中略...
[19021.980205] input: PS/2 Generic Mouse as /devices/serio4/input/input22
$ cat /sys/class/input/mouse2/device/name
PS/2 Generic Mouse
```

ひみつのLinux通信

作)くつなりょうすけ
@ryosuke927

第39回 黒い画面は仕事中？

黒い画面はMS-DOS3.3時代からの付き合いと豪語する担当編集。それはちょっと方向が違うぞ、とツッコむ作家さんの楽屋落ちマンガが読めるのは本誌だけ！

to be Continued

仕事してるフリならターミナルの黒い画面が一番です。黒い背景にグレーな文字、9ポイントぐらいの文字サイズなら上司に背中取られてもコマンド打っているのかログを見てるのかSNSを見てるのかもわかりません。w3mというターミナルで使えて画像も表示できるブラウザがありますが、サボるのには便利ですぜ！　ブラウザのカラー表示もVimと思わせればよくて、コツは、screenやtmuxなどのターミナルマルチプレクサの使い方を覚えて、一方はVim、一方はw3mを使い、上司が来たときにVim画面に切り替えれば良いんです。ここまでくればターミナルから離れられませんね！　仕事してないってことは成果は出ませんよ？　え、……当たり前じゃないですか！

May 2017

NO.67

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
りゅうちてつや　RYUCHI Tetsuya　ryuchi@ryuchi.org

会場だけじゃない。SNSでも盛り上がるシェル芸勉強会

今回は2月に行ったシェル勉強会について報告します。

USP友の会・jus共催 シェル芸勉強会

■USP友の会・jus共催 シェル芸勉強会
【講師】鳥海 秀一（USP友の会）、
石井 久治（USP友の会）、
上田 隆一（USP友の会）
【日時】2017年2月11日（土）10:00～18:00
【会場】さくらインターネット 西新宿セミナールーム

USP友の会と共催し、「第9回初心者満足度ナンバーワン（当社調べ・調べてないけど）シェル勉強会/第27回sedこわいシェル芸勉強会」を開催しました。今回は32名の参加がありました。会場は西新宿にあるさくらインターネット㈱のセミナールームでした。

今回も有志によるサテライト会場が大阪と福岡に設けられ、ネットワークで接続されて開催されました。

■午前の部

午前の部では、鳥海さんと石井さんによる勉強会が開催されました。最初のセッションは、鳥海さんによる「黒い画面と戯れよう」というタイトルのセッションです。これは前回の勉強会[注1]と同じタイトルですが、前回はキーボード編だけで終わったため、今回はその続きとして画面編が開催されました。

まずは、前回と同様に黒い画面（一般的なCLIの画面のこと）についての簡単な説明がありました。背景が黒い画面に、白い文字を表示するだけでなく、文字に色をつける、点滅などの動作をさせる、画面の任意の場所に文字を出力させる、これらを組み合わせた動作をさせるなどのデモンストレーションを行いながら進められました。

続くセッションは、石井さんより「シェル芸入門日常会話編について」というタイトルで開催されました（**写真1**）。昨年の勉強会（2016年6月18日開催）[注2]でもシェル芸入門について説明されていましたが、さらに進めて日常会話のように使えるようにするための内容として講義されました。

シェル、引数の展開、標準入出力、パイプなどの動作としくみ、UNIX哲学を使ってシェル芸で問題を解くために必要になる知識や考え方について説明されました。その実例として過去のシェル芸勉強会で使われた問題や、sshで使うキーや、リモートでサーバのメンテナンスを行う実例を使い、注意すべき点などを交えてわかりやすく進められました。

写真1　午前の部は講義形式で実施

注1） 本連載第66回（本誌2017年4月号）参照。

注2） 本連載第60回（本誌2016年10月号）参照。

■午後の部

午後は、いつものとおり参加者同士で班を作り、お互いに協力して問題に取り組む形式で開催されました（**写真2**）。今回は参加者が多く、どのテーブルにも多くの人が着席していました。講師からは、近況、シェル芸の定義、勉強会の進め方について説明があったあと、今回のテーマとしてsedの使い方を取り上げることが述べられました。このテーマになった経緯として、本誌2017年1月号のシェル30本ノック特集をグループで執筆していたときに、講師（上田さん）以外のメンバーがsedを使いこなしているのを見て決めたそうです。

今回は、基本的な使い方を中心とした問題で構成したとのことです。また、sedはGNU sedを使って解答例を作成されたそうです。1問目を除いて置換の問題がなく、sedの全般的な機能を使う問題で構成されていました。sedの簡単な問題として大文字や小文字への置換から始まり、さまざまなsedに関する問題が全部で8問用意されていました。

途中、sedだけでなくvimのコマンドを使って問題を解いてしまう解答例などが会場から示され、vimシェル芸という単語が作られる場面もありました。プロトタイプ宣言がないC++のソースファイルにおいて、関数の並びを書き換えてコンパイルする問題などで活躍し、会場で話題になりました。

シェル芸勉強会では、YouTubeで中継されたり（**図1**）、Twitterを通じて解答例が投稿されたり（**図2**）するのでSNS上でも話題になっており、SNSをうまく活用している事例の1つになっていると感じました。

写真2　午後の部は班に分かれて問題に取り組む

図1　YouTubeでの中継
（「シェル芸勉強会」で検索すると過去の勉強会も視聴できる）

■懇親会・LT大会

無事に8問の課題が終わったあと、さくらインターネットのご好意により、同じ会場でビアバッシュ形式での懇親会が開催されました。毎回、懇親会では、自己紹介やLT（ライトニングトーク）大会が開催されています。これらは自由な歓談と楽しい雰囲気の中で進められ、LT大会は発表者がそれぞれ得意な分野について発表をします。今回もどれも楽しいものになりました。

今後も2ヵ月ごとの開催が予定されています。雑用を増やさずに全員勉強会を楽しむという方針ですので、必要なことが何かを考えながら協力していきたいと思います。SD

図2　Twitterで投稿される問題や解答例

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

●Hack For Japanスタッフ
及川 卓也　OIKAWA Takuya
Twitter @takoratta
佐伯 幸治　SAEKI Koji
Twitter @widesilverz

第65回 防災4.0ハッカソンと国土強靱化ワークショップ

日本に住む我々は自然災害と常に背中合わせに生きています。東日本大震災からすでに6年が経ちましたが、その6年の間にも熊本地震が発生し、夏には常にどこかの地域が水害に襲われています。このように自然災害と生きる宿命の国である日本は、国家としても備える必要があります。今回は、そのような国としての取り組みである、防災4.0のハッカソンと国土強靱化ワークショップについてお伝えします。

防災4.0ハッカソン

自然災害にたびたび襲われる日本は国としても、その対策を進めてきました。防災1.0と呼べる最初の一歩は、災害対策基本法を制定した1959年の伊勢湾台風。防災2.0は、阪神・淡路大震災のあった1995年。このときに、建築物の耐震改修促進法や被災者生活再建支援法が制定されました。また、ボランティア元年と後から呼ばれるようになったように、公的機関からの支援だけでなく、自分たちで支援し合う自助と共助という考えが根付いたのも、この阪神・淡路大震災のときのことです。そして、防災3.0が東日本大震災。

東日本大震災以降、防災の分野でもさまざまな取り組みがされていますが、今後の気候変動がもたらす災害の激甚化（げきじんか）を考えると、従来の災害に対する取り組みだけでは対策は不可能で、公助だけでなく、自助と共助の形で国民一人一人が災害リスクと向かい合う必要があります。これが防災4.0の考えです。内閣府の防災4.0未来構想プロジェクトの資料[注1]には次のように書かれています。

> *行政（国・地方公共団体）のみならず、地域、経済界、住民、企業等の多様な主体のそれぞれが、防災を「自分ごと」として捉え、相互の繋がりやネットワークを再構築することで、社会全体の復元力（レジリエンス）を高め、多様な災害に備える社会を、「防災4.0」の目指す姿として追及していきたい。*

防災4.0関連のイベントはいくつか行われているのですが、今回筆者（及川）が関わったのは、1月21日と22日の2日間にわたり開催された「防災4.0ハッカソン」です。

ハッカソンの冒頭には、内閣府の松本洋平副大臣が登壇するという熱の入れようで、今回のハッカソンのお題である「自分の周りで災害が発生したときに必要なサービスやアプリ」を「自分ごと」として考えてほしい旨をお話しされていました。

イベント会場となったYahoo! LODGEには防災食が用意されていました（**写真1**）。ハッカソンなどで食事が用意されることは多いですが、防災食とはとてもユニークでした。

受賞した作品から

参加した9チームの中から、最優秀賞と優秀賞の2つを紹介します。

◆**写真1　防災食が用意された会場**

注1　http://www.bousai.go.jp/kaigirep/kenkyu/miraikousou/pdf/yushikisya_honbun.pdf

●最優秀賞「自分サバイバル」

最優秀賞は、今回のテーマである「自分ごと」として防災を考えるアプリである「自分サバイバル」に贈られました。これは、首都直下型地震が発生した場合には、発災直後に80%が死亡するという被害想定をもとに、発災直後を生き延びるためのアプリです。

このアプリを利用することで、過去の災害を生き延びることができた知恵である災害体験を共有し、環境や状況に応じて相応(ふさわ)しい情報を得ることができます。また、発災時には地図上に危険個所なども表示することで、生き延びる確率をあげようと工夫されています(**図1**)。

アプリに過去の被災体験を登録することができますが、沿岸地域ならば津波に関係する情報、がけ崩れの危険性がある場所にはそのような情報など、より状況に応じた情報が登録されることが期待されると審査員からはコメントがありました。また、発災時に情報を共有することよりも、まずは逃げることが大事なので、その点への考慮があるとさらに良いものになると感じられました。

●優秀賞「Support Chain」

優秀賞が贈られたのは「Support Chain」というアプリです。これは東日本大震災でも熊本地震でも問題となった指定外避難所での情報共有を解決するためにメッシュネットワークを利用するというアプリです。

表1を見ればわかるように、指定外避難所ではあらゆるものが不足することが予測されますが、もっとも重要なものは情報です。炊き出し情報や他生活支援情報などをどのように取得するかが課題です。今ではインターネットがあるので、それを用いることが考えられますが、発災直後には携帯網がダウンすることも考えられますし、バッテリーを使い果たし、スマートフォンなどの携帯機器を使えない状況に陥る人が出る可能性もあります。

Support Chainは、避難所の中の少なくとも1名がインターネットから正しい情報を取得したならば、それをあとはメッシュネットワークで数珠つなぎのように情報を伝達していくことを実現します(**図2**)。あらかじめ、取得する公式情報は登録しておきます。たとえば、自治体のTwitter情報などがそれにあたります。

Support Chain自体の情報共有・拡散はインター

◆**図1　アプリ「自分サバイバル」は発災時には災害関連の情報を地図上に表示**

◆**表1　指定避難所と指定外避難所(Support Chainの発表スライドより)**

	指定避難所	指定外避難所
首都直下地震による想定避難者数(都内)	225万人	121万人
避難所の数(都内)	4,146ヵ所	???ヵ所
避難所の整備状況	行政職員の派遣、通信手段の整備、避難者名簿の作成、水・食料の備蓄、発電装置の整備	どれもなし

※表にあるように、指定避難所に避難する人数の約半数が指定外避難所に避難すると想定されているにもかかわらず、その扱いは指定避難所に比べて大きく不足している

◆図2　Support Chainは、情報の入手と拡散、安否情報の登録、避難所内での情報共有という3つの機能を備えている

ネット接続がない状態でも可能ですので、インターネットとの接続が不安定な場合でも利用できます。

メッシュネットワークを使った災害時の支援というアイデアはすでにありますが、このアプリはそれと公式情報の拡散という点がユニークでした。

◆　◆　◆

ハッカソンの模様は運営を行ったHackcampよりレポートが公開されています[注2]ので、詳しくはそちらもご覧ください。

国土強靭化ワークショップ

次に、筆者(佐伯)が活動している減災インフォ[注3]がサポートした「国土強靭化ワークショップ」を紹介します(なお紙幅の都合上、具体的な内容をお伝えきれないので、詳しく知りたい方は国土強靭化ワークショップのイベントレポート[注4]をご覧ください)。

国土強靭化[注5]とは、災害に対して“「強さとしなやかさ」を備えた国土、経済社会システムを平時から構築するという発想に基づき継続的に取り組む”という意味合いを持った言葉で、国が取り組んでいる政策課題となります。推進役を担っている内閣官房国土強靱化推進室では、国土強靭化に関連した活動や人々の交流を拡大していくことを目的に、昨年4回にわたる「国土強靭化ワークショップ」を開催しました。

各ワークショップについて

4回のワークショップとも、専門家から話題提供をいただき、その話題に沿った内容で参加者がグループとなり、ワークショップをしながら意見や考えをまとめて発表するという形式で行われました。当たり前ではあるのですが、参加者の災害に関する考え方やとらえ方は多様で、その多様さをどのように共有して、1つにまとめていくかを体験するという点で、ワークショップはとても意義深く筆者には感じられました。専門家からの話題提供は次のような内容になります。**写真2**にまとめたグラフィックレコードも参照してください。

●第1回「支え合えるこれからのコミュニティを考える」

1回目では、浦野愛氏(認定NPOレスキューストックヤード常務理事)を迎え「災害につよい、つながりとは？」と題して、阪神・淡路大震災時の実態

注2　http://hackcamp.jp/articles/bousai/
注3　https://www.gensaiinfo.com/
注4　http://www.cas.go.jp/jp/seisaku/kokudo_kyoujinka/workshop.html
注5　http://www.cas.go.jp/jp/seisaku/kokudo_kyoujinka/

から、生活者に何が起こっていたのか、人が支え合うこととはどのようなことなのか、事例を交え提言をいただきました。

●第２回「情報をどう伝え、どう受けとめて命をまもるか？」

第2回目では、菅井賢治氏（NHK報道局 災害・気象センター災害担当部長）を迎え「災害時に情報を伝えるということ」をテーマにお話いただき、東日本大震災時の報道から当時の伝え方を振り返りました。

●第３回「いざというときに支え合えるつながりをつくるには？」

第3回目では、山崎亮氏（コミュニティデザイナー／㈱studio-L 代表取締役）を迎え「支え合える人のつながりをつくるには？」をテーマに、参加型をキーワードにしたコミュニティづくりや、香川県観音寺市の事例などのお話いただきました。

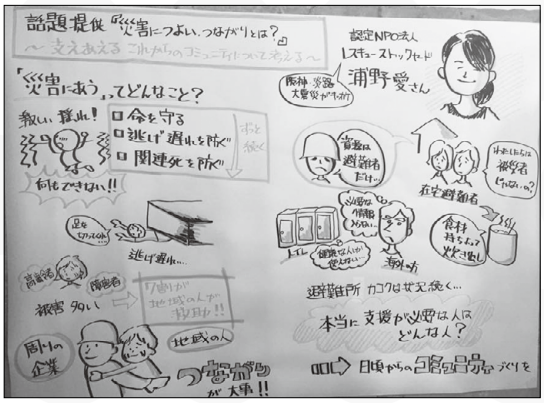
●第４回「命を守るため私達にできる５つの行動を考えよう！」

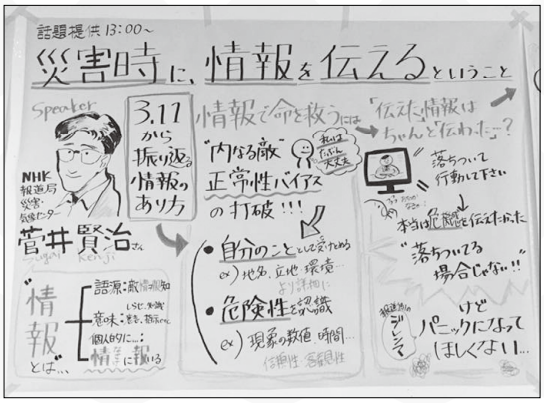
第4回目では、秋冨慎司氏（医学博士 防衛医科大学校 准教授）を迎え「自分と人の命を守る」をテーマに、東日本大震災時に現場で起きていたこと、現状の災害対応に関する課題などのお話をいただきました。

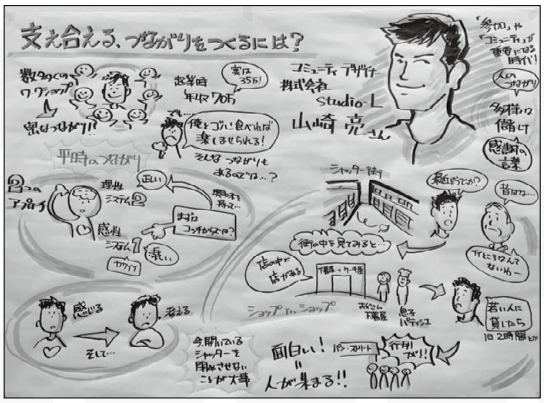
◆ ◆ ◆

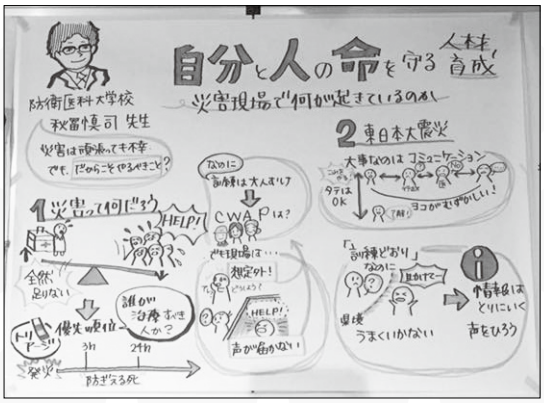
以上、簡単に紹介しましたが、2017年も何かしらの活動があればまた誌面にて紹介したいと思います。イベントなどの開催情報は、国土強靭化推進室のTwitterアカウント[注6]から入手できますので、興味のある方はそちらからご覧ください。SD

◆写真２　ワークショップでは、グラフィックファシリテーションやグラフィックレコーディングを活用
（左上：第１回、右上：第２回、左下：第３回、右下：第４回）









注6　https://twitter.com/resiliencejpn

温故知新
IT むかしばなし

第65回

OS-9
～究極の8bit CPUのために開発されたOS～

速水 祐（はやみ ゆう） http://zob.club/ twitter @yyhayami

はじめに

OS-9と聞くと、AppleのMac OS9やiPhone/iPadのiOS 9を思い浮かべると思います。同じ名ですが、1980年代初頭にこれらとは異なる8bit CPU、MC6809の末尾の数字“9”をつけて命名されたOS-9が登場し、高機能オペレーティングシステムとして大きな注目を浴びました。今回は、このOS-9のお話をしましょう。

BASIC09とOS-9

1979年に登場した、モトローラ社の8bit CPUであるMC6809で動作させる、構造化BASICであるBASIC09がモトローラ社とマイクロウェアにより開発されていました。この言語の開発・実行環境としてOS-9（Operation System for 6809）も同時に開発され、1980年にBASIC09とともに登場します。

当時マイコンで動作する主要プログラム言語は、マイコンのROM上に載って動作するBASICインタプリタでした。BASIC09は、それまでのBASICで問題になっていた行番号を使わずに、見通しの良い構造化プログラムを記述できる優れた言語になっていました。Pascal言語に似た構文になっており、数値の変数への代入はPascal的に“a:=2”のように記述することもできました。プログラム作成において、モジュール単位に分割し、コール時はパラメータを引数として渡すことができたので、再帰呼び出しのプログラムを記述することも可能でした。

BASIC09は、コマンドを入力して実行するシステムモード、プログラムを入力するエディタモード、それを実行するモード、そしてデバッグモードが用意されていて、これらのモードを切り替えてプログラムの開発・実行が行われます。エディタモードで入力された段階でプログラムソースはちょうどJava言語と同じように、中間コードに変換され、高速に動作します。同じマシンに搭載されていたF-BASICと比べて5倍以上の動作スピードを実現していました。

完成したBASIC09プログラムは、PACKコマンドにより、OS-9のコマンドモードで直接実行できるファイルに変換され（6809のバイナリコードにコンパイルされるのではない）、通常のコマンドと同じように使うことができました。Javaのようなランタイムライブラリはなく、OS-9自体がこのBASIC09の中間コードプログラムを実行する環境として構築されていたと考えられます。

OS-9の究極の機能

OS-9は、豊富なアドレッシングモードなどのMC6809の特徴を活かして、コンパクトなサイズに洗練されたUNIXライクな究極の機能を実現していました。

Shellを中心としたUNIXから受け継がれた構造でマルチタスク、マルチユーザに階層構造のディレクトリなど、当時普及していた8bit CPUのOSであるCP/Mとは、機能面で大きく差がありました。さらにUNIXと同様なファイルとデバイスを同一化するユニファイドI/Oシステムやパイプとリダイレクトをサポートしていました。

OS-9のUNIXと異なる大きな特徴に、8bit CPUの小さなメモリ空間を効率的に利用するためのモジュール構造でメモリ管理を行っていることがありま

す。OS-9自身も含めてすべてのプログラムがメモリモジュールの集合体で成り立っており、実行状況によってメモリモジュールが再配置されて、プログラムが実行されます。モジュールはひとつひとつが独立してポジションインディペンデント[注1]で自由なアドレスに配置でき、さらにリエントラント(再入可能)になっており、複数のプログラムから同時利用が可能でした。

OS-9は仮想記憶機能がないため、搭載された主記憶容量以上の記憶空間が使えない問題がありました。しかし、逆にリアルタイム性が確保でき、リアルタイムOSとして価値が高まり、さらにデバイスドライバ自体もファイルシステムと完全に切り離されているためROM化も容易になっていました。そのため、OS-9は当初のMC6809用の高機能なOS-9から、信頼性が高いコンパクトなOSとして長期間劣悪な環境で使用される制御装置などに利用されました。

FM-11とOS-9

1982年11月に、富士通からホビーユーザパソコンFM-7と同時にFM-11が発表されました。そして1984年2月にFM-11AD2が発表され、ついにOS-9 Level 2が標準搭載されるのです。

OS-9は64KBのメモリで動作するレベル1と2MBのメモリ管理が行えるレベル2があります。OS-9を本格的に使うためにはさすがに64KBでは少な過ぎ、レベル2の使用が前提になります。FM-11は、まさにOS-9レベル2を動作させるのに最適なマシンであり、メインメモリの管理は、OS-9で活用できるような構造になっています。

MC6809は、モトローラ社から専用のMMU(Memory Management Unit: メモリ管理ユニット)であるMC6829が準備されており、これを使えばブロックサイズ2KB、4セグメントで2MBの物理メモリエリアを管理し、さらにユーザ空間とシステム空間の分離を実現できました。それに対してFM-11のMMUは、ブロックサイズ4KB、16セグメントで1MBの物理メモリエリアを管理できるものになっており、ユーザ／システム空間の差異はありません。**図1**にそのメモリ管理手法を示します。16×16 MMR(Memory Mapping Register)の管理メモリテーブルが存在し、ここに格納された8bitの値が物理アドレスの上位8bitとなり、ブロックサイズ4KBの12bitアドレスと合わせて20bitで、1MBのメモリ空間をアクセスします。

4bitのMSR(Memory Segment Register)によりMMRの列位置を、6809CPUのアドレスバスの上位4bitにより行位置を指定することで、MMRの特定の8bitデータが特定され、それを使って上位8bitアドレスが表現されます。

図1では、MSRの値が$2で6809CPUのA15～A12が$Bにより、MMRの$3Cのデータを指定し、6809CPUのA11～A0の$123が合わさり、アドレス$3C123にアクセスしています。

OS-9では、MMRの複製となるDATAイメージをプロセスごとに持っており、自由度の大きなメモリ管理がOSとハードが一体化して実現できていました。

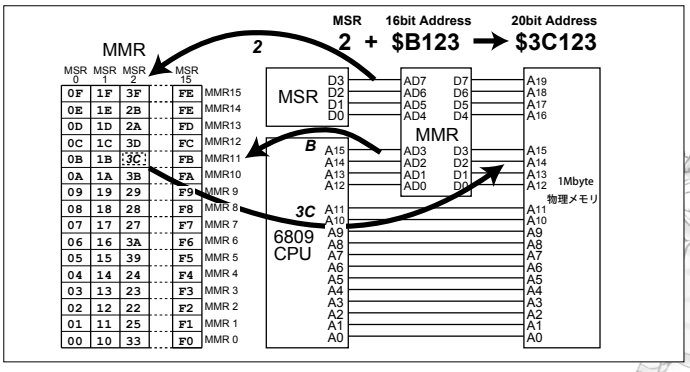
現在のOS-9

OS-9は、その高機能性が評価されて、MC68000への移植を皮切りに、MIPS、PowerPC、さらにARMにも移植され、組み込み分野を中心に現在でも利用されています[注2]。設計が優れた情報技術は、長く利用され、生き続けるのです。SD

注1) メインメモリのどこに置かれても、絶対アドレスにかかわらず正しく実行できるプログラムコード。

注2) 最新の組み込みARMボードArm adilloシリーズにも対応している。

▼図1 MMRを使ったメモリアクセス



※記事の執筆にあたり、RetroPC Foundationの貴重な資料を参考にさせていただきました

Report

サイコロとカード、JIRA Softwareを使ったゲーム感覚のスクラム体験
リックソフト ハンズオンセミナーレポート

リックソフト㈱が定期的に行っているハンズオンセミナー「もっとJIRA Softwareを活用するためにスクラムを学ぼう」に参加する機会が得られたので、その内容を体験をふまえてお伝えする。セミナーは無料で受けられ、受講時間は約2時間。同社の会議室の一室で行われた(セミナー概要は記事末の同社Webサイトを参照)。

まず自分を含めて3名のチームを結成。筆者とチームを組んだお二人はどちらもソフトウェア開発の品質保証部門に所属されており、ウォーターフォール型の開発は十分に経験あり。一人はすでにいくつかの案件でアジャイル開発を経験済みで、今回の参加目的は、自社に導入済みのJIRA Softwareをアジャイル開発の管理にうまく使う方法を知ること。もう一人は、自社開発のプロジェクト管理ツールを使っており、スクラム開発は未経験なので体験をしに来られたとのこと。

スクラム開発の中でも、作業工数の見積もりとスプリントをゲーム感覚で体験するのが本セミナーの目的で、次のような段階を踏んで進められる。

①作業工数の見積もり
②スプリントを回す
③作業の振り返り

○料理を題材に、作業の見積もりを議論

作業工数の見積もりには、身近な“料理”が題材に。「さんまの塩焼き2匹」(さんまに塩をふる/片面を焼く/ひっくり返して、片面を焼く/皿に盛る)をベースに「ホットケーキ2枚」と「ハンバーグ2個」の工数を見積もる。料理経験の少ない中年男性3人が、ホットケーキのほうが材料を混ぜるぶんだけ工数がかかるとか、ハンバーグはただ混ぜるだけではなく形を整える必要がある、などと熱い(?)議論をかわして工数を決定していく。

この見積もり作業にはプランニングポーカーと呼ばれるカードを使ったが(右ページ参照)、共通に持っている数値があることで、合意形成がしやすく、見積もりのブレが少なくなるメリットがあるとのこと。

▶裏返しで積んであるのと、その下で表になっているのがProblemカード。左に並べてあるのが各メンバーが出したSolutionカード

○サイコロを使ったゲーム感覚のスプリント体験

セミナーで行ったスプリントは、

- 1日につき午前と午後の2回サイコロを振る(各人)
- スプリントの期間は月曜日の午前から金曜の午後まで
- サイコロを1つ振って出た目(「1」以外)の数ぶんの時間を、担当する課題の作業時間から減らす
- 残り作業時間がなくなったら、その課題を「完了」のボードに移動して次の課題に取りかかる
- サイコロの目が「1」だった場合は、その課題で障害が発生したことになる
  → 作業時間を減らすことはできない
  → Problemカードを1枚めくり、書かれた障害内容への対応策を各人のSolutionカードから1枚ずつ提案。どの対策が最適か話し合ったうえで1つに絞る

というのを繰り返す。金曜の午後が終了したら、スプリントの振り返りを行う。

障害発生のイベントがゲームにスパイスを効かせているが、もう1つおもしろいルールが「残業」。作業の進行状況が思わしくないと判断した場合は、午後のサイコロを振るときに「残業します!」と宣言すればサイコロを2個振ることができる。これでグッと時間が縮められる可能性が上がるが、それとひきかえに翌日の午前のサイコロ振りでは「1」に加えて「2」でも障害が発生してしまう。

本当の仕事だったら「障害発生」も「残業」も笑えない事態だが、ゲームだと思うとワクワクしながらやってしまっている自分に気づく。

スクラムで重要なコミュニケーションについては、工数見積もりでそろわなかったときと、障害への対応策について相談するときに疑似体験ができる。限られた時間ではたいした話はできないが、プランニングポーカーやSolutionカードのおかげで素早く決められる。

筆者は“細かいスプリントを繰り返す”という方法に対して、いちいち作業が中断されて面倒そうな先入観をもっていたのだが、セミナーの疑似体験ですら達成感のようなものを感じられたし、バーンダウンチャートやベロシティチャートのグラフを理想に近づけたい、というモチベーションもわいた。ただしゲームのように楽しくできるかどうかは、チームの雰囲気にかかっているな、というのも理解できた。

今後もリックソフトでは継続的にセミナーを開催する予定。スクラムについての情報はあまたあるが、机上で学ぶよりも体験のほうが圧倒的に腑に落ちる面があることを、このセミナーを受けて感じられた。SD

スクラム体験ハンズオンセミナーの流れ

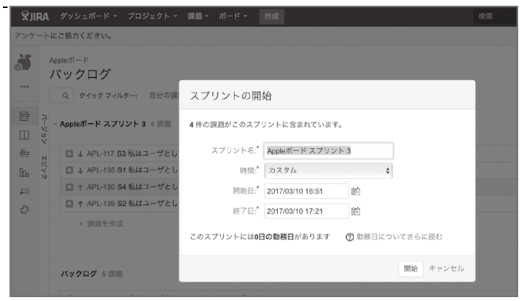
❶プランニングポーカーを使った作業工数の見積もり

作業工数の見積もりに使ったのが右写真のプランニングポーカー。各人に同じものが配られる。作業工数の見積もりをストーリーポイントという数値で表現するのがスクラム流。メンバーの合意によって各作業の見積もりを行うため、各自でポイントを考え、選んだカードを同時に見せ合う。皆が同じポイントにそろえば決定、そうでなければ自分が見積もったポイントの理由を説明したあと、あらためてカードを出し合う。

▲▶プランニングポーカーで見積もりを合議

❷スプリントの作成

プロダクトバックログはJIRA Softwareにあらかじめ登録されているものを使用（❶で行った料理の作業工数は使わない）。右の画面からスプリントを開始する。Appleというのは、筆者を含む3名のチーム名（各メンバー名はApple01、02、03）。ダイアログのうしろに見えているのは、プロダクトバックログから作った1週間分（実際の時間は30分間）のスプリントバックログ。3名での1週間のスプリントなので、4課題で3w2d4h（3週間と2日と4時間）と設定した。



▲スプリントを開始するための設定ダイアログ

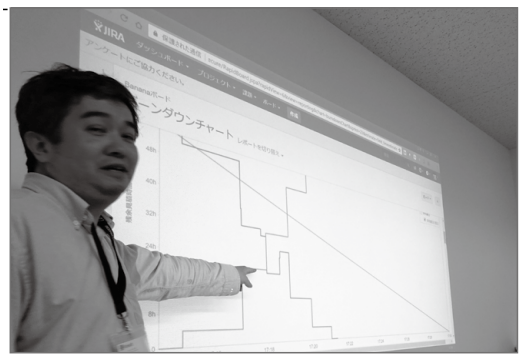
❸スプリント実施中

右の画面がスプリント中のJIRA Softwareのカンバン画面。「完了」のステータスにある課題APL-111は、障害が発生（フラグ付き）したものの、残業することによって何とか達成。メンバーのApple02が進行中の課題APL-112も、残り時間0mとなっていて、この課題は「完了」のステータスへ移動させる。Apple03さんが取り組んでいる課題APL-110も、障害が発生した課題であることがわかる。付箋紙を貼り付けるように、課題をドラッグ&ドロップで操作。

▲スプリント実行中のJIRA Softwareの画面

❹スプリントの振り返り

金曜日の午後のサイコロを全員が振り終えたら、リックソフトのエンジニアがそれぞれのチームのバーンダウンチャートとベロシティチャートをスクリーンに表示して、それぞれのグラフの見方を解説。実際のスクラム開発でも、これらのグラフを見ながら作業の進み具合や滞ってしまった原因、もともとの見積もりの適切さなどを検討して、次のスプリントへの改善を行う。



▶バーンダウンチャートで進行度合いを振り返り

CONTACT
リックソフト㈱ URL https://www.ricksoft.jp/　　**リックソフト開催のセミナー案内** URL https://www.ricksoft.jp/seminar/

Report

JAWS Days 2017、開催

3月11日、TOC五反田メッセ（東京都）にてクラウドサービスAmazon Web Services（AWS）のユーザイベント「Jaws Days 2017」が開催された。気になったセッション、会場の様子をレポートする。

コミュニティで拓く、パラレルキャリアへの道

7年間アマゾンウェブサービスジャパンでAWSのマーケティング統括を務め、JAWS-UGも立ち上げた小島英揮さんが発表したのは、「パラレルキャリア」という働き方について。小島さんは2016年8月にアマゾンウェブサービスジャパンを退社し、今年1月からInstaVR、Stripe、Rider's Garage、EventRegist、Moongiftという5社で、CMOやエヴァンジェリスト、コミュニティマネージャを務めている。これは何も、副業を多く抱えているというわけではなく、「複業＝パラレルキャリア」という働き方で、すべての会社で本業として働いているとのこと。この働き方によって、インプットの量、そして外の物差しに触れる接点が格段に増えたという。1つの軸を持ち、多くのインプットが欲しい人にお勧めの働き方とのことだった。

サーバーレスの今とこれから

昨年10月には「ServerlessConf Tokyo」を主催するなど、日本のサーバーレスの中心人物といっても過言ではない吉田真吾さん。セッションでは、現在のサーバーレスを取り巻く状況、サーバーレスを導入するにあたってのアドバイスについて話した。吉田さんによると、サーバーレスにとって重要となるのが、

- AWS、Microsoft Azureといった「プラットフォーム」
- Serverless Framework、AWS Serverless Application Modelなどの「開発・運用フレームワーク」
- 実際にサーバーレスな開発を行う「アプリ開発者」

という三位一体のエコシステムだ。この内、日本ではまだまだ「アプリ開発者」の数が少なく、情報発信が少ないと、吉田さんは懸念を持っているという。またセッションの最後では、サーバーレスを導入するうえでの注意点が話された。サーバーレスは基本的に外部サービス頼りの手法のため、

- 自分自身でそのサービス内部の問題は解決できず、また新機能を実装することはできない
- 突然のサービス終了も十分にあり得る

ということについて考えなければならないとのこと。これら問題については、「できるだけシェアが高いサービスを選択し、そのサービスで用いられている技術をよく理解しておくことが大事」とアドバイスをした。


▲小島 英揮さん


▲吉田 真吾さん

AWSカルタ・AWS麻雀

休憩スペースでは、AWSの各サービスのロゴが書かれた絵札と牌を使った「AWSカルタ」「AWS麻雀」のプレイコーナーが設けられていた。AWSカルタでは、読み手の方がサービスの絵札を組み合わせながら、参加者にデザインパターンを提案するということもされていたとのこと。AWS麻雀では、AWSの導入企業から名前を取った「スシロー」「Docomo」といった役があり、牌の組み合わせがそのまま、その企業の社内システムのデザインパターンになっている。

▲AWS麻雀


▲AWSカルタ

CONTACT
JAWS DAYS 2017 URL http://jawsdays2017.jaws-ug.jp/

Software

アクロニス・ジャパン、
個人向けバックアップソフトの最新バージョン「Acronis True Image 2017 New Generation」発表

アクロニス・ジャパン㈱は2月15日、「Acronis True Image 2017 New Generation」を発表した。

「Acronis True Image」は、Windows PCやMac対応の個人向けバックアップソフトウェア。最新の「2017 New Generation」では、次のような機能強化が成された。

- **ランサムウェアに対してリアルタイム保護が可能に**
  コンピュータ上の異常な振る舞いを特定し、悪意のあるプログラムがユーザのデータやバックアップ、バックアップソフトウェアに損害を与えないよう阻止
- **ブロックチェーン技術によるデジタル証明書**
  オリジナルのコンテンツが改ざんされていないかどうかを検証し、デジタル証明書を発行

価格は次のとおり（いずれも1年版で、1TBクラウドストレージが付属する）。アクロニスのオンラインストア限定での販売となっている。

- 1台のコンピュータ用（1年版）＝9,980円（税込み）
- 3台のコンピュータ用（1年版）＝14,980円（税込み）
- 5台のコンピュータ用（1年版）＝15,980円（税込み）

CONTACT
アクロニス・ジャパン㈱ URL http://www.acronis.com/ja-jp

Service

東陽テクニカ、
米Spirent社のセキュリティテストソリューションをアピール

㈱東陽テクニカは、パートナーである米国Spirent Communications社（以下Spirent社）のセキュリティソリューションについて、3月7日に説明会を開催した。

Spirent社は通信機器の性能試験分野で多数の商材を持つが、この日はセキュリティテストソリューションの「CyberFlood」とセキュリティ診断サービスの「Security Labs」を紹介した。

CyberFloodは、金融、医療、政府、自動車、モバイル、IoT、生活インフラなど、高いセキュリティが要求される企業や製品のテストを行うためのソフトウェア（執筆時点では専用機器の導入が必要）。Sandbox、WAF、IPS、多要素認証、権限管理、アンチウィルスなどといったセキュリティ対策の統合管理ができ、Webブラウザでの監視・操作が可能。また、脆弱性を発見するための内部への侵入テスト（ペネトレーションテスト）の機能を備えている点が大きな特徴である。ネットワークの性能試験も実行でき、セキュリティとパフォーマンスのバランスを最適化したい企業ニーズに対応する。

SecurityLabsは、Spirent社のセキュリティコンサルタント自身がペネトレーションテストを行うことで、リスクの発見、マニュアルの診断、改善点の提案などを行う。

CONTACT
㈱東陽テクニカ URL http://www.toyo.co.jp/

Topic

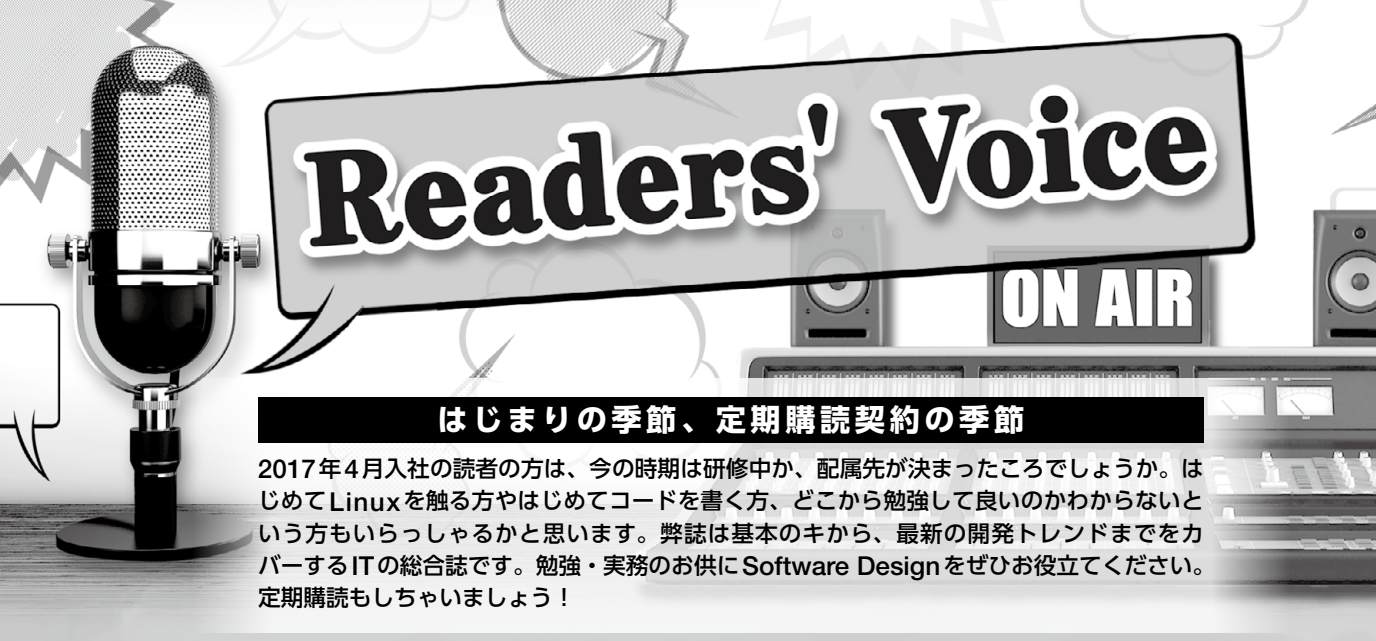
F5ネットワークスジャパン、
「2017年版アプリケーションデリバリの状況」の調査結果を発表

F5ネットワークスジャパン合同会社は3月7日、「2017年版アプリケーションデリバリの状況」の調査結果を発表した。

本調査では、F5ネットワークスのユーザ2,000名以上が、クラウドの採用状況からセキュリティの課題、DevOps、SDNなど、アプリケーションデリバリに関する質問に回答している。調査のハイライトは次のとおり。

- ファイアウォールでだけではなく、アプリケーションを狙った攻撃の増加にも対応する動きがある。導入が計画されているセキュリティサービスのトップ3は、DNSSEC（25%）、DDoS対策（21%）、WAF（20%）
- 回答者の5分の4が「すでにハイブリッドクラウドを活用」と回答。また、約3分の1が「年内にパブリッククラウドIaaSソリューションを採用する」と回答
- 企業や組織はすでに、平均で14のアプリケーションサービスを導入しており、2016年の11に比べて増加
- 半数以上の回答者が「APIを実装したインフラおよびテンプレートの重要性は高い」と回答。SDNに関しては、スケーラビリティ確保と運用コスト低減が、利用目的のトップ2に挙げられた

CONTACT
F5ネットワークスジャパン合同会社 URL https://f5.com/jp



はじまりの季節、定期購読契約の季節

2017年4月入社の読者の方は、今の時期は研修中か、配属先が決まったころでしょうか。はじめてLinuxを触る方やはじめてコードを書く方、どこから勉強して良いのかわからないという方もいらっしゃるかと思います。弊誌は基本のキから、最新の開発トレンドまでをカバーするITの総合誌です。勉強・実務のお供にSoftware Designをぜひお役立てください。定期購読もしちゃいましょう！

2017年3月号について、たくさんの声が届きました。

第1特集 ログ＆データ分析基盤入門

近年ますます重要になってきた、データ分析の基盤構築のための特集でした。Fluentd、Embulk、Hadoop、Treasure Data Serviceのしくみや使い方を解説しました。

あらためて、勉強になりました。Treasure Dataさんは本当にすごいアウトプットで半端ないです……。
n0tsさん／東京都

最近のOSS界隈のデータ分析に関するライブラリの紹介がまとまっていて、良かったです。 **yoneさん／兵庫県**

ログは永遠のテーマ。基盤と言われると読んでしまう。 **ひで魚さん／東京都**

ログとデータ分析をやったときのログを再度分析し、そのときのログをまた分析し……。結局役立てられない我が部署でした！ **南雲さん／埼玉県**

これまでログは「なんとなく取ってそのまま」という感じだったので、まずはFluentdとEmbulkで収集を始めてみようと思います。 **犬棟梁さん／埼玉県**

膨大なデータも、適切に収集・分析しないと意味のないものになってしまいます。「データはとりあえず集めてはいるけど……」といった方は、基盤から見直してみてはいかがでしょうか。

第2特集 CIでインフラ開発を効率化

ソフトウェア領域の開発手法「継続的インテグレーション（CI）」「アジャイル」「スクラム」をインフラ領域にも応用し、開発を柔軟・効率的にするための手法を紹介。Jenkinsを中心にしたCI基盤の作り方も解説しました。

理想はありますが、ある程度の組織になると開発手法の変更は難しいですよね。 **reiさん／東京都**

失敗した例も書いてあってよかった。
宮川さん／大阪府

開発手法を、例を見ながら学べるのはすごく良い。 **緑川さん／東京都**

インフラ部分をアジャイルでどう進めるかについて、勉強になりました。
ぴょへいさん／大阪府

アジャイル開発の予定の立て方がとても良いです。異業種でも使えそうなスケジューリングですね。
Tayuさん／千葉県

昔からの開発手法を変更するのは一苦労ですよね。記事にもあったとおり、CIにはデメリットも存在するそうで、相性を考えながら、まずは小規模なプロジェクトで試してみるというのもありかもしれません。

第3特集　どうなってる？ なりすましメール対策

銀行や公的機関を騙（かた）ったあやしいメールは、どのように対処すれば良いのか。SPF、DKIM、DMARK、S/MIME、安心マークといった技術を取り上げ、エンジニア・プログラマができるなりすましメール対策を解説しました。

年々と言うよりも日々巧妙化しているので、明日は我が身の思いで読みました。 **永作さん／東京都**

普段、自分のメールボックスに関してはキャリアの迷惑メールフィルタとGmailのフィルタでおおむね用が足りているのですが、知らないところでこんなしくみが動いていたとは驚きです。 **オトさん／神奈川県**

3月号のプレゼント当選者は、次の皆さまです

①ディレクターズ　10周年記念レディースバッグ（ネイビー）
榎本理恵様（東京都）

②Repro　Tシャツ＆ステッカー
伊原匠様（神奈川県）

③『アルゴリズムクイックリファレンス 第2版』
小川啓吾様（千葉県）、富澤周平様（栃木県）

④『あたらしい人工知能の教科書』
Tak様（千葉県）、糀田利貴様（滋賀県）

⑤『まつもとゆきひろ 言語のしくみ』
嶋田順夫様（三重県）、ねこねこ様（東京都）

⑥『その「エンジニア採用」が不幸を生む』
tekitoizm様（東京都）、田代海霞様（福岡県）

※当選しているにもかかわらず、本誌発売日から1ヵ月経ってもプレゼントが届かない場合は、編集部（sd@gihyo.co.jp）までご連絡ください。アカウント登録の不備（本名が記入されていない場合、プレゼント応募後に住所が変更されている場合など）によって、お届けできないことがあります。2ヵ月以上、連絡がとれない場合は、再抽選させていただくことがあります。

こういうセキュリティの情報はインプットだけで済まさず、一般の利用者にアウトプットできるようになりたい。
エゾモモンガさん／滋賀県

なりすましメールについて、家族や知り合いにも詳しく説明してあげられそうです。　ＹＹさん／神奈川県

SPF/DKIMなど、なんとなくしか理解できてなかった部分が体系的にまとまっていてわかりやすかったです。
のりぃさん／埼玉県

誰もが一度は見たことがあるなりすましメール。恐らくその手口はこれからも巧妙化していくことでしょう。まずは、手元でどのような対策が取れるのかを知っておく必要があります。

特別企画　Amazonログイン＆ペイメントのしくみ

Amazon.comの決済サービス「Amazonログイン＆ペイメント」は、自社開発のECサイトなどにも組み込むことができます。導入方法と、導入によるメリットについて解説しました。

実績のある環境の裏が紹介されるのはヨイ。　下平さん／東京都

タイムリーで知りたかったことなので、とても助かった。　なおきさん／千葉県

Amazonはよく利用するので、その有効利用法を見直すことができました。
はこじろうさん／群馬県

この機能について初めて知った。システムに取り入れることを検討したい。
齊藤さん／東京都

もはや使ったことのない人はいないであろうAmazon.com。そこで使われているしくみを利用できるということで、期待値も大きいのではないでしょうか。

特別企画　PG-Stromの構造と機能、そしてその威力とは【前編】

PG-Stromは、PostgreSQLのオープンソースの拡張モジュールです。GPUを用いて、集計・解析系の処理を非同期・並列に実行でき、高速化が見込めます。前編ではPG-Stromの概要と、AWSでの環境構築を解説。

最新のデータ高速集計・解析システムの手法がわかった。　自営隊さん／長崎県

最先端プロダクトのしくみを、無理なくわかりやすく理解できました。後編の実装がとても楽しみです。
takiponeさん／東京都

PostgreSQLの処理にこういった拡張モジュールも使えると初めて知りました。　牧田牧場さん／東京都

GPUを利用したプログラムの特性がよくわかりたいへん参考になりました。処理したいデータの扱い方をよく考えようと思いました。
出玉のタマさん／大阪府

非常にハイエンドな技術ですが、興味を持った、使ってみたいという意欲的な読者の方も多いようです。最近ではGPUを利用できるクラウドサービスも増えてきているので、ぜひ挑戦してください。

コメントを掲載させていただいた読者の方には、1,000円分のQUOカードをお送りしております。コメントは、本誌サイト**http://sd.gihyo.jp/**の「読者アンケートと資料請求」からアクセスできるアンケートにてご投稿ください。みなさまのご意見・ご感想をお待ちしています。

次号予告

Software Design
June 2017

2017年6月号
定価（本体1,220円+税）
184ページ

5月18日発売

［第1特集］プラグラミングに即戦力!
「エディタ実践入門」
Vim，Emacs，Atom，Visual Studio Code
「読んだらすぐにプログラミングに活用」＋「相性の良いエディタを見つける」

［第2特集］プログラミングを仕事に役立てていますか?
エンジニアのためのPython入門

※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

「SOURCESレッドハット系ソフトウェア最新解説」（第9回）は都合により休載させていただきます。

お詫びと訂正

以下の記事に誤りがございました。読者のみなさま、および関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

■2017年4月号
●P.125　連載「Vimの細道　第17回」右段
［誤］\.+[]{}^$　［正］\.[]^$※エスケープが必要な文字の一部です。

■2017年3月号
●P.164　連載「Linuxカーネル観光ガイド」左段2行目
［誤］読み手がrcu_read_unlock()などで変数を更新する前、そして書き手がsynchronize_rcu()で変数を読む後には、メモリバリアが必要です。
［正］読み手がrcu_read_unlock()などで変数を読む後、そして書き手がsynchronize_rcu()で変数を更新する前には、メモリバリアが必要です。

SD Staff Room

●健診で注意されて健康のため歩くことを始めた。帰宅時に4km弱。紀尾井町あたりは江戸時代への誘い。そういえば以前に当社のあった四谷三丁目は『鬼平犯科帳』の火付盗賊改方の寮があるという設定だった。権野助坂から広尾あたりまで歩く話もある。フィクションだけど心は鬼平の気分で歩き続ける。（本）

●2,000万画素以上の閲覧環境が整っていないのに気付く。撮影側は、KPは6,016×4,000。α6000は6,000×4,000。RX100M2とRX10M3は5,472×3,648。表示側は、フルHDは1,920×1,080。iPad miniは2,048×1,536。4Kモニタは3,840×2,160。iMac 21.5 Retinaで4,096×2,304。道は遠い。（幕）

●お風呂場のシャワー&カランの締まりが悪くなって、普通に締めただけでは水滴がポタ、ポタ、と。それでも「ここだ!」というピンポイントの締め方を習得して、しばらーく使っていました。が、ついに重い腰をあげてパーツを交換。なんと、締まる位置に回すだけで水がぴたりと止まるんですよ。すごい!（キ）

●新年度ですね。私は異動もないため、会社では年度替わりの新鮮さはほぼゼロ。ただ最近、たまに家庭の事情でいつもと違う路線で出勤するのですが、これは景色も車内の客層も違って新鮮です。おかげで仕事の意欲も増しそうでしたが、定期券が使えずこづかいが減ったことで、結局意欲も±0です。（よし）

●小誌の過去記事をまとめた「インフラエンジニア教本――セキュリティ実践技術編」が発売中! SSLやファイアウォールといった基礎技術、脆弱性スキャンなどの応用スキル、さらにはなりすましメール対策まで取り上げた意欲的な1冊です。表紙の紫の幾何学模様は、よく見ると鍵の形になっています。（な）

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

［E-mail］
sd@gihyo.co.jp

［FAX］
03-3513-6179

※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。

Software Design
2017年5月号

発行日
2017年4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
根岸真理子

●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。