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
4
2017
2017年4月18日発行
毎月1回18日発行
通巻384号
（発刊318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

》すぐにLinuxを使いこなしたい 》サーバーレスとは何か?

新卒準備号
第1弾

基礎から学べば役に立つ!

新人のための Linux 入門 OS基本操作編

Special Feature 1

Special Feature 2

はじめてのサーバーレスアーキテクチャ

AWS Lambdaと
Microsoft Azure Functions
で体験

Extra Feature 1 新年度も絶対活用Wireshark! Jamesのセキュリティレッスン
Extra Feature 2 ニフティクラウド mobile backendで何ができるか?（前編）
Extra Feature 3 好評 PostgreSQLのGPU拡張モジュール PG-Strom解説（後編）

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)

4
2017
［目次］
Software Design

contents

新人歓迎企画 第1弾

Special Feature 1 第1特集

目的から考える! 実作業から学ぶ Linux 入門 OS操作編

基礎から学べば役に立つ! 017

第1章 ルーキー君、Webサーバを動かしてみる! 宮原 徹 018

第2章 ディレクトリ構造と役割がわかりますか? 佐野 裕 028

第3章 ルートとユーザの違いがわかりますか? 佐野 裕 034

第4章 設定ファイルが書けますか? 鶴長 鎮一 040

第5章 コマンド、スクリプトが使えますか? 鶴長 鎮一 051

4
2017
［目次］
Software Design

Special Feature 2 第2特集

はじめてのサーバーレスアーキテクチャ

AWS LambdaとMicrosoft Azure Functionsで体験 061

Part1 “サーバーレス”とはなにか? 仲山 昌宏 062
Part2 [実践] AWS Lambda 永田 明 066
Part3 [実践] Microsoft Azure Functions 東 聡志／日本マイクロソフト㈱監修 076

Extra Feature 一般記事

[ニフティクラウドmobile backend] mBaaSのしくみ紹介[1]
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 中津川 篤司、小山 哲志 086
PG-Stromの構造と機能、そしてその威力とは[後編]
どのようにGPUは集約演算を実行するのか? 海外 浩平 096
[短期集中連載] Jamesのセキュリティレッスン[9]
IPsecの暗号化通信を復号してみよう! 吉田 英二 106

Catch up trend

うまくいくチーム開発のツール戦略[7]
もっとJIRAを活用しよう! いろいろな業務で使ってみる 樋口 晃、大塚 和彦 172
[Special Interview] 3年で300%急成長のベンチャーがSE・PGを大募集 編集部 176

àla carte アラカルト

ITエンジニア必須の最新用語解説[100] Amazon AI 杉山 貴章 ED-3
読者プレゼントのお知らせ 016
SD BOOK REVIEW 141
SD NEWS & PRODUCTS 178
バックナンバーのお知らせ 180
Readers' Voice 182

contents

Column

及川卓也のプロダクト開発の道しるべ［6］
PMとプロジェクトマネージメント 及川 卓也 ED-1

digital gadget［220］
デジタルガジェットと触覚 安藤 幸央 001

結城浩の再発見の発想法［47］
アイコン 結城 浩 004

［増井ラボノート］コロンブス日和［18］最終回
発見プログラミング 増井 俊之 006

宮原徹のオープンソース放浪記［14］
大阪は裏なんばが熱いし、OSC大阪も熱かった 宮原 徹 010

ツボイのなんでもネットにつなげちまえ道場［22］
「さくらのIoT Platform β」の通信モジュールを使ってみる（後編） 坪井 義浩 012

Hack For Japan〜エンジニアだからこそできる復興への一歩［64］
南相馬市小高の学生生活を楽しくしたい!〜小高ハッカソン 小泉 勝志郎、高橋 憲一 166

温故知新 ITむかしばなし［64］
MC68000〜黎明期の高性能16bit CPU〜 速水 祐 170

ひみつのLinux通信［38］
ねんどまつ くつなりょうすけ 181

Development

使って考える仮想化技術［11］
仮想マシンネットワークの利用 笠野 英松 112

RDB性能トラブルバスターズ奮闘記［14］
ExcelでSQL操作のイメージをつかむ法 生島 勘富、開米 瑞浩 118

Vimの細道［17］
置換コマンドの基本をおさらい mattn 124

るびきち流Emacs超入門［35］
安定の横断検索パッケージanything.el るびきち 129

書いて覚えるSwift入門［24］
型の探求 小飼 弾 132

セキュリティ実践の基本定石［42］
WordPressサイトの改ざんで問われる自主運営の是非 すずきひろのぶ 136

OS/Network

SOURCES〜レッドハット系ソフトウェア最新解説［8］
OSスキルアップのススメ 小島 啓史 142

Debian Hot Topics［45］
地道に進むDebian 9に向けたパッケージ更新 やまねひでき 145

Ubuntu Monthly Report［84］
Let's noteにUbuntu GNOME 16.04 LTSをインストールする あわしろいくや 148

Unixコマンドライン探検隊［12］
ゲーム&ジョーク特集（その2） 中島 雅弘 152

Linuxカーネル観光ガイド［60］
Linux 4.4の新機能〜NVMe SSDを活用するI/O pollingとLightNVM 青田 直大 158

Monthly News from jus［66］
できるエンジニアはクリスマスも旧正月も勉強会 りゅうちてつや、法林 浩之 164

［広告索引］
システムワークス
http://www.systemworks.co.jp/
前付
創夢
http://www.soum.co.jp/
裏表紙
日本コンピューティングシステム
http://www.jcsn.co.jp/
裏表紙の裏

［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］
oxilixo / 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

GIHYO Digital Publishing

技術評論社の本が電子版で読める!

https://gihyo.jp/dp

イチオシの1冊!

プロが教える情報セキュリティの鉄則

—守り・防ぐ・戦う科学

香山哲司，小野寺匠　著

2,480円　EPUB　PDF

サイバー攻撃の手口が高度化・巧妙化し，ウィルス対策ソフトを代表とする従来の防御策では役に立たなくなったということは，すでに多くの専門家から指摘されているとおりです。しかし，サイバー攻撃だけではなく，内部の不正やうっかりミスによっても情報は流出します。

なぜ，こうも情報流出が続くのでしょうか?

それは，情報セキュリティに対する目的の誤解，全体感のとらえ方や優先順位の決め方の誤り，科学的なアプローチ不足など守る側の課題が多いことも一因していると筆者は推察しています。そして，より本質的な課題として，担当者が情報セキュリティに関して体系立てて教育されることなく，多くの場合，ほかの業務と情報セキュリティを兼務の形で，悪く言えば片手間で担っている点を挙げられるのではないでしょうか。

少し厳しい言い方をすれば，情報セキュリティの不備は以前からありましたが，サイバー攻撃をはじめとした多くの情報流出事故によってその不備があらわになってきたのでしょう。

「では，具体的にどうすればいいのか?　どのような技術で対策すればいいのか?」という疑問や悩みに解を示すのが，本書の目的です。

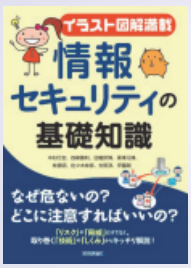
https://gihyo.jp/dp/ebook/2017/978-4-7741-8764-8

あわせて読みたい

なぜ，あなたのウェブには戦略がないのか?

—3Cで強化する5つのウェブマーケティング施策

EPUB　PDF



【イラスト図解満載】情報セキュリティの基礎知識

EPUB　PDF

プロになるなら身につけたいプログラマのコーディング基礎力

EPUB　PDF

仕事の問題地図

〜「で，どこから変える?」進捗しない，ムリ・ムダだらけの働き方

EPUB　PDF

他の電子書店でも好評発売中!

amazon kindle　honto　楽天 kobo　ヨドバシカメラ www.yodobashi.com　BookLive!

お問い合わせ

〒162-0846　新宿区市谷左内町21-13　株式会社技術評論社　クロスメディア事業部

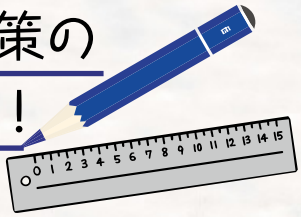
TEL：03-3513-6180　メール：gdp@gihyo.co.jp

法人などまとめてのご購入については別途お問い合わせください。

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。

あなたを
合格へと導く
一冊があります！

効率よく学習できる
試験対策の
大定番！



岡嶋裕史　著
A5判／680ページ
定価（本体2880円+税）
ISBN978-4-7741-8502-6

エディフィストラーニング株式会社　著
B5判／400ページ
定価（本体2980円+税）
ISBN978-4-7741-8505-7

角谷一成・イエローテールコンピュータ　著
A5判／544ページ
定価（本体1680円+税）
ISBN978-4-7741-8495-1

山本三雄　著
B5判／576ページ
定価（本体1480円+税）
ISBN978-4-7741-8496-8

岡嶋裕史　著
A5判／432ページ
定価（本体1880円+税）
ISBN978-4-7741-8501-9

庄司勝哉・吉川允樹　著
B5判／320ページ
定価（本体1480円+税）
ISBN978-4-7741-8504-0

大滝みや子・岡嶋裕史　著
A5判／744ページ
定価（本体2980円+税）
ISBN978-4-7741-8500-2

加藤昭・高見澤秀幸・矢野龍王　著
B5判／464ページ
定価（本体1780円+税）
ISBN978-4-7741-8503-3

金子則彦　著
A5判／640ページ
定価（本体3200円+税）
ISBN978-4-7741-8433-3

金子則彦　著
A5判／560ページ
定価（本体3200円+税）
ISBN978-4-7741-8387-9

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

「仕事が進まない、終わらない…」の原因を集大成して見える化！

仕事の問題地図

- ☑ やることが多くて、最初から終わる気がしない…
- ☑「他人の力を借りたら負け」だと思っている
- ☑ なんで自分がこんな仕事をしなくちゃいけないのか、さっぱりわからない…
- ☑ やり方を変えようとしても、他部署が抵抗してくる！
- ☑ 問題があっても、空気を乱す発言はしない

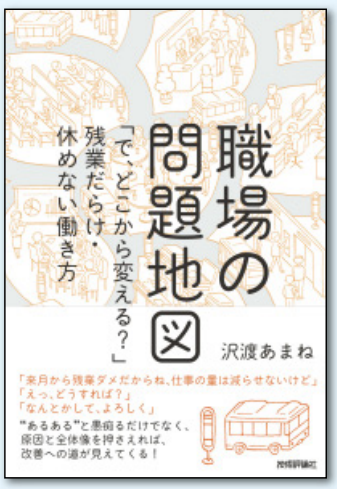
あれ、ウチの会社のこと…？

と、1つでも当てはまれば、きっとこの本は役立ちます。

「で、どこから変える？」
進捗しない、ムリ・ムダだらけの働き方
ISBN978-4-7741-8774-7

各書とも **沢渡あまね** 著　四六判／224ページ　**定価**（本体**1480円**+税）

大好評！働き方改革の新バイブル



職場の問題地図

「で、どこから変える？」
残業だらけ・休めない働き方

ISBN978-4-7741-8368-3

大反響11刷!!

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

最新刊！

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。

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

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

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

サーバ／インフラエンジニア養成読本 基礎スキル編
福田和宏、中村文則、竹本浩、木本裕紀 著
定価 1,980円＋税　ISBN 978-4-7741-7345-0

Laravelエキスパート養成読本
川瀬裕久、古川文生、松尾大、竹澤有貴、小山哲志、新原雅司 著
定価 1,980円＋税　ISBN 978-4-7741-7313-9

Pythonエンジニア養成読本
鈴木たかのり、清原弘貴、嶋田健志、池内孝啓、関根裕紀、若山史郎 著
定価 1,980円＋税　ISBN 978-4-7741-7320-7

事例から学ぶ情報セキュリティ
中村行宏、横田翔 著
定価 2,480円＋税　ISBN 978-4-7741-7114-2

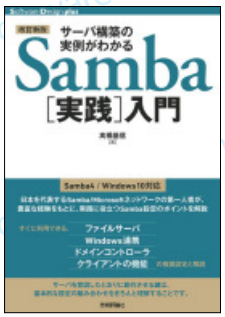
データサイエンティスト養成読本 R活用編
養成読本編集部 編
定価 1,980円＋税　ISBN 978-4-7741-7057-2

Javaエンジニア養成読本
きしだなおき、のざきひろふみ、吉田真也、菊田洋一、渡辺修司、伊賀敏樹 著
定価 1,980円＋税　ISBN 978-4-7741-6931-6

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



髙橋基信 著
A5判・256ページ
定価 2,680円（本体）＋税
ISBN 978-4-7741-8000-7

高宮安仁、鈴木一哉、松井暢之、村木暢哉、山崎泰宏 著
A5判・352ページ
定価 3,200円（本体）＋税
ISBN 978-4-7741-7983-4

斎藤祐一郎 著
A5判・160ページ
定価 2,280円（本体）＋税
ISBN 978-4-7741-7865-3

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・168ページ
定価1,980円（本体）＋税
ISBN 978-4-7741-8360-2

養成読本編集部 編
B5判・232ページ
定価2,080円（本体）＋税
ISBN 978-4-7741-8385-5

養成読本編集部 編
B5判・200ページ
定価2,080円（本体）＋税
ISBN 978-4-7741-8034-2

養成読本編集部 編
B5判・112ページ
定価 1,980円（本体）＋税
ISBN 978-4-7741-7992-6

技術評論社

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6回 PMとプロジェクトマネージメント

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

PMって言ったら

この連載をとおして「PMと言ったら、プロダクトマネージャー」と言われるように、職種としての認知を高めるべく努力していますが、世間一般ではまだまだ「PMと言ったら、プロマネ（プロジェクトマネージャー）」という認識が大きいでしょう。自己紹介のときに、「PMをしています」と言うと、「プロマネですね？」と聞かれることも多々あります[注1]。何も、PMという略語の所有権を主張するわけではないですが、実際に職務上もオーバーラップすることが多い、このプロジェクトマネージメントとPM（この場合はプロダクトマネージャー）の関係について今回は考えてみたいと思います。

プロジェクトとプロダクトの違い

プロジェクトマネージメントとは、その名のとおりプロジェクトの管理です。IT業界だけでなく、さまざまな業界や個人やコミュニティで、成し遂げたいことがあった場合、それを実現するための計画を立案しますが、その計画がプロジェクトであり、その管理がプロジェクトマネージメントです。身近なところでは、家族旅行もプロジェクトであり、ビルの建設計画もプロジェクトです。プロジェクトマネージメントとプロダクトマネージメントはどちらもマネージメント＝管理ですので、ここではまずプロジェクトとプロダクトの違いについて明確にしてみましょう。

プロジェクトマネージメントの手法は業界や組織によって異なりますが、PMI（米国プロジェクトマネージメント協会）が、網羅的に概念や手法、ベストプラクティスなどをまとめたプロジェクトマネジメント知識体系ガイド（PMBOK）がよく知られています。

このPMBOKではプロジェクトは「独自のプロダクト、サービス、所産を創造するために実施される有期性の業務である」と定義されています。

この定義の中にプロダクトが入ってきていますので、プロダクトはプロジェクトの一部とも考えられますが、一方で、重要なのは「有期性」というキーワードです[注2]。これは開始日と期限が決まっていると言い換えても良いでしょう。製造業などではQCD（Quality、Cost、Delivery）管理という考えが古くからありましたが、PMBOKでもこのDelivery、すなわち納期についてはスケジュール管理としてプロジェクトマネージメントに必要な知識体系として位置付けています。

一方、プロダクトはその製品が続く限り、終わりはありません。その製品の最初のプロジェクトは、おそらく最初のリリースを行うまでで終わりでしょう。しかし、その後も第2世代、第3世代とリリースが続くたびにプロジェクトは組織され、そして終了していきます。このよ

注1）ちなみに、以前、何のコンテキストもなしに英語でPMの紹介をしたら、Prime Minister（首相）の略かと聞かれて、びっくりしたことがありました。

注2）ところで、筆者は、この「有期性」という言葉をPMBOKのプロジェクトの定義以外ではほとんど聞いたことがありません。「有期性」という言葉を聞いたら、プロマネ関係の人だと判断しても良いかもしれません。

うに考えると、プロジェクトはプロダクトに包含されると考えるのが自然です。

先の家族旅行プロジェクトの例で考えるならば、プロダクトは家族に相当するのかもしれません。家族という製品(製品に例えるのは不謹慎かもしれませんが)を進化させるさまざまな試みの1つとして、家族旅行というプロジェクトがあると考えてみるのはどうでしょうか。家族旅行プロジェクトが終わっても、家族という製品が終わるわけではありません。それと同じように、1つのプロジェクトが終了しても、プロダクトは続いていくのです。

プロジェクトマネージメントとプロダクトマネージメント

家族というプロダクトにおける家族旅行プロジェクトというメタファーは身近な例として出しましたが、一般的にはプロダクトは複数のユーザに対して提供されるものです。この本の読者ならば、ソフトウェア製品と言われるものを考えてもらえばいいでしょう。

多くの現場では、あえてプロダクトマネージャーと呼ばなくても、プロダクトマネージャーのような立場で働いている人はいるはずです。ディレクターやプロデューサー、編集者(編集長)などなど。このような肩書の人たちはプロダクトマネージャー的な仕事をしているはずです。このようにプロダクトとプロジェクトの違いが明確になれば、そのマネージメントの違いも自ずと明らかになります。

プロダクトマネージャーは、この連載で当初から説明しているように、なぜ作るのかと何を作るのかを中心にビジョンやミッションを明確にし、将来にわたってのロードマップを明確にします。プロダクトが終了しない限り、その役割も続きます。

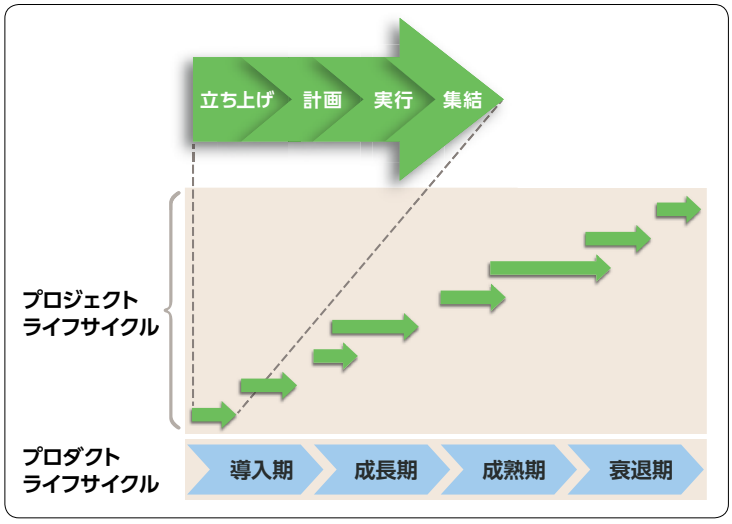
一方、プロジェクトマネージャーは、プロダクトの一部や一時期のある機能や改善を行うためなどに、期限が決められた(有期性のある)複数のタスクの計画を立案し、それを実施します。タスク完了とともに、役割は終了します。

プロジェクトはプロダクトに包含されますので、プロダクトマネージャーがプロジェクトマネージャーを兼任することも多くあります[注3]。プロジェクトもプロダクトも必要なタスクをライフサイクルという言い方で整理することもできます。**図1**を見れば、プロジェクトマネージャーとプロダクトマネージャーのそれぞれの役割の違いが明確になるでしょう。

今回はPMとプロジェクトマネージメントについて考えてみました。次回は、このプロジェクトマネージメントの中でも、昨今の開発現場でよく利用されるアジャイル開発、その中でもとくに、スクラム開発におけるPMの役割を考えてみたいと思います。プロマネと同じくらいに、その違いを聞かれることが多いプロダクトオーナー(PO)もその中で取り上げる予定です。SD

注3) ただし、スクラム開発などでのスクラムマスターはプロダクトマネージャーが兼任するのは好ましくありません。これについては次回解説します。

▼図1 プロダクトライフサイクルとプロジェクトライフサイクル



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。

第100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。

Amazon AI

AWS による AI プラットフォーム「Amazon AI」

Amazon AIは、米Amazon Web Services（AWS）が提供している機械学習および深層学習テクノロジを活用したサービス群の総称です。2016年11月に開催された「AWS re:Invent 2016」において発表されたもので、これを利用することで、開発者は手軽にAIテクノロジを活用した高精度なサービスを開発し、自前のアプリケーションに組み込むことができるようになります。

本稿執筆時点では、Amazon AIのもとで次に挙げる4つのサービスが提供されています。

◉ Amazon Lex

音声やテキストによる会話型インターフェースを構築するためのサービス。Amazon Echoシリーズなどに搭載されているインターフェースAmazon Alexaと同様の技術で、深層学習テクノロジによる自動音声認識と自然言語認識を活用した対話的なアプリケーションを開発し、自前のサービスのインターフェースとして組み込むことができる

◉ Amazon Rekognition

深層学習に基づいて画像の認識や解析を行うサービス。Amazon.comのPrime Photosにおける画像分析のノウハウを活用して開発された技術で、画像内の物体を検知して分類したり、画像内の顔を認識して表情や特徴を検出することができる。また、複数の画像を比較して同一人物であるかどうかを判断することや、多数の画像の中から指定された顔が写る画像を探し出すことなどもできるという

◉ Amazon Polly

テキストを音声に自動変換するサービス。日本語を含む多数の言語をサポートしており、同じつづりの単語でも、前後の文章から適切な発音を判断することができるという。コンソールへの文章入力やAPI経由での呼び出しによって利用することが可能で、応答時間が短いためリアルタイムな対話にも活用できるとのこと。変換された音声のオフライン再生やオーディオファイルとしての保存にも対応している

◉ Amazon Machine Learning

機械学習を利用した予測を自前で実装するための開発者向けサービス。仮想化ツールやウィザードによって簡単にカスタムの機械学習モデルを作成することが可能なため、機械学習アルゴリズムに関する知識やノウハウを持たない開発者でも利用できる点が大きな特徴。スケーラブルなサービスとして提供されているため、ニーズに応じたさまざまな規模で利用を開始することができる

高度な AI 技術が身近なものに

機械学習や深層学習といったAIテクノロジは、近年もっとも注目度の高いトピックの1つであり、先進的なサービスやプロダクトにとってはすでに不可欠な存在となりつつあります。本来であれば、機械学習や深層学習を利用したサービスを自前で実装するには高度な知識と膨大な計算量、そしてそれをまかなう豊富なハードウェアリソースが必要です。Amazon AIの特筆すべき点は、そういった前提条件を満たさなくても、APIやSDKを通じて手軽に利用できるということです。とくに、従量課金制であるため初期投資が必要ないこと、低額であること、他のさまざまなサービスと連携させられることなどが強みとして挙げられます。

このような高度なAIテクノロジを身近なものにするサービスを提供しているのはAWSだけではありません。GoogleやMicrosoftなどの大手ベンダーも、自社のクラウドサービスにおいて機械学習・深層学習の導入を進めており、APIなどの手段を通じて第三者が手軽に利用できる環境を整えています。これらはすでに高度な知識を持つ一部の研究者やエンジニアのためだけのものではなく、一般の開発者やユーザが手元で簡単に利用できる存在になっているわけです。

AIテクノロジは日常的なアプリケーションに大きな価値や新しい体験を提供できるものとして期待されています。現在はその具体的な活用方法を探って各社がしのぎを削っている段階であり、この競争は今後も加速していくものと思われます。SD

Amazon AI
https://aws.amazon.com/jp/amazon-ai/

DIGITAL GADGET

vol.220

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

デジタルガジェットと触覚

研ぎすまされていく触覚

タッチパネルのデジタルデバイスが普及し、触って操作するものが多くなってきました。小さな頃からタッチパネルに慣れ親しんだ子供達は、テレビ画面を触って動かないことを不思議に思うほどです。

単に「触覚」と聞くと、指先で触ったときの手触りの感覚のことを想起しますが、実際には幅広い感覚で、服の着心地や床の滑り具合、物を手で持ったときの感覚など、日常生活のさまざまなところで触覚に囲まれて過ごしています。はやりの最新ゲーミングデバイスやVRの世界においては、触ったときの感覚だけではなく、振動によって仮想的な物の存在を感じたり、音や音楽をより強調して感じるための振動デバイスなども、広義の「触覚」として扱われます。

タッチパネルそのものはツルツルなため、タッチしたり、キーボード入力した際に「トクッ」という振動で、触ったときの感触が戻ってくるよう配慮されている最新スマートフォンも、触覚を振動で代替し、感覚を強調しているものの1つでしょう。

触覚は、ふわふわ、さらさら、ごわごわ、じゃりじゃり、かさかさ、ぷちぷち、ちくちく、ぬめぬめ、べたべた、もこもこ、など、擬音語や擬態語でしか表現できず、実物がないと人に伝えづらく、共有の難しい感覚の1つです。さらに実物があったとしても、感覚としての触覚は人それぞれ感じ方が異なり、何が心地良いのか、何が不快なのか、感覚の研ぎすまされ方も異なります。

触覚は、人間の感覚の中で、視覚や聴覚に比べて感覚全体を占める割合は少なく、視覚が人間の感覚の半分以上を占めるのに対して、触覚は全体の1割以下です。嗅覚も全体の1割以下ですが、人間の原始的な感覚のため、香りと共に記憶がよみがえったりすることもあります。触覚は記憶との連携はそれほどなく、一度触ったことのある物を比較することなしに、記憶だけで同じものかどうかを判断するのは難しいほど繊細な感覚です。

視覚や聴覚の占める割合が大きいエンターテインメントである映画も、4Dシアターと呼ばれるMediaMation社のMX4DやCJ4DPLEX社の4DXでは、席での振動などの特殊効果込みで感覚が強化され、座席そのものが振動して動いたり、水しぶきが降りかかったり、テーマパークのライドもののアトラクションなみの体験が享受できるようになってきています。

今回は、4Dシアター映画館とまではいきませんが、触覚を活用した、または振動をうまく活用したデジタルガジェットをいくつか、次ページに写真付きで紹介します。（**次ページ** 1 ～ 8）

※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。

iPhoneの「触る」という感覚を掘り下げると……

iPhoneやApple Watchに搭載されている振動素子Tapticエンジンのおかげで、素早い立ち上がりのさまざまな通知、操作のときの表現が可能になりました。もともと画面を押しても凹んだボタンの絵しか出せなかったデジタル表示装置に、現実世界に似た触覚をもたらしてくれるのが振動デバイスの働きです。

iPhoneの3D Touchの機能によって、Webブラウザ Safariではリンク先のコンテンツを画面を押し込んでプレビューできます。その際、軽く押し込む「Peek」のときは10ミリ秒の振動、深く押し込む「Pop」のときは15ミリ秒の振動が指先に伝わってくるおかげで、押し込んだ深さを認識することができます。

また、設定画面などのトグルボタン

Apple WatchのHaptics設定

のON/OFFを切り替えると「コンッ」といった感覚、アラームの時刻設定などのピッカー（ドラム式メニュー）を回す際に「ココココッ」といった実際に回転する物体を回しているかのような感覚が得られます。さらにピンチやズーム、回転、スライダー操作の際に限界値まで到達したら、振動で知らせてくれます。

これらのTapticエンジンの活用はApple標準アプリだけでなく、アプリの開発者にも開放されていますが、慎重に扱うことが推奨されています。触覚関連で注意すべき要素としては次の事柄です。

- ユーザの操作に対応した触覚フィードバックを提供する。操作しないのに振動させない
- 一連の触覚を違う目的に使わないこと。強調させるための振動を他の通知のために使うのはダメ
- 操作の際に得られる視覚の体験と触覚の体験を微調整し、タイミングを合わせる
- 触覚だけに頼らないこと。振動モードをOFFにしている人や、再現できないデバイスもある
- 目で確認すべき要素が操作中の指などに隠れてしまう場合は、触覚で代替するのは良い方法
- 触覚デバイスが実際に動き出せるようになるまで若干のタイムラグがあるので、その点に注意
- 触覚に合わせる音を注意深く同期させる。タイミングはもちろん音色も

スマートフォン同士、スマートウォッチ同士など、デジタルデバイスに触ったときの振動を伝えて、離れた人とコミュニケーションをとることもできるようになり、私たちがあまり意識しないところで、触覚のデザインは次々に追加されているのです。

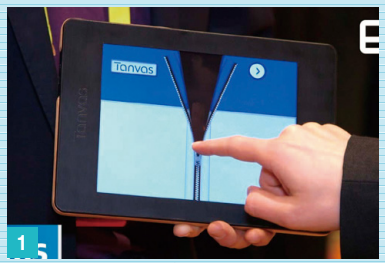
これからの触覚体験

スマートウォッチの腕に伝わる振動やスマートフォンのマナーモードのように、音の補足としての触覚、音がないときの代替としての触覚などにも使われます。触覚という要素は、目の不自由な人、聴覚が不自由な人の代替感覚としても使われます。6つの点で表現された点字による表記や点字ディスプレイによって文字を読みとったり、耳の聞こえないダンサーが触覚や振動によってリズムを認識し、曲に合わせて踊るといったことです。

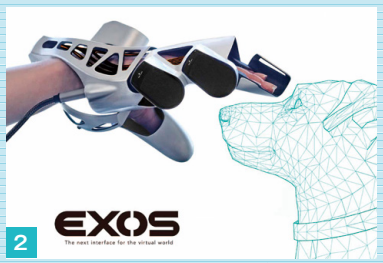
色には色見本が、音には音のサンプルがありますが、広く一般的に使われる触覚見本はないようです。壁紙の質感を確認するための見本などはありますが、特定用途に限定されたものです。また、音や画像はさまざまな二次利用、加工されて新たなものが作り出されていきますが、触覚の加工、コピーされて二次利用、のようなことはまだまだ面倒であることは否めません。

スマートウォッチの裏から腕に伝わるトントンという感覚なども、綿密に考えられた振動のデザインですが、これらを細かく編集したり、調整したり突き詰めて考えるにはまだまだ道具やノウハウが足りていません。

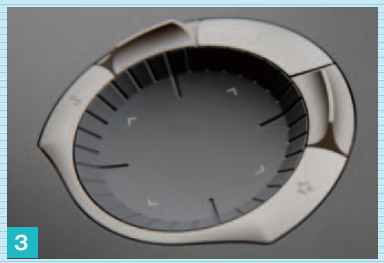
音色のように、触覚にも、初めて触ったときの要素、指で触り続けたときの要素、指を離したときに感じる余韻、振動によって模倣された触覚の代替の度合いなど、音色をデジタル的に変化させるエフェクタやシンセサイザのように、触覚に対しても細かな



1 **Tanvas**:静電気を活用してタッチパネル表面と指との摩擦を生じさせる技術
https://tanvas.co/



2 **EXOS**:振動ではなく外骨格型の触覚提示デバイス（写真は開発中のもの）
http://exiii.jp/exos/


3 **Hyundai Driver Information System**:触覚を活用した車載コントローラ
https://www.designaffairs.com/hyundai-driver-information-system/

5 **NOMIS**:触覚を活用したデジタル楽器
http://jonathansparks.com/nomis/

6 **Good Night Lamp**:タッチ操作するスイッチのデスクランプ
http://goodnightlamp.com

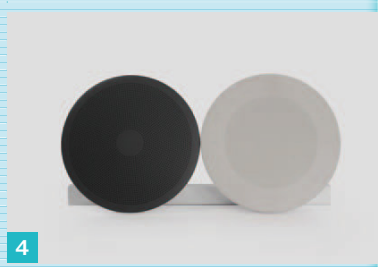
7 **HIBOU**:タッチ操作するラジオ
https://www.prote.in/journal/articles/hibou-radio

デザインが可能な装置、しくみ、ツールが出て来ても不思議ではありません。4Dシアターの座席シートの振動デザインは専門家が何度も修正、調整して演出されているそうです。

視覚を録画再生するには、カメラやビデオといった機器が存在します。絵画なども視覚を再現し、拡張する方法の1つです。音は録音再生でき、音楽には楽譜があり、ほかの人によって解釈され、楽器によって再現が可能です。触覚はほかの感覚に比べ、記録や再現が難しいものです。

触覚に関する研究開発は世界的に進んでおり、指先に微弱な電流を流して感覚をコントロールしたり、超音波を発するデバイスを活用して触覚を模倣したり、ドアノブや机、植物など、従来はデジタルデバイスとして考えていなかったものもデジタル化し、触覚センサー、スマートフォンと連動し、さまざまな試みがなされています。

私たちの身体は、脳と外部とのインターフェースであり、触覚によりモノを扱えるわけです。そう考えるとクリック操作やタッチ操作の、次に重宝されるのは「触覚」なのかもしれません。SD



4 Surface Matters:触覚も重視し、スピーカーそのものがタッチパネル的に操作できるシステム
http://leibal.com/products/surface-matters/

8 D-BOX:ホームシアター用、振動シート
http://www.d-box.com/entertainment/home-entertainment/

Gadget 1

Taclim

https://taclim.cerevo.com/

触感センサー搭載 VRシューズ&グローブ

Cerevo社のTaclim(タクリム)は、手で持つVRグローブと、足に装着するVRシューズがセットになったデバイスです。グローブ部にはそれぞれ2個、シューズ部にもそれぞれ2個、全部で合計8個の振動デバイスが搭載され、VR空間の中での腕の感覚、VR空間で足を踏みしめたときのさまざまな感覚を体感することができます。発売は2017年秋頃、10〜15万円程度を予定とのこと。
(写真は開発中のもの)

Gadget 2

KOR-FX

http://www.korfx.com/

振動バックパック

KOR-FXは、ゲーム内での出来事を体感することのできる、肩からかけるベストタイプのウェアラブルデバイスです。ゲーム内での爆発や銃撃による衝撃などを体感することができます。KOR-FXは体の上半身に取り付けるベストタイプのデバイスで、装着すると胸の左右別々に振動が伝わってくるものです。単三乾電池8本または、充電式バッテリーで30時間から40時間動作します。99ドルで販売中。日本での販売は未定です。

Gadget 3

SubPac

http://subpac.com/

音体感バックパック

Subpacはゲーム用の音の振動を体感するためのデバイスです。バックパックのように背中にしょって使うタイプのSubPac M2と、椅子に設置して使うタイプのSubPac S2があります。ヘッドフォンと併用し、Subpacは重低音を感じるために用います。SubPac M2の重さは2.3kg、一度の充電で6時間持つそうです。SubPac M2は先行割引価格299ドルで発売中。日本でも取り扱われる予定だそうです。

Gadget 4

HUMU

https://flexound.com/

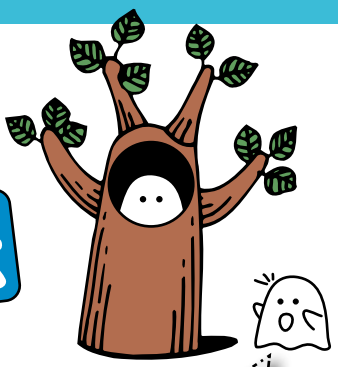
振動枕

Flexound社のHUMUは音を頭で感じることのできる、触覚を重視したスピーカーです。枕のように頭の下に置いて使ったり、抱きかかえて使ったりします。一度の充電で8時間の利用が可能。遠隔地同士の、テレビ会議、電話の通話のスピーカーとして利用するのも音の存在感をより感じることができて良いそうです。クラウドファンディングのKickstarterでは299ドルから。未発売。

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

47

結城浩の再発見の発想法



アイコン

アイコンとは

アイコン(icon)とは、**特定の個人やサービス、アプリや機能などを表す小さな画像**のことです(**図1**)。アイコンは、クリックやタップして詳しい情報を得たり、アプリや機能を呼び出したりするボタンの役目を担うことも多いでしょう。……と、いちおう解説風に書いてみました。でも、アイコンはコンピュータやスマートフォンなどで日常的に使っているものですから、わざわざこんなふうにアイコンを解説するのもぎこちないですね。

アイコンという言葉は古代ギリシャ語のエイコーン(イコン)から来ています。もともとは「神や仏の姿を表現した宗教的な画像」を意味していました。コンピュータ用語としてのアイコンは1980年代から使われています。

SNSとアイコン

TwitterやFacebookなどのSNSでは、**個人を識別**するためにアイコンが使われます。自分の写真を小さく切り取った画像や自分のお気に入りのイラストを、自分自身の代わりに使うのです。

人と直接会うときには顔や服装などが識別の手がかりになりますが、ネットの上ではその役目をアイコンが担うことになります。「この人の名前は知らないけれど、アイコンはよく覚えている」という状況はよくありますし、その人がアイコンを変えたとたん「この人、誰だっけ？」と混乱することもありますね。SNSでアイコンを変えるということは、自分の「顔」を変えるようなものですから、混乱を引き起こすのは当然と言えるでしょう。

毎日のようにSNSで眺めているアイコンには次第に親しみが湧いてきます。ネット上でレコメンデーションをするサービスでは「あなたの知っているこの方々も推薦しています」を表現するために、知人のアイコンをずらっと並べることがあります。ただの広告があるよりも、自分の知っている「顔」としてのアイコンが周りにあるほうが、確かに商品に親しみが湧きそうです。

あるときTwitterで、パクリbotが筆者のアイコンを勝手に使っていたことがありました。すぐに運営に報告しましたが、あれは自分のアイデンティティが侵されたような気持ちになるものですね。アイコンはまさに「顔」と言えます。

ネットでは「本人が誰かはわからなくていいから、発言Aと発言Bが同一人物かどうかは知りたい」という場合があります。そんなときには機械的に生成される**アイデンティコン**

▼図1　アイコン

(Identicon)が便利です。アイデンティコンとは発言者のIPアドレスなどをもとにしてハッシュ値を作り、そこから自動的に作ったグラフィックパターンをアイコンにしたもののことです(**図2**)。

アイコンとピクトグラム

アプリ中の**機能を表現する**ときもアイコンが使われることがあります。多くの人に共通に認識されるアイコンは**ピクトグラム**(pictogram)と呼ばれることも多いですね。

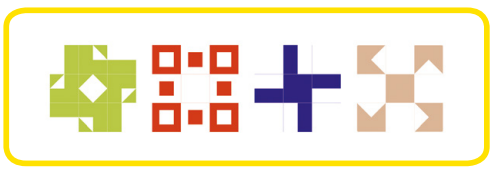
アプリの機能をどのようにして画像として表現するかは、難しい仕事だと思います。たとえば、メディアプレーヤーで、プレイは右向き三角形(▶)、ストップは正方形(■)、早送りは右向き三角形2つ(▶▶)、早戻しは左向き三角形2つ(◀◀)……というのは単純だけれどわかりやすいアイコンだと思います。

機能の中には抽象的なものも多いです。たとえば、アンドゥはぐるっと回って左向きの矢印( )、リドゥはぐるっと回って右向きの矢印( )などは、なるほどと思います。

抽象的なものを具体的な事物で表現することもよくありますが、それはときどき時代遅れになる場合もあります。たとえば、アプリの中で「保存」を表すアイコンは以前はフロッピーディスク( )が使われていましたが、最近は下矢印(⬇)で表すことが多いようです。今どきフロッピーディスクを使っている人はほとんどいませんしね。

先日、ある人が「iPhoneの電話アイコンが何を表しているかわからない」という話をしていて驚きました。確かに、iPhoneの電話アイコン(**図1**参照)は、ひと昔前の電話機の形ですね。

▼図2　アイデンティコンの例



具体的なものをアイコンとして使うのに無理を感じたことがあります。それはある画像編集ソフトの機能パレットです。そこでは、描画する機能として「鉛筆」「ペン」「ブラシ」が区別されていました。でも、それを表現するのに、みな似たような細い棒として描かれているのです。具体的なものはぜんぶ人間が手に持つ細い棒ですから、それをアイコンに流用するのは無理があるのじゃないでしょうか。

日常生活とアイコン

アイコンは私たちの生活でもときどき顔を出します。**交通標識**は日常生活で使われているアイコン(ピクトグラム)ですね。

営業職の人の中には、**顔写真を入れた名刺**を使う人がいます。あれはアイコンの一種と言えます。個人を覚えてもらうために、名前だけでなく「顔」も使うのは合理的ですね。

たとえば、複数人がチームを組んで活動するとき、その**チームのシンボルマーク**を作り、アイコンとして使うと便利じゃないでしょうか。チームとして使用するものや、チームの成果物にはそのアイコンを付けておくのです。アイコンを使えば、チームという抽象的なものに「顔」を与えることができ、アイデンティティを強化するのに役立つでしょう。

家庭内でアイコンを使う手もありそうです。**家族1人1人専用のマーク**をアイコンとして用意し、個人の持ち物を管理しやすくするのです。

アイコンという小さな画像は、個人の識別を容易にし、抽象的な概念に形を与え、アイデンティティを強化してくれます。アイコンは、なかなかあなどれないツールと言えるでしょう。

あなたの周りを見回して、明確に識別を必要としているものはありませんか。あるいはまた、ひとまとまりのものとして認識させたいことはありませんか。抽象的だけれど、具体的な「顔」が必要なものはないでしょうか。そこにアイコンを使うことはできないでしょうか。

ぜひ、考えてみてください。SD

増井ラボノート

コロンブス日和

最終回　発見プログラミング

「コロンブスの卵」的なシステムをいろいろ紹介してきたこの連載も今回が最終回になりました。コロンブスといえば「発見」ということで、最終回の今回はプログラミングと発見にまつわる話をします。

Author　増井 俊之（ますい としゆき）　慶應義塾大学

発想と発見

新しい発想を得るためには人間が時間をかけてやり方を考えたり試行錯誤したりする必要があります。前回紹介した「Scrapbox」は、少しでも関連のある情報を柔軟に思い出しやすくすることによって、発想活動を支援します。もし発想活動の一部でも計算機にやらせることができればアイデアを出す効率がアップするかもしれません。

プログラミングと発見

失敗から何かを発見したというエピソードはたくさんあります。もちろんただ失敗すれば良いというものではなく、たくさんの試行錯誤をした結果、普通と異なる挙動を見いだしたことが発見につながったということでしょう。実はプログラムを動かすことによっても、さまざまな発見ができます。こういう作業を発見的プログラミングと呼ぶことにします。

今回は、前回紹介したScrapboxの上に書いたJavaScriptコードをブラウザで実行してみたいと思います。Scrapboxには**コード記法**というものがあり、プログラムなどのテキストデータをブロック表示してファイルとして外部からURLでアクセスできるので、これを使えばScrapbox上に書いたプログラムをブラウザ上で実行できます。CanvasやSVGを利用するとJavaScriptでブラウザ上に絵を描くことはできますが、初期設定などが必要で面倒なところもあるので、描画などを極端に簡単にしたP5.jsというシステムを利用することにします。

数え上げによる答えの発見

算数の問題を解くときは、理屈を使って計算するのが普通です。たとえば「鶴と亀があわせて30匹いて、足の数は全部で100本でした。鶴と亀は何匹いたでしょう？」のような「つるかめ算」を解く場合、「全部が鶴だとすると足の数は60本で、全部が亀だとすると120本になるから……」のように算数的に解くやり方もありますし、

$$t+k=30$$
$$2t+4k=100$$

のような1次方程式を使うこともできます。数学の問題はこのように解析的に解くのが普通ですが、**図1**のような総当たりプログラムを書けば、数学的知識がなくても力技で答を得ることができます。

つるかめ算を解くのにこういうプログラムを書く人は少ないと思いますが、このような方法

▼**図1**　Scrapboxでつるかめ算を解く（tsurukame.js）

tsurukame.js

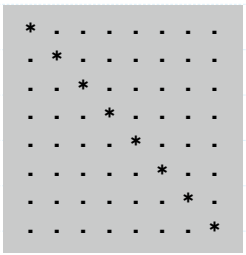
```
for(t=0;t<=30;t++){
  for(k=0;k<=30;k++){
    if(t+k==30&&t*2+k*4==100){
      alert(`鶴${t}匹, 亀${k}匹`)
    }
  }
}
```

のほうが有利な場合もあります。もし問題が「足の数が100本以上でした」のようなものであった場合、1次方程式で普通に解くことはできませんが、上のような方法ならすべての解を網羅できてしまいます。つまり、論理的に解けなくても、簡単なプログラムを書くことによって発見的に解が得られてしまうことになります。

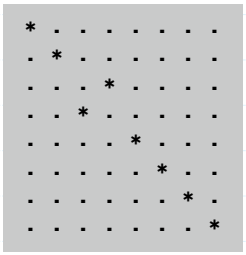
乱数で答えを発見

つるかめ算の場合は解の候補をすべて数え上げれば答えを発見できるはずですが、総当たりで発見が難しい場合は、乱数を使って試行錯誤的に答えを捜すこともできます。たとえば8-Queen問題を解きたいときは、再帰的に解を探索するアルゴリズムを使うのが常識的ですが、ランダムに行や列を入れ換えることを繰り返すだけでも意外と簡単に解を得ることができます。

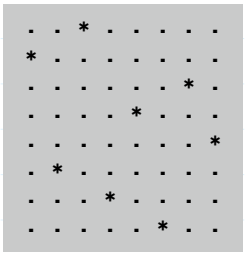
まずこのようなデータを用意します。



2つの行をランダムに交換してみると、次のように少しだけ良い結果が得られることがあります。



これを何度も繰り返してよさげなものを残していくと、最終的に次のような答えを得ることができます。



人間は数え上げ操作が苦手ですから、8-Queenのような問題に対しては試行錯誤的に取り組んで答えを見つけることが多いですが、試行錯誤を計算機に実行させることによって、解を発見できる場合も多いでしょう。

乱数の性質の発見

乱数を使って問題を解く場合、JavaScriptのMath.random()のような「一様乱数」関数がよく利用されます。これを呼ぶと、0から1までのランダムな実数が返ります。たとえばrandom()が0.1以下の値を返す確率は1/10です。

ここまでは普通の話ですが、**図2**のようにrandom()で生成した座標を使って点を表示してみると**図3**のようになります。一様な乱数というと**図4**のようなイメージを連想してしまいますが、本当の乱数はこうならないことが発見できるのが面白いところです。

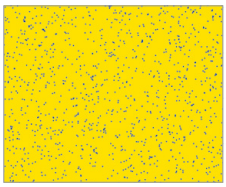
乱数はシミュレーションやゲームでよく利用されるものですので、乱数とはどのようなもの

▼**図2** 乱数で生成した座標に点を打つ(randam.js)

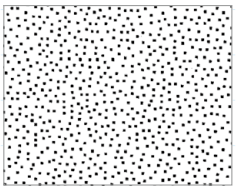
random.js

```
function setup(){
  createCanvas(400,400)
  fill('yellow')
  strokeWeight(0)
  rect(0,0,400,400)
}
function draw(){
  fill('blue')
  rect(random(400),random(400),2,2)
}
```

▼図3　生成された点



▼図4　一様な乱数



かについては、エンジニアであれば誰でもなんとなくイメージを持っていると思いますが、乱数をたくさん生成して視覚化してみるとさまざまな発見があったりします。

乱数の足し算

いくつかの乱数の平均をとると何が起こるでしょうか？　ランダムなものをいくら集めてもランダムに決まっているだろうと思うかもしれませんが、実は一様な乱数をたくさん足したものは一様乱数にならず、正規分布に近づくことが**中央極限定理**として知られています。2つのサイコロをふった場合、その数の和が2や12になる確率よりも7になる確率のほうが大きいだろうということは直感的にわかりますが、一様乱数を足した場合も同じだということはよく考えないとわからないかもしれません。そういう場合、何も考えなくてもプログラムで実行してみればそのような性質はすぐ発見できます(**図5**、**6**)。

▼図5　乱数を何回か足した結果のヒストグラムを計算(normal.js)

```
normal.js
  function setup(){
    createCanvas(400,400)
    histogram = []
    for(i=0;i<100;i++) histogram[i] = 0
    strokeWeight(0)
  }
  function draw(){
    val = 0
    for(i=0;i<10;i++) val += random(1/10)
    histogram[Math.floor(val * 100)] += 1
    fill('yellow')
    rect(0,0,400,400)
    fill('blue')
    for(i=0;i<100;i++)
      rect(i*4,400-histogram[i],4,histogram[i])
  }
```

▼図6　計算結果

乱数の掛け算

乱数をたくさん足すと正規分布になるわけですが、では乱数をたくさん掛けるとどうなるでしょうか？　実は乱数を何度も掛け算したものは「冪分布(べきぶんぷ)」になります。**図7**のプログラムを実行すると**図8**のような冪分布になります。リニアスケールで青い点を、対数スケールで赤い点を表示しています。

世の中には「ロングテール」と表現される冪分布がたくさんありますが、どうやらランダムな事象が掛け算的に繰り返されるとロングテール的な現象が発生するようです。

勝負の発見

まったく五分五分の勝負を繰り返した場合で

▼図7　乱数の掛け算のプログラム例(beki.js)

```
beki.js
  function setup(){
    values = []
    createCanvas(400,400)
    strokeWeight(0)
  }
  function draw(){
    value = 1.0
    for(i=0;i<10;i++){ value *= random() }
    values.push(value)
    values = values.sort((a,b) => b-a)
    fill('#ddd'); rect(0,0,400,400)
    for(i=0;i<values.length;i++){
      fill('#00f'); rect(i,400-values[i]*10000,2,2)
      fill('#f00'); rect(50*Math.log(i),
                  300-10*Math.log(values[i]),2,2)
    }
  }
```

も意外な片寄りを発見できます。

2人に配ったトランプの札を1枚ずつ出して勝負する「戦争ゲーム」というものがあります。このゲームは単純であまり面白いものではありませんが、そもそも勝負はつくものなのでしょうか？　Scrapbox上で勝負するプログラムを**図9**のように書くことができます。

P5.jsのdraw()がメインプロセスになって勝負の結果を表示しています。2人のプレーヤはplayer()というgeneratorで表現されており、2つのプレーヤプロセスがsetup()で初期化されています。player()もメインプロセスも無限ループになっていますが、yieldでメインプロセスと通信することにより同期しながら対戦が行われます。generator機能を利用すると、このようなシミュレータを柔軟に記述できるので便利です。

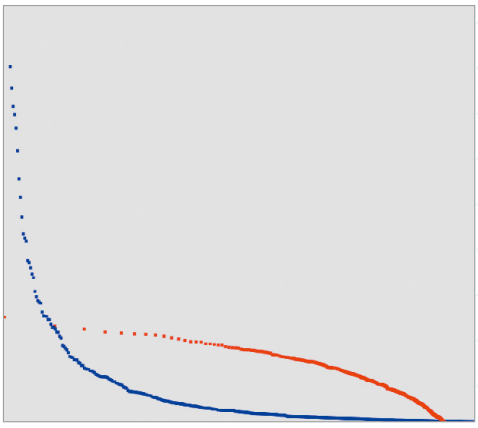
このプログラムを実行すると、意外にもすぐに決着がついてしまうことがわかります(**図10**)。最初の手札は同じですし、ランダムにしているにもかかわらずあっという間に総取り状態になってしまうということは感慨深いものがあります。条件がまったく同じであっても大きく勝ったり負けたりすることが発見できたことになりますが、ほんの少しのきっかけで運命が変わってしまうのは恐いものだと思います。

終わりに

今回紹介したプログラムはすべてScrapbox上に置いてあり(http://scrapbox.io/prog-exercises/)、クリックするだけでブラウザ上で実行できます。Scrapboxは情報整理にたいへん便利なシステムですが、そのうえでプログラムを気軽に作成してさまざまな発見ができることでさらに広い応用が考えられると思います。

18回に渡ってコロンブスの卵的な実用的なシステムをいろいろ紹介してきました。私はこれらを毎日活用していますが、読者の皆さんもぜひ試してみていただければと思います。SD

▼**図8**　実行結果



▼**図10**　勝負の途中

```
player1: 12 13 5 10 11 9 4 4 6 9 1 8 2 8
player2: 7 11 6 7 2 1 5 13 10 3
```

▼**図9**　戦争ゲームのプログラム(sensou.js)

sensou.js

```
var cards1, cards2, table = []
function* player(cards){ // プレーヤプロセス
  while(true){
    var result = yield cards.shift() // 手を出す
    yield                            // 結果を待つ
    result.forEach((v) => cards.push(v))
  }
}
function setup(){
  cards1 = new Array(13) // 1〜13のカードを用意
  for(var i=0;i<13;i++) cards1[i] = i+1
  cards2 = cards1.concat() // 配列をコピー
  shuffle(cards1,true)
  shuffle(cards2,true)
  p1 = player(cards1)
  p2 = player(cards2)
  createCanvas(400,400)
  frameRate(10) // 毎秒10回勝負する
}
function draw(){
  if(cards1.length == 0 || cards2.length == 0) return
  var hand1 = p1.next().value     // プレーヤの手を取得
  var hand2 = p2.next().value
  table.push(hand1); table.push(hand2)
  shuffle(table,true)             // 場の札をシャッフル
  v = hand1 == hand2 ? 0 :        // 勝敗判定
      hand1 == 1 && hand2 == 2 ? 2 :
      hand1 == 2 && hand2 == 1 ? 1 :
      hand1 > hand2 ? 1 : 2
  switch(v){
    case 0: p1.next([]); p2.next([]); break;
    case 1: p1.next(table); p2.next([]); table = []; break;
    case 2: p1.next([]); p2.next(table); table = []; break;
  }
  clear()
  text('player1: '+cards1.join(' '),10,10)
  text('player2: '+cards2.join(' '),10,30)
}
```

第14回 大阪は裏なんばが熱いし、OSC大阪も熱かった

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

大盛況のOSC大阪で2017年はスタート

2017年最初のオープンソースカンファレンスは大阪からスタートです。大阪での開催は、これまで企業向けの内容に絞った「OSC.Enterprise」という名前での開催でした。企業向けなので、セミナーや展示も協賛企業中心ですし、平日開催ということで、参加者も200名ほどのややこぢんまりとした開催になっていました。

「大阪をもっと盛り上げたい！」ということで、今回は土曜日開催、会場も広い展示スペースとたくさんのセミナー会場を用意して、いろいろなOSSコミュニティのみなさんにも積極的に出展してもらえるように変更しました。

これだけの規模になると、搬入する荷物や事前準備も必要となるので、前日準備の時間を取るようにしています。一方、ただ準備だけしていても時間がもったいないので、大きめのセミナー会場を1つだけ用意して、お仕事的にも興味を持ってもらえそうなテーマのセミナーを開催している裏で、前日準備を行うようにしています。これを「1.5日開催」と呼んでいます。今のところ、北海道、名古屋、大阪、福岡といった中規模な開催はすべてこの1.5日開催のスタイルをとっています。

今回は、たとえば筆者はDevOpsについてお話をしたこともあり、金曜日であるにもかかわらず、それぞれのセミナーに常時50名近く、1日全体で見るとユニークで100名弱の方が参加してくれました（**写真1**）。

さらに翌日の本番は400名の来場者がありましたので、合計でのべ500名近くの方にお越しいただきました。想定以上の方が来てくれたのは、企画者冥利に尽きるというものです（**写真2**）。次回はさらにたくさんの方に楽しんでもらえるように頑張りたいですね。

大阪は今、裏なんばが熱い!?

「食い倒れの街」大阪ですが、グリコの看板で有名な道頓堀のあたりは、観光客で溢（あふ）れかえっています。そこで今、飲み屋街として注目されているのが「裏なんば」と呼ばれるあたりと紹介されて、ここ最近は大阪出張のたびに開拓に勤（いそ）しんでいます。

場所としては、千日前から日本橋のあたりなのですが、行き方としては吉本興業のお笑いで有名な「なんばグランド花月」から千日前の道具屋筋（調理器具を扱っているお店がたくさん集まっている、東京でいえばかっぱ橋みたいなところ）を少し入っていったあたりです。大小取り混ぜたいろいろな飲み屋さんがたくさん並んでいて、多くのお客さんで賑わっています。

筆者が好きなのは和風の立ち飲み屋さんなので、自然と日本酒中心になります。最近、毎回お伺いしているのが「地酒立呑み『最』」という立ち飲みのお店で、日本酒好きの人たちが集まっています。大きな業務用冷蔵庫に入れられた日本酒（お米のジュースと呼ばれている）から、呑みきりサイズのグラ

▼写真1　1日目終了後、準備スタッフで食事。今回も香川大学から学生スタッフが3名参加してくれました

▼写真2　出展ブースになぜかPC-6001（愛称パピコン）が！　筆者が一番最初に買ったコンピュータです

▼写真3　裏なんば「地酒立呑み『最』」の店内。こんなふうに日本酒がズラッと並んでいます

▶写真4　懇親会の様子。次のコミュニティ活動のための英気を養っています

スでいろいろな種類のお酒が楽しめるという、筆者にとっては最高のお店です(**写真3**)。

今回は前夜祭ということで、木曜日の夜に出撃しました。たまたま隣になった常連さんに、お勧めのお酒を聞いたりして、楽しく過ごしました。こういう楽しみがあるから、全国各地のOSC開催に廻れるんだよな〜、とモチベーションアップにつながっているのを感じますね。

終了後の懇親会も大盛り上がり

土曜日の開催になったおかげで、OSC終了後の懇親会は70名以上が参加し、大盛り上がりでした(椅子が足りないハメに)。会場は「ドルフィンズ」というベルギービールのお店で、昔からなぜかオープンソース界隈では人気のお店です(**写真4**)。普通のビールが飲み放題だったのですが、ベルギービールが通常の半額ということで、カウンターには長蛇の列が。私もベルギービールの代表格「ヒューガルデンホワイト」を注文してみました。それもダブルで。ダブルにすると、巨大なグラスで出てきます。どれぐらい巨大なのかわかるように、今回出展スタッフとして参加して、懇親会にも来てくれていた「IoT女子[注1]」の菅野あすかさんに持ってもらいました(**写真5**)。だいたい600mlぐらい入っていて、ゆっくりと楽しめます。

▼写真5
「IoT女子」の菅野あすかさん。顔が小さいので、なおさらグラスが大きく見えます

さらに有名なフルーツビール(ビールに果物を漬け込んで造る)の「ベルビュークリーク」も呑んだりして、ベルギービールを堪能して大盛り上がりの懇親会でした。多分、次回もこの店になると思いますが、椅子が足りないのをどうするか、ですね。SD

注1)　http://iot-jyoshi.com/

Report

京都でお香作りを楽しむ

突然ですが、筆者はお香が好きです。去年のOSC京都終了後、京都のお香の老舗「松栄堂」本店で買い物をしたところ、京都・嵐山店で使えるお香作り体験の無料券が当たりました。そこでOSC大阪終了後、嵐山に立ち寄ってお香作りを体験してみました。お香にはさまざまな種類があり、それぞれ香りもかなり違います。タブレット状に固めてある7種類あるお香から10個選んで、乳鉢ですり潰します。粉末状になったお香を不織布の袋に入れ、和紙で包むとオリジナル匂い袋の完成です。

女子力高めの趣味に見えるかもしれませんが、夜に自分の好きなお香を焚くとリラックスできます。アロマに比べて、線香に火を点けるだけで簡単なのも魅力です。機会があれば、ぜひ試してみてください。

左上のカラフルなお香を10個選んで自分の香りを作ります。理科の実験みたいですね

できあがった匂い袋。大きいほうは箪笥(たんす)に入れて、衣類に香りが移るのを楽しんでいます

ツボイのなんでもネットにつなげちまえ道場

第22回

「さくらのIoT Platform β」の通信モジュールを使ってみる(後編)

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

前回では

前回は、さくらの通信モジュール(LTE)β版や、それがサポートしているLTE カテゴリ1について簡単に説明し、Arduino Unoで動かしてみました。

このとき、連載でいつも使っているmbedではなくArduinoで使ってみたのは、筆者がmbedで試したときには、サンプルプログラムをうまく動かすことができなかったためでした。その後、通信モジュールの消費電力を測るために動かしていて気づいたのですが、どうもオンラインにならない(通信モジュールがセルラーネットワークに接続できない)という問題が発生することがあるようです。

残念なことに、さくらの通信モジュール(LTE)β版は接続ステータスをあまり詳細には返してくれません。このため、どういった理由で通信モジュールがセルラーネットワークに接続できないのか把握するのが難しいです。接続できないと、原因がわからないまま、ただ接続状態になるまで待つしかできなくなり、開発や確認作業が進まなくなるのがストレスです。

このような現象は筆者だけで起きている特有の現象だと思い、mbedでうまく動かせなかったと前回記したのですが、その後ほかのユーザからも同様に接続できなくなることがある現象を聞いたため、このサービスかモジュールに一般的に起こり得る現象だということがわかりました。これもβ版ということで、しかたがないのでしょうか。

消費電力

さて、前回記したように、LTE カテゴリ1に期待するのは省電力とモデムのコストです。確かにさくらの通信モジュール(LTE)β版は、税込10,756円と、一緒に使うArduinoシールドボード-β版 -(税込8,640円)と合わせても、従来のLTE通信モジュールよりも安価です。気軽に使うことができるほど安価でないのは、量産効果(生産量が多くなることで、製造原価が下がること)の都合も大きいですが、やはり前編(2017年3月号)の**表1**に記したように、モデムの複雑さが十分に低くないことが原因と言えそうです。

さて、さくらの通信モジュール(LTE)β版の消費電力はどの程度なのでしょうか。前編でArduino Unoでこの通信モジュールを使ったとき、電力はパソコンのUSBからArduino Unoを通じて供給していました。一般的に、セルラー通信モジュールはマイコンと比較して電力を大きく使いますので、規格で500mAと定められているUSB 2.0からではなく、ACアダプタなどほかの方法で給電します。前編での実験の際にも、Arduino UnoにはACアダプタから給電したほうがよいのではないかと思い、筆者は少し調べてみていました。

今のところ、さくらの通信モジュール(LTE)β版の入力電流はRRC_ACTIVE Uplink Po = +23[dBm]のとき最大で800mAと記されており、これが唯一の手がかりです。この「RRC_ACTIVE Uplink Po = +23[dBm]」というのは、基地局に対して+23dBmという強さで送信をし

ているときの電流を示しているようです。セルラーは、基地局との距離や障害物などが一定ではないので、常に最大出力でなく、必要な強さで電波を送るようなしくみが搭載されています。常に最大出力で送らないのは、バッテリをムダに消費しないためや、電波をムダに出力してほかの通信を阻害する可能性をできるだけ排除するためでしょう。

前置きが長くなってしまいましたが、このモジュールがいったいどれだけ電力を消費しているのか、実験して概要を掴（つか）んでみました。

電流の測り方

ところで、消費電力、機器が消費している電流を測るにはどうすればよいのでしょうか。電流を測るというと、ぱっと思いつくのは電流計やテスター（デジタルマルチメータ、DMM）だと思います。確かにテスターや電流計は比較的手軽に電流を測ることができますが、消費する電流の変化の激しい機器を測定するのには向きません。たとえば、10ミリ秒（0.01秒）だけ500mA消費し、普段は50mA消費している機器を測定するときに、これを針や数値で表示されても読み取ることはできませんよね。

こういったものを測定するときには、ログを取ることのできる高機能なデジタルマルチメータや電源装置を使ったりするのが手っ取り早いのですが、20万円以上することがざらです。そのような高機能な測定器は、ログをCSVファイルとして取得して、USB Flashドライブに書き出したり、LANでパソコンに転送したりといったことも可能だったりします。バッテリ駆動の機器を頻繁に開発しているのであれば、こういった測定器を使って仕事をしたくなりますが、おいそれと買える機器ではありません。

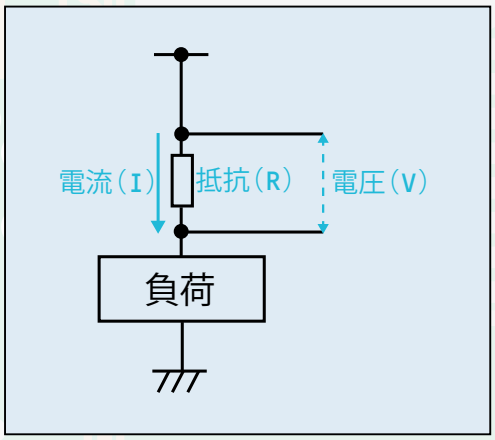
また、オシロスコープで電流を測るための「電流プローブ」というプローブがオプションとしてさまざまなオシロスコープメーカーから発売されています。これらのプローブは、先ほどの計測器より比較的安価なのですが、それでも標準価格は10万円前後です。やはり、気軽に買えるようなものではありません。

オシロスコープで電流を測る方法には、電流プローブを使う方法以外のものもあります。 本連載の第1回（2015年7月号）でも紹介したオームの法則を使えば、電流を知ることができるのです。オームの法則は、

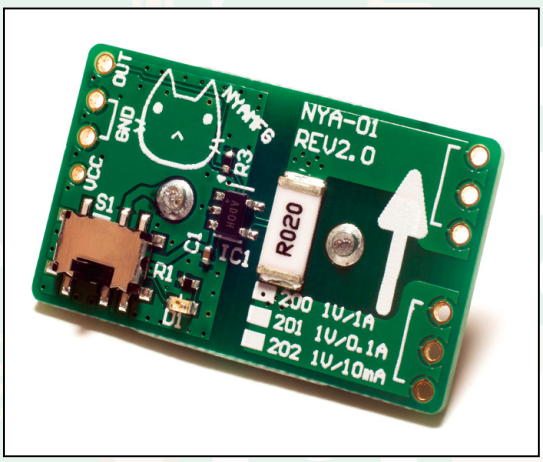
電圧（V）＝電流（I）×抵抗（R）

というものでした。**図1**のように電源と負荷（どれだけ電力を消費するか知りたい対象機器）の間に抵抗を取り付け、抵抗の両端の電圧（電位差）をオシロスコープで測れば、電流を計算できます。

▼図1　電流検出



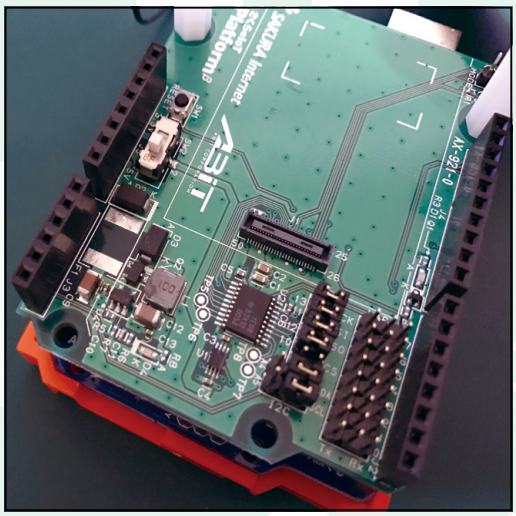
▼写真1　簡易型電流プローブNYA-01



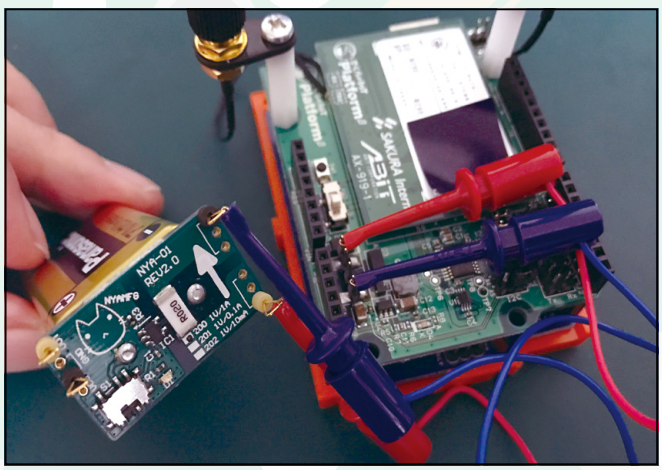
電流検出用の抵抗器は高精度に作られているので、オシロスコープで測った電圧を抵抗値で割るだけで、そこそこ正確な電流を求めることができます。しかし、画面を見ただけでは電流が直感的に読み取れないというのが課題です。こういった問題を解決する商品に「簡易型電流プローブ NYA-01」というものがあります(**写真1**)。

NYA-01には測定対象の電流の大きさに合わせて、いくつかのバリエーションがあります。今回は、このNYA-01のうち、1Aを1Vとして出力するNYA-01-200[注1]を使用しました。NYA-01-200は電流が1mA流れているときに1mVが出力されますので、読み替えはとても簡単です。

▼写真2　ヒューズを取り外したArduinoシールドボード



▼写真3　ArduinoシールドボードにNYA-01を接続したところ



通信モジュールの消費電力

では、さくらの通信モジュール(LTE) β版の消費電力を測ってみましょう。本当はモジュールの直近で電流を測るとよいのですが、今回は手軽に測定するためにArduinoシールドボードの上にある、さくらの通信モジュールに電源を供給するための電源回路の入力についているヒューズを取り外しました(**写真2**)。

取り外したヒューズの両端子にNYA-01の「負荷IN」と「負荷OUT」を接続します(**写真3**)。

こうすることで、NYA-01に搭載されている電流検出用の抵抗が、先ほどの**図1**のように負荷(電源回路と通信モジュール)に接続できます。

そこで、NYA-01のVOUTとGNDにオシロスコープのプローブを接続すると、負荷(Arduinoシールドボードとさくらの通信モジュール(LTE) β版)に流れている電流を簡易的に見ることができます(**写真4**)。

Arduinoシールドボードに搭載されている、さくらの通信モジュール(LTE) β版用の電源は、DC/DCコンバーターというタイプのチップを使って、5Vから3.9Vを作り出しています。DC/DCコンバーターは、おおむね、この連載の第8回で紹介したPWMのように、ON/OFFを繰り返して電圧を変えています。ここで使われているチップの場合、効率80～90%程度で電圧を変えていますので、今回のようにDC/DCコンバーターよりも手前で測定をすると、通信モジュールの正確な消費電力は測定できません。また、Arduinoシールドボードには、Arduinoのマイコンが出す5Vの信号を通信モジュールが使用する1.8Vと変換するためのチップが

注1)　ssci.to/2862。2,160円

搭載されています。このチップも電力を消費するという点に注意が必要です。

こうした多くの問題はあるものの、とりあえず測定をしてみたところ、**写真5、6**のようなデータが取れました。

どうやら、モジュールの起動時には5V 190mA程度の電流が流れていたようです（**写真5**）。

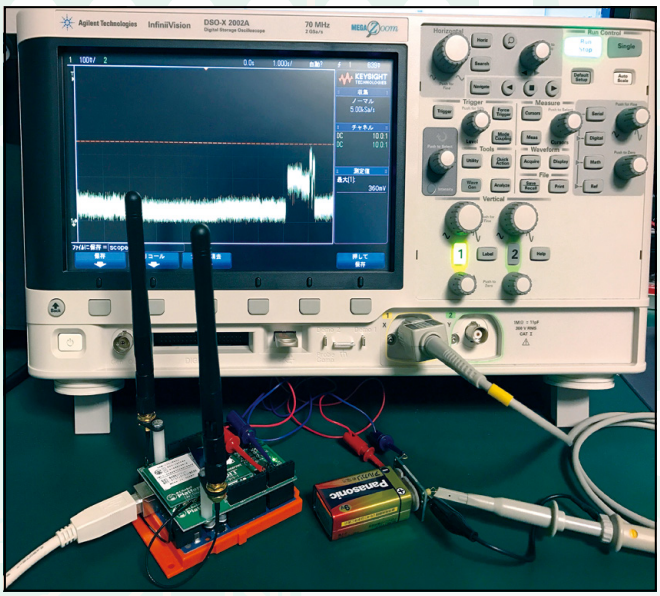
モジュールが起動を終え、セルラーネットワークに接続して通信を始めると、電波を発しているときに、500mA以上が流れているように見受けられました（**写真6**）。

こうして、とてもざっくりした値ですが、実際にモジュールが消費する電力を測ることができました。

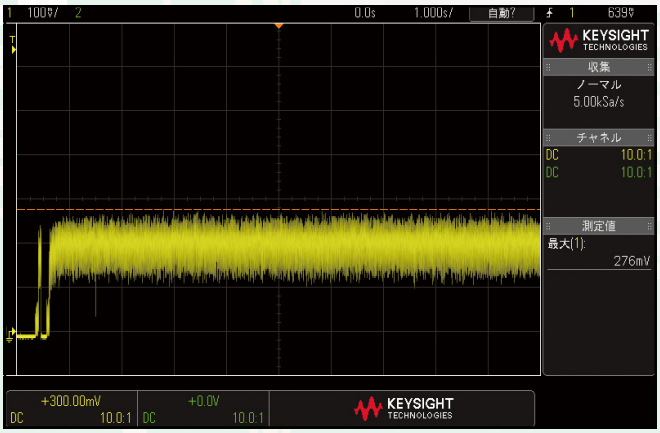
考えてみれば、電波を発しているときには、従来の携帯電話と同程度の強さの電波を発しているので、消費電力が大きいのは当たり前でした。省電力を実現するには、電波を発していない状態のときにどれだけ省電力にするのかが大切でしょう。

そういった点で、eDRX[注2]とモジュールでの省電力機能の搭載が待ち通しいです。現状、パソコンのUSB 2.0からの給電だけでも動かすことはできますが、ACアダプタで給電したほうがよいでしょうし、電池で長時間動かすIoTのエンドノードに気軽に採用できるようになるのはもう少し先になりそうです。SD

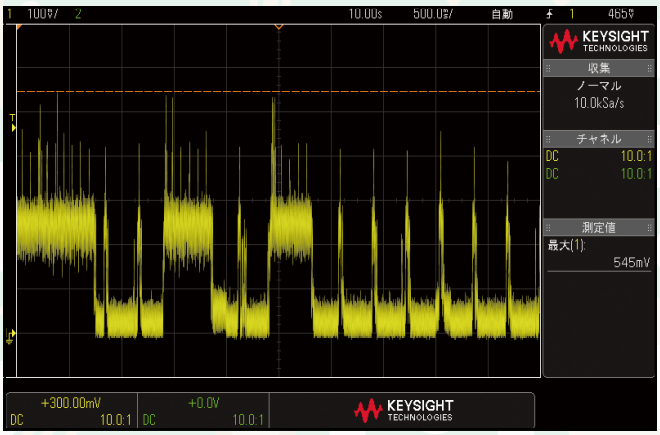
▼写真4　オシロスコープで電流を測定してみた



▼写真5　モジュールを起動したときの消費電力



▼写真6　モジュールが通信しているときの消費電力



注2）Extended Discontinuous Reception。広域対応のIoT向け低消費電力技術の1つ。

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4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 無線LANギガビットルータ「WRC-1750GHBK-E」 1名

IEEE802.11ac規格、1,300Mbps（理論値）に対応したWi-Fiルータです。有線ポートもギガビットに対応しているため、無線と有線の両方で高速光回線サービスの速度を最大限に利用できます。また本機では、自宅Webサーバの公開やネットワークカメラへのアクセスなどが簡単に行えるようになるDNSサービス「SkyLinkDDNS」を無料で利用できます。

提供元 エレコム http://www.elecom.co.jp

02 世界巡業「JF-PEACE4」 2名

USBポートを2ポート、ACコンセントを2口備えた、ACプラグ内蔵のモバイルバッテリー（2,600mAh）。「旅先でコンセントが足りない」「モバイルバッテリーの充電を忘れた」といった問題をこれ1つで解決できます。白・黒どちらかをプレゼント。

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

03 Paragon Hard Disk Manager for Mac 2名

Macで利用しているHDD・SSDに対して、「システム全体をまるごとバックアップ」「パーティション単位のバックアップ」「各パーティションのサイズ変更」「HDD・SSDの複製コピー」などの操作を行えるソフトです。対応OSは、OS X 10.9 Mavericks～macOS 10.12 Sierra。

提供元 パラゴンソフトウェア https://www.paragon-software.com/jp/

04 Linuxカーネル「ソースコード」を読み解く 2名

平田 豊 著

「アプリケーションやライブラリが、どのようにしてLinuxカーネルの機能を活用しているか」に着目して、カーネルのソースコードを読み解く1冊です。対象OSは、Fedora24（Linuxカーネル 4.5）。

提供元 工学社 https://www.kohgakusha.co.jp

05 プログラミング作法 2名

Brian W. Kernighan、Rob Pike 著

「アルゴリズムとデータ構造」「インターフェイス」といった、プログラミングをする際に出会うさまざまな課題について、C/C++/Javaの実例コードを示しながら実践的なアドバイスを与えてくれます。

提供元 アスキードワンゴ http://asciidwango.jp

06 進化する銀行システム 2名

星野 武史 著／花井 志生 監修

24時間365日止まらずに稼動し続けることが求められている銀行システム。本書はその銀行システムで多く採用されているメインフレームのしくみ、とくに信頼性を高める技術について解説します。

提供元 技術評論社 http://gihyo.jp

07 プロが教える情報セキュリティの鉄則 2名

香山 哲司、小野寺 匠 著

サイバー攻撃、内部の不正、うっかりミス……。情報流失をいかに防ぐかを、情報セキュリティの考え方と、Windowsやディレクトリサービスを連携させた具体的な対策方法を示しながら解説します。

提供元 技術評論社 http://gihyo.jp

第1特集　新人歓迎企画【第1弾】

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）

【イラスト】高野 涼香

今回の特集は、新人歓迎企画の第1弾として、入社したて（もしくは他部署から配属したて）の新人が先輩から依頼を受け、その問答を通して最終的なゴールに向かってどのように考え、作業したら良いかを解説していきます。

GUIしか触ったことのない新人が、LinuxをCUI環境で動作させ、Webサーバを立ち上げるまでを想定しました。

それでは、Linuxの仮想環境を、仮想先輩・新人が作成する二重仮想劇場の幕開けです。

第1章 ルーキー君、Webサーバを動かしてみる! 18
Author 宮原 徹

第2章 ディレクトリ構造と役割がわかりますか? 28
Author 佐野 裕

第3章 ルートとユーザの違いがわかりますか? 34
Author 佐野 裕

第4章 設定ファイルが書けますか? 40
Author 鶴長 鎮一

第5章 コマンド、スクリプトが使えますか? 51
Author 鶴長 鎮一

第1章 ルーキー君、Webサーバを動かしてみる!

Author 宮原 徹(みやはら とおる) (株)びぎねっと

登場人物

M原
入社5年目の先輩社員。サーバ管理部門の係長。スパルタ気味だが面倒見が良い。

A垣
大学卒業したての新入社員。一通りの勉強はしてきたが、実践経験がない。

とあるSI企業の先輩・後輩

とあるSI企業に就職した新人。配属された部門の仕事は、おもにお客様のサーバ構築や運用保守を行っています。学校で一通りの基礎は勉強してきているけど、業務で役立つスキルとしてはまだまだ、といったところです。

ルーキー君の面倒を見る先輩は入社5年目。初めての直属の後輩ができて、張り切っているようです。

仮想マシンで自分だけの検証環境を作る

ルーキー。手始めに自分のPCに検証環境を作るぞ。まずは仮想マシンにOSをインストールして、Webサーバを動かすところまでやってみるからな。

はい。学校では用意された環境での実習しかやったことがないので、1からバッチリ覚えます! でも、検証環境ってなんですか?

システムに関わる業務において、事前の検証は重要な作業です。業務時間の大部分が検証作業であると言ってもいいでしょう。検証作業とは、触れたことのないソフトウェアを試してみたり、システムのトラブルを再現してみたりする作業です。そのため、スクラップ&ビルドで何度もやりなおしができる検証環境を作ることがスキルアップの近道であるといえます。

最近のPCは、CPUやメモリなどが高性能化し、使い切れないぐらいのパワーがあります。この余剰リソースを使って「仮想マシン」を動かせば、自分だけの検証環境を作ることができます。

VirtualBoxで仮想マシンを動かす

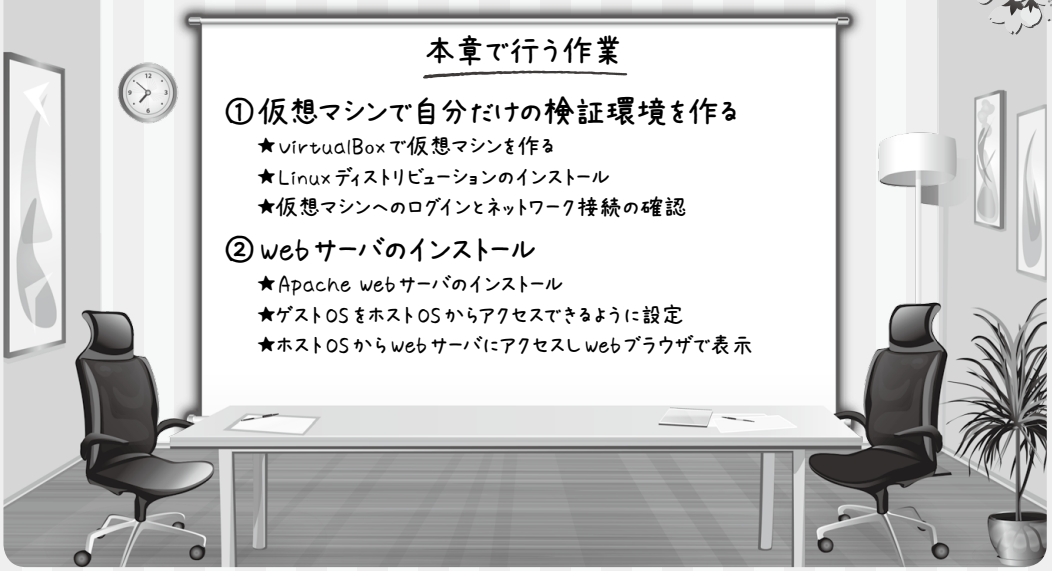
仮想マシンって使ったことないんですけれど?

仮想マシンを動かすにはいろいろなソフトウェアがあるけど、手っ取り早いのはVirtual Boxだな。無償で使えるから、まずはインストールしてみるか。

VirtualBox[注1]は、Oracle社が開発している仮想マシンソフトウェアです。オープンソースライセンスであるGPL[注2]でライセンスされているため、基本機能は無償で利用できます。

注1) https://www.virtualbox.org/

注2) GNU General Public License。フリーソフトウェア財団が公開するフリーソフトウェアのライセンスの1つ。https://www.gnu.org/gnu/gnu.html



USB対応など一部の機能を使うためのExtension PackはPUEL（VirtualBox Personal Use and Evaluation License）でライセンスされていますが、一般的な利用方法ではExtension Packは必要ありません。

VirtualBoxが動作するホストOSもWindowsやmacOS、Linuxと幅広く対応しているので、動作環境を選ばないのもうれしいポイントです。

VirtualBoxのインストーラは、VirtualBoxの公式サイト[注3]からダウンロードできます。ホストOSに合わせたインストーラをダウンロードして、インストールしましょう。

Linuxディストリビューションの選択と入手

LinuxでWebサーバを動かせ、ということなんですが、Linuxっていろいろな種類がありますよね。どれがいいんですか？

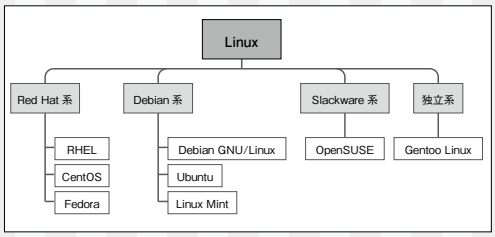
たしかにLinuxはいろいろなディストリビューションが用意されていて、選択に迷うことが多いよな。業務システムではRHEL（Red Hat Enterprise Linux）が使われることが多いけど、簡単なシステムならRHELクローンのCentOSとか、最近はUbuntuを使うことも増えてきているな。ルーキーは学校ではCentOSを使っていたみたいだから、今回はUbuntuに挑戦してみるか。

Linuxと言っても、さまざまなディストリビューションが用意されています（**図1**）。業務では目的に応じて使い分けられることもあるので、いろいろなディストリビューションをインストールして触ってみるとよいでしょう。

Linuxディストリビューションのインストールメディアは、インターネットからISOイメージという形でダウンロードできます。ISOイメージをDVD-Rなどの物理メディアに書き込むと、インストールディスクができますが、仮想マシンの場合にはISOイメージを仮想マシンでマウントし、直接インストールが行えるので便利です。

ISOイメージはそのディストリビューションの公式サイトからダウンロードできますが、サ

▼図1　代表的なLinuxディストリビューション



注3）https://www.virtualbox.org/

イズが大きいので世界中にミラーサイトが用意されています。もちろん、日本国内にもミラーサイトが複数用意されているので、インターネット的に距離が近い(=通信距離が短い)ミラーサイトからダウンロードするとよいでしょう。

今回はUbuntuをインストールしたいので、Ubuntuの公式サイトのダウンロードページ[注4]からUbuntu Serverを選び、「Ubuntu Server 16.04.1 LTS」をダウンロードします。Ubuntuのダウンロードの場合、自動的にミラーサイトを選択してダウンロードしてくれるので、自分でミラーサイトを選ばなくても大丈夫です。

「LTS」は「Long Term Support」の略で、長期間サポートされるという意味です。バグなどの不具合を直した修正版が長期間提供されます。業務システムは3年、5年と長期間に渡って動き続け、途中でバージョンアップすることが難しい場合も多いので、LTSのように長期間サポートされ続けるバージョンが選択されることを覚えておきましょう。

仮想マシンの作成

準備もできたし、さっそくOSのインストールですね!?

OSのインストールをするには、まずは入れ物を作らないとな。仮想マシンを作成するぞ。

仮想マシンの「仮想」とは、「ソフトウェアで論理的に」という意味です。通常、コンピュータはケースの中にCPUやメモリ、ディスク、ネットワークインターフェースなどが入っていますが、仮想マシンではこれらのリソースをすべてソフトウェアで論理的に再現します。仮想マシンを作成する作業は、CPUやメモリなどのリソースをどの程度割り当てるのか、ということを設定する作業です。

注4) https://www.ubuntu.com/download

とくにメモリは、実際にホストOS側のメモリを消費することになるので、使っているPCに搭載されているメモリの容量が少ないと、仮想マシンに必要なメモリが確保できない場合があります。LinuxでWebサーバを動かすだけならメモリは1GBも割り当てれば多過ぎるぐらいですが、ホストOSが使う分のメモリも必要ですので、ホストPCには8GBぐらいはメモリが搭載されていることが望ましいでしょう。

仮想マシンはディスクとして、仮想ディスクファイルをホストPCのディスク上に作成します。仮想マシンのディスクの読み書きは遅いため、ホストPCのディスクがハードディスクだと仮想マシンの動作が遅く感じられる場合があります。できれば、ハードディスクではなく、高速に読み書きが行えるSSDを使いたいところです。仮想ディスクファイルは使った分だけ容量が消費されるので、最大サイズだけ決めて作成します。単なる検証目的であればそれほどの容量は必要ありませんが、大量のデータを使って検証を行う場合には、最大サイズを大きめに設定しておきましょう。

VirtualBoxでは、対話型のウイザード形式で仮想マシンが作成できます。起動するとOracle VM VirtualBoxマネージャーが起動するので、「新規」を選択し、インストールするOSとしてLinuxを選択すると(**図2**)、デフォルト設定でメモリは1GB割り当てられます。仮想ディスクファイルも最大サイズ8GBに設定されます。

▼図2 名前とオペレーティングシステムの選択

検証環境としては十分ですので、ほかの設定もデフォルトのままで十分でしょう（**図3**）。

Webサーバとして動作させるということは、Webブラウザとネットワークで接続できなければなりませんが、仮想マシンのネットワークについては、あとでまとめて解説します。

OSのインストール

やっとOSインストールが始められますね。Ubuntuのインストールは初めてだから楽しみです。

最近のLinuxディストリビューションはインストーラがよくできているから、昔に比べると作業はずいぶんと楽になったな。何度でもやりなおしできるから、失敗を恐れずにまずはやってみるぞ。

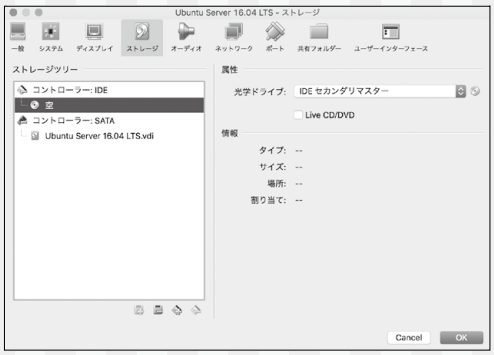
Oracle VM VirtualBoxマネージャー上の設定ボタンをクリックすると表示されるダイアログ上でストレージを選択します。すると**図4**のようなウィンドウが開きます。ここで「ストレージツリー」から「コントローラ：IDE」の下のCD-ROMマークを選択し、「属性」→「光学ドライブ」の右側にあるCDアイコンをクリックします。すると**図5**のボックスがポップアップするので「仮想光学ディスクファイルを選択...」を選び、ダウンロードしておいたOSインストール用のISOイメージファイル(今回はubuntu-16.04.*-server-amd64.iso）を選択すると、仮想マシンにISOイメージがマウントされます（**図6**）。

「OK」を押してダイアログを閉じ、Oracle VM VirtualBoxマネージャーの起動ボタンを押すとOSが起動し、Ubuntu Serverのインストーラが表示されます（**図7**）。このウィンドウではマウスは使えないので、矢印キーと Tab キー、Enter キーで操作を進めていきます。

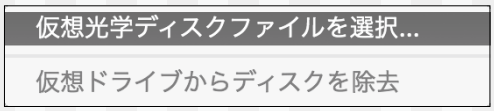
最近のLinuxディストリビューションは、インストール時には基本的な設定のみを行い、インストール後に詳細な設定を行わせることが多くなりました。インストールされるソフトも最初は最低限だけで、インストール後に必要に応

▼図3　Oracle VM VirtualBoxマネージャーに仮想マシンのスペックが表示されている

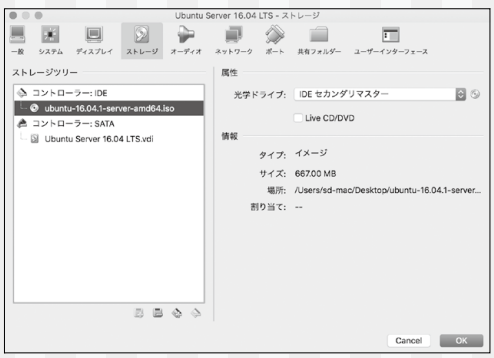
▼図4　ストレージの選択



▼図5　仮想光学ディスクファイルを選択する



▼図6　仮想マシンにISOイメージがマウントされる



▼図7　仮想マシンが起動し、Ubuntuのインストール画面が表示される

じて追加でインストールする必要があります。インストールは基本的にデフォルトを選択していけばOKです[注5]。

今回はWebサーバを動作させることが目的ですので、のちほどWebサーバのパッケージを追加でインストールします。パッケージには、ソフトウェアの実行ファイルのほか、設定ファイルやマニュアルなどがまとめられています。パッケージをインストールすると、それらのファイルがその種類に応じたディレクトリに保存されます。どんなディレクトリがあるのか、各ディレクトリの役割などは第2章で解説します。

ログインする

待つこと数分、インストールが終わりましたね。仮想マシンを再起動したら、無事にログインプロンプトが表示されましたよ!

まずは第1関門は突破というところだな。このあとはログインして、Webサーバを動かすためにいろいろな設定をしていくぞ!

インストールが終わって、システムを再起動すると最初に表示されるのがログインプロンプトです(図8)。インストール時に設定したユーザ名とパスワードを入力してログインし、いろいろなコマンドを実行してシステムを設定していきます[注6]。

ユーザ管理については、セキュリティポリシーと関係してきますので、しっかりと把握しておく必要があるでしょう。ユーザ管理については第3章で解説します。

ネットワーク接続の確認

うまくログインできました。さっそくWebサーバをインストールしましょう!

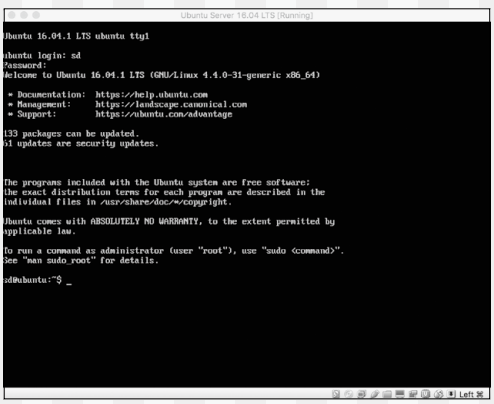
まあ、落ち着け。いきなり目的の作業に入るのではなく、肩慣らしも兼ねてシステムが正しく動作しているかどうか確認するのがお作法というものだ。この場合、インターネットにアクセスできるか確認するのが正解だな。

正常にログインできたら、さっそく作業に取りかかりたいところですが、OSが正しく動作していると思っても、実は想定していた設定と異なっている場合がよくあります。

注5) 本稿ではインストールの解説は省きます。インストール方法がわからない場合は各自ネットなどで調べてください。

注6) 今回インストールしたUbuntu Serverは、セキュリティも考慮してインストール時に一般ユーザしか作成していません。Linuxの管理者ユーザである「root」にパスワードを設定していませんので、ログインプロンプトから直接rootではログインできません。RHELやCentOSはrootパスワードを設定しますが、初期起動時に一般ユーザを追加で作成するように促してきます。

▼図8　Ubuntuにログインしたところ



今回は、インストール時にネットワーク接続の設定を行っていないため、DHCPを使って自動的にIPアドレスが取得されるようになっています。また、VirtualBoxで作成した仮想マシンは、デフォルトではNATを使って外部ネットワークに接続できるようになっています。しかし、外部から、あるいはホストOSからは仮想マシンにはアクセスできません。ちょうど、ブロードバンドルータのようなネットワーク機器が間に入っているイメージです。

まず、IPアドレスが正しく設定されているかを確認しましょう。確認にはipコマンドを実行します(**図9**)。

この実行例では、2つ目に「enp0s3」というネットワークインターフェースが動作しており、「10.0.2.15」というIPアドレスが割り当てられているのが確認できます。

次に、インターネットに接続できることを確認しましょう。確認方法はさまざまありますが、ここではcurlコマンドを使って外部のWebサーバにアクセスできることを確認します(**図10**)。

本来であれば、Webブラウザで表示されるHTMLがそのまま表示されたら成功です。もし、正常にアクセスできない場合には、原因を探って解決するトラブルシューティングを行わなければなりませんが、ネットワークについては次号で解説する予定です。

Webサーバのインストール

(トラブルが起きなくてよかった……。ネットワークトラブルは問題点の切り分けが面倒なんだよな……。ルーティングに名前解決、実はProxy設定が必要とか……。)

インターネットに接続できているようです。仮想マシンってすごいですね。……先輩?

お、おう。じゃあ、次はWebサーバをインストールするか。

事前にインターネットへの接続を確認したのは、このあとに行うパッケージのインストールの際、必要なパッケージをインターネット上のリポジトリから取得するためです。リポジトリとは、ディストリビューションが用意している

▼図9 IPアドレスの表示

```
$ ip addr show
(...中略...)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:ab:0e:ea brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feab:eea/64 scope link
       valid_lft forever preferred_lft forever
```

▼図10 curlコマンドの出力例

```
$ curl gihyo.jp
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:og="http://opengraphprotocol.org/schema/"
xmlns:fb="http://www.facebook.com/2008/fbml" xml:lang="ja" lang="ja">
<head>
(...中略...)
html = html + "</ul>";
document.getElementById("eventInfo").innerHTML = html;
</script>

</body>
</html>
```

パッケージが置いてある場所であり、実体はWebサーバです。

Ubuntuでは、パッケージの管理にAPTというしくみを使っています。リポジトリからパッケージをインストールするには、apt-getコマンドを利用します。

まず、apt-get updateコマンドを実行して、リポジトリから利用可能なパッケージに関する情報を取得します。次にapt-get installコマンドを実行して、Apache Webサーバをインストールします(**図11**)。

Proxy接続が必要なときには

企業では、セキュリティの関係でインターネットにアクセスする際にProxyの設定が必要な場合があります。このようなネットワーク環境では、外部のリポジトリにアクセスする際にはaptコマンドが参照できない場合があります。そのような場合には、ネットワーク管理者に相談して、Proxy経由でインターネットにアクセスする方法を確認してください。

apt-getコマンドは、実行時にrootユーザの管理者権限で実行する必要があります。そのため、コマンドの先頭にsudoコマンドをつけ加えておきます。sudoコマンドは初回実行時に実行ユーザのパスワードを入力する必要があります。また、ユーザによってはsudoコマンドの実行が許可されていない場合もあります。これらのポイントは第3章で解説します。

パッケージには依存関係があり、インストールしようとするパッケージにはその他のパッケージが必要となることがあります。apt-getコマンドは、この依存関係を自動的に解消して、同時にインストールを行ってくれます。

インストールが終わったら、Webサーバのサービスを起動します(**図12**)。サービス起動には今回はsystemctlコマンドを使用します。

表示を終了させるにはキーボードでqキー(Quitの意)を入力します。

Webサーバは正常に動作しているようですので、まずはローカル接続で動作テストをしてみましょう。インターネット接続のテストを行ったのと同様にcurlコマンドを使います。自分自

▼図11　apache2パッケージのインストール

```
$ sudo apt-get update
Hit:1 http://jp.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://jp.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
(...中略...)
Fetched 1,435 kB in 2s (686 kB/s)
Reading package lists... Done

$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
(...中略...)
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for ufw (0.35-0ubuntu2) ...
```

▼図12　apache2サーバのサービスを起動する

```
$ sudo systemctl start apache2
$ sudo systemctl status apache2
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
(...中略...)
Feb 16 16:50:15 ubuntu systemd[1]: Started LSB: Apache2 web server.
Feb 16 16:56:35 ubuntu systemd[1]: Started LSB: Apache2 web server.
```

身にアクセスするにはlocalhostというホスト名を使用します（**図13**）。

インターネット接続のテストのときと同様、HTMLが表示されたら動作テストは成功です。

ホストOSからWebサーバへのアクセス

コマンドでのアクセスだと、今ひとつWebサーバが動いているという感じがしませんね。やっぱりWebブラウザからアクセスしたいです！

たしかにそうだな。仮想マシンの設定を変更すれば、ホストOSから仮想マシンにアクセスできるようになるぞ！

VirtualBoxの仮想マシンはデフォルトでNATに設定されており、DHCPでIPアドレスなども自動的に割り当てられるため、何もしないでもインターネットに接続できました。しかし、この状態ではホストOSからはアクセスできないので、さまざまな検証を行うための環境としては不十分です。

VirtualBoxは仮想ネットワークの設定を変更することで、ホストOSから、あるいは外部のPCからも仮想マシンにアクセスできるようになります。ここでは「ホストオンリーアダプタ」を設定して、ホストOS上で動作しているWebブラウザから仮想マシン上で動作しているWebサーバにアクセスできるようにします。

まず、VirtualBoxの仮想ネットワークを設定します。「Oracle VM VirtualBox マネージャー」のメニューから「環境設定」を呼び出し、「ネットワーク」を選択し、「ホストオンリーネットワーク」タブを選択します（**図14**）。

次に、リストの右側にある「+」が書かれたボタンをクリックすると、「vboxnet0」が追加されるので、それを選択し、リストの右側にあるドライバのボタンをクリックします（**図15**）。

「アダプタ」タブのIPv4アドレスが「192.168.56.1」に設定されていることを確認してください（**図16**）。このアドレスはホストOS側

▼図13　localhostにアクセスして動作を確認する

```
$ curl localhost
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
(...中略...)
    </div>
  </body>
</html>
```

▼図14　「ホストオンリーネットワーク」タブを選択

▼図15　vboxnet0が追加される

▼図16　アダプタのIPアドレスをチェックする

で使用されます。

さらに、「DHCPサーバー」タブを選択し、「サーバーを有効化」をチェックしてください。そこで**図17**のように各パラメータに値を入力してください。設定が終わったら、OKボタンをクリックして環境設定ウィンドウは閉じます。

次に仮想マシンに仮想ネットワークインターフェースを追加します。このような仮想マシンの設定変更には、ゲストOSをシャットダウンする必要があります。

```
$ sudo shutdown -h now
```

シャットダウンが終了すると、仮想マシンが電源オフの状態になり、設定を変更できるようになります。「Oracle VM VirtualBox マネージャー」のウィンドウで変更したい仮想マシンを選択し、ウィンドウ右側下方の各種設定項目の中から「ネットワーク」の見出しをクリックします。

表示されたダイアログで「アダプター2」タブを選択し、「ネットワークアダプターを有効化」をチェックしてください。そして「割り当て」で「ホストオンリーアダプター」を選択します。「名前」は「vboxnet0」になります(**図18**)。

これでアダプタ2を設定すると、仮想ネットワークインターフェースが追加されます。OKボタンをクリックして設定ウィンドウを閉じてください。

それでは、新規に追加された仮想ネットワークインターフェースを確認してみましょう。仮想マシンを起動してログインし、再度、**図19**のようにipコマンドを使用します。

新たにenp0s8というネットワークインターフェースが追加されているのが確認できます。IPアドレスが割り当てられていないので、設定を変更してDHCPでIPアドレスが割り当てられるようにします。viエディタを使って、設定ファイル/etc/network/interfacesを編集します。

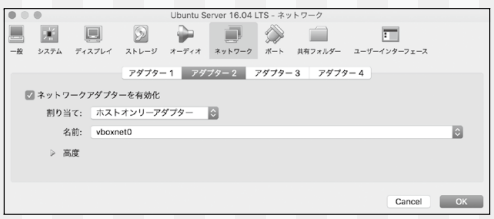
```
$ sudo vi /etc/network/interfaces
```

viエディタはモーダルエディタと呼ばれる、操作にやや癖があるエディタですが、次の手順で簡単に設定を書くことができます(**図20**)。

・編集位置を決めるカーソルはカーソルキーで移動させることができる

▼図17　各パラメータに値を入力する

▼図18　ネットワークアダプタでアダプタ2の設定



▼図19　ipコマンドで仮想ネットワークインターフェースを確認する

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
(...中略...)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group ➥
default qlen 1000
(...中略...)
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:42:0c:b8 brd ff:ff:ff:ff:ff:ff
```

・カーソルを、すでに記述されている設定「auto enp0s3」の行まで移動させる
・2 y y とキー入力する。これは「2行ヤンク(yank：コピー用のバッファに取り込むこと)するという意味です
・カーソルを、ファイルの末尾に移動させる
・p とキー入力する。ヤンクした2行がカーソルの下にペースト(Paste)される
・カーソルを、ペーストされた行の「auto enp0s3」の「3」に合わせる
・r とキー入力し、8 と入力する。「3」が「8」に置き換え(Replace)される
・同様に「iface enp0s3 inet dhcp」の行も、「3」を「8」に置き換える
・: w q Enter と入力して編集結果を保存し、viエディタを終了する

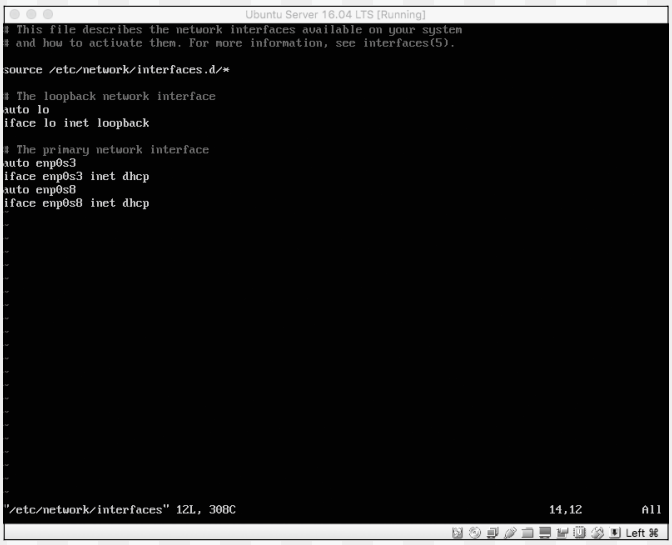
設定を反映させるため、systemctlコマンドを実行してネットワークを再設定します。

```
$ sudo systemctl restart networking
```

新しい仮想ネットワークインターフェースにIPアドレスが割り当てられたことをipコマンドで確認します(**図21**)。

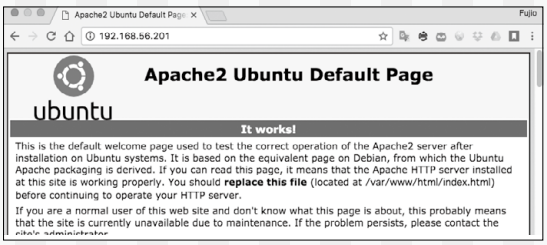
VirtualBoxのDHCPサーバで設定したIPアドレスの範囲のうち、一番最初のアドレスである「192.168.56.201」が設定されているのが確認できます。

▼図20　viエディタの編集画面



それでは、ホストOSのWebブラウザにこのIPアドレスを入力して、仮想マシンのWebサーバにアクセスできることを確認しましょう(**図22**)。「Apache2 Ubuntu Default Page」と書かれたページが表示されれば、Webサーバへの接続は成功です。SD

▼図22　仮想マシンのWebサーバのページが表示された



▼図21　新しい仮想ネットワークインターフェースを確認する

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
(...中略...)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group ➥
default qlen 1000
(...中略...)
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group ➥
default qlen 1000
    link/ether 08:00:27:42:0c:b8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.201/24 brd 192.168.56.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe42:cb8/64 scope link
       valid_lft forever preferred_lft forever
```

第2章 ディレクトリ構造と役割がわかりますか?

Author 佐野 裕(さの ゆたか) Twitter @sanonosa LINE㈱

登場人物

S野
データセンターの所長。インフラエンジニアとして一目置かれている。

A垣
大学卒業したての新入社員。学習意欲があり、将来有望。

ディストリビューションの違いが気になる

先輩、今回Ubuntu上にWebサーバをインストールしていますが、ほかのディストリビューションを使うときでも同じようにインストールすればいいんですか?

いや。Linuxのディストリビューションには無名なものも含めて数百種類あると言われているけれど、現在よく使われているものはおもにDebian系とRed Hat系の2系統にわかれている。Debian系ではapt-get(アプトゲット)コマンドを使うのに対してRed Hat系ではyum(ヤム)コマンドが使われている。今回使っているUbuntuはDebian系に属しているのでapt-getコマンドを使っているわけだ。

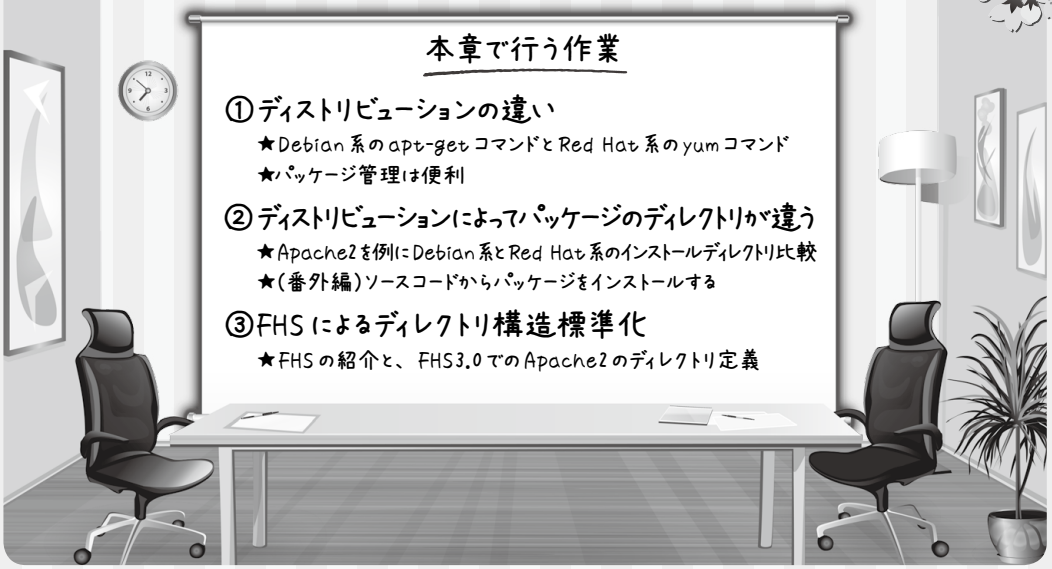
第1章でも少し紹介しましたが、Ubuntuを含むDebian系ディストリビューションではdeb形式と呼ばれるパッケージをapt-getコマンドを使ってインストールします。

```
$ sudo apt-get install apache2
```

それに対してRHEL(Red Hat Enterprise Linux)、CentOS、Fedoraなどを含むRed Hat

▼図1 apache2のインストール時に、ほかのライブラリもインストールされる

```
$ sudo apt-get install apache2

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.1-0 ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 129 not upgraded.
Need to get 1,554 kB of archives.
After this operation, 6,412 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```



系ディストリビューションではRPM形式と呼ばれるパッケージを、yumコマンドを使ってインストールします。

```
$ sudo yum install httpd
```

apt-getやyum[注1]はパッケージ管理システムの一種です。これらは各種のソフトウェアのインストール、削除、およびソフトウェア同士やライブラリとの依存関係などを管理しています。

Apacheを含む多くのソフトウェアは、さまざまなソフトウェアやライブラリを用いて作られています。パッケージ管理システムでは、1つのソフトウェアをインストールしたいときに足りないライブラリがあると、自動的にそのライブラリもインストール対象としたり、別のライブラリで必要とされるバージョンが、すでにほかで使われているソフトウェアと違うバージョンであるとき警告を出したり、といったことを実現してくれます。

図1の例では、Apacheをインストールしようとすると、apache2-bin、apache2-data、apache2-utils、libapr1、libaprutil1、libaprutil1-dbd-sqlite3、libaprutil1-ldap、およびliblua5.1-0 ssl-certなどの、追加で必要なライブラリも自動的にインストールしてくれます。

もし誤ってパッケージを入れてしまったときは削除することもできます(**図2**)。

注1) Red Hat系では将来yumに変わってdnfに移行される見込みです。

▼図2 パッケージの削除

```
$ sudo apt-get remove apache2

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.1-0 ssl-cert
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
  apache2
0 upgraded, 0 newly installed, 1 to remove and 89 not upgraded.
After this operation, 500 kB disk space will be freed.
Do you want to continue? [Y/n]
```

ディストリビューションによってパッケージが置かれるディレクトリが異なる

ディストリビューションによって使うコマンドが違うのはわかりましたが、違うのはコマンドだけなんですか?

ディストリビューションによってパッケージが置かれるディレクトリなんかも違うんだ。たとえば先ほどインストールしたApacheを見てみよう(**図3**)。

あ! 同じApacheでも雰囲気が違いますね。

そうなんだよ。ディストリビューションごとにパッケージの管理方法が異なるので、パッケージ内のファイルが置かれる位置もディストリビューションごとに異なっていることがあるんだ。

ディストリビューションごとに何がどこにインストールされるのか確かめるために、Apacheのソースコードをダウンロードして中身を見てみましょう。

まずは、https://httpd.apache.org/download.cgiからhttpd-2.4.25.tar.gzをダウンロードして展開します。

```
$ wget http://ftp.tsukuba.wide.ad.jp/
software/apache//httpd/httpd-2.4.25.
tar.gz
$ tar xvfz httpd-2.4.25.tar.gz
```

そしてconfig.layoutというファイルの中身を見ると、Ubuntuが属するDebian系(**図4**)やRed

▼図3 UbuntuとRed Hat系でのApacheインストール先の違い

```
・Ubuntu
$ sudo dpkg -L apache2
/.
/etc
/etc/logrotate.d
/etc/logrotate.d/apache2
/etc/apache2
/etc/apache2/sites-enabled
/etc/apache2/apache2.conf
/etc/apache2/mods-enabled
/etc/apache2/mods-available
/etc/apache2/mods-available/ssl.load
/etc/apache2/mods-available/ldap.load
/etc/apache2/mods-available/proxy_fcgi.load
/etc/apache2/mods-available/file_cache.load
/etc/apache2/mods-available/authz_host.load
/etc/apache2/mods-available/deflate.load
/etc/apache2/mods-available/cgid.load
(...後略...)

・Red Hat系
$ sudo rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/rc.d/init.d/htcacheclean
/etc/rc.d/init.d/httpd
/etc/sysconfig/htcacheclean・
(...後略...)
```

▼図4 Ubuntu(Debian系Linux)でのレイアウト

```
$ cd httpd-2.4.25/
$ more config.layout
(...中略...)
# Debian layout
<Layout Debian>
    prefix:
    exec_prefix:   ${prefix}/usr
    bindir:        ${exec_prefix}/bin
    sbindir:       ${exec_prefix}/sbin
    libdir:        ${exec_prefix}/lib
    libexecdir:    ${exec_prefix}/lib/
apache2/modules
    mandir:        ${exec_prefix}/share/man
    sysconfdir:    ${prefix}/etc/apache2
    datadir:       ${exec_prefix}/share/
apache2
    iconsdir:      ${datadir}/icons
    htdocsdir:     ${prefix}/usr/share/
apache2/default-site/htdocs
    manualdir:     ${htdocsdir}/manual
    cgidir:        ${prefix}/usr/lib/cgi-bin
    includedir:    ${exec_prefix}/include/
apache2
    localstatedir: ${prefix}/var/lock/apache2
    runtimedir:    ${prefix}/var/run/apache2
    logfiledir:    ${prefix}/var/log/apache2
    proxycachedir: ${prefix}/var/cache/
apache2/proxy
    infodir:       ${exec_prefix}/share/info
    installbuilddir: ${prefix}/usr/share/
apache2/build
    errordir:      ${datadir}/error
</Layout>
(...後略...)
```

Hat系（**図5**）でのディレクトリレイアウトが記載されているのを確認できます。

これを比較表にしたのが**表1**となります。

大きな違いとしては、データディレクトリ(datadir)が、Ubuntuの場合は「/usr/share/apache2」配下に、Red Hat系の場合は「/var/www」配下に置かれていることがわかります。

またApacheの設定ファイルは、Ubuntuでは「/etc/apache2/apache.conf」なのに対して、Red Hat系の場合は「/etc/httpd/conf/httpd.conf」であるといった違いなどもあります。

FHSによるディレクトリ構造標準化

ディストリビューションの種類にかかわらずディレクトリ構造が完全に統一されたらいいんですけどね。

Apacheなどのオープンソースのディレクトリ構造が詳細まで統一されることはないかもしれないが、実はLinux Foundationという非営利団体が各ディレクトリの標準化を行っ

▼図5　Red Hat系でのレイアウト

```
$ cd httpd-2.4.25/
$ more config.layout
(...中略...)
#   Red Hat Linux 7.x layout
<Layout RedHat>
    prefix:         /usr
    exec_prefix:    ${prefix}
    bindir:         ${prefix}/bin
    sbindir:        ${prefix}/sbin
    libdir:         ${prefix}/lib
    libexecdir:     ${prefix}/lib/apache
    mandir:         ${prefix}/man
    sysconfdir:     /etc/httpd/conf
    datadir:        /var/www
    installbuilddir: ${datadir}/build
    errordir:       ${datadir}/error
    iconsdir:       ${datadir}/icons
    htdocsdir:      ${datadir}/html
    manualdir:      ${datadir}/manual
    cgidir:         ${datadir}/cgi-bin
    includedir:     ${prefix}/include/ ➥
apache
    localstatedir: /var
    runtimedir:     ${localstatedir}/run
    logfiledir:     ${localstatedir}/ ➥
log/httpd
    proxycachedir: ${localstatedir}/ ➥
cache/httpd
</Layout>
(...後略...)
```

▼表1　ApacheのインストールディレクトリのDebian系とRed Hat系の比較

ディレクトリ名	Debian系（Ubuntu）	Red Hat系
exec_prefix	/usr	/usr
bindir	/usr/bin	/usr/bin
sbindir	/usr/sbin	/usr/sbin
libdir	/usr/lib	/usr/lib/apache
libexecdir	/usr/lib/apache2/modules	/var
mandir	/usr/share/man	/var/www/manual
sysconfdir	/etc/apache2	/etc/httpd/conf
datadir	/usr/share/apache2	/var/www
iconsdir	/usr/share/apache2/icons	/var/www/icons
htdocsdir	/usr/share/apache2/default-site/htdocs	/var/www/html
manualdir	/usr/share/apache2/default-site/htdocs/manual	/usr
cgidir	/usr/lib/cgi-bin	/var/www/cgi-bin
includedir	/usr/include/apache2	/usr/include/apache
localstatedir	/var/lock/apache2	/var/log/httpd
runtimedir	/var/run/apache2	/var/run
logfiledir	/var/log/apache2	/usr/man
proxycachedir	/var/cache/apache2/proxy	/var/cache/httpd
infodir	/usr/share/info	/var/www/build
installbuilddir	/usr/share/apache2/build	/usr/lib
errordir	/usr/share/apache2/error	/var/www/error

ソースコードをコンパイルしてApacheをインストールする方法

せっかくソースコードのダウンロードを行ったので、このままコンパイルしてインストールまで行う方法も記します。コンパイルとはC言語などで書かれたソースコードをコンピュータが解釈できるバイナリファイルに変換する処理のことを指します。

たいていのオープンソースでは、慣習的にconfigure→make→make installという手順でコンパイルとインストールを行えるように作られています(**図6**)。

運がよければconfigure→make→make installと実行することでインストールが完了します。しかしエラーや警告(Warning)が出たら、表示されるメッセージにしたがって対処が必要となります。

Apacheの場合はどうでしょうか。さっそくやってみましょう(**図7**)。

APRが見つからないというエラーで止まってしまいました。「ドキュメントを読んでください」と書いてあるので、ネットで検索してみると、どうやら別途「APR」と「APR-util」というソフトウェアが必要なようです[注A]。

解決方法がわかったので続けましょう(**図8**)。

configureがうまく通れば[注B]、あとはmakeとmake installです(**図9**)。makeには多少時間がかかります。またmake installにはルート権限が必要となります。

これでApacheのインストールが完了です。apt-getを使わないとけっこう手間がかかることがわかると思います。なおアンインストールしたいときにはmake cleanを実行します。

```
$ sudo make clean
```

注A) 参考までにAPR(Apache Portable Runtime)とは、OSとソフトウェアの間でOSなどの環境の違いを吸収するAPIで、ほかのOSに一般的にある機能が存在しないOSではAPRが代替を提供するといった役割があります。

注B) 状況によってPCREライブラリが必要というエラーが出る場合があります。その場合、エラーの最後の部分のURLを見て解決してください。

▼図7　Apacheで./configureを試みる

```
$ cd httpd-2.4.25/
$ ./configure
checking for chosen layout... Apache
checking for working mkdir -p... yes
checking for grep that handles long ➥
lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking build system type... x86_64- ➥
unknown-linux-gnu
checking host system type... x86_64- ➥
unknown-linux-gnu
checking target system type... x86_64- ➥
unknown-linux-gnu
configure:
configure: Configuring Apache Portable ➥
Runtime library...
configure:
checking for APR... no
configure: error: APR not found.   ➥
Please read the documentation.
```

▼図8　「APR」と「APR-util」をインストール後、./configureを実行する

```
$ cd ..
$ wget http://ftp.riken.jp/net/apache// ➥
apr/apr-1.5.2.tar.gz
$ wget http://ftp.riken.jp/net/apache// ➥
apr/apr-util-1.5.4.tar.gz

$ tar xvfz apr-1.5.2.tar.gz
$ tar xvfx apr-util-1.5.4.tar.gz
$ mv apr-1.5.2 httpd-2.4.25/srclib/apr
$ mv apr-util-1.5.4 httpd-2.4.25/srclib ➥
/apr-util
$ cd httpd-2.4.25
$ ./configure
(...中略...)
configure: summary of build options:

    Server Version: 2.4.25
    Install prefix: /usr/local/apache2
    C compiler:     gcc
    CFLAGS:          -g -O2 -pthread
    LDFLAGS:
    LIBS:
    CPPFLAGS:        -DLINUX -D_ ➥
REENTRANT -D_GNU_SOURCE
    C preprocessor: gcc -E
```

▼図6　ソースコードからコンパイルしてインストールする一般的な方法

```
$ ./configure          ←システムに合ったMakefile(コンパイル用設定ファイル)を作る
$ make                 ←コンパイルする
$ sudo make install    ←生成されたバイナリファイルを所定の位置に配置する
```

▼図9　makeとinstall

```
$ make
(...中略...)
make[4]: Leaving directory '//home/user1/httpd-2.4.25/modules/mappers'
make[3]: Leaving directory '//home/user1/httpd-2.4.25/modules/mappers'
make[2]: Leaving directory '//home/user1/httpd-2.4.25/modules'
make[2]: Entering directory '//home/user1/httpd-2.4.25/support'
make[2]: Leaving directory '//home/user1/httpd-2.4.25/support'

make[1]: Leaving directory '//home/user1/httpd-2.4.25'

$ sudo make install
(...中略...)
Installing man pages and online manual
mkdir /usr/local/apache2/man
mkdir /usr/local/apache2/man/man1
mkdir /usr/local/apache2/man/man8
mkdir /usr/local/apache2/manual
make[1]: Leaving directory '//home/user1/httpd-2.4.25'
```

ているFHSというものがすでにあったりするんだ。各ディストリビューションではFHSに沿ったディレクトリ構造がすでに採用されている。

FHS（File system Hierarchy Standard；ファイルシステム階層標準）とはLinuxを含むUNIX系OSにおけるディレクトリ名とその内容を定めているものです。FHSが制定される以前は、慣習的に使われていたディレクトリ名とその用途を、各Linuxディストリビューション開発者がそれぞれ独自解釈して用いていたので、結果としてディストリビューションごとにディレクトリの使われ方が異なるという状況でした。そこでLinuxでのファイルシステム階層標準化の試みが1993年より開始され、その後ほかのUNIX系OSも参加して現在に至ります。

FHSは現在、2015年6月3日に公開されたv3.0が最新版となっています[注2]。

Debian系（Ubuntu）やRed Hat系のApache2パッケージで用いられているディレクトリをFHS3.0に書かれている内容とマッピングさせたのが**表2**です。これを見るとFHSでは各ディレクトリについて厳密に定義しているというよりは、おおよその使い方が示されているものと言えそうです。このいろいろな解釈が成り立つ大らかさが、標準化されつつも、ディストリビューションごとの差異が生じる理由の1つとなっているように思います。SD

注2）http://refspecs.linuxfoundation.org/fhs.shtml

▼表2　FHS3.0でのApache2のディレクトリ定義

ディレクトリ名	FHS3.0
/etc/	ホスト固有のシステム設定
/usr	共有される読み出し専用データ
/usr/bin	大部分のユーザ用コマンド
/usr/lib	プログラミングとパッケージのためのライブラリ
/usr/sbin	必須ではない標準的なシステムバイナリ
/usr/share	構造から独立したデータ

ディレクトリ名	FHS3.0
/usr/share/man	マニュアルページ
/var	可変的なデータファイル
/var/cache	アプリケーションのキャッシュデータ
/var/lock	ロックファイル
/var/log	ログファイルとディレクトリ
/var/run	ランタイムの可変データ

第3章 ルートとユーザの違いがわかりますか?

Author 佐野 裕(さの ゆたか)
Twitter @sanonosa LINE㈱

登場人物

S野
データセンターの所長。インフラエンジニアとして一目置かれている。

A垣
大学卒業したての新入社員。疑問があるとすぐに質問する。

一般ユーザとルートの違い

さっきから気になっていたんですが、sudoを付けるときと付けないときがありますよね。どんな場合にsudoが必要なんですか?

一般ユーザがルート権限で作業が必要なときにsudoが必要となるんだ。

ルート(root)権限とはシステム管理に使う管理者権限のことです。LinuxなどのOSでは1台のマシンに大勢のユーザが接続して使うことを想定して作られています。そのために、一般ユーザでできることと、サーバ管理者がルート権限でできることとが明確に区別されています。ルート権限ではほぼすべてのファイルやディレクトリの閲覧・編集・削除ができます。

また、サーバのシャットダウン、リブート、もしくはアカウント作成などといった操作を行うときにはルート権限が必要となります。これらのコマンドを一般ユーザで実行しようとすると警告が出ます[注1]。

注1) ディストリビューションなどによって、警告なく実行できる場合もあるようです。

・サーバのシャットダウンやリブート

```
$ /sbin/shutdown -h now
shutdown: Need to be root

$ reboot
reboot: Need to be root
```

・アカウント作成

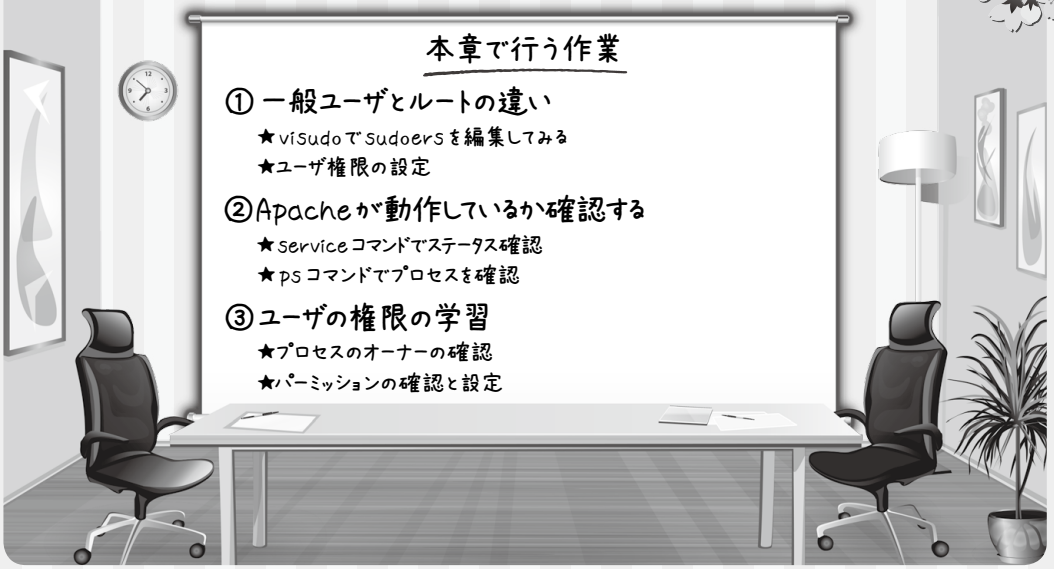
```
$ /usr/sbin/useradd
-bash: /usr/sbin/useradd: Permission denied
```

ルート権限で作業するためには、一般ユーザがsudoを使って作業をする方法と、rootアカウントで作業する2種類の方法があります。

sudoではルート権限で行えることを制限できる

sudoを付ければルート権限での作業ができるなら、一般ユーザとルートを分けている意味がない気がするんですが。

sudoは、一般ユーザがどうしてもルート権限を使って作業することが必要なときに使うんだ。ただしsudoをつければ常にすべてのルート権限が使えるわけではなくて、visudoコマンドを使って/etc/sudoersというファイルを編集することでsudoで行えることを制限することができる。もちろんsudoを使えるアカウントを限定することもできる。



visudoコマンドを使って/etc/sudoers設定ファイルを編集することで、sudoで実行できるコマンドを制限できます。

ただしsudoersの編集時に設定ミスがあると、最悪sudoが使えなくなってしまう恐れがあるので、sudoersを編集するときは通常visudoコマンドというsudoers編集専用コマンドを使います。このコマンドを使うと設定ミスがあったときに警告を出してくれます。

たとえば/etc/sudoersの最後の行に**リスト1**の記述を追加すると、user1というアカウントに対して、Apacheの起動、停止、およびステータス確認の操作を許可します。

次の記述を加えるとsudoを通してOSのシャットダウンと再起動の操作を許可します。

```
user1    ALL=PASSWD: /sbin/shutdown
user1    ALL=PASSWD: /sbin/reboot
```

sudoを使うときに、その都度一般ユーザのパスワードを入力させることも、パスワード入力なしにすることもできます。PASSWDの部分をNOPASSWDにするとsudoをパスワード入力なしに使えるようになります。

次の記述はsudoを通してすべてのコマンドの操作を許可します。ここではNOPASSWDですのでパスワード入力なしでsudoが使えるようになります。

```
user1    ALL=(ALL) NOPASSWD: ALL
```

ここではALLが3つ出てきていますが、それぞれ次のような意味があります。

・1個目のALL：すべてのホストから
・2個目の（ALL）：すべてのユーザ権限で
・3個目のALL：すべてのコマンドを実行できる

ところでsudoでは、ルートアカウントのパスワードを一般ユーザに教えなくてもルート権限が使えるというメリットもあります。もしsudoがなかったとしたらルート権限が必要なユーザすべてにルートのIDとパスワードを配布しなければなりません。

不特定多数が使うサーバで、管理者以外のユーザがルート権限を行使できるとしたら、サーバの安定稼働も個人情報保護もへったくれもありません。

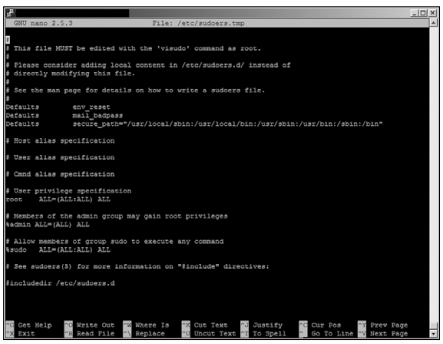
▼リスト1　/etc/sudoersに追記する

```
user1    ALL=PASSWD: /usr/sbin/service apache2 start
user1    ALL=PASSWD: /usr/sbin/service apache2 stop
user1    ALL=PASSWD: /usr/sbin/service apache2 status
```

COLUMN visudoの操作方法

visudoコマンドを実行するとGNU nano[注2]というエディタが起動します。

▼図1 visudo(GNU nanoエディタ)の編集画面



GNU nanoの操作は独特ですが、画面下位によく使う操作方法が記載されています。とくによく使うキー操作は次のとおりです。

・行カット:Ctrl+X(複数行まとめてカットすることもできる)
・行ペースト:Ctrl+U(複数行まとめてペーストすることもできる)
・ファイル保存:Ctrl+O
・終了:Ctrl+X
・ヘルプ:Ctrl+G

編集が完了してCtrl+Xを押すと**図2**のような選択画面が表示されます。ここでYを押すと保存するファイル名が表示されます。

```
File Name to Write: /etc/sudoers.tmp
```

そのままENTERキーを押してください。

▼図2 GNU nanoの「保存しますか?」の選択画面

```
Save modified buffer (ANSWERING ↵
"No" WILL DESTROY CHANGES) ?
 Y Yes
 N No            ^C Cancel
```

それくらいルート権限は強大ですので、不慮の事故を防ぐ意味でも、普段は極力一般ユーザアカウントでsudoだけを使うようにして、ルートアカウントでのアクセスは必要最小限にするというのが正しいサーバ運用の姿です。

Apacheが動作しているか確認する

sudoをつけてApacheを起動したので、ちゃんと動いていると思うんですが、本当に動いているか確認する方法ってありますか。

Webブラウザからページが見えるか確認するのが一番わかりやすいけれど、Apacheのプロセスがきちんと動いているか見てみることで確認する方法もあるんだ。

Apacheがきちんと動作しているかをサーバの中から確認する一番簡単な方法としては、serviceコマンドを使ったステータス確認方法があります(**図3**)。

もしくはpsコマンドを使ってOS上で稼働しているプロセスを見る方法もあります(**図4**)。

Linuxを含むたいていのOSでは、プログラムを実行するとプロセスという単位で起動します。プロセスはコンピュータのメモリ上にプロセス用のメモリ空間が確保され、そのうえにプログラムやデータが置かれます。そしてプロセスが何か処理を行うたびにCPUリソースの割り当てを受けて動作します。

と、難しく書きましたが、要するにLinux上で何かプログラムを実行すると、OS上にプロセスとして起動して実行されると思ってください。

プロセスにはオーナー(所有者)という概念があります。先ほどのpsコマンドの実行例では、1つ目のプロセスが「root(ルート)」、2つ目と3つ目のプロセスでは「www-data」という名前の一般ユーザがプロセスのオーナーとなります。

注2) UbuntuではGNU nanoが起動しますが、CentOSなどではviが起動するものもあります。解説と同じようにGNU nanoを起動するには「sudo nano /etc/sudoers」としてください。

▼図3　serviceコマンドを使ったステータス確認

```
$ sudo service apache2 status
apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Sat 2017-02-18 15:43:31 JST; 26min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 55
   Memory: 6.4M
      CPU: 829ms
   CGroup: /system.slice/apache2.service
           ├─4471 /usr/sbin/apache2 -k start
           ├─4474 /usr/sbin/apache2 -k start
           └─4475 /usr/sbin/apache2 -k start
(...後略...)
```

▼図4　psコマンドでプロセスを確認

```
$ ps -ef | grep apache2
root      4449     1  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start
www-data  4452  4449  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start
www-data  4453  4449  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start
```

プロセスは、場合によってさまざまなファイルにアクセスしたり、いろいろなプログラムを実行したりすることがあります。しかしプロセスは、無制限にそのような操作ができるわけではなく、プロセスのオーナーの権限が有効になります。

たとえばオーナーが一般ユーザのプロセスであれば、そのユーザがアクセスできるファイルにしかアクセスができませんが、rootがオーナーであればルート権限でのアクセスとなりますので、プロセスはほとんどのファイルにアクセス可能になります。

Apacheは一般ユーザの権限で動作している

そういえばApacheを使うと、いろいろなユーザのpublic_htmlディレクトリ内のデータを表示できますよね。これってもしかしてApacheのプロセスがルート権限で動いているから、どのユーザのデータでも扱えるんですか？

いや、Apacheは一般ユーザの権限で動作しているんだ。public_htmlディレクトリがApacheのプロセスから閲覧できるようなパーミッション設定がされている場合にのみ、Apacheから見ることができるよ。

Apacheの起動自体はルート権限で行う必要があります[注3]。しかしApacheがWebサーバと

COLUMN　マルチプロセス

プログラムが実行されるときにプロセスが1つだけしか起動しないとは限りません。プログラム内で複数の子プロセスが作られて複数の処理が並列に行われるようにしているプログラムもあります。このようなしくみをマルチプロセスと呼びます。Apacheもマルチプロセス型のプログラムの1つです。

プロセスが動作する際CPUコアが1つ割り当てられます。最近のコンピュータでは複数個のCPUコアを搭載するのが当たり前になりましたが、こういったコンピュータを使うと搭載CPUコアの数まで複数プロセスを並列に動作させることができます。

注3）厳密に言うとWebで標準的に使われるTCP80番ポートなど、1023番ポート以下のポート番号でApache2を起動するときだけルート権限で起動する必要があります。

して動作するためのプロセスは一般ユーザの権限で動作しています。

UbuntuのApacheパッケージの場合、標準ではApacheプロセスのオーナーが「www-data」という一般ユーザがであることが確認できます(**図5**)。

Apacheが「www-data」というユーザの権限で動作しているということは、普通に考えると該当プロセスからは「www-data」ユーザのデータしか扱えないはずです。しかし実際には、ほかのユーザのpublic_html配下も見えています。その秘密はパーミッションと呼ばれる権限設定にあります。

ファイルには「所有者」「所有グループ」「その他」という単位で「読み(r)」「書き(w)」「実行(x)」という権限を設定できます。また同じくディレクトリにも「読み(r)」「書き(w)」「ディレクトリ内に入ることができる(x)」という権限を設定できます。

ファイルやディレクトリのパーミッション設定は「ls -l」で確認できます(**図6**)。

実行結果の一番左が「-」の場合はファイルを、「d」の場合はディレクトリを表します。そのあとに続く「rwx」の部分はファイルの場合「読み(r)」「書き(w)」「実行(x)」という権限を、ディレクトリの場合は「読み(r)」「書き(w)」「ディレクトリ内に入ることができる(x)」と権限を表します。「rwx」の部分が3つ続いているのは、それぞれ「所有者」「所有グループ」「その他」を表します。

なお「rwx」については1〜7の数値で表すことがあります(**図7**)。

パーミッション設定の**図7**の例では、ファイルは所有者が読み書き実行でき、所有グループとその他では読みと実行ができますが、ファイルに対する書き込みや編集ができないことを意味します。同様にディレクトリは所有者が該当

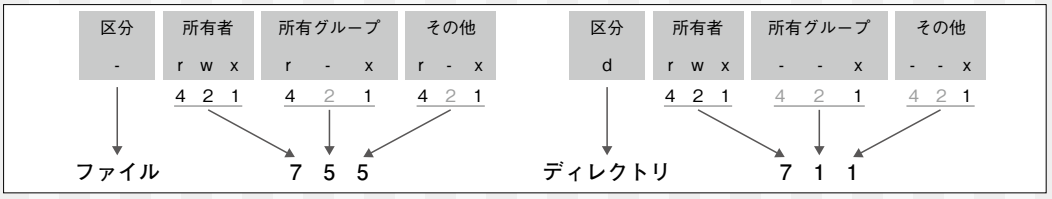
▼図5　Ubuntu上でApacheプロセスのオーナーを確認する

```
$ ps -ef | grep apache2
root      4449     1  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start
www-data  4452  4449  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start
www-data  4453  4449  0 01:10 ?        00:00:00 /usr/sbin/apache2 -k start

$ cat /etc/apache2/envvars | grep RUN
export APACHE_RUN_USER=www-data
export APACHE_RUN_GROUP=www-data
export APACHE_RUN_DIR=/var/run/apache2$SUFFIX
```

▼図6　パーミッションの確認

```
・ファイル
$ ls -l /home/user1/ | grep sample_file
-rwxr-xr-x 3 user1 user1    10 Feb 11 00:40 sample_file

・ディレクトリ
$ ls -l /home/user1/ | grep sample_dir
drwx--x--x  2 user1 user1    4096 Feb 20 00:33 sample_dir
```

▼図7　パーミッション設定



ディレクトリに読み書きとディレクトリ内への移動ができますが、所有グループとその他ではディレクトリ内に移動はできますが、ディレクトリの中を見たり何か書き込みや編集をできたりしないことを意味します。

ディレクトリの所有者でないユーザを使って実際に操作してみましょう(**図8**)。

さて、いよいよ本題ですが、「user1」というユーザの「public_html」ディレクトリが「/home/user1」配下にあるとすると、権限設定としては「/home」、「/home/user1」、および「/home/user1/public_html」の3つのディレクトリに対して、「その他」のユーザが「ディレクトリ内に入ることができる(x)」のパーミッション設定がされる必要があります。加えて「/home/user1/public_html」には「読み(r)」の設定も必要です。これらのパーミッション設定があって、初めてApacheプロセスから各ユーザの「public_html」配下を見ることができるようになります。

具体的に見てみましょう。**図9**の場合は3つ目の「rwx」の部分がいずれも「r-x」となっているため、Apacheプロセスは、そのプロセスの所有者である「www-data」の権限で該当ディレクトリの中身を見ることができます。

しかし次の場合は「/home」のその他の部分が「r--」となっているため、Apacheの実行ユーザである「www-data」の権限で該当ディレクトリの中身を見ることができません。

```
$ ls -l / | grep home
drwxr-xr-- 3 root root  4096 Feb 11 ↵
00:34 home
```

このような場合はchmodコマンドで「x」を有効にすることができます(**図10**)。SD

▼図8　ディレクトリ非所有者の動作

```
$ whoami
anotheruser2  ←ディレクトリの所有者でないユーザ

$ ls -l | grep sample_dir
drwx--x--x  2 user1 user1    4096 Feb 20 00:33 sample_dir

$ ls sample_dir/
ls: cannot open directory 'sample_dir/': Permission denied  ←読めない

$ cd sample_dir/
/sample_dir$  ←移動できた
```

▼図9　「www-data」の権限で該当ディレクトリの中身を見る

```
$ ls -l / | grep home
drwxr-xr-x 3 root root  4096 Feb 11 00:34 home

$ ls -l /home/user1 | grep user1
drwxr-xr-x 3 user1 user1  4096 Feb 11 00:38 user1

$ ls -l /home/user1/ | grep public_html
drwxr-xr-x 3 user1 user1  4096 Feb 11 00:38 public_html
```

▼図10　chmodコマンドで「x」を有効にする

```
$ sudo chmod 755 /home
もしくは
$ sudo chmod o+x /home

$ ls -l / | grep home
drwxr-xr-x 3 root root  4096 Feb 11 00:36 home
```

第4章

設定ファイルが書けますか?

Author 鶴長 鎮一(つるなが しんいち)

登場人物

T長
サーバ運用部門の先輩社員。システム企画開発なども手がける。

A垣
大学卒業したての新入社員。第1～3章の体験をしつつ、さらに学習意欲が湧く。

はじめに

仮想マシンで自分だけの検証環境を用意しWebサーバを動かせるようになった新人。Linuxのユーザ管理やディレクトリ構造も理解し、いよいよ実践的な設定作業にとりかかります。目標は、1台のサーバで複数のWebサーバを擬似的に運用する「バーチャルホスト」を構築することです[注1]。それにはLinuxの動作を制御する設定ファイルの役割や中身を理解する必要があります。

設定ファイルの基本:設定ファイルって何ですか?

新人。検証環境上のLinuxサーバを設定し、Webサーバをカスタマイズするぞ。それには設定ファイルを開いて修正するからな。

はい。で、設定ファイルって何ですか? コントロールパネルで変更できないんですか?

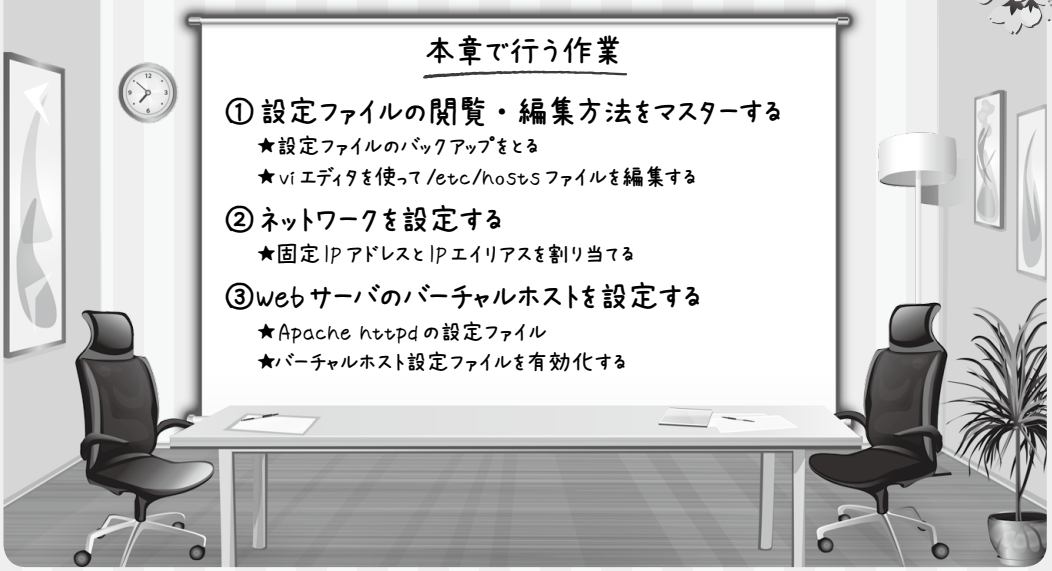
Linuxでの操作は、基本コマンドベースのCUI(キャラクタユーザインターフェース)です。そのため設定ファイルを開き、決められた書式で項目を追加したり値を変更したりすることで、OSやプログラムの動作を変更します。GUI(グラフィカルユーザインターフェース)ベースの設定ツールも一部出回っていますが(**コラム:LinuxのGUI管理ツール参照**)、Linuxをインストールしただけでは、そうしたツールを使うことができません。

設定ファイルはプレーンテキスト形式で記述されています。プログラムにしか解釈できないバイナリファイル形式とは違い、人の目で見て理解できます。そのため、ページャと呼ばれる「more」コマンドや「less」コマンドを使って、ファイルの中身を1ページずつ見ることができます。たとえば「/etc/hosts」ファイルを見てみましょう。

```
$ less /etc/hosts
```

moreやlessのキーバインド(キーへの機能割り当て)は**表1**のとおりです。「/etc/hosts」ファイルの中身を見ると、1行ごとに1つのIPアドレスと、それに対応するホスト名が空白で区切ったあとに記述されているのがわかります(**図1**)。「/etc/hosts」ファイルはIPアドレスとホスト名を対応させ、ホスト名を用いて通信相手を指定できるようにするものです。覚えづら

注1) 執筆にあたり、Ubuntu Server 16.04.1 LTS / Apache httpd 2.4.18を使用しています。



▼表1　more/lessの主要キーバインド

more/less共通キーバインド	
キーバインド	**説明**
Space	1画面分下にスクロール
f	1画面分下にスクロール
b	1画面分上にスクロール
q	表示終了
lessにだけ使えるキーバインド(一部)	
キーバインド	**説明**
g	ファイルの先頭に移動
Shift + <	ファイルの先頭に移動
Shift + >	ファイルの末尾に移動
Shift + F	監視モード

いIPアドレスを使わなくても済むため、作業効率が上がります。

moreとlessの違いは、moreが下までスクロールすると表示が終了するのに対し、lessは下までスクロールしても表示が終了せず、qキーをタイプするまで表示し続ける点です。そのため末尾から再びスクロールアップできます。

そのほか豊富なオプションを備えるなど、lessの方が高機能です。たとえば「-N」オプションをつけてlessでファイルを開くと、行番号をつけて表示します(**図2**)。ただしLinuxをインストールしただけの状態では、lessが使えないディストリビューションもあります。対してmoreは、どんな状況でも使えることが多く、組み込み系デバイスのようなリソースに制限がある環境でも、たいてい使用できます。

設定ファイルのバックアップをとる

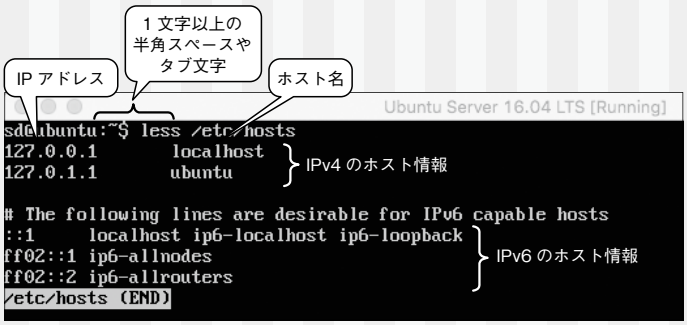
では、さっそく設定ファイルを開いて編集してみますね。

ちょっと待って！　いきなり設定ファイルを編集する前に、バックアップをとっておこう。

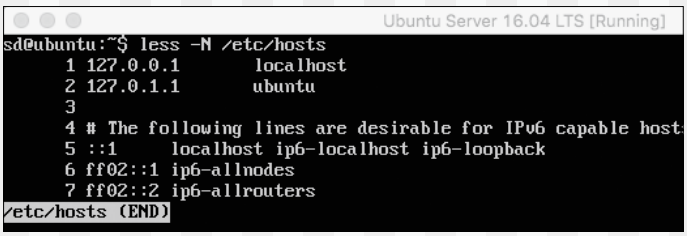
設定ファイルを編集したあとに、動作が不安定になったり、不具合が発生したりすることが多々あります。そうした場合に備え、設定を元に戻せるよう設定ファイルのバックアップを取っておくようにします。バックアップと言っても特別なツールを使うわけではありません。単に編集前の設定ファイルのコピーをとっておくだけです。バックアップファイルには「.bak」や「.org」といった拡張子をつけておくと、あとでそれとわかるため、まとめてバックアップファイルを削除する際に役立ちます。

```
$ cd /etc/
$ cp hosts hosts.org
```

▼図1　lessコマンドで「/etc/hosts」ファイルを表示



▼図2　「less -N /etc/hosts」を実行し行番号を表示



また「~(チルダ)」や「_(アンダースコア)」のようなサフィックスを用いる場合もあります。バックアップファイルに用いる拡張子やサフィックスは、チームや組織で統一しておきましょう。

また設定ファイルを編集するたびにバックアップファイルを作成し、前の前の状態に戻したり、さらに前の前の前の状態に戻したりするには、拡張子やサフィックスに番号をつけて、「hosts.org1」「hosts.org2」とします。こうした履歴を保ちながらバックアップファイルを管理する方法を「世代管理」と呼びます(**図4**)。

世代管理は、どの時点でどんな変更を加えたか確認できる反面、世代数が多くなるとファイルの移動が面倒になります。複数世代のバックアップファイルを保存する場合は、日付を用いると管理が簡単になります。

```
$ cp hosts hosts.20170204
```

実はcpコマンドにはバックアップに便利なオプションが用意されており、履歴管理のためのバージョン番号を自動で振り、1世代前のファ

▼図4　設定ファイルを世代管理する

・1世代目のバックアップファイルを作成

```
$ cp hosts hosts~1
```

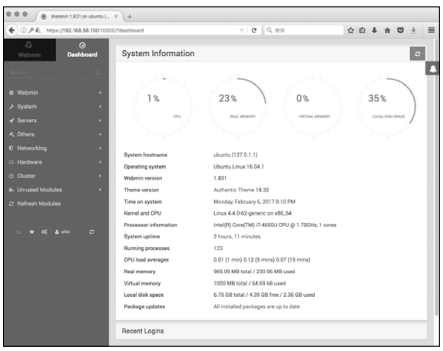
・1世代目のバックアップファイルを2世代目に移動し、あらたに1世代目のバックアップファイルを作成

```
$ mv hosts~1 hosts~2
$ cp hosts hosts~1
```

COLUMN Linuxの GUI管理ツール

LinuxにもGUI管理ツールがあります。多くはWebブラウザを使って管理するものです(**図3**)。そのためWebサーバの環境を整える必要があり、単にLinuxをインストールしただけでは使えません。またGUIツールだけでは管理できない機能もあるため、通常は補助的なツールとして利用します。

▼図3　Linux用GUI管理ツール「Webmin」のメイン画面



▼図5　cpコマンドの「--backup=numbered」オプションで世代管理する

・1回目のバックアップ

```
$ cp --backup=numbered foo foo.org
$ ls -l
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo.org
```

・2回目のバックアップ

```
$ cp --backup=numbered foo foo.org
$ ls -l
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo
-rw-rw-r-- 1 shin shin 0  2月  7 07:17 foo.org
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo.org.~1~
```

・3回目のバックアップ

```
$ cp --backup=numbered foo foo.org
$ ls -l
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo
-rw-rw-r-- 1 shin shin 0  2月  7 07:18 foo.org
-rw-rw-r-- 1 shin shin 0  2月  7 07:16 foo.org.~1~
-rw-rw-r-- 1 shin shin 0  2月  7 07:17 foo.org.~2~
```

イルを2世代前に移行するといったファイルのローテションも自動で行います。それには、「--backup=numbered（または--backup=t）」オプションを使用します（**図5**）。

viエディタを使って/etc/hostsファイルを編集する

では、さっそく/etc/hostsファイルにデフォルトゲートウェイのIPアドレスとホスト名を追加して、ホスト名でアクセスできるようにしてみます。編集はviエディタでいいんですよね？

基本はviだ。viはGUIベースのエディタと違って、すべての操作をキーボードで行うから、最初は使いづらいかもしれないが、操作に慣れておこう

設定ファイルの編集にはviエディタを使用します。viは多くのLinuxディストリビューションに標準でインストールされているため、追加インストールしなくても使用できます。ホームポジションに手を置いたまま操作できるよう、すべての操作をキーボードで行います。そのため多少操作方法に慣れる必要があります。なお、現状ではviを高機能化させたvimが主流になっており、viエディタを起動したつもりでも、実際はエイリアス（第5章で説明）されたvimエディタが起動します（**図6**）。

vi/vimではモードを使い分けて操作します（**表2**）。起動直後は「コマンドモード」になっており、**表3**のようなコマンドを使用します。編集するときにはiキーをタイプし「編集モード」に移行します。モードを移行すると、左下に「-- 挿入 --（英語表記では-- INSERT --）」と表示されるため**表4**のようなキーを使ってファイルを編集します。編集が完了し変更を保存するには

▼表2　viのモード

モードの種類表示	左下の表示	内容
コマンドモード	なし	各種コマンドを利用できる
編集モード	―挿入―、―置換―	入力できる

▼表3　viのコマンドモードで使用する主なコマンド

操作	コマンド
上書き保存	：w
終了	：q
保存して終了	：wq
保存せず終了（強制終了）	：q!
行番号を表示	：set number

▼図6　viはvimのエイリアス

```
$ ls -l /usr/bin/vi
lrwxrwxrwx 1 root root 20  2月  4 02:56 /usr/bin/vi -> /etc/alternatives/vi
$ ls -l /etc/alternatives/vi
lrwxrwxrwx 1 root root 18  2月  4 03:00 /etc/alternatives/vi -> /usr/bin/vim.basic
```

↑vimのシンボリックリンク

▼表4　viの主なキーバインド

操作	コマンド
カーソルを左へ移動	h(または←)
カーソルを右へ移動	j(または↓)
カーソルを上へ移動	k(または↑)
カーソルを右へ移動	l(または→)
行頭に移動	0
カーソルのある場所の1文字を削除	x
カーソルのある場所の前の1文字を削除	Shift＋x
1行カット	dd
指定行数を削除	数字dd
1行ヤンク	yy
数字の行数だけヤンク	数字yy
ペースト	p
カーソルを左へ移動	h(または←)

Escキーをタイプしコマンドモードに戻り、続けてwキーをタイプします。vi/vimではモードを切り替えながら操作します。どちらのモードなのかわからなくなったら、Escキーを1回、または2回タイプしコマンドモードに戻るようにしてください。

それでは、「/etc/hosts」ファイルにデフォルトゲートウェイのIPアドレスとホスト名を追加して、ホスト名でアクセスできるようにしてみます。現在設定されているデフォルトゲートウェイのIPアドレスをipコマンドで確認します(**図7**)。

この実行例ではデフォルトゲートウェイのIPアドレスは「10.0.2.2」です。ホスト名にはASCII文字の「a～z」、「A～Z」、「0～9」と、「-(ハイフン)」だけを使うことができます。ただし、ホスト名の最初と最後に「-」を使うことはできません。今回は「default-gw」とします。

▼リスト1　/etc/hostsにデフォルトゲートウェイのIPアドレスとホスト名を追加

```
10.0.2.2 【空白文字】 default-gw
```

「/etc/hosts」ファイルをviエディタで開きます。「/etc/hosts」ファイルを開くには次のようにします。設定ファイルの編集には管理者権限が必要になるため、Ubuntuではsudoをつけて実行します。

```
$ sudo vi /etc/hosts
```

「127.0.……」で始まる冒頭の2行は自ホストを表しています。1行目はどんなLinuxディストリビューションでも共通して使われる「localhost」に関する記述、2行目はUbuntuを始めとするDebian系ディストリビューションに見られる、自身のホスト名に関する設定です。現段階では変更しないようにします。カーソルを2行目の末尾まで移動し、iキーをタイプし編集モードに移行します。左下に「-- 挿入 --」と表示されているのを確認できたら、改行を挿入し、**リスト1**の1行を追加してください。空白文字には、一文字以上の半角スペースやタブ文字を使用します。

編集が終わったらファイルを上書き保存しviエディタを終了します。それにはEscキーでコマンドモードに移行し、「:wq」を入力します。

追記した内容は即座に反映されます。pingコマンドでホスト名が有効かどうか確認します(**図8**)。ホスト名からIPアドレスへの名前解決が行われ、pingによる応答確認に成功しているのがわかります。なおホスト名では、大文字小文字は無視されます。そのためホスト名に大文字を指定しても、同じように名前解決に成功します。

ローカルネットワーク上のホストの名前解決に「/etc/hosts」は便利です。ただしDHCPサーバのように動的にIPアドレスが変わる状況では、あまり役に立ちません。

▼図7　ipコマンドでデフォルトゲートウェイのIPアドレスを確認

```
$ ip route
default via 10.0.2.2 dev enp0s3 ←デフォルトゲートウェイ
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15
192.168.56.0/24 dev enp0s8  proto kernel  scope link  src 192.168.56.201
```

▼図8　pingコマンドで登録したホスト名が有効かどうか確認

・登録したホスト名でping

```
$ ping default-gw
PING default-gw (10.0.2.2) 56(84) ▶
bytes of data.
64 bytes from default-gw (10.0.2.2):
icmp_seq=1 ttl=64 time=0.255 ms
64 bytes from default-gw (10.0.2.2):
icmp_seq=2 ttl=64 time=0.228 ms
^C
(Ctrl+Cで終了)
```

・ホスト名に大文字を指定しても結果は同じ

```
$ ping DEFAULT-GW
PING default-gw (10.0.2.2) 56(84) ▶
bytes of data.
64 bytes from default-gw (10.0.2.2):
icmp_seq=1 ttl=64 time=0.695 ms
64 bytes from default-gw (10.0.2.2):
icmp_seq=2 ttl=64 time=0.579 ms
^C
(Ctrl+Cで終了)
```

コメント行の挿入

設定ファイルを修正する際に、どこを修正したのかコメントを残しておくと、あとあと運用を引き継いだときやトラブルが発生した際に役立てることができるんだ。

設定ファイルにコメントを書いてしまっても大丈夫ですか？　設定がおかしくなったりしないんですか？

決められた書式で書けば問題ないんだ。

設定ファイルにコメントを残すには、行頭に「#（シャープ）」を加えます。すると行全体がコメントアウトされ、設定ファイルを読み込む際に無視されます。

```
# Comment By Shin. 20170206
(...設定...)
```

また設定を書き換える際に、元の行をヤンク（コピー）し新しい行としてペーストしたあと、元の行の前に「#」をつけコメントアウトします。そうすることで、書き換える前の状態を残すことができ、トラブルが発生した際には、新しい行のほうをコメントアウトし、元の行の「#」を外すことで、設定を元に戻すことができます。

行途中からの#は、設定ファイルによって扱いが異なり、#から行末までコメントとみなされる場合もあれば、設定値の一部として扱われる場合もあるため、注意します。

ネットワークの設定：IPアドレスの固定化

サーバとして運用するからには、IPアドレスを固定化し毎回アクセス先のIPアドレスが変わらないようにしよう。

DHCPでは駄目なんですか？　固定化するとして、どの設定ファイルを修正すればいいんですか？

DHCPのように動的にIPアドレスを割り当てるシステムでは、ある一定の時間でIPアドレスが変更されたり、ホストを再起動するたびにIPアドレスが変更されたりする可能性があります。そのためWebサーバのように、クライアントからアクセスされるホストでは、IPアドレスを固定化し、アクセス先のIPアドレスが毎回変わらないようにします。DHCPサーバの予約機能で、割り当てるIPアドレスを固定化することもできます。最近は、仮想マシンやIaaS基盤の普及でDHCPサーバを使うことが多くなっていますが、サーバ側でIPアドレスを固定化したほうがトラブルが少なくなります。

IPアドレスを固定にするには「IPアドレス」と「ネットマスク」をネットワーク管理者から払い出してもらうようにします。今回のように自分のPCに仮想マシンを立ち上げている環境下では、自分で調べることもできます。それにはipコマンドを使用します。

```
$ ip addr show
```

enp0s8というネットワークインターフェースに対し、DHCPサーバから「192.168.56.201」がIPアドレスとして動的に割り当てられているのがわかります。LANに使用可能なIPアドレス「192.168.56.1～192.168.56.254」のうち、DHCPサーバ側は「192.168.56.200～192.168.56.250」のアドレスを使用しているため(第1章参考)、それ以外のIPアドレスをWebサーバに割り当てます。今回は「192.168.56.100」を固定IPアドレスとして静的に割り当てます。ネットマスクはipコマンドの結果から、「255.255.255.0」だとわかります。以上から**表5**のようなネットワーク情報を使用し、Webサーバを固定IP化します。

IPアドレスを固定化し、外部と通信するには、ほかに「デフォルトゲートウェイ」と「DNSサーバ」といった情報も必要になりますが、今回作成した検証環境では、ネットワークインターフェースとしてenp0s3とenp0s8の2つが割り当てられており、仮想マシンから外部への接続にはenp0s3が使用されています。そのためenp0s8にはデフォルトゲートウェイとDNSサーバを設定していません。ネットワークインターフェースが1つしかない環境では、**リスト2**のように設定してください。なお現在使用しているデフォルトゲートウェイは、本章前半で解説した「ip route」を使って調べることができます。使用中のDNSサーバは「/etc/resolv.conf」ファイルを開いて確認します。「nameserver」という項目に続けて記述されているIPアドレスがDNSサーバになります。

設定ファイルの/etc/network/interfacesをcpコマンドでバックアップし、その後viエディタを使って編集します。

```
$ sudo cp --backup=numbered /etc/ ➡
network/interfaces /etc/network/ ➡
interfaces.org
$ sudo vi /etc/network/interfaces
```

第1章で解説した手順が完了した状態では、**リスト3**のように設定されておりDHCPを使っているのがわかります。これを固定IP化するには、**表5**のネットワーク情報をもとに**リスト4**のように修正します。

設定を反映させるため、ゲストOSを再起動します。「$ sudo systemctl restart networking」を実行してネットワークを再設定したいところですが、DHCPサーバから割り当てられたIPアドレスが削除できないため、再起動します。

```
$ sudo shutdown -r -q now
(または単に「reboot」)
```

再起動完了後、「$ ip addr show」を実行し設定したIPアドレスがネットワークインターフェースに割り当てられていることを確認します。またホストOS側からping

▼表5　ホストに割り当てる固定IPアドレスとネットマスク

IPアドレス	192.168.56.100
ネットマスク	255.255.255.0

▼リスト2　/etc/network/interfacesファイルで固定IPアドレスを設定

```
(...省略...)
auto インターフェース名
  iface インターフェース名 inet staic ←静的割り当てでは「static」を指定
  address IPアドレス
  netmask ネットマスク
  gateway デフォルトゲートウェイ
  dns-nameservers DNSサーバ
(インデントは任意で実施)
```

▼リスト3　第1章で解説した手順が完了した状態での/etc/network/interfacesファイル

```
(...省略...)
auto enp0s3
iface enp0s3 inet dhcp  ←「dhcp」を指定しDHCPでIPアドレスを設定

auto enp0s8
iface enp0s8 inet dhcp  ←「dhcp」を指定しDHCPでIPアドレスを設定
```

▼リスト4　/etc/network/interfacesファイルを修正しネットワークインターフェースenp0s8を固定IP化

```
(...省略...)
auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
#iface enp0s8 inet dhcp          ←コメントアウト
  iface enp0s8 inet staic        ←静的割り当てを実施
  address 192.168.56.100         ←IPアドレスを指定
  netmask 255.255.255.0          ←ネットマスクを指定
(インデントは任意で実施)
```

を実行し「192.168.56.100」に疎通できることも確認しておきましょう。

複数のIPアドレスを割り当てる

1台のサーバで複数のWebサーバを擬似的に運用できるようネットワークインターフェースにIPアドレスを追加しよう。

仮想ネットワークインターフェースをもう1つ追加する方法では駄目なんですか?

第1章でも解説しましたが、仮想マシンなら簡単に仮想ネットワークインターフェースを追加できますが、物理サーバを使った場合、新たにNIC[注2]を購入しない限り追加できません。複数のIPアドレスを1台のサーバに持たせることが目的なら、LinuxのIPエイリアス機能を使うほうが簡単で経済的です。

IPエイリアス機能の使い方は簡単です。ネットワークインターフェース名に対し、「:番号」をつけたネットワークインターフェースを定義します。「enp0s8」に対しては、「enp0s8:1」を使用します。設定は/etc/network/interfacesで行います。IPエイリアスアドレスには「192.168.55.101」を使用します。/etc/network/interfacesをviエディタで開いて、**図10**のような設定を末尾に追加します。

注2）Network Interface Card。LANに接続するためのハードウェア。物理サーバの筐体背面や側面などに用意された拡張スロットなどに挿入して使用するカード型が一般的。

編集が終わったら、ネットワーク設定を再読み込みするように、「$ sudo systemctl restart networking」を実行します。その後「$ ip addr show」を実行し、IPエイリアスアドレスが割り当てられたことを確認します。またホストOS側からpingコマンドで「192.168.56.101」に対し疎通できることも確認しておきましょう。

Webサーバの設定：Apache httpdの設定ファイル

いよいよWebサーバの設定ですね。どの設定ファイルを編集すればいいんですか?

Webサーバの設定は今までの設定ファイルと違って、Apache httpd専用のものになるんだ。書式や構文が違うため注意が必要になる。それではWebサーバであるApache httpdの設定ファイルを見てみよう。

Apache httpd（以降、単にApache）には、どのような設定ファイルがあるのか確認してみましょう。設定ファイルの名称はディストリビューションによって異なります。Ubuntuでは/etc/apache2/ディレクトリに設定ファイルがあります（**図11**）。

メインとなる設定ファイルはapache2.confですが、Ubuntuでは機能ごとに設定ファイルが細かく分けられています。また設定ファイルを作成しただけでは有効になりません。○○-available/ディレクトリ内に設定ファイルを作成し、そのうち有効化したいものを、○○-enabled/ディレクトリ内にシンボリックリンクを作成します。たとえばバーチャルホストに

▼図10　ネットワークインターフェースenp0s8にIPエイリアスを設定

```
(ファイル末尾まで移動)

auto enp0s8:1
  iface enp0s8:1 inet static
  address 192.168.56.101
  netmask 255.255.255.0
```

▼リスト5　ディレクティブを使って設定する

```
ServerAdmin (1文字以上の半角スペースやタブ文字) foo@example.jp
↑ディレクティブ                                  ↑引数
```

関する設定ファイルは、sites-available/ディレクトリ内に作成しますが、作成しただけでは有効になりません。a2ensiteコマンドを実行することで、初めて設定ファイルのシンボリックリンクがsites-enabled/ディレクトリ内に作成され有効になります。同じように外部設定ファイルを有効化するにはa2enconfコマンドを、拡張モジュールに関する設定を有効化するにはa2enmodコマンドを使用します。

Apacheでは行単位で設定を行い、1行で1つの事柄を設定します。行の先頭に設定項目名を記述し、設定内容を記述します。設定項目名には、「ServerName」や「DirectoryIndex」といったものを使います。Apacheでは、こうした項目名を「ディレクティブ(Directive)」と呼びます(**リスト5**)。

サーバ全体に対し設定を行うなら、どこに設定を追加しても構いませんが、特定のディレクトリやURLなどに設定を限定する場合は、「<Directory ○○>~</Directory>」のようなブロック形式で、設定範囲を限定する必要があります。この際使われるのが「コンテナ(Container)」指示子です(**リスト6**)。

バーチャルホストの設定

バーチャルホストの設定ファイルを/etc/apache2/sites-available/に作成すればいいんですよね。どんなふうに設定ファイルを作成すればいいんですか?

/etc/apache2/sites-available/にあるほかのファイルを参考に作成しよう。設定ファイルをコピーし、修正の必要な個所をviエディタで書き換えよう。

▼リスト6　コンテナ指示子で設定対象を限定する

```
<コンテナ指示子 引数>
    設定内容,,,
    ...
</コンテナ指示子>
```

▼図11　UbuntuにインストールされるApache httpdの設定ファイル

```
/etc/apache2/
├── apache2.conf        メインの設定ファイル
├── conf-available/     外部設定ファイルを収めたディレクトリ
│   ├── ...
│   └── ...
├── conf-enabled/       有効化された外部設定ファイルのシンボリックリンクを収めたディレクトリ
│   ├── ...
│   └── ...
├── envvars             apache2ctl コマンドの環境変数を設定するファイル
├── magic               mod_mime_magic モジュール用の設定ファイル
├── mods-available/     拡張モジュールに関する設定ファイルを収めたディレクトリ
│   ├── ...
│   └── ...
├── mods-enabled/       有効化された拡張モジュールに関する設定ファイルのシンボリックリンクを収めたディレクトリ
│   ├── ...
│   └── ...
├── ports.conf          サービスポート番号に関する設定ファイル
├── sites-available/    バーチャルホストに関する設定ファイルを収めたディレクトリ
│   ├── ...
│   └── ...
└── sites-enabled/      有効化されたバーチャルホストに関する設定ファイルのシンボリックリンクを収めたディレクトリ
    ├── ...
    └── ...
```

バーチャルホストに関する設定ファイルは「/etc/apache2/sites-available」ディレクトリ内に作成します。デフォルトでは、通常接続（HTTP）用の000-default.confファイルと、セキュア接続（HTTPS）用のdefault-ssl.confファイルが存在しています（**図12**）。

000-default.confファイルを元に、バーチャルホスト用のファイルを作成します。任意のファイル名を使用できますが、バーチャルホストに用いるホスト名やIPアドレスを使うのが一般的です。必ず拡張子として「.conf」をつけるようにします（**図13**）。

192.168.56.101.confファイルを**リスト7**のように修正します。変更必須なのは、先頭のVirtualHostディレクティブとDocumentRootディレクティブです。ほかのServerAdmin、ErrorLog、CustomLogディレクティブは必須ではありませんが、変更しておくと、あとあと役に立ちます。

続いてDocumentRootディレクティブで指定した、バーチャルホストがドキュメントルートとして使用するディレクトリを作成します。ここでは「/var/www/html2」を指定しています。続けてドキュメントルート内にindex.htmlファイルを作成します。/var/www/html2/index.htmlファイルを、「<h1>Hello 192.168.56.101!</h>」といった内容で新規に作成します。

```
$ sudo mkidr /var/www/html2
$ sudo vi /var/www/html2/index.html
```

バーチャルホスト設定ファイルを有効化する

作成したバーチャルホスト設定ファイルを有効化する必要があるんですよね？ どういった手順を実行すればいいんですか？

Ubuntuにインストールされる Apache httpd では、/etc/apache2/sites-enabledディレクトリに設定ファイルを作成しただけでは有効化されないんだ。決まった手順で有効化し、その後Apache側で設定ファイルの

▼図12　バーチャルホストに関する既存設定ファイルを確認

```
$ ls -al /etc/apache2/sites-available
合計 20
drwxr-xr-x 2 root root 4096  2月  5 16:17 .
drwxr-xr-x 8 root root 4096  2月  5 16:17 ..
-rw-r--r-- 1 root root 1332  3月 19  2016 000-default.conf
-rw-r--r-- 1 root root 6338  4月  6  2016 default-ssl.conf
```

▼図13　バーチャルホスト用のファイルを作成する

```
$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites- ➲
available/192.168.56.101.conf
$ sudo vi /etc/apache2/sites-available/192.168.56.101.conf
```

▼リスト7　バーチャルホスト設定ファイル/etc/apache2/sites-available/192.168.56.101.confを修正

```
<VirtualHost 192.168.56.101:80>          ←アクセスを待ち受けるIPアドレスとポート番号を設定
(...省略...)
    ServerAdmin webmaster@localhost      ←管理者のメールアドレスを設定
    DocumentRoot /var/www/html2          ←ドキュメントルートを設定
(...省略...)
    ErrorLog ${APACHE_LOG_DIR}/error_vhost.log                  ←エラーログの出力先を設定
    CustomLog ${APACHE_LOG_DIR}/access_vhost.log combined       ←アクセスログの出力先とフォーマットを設定
(...省略...)
</VirtualHost>
```

再読込を実行すれば、いよいよバーチャルホストにアクセスできるようになるぞ。

/etc/apache2/sites-available/ディレクトリに作成した192.168.56.101.confを有効化するには、a2ensiteコマンドを実行します。

```
$ sudo a2ensite 192.168.56.101
```

これにより/etc/apache2/sites-enabled/ディレクトリに192.168.56.101.confのシンボリックリンクが作成されます(**図14**)。

最後にApacheの設定ファイルを再読み込みするようserviceコマンドを実行します。

```
$ sudo service apache2 reload
```

これでホストOSからバーチャルホストへのアクセスが可能になります。ホストOS側でWebブラウザを立ち上げ、「http://192.168.56.101」にアクセスしてみましょう。

図15のような「Hello 192.168.56.101!」と書かれたページが表示されれば、バーチャルホストへの接続は成功です。また「http://192.168.56.100」にアクセスすると、従来の「Apache2 Ubuntu Default Page」と書かれたページが表示され、1台のサーバに2つのWebサーバが立ち上がっているのがわかります。

192.168.56.101.confを無効化するには、a2dissiteコマンドを実行し、Apacheの設定ファイルを再読み込みします。

```
$ sudo a2dissite 192.168.56.101
$ sudo service apache2 reload
```

ほかの設定ファイルにも**表6**のような有効化／無効化コマンドが用意されています。SD

▼図15　Webブラウザを使って「http://192.168.56.101」へのアクセスを確認

▼図14　etc/apache2/sites-enabled/ディレクトリに作成されたシンボリックリンクを確認

```
$ ls -l /etc/apache2/sites-enabled/
合計 0
lrwxrwxrwx 1 root root 35  2月  5 16:17 000-default.conf -> ../sites-available/
000-default.conf
lrwxrwxrwx 1 root root 38  2月  6 00:39 192.168.56.101.conf -> ../sites-
available/192.168.56.101.conf
```

▼表6　Apacheの機能別設定ファイルを有効化／無効化するコマンド

設定ファイルの種類	設定ファイルを収めたディレクトリ	有効化された設定ファイルのシンボリックリンクを収めたディレクトリ	有効化コマンド	無効化コマンド
全体の動作に関わる設定	/etc/apache2/conf-available/	/etc/apache2/conf-enabled/	a2enconf	a2disconf
拡張モジュールに関する設定	/etc/apache2/mods-available/	/etc/apache2/mods-enabled/	a2enmod	a2dismod
バーチャルホストに関する設定	/etc/apache2/sites-available/	/etc/apache2/sites-enabled/	a2ensite	a2dissite

第5章 コマンド、スクリプトが使えますか?

Author 鶴長 鎮一（つるなが しんいち）

登場人物

T長
サーバ運用部門の先輩社員。システム企画開発なども手がける。

A垣
大学卒業したての新入社員。ちょっと覚えることが多いが頑張っている。

はじめに

Linuxを実際に使いこなすためには、UNIXコマンドに慣れ親しむ必要があります。本章ではUNIXコマンドの基礎知識から始まり、環境変数を設定し作業しやすい環境を整えたり、複数の処理をまとめて実行できるシェルスクリプトを作成する手順まで、実践的な利用方法を解説します[注1]。

コマンドの実行：コマンド実行の基本

Linuxを使いこなすには、コマンド操作に慣れ親しむことが重要だ。これまでの章でも、ユーザ管理やネットワーク管理にコマンドを使用したが、ほかにも多くのコマンドがLinuxには用意されているんだ。

数が多そうですが、全部を覚えないといけないんですか？

システム管理、プロセス管理など、それぞれの分野ごとに基本的なものを押さえておけば大丈夫。使っていくうちにレパートリが増えるし、手で覚えるようになるから、とにかく慣れることが大事だ。

Linuxを活用するには、コマンド操作に慣れ親しみ、より多くのコマンドやオプションを使いこなせるようにしておきます。また状況に応じてコマンドを組み合わせるような柔軟性も身につけておくようにします。

コマンドは、次のように「オプション」と「引数」を指定することで動作します。オプションはその名のとおり、指定しなくても実行できます。引数もコマンドによっては指定不要です。

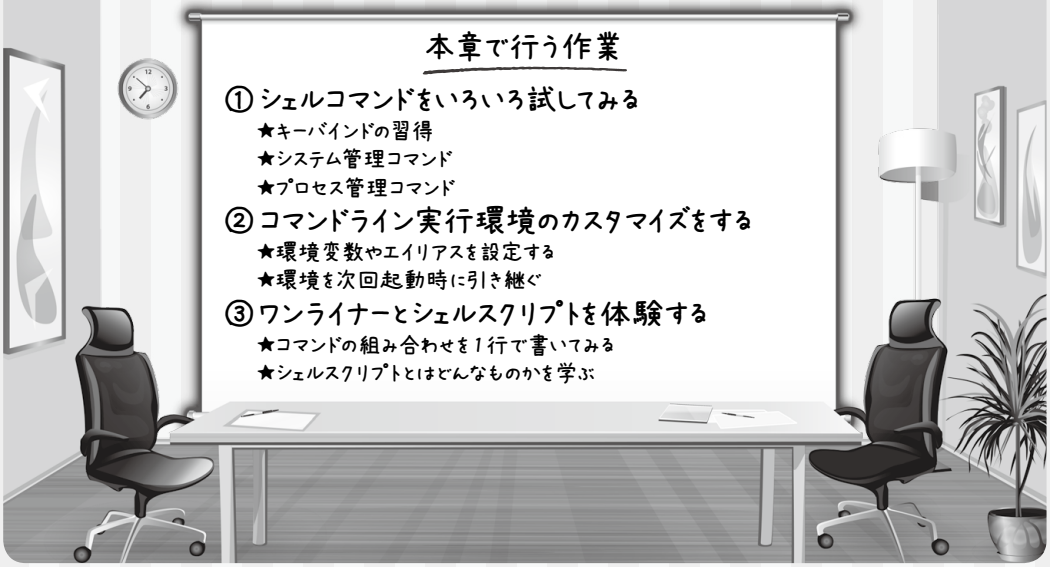
```
コマンド名 [オプション, ...] [引数, ...]
```

オプションは「-a」や「-h」のように「-（ハイフン）」をつけて指定します。指定可能なオプションはコマンドにより異なります。ただし例外はありますが、「-a」はすべてを対象に、「-h」はヘルプを表示するといったような、一定の共通性があります。コマンドを入力したのち、行の途中や行末で Enter キーを押下することでコマンドを実行します。

コマンド実行環境

Linuxではコマンドを入力し実行するのに“シェル”と呼ばれる実行環境を使うんだ。

注1）執筆にあたり、Ubuntu Server 16.04.1 LTSを使用しています。



シェルですか?

OSのコアにあたるLinuxカーネルと人が直接対話できない代わりに、シェルがユーザとLinuxカーネルを仲介しているんだ。

つまり、シェルがコマンドを探し起動することで、Linuxを操作するんですね。

そう、それが"コマンドインタプリタ"と呼ばれるシェルの重要な機能なんだ。ほかにもコマンド名を補完する機能や、過去に実行したコマンドを再実行する履歴機能といったものも、シェルが提供してくれるんだ。シェルを使いこなすことで、コマンドラインを使いこなすことができるようになるぞ。

「コマンドプロンプトを表示する→コマンドの入力を受け付ける→コマンドを解釈する→プログラムを探して起動する→結果を表示する」といった機能を提供するのがシェルです。シェルにはコマンド入力を補助するようなユーザ支援機能があり、コマンドライン操作を使いやすいものにしています。たとえば入力している文字列上で←や→キーでカーソルを移動したり、Tabキーでコマンド名を補完したりといったものがあります。UNIX系OSの操作系が同じなのは、こうしたシェルの操作性が共通なためです。

Linuxでよく利用されているシェルは「bash」です。bashでは**表1**のようなキーバインドでカーソルを移動したり、コマンドライン上の文字列を編集したりできます。

新たに入力するコマンドにはテキスト補完機能が役立ちます。たとえばtに続けてTabキー

▼表1 bashで利用可能なキーバインド

キーバインド	説明
Ctrl+B または ←	1文字戻る
Ctrl+F または →	1文字進む
Ctrl+H または BackSpace	カーソル前方(左)の1文字削除
Ctrl+D	カーソル後方(右)の1文字削除
Esc B	1ワード戻る
Esc F	1ワード進む
Esc BackSpace または Esc Ctrl+H	カーソル前方(左)の1ワード削除
Esc D	カーソル 後方(右)の1ワード削除
Ctrl+Y	最後に削除されたものをペースト
Ctrl+A	行頭に移動
Ctrl+E	行末に移動
Ctrl+K	カーソル位置から行の最後まで削除
Ctrl+P または ↑	前の履歴を表示
Ctrl+N または ↓	次の履歴を表示
Ctrl+S	履歴を検索
Ctrl+R	履歴をさかのぼって検索
Esc ←	履歴の先頭を表示
Esc →	履歴の最後を表示

※キー単体のものは、その順番でキーを押し、+のものは同時に押してください

を2回入力すると、tではじまるコマンドを一覧で表示します。

```
$ t[Tab][Tab]
tabs        tcpdump     tipc        troff
tac         tee         tload       true
tail        telinit     tmux        truncate
tailf       telnet      toe         try-from
(...省略...)
```

補完機能を使うことで、うろ覚えだったコマンドを呼び出すことができ、またコマンドの打ち間違いを防ぐことができます（**コラム：lsのタイプミス参照**）。コマンド名のほかにもファイル名、ディストリ名、ホスト名、ユーザ名といったものも補完できます。それには**表2**のようなキーバインドを使用します。

過去に実行したコマンドを再度実行したいときには履歴（history）機能が役立ちます。たとえば直前に実行したコマンドを、もう一度実行したいときには、[!][!][Enter]を入力します。

```
$ date
2017年  2月  8日 水曜日 23:38:26 JST
$ !![Enter]    ←直前のコマンドを再実行
date
2017年  2月  8日 水曜日 23:38:27 JST
```

ほかにも[↑]（または[Ctrl]＋[P]）で1つ前のコマンドを表示したり、[↓]（または[Ctrl]＋[N]）で1つ後ろのコマンドを表示するといったことができます。コマンド履歴を一覧で表示するには、historyコマンドを実行します。

▼表2　bashの補完機能を利用するためのキーバインド

キーバインド	説明
[Tab]	コマンド名、ファイル名、ディレクトリ名の補完
[Esc][?]または[Tab][Tab]	補完候補の一覧表示
[Ctrl]+[X][!]	コマンド名の補完候補を一覧表示
[Ctrl]+[X][/]	ファイル名／ディストリ名の補完候補を一覧表示
[Ctrl]+[X][~]	ユーザ名の補完候補を一覧表示
[Ctrl]+[X][@]	ホスト名の補完候補を一覧表示

※キー単体のものは、その順番でキーを押し、+のものは同時に押してください

```
$ history
(...省略...)
   21  ip a
   22  ip r
   23  sudo shutdown -h now
   24  ls -l
   25  date
   26  histroy
```

最近実行した10個だけ表示するには、引数に10を指定した「history 10」を実行します。履歴の左側に表示されている数字を使って「!**番号**」を実行すれば、番号で指定されたコマンドを再実行できます。たとえば先の例で「!24」を実行すると、「ls -l」が実行されます。

システム管理コマンド

状況に応じて使用できるコマンドを覚えておくと、簡単にレパートリを増やすことができるぞ。

ユーザ管理やネットワーク管理で使用できるコマンドは、前の章で教えてもらいましたが、ほかにどんな状況で使用できるコマンドがありますか？

まずはシステム管理に使えるコマンドを見てみよう。

lsのタイプミス

ファイル一覧を出力するlsコマンドは、頻繁に使用するぶんタイプミスすることも多く、「sl」と打ち間違えることがあります。そうした状況をジョークにしたのが、slコマンドです。Ubuntuでは次のようにインストールできます。

```
$ sudo apt install sl
```

slコマンドを実行するとどうなるかぜひ、たしかめてください。「-a」「-l」「-F」といったオプションも同じように指定できます。

システムを直ちに停止するには、shutdownコマンドに「-h now」オプションをつけて実行します。オプションを「-r now」に変えると、再起動させることができます。

```
$ sudo shutdown -h now
$ sudo shutdown -r now
```

「now +10」とすると10分後に停止／再起動させることがきます。指定した時間に停止／再起動させるには「hh:ss」と時間を指定します。22:00に停止するには次のようにします。停止をキャンセルするには「sudo shutdown -c」を実行します。

```
$ sudo shutdown -h 22:00
```

OS、CPUアーキテクチャ、ホスト名、カーネルのバージョンといったシステム情報を表示するにはunameコマンドを用います。「-a」オプションを指定すると各種情報を表示します。

```
$ uname -a
Linux ubuntu 4.4.0-62-generic ⏎
#83-Ubuntu SMP Wed Jan 18 14:10:15 UTC ⏎
2017 x86_64 x86_64 x86_64 GNU/Linux
```

システムがどのくらい連続して稼働しているか起動時間を表示したり、システムの負荷を表示したりするにはuptimeコマンドを使用します。

```
$ uptime
 01:45:26 up 6 min,  1 user,  load ⏎
average: 0.00, 0.03, 0.01
```

この例では現在時刻に続けて、稼働から6分が経過しており、ログイン中のユーザが1名いるのがわかります。load averageは直近1分間、5分間、15分間の平均負荷を、実行待ちのプロセスの数で表しています。この値がCPU数（コア数）を超えていると負荷がかかった状態になっていると言えます。

カーネルログを見るには、dmesgコマンドを実行します。

```
$ dmesg
```

空きメモリの容量を確認するにはfreeコマンドを使用します、「-m」オプションを指定するとMB単位に換算されます（**図1**）。「total」が搭載しているメモリの合計、「used」が使用中のメモリ容量、「free」が空き容量を表しています。

プロセス管理コマンド

プロセス管理コマンドもシステムの状況を把握するのに欠かせない重要なものだ。

プロセスって何ですか？（あれ？　前に習ったっけ？）

プログラムの基本的な実行単位がプロセスなんだ。それぞれのプロセスにはIDが振られており、プロセスIDを使うことで、プログラムの動作状況を把握したり終了させたりすることができるんだ。

実行中のプロセスやそのプロセスの状況を把握するにはpsコマンドを使用します。「-aef」オプションをつけると実行中の全プロセスが表示されます。

▼図1　freeコマンドで空きメモリをMB単位で表示

```
$ free -m
              total        used        free      shared  buff/cache   available
Mem:            974          70         732           4         171         729
Swap:          1023           0        1023
```

```
$ ps -aef
```

継続的にプロセスを監視するには「top」コマンドを利用します。上部にタスク数やCPUの使用率、メモリやスワップの使用状況など、システム全体の稼働情報が表示されます（**図2**）。

一定間隔で更新されるため、システムの変化を確認するのに便利です。表示内容はpsコマンドと同じですが、一部異なるものがあり、プロセスの仮想メモリサイズは「VIRT」、プロセスが使用している物理メモリサイズは「RES」で表されます。psコマンド同様、単位はKBです。topは一定間隔で表示内容を更新し、CPU時間の占有率（%CPU）順にプロセスの一覧を表示します。使用メモリ（%MEM）順に並び替えるには Shift + M をタイプします。終了するには q をタイプします。

▼図2　topコマンドの実行例

```
shin@ubuntu: ~
top - 04:12:39 up 30 min,  2 users,  load average: 0.21, 0.25, 0.17
Tasks: 253 total,   2 running, 251 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.2 us,  2.4 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   998308 total,    84868 free,   733360 used,   180080 buff/cache
KiB Swap:  1046524 total,   851652 free,   194872 used.    78168 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 8470 shin      20   0 1203716 164976  27604 S  3.3 16.5   0:12.59 compiz
 7933 root      20   0  353584  43084  15064 R  3.0  4.3   0:09.42 Xorg
 8958 shin      20   0  558896  31656  26388 S  1.3  3.2   0:00.19 gnome-scre+
 8855 shin      20   0  588496  33652  26264 S  0.7  3.4   0:00.62 gnome-term+
  917 root      20   0  112012   3668   3340 S  0.3  0.4   0:02.20 vmtoolsd
 8227 shin      20   0   87360   5096   4908 S  0.3  0.5   0:00.02 window-sta+
 8281 shin      20   0  517604  22364  16068 S  0.3  2.2   0:00.43 bamfdaemon
 8288 shin      20   0   33912   1260   1040 S  0.3  0.1   0:00.07 upstart-db+
 8306 shin      20   0  206968   5436   5284 S  0.3  0.5   0:00.03 at-spi2-re+
 8359 shin      20   0  564416  24716  19220 S  0.3  2.5   0:00.53 unity-pane+
 8956 shin      20   0   42924   3744   3020 R  0.3  0.4   0:00.02 top
    1 root      20   0  185388   4344   2876 S  0.0  0.4   0:01.35 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.43 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.71 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
```

▼図3　pstreeコマンドでプロセスツリーを表示

```
$ pstree
systemd─┬─accounts-daemon─┬─{gdbus}
        │                 └─{gmain}
        ├─cron
        ├─dbus-daemon
        ├─dhclient
        ├─login─────bash─────sudo─────apt─────http
        ├─rsyslogd─┬─{in:imklog}
        │          ├─{in:imuxsock}
        │          └─{rs:main Q:Reg}
        ├─sshd─────sshd─────sshd─────bash─────pstree
        ├─systemd─────(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-timesyn─────{sd-resolve}
        └─systemd-udevd
```

プロセスには親子関係があります。あるプロセスを起動するには、起動するための別のプロセスが存在し先に起動しています。pstreeコマンドを使えば、そうしたプロセスの親子関係をツリー形式で見ることができます（**図3**）。

プログラムが暴走し正常終了できない場合には、killコマンドでプロセスを終了させます。引数としてプロセスID（PID）を指定します。たとえばPIDが1454のプロセスを終了させるには次のようにします。

```
$ sudo kill 1454
```

またkillコマンドはプロセスを終了するためだけでなく、プロセスに各種シグナルを通知するのにも使用できます。たとえばHUPシグナルを送ると、再起動や設定の再読み込みを実行します。

```
$ sudo kill -HUP 1454
```

コマンドライン実行環境のカスタマイズ：環境変数を設定する

シェルは本当に便利ですね。でも、もっと使いやすくできないんですか？

カスタマイズしたいなら環境変数を利用するといい。環境変数を設定することでコマンド履歴を増やしたり、英語表記に変更したりできるんだ。

▼表3　主な環境変数

変数	説明	値の例
PWD	カレントディレクトリ	/home/shin
HOSTNAME	ホスト名	ubuntu
USER	現在のユーザ名	shin
TERM	ターミナル名	xterm-256color
MACHTYPE	マシンのアーキテクチャのタイプ	x86_64-pc-linux-gnu
LANG	使用する言語	ja_JP.UTF-8
HOME	ユーザのホームディレクトリ	/home/shin
PATH	コマンドサーチパス(本文参照)	/home/shin/bin:/home/shin/.local/bin:/usr/local/sbin...
SHELL	現在のシェル	/bin/bash
PS1	プロンプトの文字列	\h:\W \u\$
HISTSIZE	コマンド履歴の最大数	1000

※printenvコマンドを実行すると環境変数を一覧表示します

環境変数はシェル上で利用できる変数の1つです。ユーザの利用環境を定義するために使用され、シェル上で実行されるコマンドや、その子プロセスに引き継ぐことができます。たとえば、コマンドを入力する際、/bin/psや/usr/bin/topとパスをつけなくても、psやtopだけで実行できるのは、/binや/usr/binディレクトリにコマンドサーチパスが通っており、自動的に検索されるためです。コマンドサーチパスは、環境変数「$PATH」に格納されています。echoコマンドで表示してみましょう。

```
$ echo $PATH
/home/sd/bin:/home/sd/.local/bin:/usr/ ➥
local/sbin:/usr/local/bin:/usr/sbin:/us ➥
r/bin:/sbin:/bin:/usr/games:/usr/local/ ➥
games:/snap/bin
```

「:(コロン)」によって区切られた複数のディレクトリが表示されます。これがコマンドサーチパスに含まれているディレクトリです。前に書かれているディレクトリから順番に検索されます。$PATHに新たなディレクトリを追加するには、環境変数を設定します。環境変数を設定する方法はシェルによって違いますが、bashではexportコマンドを使用します。$PATHに/opt/binを加えるには次のようにします。

```
$ export PATH=$PATH:/opt/bin
```

環境変数を設定するには「export **環境変数名=値**」といった形式を用います。「echo $PATH」を実行し追加されていることを確認してください。なお「export PATH=/opt/bin」としてしまうと、既存のコマンドサーチパスが/opt/binに上書きされてしまいます。主な環境変数は**表3**のとおりです。

コマンド履歴の最大保持件数は1,000件ですが、変更するには環境変数「HISTSIZE」を設定します。併せて環境変数「HISTTIMEFORMAT」を設定すると、コマンドを実行した日時も記録するようになります。

```
$ export HISTSIZE=2000
$ export HISTTIMEFORMAT='%Y/%m/%d %H:%M:%S '
```

日本語表示可能なターミナルを使ってリモートログインしている場合は日本語表示できますが、コンソールからログインすると日本語が文字化けします(**図4**)。その場合は、環境変数$LANGに「C」を設定し日本語に翻訳しないようにします。

```
$ export LANG=C
$ date
Thu Feb  9 05:04:27 JST 2017 ←ASCII表記
```

aliasを設定する

手に馴染む感覚がわかってきました。ところで、長くなりがちなコマンドオプションを省略する方法はありませんか？

それならエイリアスを使うと良い。頻繁に利用するコマンドとオプションの組み合わせや、コマンドの短縮形をエイリアスを使って別名で実行できるようにしよう。

「エイリアス（alias）」を使うと頻繁に利用するコマンドとオプションの組み合わせを、コマンドの短縮形として別名で実行できるようになります。たとえばファイルを削除する際に、誤って削除するのを防ぐよう確認メッセージを表示するには、「**rm -i ファイル名**」を実行しますが、次のようにalias登録しておくと「**rmi ファイル名**」で削除確認メッセージが表示できるようになります。

```
$ alias rmi='rm -i'
```

▼図4　文字化けしたコンソール

```
Ubuntu 16.04.1 LTS ubuntu tty1

shintu login:
Password:

Login incorrect
ubuntu login: shin
Password:
Last login: Sun Feb  5 16:16:30 JST 2017 on tty1
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

131 ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
63 ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

shin@ubuntu:~$ _
```

aliasコマンドは次のように使用します。

```
alias 登録名='コマンド -オプション'
```

既存のコマンド名を上書きすることもできます。たとえばgrepやlsに毎回指定するオプションをつけて、あらためて同名のコマンドとして登録するには次のようにします。

```
$ alias grep='grep --color=auto'
$ alias ls='ls -F'
```

grepを実行しただけで、「--color=auto」オプションが付いたり、lsを実行しただけで「-F」オプションが付くようになります。単にaliasコマンドを実行すると、エイリアス登録されているコマンドが一覧表示されます。

環境変数やエイリアスを次回ログイン時にも引き継ぐ

設定した環境変数やエイリアスがログアウトすると消えてしまうようなんですが……。次にログインしたときに、もう一度設定しなおさないといけないんですか？

環境変数やエイリアスを.bashrcファイルに記述しておくと次回ログイン時に引き継ぐことができるぞ。

aliasコマンドで設定した内容や環境変数はログアウト時に失われます。次回ログイン時にも引き継がれるようにするには、ホームディレクトリにある「~/.bashrc」ファイルに設定内容を記述します。viエディタで.bashrcファイルを開き**リスト1**のように環境変数やエイリアスを記述します。設定した内容は再ログイン後から反映されますが、すぐに反映させるには、sourceコマンドを使用します。記述を間違えると、ログインできな

くなることもあるため注意してください。

```
$ source .bashrc
```

コマンドの組み合わせ、ワンライナーの活用

コマンドを使えるようになったら、次にコマンドを組み合わせて使えるようにしておこう。

コマンドを組み合わせるってどういうことですか？　コマンドは一度に1つしか実行できないんじゃないんですか？

パイプ処理と言って、コマンドの出力を次のコマンドに渡すことができるんだ。

へぇ便利ですね。

ほかにも、ファイルを読み書きするリダイレクトや、if文やwhile文といった制御式を使うこともできるぞ。こうしたコマンドの組み合わせをワンライナーと言うんだ。ワンライナーを状況に応じて反射的に組み立てられるようにしておくと、システム管理に非常に役立つぞ。

コマンドを組み合わせれば、大方の処理を自動化できます。たとえばホームティレクトリにあるファイルをサイズの大きなものから順に表示するには、duとsortコマンドを「| (パイプ)」で連結して次のように実行します。

```
$ du -s ~/* | sort -nr
```

▼リスト1　「~/.bashrc」ファイルに追加する内容

```
HISTSIZE=2000
HISTTIMEFORMAT='%Y/%m/%d %H:%M:%S '

alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias ls='ls -F'
(...省略...)
```

さらに、コマンドを組み合わせ、headコマンドでソートした結果を上位10個にしぼり、「>(リダイレクト)」を使ってファイルに出力するようにするには、次のような1行を実行します。

```
$ du -s ~/* | sort -nr | head -10 >> ~/ ↵
ファイル名
```

ファイル名にdateコマンドで実行した日付を付加し、「du_20170210.log」のような名前にするには次のようにします。

```
$ du -s ~/* | sort -nr | head -10 >> ~/ ↵
du_$(date +%Y%m%d).log
```

こうした1行スクリプトは「ワンライナー」と呼ばれ、ターミナルやコンソールから直接実行できるうえ、実行に多くのリソースを必要としないため、何度でも気軽に実行できます。そのため大量のファイルに同じ処理を繰り返すような作業や、日時で決まった処理をするような作業に有効です。さらにif文やwhile文といった制御式を使えば、条件判定やループ処理もできます。たとえば5秒ごとにvmstatコマンドを実行し、最新のシステムリソース状況を表示し続けるには、次のワンライナーを実行します。

```
$ while true ; do vmstat ; sleep 5 ; done ;
```

※終了するには Ctrl +c を押下します

シェルスクリプトの作成と実行：シェルスクリプトの基本

コマンドを実行し結果を見て、また次のコマンドを実行するって、システムと対話しているみたいですね。

そう、シェルはシステムと人が対話するためのしくみなんだ。

でもひとつひとつコマンドを打つのってたいへんじゃないですか？　まとめて実

行できないんですか？

まとめて実行したいなら、シェルスクリプトを利用するといいぞ。

シェルの利用方法には対話形式でコマンドを入力し実行する方法以外に、コマンドをスクリプト化した「シェルスクリプト」を実行する方法もあります。実行するコマンドを連ねることで複数の処理をまとめて行ったり、定型的な作業を自動で行ったりできるようになります。オプションや引数の指定が複雑になりがちなコマンドラインも、シェルスクリプトを利用することで簡単に実行できるようになります。またif文やfor文といった制御構文を取り入れることができるため、より高度な処理を実行できるようになります。

シェルスクリプトを作成するには、viなどのエディタを起動し、ファイルに実行したいコマンドを上から順番に書き連ねるだけです。試しに次の内容でファイルを作成し、「sample.sh」として保存してください。なおbashで動作するシェルスクリプトのファイル名には、伝統的に拡張子「.sh」を使いますが必須ではありません。

```
#!/bin/bash
echo "Hello bash!"
```

シェルスクリプトの1行目には、「#!」に続けて実行するシェルの種類を指定します。ファイル1行目の先頭2文字が「#!」だった場合、続く3文字目以降に書かれているプログラム名（ここでは/bin/bash）を使用してファイルの中身が実行されます。2行目以降に実行させたいコマンドを連ねて記述します。先の例では、echoコマンドにより引数として指定された「Hello bash!」が表示されるはずです。

実行はbashの引数としてファイル名を指定する方法と、スクリプトに実行権を付与し、スクリプト名で起動できるようにする方法があります。

```
$ bash sample.sh
```

または、

```
$ chmod +x sample.sh
$ ./sample.sh
```

で実行できます。

スクリプトに実行権を付与し、コマンドサーチパスの通ったディレクトリに保存しておけば、パス指定を不要にし通常のコマンドと同じように実行できるようになります。

シェルスクリプトを使ったシステム管理

シェルスクリプトを使ったシステム管理って、どんなことができるんですか？

システム構築から運用まで、どんな用途にもシェルスクリプトは使えるぞ。

定番のシェルスクリプトとかを、どこかからダウンロードできないんですか？

シェルスクリプトをマスターするには、試行錯誤を重ねながら、自分自身で組み立てるのが早道なんだ。他人が作ったシェルスクリプトをそのまま使うだけでは、ツールとして自在に操ることができないぞ。とにかくシェルスクリプトの例を1つみてみよう。

psコマンドで表示されるプロセス情報を、並び替えたり集計したりするシェルスクリプトを作成します。psコマンドは「-o」オプションを指定することで、見たい項目だけ表示することができます。たとえば「ps -o user,rss」を実行すると、各プロセスを実行しているユーザとメモリ使用量を表示します。さらに「--sort=user」オプションでユーザ名で並び替えたり、「--no-headers」オプションで不必要なヘッダ情報を表示しないようにしたりできます。こうしたpsコマンドの機能をシェルスクリプトに組み込み、ユーザごとのメモリ使用量を集計できるようにしたのが**リスト2**の「sample2.sh」

です。シェルスクリプトを実行すると、各ユーザのメモリ使用量を合計し表示します。

```
$ bash ./sample2.sh
```

スクリプトを数行改造するだけで、ほかの情報を表示できます。たとえば18～31行目の「for～done」を**リスト3**のように書き換えると、各ユーザごとの総プロセス数を表示するようになり、大量のプロセスを実行しているユーザをあぶり出したり、普段とは違った使い方をしているユーザを見つけたりできます。そのほかpsコマンドのオプションを修正することで、CPU時間やプロセス名などを表示し、使用率の高いプロセスを割り出したり、長時間起動したままのプロセスを検索したりといったこともできます。psコマンドの代わりにtopコマンドを使用したり、vmstatコマンドで仮想メモリに関する情報を表示したり、freeコマンドでメモリの使用状況を確認したりと、シェルスクリプトの活用方法はさまざまです。SD

▼リスト3　各ユーザの総プロセス数を表示するよう改造

```
# 18～31行目の「for～done」を次の4行に置き換える
for user in $(cat $tmpfile2); do
    cnt=`grep -c $user $tmpfile1`
    echo "$user $cnt個です"
done
```

▼リスト2　各ユーザのメモリ使用量を表示するシェルスクリプト(sample2.sh)

```
#!/bin/sh

#テンポラリファイルの作成
tmpfile1=`mktemp -t tmp.XXXXXXXXXX` || exit 1
tmpfile2=`mktemp -t tmp.XXXXXXXXXX` || exit 1

#psで実行中プロセスのユーザ名と使用メモリを取得しテンポラリファイル($tmpfile1)に格納
#その際、trコマンドで複数のスペース文字をスペース文字1個に置き換える
ps -ae -o user,rss --sort=user --no-headers | tr -s " " > $tmpfile1 2> /dev/null

#テンポラリファイル($tmpfile1)に対し、cutコマンドを使ってスペース文字で隔てられた
#各フィールドの内、1つ目のフィールドを取り出すことでユーザ名を取り出す。
#その結果をuniqコマンドで重複行を排除し、テンポラリファイル($tmpfile2)ファイルに格納
cut -d " " -f 1 $tmpfile1 | uniq > $tmpfile2 2> /dev/null

#ユーザ毎の使用メモリを集計
#$tmpfile2からユーザ名を取り出し、各ユーザが実行しているプロセスの使用メモリを集計
for user in $(cat $tmpfile2); do
    add="0";
    for memory in $(grep $user $tmpfile1 | cut -d " " -f2); do
        add=`expr $add + $memory`
    done

    if [ $add -gt 1024 ]; then
        #1024Kバイトを超えるものは単位をMバイトに変換
        add=`expr $add / 1024`
        echo "$user\t$add Mバイトです"
    else
        echo "$user\t$add Kバイトです"
    fi
done

#テンポラリファイルの削除
rm -f $tmpfile1 $tmpfile2

#最後にスクリプトを終了
exit 0
```

リスト3で置き換える範囲

第2特集

Serverless

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

昨今、大規模なWebサービスへの導入も聞くようになってきた「サーバーレスアーキテクチャ」。この新しい開発手法について、みなさんはどのようなイメージをお持ちでしょうか？ 「サーバーレスとはどういう状態なのか」「ここでのサーバとは何を指すのか」など、そもそも論な疑問を持つ方も多いはずです。本特集ではまずこういった素朴な問いからスタートし、新手法が確立されるに至った経緯を解説します。百聞は一見に如かず、「AWS Lambda」「Microsoft Azure Functions」という2つのサービスを実際に使って、これまでの開発がどのように変わるのかを体験しましょう。

Part 1 “サーバーレス”とはなにか? P.62
Author 仲山 昌宏

Part 2 [実践] AWS Lambda P.66
Author 永田 明

Part 3 [実践] Microsoft Azure Functions P.76
Author 東 聡志／日本マイクロソフト㈱ 監修

Part 1

“サーバーレス”とはなにか?

Author 仲山 昌宏(なかやま まさひろ) ㈱WHERE Twitter @nekoruri

実際のクラウドサービスを使ったサーバーレスアーキテクチャの構築に入る前に、「“サーバーレス”とはなにか」「そもそも“サーバ”とは何を指しているのか」という疑問について考えましょう。本章では3種類のサーバを定義して、どのような文脈でサーバーレスという考え方が登場したのか、どのような活用事例があるのかを解説します。

はじめに

「サーバーレスアーキテクチャ」と聞いたとき、どんなものをイメージするでしょうか。

このキーワードについて話をするとき、まず議論になるのがその定義です。言葉が登場した直後の「クラウドコンピューティング」がそうであったように、今はまだサーバーレスアーキテクチャの定義は、十分な共通認識が不足しているあいまいな状況です。この特集が、ふわっとした「サーバーレスアーキテクチャ」について整理する機会となり、より「クラウドらしい」実装方法を試すきっかけとなれば幸いです。

「サーバ」が「ない(レス)」とは

この1~2年でサーバーレスアーキテクチャをキーワードとした事例が増えてきましたが、どんな「サーバ」を「なくした」のかに注目して整理すると、大きく3つの考え方に分けることができます。

①「自分で管理するサーバ」をなくしてクラウド側にバックエンド処理全体をお任せするBaaS、PaaS
②「管理する粒度としてのサーバ」をなくして、起動が軽くてステートレスな関数という制約を活用するFaaS
③処理全体に責任を持つ「指揮者としてのサーバ」をなくし、クラウド上で提供されるコンポーネントを結び付けてシステムを作り上げるメッセージ駆動の手法

共通していることは、「実際にソフトウェアを実行する物理的なサーバがなくなるわけではない(=どこかにはある)」ことと、「クラウドが提供するアリモノを活用する」ということです。それぞれ説明していきます。

「自分で管理するサーバ」をなくそう=BaaS、PaaS

スマートフォンアプリやJavaScriptによるSPA(シングルページアプリケーション)など、フロントエンド側のリッチ化により、バックエンド側は認証処理やデバイス間のデータ同期、プッシュやメール通知、矛盾が許されないデータの保存など、本当に必要な処理に特化できるようになりました。

それを受けて、バックエンド側に必要な機能を提供するBaaS(Backend as a Service)、もしくはMBaaS(モバイルBaaS)と呼ばれるサービスが登場しました。残念ながら、MBaaSの代表格であったParse(2012年正式サービス開始)はつい先日にサービスを終了してしまいましたが、それと入れ替わるかのようにAmazon、Microsoft、Googleなど大規模クラウド事業者が、クラウドサービスの一機能として力を入れるようになってきています[注1]。サーバーレスアーキテクチャの1つめの定義では、このようなバックエンド側にBaaSを使ったシステムを指します。

注1) 国内事業者でも、ニフティクラウドなどがMBaaSを提供しています。

これらBaaSの最大の特長は「自分でバックエンド側のサーバを管理しなくて良い」ということに尽きます。BaaSを利用したシステムを作りたいときは、Webブラウザから使いたい機能を登録すれば、フロントエンド側からのアクセスに必要な認証情報が手に入ります。あとはBaaSのSDKに認証情報を渡せば、BaaSの豊富な機能を利用できます。もう自分でサーバを確保して、実行環境を構築したり、障害に備えたりする必要はありません。まさに「所有から利用へ」のクラウドらしい考え方と言えます。バックエンドとの通信など、フロントエンド側の開発も楽になります。

自分でサーバを用意しなくて良いものとして、HerokuやEngine YardのようなPaaS（Platform as a Service）もサーバーレスアーキテクチャに含めることがありますが、これについては次節であらためて取り上げます。

良き制約で「管理する粒度としてのサーバ」をなくそう＝FaaS

クラウド風のスケールアウト型のベストプラクティスを推し進めることで、管理する粒度としてのサーバという単位を考えずに済むようにしたFaaS（Function as a Service）が、2つめのサーバーレスアーキテクチャです。

◉Functionで処理を分割

システムをうまくスケールさせるための手法として、「プログラム自身の中に、複数の処理で永続する変数などを持たない」という制約が、The Twelve-Factor App[注2]などにより広く知られるようになりました。さらに小さく、ステートレスなFunction（関数）という単位で処理そのものを分割することで、起動コストなどのオーバーヘッドが小さくなり、プラットフォーム側がその負荷に見合ったサーバ数やプロセス数を割り当てるといったオートスケール処理を動的に実現できるようになります。

その結果、動的に増減するサーバ台数はブラックボックスとなり、利用者は実際に消費した処理時間の合計（しかも100ミリ秒などの小さな単位）だけを気にすれば良い、という実行環境が実現しました。「所有から利用への転換」がクラウドの特徴ですが、サーバーレスアーキテクチャはさらに一歩踏み込んで、「確保量から消費量への転換」を実現します。一般的にサーバーレスアーキテクチャと呼んだ場合、狭義にはなんとなくこの、「サーバという単位のことを考えずに済む」FaaSのプラットフォームや実行環境のことを指していることが多いようです。

◉サービスの登場

このようなステートレスな制約を活かしてスケールアウトさせる実行環境としては、実は2008年に登場したGoogle App Engineがその元祖にあたります[注3]。しかしその制約の厳しさが人類には早過ぎたこともあり、FaaSが広く活用されるには、そこから6年後のAWS Lambda（**第2章参照**）の登場を待つことになります。

AWS Lambdaによってあらためて再発明されたFaaSでは、APIでFunctionを呼び出せるほか、ほかのコンポーネントのイベントをトリガーとして起動したり、HTTP APIとして公開したりもできます。これらほかのコンポーネントとの連携を重視し、小さなコードでさまざまな事例を実現できるようにしたことが普及の要因と言えます。また、関数の出力を判定して別の関数につなぎ込むAWS Step Functionsや、関数の出力をそのまま別のサービスに流し込むAzure Functions（**第3章参照**）の出力バインドなどにより、さまざまな形での組み合わせが容易にできるようになり、FaaSの活用が一層進んでいくものと思われます。

◉PaaSとどう違う？

先ほど例に挙げたPaaSとFaaSの違いとし

注2）「The Twelve-Factor App（日本語訳）」 URL https://12factor.net/ja/
注3）その証拠として、プレビューリリースの時点では実行時間での課金モデルが採用されていました。

て、サーバーレスであるかどうかという視点で次のツイート[注4]がよく引用されます[注5]。

> *If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.*

従来の大きなソフトウェアをそのまま動かすPaaSでは、起動時間が長い傾向にあるため、この視点ではサーバーレスではないと言えます。ですが、起動時間を軽くすることに重きを置いて、小さなFunctionの集合という新しいやり方を強制するFaaSに対して、コンテナ技術を活用することで旧来のソフトウェアをFaaS的にスケールアウトできるようにする「FastContainerアーキテクチャ」のような、「ちょうど良さ」を求める解決策も提案されています[注6]。

マイクロサービスアーキテクチャとの違いもよく議論されますが、こちらは変化を受け入れるために意志決定のサイズを小さくするという組織論からの視点であることに対し、FaaSはあくまで実装手法にすぎません。1つのマイクロサービスを単一のFunctionで実装することもあれば、複数のFunctionを組み合わせて実現することもあるでしょう。

「指揮者としてのサーバ」をなくそう=メッセージ駆動

一般的なシステムであれば、クライアントアプリからのリクエストを受け付けたサーバーアプリが、そのレスポンスを返すところまでの機能全体に責任を持ちます。

たとえばWebの掲示板であれば、ブラウザから投稿する内容をHTTPリクエストとして受け付けたサーバーアプリのプロセスが、HTTP Cookieの内容をもとにユーザを識別、その投稿内容をデータベースに保存し、最新の投稿結果をあらためてデータベースから取得して、その結果をHTMLやAPIのレスポンスとして返します。ここには処理全体の「指揮者」としてサーバが存在します。

メッセージ駆動で処理を組み立て、このサーバという役割そのものをなくそうというのが3つめのサーバーレスアーキテクチャの考え方[注7]です。

◉コンポーネント同士をつないでいく

最近のクラウド基盤は、Functional SaaSなどとも呼ばれる高機能なコンポーネント[注8]を数多く提供しているため、それらをパズルのように組み合わせることで1つのシステムを組み上げることができます。コンポーネント間をつなぐメッセージの転送・変形処理は、前述のFaaSで実装するのがちょうど良く、それらの活用例がFaaS(とくにAWS Lambda)を中心にすえた実装として紹介されました。その結果、サーバーレスなFaaSを使って、高機能なコンポーネントをピタゴラスイッチのように組み合わせてシステムを実現する手法全体のことを、広義での「サーバーレスアーキテクチャ」と呼ぶようになりました。これが本稿で紹介する定義の3つめであり、現在盛り上がっている技術的流行の核となっているものです。

これは「餅は餅屋」戦略の到達点と言えます。これまでソフトウェアとしてすべて自前で実装していた機能を、それを得意とする外部のコンポーネントに切り出していった結果、外部コンポーネントでほとんどの機能が完結し、自分のソフトウェアはそれを「のり付け」するだけで良

注4) URL https://twitter.com/adrianco/status/736553530689998848
注5) 別の定義として、AWSのre:Invent 2016で紹介された「The serverless compute manifesto」があります。
URL https://youtu.be/yCOgc3MRUrs?t=294
注6) 従来のアプリも、リクエストハンドラ関数で呼び出せるという意味では「重いFaaS」と言えなくもありません。
注7) この考え方を、独裁的な指揮者を持つ「オーケストレーション」に対する「コレオグラフィー」と分類することがあります。
注8) これらのFunctional SaaSは直接クライアントアプリなどから利用することもでき、BaaSとの境界は視点の違いに過ぎません。本稿では、おもにフロントエンド(クライアントアプリ)から直接利用されるものをBaaS、おもにほかのコンポーネントと組み合わされることで価値を生むものをFunctional SaaSと呼び分けています。

くなったわけです。

いわばこれは、呼び出される側から呼び出す側へというソフトウェア開発のパラダイムシフトです。ソフトウェア開発に携わる人であれば、DI(Dependency Injection：依存性の注入)による制御の逆転が連想されるかもしれません。あるいは、良くできた単機能なツールをパイプで組み合わせるという意味では、UNIX哲学に通じるものがあります。さらに言えば、外部コンポーネントのインターフェースがより最適化されることにより、「のり付けするFaaSすら不要となる」場合も増えていくでしょう。

◉リアクティブシステム

メッセージ駆動でシステムを構築するときの考え方として、「リアクティブシステム」という設計原則がまとめられています[注9]。システムを複数のコンポーネントに分割して非同期なメッセージでつなぐこと(メッセージ駆動)で、コンポーネントに障害があればそのコンポーネントごと再起動してメッセージを再処理でき(耐障害性)、メッセージが溢れたらコンポーネントの並列実行数を増やして捌ききる(弾力性)、その結果としてユーザの期待を満たして速やかに反応できる(即応性)というものです。

メッセージ駆動を軸とするサーバーレスアーキテクチャの考え方は、情報科学分野で古くからErlang/OTPやAkkaのアクターモデルとして整理されてきたリアクティブシステムが、クラウドやFaaSを背景にしてベストプラクティスとして再発見されたものといえます。

サーバーレスアーキテクチャの活用

最後に紹介したFaaSとクラウドコンポーネントを組み合わせたサーバーレスアーキテクチャの事例には、いくつかのパターンがあります。

◉単純なAPIやWebアプリをFaaSで実装

制約が多いプログラミングモデルを受け入れる代わりに、さまざまなメリットを得ることができます。また現在の課金体系では、実行時間の課金になることで大幅なコスト削減になることが多いようです。

◉各コンポーネントのイベント処理

必要なときだけ起動するFaaSは、たとえばAmazon S3にファイルがアップロードされたら起動してサムネイル画像を追加で生成する、といったイベント駆動の処理に向いています。この分野ではAmazonが大きく先行し、本当に多種多様な事例が存在します。とくに監視や運用の分野で、何らかの閾値(しきいち)に基づいた自動化などが容易に実現しやすくなりました。

◉ストリーミング処理

リアクティブシステムとしての即応性により、ログの集計やIoT、あるいはLINEやSlackのbotなど、リアルタイムで流れてくるイベントをストリーミング処理していくようなケースに向いています。性能をパラメータだけで容易にスケーリング(いわゆる「札束ビンタ」)できるのが大きなメリットです。とくにKinesis AnalyticsやAzure Stream Analyticsなど、専門のストリーム処理エンジンを併せて使うことで、バッチに加えてリアルタイムに集計を実現するラムダアーキテクチャのような複雑なシステムも容易に実現ができるようになりました。

◉APIマネジメント

FaaSと組み合わせて活用されるようになったものが、APIマネジメントという機能です。FaaSの前にHTTPプロキシなどの形で介在し、認証処理やレート制限、複数のFaaSの統合、ドキュメント提供など、APIに必要な機能を一般化して提供します。SD

注9)「リアクティブ宣言」URL http://www.reactivemanifesto.org/ja

Part 2

[実践] AWS Lambda

Author 永田 明(ながた あきら) ㈱サーバーワークス クラウドインテグレーション部 技術4課
URL http://www.slideshare.net/nagaaki46 Twitter @nagaaki46

本稿ではAWS Lambdaを利用して簡単な動的Webサイトを構築します。サーバーレスなWebサービス基盤とはどういうものかを体験してみてください。

AWSにおけるサーバーレスアーキテクチャ

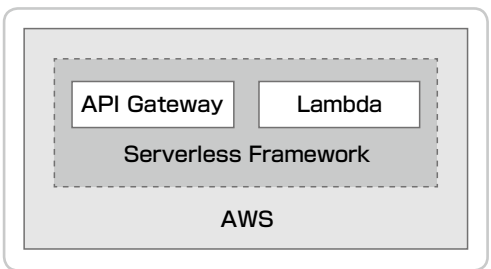
はじめに

従来Amazon Web Services(AWS)では、Simple Storage Service(S3)と呼ばれるオンラインファイルストレージサービスに静的ページ(HTMLやJavaScriptなど)を配置・公開することで、サーバーレスWebサイト環境を実現することができました。JavaScript用のAWS-SDKも提供されており、JavaScriptでAWSのAPIを操作することも可能ではありましたが、S3だけでサービスを実現するには、用途が限定的でした。

しかし、2014年11月に発表されたサーバーレスアプリケーションの実行基盤「AWS Lambda」(以下、Lambda)を皮切りに、さまざまなユースケースでサーバーレスアーキテクチャが実現可能となりました。

ここでは最初にAWSでサーバーレスアーキテクチャを構成する例として、本稿で構築する**図1**の各技術要素について紹介します。1つめはアプリケーションロジックの実行基盤となるLambda、2つめはWeb APIとしてHTTPリクエストを受け付けるAPI Gateway、3つめはLambdaとAPI Gatewayをより容易に開発するためのServerless Frameworkです。

▼図1 本稿の構成イメージ



AWS Lambdaとは

Lambdaは、AWSにおけるサーバーレスアプリケーションの実行基盤です。昨今のサーバーレスアーキテクチャの火付け役となったサービスと言えます。次の説明文は、Lambda公式ページ[注1]からの引用です。

> *AWS Lambdaを使用すれば、サーバーのプロビジョニングや管理なしでコードを実行できます。課金は実際に使用したコンピューティング時間に対してのみ発生し、コードが実行されていないときには料金も発生しません。Lambdaを使用すれば、実質どのようなタイプのアプリケーションやバックエンドサービスでも管理を必要とせずに実行できます。コードさえアップロードすれば、高可用性を実現しながらコードを実行およびスケーリングするために必要なことは、すべてLambdaにより行われます。*

Lambdaはシンプルなバッチ処理だけでなく、後述のAmazon API Gateway(以下、API Gateway)を併用することで、完全にサーバーレスな動的Webアプリケーションを構築することもできます。近年AWSでは、Lambdaの汎用性を活かし、

注1) URL https://aws.amazon.com/jp/lambda/

他のAWSサービスにもLambdaを組み込めるアーキテクチャを採用しています。Lambdaは、AWS基盤の一部になりつつあります。

Amazon API Gatewayとは

API Gatewayは、LambdaやWebアプリケーションをWeb API化できるサービスです。Lambda単体でも、サーバーレスアプリケーションとして、シンプルなバッチ処理を実行することは可能です。しかし、Webアプリケーションとして動作させるには、API Gatewayも必要となります。次の説明文は、API Gateway公式ページ[注2]からの引用です。

> *Amazon API Gatewayは全面的に管理されたサービスで、開発者はこれを利用することにより、どのようなスケールであっても、簡単にAPIの作成、配布、保守、監視、保護が行えます。AWSマネジメントコンソールで数回クリックするだけで、Amazon Elastic Compute Cloud(EC2)で稼働中のワークロードや、AWS Lambdaで稼働中のコード、または任意のウェブアプリケーションといった、バックエンドからのデータ、ビジネスロジック、機能にアクセスする、アプリケーションの玄関として振る舞うAPIを作成できます。Amazon API Gatewayでは、最大で数十万の同時API呼び出しを受け付け、処理することが可能です。これにはトラフィック管理、認証とアクセス管理、モニタリング、APIバージョン管理が含まれます。Amazon API Gatewayでは最低利用料や初期費用はかかりません。受信したAPI呼び出しと、送出したデータ量に対してのみ料金が発生します。*

便利なAPI Gatewayですが、プライベートネットワークに限定した利用ができない、リクエストのタイムアウトが最大約30秒しかない、などの制約はあります。そういったことが問題にならなければ、従来のサーバのようにスケールさせることは意識不要ですし、もちろん基盤のメンテナンス行為からも解放され、本来のサービス開発・運用に集中することが可能です。

Serverless Frameworkとは

筆者の業務において、API GatewayとLambdaの組み合わせでBtoC向けWebサービスをリリースしたことがあります。API GatewayとLambdaは別々のサービスであるため、一般的なWebアプリケーションのようにシームレスな開発が困難でした。とくに、API Gatewayの実装は各種定義となり、アプリケーションコードやコンフィグファイルとして管理できず、開発・運用は容易ではありませんでした。それらの課題を解決してくれるのが、各種サードパーティが公開しているサーバーレスアーキテクチャ専用のフレームワークです。

フレームワークにより、本来別々のサービスであるAPI GatewayとLambdaを、あたかも1つのサービスであるかのようにシームレスに開発することができます。また、実装アーキテクチャ、デプロイ方法なども、フレームワークの仕様で統一させることができます。今では、AWSのサーバーレスアーキテクチャにおいても、フレームワークの適用が一般的になりつつあります。

選択肢はいくつかあるのですが、中でもServerless Framework[注3]はメジャーなものの1つで、Serverless, Inc.が提供するオープンソースのフレームワークです。本稿では最終的に、Serverless Frameworkを適用した簡易なサンプルWebアプリケーションを開発してみます。

Lambdaを体験してみよう

サンプルWebアプリケーションを作成してみる前に、まずはLambdaを体験してみましょう。Lambdaは、ブラウザ上のAWSマネジメ

注2） URL https://aws.amazon.com/jp/api-gateway/
注3） URL https://serverless.com/

ントコンソールから登録し、短いプログラムであれば簡単に実行できます。

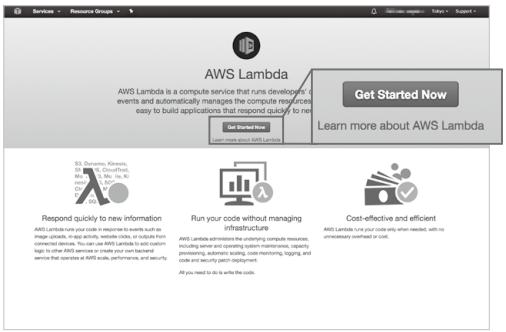
AWSマネジメントコンソールにログインし、Lambdaを選択します[注4]。初めてLambdaを設定する場合は、**図2**の画面が表示されます。中央のボタン[Get Started Now]を押下すると、blueprint(青写真)と呼ばれるLambdaのプログラムコードの雛形を選択する画面へ遷移します(**図3**)。利用予定のRuntime(言語環境)ごとにblueprintを絞ることも可能です。今回は「Blank Function」を選択します。

図4のようにトリガーの設定画面へ遷移しました。ここでは、Lambdaを起動するさまざまなトリガーを設定できます。AWSではトリガーとなるさまざまなサービスとLambdaを連携できるというわけです。たとえば、API GatewayからLambdaを呼び出すWeb APIを作成する場合は、トリガーとしてAPI Gatewayを選択することになります。しかし、ここでは直接Lambda起動を試すだけなので、何も選ばずにボタン[Next]を押下します。

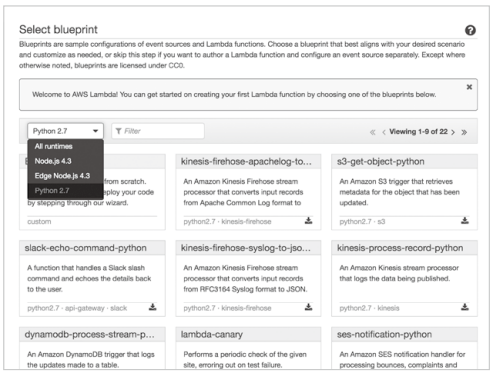
Lambda関数の設定画面へ遷移しました(**図5**)。Name値(Lambda関数名)と利用予定のRuntime値(言語環境)を最低限指定します。テキストエリアで実際のコード編集も可能ですが、今回はBlank Functionで提供されるコードをそのまま使ってみます。文字列'Hello from Lambda'を返すだけのシンプルなサンプルプログラムです。

引き続き下へスクロールし、Handler値(callされる関数名)とRole値(Lambdaに与えるAWSの実行権限)を設定します(**図6**)。今回Role値は、「Create new role from template(s)」を選択してみます。Roleの具体的な権限設定は今回はなしでかまいません。**図7**のように、Role name値だけ設定しておきます。そのほか

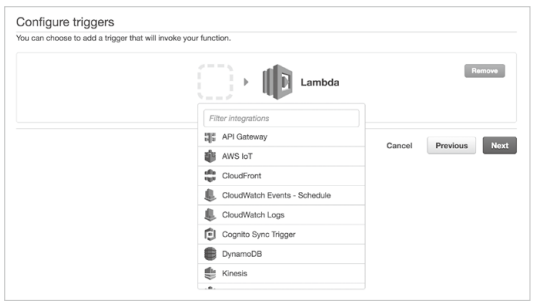
▼図2　Lambdaのサービス画面



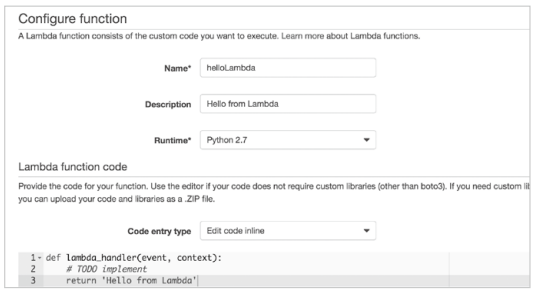
▼図3　blueprintの選択画面



▼図4　トリガーの設定画面



▼図5　Lambda関数の設定画面1



注4) AWSアカウントをお持ちでない読者の方は、Amazon Web Services(https://aws.amazon.com/jp/)の「サインアップ」から、新たにアカウントを取得してください。呼び出し回数やWebページ容量など、過度な利用でなければ課金はほとんど発生しませんが、念のため事前に料金体系(https://aws.amazon.com/jp/api-gateway/pricing/、https://aws.amazon.com/jp/lambda/pricing/、https://aws.amazon.com/jp/s3/pricing/)をご確認ください。

は初期値のまま、ボタン[Next]を押下します。

Review画面(**図8**)で今回設定した内容を確認し、ボタン[Create function]を押下します。

それでは作成したLambda関数を稼働確認してみましょう。**図9**の画面でボタン[Test]を押下します。開いた画面(**図10**)にあるSample event template欄で、Lambdaがサポートするイベントごとに擬似的にテストパラメータを渡して実行・テストすることが可能です。今回はイベントなしの'Hello World'を選択し、ボタン[Save and test]を押下します。このボタンによりテスト用のイベントデータが保存されると同時に、処理が実行されます。

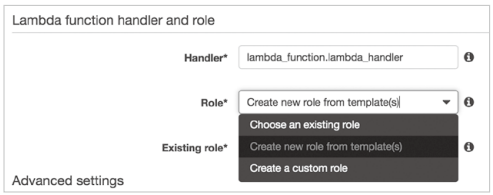
図11のExecution result欄に注目です。"succeeded"で処理は正常に成功しており、下段にはLambda関数が返した文字列'Hello from Lambda'が出力されていることがわかります。また、処理実行時間や使用された最大メモリ量も確認できます。

いかがでしょう。プログラムコードと最低限の設定だけで、簡単に処理を実行・テストすることができました。もちろん、実際にはもっと複雑な処理を担うプログラムコードが必要となり、連携するサービスも多岐にわたることでしょう。その複雑さを救ってくれるのが、後述のServerless Frameworkに代表されるフレームワークでもあります。

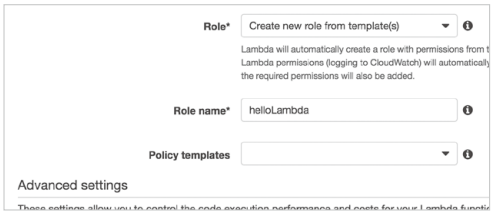
サンプルWebアプリケーションを作成しよう

それでは、Serverless Framework(API GatewayとLambda)と静的HTMLを組み合わせ、簡単な動的Webページを作成してみましょう。

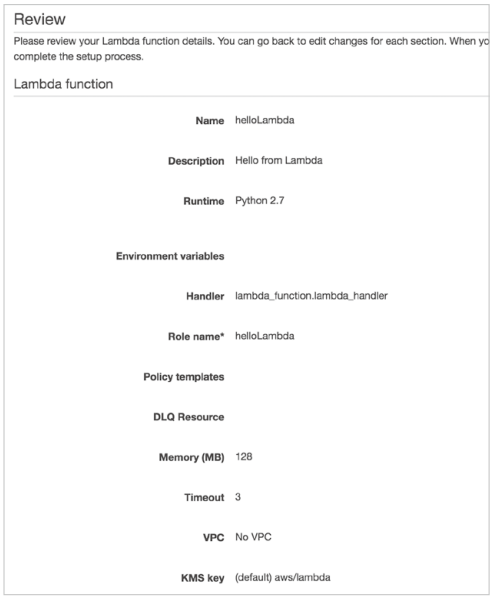
▼図6　Lambda関数の設定画面2



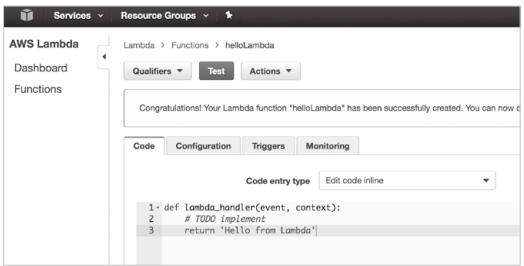
▼図7　Lambda関数の設定画面3



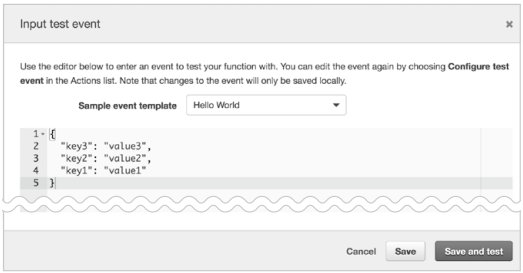
▼図8　Lambda関数の設定確認画面



▼図9　helloLambdaの画面



▼図10　Lambda関数の設定画面3



▼図11　実行されたテスト結果の確認

全体構成

静的HTMLの環境もサーバーレス構成とするため、本稿ではS3を利用します。構成は**図12**の想定です。作成のシナリオは次のとおりです。

①API GatewayとLambdaによるWeb API作成
②S3上の静的HTMLによるWeb画面作成

前提条件

今回のサンプルWebアプリケーションの前提条件は次のとおりです。

・開発端末：macOS Sierra
・開発言語：Python
・開発環境モジュール：Node.js v4.3.2、Serverless Framework 1.6.1（執筆時最新版）
・その他：
AWSアカウントを取得済みであること

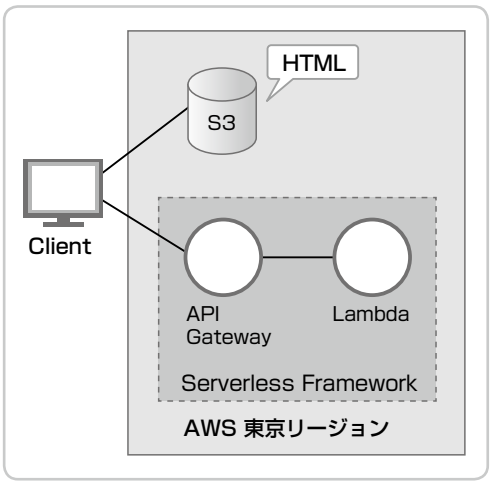
開発環境準備

まずは、開発端末にServerless Frameworkをインストールし、Web APIを開発できる環境を準備します。

◉Serverless Frameworkのインストール

今回の開発言語にはPythonを採用しますが、Serverless Framework自体はNode.jsを必要とします。Lambda実行環境仕様に合わせ、Node.js v4.3.2を事前にインストールしておきます。次のようにインストールコマンドを実行してください。

▼図12　サンプルWebアプリケーションの詳細構成イメージ




```
$ node -v
v4.3.2  ←Node.jsがインストール済み

$ npm install -g serverless
↑Serverless Frameworkのインストール
/Users/nagaaki46/.nodebrew/node/v4.3.2/bin/➥
serverless -> /Users/nagaaki46/.nodebrew/➥
node/v4.3.2/lib/node_modules/serverless/bin/➥
serverless
...略...
```

インストールされたServerless Frameworkのバージョンを確認してみましょう。

```
$ serverless --version
1.6.1
```

最新のServerless Framework 1.6.1が無事インストールされたようです。

実装

それでは、アプリケーションを作成していきましょう。Serverless Frameworkのscaffold機能で、アプリケーションプロジェクトの雛形(ひながた)を生成します。アプリケーション名は「helloServerless」として、**図13**のようにコマンドを実行してください。

生成が成功したようです。アプリケーション名と同じディレクトリ(helloServerless/)が生成されています。

```
$ ls helloServerless/
handler.py  serverless.yml
```

Serverless Frameworkのconfigファイルである

▼リスト1　serverless.yml(抜粋)

```
# Welcome to Serverless!
 ...略...
service: helloServerless
 ...略...
functions:
  hello:
    handler: handler.hello
     ★
#    The following are a few example events ↵
you can configure
#    NOTE: Please make sure to change your ↵
handler code to work with those events
#    Check the event documentation for details
#    events:
 ...略...
```

▼図13　アプリケーションプロジェクトの雛形を作成

```
$ serverless create --template aws-python --path helloServerless
Serverless: Generating boilerplate...
Serverless: Generating boilerplate in "/Users/nagaaki46/helloServerless"
 _______                             __
|   _   .-----.----.--.--.-----.----|  .-----.-----.-----.
|   |___|  -__|   _|  |  |  -__|   _|  |  -__|__ --|__ --|
|____   |_____|__|  \___/|_____|__| |__|_____|_____|_____|
|   |   |             The Serverless Application Framework
|       |                           serverless.com, v1.6.1
 -------'

Serverless: Successfully generated boilerplate for template: "aws-python"
```

るserverless.ymlをのぞいてみましょう(**リスト1**)。紙幅の都合で省略していますが、最低限の初期設定値と、コメントアウトされたサンプル設定値が記載されています。

今回はAPI Gatewayを利用したWeb APIを作成します。events要素のhttpの定義をhelloハンドラの下に追記します。また、後に作成するHTML内のJavaScriptから、異ドメインのAPI Gatewayを利用できるよう、CORS(Cross-Origin Resource Sharing)設定も有効化します。次の濃い黒色部分のコードを**リスト1**の★印に追記し、保存してください。

```
functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: api/hello
          method: get
          cors: true
```

これによりhelloハンドラは、HTTPイベント(つまり、API Gateway経由)で起動することができます。

次に、生成されたプログラムコードも見てみましょう(**リスト2**)。レスポンスボディとして固定の"Go Serverless v1.0! Your function executed successfully!"というテキストを含むJSONを返すサンプルコードです。今回はこのサンプルコードのまま利用しますが、CORS対応にしておくため、ヘッダ"Access-Control-Allow-Origin"を返すように、**リスト2**の★★印の個所をその右の吹き出し内のように書き換えます。

デプロイ

さて、早速デプロイを実施したいところですが、もう少し準備が必要です。Serverless Frameworkのデプロイコマンドの実行

▼リスト2　hello.py

```
import json

def hello(event, context):
    body = {
        "message": "Go Serverless v1.0! Your function executed successfully!",
        "input": event
    }

    response =
        "statusCode": 200,
        "body": json.dumps(body)
    }

    return response

    # Use this code if you don't use the http event with the LAMBDA-PROXY integration
    """
    return {
        "message": "Go Serverless v1.0! Your function executed successfully!",
        "event": event
    }
    """
```

★★

```
response = {
    "statusCode": 200,
    "headers": {
        "Access-Control-Allow-Origin": "*"
    },
    "body": json.dumps(body)
}
```

には、IAMユーザのアクセスキーが必要です。Serverless Frameworkのコマンドが、AWSのAPIを通して、アプリケーションモジュールのデプロイを行うためです。それでは、IAMユーザのアクセスキー発行を実行してみましょう。AWSマネジメントコンソールのメニューより、IAMのサービス画面へ遷移します(**図14**)。リンク[ユーザー]からユーザ設定画面へ遷移し、[ユーザーを追加]を押下して新規ユーザを作成します。

表示された画面(**図15**)で、ユーザー名欄には任意の名称を入力します。アクセスの種類は、今回は「プログラムによるアクセス」のみを選択します。画面右下のボタン[次のステップ：アクセス権限]を押下すると、作成するユーザのアクセス権限設定画面へ遷移します(**図16**)。

Serverless Frameworkの仕様では、"AdministratorAccess"を付与することが指定されています。**図16**の吹き出し内のように、フィルター欄で"AdministratorAccess"を検索し、チェックします。画面右下に現れるボタン[次のステップ：確認]を押下します。

▼図14　IAMサービス画面

▼図15　ユーザの追加設定1

Attention

「AdministratorAccess」はAWSの操作が無制限に実行可能な権限であり、一般には「AdministratorAccess」を付与するアカウントを作成することは非推奨とされています。以降で発行するアクセスキーが外部に漏れないようご注意ください。また、当チュートリアルが終わりましたら、作成したIAMアカウントを削除するなど必ず無効化してください。

ユーザ確認画面へ遷移するので、内容が正しければ右下の[ユーザーの作成]を押下します。

アクセスキーが発行されました(**図17**)。後から同じアクセスキーを取得することはできないので、ボタン[.csvのダウンロード]でアクセスキー情報を取得し大切に保管しておきます。最後に、ボタン[閉じる]を押下して終了します。

アクセスキーを開発端末に設定します。アクセスキーの設定方法はいくつかあるのですが、export方式を適用してみます。開発端末のターミナルから**図18**のコマンドを実行し、先ほどダウンロードしたcsvファイルにあるアクセスキーを環境変数に設定します。

それでは、東京リージョンにデプロイしてみましょう(**図19**)。Serverless Frameworkのdeployコマンドは、CloudFormation(AWSが提供するプロビジョニングサービス)経由で実行されます。詳細な流れを**図19**を見ながら追ってみましょう。まず、serverless.ymlの情報を元に、CloudFormation用のテンプレートファイルに自動変換されます。そしてソースコード群はデプロイモジュールとしてzip化され、デプロイ用のS3バケットにアップロードされます。

フレームワークを採用せずAPI GatewayやLambdaをそのまま利用すると、デプロイの手続きを開発者が設計、準備をする必要があります。また、API Gatewayはソースコードでの実装ではなく、各種設定で構成されるサービスになります。アプリケーション開発という観点からするとソースコードやコンフィグファイルによるバージョン管理ができないので悩ましいところですが、そこにもServerless Frameworkを使う利点が生まれます。Serverless Frameworkはserverless.ymlによって管理が可能なので、serverless.ymlを管理してさえいれば、deployコマンドで反映できるためです。

それでは、curlコマンドでサービスの稼働確認を実施してみます。**図19**の下から3行目で得られたエンドポイント値(https://……)を指定します。

```
$ curl -G https://0baxasi8lf.execute-api.
ap-northeast-1.amazonaws.com/production/api/
hello
{"input": {"body": null, "resource": "/api/
hello", ...略... "isBase64Encoded": false},
"message": "Go Serverless v1.0! Your function
 executed successfully!"}n
```

正しいレスポンス("message": "Go Serverless v1.0! Your function executed successfully!")が

▼図16　ユーザの追加設定2(アクセス権限)

▼図17　アクセスキーIDとシークレットアクセスキーの発行

▼図18　exportによるアクセスキーの設定

```
$ export AWS_ACCESS_KEY_ID=<取得したアクセスキーID>
$ export AWS_SECRET_ACCESS_KEY=<取得したシークレットアクセスキー>
```

▼図19　東京リージョンへのデプロイ

```
$ serverless deploy --stage production --region ap-northeast-1
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading service .zip file to S3 (1.27 KB)...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.........................
Serverless: Stack update finished...
Service Information
service: helloServerless
stage: production
region: ap-northeast-1
api keys:
  None
endpoints:
  GET - https://0baxasi8lf.execute-api.ap-northeast-1.amazonaws.com/production/api/hello
functions:                                          ↑エンドポイント値が取得できた
  helloServerless-production-hello
```

返ってくることが確認できました。

Webページの配置

次に、Web画面からもWeb APIを呼び出せるよう、**リスト3**の静的HTMLを新たに用意します。

作成したhello.htmlをWebブラウザで開くと、デプロイ済みのAPI Gatewayのhelloサービスへ GETリクエストが送られます。取得した文字列が画面に表示されれば呼び出し成功です(**図20**)。

さあ、仕上げとして、作成した静的HTMLもサーバーレス環境でWebホスティングしてみましょう。静的HTMLをS3にアップロードし、静的Webサイトホスティング設定を施すだけで実現が可能です。具体的な設定は、AWSの公式ドキュメント「静的ウェブサイトホスティング用にS3バケットを設定する方法[注5]」を参照ください。

以上で、完全にサーバーレスな環境で、動的Webサイトを構築することができました。

なお、今回は対象としませんが、前段にCDNサービスであるAmazon CloudFront[注6]を構築する構成もあります(**図21**)。実は、API Gatewayのしくみも CloudFrontで構成されているのですが、隠蔽されているため簡易な機能しか利用することができません。前段にCloudFrontを用いる構成で実現できることは、独自ドメインでのSSL通信、S3上のコンテンツのキャッシュ、(AWS WAFおよびAPI GatewayのAPIキー認証との併用による)API GatewayのIP制限などがあります。ただし、CloudFrontを経由することで、AWS利用料が増加する可能性があること、Webサービスのレイテンシは大きくなることのトレードオフがあります。

まとめ

いかがでしたか。実際の本番Webサービスを実装するとなると、プログラムのコードやクラス、利用するライブラリももちろん増えます。また、本番運用を考慮したパラメータ設定、監視のしくみなど、API Gateway・Lambdaならではの注意点もたくさんあるのですが、本稿ではここまでとします。

注5) URL http://docs.aws.amazon.com/ja_jp/AmazonS3/latest/user-guide/static-website-hosting.html
注6) URL https://aws.amazon.com/jp/cloudfront/

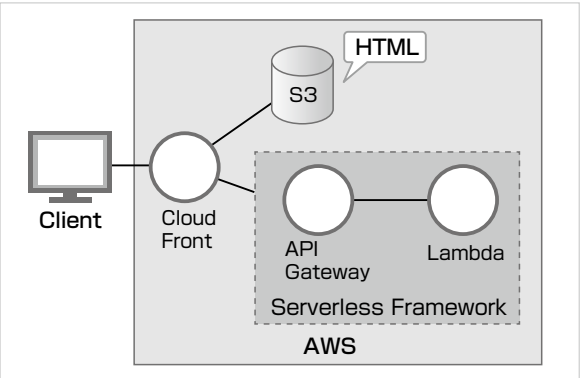
▼リスト3　hello.html

```
<!doctype html>
<html>
  <head>
    <title>Hello Serverless</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-md-7 col-md-offset-2">
          <h1>Hello Serverless</h1>
          <div>
            <textarea id="output" class="form-control" rows="10"></textarea>
          </div>
        </div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <script>
      (function() {
        $.ajax({
          type: 'GET',
          url: 'https://0baxasi8lf.execute-api.ap-northeast-1.amazonaws.com/production/api/hello',
          success: function(data) {                    ↑下線部分には取得したエンドポイント値を入れる
            $("#output").val(data.message);
          },
          error: function(xmlHttpRequest, textStatus) {
            $("#output").val('failed.');
          }
        });
      })();
    </script>
  </body>
</html>
```

サーバーレスというジャンルは、本稿で紹介したようなフレームワークも含め、今後まだまだ進化していきます。枯れていないサービスを採用することを敬遠してしまうケースも多いとは思います。常に新しい技術仕様に追従していくのは大変ですが、サーバーレスにおいては実行基盤の管理から完全に解放されることで、運用フェーズで絶大な効果を発揮します。今回の記事をきっかけに、積極的にサーバーレスのサービスを活用いただければ幸いです。SD

▼図20　hello.html実行結果

▼図21　CloudFrontを利用したサーバーレスWebアプリケーションの構成例



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

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

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

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

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

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

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

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

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

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

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1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

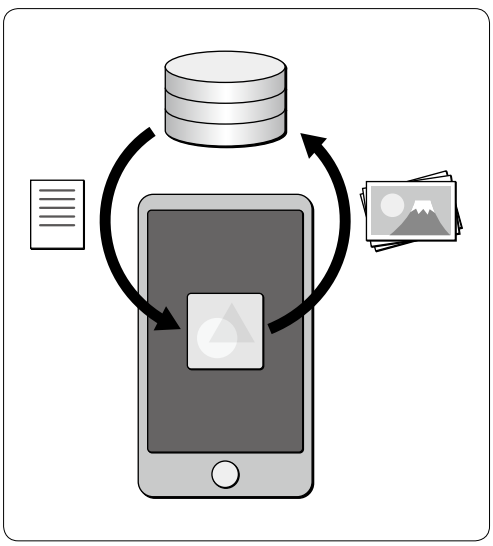
Part1 ニフティクラウド mobile backendとは何か?

● Author 中津川 篤司(なかつがわ あつし)

Before iPhone、After iPhone

iPhoneが世の中に登場して早くも10年経とうとしています。iPhone以前の世界が信じられなくなってしまっている人も多いでしょう。また、AppleやGoogleが提供しているアプリストアによって、これまでと生活が一変してしまった開発者も多いはずです。少なくとも10年前にはアプリ開発者なんて言葉は存在しなかったわけで、そのアプリストアが多くの開発者の需要を生み出したと言えるでしょう。

▼図1 アプリのほとんどがデータをサーバと送受信している



iPhoneにおいては登場した時点ではアプリストアはありませんでした。初めてできたのは2008年6月です。そこでやっと開発者にその門戸が開かれました。しかし、その前からJail Breakとして知られているハックによって独自のアプリストア(インストーラ)が存在しました。何のドキュメントも存在しなかったときから、試行錯誤のうえでアプリが開発できる環境が作られていました。そのようにアプリ開発はプログラマを魅了し続けたのです。

スマホアプリのネットワーク依存

スマートフォンアプリは実に多岐に渡りますが、その共通の特徴として「ネットワークを利用する」という点が挙げられます。ネットワークを利用していないアプリを探すほうが難しいかもしれません。恐らく電卓や水平器、時計、マルバツゲームくらいではないでしょうか。スマートフォンはもともとインターネットアクセスできる点に魅力があったわけで、ネットワー

クの存在は切っても切り離せないものと言えます(**図1**)。

ではそのネットワークはどのように使われているのでしょうか。たとえばアプリ内のデータをサーバ上に保存するといった使い方があります。これはバックアップ、ほかのデバイスとの同期、デスクトップなどのブラウザからもアクセスするためといった目的が考えられます。たとえばEvernote、Dropbox、Googleドキュメント、Keynote、メモアプリなどほとんどのサービスがそういった目的でデータを保存します。

さらにコミュニケーションの手段として使うものがあります。LINE、Facebook、Instagram、Twitterなどの各種ソーシャルネットワーク系アプリはその最たるものでしょう。この場合、自分のプライバシーは守りつつも、データをほかの人と共有する目的でアップロードします。そしてソーシャル系アプリではプッシュ通知がよく使われます。友達がサービスにジョインした、コメントした、いいねしたといったアクションを通知して積極的な参加を促しています。

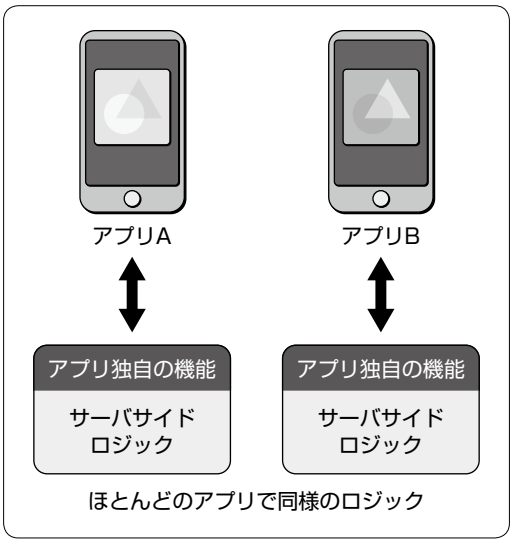
ほかにもゲームであれば自分のスコアを保存して世界中のユーザとランキングを競い合ったり、友人と一緒に戦うためにステータスデータを利用します。そして、そうしたデータの保存に対してキーになるのが認証になります。ID、パスワードを使ったシンプルなものから、各種ソーシャルサービスを使った認証技術もあります。

アプリは多種多様に存在しますが、このようにネットワークの利用法や目的といった観点で区切ると実はそれほど多くのパターンはないことに気づくでしょう。つまりアプリをたくさん作ってきた開発者にすると、新しいアプリを作るたびに「またこれを作るのか」と思ってしまうようになります。

繰り返される似たような開発からの発展的脱出方法

この傾向自体はWebシステムを構築するときにも大差はありませんでした。その結果、各種フレームワークが出てきたことで開発コストを下げてきた歴史があります。しかし、Webシステムはサーバのコードが HTMLを生成する形を取ってきたので、ロジックとUIの分離が進まず、結果としてロジックは似たようなものながらもUI部分で差別化を図ってきました(現在はJavaScriptフレームワークも進化しているので似たような傾向になっていますが)。しかしアプリとサーバとではRESTful APIを使ってJSONまたはXMLでデータの送受信を行うのが通例になってきています(**図2**)。UIがなければもはや毎度同じことの繰り返しに見えてしまうでしょう。

▼図2 アプリにおけるサーバサイドのロジックはほとんど共通している



同じものを何度も作るほど開発者を退屈させるものはありません。もっと違う部分(アプリのロジックやUIなど)を開発したいと思うはずです。そんな開発者の思いが形になったのがmBaaS(mobile Backend as a Service)です。

mBaaSって、どんなサービスなの？

mBaaSはその名のとおり、モバイルアプリのバックエンド(サーバサイド)をサービス化したものです。つまりこれまでに挙げた、アプリ

▼図3　ニフティクラウド mobile backend のおもな機能

開発でよく望まれる機能(認証、データ保存、ファイル保存、プッシュ通知など)をクラウドベースで提供しています。最近ではモバイルに限らず、IoTなどでも使われるのでmを外してBaaSと呼ばれたりします。

ある日突然、Parse.comが終了

このmBaaSを世に知らしめたのはParse.comというサービスになります。Parse.comは多くのアプリを生み出す切っ掛けになったと言えるでしょう。2013年にFacebookによって買収され、2016年2月に突然サービスの終了をアナウンスしました(2017年1月末でサービスは完全に終了)。これは同じmBaaSを提供する事業者に衝撃を与えましたが、Facebookの目指している世界とずれが出てしまったためと思われます。mBaaS自体は今後も引き続き必要なサービスでしょう。

ニフティクラウド mobile backend(以下NCMB)もmBaaSの1つになります。運営母体はニフティ㈱で、自社のクラウドサービスであるニフティクラウドの上に構築されています。主な機能は前述のとおりですが、ほかにも位置情報検索や、サーバ側にロジックを保存できるスクリプトという機能があります(**図3**)。

ACLとNCMB

データはACL(Access Control List)で参照、更新および削除が管理できるようになっています。ACLについては個人、またはグループで管理できるようになっています。一般的にデータのアクセス権限を管理するのは面倒で複雑なロジックが必要になりますが、mBaaSを使うことによって開発者はいっさいの手間から解放されます。

さらに認証にしても同様です。パスワードの暗号化、セッション管理などは面倒です。さらにソーシャルサービスを使ったログインも実装しないとアプリ利用者が増えないかもしれません。ソーシャルサービスではOAuth2を使った認証が基本で、そのための実装コストは小さくありません。

NCMBで仕事が楽になる

位置情報検索はスマートフォンやタブレットならではの位置情報を使ったアプリを作る際によく使われます。チェックインアプリを作ったり、レストラン検索、屋外イベントなど多くの場面で位置情報が使われています。何より位置情報はユーザが手で入力することなく、自分の身の回りにある情報を検索できる手軽な手段です。リアルにあるデータを扱う場合には使わない手はないでしょう。

そして写真などのバイナリデータを保存しておく機能もあります。データ、ファイルの保存、認証、検索、プッシュ通知といった機能が1つのSDK(ソフトウェア開発キット)をインストールするだけで共通のインターフェースを使って操作できるようになるのがmBaaSの魅力と言えます。個別にバラバラなライブラリをインストールしている場合、扱いが面倒ですし、共通で使っているライブラリがコンフリクトする恐れもあります。

mBaaSを使うことによって、開発者はアプリ開発における面倒ごとから解放され、差別化要因(アプリのUI/UX)の開発に集中できるようになります。

mBaaSを使った開発について

NCMBではSDKを提供しています。対応しているのはiOS(Objective-C)/Android(Java)/

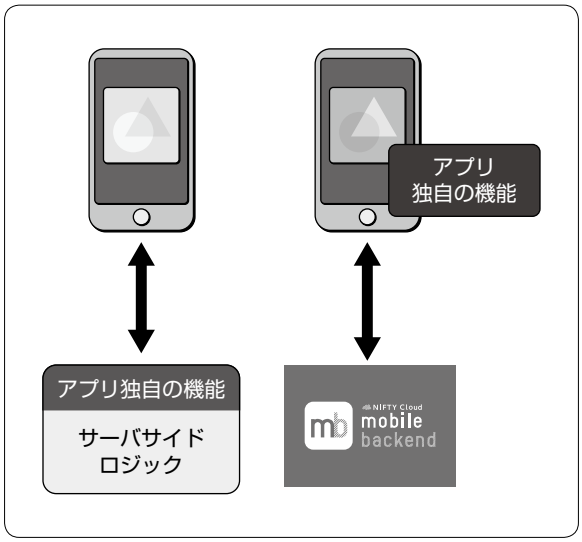
Unity/JavaScriptです。JavaScriptはWebブラウザでも使えますが、おもにMonaca/Cordova向けのSDKになります。MonacaはHTML5/JavaScriptを使ってスマートフォンアプリを開発できる、ハイブリッドアプリを作るサービスです。もちろんJavaScriptなのでNode.jsを使ってサーバサイドや、Raspberry Pi上で動かすこともできます。SDKはすでにオープンソースで開発されていて、GitHub上で公開されています。また、一部の言語(Ruby/PHP/Python/Xojo)ではサードパーティー製のSDKも開発されています。これらもオープンソースです。

これらのSDKを使えば、認証に用いるアプリケーションキー、クライアントキーをセットするだけですぐに前述のデータ保存やバイナリデータの保存、認証、プッシュ通知といったような機能が使えるようになります。NCMBはRESTful APIを提供しており、各SDKはこれを使ってデータの操作を行っています。インターフェースはシンプルなCRUD(Create：生成、Read：読み取り、Update：更新、Delete：削除)操作を行うものなので、他の言語での実装もそれほど難しくありません。そのため、足りない機能や、自社の基幹システム連携部分を自作しているという話もよく聞きます。とは言え、多くのプラットフォームにおいては提供されているSDKを利用することでWeb APIの認証処理やネットワークなどを意識することなく利用できます。

従来のアプリ開発との違い

mBaaSを使わない従来の開発と、使った場合の開発スタイルの違いを図で表すと**図4**のようになります。従来の開発の場合、サーバ側にはよくあるロジック(データ保存や認証など)と、アプリ独自の機能が一緒になっています。そしてmBaaSを使った開発の場合、サーバサイドの大部分のロジックはなくなります。そしてアプリ独自の機能はサーバ側ではなく、アプリ側に実装することになります。このようにして開発負荷を大幅に低減できるのがmBaaSの魅力です。なお、NCMBにおいてはスクリプトという機能によってアプリ独自の機能をサーバに保存することもできます。スクリプト機能を使えばさらにアプリ側のコード量を減らせるようになります。

▼図4　従来のアプリ開発とNCMBを使った開発の違い



現在のSDKはバージョン2系で、一度大幅な刷新とオープンソース公開を行っています。なるべくモダンな開発スタイルに合わせられるAPIにしたり、バージョンアップによって生じたインターフェースの違いを吸収したりしました。そのため1系と2系では互換性はありませんが、より現代風のコーディングスタイルが使えるようになっています。これはSDKの環境ごとに分かれており、Objective-C/Java/Unity/JavaScriptでそれぞれAPIが違います。その環境ごとのモダンな書き方、使いやすさに合わせています。SDKを扱う開発者としては、こうした視点は大きな問題になりますので、それぞれのプラットフォームを扱う開発者の方が使いやすい形にしています。

mBaaS自体としては年々扱うデータ量が増加しており、当初の予定を大幅に超える規模に

▼表1　各クラウドサービスの比較

名称	サービス内容	カスタマイズ性
IaaS	サーバを提供	高い
PaaS	動作・実行環境を提供	中間
mBaaS	バックエンドを提供	中間
SaaS	サービスを提供	低い

成長しています。そのような中とあって、サーバサイドのアーキテクチャは何度か大きな改変を行っています。Web APIは変えていませんので利用している開発者の方が気づくことはほぼありませんが、レスポンスを良くするための努力や、安定性を増すための努力は継続して行っています。とは言え、mBaaSは利用者全体でリソースを共有したサービスになります。クラウドサーバのように自分たちが占有できるリソースを確保して利用するものではありません。そのため1つのアプリが過剰な負荷を与えるような処理を実行するとほかのユーザに影響を与えてしまいます。そうならないためのアクセスコントロールが必須になります。

mBaaSとほかのクラウドサービスの違い

世の中にはたくさんのクラウドサービスが存在します。そのベースにあるのがIaaSと呼ばれるもので、クラウドサーバのサービスです。管理画面やAPIで実行すると、クラウド上にサーバが立ち上がります。筆者たちのサービスではニフティクラウドがこれに相当します。そしてそのうえで展開されるサービスがSaaSと呼ばれます。有名なところとしてはGoogle AppsやSalesforceなどが挙げられます。メーラーやオフィススイート、CRMといったサービスをクラウドベースで提供するものです。PaaS(Platform as a Service)と呼ばれる動作・実行環境を提供するサービスもあります(**表1**)。

mBaaSもこのPaaSの1つと言えますが、よりモバイルアプリ開発に特化した機能を提供しています。IaaSほどの汎用性はありませんが、機能を絞り込んだ分、安価ですぐに使えるサービスとしてパッケージされています。IaaSを使って自分でmBaaSを建てることもできます(オープンソースのmBaaSも各種存在します)が、サーバのメンテナンスやセキュリティアップデート、ソフトウェアのバージョン管理などの面倒ごとも背負い込むことになります。mBaaSを使っている限りはそうしたことを気にせずに済みます。

NCMBを使うと、これまでアプリを開発するときに感じていた面倒さが一気に解消されます。ぜひNCMBと一緒にアプリ開発を始めてみませんか?

Part2 環境センサーとNCMBを組み合わせる——オムロンとバイタリフィの事例

● Author 小山 哲志(こやま てつじ)

実際の開発はどうか——バイタリフィの取り組み

NCMBは、スマホアプリに有用な機能をピックアップしてAPIサービスとして提供しています。実際に使用されている代表例としては、スマホ向けソーシャルゲームやウェブサービスをアプリ化したものが多いのですが、本稿では少し変わった例としてオムロン㈱(以下オムロン)が開発した環境センサーを紹介します(実際に開発作業を行った㈱バイタリフィ[注1]の石川貴文氏(**写真1**)へのインタビューをもとに構成しています)。

オムロンの環境センサー

近年ではその単語を聞かない日はないというくらいにバズワードとなっているのが「IoT」で

注1) http://vitalify.asia/

▼写真1 ㈱バイタリフィ・石川 貴文氏

す。これまで独立して動作していたさまざまな機器やセンサーをネットワークに接続して、データをネット上にアップロードして集中的に管理解析したり、逆にリモートから機器をコントロールしたりします。

オムロンは家庭用健康器具などで有名なように、一般向けのデバイスを多数販売している会社です。健康器具としてはすでに多数のインターネット連携した製品が出されていますが、オムロン全体の製品としてみるとまだまだネットとの連携余地は多くある状況です。そこでIoTが盛り上がっている現状に合わせて、自分たちが強い分野であるセンサーから吐き出される膨大なデータを活用した、新たなビジネスを立ち上げる企画が生まれました。

そこで開発されたのが「環境センサー[注2]」です（**写真2**）。この製品は約4cm × 5cmで16gの本体に、温度、湿度、照度、気圧、音、UVという6つのセンサーを内蔵し、電池で約6ヵ月稼働するというものです。通信路としてはBluetooh Low Energy（BLE）を搭載していますが、それ単体でインターネットの接続機能やTCP/IPスタックは持っていません。そのため何らかの中継機器がデータをインターネット上にアップロードする必要があります。

センサーデータの保管場所としてNCMBを運用するニフティに相談が来たのが2015年9月のことです。インターネットへ中継する役割はスマホに任せると同時に、データをグラフィカルに表示するアプリを開発することで、ユーザの利便性を高めることが目的とされました。

環境センサーからBLEで取得したデータをクラウドにアップロードしつつ、同時にユーザへのデータ表示UIも提供するには、高度な並列処理が必要です。そのため当然ながらiOS、Androidそれぞれにネイティブアプリを開発する必要がありました。そこで開発を担当したのが、㈱バイタリフィです。バイタリフィは、IoTなどの特殊技術に特化したアプリ開発プロダクションです。日本とベトナムに拠点を持ち、ベトナムオフショア開発なども組み合わせた柔軟な開発を得意としていました。オムロンとしても、ノウハウの少ないモバイル向けIoTアプリケーションを、短納期かつ開発費用を極力抑えて実現するという状況での相談ということでしたが、2016年4月に要求仕様をほぼクリアして無事に開発完了しました。

2015年は仕様の策定に費やされたので、実質的な開発期間は2016年1月後半からの約2ヵ月です。開発メンバーの構成は次のようになります。

・iOSエンジニア3名
・Androidエンジニア3名

▼写真2 オムロンの環境センサー

注2） http://www.omron.co.jp/ecb/products/sensor/special/environmentsensor/

▼リスト1　パスワードによるログイン部分のコード（iOS版）

```
- (void) login:(NSString*) username password:(NSString*) pass onSuccess:(void(^)(NCMBObject* ➥
object, NSError* error))success onFail:(void(^)(NSException* ex)) fail{
    @try {
        [NCMBUser logInWithUsernameInBackground:username password:pass block:^(NCMBUser *user, ➥
NSError *error) {
            BOOL flag = NO;
            if(error == nil){
                [DBManager  setIsLogin:YES];
                NSArray *userList = [DBManager select:[DBUser class]];
                if(userList.count > 0){
                    DBUser *newUser = [NCMBHelper dbObjectFromCloudObject:[DBUser class] ➥
cloudObject:user];
                    newUser.password = [OMUtilities encodeAES:pass];
                    DBUser *dbUser = [userList objectAtIndex:0];
                    newUser.sessionToken = user.sessionToken;
                    if([dbUser.objectId isEqualToString:[user objectId]]){
                        //Update
                        flag = [DBManager update:newUser];
                    } else {
                        //Delete and create new
                        [[DBManager getInstance] cleanAllData];
                        flag = [DBManager insert:newUser];
                    }

                } else {
                    //Add new
                    DBUser *newUser = [NCMBHelper dbObjectFromCloudObject:[DBUser class] ➥
cloudObject:user];
                    newUser.password =  [OMUtilities encodeAES:pass];
                    newUser.sessionToken = user.sessionToken;
                    flag = [DBManager insert:newUser];
                }
            }
            success(user, error);

        }];
    }
    @catch (NSException *exception) {
        fail(exception);
    }
    @finally {}
}
```

▼図5　環境センサーアプリのスクリーンショット

・テスト2名
・ディレクター、アシスタント各1名

オムロンの環境センサー側の開発も同時並行して進んでいることを考えると、驚異的な短納期と言えます。

リリースされた環境センサーとアプリ(**図5**)は、実際にさまざまな企業に導入されていて、すでに大量に出荷されているそうです。興味のある方はニフティが提供する「IoTトライアルキット[注3]」で入手できます(有償)。

環境センサーアプリ開発の実際

次は環境センサーアプリが使用しているNCMBの機能について解説します。

機能概要

環境センサーアプリの機能概要をあらためて表記します(**図6**)。

・パスワード／メールアドレスによるユーザ登録、ログイン
・環境センサー本体とBluetoothで通信を行い、各センサーのデータを取得する
・取得した計測データをアプリ画面に表示する
・取得したデータをNCMBにアップロードする
・NCMBに保存した各センサーデータをグラフ表示する
・プッシュ通知
・環境センサーの設定
・環境センサーのファームウェアアップデート

ユーザ管理

ユーザ登録やログインなどのユーザ管理機能は、NCMBのものをそのまま使っています。ソーシャルログイン機能は使っておらず、パスワードかメールアドレスによるユーザ登録です。iOS版(Objective-C)の実際のソースコードのログイン部を表示します(**リスト1**)。

BLEで環境センサーとやりとり

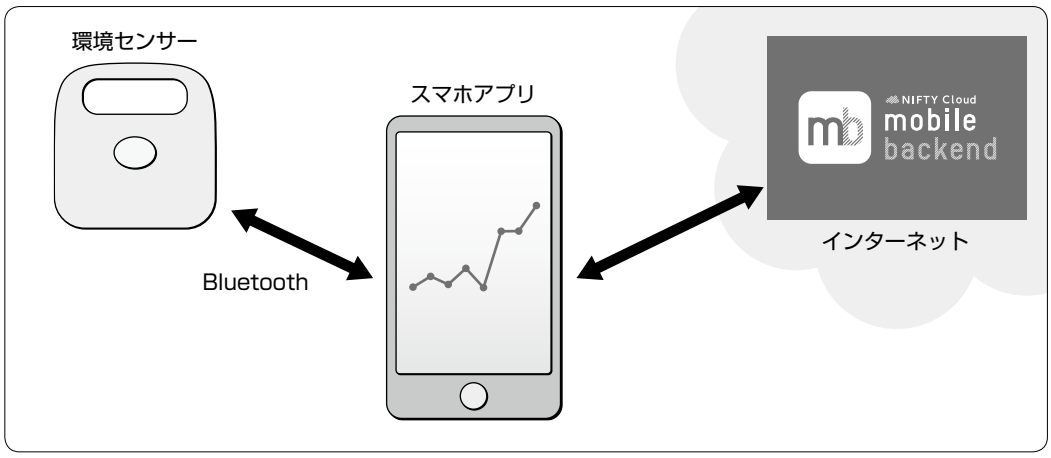
このアプリの特徴的なところはスマホ内の処理だけで閉じているわけではなく、外部機器である環境センサーとBLEでやりとりしながら動作する部分です。このような連携があるので、通常のアプリ開発より調整には時間がかかるだろうなと当初から想定していたそうです。

環境センサーとのやりとりは、マルチスレッドを用いてアプリのバックグラウンドで行う必要があるので、開発的にも苦労したところだそうです。

あとはiOS、Androidそれぞれの Bluetooth

注3) http://mb.cloud.nifty.com/iot-trialkit/

▼図6 環境センサー構成図



▼リスト2　データストアのバックグラウンド保存部分（iOS版）

```
+(void) saveInBackgroundRetry:(NCMBObject*) object retry:(NSUInteger) times ➥
done:(NCMBErrorResultBlock)userBlock{
    [object saveInBackgroundWithBlock:^(NSError *error) {
        if(error == nil){
            if(userBlock){
                userBlock(error);
            }
        } else {
            NSInteger count = times -1;
            NSError *err = nil;
            while(count > 0){
                err = nil;
                [object save:&err];
                if(err == nil){
                    count = 0;
                } else {
                    count--;
                }
            }
            if(userBlock){
                userBlock(err);
            }
        }
    }];

}
```

▼リスト3　データストアの保存部分（iOS版）

```
+(NCMBObject*) saveRetry:(NCMBObject*) object retry:(NSUInteger) times error:(NSError**)error{

    for(int i=0; i< times; i++){
        [object save:error];
        if(error == nil){
            return object;
        } else if(*error == nil){
            return object;
        }
    }
    return object;
}
```

▼リスト4　実際のデータ保存部分

```
NCMBObject *cloudObject = [NCMBHelper cloudObjectFromDbOject:CLOUD_TABLE_NAME_MEASURE_DATA ➥
dbObject:cloud];
                [NCMBHelper saveRetry:cloudObject retry:2 error:nil];
```

スタックの扱いに苦労がありました。Androidでは機種ごとの差が大きく、それぞれつながりやすさが違いました。うまく動作しないケースがあった場合に、それが機種依存なのかプログラムの問題なのか切り分けるのがたいへんでした。

iOSでは、OS標準のBluetooth機能の自由度がそもそも低く、欲しい情報を取得できなかったり、iBeaconの仕様が告知なしで変わったりで、また別の苦労がありました。

センサーからのデータは、基本的には常にアプリ側に送られ続けます。それらを速報値として画面上に表示しつつ、後述のアップロードのためのバッファに保持します。

取得したデータをNCMBにアップロード

センサーからアプリが取得したデータは、NCMBのデータストアに順次アップロードされます。保存に関してはNCMB側が用意するAPIをコールするだけですので、処理としては簡単です。保存した後のデータは、サービス側で勝手に冗長度を高めて保持してくれるのがmBaaSサービスのありがたいところです。

このアプリで工夫している部分は、内部的なヘルパーメソッドでデータストアへの保存や検索部分をラップして、リトライ回数を指定できるようにしていることです(**リスト2**)。不安定な接続回線でもAPI呼び出しの成功率を少しでも上げるようにこのようになっています(**リスト3**)。

アプリ上でセンサーデータをグラフ表示する場合も、データストアに保存してある値をAPI経由で取得して、それを加工したものをグラフにしています(**リスト4**)。環境によってはアップロードに時間がかかる場合もあるため、その場合はアプリローカルにデータが存在する場合はそれも含めてグラフ表示しています。

プッシュ通知

センサーが出力するデータにはしきい値を設定できて、その範囲を外れた場合はアラートを飛ばすようになっています。アラートにはNCMBのプッシュ通知機能を用いていて、データ転送端末以外の端末に送ることもできます。

ファームウェアアップデート

ある程度の大きさのデータをオブジェクトストレージ的に保存できるのがNCMBのファイルストア機能ですが、環境センサーアプリではこれをデバイスファームウェアイメージの配布場所として使っています。さまざまな不具合を随時修正していく手段としてファームウェアアップデートはIoT機器の必須機能となっています。そのファームウェアイメージのためだけにサーバを用意するのも運用負荷になるので、NCMBのファイルストアを用いることで統合的に管理しています。

mBaaSの未来に期待すること

最後にバイタリフィに、mBaaSサービスの未来に期待することを伺ってみました。

バイタリフィにはさまざまな案件の相談がやって来るわけですが、依頼者のITサービスに関する理解もピンキリで、サーバサイドの開発運用予算が捻出できないまま立ち消えになるケースもあるそうです。そのような案件のためには、サーバなしでサービス型のアプリを運用できるのが理想であり、アプリに必要な機能群が適切な価格で利用できれば問題ないわけです。

もし今回の環境センサーアプリをサーバサイドも含めてフルスクラッチで開発していたとしたら、1.2〜1.3倍の期間と予算がかかっていただろうし、日々の運用コストも増大していただろうことを考えると、サービスに任せられることは任せるという開発方法はありだと実感されたということです。

mBaaSはその名前からみても、最初はモバイルアプリに特化したAPIを備えたサービスからスタートしていますが、ユーザ認証だったりDBだったり、さまざまなサービスの汎用基盤としてとらえることもできます。今後はより対応分野が広がっていくことが期待されます。

次号では

NCMBを支える技術について開発・運用を行っているエンジニアにお話を伺います。またアプリケーションをどのように作成するのか、具体的に解説を行います。SD

特別企画

PostgreSQLのGPU拡張モジュール

PG-Stromの構造と機能、そしてその威力とは 後編

どのようにGPUは集約演算を実行するのか？

Author 海外 浩平（かいがい こうへい）
PG-Strom Devloper Team
Twitter @kkaigai

はじめに

先月号では、GPUを用いてJOINやGROUP BYなどの集計・解析系処理を高速化するPostgreSQL向け拡張モジュールである、PG-Stromの基本的な構造とその基盤技術であるGPUの特徴を説明し、さらにAWSのGPUインスタンスを用いた環境構築方法を紹介しました。

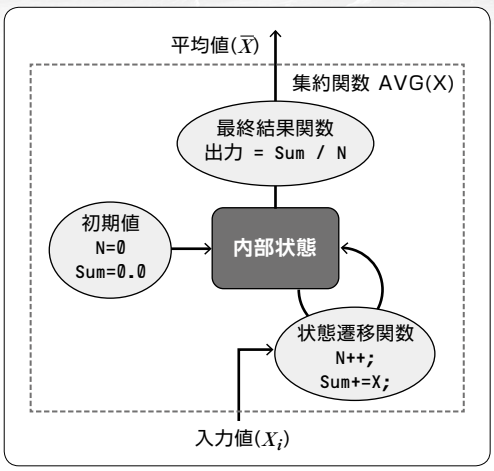
今回は、SQLにおける基本的な操作の1つである集約関数GROUP BYを例[注1]に、これをどのようにGPU上で実行するのか掘り下げるとともに、現在進行中の機能強化について紹介します。

集約関数GROUP BYとは

集約関数とは、行カウント（COUNT関数）や平均値（AVG関数）のように、複数の入力行を集計し単一の結果を出力するタイプの関数で、レポート作成や統計解析のワークロードで非常によく利用されます（**図1**）。

集約関数の構成要素は3つあり、初期値、状態遷移関数、および最終結果関数です。例として、平均値を出力するAVG()関数を考えてみることにします。ご存じのとおり、平均値を求めるには母集団の総和を要素数で除算します。PostgreSQLの場合、集約関数AVG()を実行すると対応する内部状態が初期値 N=0, Sum=0.0 で作られます[注2]。集約関数に対してレコードが入力される度に状態遷移関数が呼び出され、AVG()の状態遷移関数は要素数Nを1加算、総和Sumに入力値を加算して内部状態を更新します。そして最後に最終結果関数が呼び出され、総和Sumを要素数Nで除算した値を出力します。これが集約関数AVG()の出力結果となるわけ

▼図1　集約関数の動作



注1）前編では「JOINを例に～」と予告しましたが、集約関数GROUP BYの方がより"GPUらしい"実装になっているため変更しました。

注2）実際には、AVG()関数のデータ型によって微妙に実装は異なります。

▼図2　PG-Stromが無効の場合（標準のPostgreSQL）

```
postgres=# EXPLAIN VERBOSE SELECT cat, count(*), avg(v) FROM t0 GROUP BY cat;
                                QUERY PLAN
---------------------------------------------------------------------------
 HashAggregate  (cost=2386943.00..2386943.32 rows=26 width=12)
   Output: cat, count(*), avg(v)
   Group Key: t0.cat
   ->  Seq Scan on public.t0  (cost=0.00..1636943.00 rows=100000000 width=12)
         Output: id, cat, aid, bid, v
(5 rows)
```

▼図3　PG-Stromが有効の場合

```
postgres=# EXPLAIN VERBOSE SELECT cat, count(*), avg(v) FROM t0 GROUP BY cat;
                                QUERY PLAN
---------------------------------------------------------------------------
 HashAggregate  (cost=2386943.00..2386943.32 rows=26 width=12)
   Output: cat, pgstrom.count((pgstrom.nrows())),
                pgstrom.avg((pgstrom.nrows((v IS NOT NULL))), (pgstrom.psum(v)))
   Group Key: t0.cat
   ->  Custom Scan (GpuPreAgg) on public.t0  (cost=4000.00..1672195.34 rows=234 width=48)
         Output: cat, pgstrom.nrows(), pgstrom.nrows((v IS NOT NULL)), pgstrom.psum(v)
         Reduction: Local + Global
         GPU Projection: t0.id, t0.cat, t0.aid, t0.bid, t0.v
         Logic Parameter: SafetyLimit: 2147483647, KeyDistSalt: 9
(9 rows)
```

です。

集約関数とGROUP BY句を併用すると、GROUP BY句で指定した列の値ごとに集約関数の内部状態が作られ、同じ数の最終結果が出力されます。たとえば次のクエリは都市ごとに降水量(rainfall)を集計し、その平均値を出力します。

```
SELECT city, AVG(rainfall)
  FROM weather
 GROUP BY city
```

PG-Stromの集約演算サポート

次に、PG-Stromの集約関数サポートを見てみることにしましょう。**図2・図3**は、集約関数COUNT()、AVG()と、GROUP BY句を含む比較的単純なクエリの実行計画を、PG-Stromが無効な場合、有効な場合でそれぞれ表示させたものです。

標準のPostgreSQLでは、HashAggregateはt0テーブルに対するSeqScan(全件スキャン)の結果を入力として受け取り、SQLのGROUP BY句で指定した列と集約関数を出力します。

一方、PG-Stromが有効である場合、最上位にHashAggregateが位置しているのは同様ですが、その入力はPG-Stromの提供するGpuPreAggになっており、出力する集約関数もSQLの指定とは異なるものが指定されています。

これは、GPU側である程度大雑把な集約演算を行った後、これら“中間集計”とでも呼ぶべきデータをCPU側で取りまとめて最終的な集約関数の結果を出力するという構造を取っているためです。

たとえば、集約関数count(*)が出力されるべき位置は、pgstrom.count()という別の集約関数に置き換えられており、この関数はGpuPreAggの出力pgstrom.nrows()を入力として受け取ります。

PostgreSQLの集約関数count(*)の内部状態は“行数”だけで、入力行があるたびに1ずつ

加算されていきます。しかし、これは必ずしも律儀に1行ずつ加算しなければならないわけではなく、仮にGpuPreAggが100万行分の“中間集計”を出力するのであれば、“中間集計”に含まれる100万行分を一度に加算しても良いハズです。

集約関数count(*)を置き換えたpgstrom.count()とpgstrom.nrows()のペアは、GpuPreAggがGPU側で中間集計した行数を一度に集約関数の内部状態に加算します。

集約関数avg(v)も同様で、これを置き換えたpgstrom.avg()関数は、GpuPreAggがGPU側で中間集計した行数(nrows()関数)と部分和(psum()関数)の2つを引数にとり、数百万行以上に相当する内部状態の更新を一度に行います。

このように集約関数の処理を前処理／後処理に分けることで、負荷の大部分をGPU側が引き受けることになり、ごく僅かな行しか処理しないCPU側の負荷は事実上無視できる程度に軽減されることになります。

ほかにも、PostgreSQLでは総和、最大値／最小値といった一般的な集約関数、分散／標準偏差や共分散／回帰分析といった統計処理用の集約関数を持っており、PG-Stromではこれらを前処理／後処理に分解して処理できます[注3]。

どのようにGPUは集約演算を実行するのか?

それでは、集約演算を前処理／後処理に分解した後、前処理であるGpuPreAggがどのように動作するのかを見てみることにします。

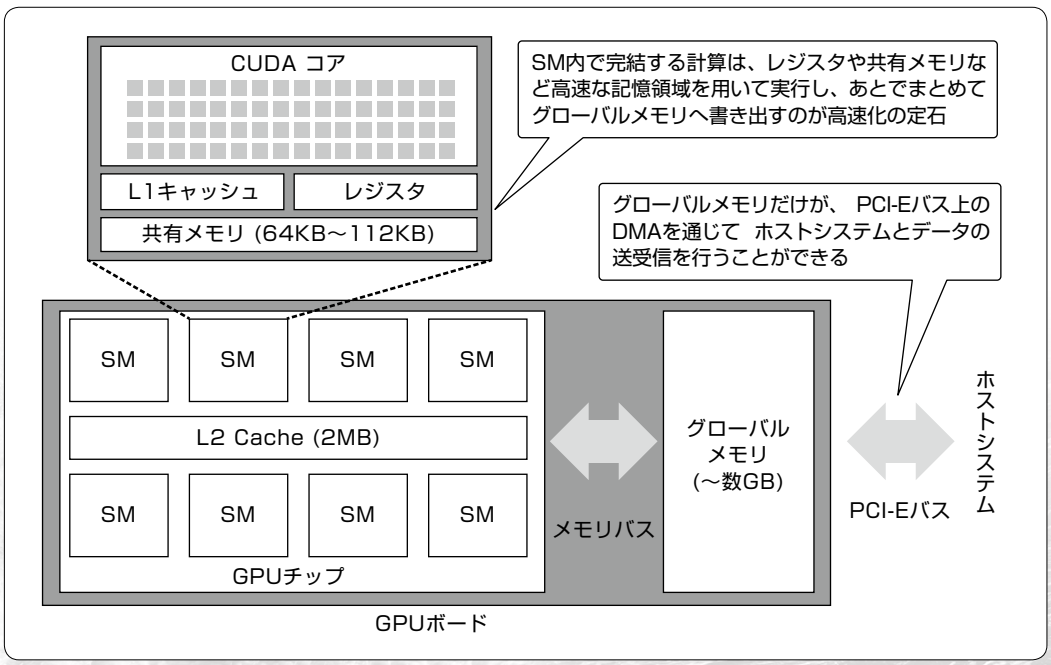
図4はGPU上のコンポーネントがどのように配置されているかを示すブロック図です。

GPUには数百～数千個のCUDAコアが搭載されていますが、これらは64コアや128コア[注4]単位でSM(Streaming Multiprocessor)というユニットに搭載されています。SMはこのほか、レジスタ、共有メモリやL1キャッシュなどのリソースを持っており、SM上で同時実

注3) アルゴリズム上、最頻値(mode関数)などは前処理／後処理に分解できないため、PG-Stromでは対応していません。PostgreSQL v9.6以降の並列クエリでも同様です。

注4) GPUの世代やモデルによって異なります。

▼図4　GPUのブロック図



行できるスレッドの数はこれらリソースの消費量によって決まります[注5]。

レジスタはスレッドに固有の記憶領域で、共有メモリは同一のSMで実行中のスレッドが共有する記憶領域です。決して潤沢な容量が確保されているわけではありませんが、CUDAコアに近く、他のSMとの同期を考えなくてよい分、極めて高速にアクセスすることができる記憶領域です。

一方、これらSM内の記憶領域と比べると、特にアクセスのレイテンシを考えた場合、全てのSMで共有するグローバルメモリ(DRAM)は決して高速なメモリではありません。そのため、GPUでパフォーマンスチューニングを行う時の定石は、先ずSM内で完結する計算は高速なレジスタ／共有メモリを用いて実行し、その後で結果をグローバルメモリへ書き出すということになります(**図4**)。

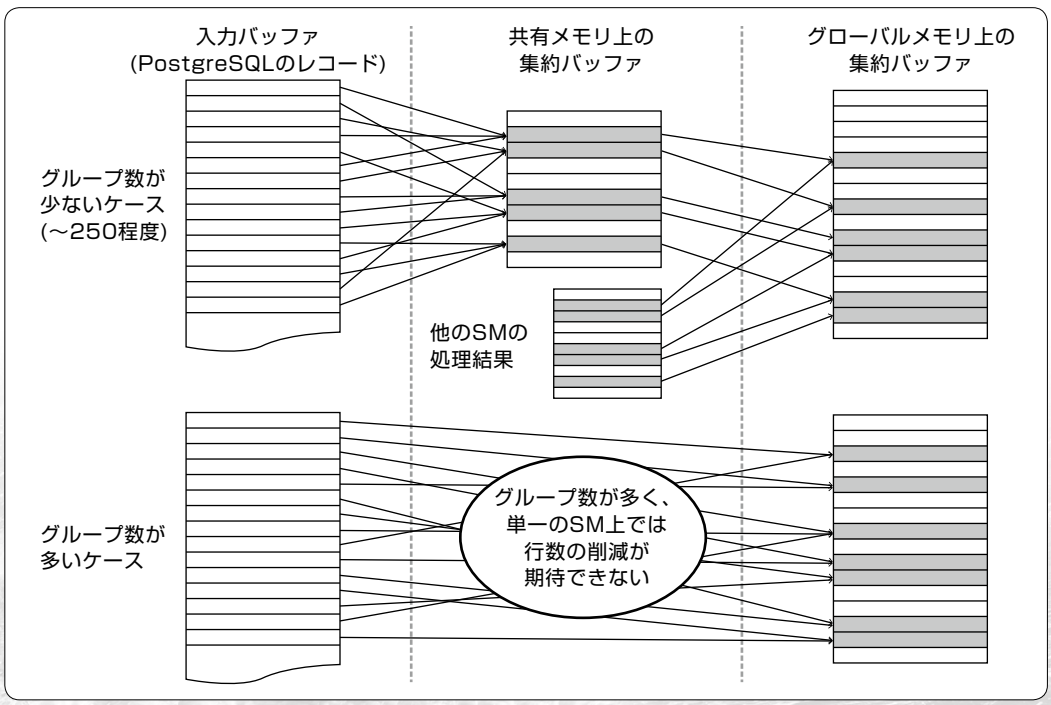
PG-StromのGpuPreAggは、内部的にハッシュ表を用いてGROUP BY句の処理を行います。GROUP BY句で指定された列の内容から32bitのハッシュ値を計算し、これを元に集約バッファ上の位置(Hashスロット)を特定します。次に、このHashスロットへ入力行の内容から導出した初期値を加算[注6]していきます。この初期値はたとえばCOUNT()関数であれば1、AVG(X)関数であれば(Xが非NULLであれば)1とXというように、集約関数の種類によって異なります。

当然、複数行が同じグループに属しており、同じHashスロットを更新する場合があります。このとき並行するスレッドとの競合で加算が不正な結果をまねかないよう、GpuPreAggはアトミック演算を用いて集計操作を行います。

注5) たとえば、前世代のMaxwellアーキテクチャではSMあたり65,536個のレジスタが搭載されています。したがって、1個のSMで実行可能な1,024個のスレッドを投入するには、スレッドあたりのレジスタ使用量は64個以下でなければいけません。

注6) MAX/MIN関数の場合は最大値／最小値のセットです。

▼図5　GpuPreAggの処理イメージ



ただ、アトミック演算は非常に“重い”メモリ操作なので可能な限り回数を減らし、特にアトミック演算の競合によって“待たされる”回数を減らすことが性能面からは重要になってきます。

PostgreSQLの統計情報などから、GROUP BYの結果生成されるグループ数を推定することができます。グループ数がさほど多くない場合、PG-Stromはまず(高速な)共有メモリで第1段階の集約を行い、次いで(比較的低速な)グローバルメモリ上で第2段階の集約を行います。グループ数が少ないということは、アトミック演算で衝突する可能性が高いということでもあるので、あらかじめSMの中で集約できる部分は集約しておくことで、より低速なグローバルメモリ上でのアトミック演算の競合を減らすことができます(**図5**)。

図3の実行計画の中で「Reduction: Local + Global」と出力されていますが、これはグループ数が十分に少ないと予測されるため、2段階での集約演算を行うという意味です。

一方で、SM上で同時に実行できるスレッド数は1,024が上限なので、グループ数が多い場合には第1段階の集約が十分に機能しません。たとえば1,024行を950行に削減できたとしても性能面でのアドバンテージはないでしょう。こういったケースでは最初からグローバルメモリに対してアトミック演算を行いますが、もともと多くのグループに分散するためにアトミック演算の衝突が起こることはそれほど多くはありません。

逆に、グループ数が10や20といった集約率の高いケースでは、アトミック演算の衝突が多発します。こういったケースの性能低下を避けるため、PG-StromはGROUP BY句の値に追加のランダム性を加えてハッシュ値を計算します。実行計画の中で“KeyDistSalt: 9”と表示されているものがそれですが、この場合、同一のグループであっても9個の異なるHashスロットにランダムに放り込まれるため、アトミック演算の衝突可能性が減少するということになります。

このKeyDistSalt値はグループ数が少ないときにしか設定されないため、たとえばGpuPreAggの出力する行数が20行から200行といった形で増加することにはなるものの、全体的な処理コストの増加は微々たるものです。

GROUP BYのない集約演算

GROUP BY句付きの集約演算の場合、どの行がどのグループに属するかを事前に知ることはできないため、ハッシュ表とアトミック演算を用いてグループごとの集計演算を実装する必要がありました。しかし、GROUP BY句がなく、すべての入力行を1個の出力行に集約する場合は、非常に“GPUらしい”実装ができます。

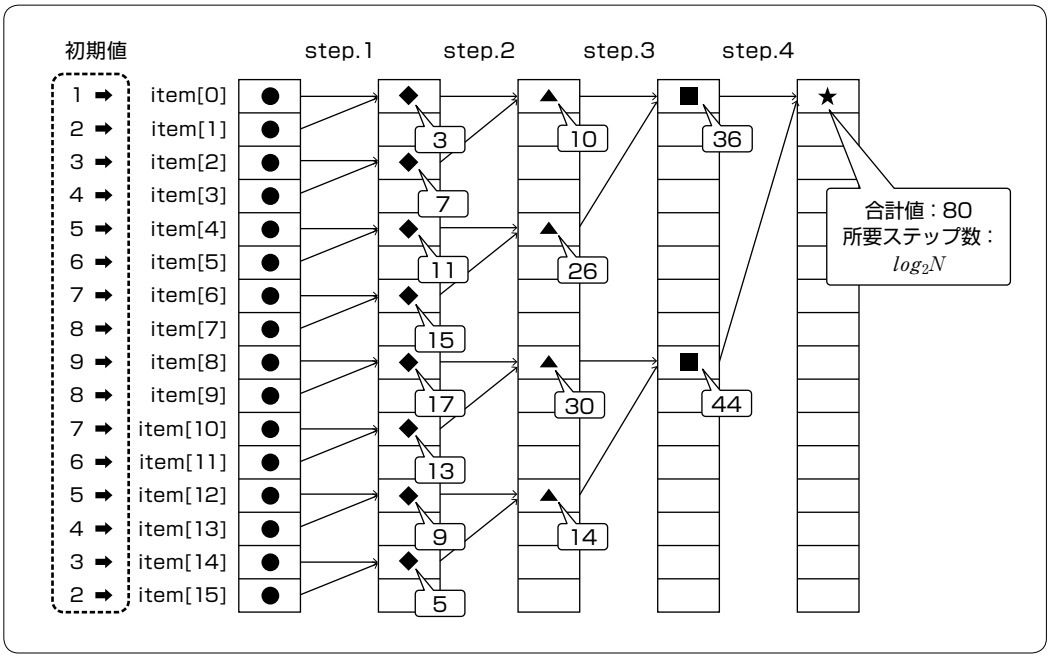
前編で紹介したように、NVIDIA社のGPUではWarpと呼ばれる32スレッドを単位としてスケジューリングが行われ、同一のWarpに属する各スレッドは同じタイミングで同じ命令を実行します。また、同一のSMで動作するスレッド同士は`__syncthreads()`というビルトイン命令によって容易に同期を取ることができます。

このような特性を利用して配列の各要素の和を計算するといった縮約計算は、GPUプログラミングにおける定石の1つです(**図6**)。

たとえば、入力レコードから列Xを参照して合計値を求める処理を考えてみます。配列要素が各レコードに対応するとして、その初期値として列Xの内容を読み出してセットします。次に、配列の奇数要素の値を1つ前の偶数要素に加算し、その次のステップでは4の剰余が2の配列要素を2つ前の配列要素に加算し……という操作を繰り返していきます。すると、わずか10ステップで1,024個の配列要素の合計を計算できます。

図7の実行計画では「Reduction: NoGroup」と表示されていますが、これはアトミック演算を伴わない縮約計算によって、集約関数の値を計算してます。単に合計値を求めるだけではなく、

▼図6　GPUによる縮約計算



▼図7　GROUP BY句を伴わない集約関数

```
postgres=# EXPLAIN SELECT count(*), sum(v) FROM t0;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=2136943.00..2136943.01 rows=1 width=8)
   ->  Custom Scan (GpuPreAgg) on t0  (cost=4000.00..1664383.06 rows=256 width=12)
         Reduction: NoGroup
         GPU Projection: id, cat, aid, bid, v
(4 rows)
```

配列要素の初期値に1をセットして縮約計算を行えばcount(*)の結果が得られますし、行数と合計値を縮約計算で求めれば平均値を導出できます。

CPU+GPUハイブリッド並列

PostgreSQL v9.6の目玉機能の1つがパラレルクエリへの対応で、全件スキャンやJOINのほか、集約演算の実行にもパラレルクエリを使用できます[注7]。**図8**の実行計画を見ると、パラレルクエリ環境下では集約演算はPartialとFinalizeの2段階に分けて実行されることがわかります。ワーカープロセス側ではPartial HashAggregate、つまり平均値であれば要素数と総和という集約関数の内部状態を出力する実行計画が選択され、これを受け取ったマスタープロセス側ではFinalize GroupAggregateによって最終結果を計算、出力します。

このしくみは、きわめて容易にPG-StromのGpuPreAggと組み合わせられます。標準のPostgreSQLでPartial HashAggregateが配置されている個所をPG-StromのGpuPreAggで置き換え、このGpuPreAggをパラレルクエリのしくみを用いて複数のワーカープロセスで

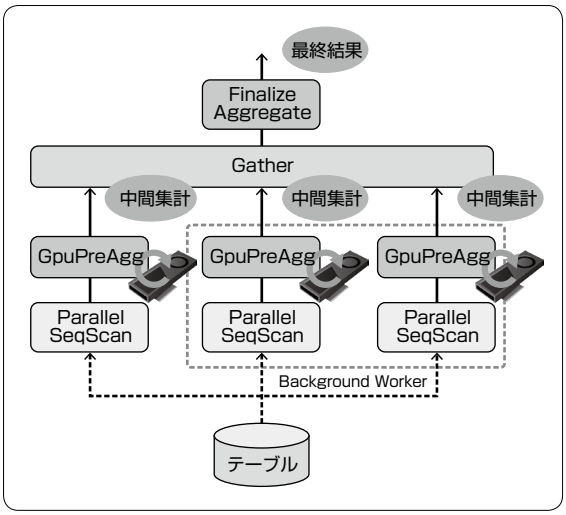
注7）パラレルクエリの選択にはプロセス間通信のコストも考慮されるため、パラレルクエリを含む実行計画が作成されるのは、集約演算GROUP BYを含むクエリであることが大半だと思われます。

▼図8　PostgreSQL v9.6の集約演算（PG-Strom無効）

```
postgres=# SET max_parallel_workers_per_gather = 100;
SET
postgres=# EXPLAIN VERBOSE SELECT cat,count(*),avg(v) FROM t0 GROUP BY cat;
                                QUERY PLAN
---------------------------------------------------------------------------
 Finalize GroupAggregate  (cost=1185605.29..1185607.43 rows=26 width=20)
   Output: cat, count(*), avg(v)
   Group Key: t0.cat
   ->  Sort  (cost=1185605.29..1185605.74 rows=182 width=44)
         Output: cat, (PARTIAL count(*)), (PARTIAL avg(v))
         Sort Key: t0.cat
         ->  Gather  (cost=1185580.00..1185598.46 rows=182 width=44)
               Output: cat, (PARTIAL count(*)), (PARTIAL avg(v))
               Workers Planned: 7
               ->  Partial HashAggregate  (cost=1184580.00..1184580.26 rows=26 width=44)
                     Output: cat, PARTIAL count(*), PARTIAL avg(v)
                     Group Key: t0.cat
                     ->  Parallel Seq Scan on public.t0  (cost=0.00..1077437.14 rows=14285714 width=12)
                           Output: cat, v
(14 rows)
```

並列に動かすのです。ある程度スペックの高いGPUであれば、CPUがPCIeバスを通じてデータをGPUに転送するよりも速いペースで集約演算を完了できるため、CPU並列によるデータ供給レートの引き上げと、GPUによる高速な集約演算を両立させることができるようになります。CPU+GPUハイブリッド並列機能は、PostgreSQL v9.6に対応したPG-Strom v2.0で対応の予定です（**図9**）。

▼図9　CPU+GPUハイブリッド配列



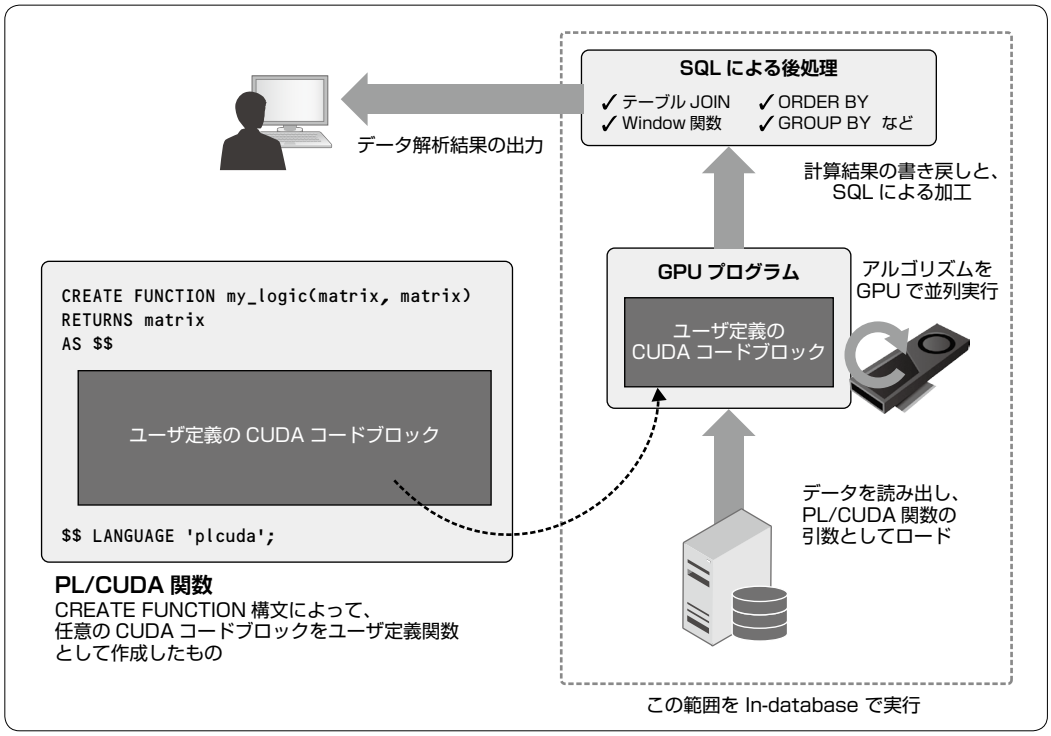
PG-Stromの今後

本稿の最後に、現在開発を進めているPG-Strom v2.0の先進機能を2つ紹介します。

PL/CUDA ――計算集約的な問題に対して

PG-StromはSQLからGPU用のコードを自動生成できます。そのため、SELECT句の列選択リストに非常に複雑な計算式を記述し、これをGPUで計算させてその計算結果をCPU側で受け取るという使い方も可能です。以前に筆者が42次元ユークリッド距離の計算ワークロードで処理性能を評価したときには、PostgreSQLに比べ8倍程度の速度改善を得ることができました。しかし、このアプローチには何点か問題がありました。第一に、そもそもSQLは複雑な統計解析アルゴリズムの記述に向いた言語ではないことです。とくに繰り返し処理の記述が困難です。第二に、自動生成されたGPUコードはPostgreSQLの関数・演算子の挙動を正確に反映する必要があるため、たとえばNULLチェックや計算結果のオーバーフローチェックなどSQL互換性に由来するオーバーヘッドが無視できません。第三に、PostgreSQLのデータ構造は必ずしもGPU向きではないために、

▼図10　PL/CUDA関数



メモリ参照でプロセッサがストールしてしまい、その計算能力をフルに発揮できません。

これらの問題に対処するため、PG-Stromの“透過的な高速化”というコンセプト[注8]を曲げて、人間の手で最適化したCUDAコードブロックをユーザ定義関数として実行する機能を追加しました(**図10**)。

PL/CUDA関数が呼び出されると、PL/CUDA言語ハンドラは関数定義に含まれるCUDAコードブロックをGPUプログラムのテンプレートに埋め込んで実行時ビルドし、GPU上で並列実行します。SQL関数の引数は自動的にGPU側に転送され、また実行結果も同様にPostgreSQL側に書き戻されます。これにより、PL/CUDAの開発者は煩雑なGPUとのデータ転送にかまうことなくアルゴリズムの記述に集中できます。

また、出来合いのアルゴリズムを利用することを考えた場合でも、DBMSからデータをエクスポートせず高度な統計解析を行った“結果”だけを出力できることや、SQLの機能を用いてフレキシブルに前処理／後処理を行えるのは大きなアドバンテージと言えるでしょう。

紙幅の都合で割愛しますが、創薬分野でPL/CUDAを用いた類似化合物の検索ワークロードを100倍以上高速化した検証事例がPGconf.ASIA 2016[注9]で公開されています。併せてご覧ください。

現状、PL/CUDA関数を利用するにはスクラッチの開発が必要ですが、開発チームでは、MADLib[注10]のような汎用の統計解析関数のコレクションをPL/CUDA上に実装することを検討しています。

注8) そもそも、統計解析アルゴリズムをIn-database実行するために強烈に複雑なSQLを書く時点で、透過的な高速化の意義はほぼ失われているわけです。

注9) http://www.pgconf.asia/JP/material/

注10) http://madlib.incubator.apache.org/

NVMe-Strom——I/O中心のワークロードに対して

本来、GPUは計算のアクセラレータであり、CPUの過負荷をアシストすることで処理時間を短縮するものです。CPUやRAMの動作速度に比べるとI/Oは文字どおり“桁違い”に低速で、I/Oが発生した瞬間、全体の処理速度がI/Oに引きずられてしまいます。そのため、PG-Stromの利用時はデータをあらかじめRAMにロードすることを推奨していたのですが、これはデータサイズによっては難しい場合もあります。

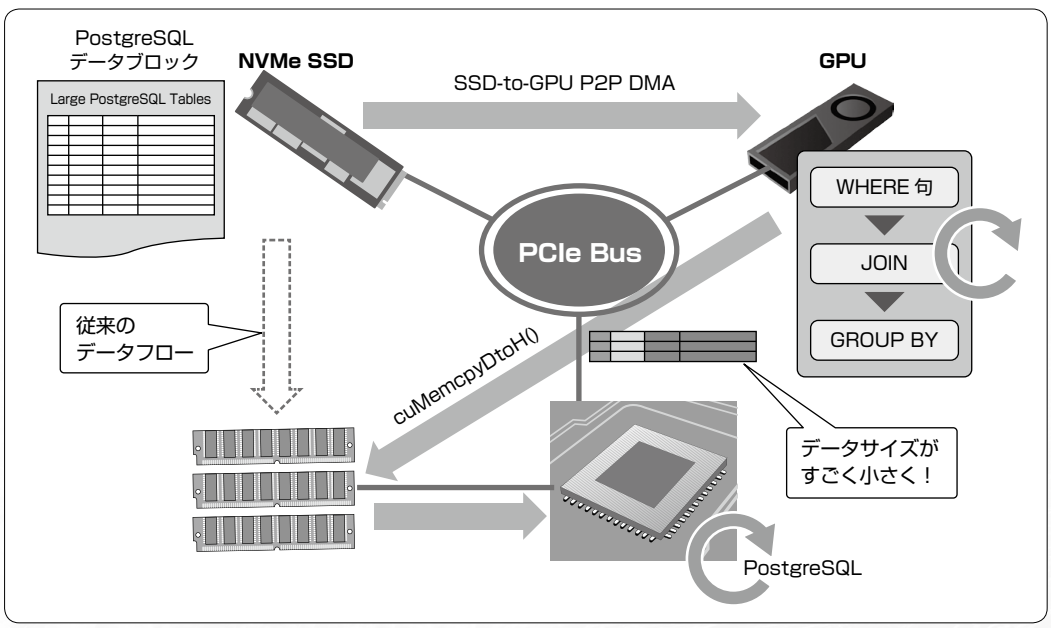
Tesla/QuadroブランドのGPUにはGPUDirect RDMAという機能があり、同じPCIeバスに接続されたサードパーティ製のデバイスとの間でPeer-to-Peerのデータ転送ができます。本来はInfinibandのHBAを介してリモートのGPUとの間でRDMAを行うために提供された機能なのですが、たとえばNVMe-SSDなど、PCIeバスに挿さっているデバイスであれば本機能を利用できます。

筆者が目を付けたのは、NVMe-SSD上に格納されているPostgreSQLのデータブロックを直接GPUに転送し、そこでWHERE句の評価やJOIN、GROUP BYを実行してデータ量を減らすことで、間接的にI/Oの負荷を減らすことに使えないか？——というものでした(**図11**)。

幸い、PG-StromはすでにPostgreSQLの行形式データに対してJOINやGROUP BYをGPU上で実行するためのしくみを持っています。したがって、ディスクキャッシュなどの状態と整合性を取りつつNVMe-SSDからGPUへのデータ転送を制御するモジュールを実装すれば、“GPUでI/O量を減らす”機能の実現はさほど難しいものではありません。

図12はStar Schema Benchmarkによる性能比較です。このワークロードは、1個の巨大なテーブルと複数個の小規模なテーブルからなるスタースキーマ構造に対して、全件スキャンやJOIN、GROUP BYを伴うクエリを何種類か実行するものです。中心となるlineorderテーブルのサイズが351GBであるのに対し、測定環境のRAMサイズは128GBですので、クエリの応答時間を支配するのはI/Oのスループット

▼図11　SSD-to-GPU Peer-to-Peer DMA



ということになります。

測定に使用したNVMe-SSDのカタログスペックは、順次読み出し2200MB/sです。これをRAID0によるストライピングで2枚束ねているのですが、NVMe-Stromを使用したケースでは理論速度の85%に相当する3,800MB/sものスループットでクエリを処理できていることがわかります[注11]。一方、PostgreSQLがファイルシステムを介してディスクやSSDからデータを読み出したケースでは、最大でも1,600MB/s程度のスループットに留まっており、GPUとSSDを密結合することでI/Oの高速化に寄与していることがわかります。

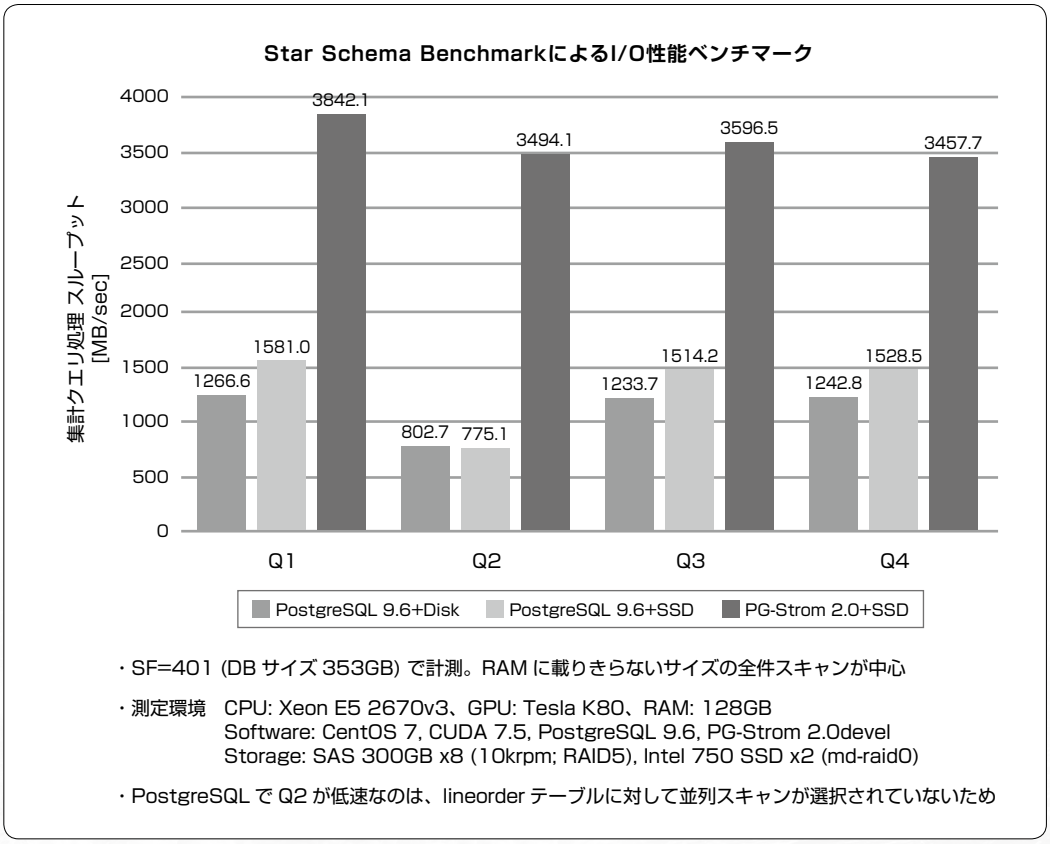
おわりに

本稿では、PG-StromによるSQL処理のうち、もっともGPUらしい実装である集約演算GROUP BYの処理を掘り下げて説明しました。

データベース管理システムが処理すべきデータ量の増大に伴い、性能改善の手法としてさまざまな方法が提案されていますが、PG-Stromは価格対計算能力の高いGPUを使用し、クエリ修正やデータ移行を伴わずに利用できる“お手軽”な方法です。今回の記事で、PostgreSQLにはこのような高速化のアプローチが存在するということを知っていただければ幸いです。SD

注11）DBサイズをクエリ実行時間で割って処理スループットを導出しています。

▼図12　Star Schema Benchmarkによる比較



・SF=401 (DB サイズ 353GB) で計測。RAM に載りきらないサイズの全件スキャンが中心

・測定環境　CPU: Xeon E5 2670v3、GPU: Tesla K80、RAM: 128GB
Software: CentOS 7, CUDA 7.5, PostgreSQL 9.6, PG-Strom 2.0devel
Storage: SAS 300GB x8 (10krpm; RAID5), Intel 750 SSD x2 (md-raid0)

・PostgreSQL で Q2 が低速なのは、lineorder テーブルに対して並列スキャンが選択されていないため

短期集中連載

Jamesのセキュリティレッスン

パケットキャプチャWiresharkの新展開

第9回 IPsecの暗号化通信を復号してみよう!

Writer 吉田 英二(Eiji James Yoshida)
合同会社セキュリティ・プロフェッショナルズ・ネットワーク(http://www.sec-pro.net/)

はじめに

みなさん、こんにちは! 最近は、Talking Headsを聴いているEiji James Yoshidaです。もはや昔のバンドなので知らないかもしれませんが、白人音楽と黒人音楽を融合したような曲調とDavid Byrneの演説のような歌い方が実験的で筆者は大好きだったりします。とくにアルバム「Remain in Light」は後世に語り継がれるような作品なので、ぜひ聴いてみてください。

あと、本誌巻末にある読者コーナーの「Readers' Voice」、毎回ドキドキしながら読んでいます。本連載への励みになるたくさんのコメント、本当にありがとうございます!

さて、前回(2017年1月号)はWiresharkで無線LANの暗号化通信を復号する方法を説明したので、今回はIPsecの暗号化通信を復号する方法について説明しましょう。

環境説明

本稿を書く際に使用した環境は、Windows 10でWireshark 2.2.3です。

・Wireshark
https://www.wireshark.org/download.html

Wiresharkのインストール方法はお任せしますが、とくにこだわりがない場合はデフォルトでインストールしてください。

また、筆者のブログから下記のファイルをダウンロードしてください。

・Software Design 短期集中連載「Jamesのセキュリティレッスン」用キャプチャファイル
http://d.hatena.ne.jp/EijiYoshida/20140907/1410071296
・sd1704_ipsec.pcapng
・sd1704_ipsec.txt

あとは何かBGMでも流しましょう。筆者はTalking Headsのアルバム「Remain in Light」をBGMにして本稿を書いています。「Born Under Punches」や「Crosseyed And Painless」は、最高にカッコイイです。

ネットワーク構成

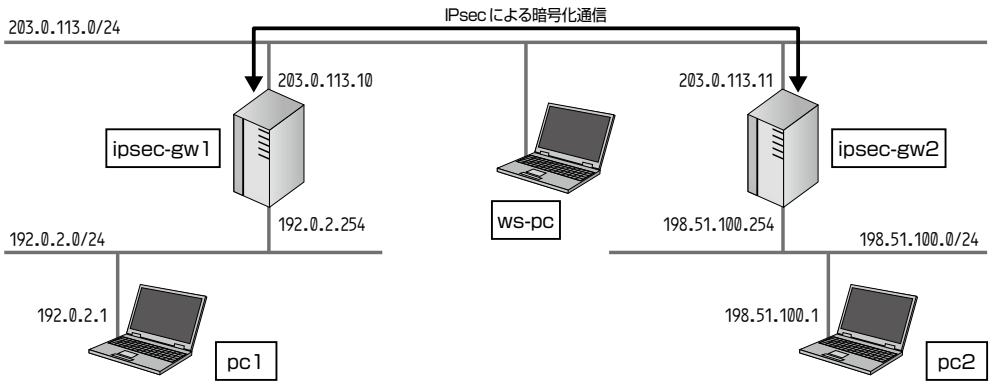
IPsecの暗号化通信をキャプチャするために筆者が用意したネットワーク構成は**図1**になります。

図の中の「ws-pc」は、環境説明の節で書いたWindows 10にWiresharkをインストールしたPCで、IPsecの暗号化通信でやりとりされるパケットはここでキャプチャしました。

「ipsec-gw1」と「ipsec-gw2」は、Ubuntuにstrongswan[注1]をインストールして作ったIPsec Gatewayで、両者はIPsecのトンネルモードで接続されています。

注1) URL https://www.strongswan.org

▼図1　今回のネットワーク構成



「pc1」と「pc2」はクライアントPCで、pingコマンドを使ってpc1とpc2でパケットを送受信するために用意しました。

筆者のブログからダウンロードしたキャプチャファイル「sd1704_ipsec.pcapng」には、pc1で**ping 198.51.100.1**を実行してpc1とpc2で送受信したパケットを、ws-pcでキャプチャしたものが保存されています。

IPsecの暗号化通信

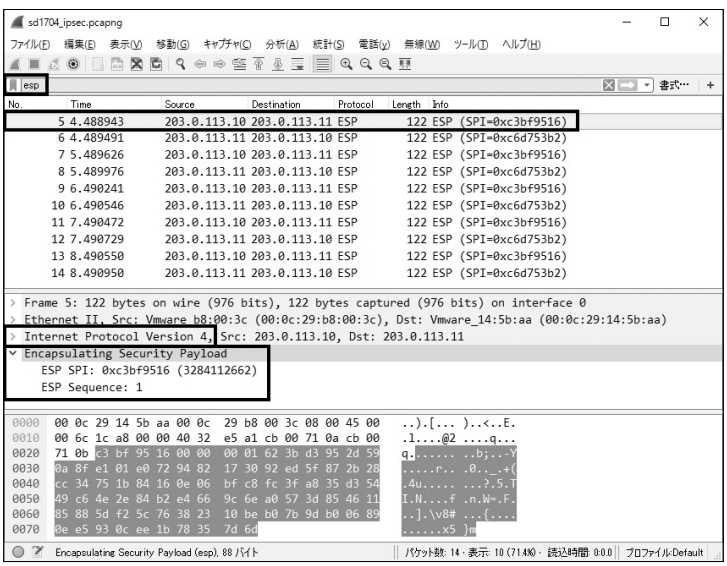
IPsecとは、IPレベルでの認証や暗号化などを実現するしくみの1つで、拠点間の通信を保護するために使っている人も少なくないと思います。

IPsecは複数のプロトコルから構成されていて、通信の暗号化にはESP(Encapsulating Security Payload)というプロトコルが使われます。それでは試しに、キャプチャファイル「sd1704_ipsec.pcapng」をWiresharkで開いてみましょう。

まずはESPで暗号化されたパケット(以降ESPパケット)だけを表示したいので、ディスプレイフィルタとしてFilterボックスに**esp**と入力して Enter を押してください。するとESPパケットだけが表示されるので、Wiresharkの上側に表示されている[パケット一覧]ペインでNo.5のパケットをクリックします。

続いて、Wiresharkの中央に表示されている[パケット詳細]ペインで、[Encapsulating Security Payload]をクリックし、**図2**のように展開してください。[ESP SPI]と[ESP Sequence]の解析結果は表示されますが、それ以降の部分については暗号化されているため、解析結果が表

▼図2　ESPで暗号化されたパケット



示されないことがわかります。

ちなみに、SPI(Security Parameters Index)は、SA(Security Association)と呼ばれるIPsecの一方向通信路のどれで流れているパケットなのかを識別するための値です。Sequenceは、SAでESPパケットを送信する際に1ずつ増加する番号です。

SAとESPパケットの関係がわかり辛いかもしれませんが、簡単に例えると「一方向に流れる川(SA)」に「桃(ESPパケット)」がどんぶらこと流れているような感じになります。

今回は、このIPsecの暗号化通信で流れるESPパケットを、Wiresharkで復号する方法について解説します。

復号に必要な情報

ESPパケットをWiresharkで復号するには、ESPパケットが流れるSAごとに**表1**の情報が必要になります。

IPヘッダ

ProtocolはWiresharkでESPパケットのIPヘッダを見るとわかります。**図2**を見るとNo.5のESPパケットは[パケット詳細]ペインに「Internet Protocol Version 4」と表示されていることから、ProtocolはIPv4ということがわかります。

SA

Src IP、Dest IP、SPIはWiresharkの[パケット一覧]ペインでESPパケットが流れるSAを調べるとわかります。**図2**の[パケット一覧]ペインを見ると、No.5～No.14のESPパケットは、

- 送信元IPアドレスが**203.0.113.10**で、送信先IPアドレスが**203.0.113.11**のSPIが**0xc3bf9516**のSA

- 送信元IPアドレスが**203.0.113.11**で、送信先IPアドレスが**203.0.113.10**のSPIが**0xc6d753b2**のSA

の2つということがわかります。

つまり**図3**のように、No.5、7、9、11、13のESPパケットは、

- **203.0.113.10→203.0.113.11**
  (**SPI=0xc3bf9516**)

のSAで流れていて、No.6、8、10、12、14のESPパケットは、

- **203.0.113.11→203.0.113.10**
  (**SPI=0xc6d753b2**)

のSAで流れていることがわかります。

SAD

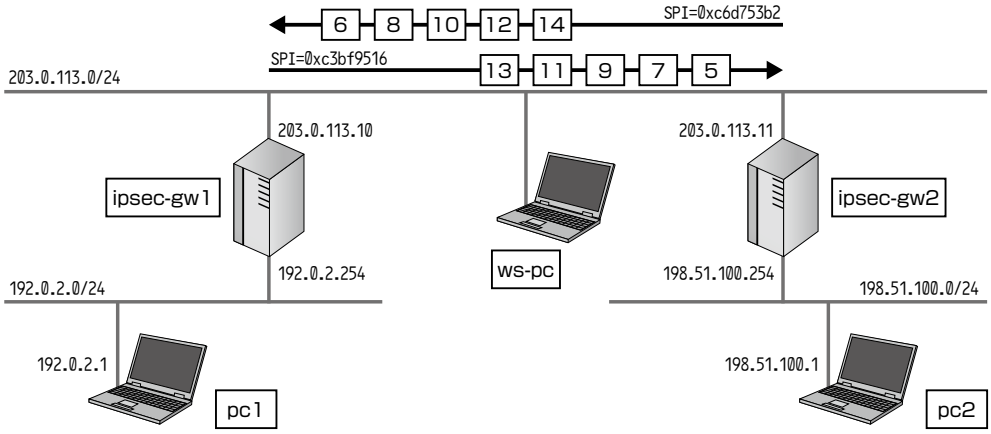
Encryption、Encryption Key、Authentication、Authentication Keyは、SAについてのさまざまな情報が格納されているSAD(Security Association Database)を調べるとわかります。

SADはSAの送信元IPアドレスや送信先IPアドレスのマシンにあるので、**203.0.113.10**のipsec-gw1や**203.0.113.11**のipsec-gw2でSADを調べます。これらのマシンは、UbuntuにstrongSwanをインストールして作ったIPsec

▼表1　ESPパケット復号のための情報

名称	説明	調べる場所
Protocol	IPsecを使用しているIPのバージョン	IPヘッダ
Src IP	送信元IPアドレス	SA
Dest IP	送信先IPアドレス	SA
SPI	16進数のSPI	SA
Encryption	暗号アルゴリズム	SAD
Encryption Key	16進数の暗号鍵	SAD
Authentication	認証アルゴリズム	SAD
Authentication Key	16進数の認証鍵	SAD

▼図3　各SAに流れるESPパケット



Gatewayなので、SADを表示するために**sudo ip xfrm state**を実行します。ipsec-gw1で実行した結果は**図4**で、この内容を保存したものが環境説明のときに筆者のブログからダウンロードした「sd1704_ipsec.txt」ファイルになります。

フォーマットは見たとおりで、通信路の送信元と送信先のIPアドレスがあり、その通信路の情報が字下げされた部分に書かれています。

白枠上部の①の行から、送信元IPアドレスが**203.0.113.10**で送信先IPアドレスが**203.0.113.11**の通信路についての情報であることがわかります。

字下げされた②の行に、プロトコルは**ESP**でSPIが**0xc3bf9516**と表示されていることから、①から④までの行は**203.0.113.10**→**203.0.113.11**(**SPI=0xc3bf9516**)のSAについての情報ということがわかります。

③の行には、行頭に**auth-trunc hmac(sha256)**、行末に**128**とあることから、Authentication は **HMAC-SHA-256-128** で、Authentication Key は**0xd901…(略)…dd92**ということがわかります。

④の行には**enc cbc(aes)**とあることから、Encryption は **AES-CBC** で、Encryption Key は**0x5404…(略)…bb87**ということがわかります。

ここまでが、**203.0.113.10**→**203.0.113.11**(**SPI=0xc3bf9516**)のSAについての情報になります。次の行の白枠下部から、**203.0.113.11**→**203.0.113.10**(**SPI=0xc6d753b2**)のSAの情報になるので、同じ方法で読み解きましょう。

ここまでで、ESPパケットを復号するために必要な情報はすべて入手できたので、**表2**に

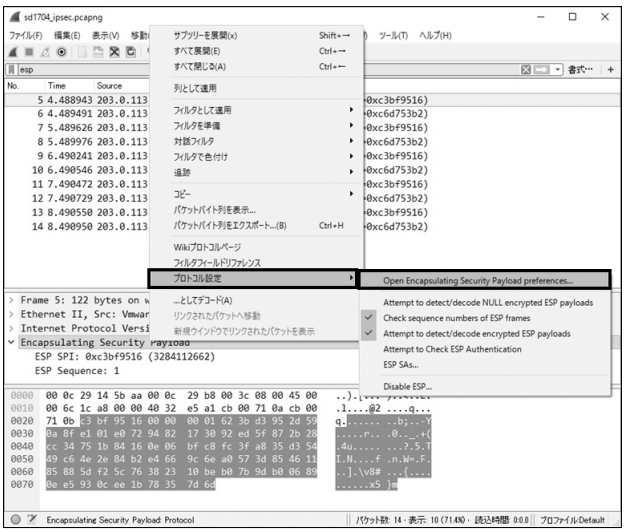
▼図4　ipsec-gw1のSAD

```
user@ipsec-gw1:~$ sudo ip xfrm state
src 203.0.113.10 dst 203.0.113.11 ←①
    proto esp spi 0xc3bf9516 reqid 1 mode tunnel ←②
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha256) 0xd90192ef5431bc592749d9e684dd911ce3a2a81e7b1b56189
cdf990f2968dd92 128 ←③
    enc cbc(aes) 0x5404432c191c77832f0401df3758982326a0fc68d8e3a5440d341a5547d3bb87 ←④
src 203.0.113.11 dst 203.0.113.10
    proto esp spi 0xc6d753b2 reqid 1 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha256) 0x10253bfb59cf50dc796c08b9650270426994ccc6e8b02136b
6bc3baf9d9ea9e5 128
    enc cbc(aes) 0xc98e4e1fcc5ee1a5e48b94e6ce11ff033f798c9b311f54a0d587bee9ff6bf92d
user@ipsec-gw1:~$
```

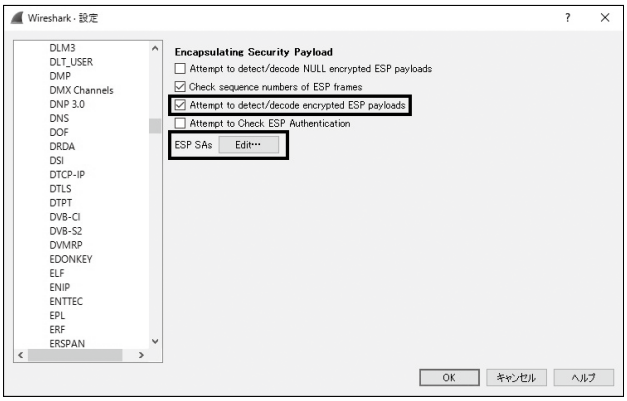
▼表2　ESPパケットを復号するために入手した情報

Protocol	Src IP	Dest IP	SPI
IPv4	203.0.113.10	203.0.113.11	0xc3bf9516
IPv4	203.0.113.11	203.0.113.10	0xc6d753b2
Encryption	**Encryption Key**	**Authentication**	**Authentication Key**
AES-CBC	0x5404…（略）…bb87	HMAC-SHA-256-128	0xd901…（略）…dd92
AES-CBC	0xc98e…（略）…f92d	HMAC-SHA-256-128	0x1025…（略）…a9e5

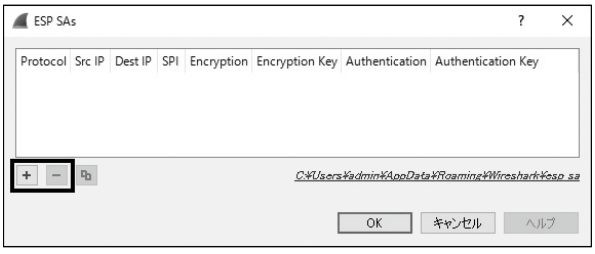
▼図5　［Encapsulating Security Payload］のコンテキストメニュー



▼図6　［Wireshark・設定］ウィンドウ



▼図7　［ESP SAs］ウィンドウ



まとめてみました。ipsec-gw1のSADには必要な情報がすべて含まれていたので、ipsec-gw2のSADの調査は省略します。

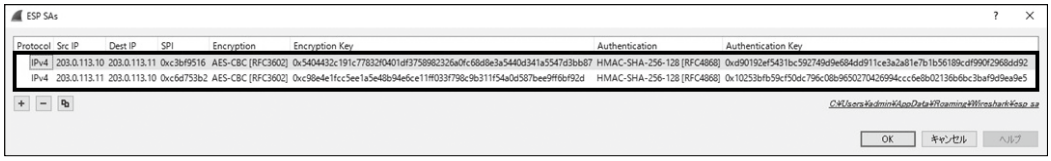
ESPパケットを復号する

WiresharkでESPパケットを復号するには［パケット一覧］ペインでNo.5のESPパケットをクリックしてから、［パケット詳細］ペインで［Encapsulating Security Payload］を右クリックしてコンテキストメニュー（**図5**）を表示します。コンテキストメニューの［プロトコル設定］にある、［Open Encapsulating Security Payload preferences］をクリックしてください。

［Wireshark・設定］ウィンドウ（**図6**）が表示されたら、［Attempt to detect/decode encrypted ESP payloads］にチェックを入れてから、ESP SAsの右にある［Edit］ボタンをクリックします。

［ESP SAs］ウィンドウ（**図7**）が表示されたら、左下の［+］ボタンをクリックしてESPパケットの復号に必要な情報を設定します。各項目をダブルクリックすると入力に切り替わります。**表2**と、「sd1704_ipsec.txt」ファイルを参照しながら各項目を入力してください。ドロップダウンリストが狭くて表示が省略されてしまう場合は、とりあえず値を選択してから隣の項目をクリックすると、サイズが調整されて値が表示されます。

▼図8　入力が終わった[ESP SAs]ウィンドウ



入力がすべて終わると**図8**のような内容になります。設定が正しいことを確認したら[OK]ボタンをクリックして[ESP SAs]ウィンドウを閉じます。さらに[Wireshark・設定]ウィンドウ(**図6**)の[OK]ボタンをクリックしてウィンドウを閉じます。Wiresharkのメインウィンドウが設定を反映した表示になっていないこともあるので、念のためメイン・ツールバーにある[このファイルを再読み込み]ボタン(**図9**)をクリックしてキャプチャファイルを再読み込みしてください。

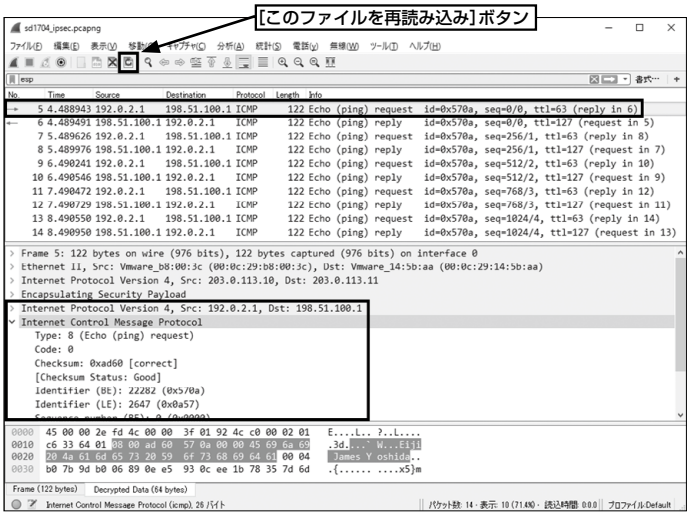
図9のようにNo.5のパケットの表示が変わって[パケット一覧]ペインのProtocol項目が**ESP**から**ICMP**になり、Info項目にICMPの情報が表示されています。さらに[パケット一覧]ペインでNo.5のパケットをクリックすると、[パケット詳細]ペインにある[Encapsulating Security Payload]の下に、新しく[Internet Protocol Version 4]と[Internet Control Message Protocol]が表示されています。この[Internet Control Message Protocol]を展開すると、ESPで暗号化されていたICMPパケットが復号され、pc1で**ping 198.51.100.1**を実行して送受信されたICMPパケットの内容が平文で表示されていることがわかると思います。

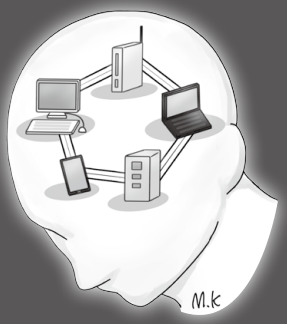
このように、WiresharkはIPsecの暗号化通信で流れるESPパケットを復号できます。[ESP SAs]ウィンドウで設定した内容は削除しない限り残るので、安全性を考えて設定を削除しましょう。先ほどの手順に従って[ESP SAs]ウィンドウ(**図7**)を表示したら、追加した行を選択し[−]ボタンをクリックすれば削除されます。あとは[ESP SAs]ウィンドウの[OK]ボタンをクリックしてウィンドウを閉じたら、[Wireshark・設定]ウィンドウ(**図6**)の[OK]ボタンをクリックしてウィンドウを閉じます。

おわりに

今回はWiresharkを使って、IPsecの暗号化通信を復号する方法について説明しました。最近ではクラウドに展開した環境にIPsecで接続する人も増えているので、今後はインシデント調査でIPsecの暗号化通信を復号したい場面も出てくると思います。そのときに本稿の内容が少しでもお役に立てば幸いです。SD

▼図9　復号されたESPパケット





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11回 仮想マシンネットワークの利用

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5回目です。

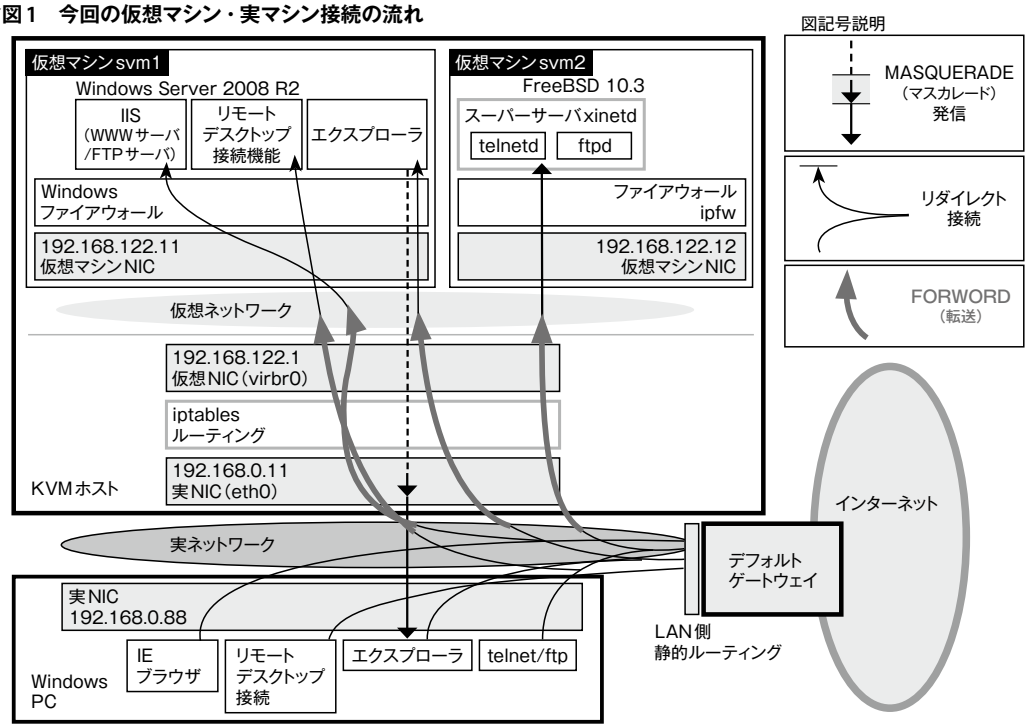
今回は、前々回(2017年2月号 第9回)および前回(2017年3月号 第10回)で作成・起動した「仮想マシンネットワークの利用」を、仮想マシン、KVM物理ホスト、実LAN、そしてインターネット、との間のコミュニケーションの面からいろいろ見てみます。

次のような環境下で、仮想マシンを利用した内外ネットワーク通信とインターネット通信を行います(**図1**)。

【今回使用する仮想ネットワーク環境】

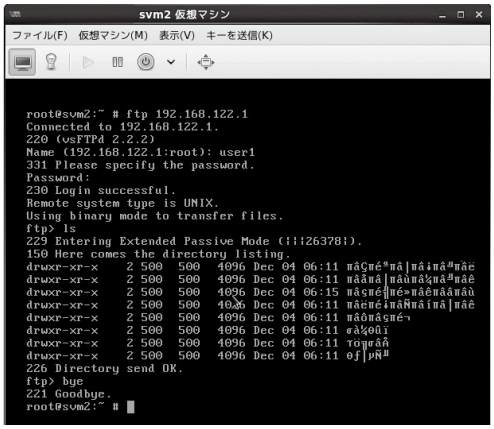
・仮想ネットワークアドレス：192.168.122.0/24
・仮想ネットワークドメイン名：example.com
・KVM物理ホスト：
　ホスト名＝vhost1.example.com
　IPアドレス＝192.168.0.11(実NIC)/

▼図1　今回の仮想マシン・実マシン接続の流れ



▼図2　svm1からKVM物理ホストへのtelnetアクセス

▼図3　svm2からKVM物理ホストへのftpアクセス



192.168.122.1（仮想NIC）

・仮想マシン1：

ホスト名＝svm1.example.com

IPアドレス＝192.168.122.11

OS＝Windows Server 2008 R2

・仮想マシン2：

ホスト名＝svm2.example.com

IPアドレス192.168.122.12

OS＝FreeBSD 10.3

仮想マシンから仮想マシン外へのアクセス

仮想マシン（svm1/2）から①KVM物理ホスト（vhost1）自体、②KVM物理ホスト側実LAN上のシステム、そして③インターネットへアクセスしてみます。

▼図4　svm1から実LAN上のWindowsへの共有ファイルアクセス

▼図5　svm1から技術評論社ホームページへのアクセス

①KVM物理ホストへのアクセス

仮想マシンからKVM物理ホスト（の仮想NIC側）へtelnet（**図2**）やftp（**図3**）で接続すると問題なくアクセスできました。なお、これは仮想LAN内での通信です。

②KVM物理ホスト側実LAN上のシステムへのアクセス

図4のように、仮想マシンsvm1からKVM物理ホスト側実LAN上のWindowsシステムへエクスプローラでアクセスしました。問題なくアクセスできます。

③インターネットアクセス

仮想マシンsvm1から技術評論社ホームペー

ジへアクセスしてみました(**図5**)。これも問題ありません。

仮想ネットワーク外から仮想マシンへのアクセス

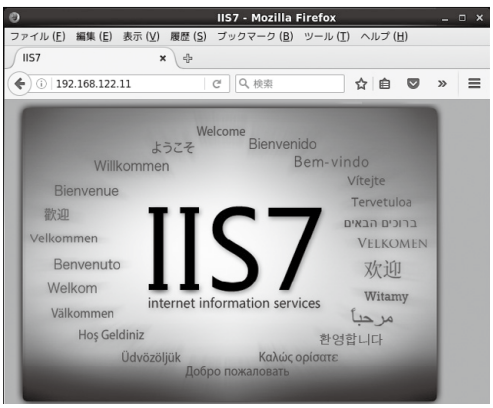
逆に外部から仮想マシンへ接続してみます。

なお、KVM物理ホストは物理LANと仮想マシン、両方のインターフェースを持つのでデフォルトで両方へアクセス可能です。実際、KVM物理ホストから仮想マシンsvm1へのホームページアクセスは**図6**のようになりました。

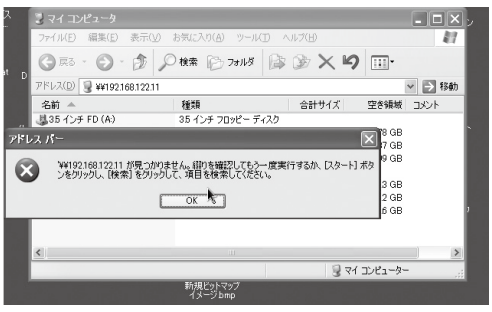
着信接続エラー

実LAN上のPCから仮想マシンへ接続してみると、エクスプローラ接続(**図7**)やリモートデスクトップ接続(**図8**)、telnetアクセス(**図9**)などすべて接続エラーとなってしまいます。

▼図6　KVM物理ホストからsvm1(IIS)へのホームページアクセス



▼図7　実LAN上のWindows PCからsvm1へのエクスプローラ接続



着信接続を可能にする

外部から仮想マシンへアクセスする場合には、(2016年11月号 第6回で解説したように)KVM物理ホストのファイアウォールでの着信接続転送設定と、仮想マシンのファイアウォールでの着信接続設定が必要です。さもないと、各々のネットワークアプリケーションでトラブルが生じます。

また、仮想マシンではアプリケーションのサーバ稼働も必須です。さらに忘れてはならないのが、実LAN側のパケットルーティングの設定です。

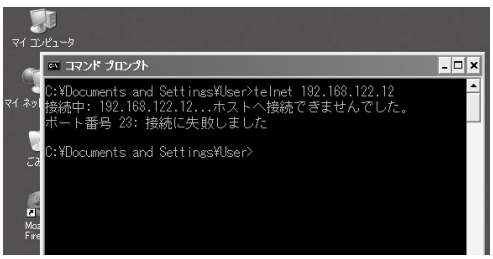
以下でこれらを説明します。

KVM物理ホストのファイアウォールでの着信接続転送許可

まず、KVM物理ホストのファイアウォール

▼図8　実LAN上のWindows PCからsvm1へのリモートデスクトップ接続

▼図9　実LAN上のWindows PCからsvm2へのtelnetアクセス



▼図10　仮想マシンネットワーク側への宛先着信をFORWARDする

```
 仮想ネットワークへのすべての着信転送を許可する
[root@vhost1 ~]# iptables -I FORWARD -d 192.168.122.0/24 -j ACCEPT
 設定を確認する
[root@vhost1 ~]# iptables --list -v
・・・
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  any    any     anywhere             192.168.122.0/24  ←ココ
   38  2888 ACCEPT     all  --  any    virbr0  anywhere             192.168.122.0/24    ↵
state RELATED,ESTABLISHED
   38  2888 ACCEPT     all  --  virbr0 any     192.168.122.0/24     anywhere
    0     0 ACCEPT     all  --  virbr0 virbr0  anywhere             anywhere
    0     0 REJECT     all  --  any    virbr0  anywhere             anywhere            ↵
reject-with icmp-port-unreachable
    0     0 REJECT     all  --  virbr0 any     anywhere             anywhere            ↵
reject-with icmp-port-unreachable
 ...略...
[root@vhost1 ~]#
```

iptablesで、仮想マシンネットワーク側への宛先着信をFORWARDする設定を行います(**図10**)。

仮想マシンでの着信接続許可

仮想マシンsvm1(Windows Server 2008 R2)のファイアウォールは、サーバや着信接続アプリケーションを有効にすることで自動的に許可設定されます。ただし、詳細な設定(発信やプロトコルなど)はWindowsファイアウォールや共有設定などで行います。

また、あらかじめパブリックからプライベートのネットワーク設定に変更しておきます。

仮想マシンsvm2(FreeBSD 10.3)では、ファイアウォールのデフォルトはすべて許可となっているので、(セキュリティ強化のためには)逆にipfwなどで「デフォルト：禁止」で「許可するものだけ」に設定します。

サーバ着信設定

仮想マシンでサーバアプリケーションを稼働させます。仮想マシンsvm1(Windows Server 2008 R2)ではIIS(Internet Information Server)-WWW/FTPサーバやリモートデスクトップ接続を稼働させ、ファイル共有サービスなどを有効にします。仮想マシンsvm2(FreeBSD 10.3)では、xinetdでtelnetdやftpdを稼働させます。

ルーティング設定

一般のTCP/IPシステムのゲートウェイはおもにインターネットへの接続に使用します。今回のネットワーク構成では、仮想ネットワークへのパケットルート(192.168.122.0/24への接続パケットはKVM物理ホスト192.168.0.11へ回す)を追加します。

ただし、接続するクライアント個別に設定するのは実際にはたいへんなので、192.168.0.0/24ネットワークのデフォルトゲートウェイに「static(静的)ルーティング」で、このFOR

▼図11　実LAN上のWindows PCからsvm1(IIS)へのホームページアクセス

▼図12　Windows PCから仮想マシンsvm2(FreeBSD 10.3)へのtelnetアクセス(ルーティング設定後)

```
C:¥Documents and Settings¥User> telnet 192.168.122.12

login: user1
Password for user1@svm2.example.com:
Last login: Sat Jan 28 17:24:05 from 192.168.122.1
FreeBSD 10.3-RELEASE (GENERIC) #0 r297264: Fri Mar 25 02:10:02 UTC 2016

Welcome to FreeBSD!

...

Edit /etc/motd to change this login announcement.
user1@svm2:~ %
```

▼図13　仮想マシン発信パケットの発信アドレス／ポートのマスカレード(見せかけ)

```
[root@vhost1 ~]# iptables --list -t nat
 ...略...
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24    masq ports: 1024-65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24    masq ports: 1024-65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24
 ...略...
[root@vhost1 ~]#
```

WARD設定をすることが実際的です。

実LAN側から仮想マシンへの接続テスト

以上のような転送、着信、およびルーティングの許可設定を行ってから、あらためて実LAN上のWindows PCから、仮想マシンsvm1(Windows Server 2008のIIS)へWebブラウザでアクセスしてみます(**図11**)。

また、**図12**のように仮想マシンsvm2(FreeBSD 10.3)へのtelnetアクセスもできました。

仮想マシン発信の接続に対する応答パケットの処理

それでは、仮想マシン側からの接続に対する応答パケットは(ルーティング設定を行う前に応答着信可能であったが)どのように処理されたのでしょうか。

仮想マシン発信のパケットは、実は、KVM物理ホストのiptablesで、KVM物理ホスト自身のIPアドレスを発信アドレスとして送出されています。

図13のように、iptablesのチェイン「POSTROUTING」のNATテーブルで、マスカレード(MASQUERADE)により仮想マシン側から発信する、"IPデータグラムの発信IPアドレスはすべてKVM物理ホストのIPアドレスを"、"アプリケーションはKVM物理ホストで空いているTCP/UDPポート(1024～65535)を"使用して発信したように「見せかけて」(IPマスカレード)いるのです。

これにより、実LAN上のKVM物理ホストと他のシステムとの間の通信になるので、ルーティング設定が不要になります。

この設定はlibvirtdサービスによって付け加えられたものです。

インターネットから仮想ネットワークへの接続

インターネットから仮想ネットワーク内サー

▼図14　外部クライアントPCからの通常のインターネットアクセスが遅い

```
C:¥Documents and Settings¥User>ping www.yahoo.co.jp
 応答が数分
Pinging www.g.yahoo.co.jp [183.79.196.239] with 32 bytes of data:

Reply from 183.79.196.239: bytes=32 time=24ms TTL=52
Reply from 183.79.196.239: bytes=32 time=23ms TTL=52
Reply from 183.79.196.239: bytes=32 time=23ms TTL=52
Reply from 183.79.196.239: bytes=32 time=23ms TTL=52
 ...略...
C:¥Documents and Settings¥User>ping www.yahoo.co.jp
 即応答（dnsキャッシュ）
Pinging www.g.yahoo.co.jp [183.79.43.200] with 32 bytes of data:

Reply from 183.79.43.200: bytes=32 time=24ms TTL=53
Reply from 183.79.43.200: bytes=32 time=22ms TTL=53
Reply from 183.79.43.200: bytes=32 time=22ms TTL=53
Reply from 183.79.43.200: bytes=32 time=22ms TTL=53
 ...略...
C:¥Documents and Settings¥User>
```

バへ着信する場合はどうなるのでしょうか。

サーバ着信なのでインターネットとLANとの間のルータで、NATまたはNAPTの接続着信を設定することで、宛先が仮想ネットワーク内のそのサーバとなります。

仮想マシン通信で起こる問題と対策

トラブルの多くは設定が必要な外部クライアント発信接続です。

外部クライアントから仮想ネットワーク内のサーバへ接続できない

複数の原因でサーバアクセスができない場合があります。その場合、先述の設定の1つ1つを確認しなければなりません。

外部クライアントからの通常のインターネットアクセスが遅い

KVM物理ホストと同じ実LAN上にあるWindowsで仮想マシン用にルーティング設定を追加すると、Windowsからの通常のインターネットアクセス（の最初）の反応が少し遅くなることがあります（**図14**）。

対処策

クライアントから最初のDNSサーバへの名前検索に時間がかかるためで、2回目以降はその名前検索情報はDNSキャッシュに保存されているので遅延は解消されます。

個別のクライアントにではなくデフォルトゲートウェイに静的ルーティングをするか、最初は少し待ち、あとは、DNSキャッシュを利用します。

次回予告

次回は、利用運用管理における、管理者が「行うべき」（あるいは「可能な」）作業範囲と仮想マシン／ゲストシステム利用者が「行うべき」（あるいは「可能な」）作業範囲を明確にし、その問題点や考え方を突き詰めて行きます。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

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14回 ExcelでSQL操作のイメージをつかむ法

「SQLは同じ性質を持ったデータの集合に対して同じ操作を一括適用するのが基本」というのは本連載第1回（本誌2016年3月号）で出てきた言葉です。しかし、この言葉の意味合いを理解し、使いこなすのはなかなか難しいものです。そこで今回は、SQLでデータを操作するイメージをつかむ方法を紹介します。

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。

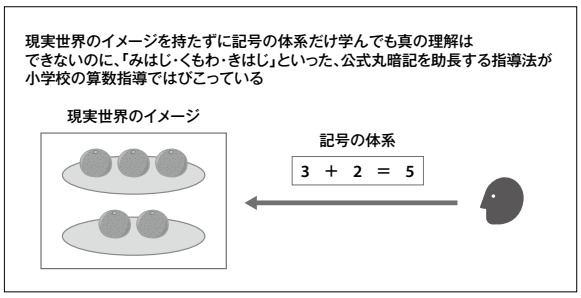
五代氏
大道君の上司。プロジェクトリーダーでもある。

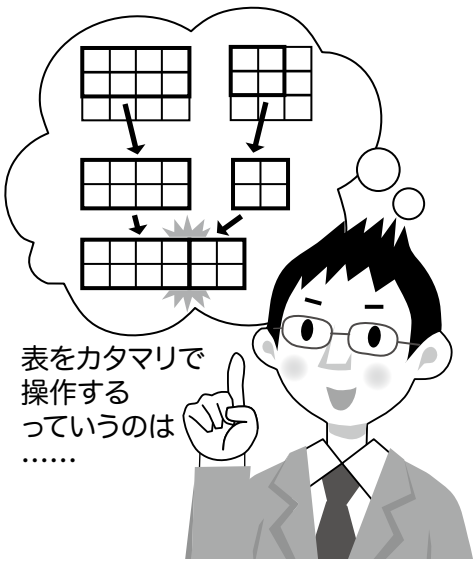
記号が表現している世界のイメージを持とう

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。

「勉強法を間違えると、いくら勉強しても成果が上がらないってこと、こんなところでもあるんですねえ～」と大道君が言い出しました。何のことかと思えば、小学生の算数の勉強について、**図1**のような問題が起きているそうです。たとえば「3 + 2 = 5」という足し算の式は現実世界で3個と2個のものを合わせる操作を抽象化して記号で表現していますが、式を使えるようになるためには、本来は、それが現実世界で何を表しているのかというイメージを持っていなければなりません。ところが、子供にイメージを持たせることをおろそかにして、手っ取り早く点を取るために公式の丸暗記を助長するような指導法が小学生への算数指導では蔓延（まんえん）しているそうです。DBの話ではないので本稿では詳しく書きませんが、興味のある方は「みはじ・くもわ・きはじ　批判」といったキーワードで検索してみてください。

▼図1　現実のイメージを持たずに記号の体系を暗記するのは無駄





「ああ、そうやね。SQLでもそうやったろ？SQLは文法から考えても理解できん。表をカタマリで操作するイメージを持つことが大事や」

というわけで今回は、「SQLで表を操作するイメージを持つ」ために役に立つ、ある変わった勉強法を紹介しましょう。

複雑な場合分けをパラメータで処理

まずはこんな例を考えてみてください。携帯電話の通話料金を計算する処理です。

「携帯電話の料金プランって、固定料金制と従量制とを組み合わせた、こういう感じのがよくあるやろ？」

図2－①の「契約プラン仕様」がその例で、契約Bが夜間かけ放題、契約Cは昼間かけ放題のプランになります。もちろん実際の携帯電話会社ではもっと複雑なプランが提供されていますが、本稿では簡単な例を示しています。

一方、通話記録はプランとは別に単純に契約者IDと開始時間、終了時間で記録されているとします(**図2－②**)。すると月末には料金請求のために通話記録ごとに従量課金額を計算しなければなりません。ロジックとしては、通話記録1件ごとに「かけ放題」対象かどうかを判定し、かけ放題対象なら0円、従量課金対象なら通話分数を算出し契約プランごとに異なる単価を掛けて1通話ごとの従量課金額を算出します。かけ放題対象かどうかを判断するには、契約者テーブル(**図2－③**)を参照してプランを調べ、通話開始時間がそのプランのかけ放題時間に含まれるかどうかを判断します。最終的に、契約者IDごとの通話分数、プラン、基本料金、従量単価、課金分数、従量課金額をまとめた「課金テーブル」(**図2－④**)を作ることが目標です。

「まあ、今の携帯電話サービスだと一通話終わるごとに計算してたりするけどな。話の都合上、月末締め処理で一気にやるものと考えてみてや。こういう処理をするなら、どういうふうに作る？」

「えーと、これをSQLでやるとすると、契約プランによってロジックが違うわけですから、直感的には、ABCそれぞれの処理用のSQLを作って3回発行すれば、とか考えちゃいますが……」

「ほとんどの人はそう考えるんやけど、それは典型的な手続き型の発想で、SQL的じゃないんよね～」

その方法でもできますが、通話記録という件数の多いテーブルを3回走査することになるため、性能問題が起きやすいのと、メンテナンスしなければいけないドキュメントやSQL文が3つ以上に分散するため、あまりお勧めできません。

▼図2　月末の料金集計処理

①契約プラン仕様

契約A
　基本料金1,000円
　いつでも1分15円

契約B
　基本料金5,000円
　22時～翌朝6時はかけ放題
　上記以外の時間は1分30円

契約C
　基本料金7,000円
　朝9時～夜22時はかけ放題
　上記以外の時間は1分20円

※通話開始時間が含まれるかどうかで、かけ放題の時間を判断する

②通話記録テーブル

契約者ID	開始時間	終了時間
100000001	2016/6/2 12:31:16	2016/6/2 12:38:16
100000001	2016/6/3 13:26:07	2016/6/3 13:30:59
100000001	2016/6/9 23:01:59	2016/6/9 23:05:11

③契約者テーブル

契約者ID	名前	プラン
100000001	佐藤 悠太	A
100000002	鈴木 優子	B
100000003	高橋 博	C

④課金テーブル

契約者ID	通話時間	通話分数	プラン	基本料金	従量単価	課金分数	従量課金額
100000001	0:16:30	17	A	1000	15	16	240
100000002	1:07:03	68	B	5000	30	64	1920
100000003	1:08:00	68	C	7000	20	2	40

CASE式にパラメータテーブルを組み合わせる

「これも1本でいくんですか……あ、CASE式を使うんですか？」

「そのとおり！　具体的にはこんなコードでいけるよ」

当連載の第7回（本誌2016年9月号）で触れたCASE式を使うと、このぐらいの場合分け処理は十分可能です。**リスト1**がそのSQL文です。ここで重要なのが**図3**に示すパラメータテーブルで、契約プランの仕様をテーブルの形で表現したものです。プランABCをキーにして通話記録および契約者テーブルをJOINすることにより、個々の通話記録に適用されるパラメータが自動的に選択されます。このため、CASE式で書かれた「課金分数」を判定するロジックをプランABCすべてで共通化することができます。

「あ、本当だ……CASE式中の開始時間1、2や終了時間1、2は契約プランに応じて変わるから……ABCの全部をこれ1本で処理できるんですね」

「かけ放題時間が日をまたぐ部分の処理に注意がいるけど、基本のロジックがこのとおりなら、新しい料金プランができてもパラメータテーブルを修正するだけで済むんよ」

「携帯電話の料金って、○○分までを越えたら超過料金がかかったりするのがあるじゃないですか。そういうのもこれでいけますか」

「もちろんいけるよ。試しに、かけ放題時間の通話が400分を越えたら超過単価が1分5円または3円ずつかかる設定でいこうか。その場合はパラメータテーブルを**図4**のように変えて、SQLは**リスト2**に変える。これで通常の従量料金分と超過料金分を別

▼図3　パラメータテーブル（かけ放題対象は完全固定料金型）

プラン	基本料金	従量単価	開始時間1	終了時間1	開始時間2	終了時間2
A	1000	15				
B	5000	30	22:00:00	23:59:59	0:00:00	6:00:00
C	7000	20	9:00:00	22:00:00		

▼リスト1　CASE式で処理

```
SELECT
    m.* -- サブクエリ中の全項目を列挙
    , 課金分数 * 従量単価 AS 従量課金
FROM
    (SELECT
        a.契約者ID
        , SUM(a.終了時間 - a.開始時間) AS 通話時間
        , SUM(MINUTE(a.終了時間 - a.開始時間 + 59秒)) AS 通話分数
        , b.プラン
        , c.基本料金
        , c.従量単価
        , SUM(CASE
            WHEN (a.開始時間 >= c.開始時間1 AND a.開始時間 <= c.終了時間1)
                OR (a.開始時間 >= c.開始時間2 AND a.開始時間 <= c.終了時間2) THEN 0
            ELSE
                MINUTE(a.終了時間 - a.開始時間 + 59秒)
            END) AS 課金分数
    FROM
        通話記録 a
        INNER JOIN 契約者 b
            ON a.契約者ID  = b.契約者ID
        INNER JOIN 料金プランパラメータ c
            ON b.プラン = c.プラン
    GROUP BY
        a.契約者ID
        , b.プラン
        , c.基本料金
        , c.従量単価) m
    ;
```

※このサンプルでは日付関数は省略している

通話開始時刻がかけ放題時間に入っていれば、課金分数はゼロ

課金対象なら1分未満切り上げで課金分数を計算

料金プラン別のパラメータテーブルをJOINしておく

にカウントできる」

「おおー!!　なるほど！　これもやっぱり、課金ロジックがこのパターンにはまる限り、プランが増えてもパラメータテーブルを変えるだけで済むんですね！」

「そういうこっちゃ。今どきの携帯電話はもっとプランの種類が多いやろ？　種類が多ければ多いほど、パラメータの変更だけで済むから楽なんよ」

2万ステップのJavaがたった3つのSQLに?

「パラメータテーブルを作ってJOINで場合分けを処理してしまうなんて、こんな発想、考えもしなかったです」

「手続き型の感覚だと、どうしてもプランで場合分けする方向で考えちゃうからねえ。この方法は誰かに教えられないとなかなか思いつけないけど、こういう細かい条件で精算処理の仕様を変えるような業務は実務的にはえらく多いから、知っておくと役に立つで〜」

私の経験では、27パターンある仕訳を2万ステップのJavaで書いた処理の改修を依頼されたことがあります。そのコードの実装と数百ページにもなる詳細設計書の整理には10人月前後を要したそうで、当然バグも多く、ほんの少しの仕様変更にも膨大な工数がかかり、実行も遅いものでした。しかし最終的にはそれを3つのパラメータテーブルと3つのSQLで処理するように改修できました。仕様書を理解するのに数日かかりましたが、SQLを書くこと自体は1日も要さず、性能はざっと100倍に上がりました。しかも複雑とはいえたった3つのSQLと、2万ステップのソースとを比べれば、圧倒的にSQL

▼図4　パラメータテーブル（かけ放題に制限があり超過料金がかかる型）

プラン	基本料金	従量単価	開始時間1	終了時間1	開始時間2	終了時間2	最大かけ放題時間	超過単価
A	1000	15						
B	5000	30	22:00:00	23:59:59	0:00:00	6:00:00	400	5
C	7000	20	9:00:00	22:00:00			500	3

▼リスト2　超過料金計算を処理するSQL

※このサンプルでは日付関数は省略している

```
SELECT
    m.* -- サブクエリ中の全項目を列挙
    , CASE WHEN 放題分数 > 最大かけ放題時間 THEN
            (放題分数 - 最大かけ放題時間) * 超過単価
        ELSE 0 END AS 超過課金 ←― かけ放題通話分数が制限時間を越えていれば超過課金を計算
    , 課金分数 * 従量単価 AS 従量課金
FROM
    (SELECT
        a.契約者ID
        , SUM(a.終了時間 - a.開始時間) AS 通話時間
        , SUM(MINUTE(a.終了時間 - a.開始時間 + 59秒)) AS 通話分数
        , b.プラン, c.基本料金, c.従量単価, c.最大かけ放題時間, c.超過単価
        , SUM(CASE
            WHEN (a.開始時間 >= c.開始時間1 AND a.開始時間 <= c.終了時間1)
                OR (a.開始時間 >= c.開始時間2 AND a.開始時間 <= c.終了時間2) THEN
                    MINUTE(a.終了時間 - a.開始時間 + 59秒)
            ELSE 0 END) AS 放題分数 ←― かけ放題時間の通話分数を算出
        , SUM(CASE
            WHEN (a.開始時間 >= c.開始時間1 AND a.開始時間 <= c.終了時間1)
                OR (a.開始時間 >= c.開始時間2 AND a.開始時間 <= c.終了時間2) THEN 0
            ELSE
                MINUTE(a.終了時間 - a.開始時間 + 59秒) END) AS 課金分数
    FROM
(以下リスト1と同じにつき省略)
```

のほうがメンテナンスしやすくなります。

「うーん、でも……」と、そこまでは感心していた大道君ですが、何やら気になることがある様子。

「ただ、やっぱり一見して長くて複雑なSQL文ですし、こういうコードを使いたがらない、SQL嫌いな人はいると思うんですよね」

「そりゃあ、どうしてもいるなあ。だからDB担当を分けてAPIファースト開発[注1]をやろうや、と言うてるんやけど」

「ええ、ただ、彼らの気持ちもわかるんですよね。CASE式とかもちゃんと勉強して生島さんの説明も聞かないと、**リスト1、2**みたいなSQL文って、パッと見ただけでは何をやっているのかわけがわからないんですよ。こういうSQL処理のイメージが持てるようになる方法って何かないんでしょうかね……」

「おお、それな……実はあるんやで！　ああいうSQLのイメージがつかめて自然に書けるようになる方法が！」

「え！　どんな方法なんですか？」

Excel計算式でSQLの感覚をつかむ法

この方法はExcelを使います。段取りは**図5**のとおりで、Excelで「基になる表」の横に❶関連データを付加し、それらのデータを❷判定・演算して導出項目を出し、それを❸集計して目的のデータを作る、という操作をExcelの計算式／関数を使って「1枚の表になるように」作ると、それがSQL文のSELECT、JOINやCASE式などに対応するため、SQLでやっていることのイメージがつかめるようになります。

具体例が**図6**です。まず前提として、**図2**に示した通話記録テーブルと契約者テーブル、**図3**に示したパラメータテーブルはExcel上に個別の表（シート）として作ってあると思ってください。**図6**で「基になる表」と記した部分は通話記録テーブルのことで、ここにまずそれぞれの通話がどのプランなのかを示す「プラン」を関連データ1として付加します。SQLなら契約者IDをキーにして契約者テーブルとJOINすればプランがわかりますが、Excelでは同じ処理をVLOOKUP関数で行います。導出項目1の内容は「基になる表」の部分から計算式で算出します。

「関連データ2」の部分はパラメータテーブルの情報であり、これは契約プラン名をキーにしてパラメータテーブル表をVLOOKUPで検索して出します。こうして必要な関連データや導出項目を1行にまとめて出してから、それを使って導出項目2を計算するわけです。

Excelを使うと表の操作が見えてくる

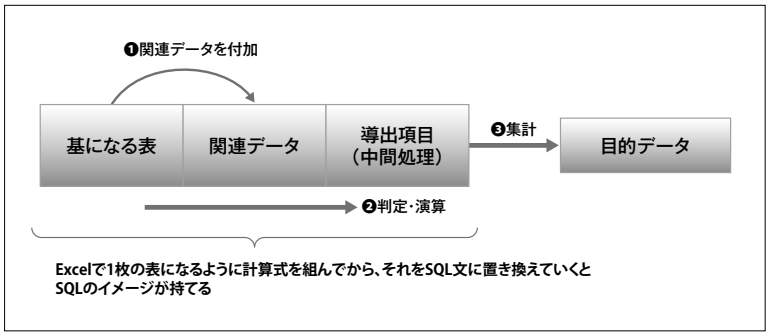
「あ……これってつまり、『現実世界』を見えるようにしてるんですね？」

「そういうこと！　これで感覚つかんでおけば、SQL文を読むときはExcelのイメージに戻しながら読めるから楽になるんよ」

リスト1、2のようなSQL文は、文法的な知識で読もうとしてもよくわかりません。けれど、やっていることは要するにこういうことなんです。JOINをするというのは必要なデータを1枚の表にまとめるということであ

注1）本連載第6回（本誌2016年8月号）、第9回（2016年11月号）を参照。

▼図5　Excel操作でイメージを持ってからSQL文に変換する方法



り、WHEREやCASEを使うというのはExcelで言えばIF関数でフィルタリングや場合分けをすることなので、このようなExcel操作で「何をやっているのかのイメージをつかむ」ことができれば、複雑なSQL文を読むのも書くのも非常に楽になります。

事務オペレータのほうが速くSQLを習得できる?

実は派遣で事務職をしているオペレータの人々は、ExcelでVLOOKUP関数やIF関数を使ってこのような処理をするのに慣れています。彼らはExcelで複数の表を突き合わせてこの種の作業をすることが多いので、SQLのイメージはすでにできていますから習得が非常に速いです。2日間のセミナーに参加された事務職のオペレータは、2日目にはDML(Data Manipulation Language)[注2]について、私と同じレベルでできるようになった人もいました。

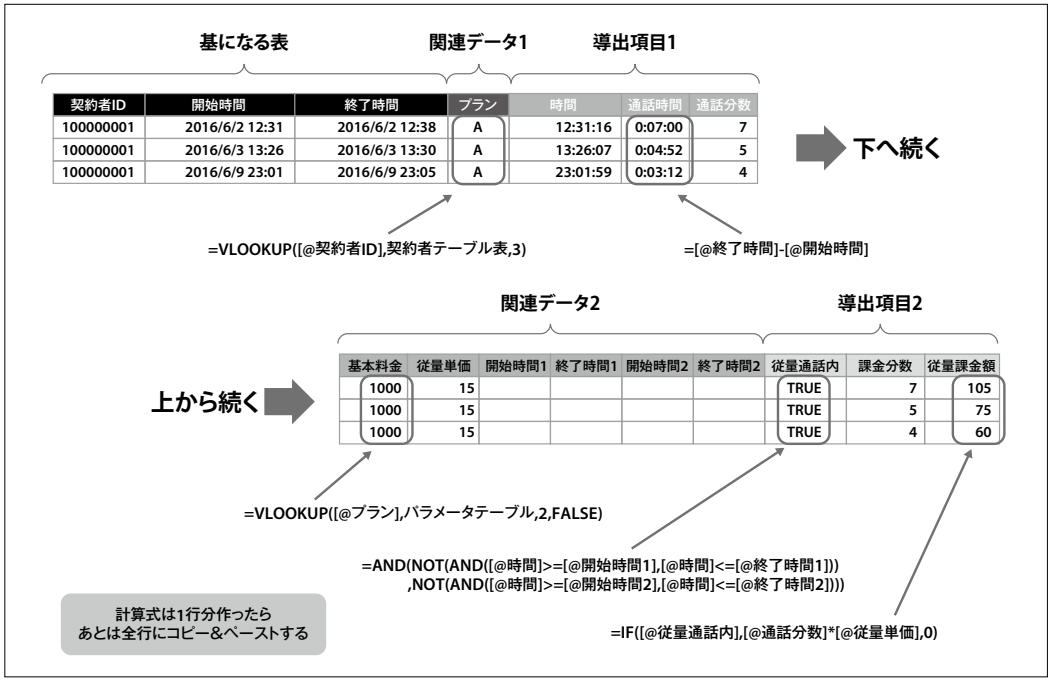
注2) SQLのうち、抽出/追加/更新/削除などデータを操作するための言語。一方、データの構造を定義する言語をDDL(Data Definition Language)と言う。

図6にはExcel計算式を一部だけ載せておきましたが、とくに論理式やIF関数の記述などはセルの中に書く関係上、かえってSQLよりも読みづらいことがわかると思います。この点ではSQLのほうがExcelよりも簡単なのです。実際のところ事務処理業務に関してExcelでできないことはほとんどありませんが、それはつまりほとんどの処理はSQLのほうが簡単にできるということでもあります。

結局SQLはこの種の操作をすることに特化して設計された言語なのですから、それを使わずに、PHPだろうとJavaやC#だろうと(オブジェクト指向的とは言っても本質的には)手続き型の言語を使ってこの種の処理をするのはお勧めできません。

そしてSQLの習得には「1枚の表にデータを集めて操作するイメージ」を持って考えることが重要で、そのために役に立つのが今回紹介したExcel計算式でSQL相当の処理を作ってみる方法です。SQLの働きがピンとこないときにはぜひ試してみてください。SD

▼図6 Excel計算式でSQL感覚をつかむ表



一歩進んだ使い方のためのイロハ

Vimの細道

第17回

置換コマンドの基本をおさらい

mattn
twitter:@mattn_jp

長いテキスト内で一括して変換を行いたい場合に便利な、Vimの置換機能を紹介します。置換コマンドの組み立て方やメタ文字、ビジュアルモードでの置換方法といった基礎をおさらいしたあとは、応用的な置換術として、式を使った置換なども紹介します。

基礎的な置換術

Vimはモードのあるテキストエディタです。インサートモード、ノーマルモード、コマンドモードをそれぞれ行き来してテキストを編集します。カーソル付近の小さな変更であればインサートモードで書き換えてしまうのが手っ取り早いです。またカーソル付近のテキストを移動したり削除するのであれば、ノーマルモードで編集するのが効率的でしょう。そしてテキスト全体を編集する場合には、コマンドモードを使うと便利です。

とくにテキスト全体で文章を置き換えたい場合には、置換コマンドを使うことになります。置換コマンドは置き換え前のテキストと、置き換えた後のテキストを指定して行いますが、Vimの便利な機能を使いこなすことで高度な操作も扱えるようになります。今回はこの置換コマンドの便利な使い方を紹介します。置換コマンドなんて初歩的だと思っている方も、再度見直してみると実は使っていなかった便利な機能に気づくことがあります。

簡単な文字列の置換

まずは簡単な置換コマンドを例に挙げておさらいをしてみます。

```
Hallo World, Hallo Japan.
```

このテキストの「Hallo」を「Hello」に修正するにはどうすれば良いでしょうか。インサートモードで「a」を消し、「e」を打ち直すのも良いでしょう。ノーマルモードで「a」の上に行き、reとタイプするのも良いでしょう。しかし、このタイプミスがテキストファイル全体に渡って何度も書かれていたら、置換を使うほうが効率的です。

```
:s/Hallo/Hello/g
```

簡単ですね。最後のgは、その行の中で現れるすべての「Hallo」に対して置換を行うというフラグになります。このフラグを付けない場合、行の中で最初に現れた「Hallo」だけが置換されます。またこのテキストが数十行あり、ファイル内のすべての「Hallo」を「Hello」に置換したいのであれば次のように実行します。

```
:%s/Hallo/Hello/g
```

%はすべての行を意味するrange（範囲）を表します。rangeは**表1**の形式を取ることができます。たとえばファイルの先頭行から10行目までに現れる「Hallo」を「Hello」に置換したいのであれば次を実行します。

```
:1,10s/Hallo/Hello/g
```

▼表1　rangeの取る形式

range	意味
:%s/	全行を置換
:4s/	4行目を置換
:4,7s/	4行目から7行目を置換
:.,$s/	現在行から最終行を置換
:'<,'>s/	ラインワイズ選択(後述)を置換
:/PAT/s/	PATが含まれる行を置換

▼表2　ignorecaseとsmartcaseの挙動

パターン	ignorecase	smartcase	マッチ
foo	OFF	-	foo
foo	ON	-	foo Foo FOO
Foo	ON	OFF	foo Foo FOO
Foo	ON	ON	Foo
\cfoo	-	-	foo Foo FOO
foo\C	-	-	foo

「-」はON/OFFどちらでも可。

大文字と小文字のマッチ

次の例を見てください。

```
foo bar FOO bar Foo
```

設定ファイルで、set ignorecaseとオプションが設定された状態で:s/foo/bar/gを実行すると小文字の「foo」と大文字の「FOO」、両方混じった「Foo」も「bar」へ置換されます。これが意図した置換であれば良いのですが、気づかず置換されてしまったならば事故になります。そうならないためにもignorecaseとsmartcaseの動作を知っておいたほうが良いでしょう。前者は大文字と小文字を区別せずに検索し、後者は大文字と小文字が混じったパターンで検索した場合にのみ、大文字と小文字を区別して検索します。**表2**に、この2つのオプションの設定内容に伴うパターンとマッチの関係を示します。

これらは/による検索だけでなく:sコマンドのパターン指定にも適用されます。ignorecaseやsmartcaseを無効にして検索するには\Cを付けます。また逆に、ignorecaseやsmartcaseを指定した状態と同じ検索をするには\cを付けます。さきほどのテキスト内の「foo」のみを「bar」に置換するのであれば、次のように置換します。

```
:%s/foo\C/bar/g
```

これらは、置換コマンドのフラグ:%s/foo/bar/giと:%s/foo/bar/gIを使うことでも指定できます。

メタ文字マッチの回避

ここでちょっと、メタ文字について説明したいと思います。Vimの/での検索や、:sコマンドのパターンでは、正規表現を指定します。したがって、正規表現で使用するメタ文字を検索文字としてそのまま使うことはできません。使いたい場合には\でエスケープする必要があります。たとえば.という文字を検索したいのであれば\.を入力します。Vimの正規表現は一般的な正規表現と少し異なるため、エスケープには少し慣れが必要です。

一般的なテキストエディタで正規表現を使用する場合、次のメタ文字がエスケープ対象となります。

```
\ . + * ? ( ) | [ ] { } ^ $
```

Vimの場合、次のメタ文字が正規表現でエスケープ対象となります(エスケープが必要な文字の一部です)。

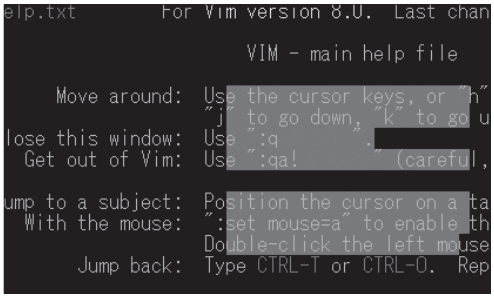
```
\ . [ ] ^ $
```

これらのメタ文字はmagicオプションにより動作を変えられます。詳しくは:help /magicを参照ください。

置換コマンドのcフラグ

一般的なテキストエディタでは、表示されたダイアログ上で「次を置換」「すべて置換」といったボタンをユーザがクリックしながら置換を行います。Vimにも同様な機能があります。

▼図1　ブロックワイズの選択例




```
foo bar foo
```

このテキストに対して:%s/foo/bar/cコマンドを実行するとカーソル位置の「foo」を置換するのかスキップするのか、それとも残りすべて置換するのか、または中断するのかを、都度確認しながら進められます。

そのほかにも、Vimにはたくさんの置換コマンドフラグが存在します。詳しくは:help :s_flagsを参照ください。

単語の区切りを指定する

さてこれまでの検索だと、「foobar」のようにつながった「foo」も置換対象になってしまいます。たとえば次のテキストで、「$」と「/」で囲まれた「foo」だけをパターンとして扱う場合にはどうすれば良いでしょうか。

```
!bar$foo/foobar_barfoo#
```

/fooで検索してしまうと、「foobar」や「barfoo」にもマッチしてしまいます。実際に検索したいのは「foo」の前後がアルファベットでないというルールになるので、次のように検索します。

```
/\<foo\>
```

大文字と小文字を区別したいなら/\<foo\>\Cで検索します。

ビジュアル選択した部分だけで置換

まず、ビジュアル選択とは何かを説明する必要があります。Vimのビジュアル選択には3つの選択方法があります。

▼リスト1　サンプルテキスト

```
##################################
#                                #
# #      ####  ######## #    # #
# #     #    #    #    #    # #
# #     #         #    #  ## # #
# #     #  ###    #    # ## # #
# #     #    #    #    #    # #
# #     #   ##    #    #    # #
# ######  ### #    #    #    # #
#                                #
##################################
```

・キャラクタワイズ
・ラインワイズ
・ブロックワイズ

キャラクタワイズは行内の文字を選択する方法です。行の途中からvをタイプして選択を始めます。yをタイプしてヤンクすると、選択した文字がコピー(レジスタに格納)されます。

ラインワイズはVをタイプして行の選択を開始します。同様にyで改行付きの行がコピーされます。

ブロックワイズは複数行のあるテキストの一部分を選択する方法で、Ctrl-vをタイプして選択を開始します(**図1**)。

このいずれかのビジュアル選択をした状態で、その選択領域内でだけ置換を行いたい場合には\%Vを使います。たとえば**リスト1**を見てください。こんなテキストはネタでしか使いませんが、たとえばこのテキストの内側の「#」だけを「@」に書き換えたい場合は、左上の内側でCtrl-vでブロックワイズ選択モードに入り、右下の内側までを選択します。その状態で:をタイプすると、:'<,'>までが自動で入力されるので、あとは次のように\%Vを使った置換コマンドを実行します。

```
:'<,'>s/\%V#/@/g
```

これにより「#」で囲まれた内側の「#」だけが「@」で置き換えられます。

一歩進んだ置換術

通常の置換は、あるパターンにマッチするテキストを別のテキストに置き換えるものですが、Vimの置換はもう少しリッチな置換が行えます。

大文字に置換する

置換後の文字列を大文字にする場合、置換文字列に\uまたは\Uを付与します。

```
pen pineapple apple pen
```

このテキストの「pen」を大文字に置換して、「PEN pineapple apple PEN」にしたいのであれば次を実行します。

```
:s/pen/\U&/g
```

また先のテキストを、「Pen pineapple apple Pen」という先頭大文字の文章に置換したいのであれば次を実行します。

```
:s/pen/\u&/g
```

Vimの置換文字列には、いろいろな参照方法(この例だと&)や修飾(\Uや\u)を行えます。**表3**にその一部をまとめます。

▼表3　置換文字列の参照方法

特殊なサブ置換	意味
&、\0	マッチした行全体に置き換え
\&	「&」自身に置き換え
\1〜\9	サブマッチに置き換え
~	前回実行した置き換え文字列で置き換え
\~	「~」自身に置き換え
\u	次の文字を大文字に置き換え
\U	\Eが現れるまで後続の文字を大文字に置き換え
\l	次の文字を小文字に置き換え
\L	\Eが現れるまで後続の文字を小文字に置き換え
\e、\E	\u、\U、\l、\Lの終端
^M、\r、\<CR>	この位置で行を分割 (^Mは Ctrl -V- Enter で入力)
\r	^Mと同じ
\n	＜NUL＞文字を挿入
\b	＜BS＞文字を挿入
\t	＜TAB＞文字を挿入
\\	「\」を挿入

式で置換する

Vimの置換はこれだけでも十分便利なのですが、ここで紹介する式の置換を使うことで、ほぼ何でもできるようになります。

たとえば、文章の中に登場する西暦を一括で+1するにはどうしたら良いでしょうか。ほかのテキストエディタでここまでできるものは、実はあまりありません。Vimにおいてはそれほど難しくないのですが、Vim独特の指定方法が必要になります。以降で作りながら説明します。

まず西暦にマッチするパターンを作ります。西暦は4桁の数字とします。今回のように西暦を+1したいような要件は、だいたいの場合、年度末に作る資料の更新作業です。3桁以下ということはまずないでしょう。だからといって、文書中に登場する4桁の数字を西暦とみなしてしまうのは危険です。一括置換で予想していなかった結果になってしまいます。「年」という文字が続くかどうかを見ておいたほうが良いでしょう。もちろんこのパターンであってもたとえば「今から5000年前」といったテキストにもマッチし得ます。置換の際にはテキストの中にそういった文言が登場しないことを確信したうえで実行しましょう。

Vimの正規表現で4桁の数字は\d\{4}で表せます。\dは数字を意味し、[0-9]と等価になります[注1]。

\{n}は量指定子と言い、前のパターンの個数を示せます。今回は4桁の数字を西暦として扱いますが、3桁も扱いたいということであれば、\d\{3,4}と書くことで3桁以上、4桁以下の数字にマッチします。量指定子については:help \{でヘルプを参照することができます。とても便利な機能ですので、別の号でも紹介したいと

注1) 実際には先頭が0で始まる西暦はありませんから、本来は[1-9][0-9]\{3}と限定して入力すべきです。

思います。

さて、\d\{4}ですとたとえば、「10000」という5桁の数字の「1000」の部分にもマッチしてしまいます。単語の区切りを\<\d\{4}\>と与えることで4桁に限定しましょう。

パターンができたら置換後の式のことを考えます。数字4つの部分を得たいので\(\)を使ってグルーピングします。

```
:%s/\<\(\d\{4}\)\>/何か/g
```

サブマッチした部分は、式の中でsubmatch関数を使って得ることができます。引数は\1や\2で使用する番号と同じになります。Vim scriptは文字列が数字のみであればそのまま演算することができるので、+1する式はsubmatch(1)+1と書けます。置換後の文字列に式を使うには、:s/PATTERN/\=submatch(N)/を使います。よって4桁の数字を+1する正規表現は次のように書けます。

```
:%s/\<\(\d\{4}\)\>/\=submatch(1)+1/g
```

なお、サブマッチで参照する必要がない場合には\%(パターン)を使えます。このグルーピングは\1やsubmatch(N)の対象からスキップすることができます。パターンを構築したあと、マッチし得る候補が増えてしまった場合に、いちいち参照番号をずらさなくても良くなります。

```
^\%(パソコン\|PC\)を\(10万円\)で買いました
```

このパターンの場合、\1やsubmatch(1)は「10万円」を参照します。

\zs、\ze

さて、ここまでで、できあがったのは「4桁の数字を+1する置換コマンド」です。前述のとおり「年」を足すことで西暦しばりにする対応を入れましょう。しかしパターンの中に純粋に「年」を入れてしまうと、置換後の文字列にも「年」を入れないといけません。そうしないと「年」が消えてしまいますからね。

```
:%s/\<\(\d\{4}\)\>年/\=submatch(1)+1 .'年'/g
```

2ヵ所に「年」が登場し、あまりかっこよくありませんね。もし検出したい文字列が「年」だけではなく、もっと長いものだったらたいへんです。そこで便利なのが、\zsと\zeです。それぞれパターンにマッチする開始／終端位置を示します。ですので、Make \zsAmerica\ze Great Againは「Make」が先行し「Great Again」が後続する「America」の部分だけにマッチします。

これは一般的な正規表現での、肯定先読み(?=regex)と肯定後読み(?<=regex)に該当しますが、括弧を付ける必要もなく、可読性が高いのがメリットです。置換文字列で前述のような「年」を両方に入れないといけない問題もなくなります。よってこれまで説明してきた「西暦を+1する置換コマンド」は次となります。

```
:%s/\<\(\d\{4}\)\ze年/\=submatch(1)+1/g
```

単語区切りの\>は、「年」が続くので必要なくなりますね。

s!パターン!文字列!

正規表現でUNIXのパス「/usr/local/bin」を置換する場合、:%s/\/usr\/local\/bin/のようにエスケープ文字を多用しないといけません。そんな場合は置換コマンドのセパレータに/ではなく！や#のようなほかの記号を使えます。

```
:%s!/usr/local/bin!$(INSTALDIR)!g
```

可読性も良くなりミスも少なくなりますね。

◆ ◆ ◆

今回紹介した置換コマンドはVimの置換コマンドが持っている機能の一部にしか過ぎません。だからといってすべてを覚える必要はありません。便利な機能をかい摘んで自分のお気に入りにしてしまうのが、その機能を覚える一番の方法だと思っています。今回紹介しきれなかった正規表現パターンや置換コマンドの拡張方法には、また別の号で紹介したいと思います。SD

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/

第35回 安定の横断検索パッケージanything.el

複数の情報源に対して検索・各種操作を行えるパッケージ「anything.el」を紹介します。同型のパッケージhelmと比べ、安定した動作が強みです。メンテナンスを担当している筆者だからこそ言える、anything.elのミッションについても注目。

帰ってきたanything.el

ども、るびきちです。現在人気のEmacsパッケージであるhelmは、早いもので誕生10周年になります。helmはかつてanything.elという名前で開発されていました。anything.el登場時は話題沸騰し、多くのEmacsユーザの関心を集めました。anything.elはunite.vimとして移植され、anything.elを参考にしたUIのシェルプログラムも数々開発されました。

2012年からは、helmという名前でforkされて生き続けています。筆者は9年前に原作者から開発を引き継ぎ、今なおanything.elを開発しています。helmの機能のほぼすべては、anything.elの段階で完成していました。

anything.elはhelmと袂(たもと)を分かってから、更新頻度がめっきり少なくなりました。そのうちanything.elはhelmの陰に隠れることになり、誰からも見向きをされなくなりました。2016年にあらためてanything.elのミッション・ビジョンが明確になりましたので、復活宣言をしました。

anything.elとhelmはユーザ側からはほとんど違いがわかりません。具体的にどういうものであるかは、過去記事[注1]を見ていただけると理解できるでしょう。

注1）URL http://emacs.rubikitch.com/sd1503-helm/

anything.elのミッション

helmはものすごい勢いで開発されています。helmが大人気になった理由の1つとして、高い更新頻度による露出もあります。ただ、筆者もいちhelmユーザとして観察していると、あまりにもスピードが速過ぎてユーザがついていけてないという印象を持っています。

helmの内部はかなり複雑になりました。加えて、予告なしに何度もインターフェースを変更したり関数・コマンドを削除したりするため、そのたびに混乱させられました。

Emacsを手足のごとく扱い、書籍・連載を執筆している筆者でさえも対応に困窮することがありました。Emacs初中級者の場合はもっと時間がかかりますし、場合によっては使うのを諦めることになってしまいます。正直、helmの開発はひとりよがりだと言わざるを得ません。

helmの現状を見て、あらためてanything.elのミッションが見えてきました。anything.elは筆者が開発に関わる前から優れたフレームワークです。情報源を組み合わせてオリジナルのコマンドを作ったり、適切なタイミングで候補を表示してくれたりと、Emacs Lisp初中級者でも簡単に便利機能が自作できるのです。

そこでanything.elは次のミッションを掲げることにしました。

初中級者にも安心と機能性を提供するEmacs Lisp学習キット

Emacs Lisp初級者にとって、自作のコマンドが動作することは、とても大きな成功体験であり、自信がつきます。初級者でも簡単に情報源・コマンドを作成でき、自由に変更でき、一度書いたコードはEmacs・anything.elのバージョンが上がってもずっと動き続けるユーザ目線のフレームワークを目指して開発を続けていきます。

複数情報源×複数アクション＝anything.el

原点回帰

anything.elはもともとmacOSのQuicksilverをもとに作られました。初期のanything.elは自分で情報源を設定してM-x anything(現在非推奨)で起動するものでした。

ここでの「情報源」とは、開いているバッファ、最近使ったファイル、ブックマーク、カレントディレクトリのファイルなどanything.elの「部品」となる部分です。

各情報源には、1つ以上のアクションが設定されています。たとえばバッファに対しては、バッファを切り替えたり、バッファを別ウィンドウで表示したり、バッファの内容を挿入したりなどです。M-x anythingでは複数の情報源を設定でき、各情報源が持つアクションもさらに複数ありますので、たった1つのコマンドが情報源の個数×アクションの個数通りの機能を持つことになります。しかも絞り込み検索が強力ですので、あっという間に望む候補を選択できます。快適な環境をたった1つのキーバインドのみで実現させるanything.elは、爆発的なヒットになりました。

時代はhelmへと移り変わり、絞り込み検索ばかりが注目され、anything.elの基本哲学である「複数情報源×複数アクション」が軽視されている傾向にあります。M-x helm-miniやM-x helm-for-filesにはその基本哲学が受け継がれていますので、helmユーザにはそのすごさを感じてもらいたいです。

▼リスト1　M-x anything-miniの設定

```
(defun anything-mini ()
  (interactive)
  (anything-other-buffer
   '(anything-c-source-buffers-list
     anything-c-source-recentf
     anything-c-source-buffer-not-found)
   "*anything mini*"))
```

オリジナルのanythingコマンドを作る

anything.elは、バッファ・ファイルなどの串刺し検索や、状況に応じて候補を表示させる(現在位置の関数名など)こともできます。そのため、クエリを数文字タイプすれば適切な候補がすぐに見つかります。

とはいえ、M-x anythingにむやみやたらと情報源を詰め込むと、動作が重くなってしまううえに候補の表示が後ろにまでスクロールする必要が出てきます。

そこで、anythingコマンドを自分で定義できるようになっています。anythingコマンドを複数個使いこなすことで、最小のキーバインドと快適さを両立できます。

anythingコマンドを自作するには、M-x anything-miniを参考にすると良いです(**リスト1**)。ここから次の部分を書き換えれば、自作anythingコマンドができあがります。

- コマンド名(anything-mini)
- 情報源となる変数(anything-c-source-*)
- バッファ名("*anything mini*")

おもな情報源には、**表1**のものがあります。情報源のリストはM-x anything-call-sourceで見られます。Enterで選択すれば情報源をそのまま試します。C-eで選択すれば情報源の変数名をコピーします。自作anythingコマンドを作る際に役立つはずです。

筆者は次のanythingコマンドを設定しています。

▼表1　おもな情報源

変数	説明
anything-c-source-buffers-list	開いているバッファ
anything-c-source-recentf	最近使ったファイル
anything-c-source-buffer-not-found	新規バッファ作成
anything-c-source-ffap-line	現在位置のファイルと行番号
anything-c-source-ffap-guesser	現在位置のファイル
anything-c-source-bookmarks	ブックマーク
anything-c-source-files-in-current-dir+	カレントディレクトリのファイル
anything-c-source-locate	locateプログラムで得られるファイル名

・C-: にファイル・バッファを開く情報源
・C-; にそのほかの情報源

ツータッチ選択

ツータッチ選択はanything.elが持っていて、helmで削除された重要機能です。普通に使っていると、カーソルを候補に合わせる必要があります。けれども、目に見えた瞬間に候補選択できたほうがカーソルを合わせる手間が省け、すばやくアクションを実行できます。

リスト2の設定を加え、C-x bでM-x anything-miniを開きます。すると、**図1**のような画面になります。左端のA〜Gの文字「ヒント文字」に注目してください。ここで*Shell Command Output*を開きたければ@dとタイプします。バッファのデフォルトのアクションが「バッファを切り替える」ですので、バッファが切り替わります。また、3番めのアクションである「ほかのウィンドウでバッファを開く」を実行したければ@3dとタイプします。使い慣れてきたら重宝する機能です。

◆ ◆ ◆

今回は筆者がずっと暖めてきたanything.elについてお届けしました。helmと袂(たもと)を分かってもanything.elは存続しますので、安心してお使いください。

次回は本連載の最終回となります。最終回にふさわしい内容にするべく、Emacsの有力な情報源についてお届けする予定です。あなたのEmacs力のさらなる発展を願っています。

筆者はいろいろなことに興味を持っているため、サイトはサブドメイン単位で再構築しました。「日刊Emacs」改め「新生日刊Emacs」(http://emacs.rubikitch.com/)は日本語版Emacs辞典を目指し、毎日更新しています。やりたいことやパッケージから探せるようにしましたので、Emacsで何か実現したいことがあればぜひともご覧になってください。手元でgrep検索できるよう全文をGitHubに置いています。

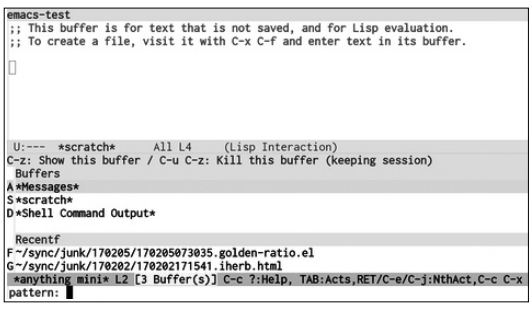
またEmacs病院兼メルマガのサービスを運営しています。Emacsに関すること関しないこと、わかる範囲でなんでもお答えします。「こんなパッケージ知らない？」「挙動がおかしいからなんとかしてよ！」はもちろんのこと、自作elispプログラムや文章の添削もします。集中力を上げるなどのライフハック・マインド系も得意としています。SD

登録はこちら➡http://www.mag2.com/m/0001373131.html

▼リスト2　ツータッチ選択の有効化

```
(require 'anything-config)
(setq anything-enable-shortcuts 'prefix)
  (define-key anything-map (kbd "@")
  'anything-select-with-prefix-shortcut)
(global-set-key (kbd "C-x b") 'anything-mini)
```

▼図1　M-x anything-mini



書いて覚える Swift 入門

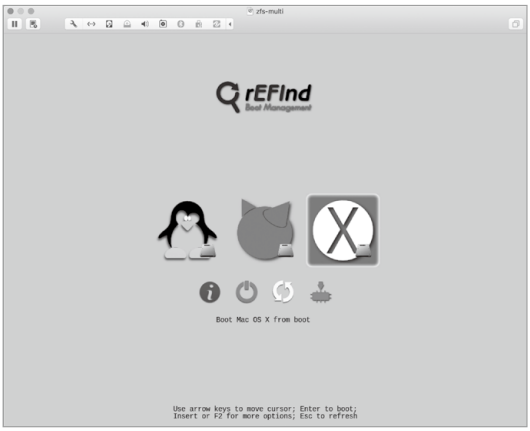
第24回 型の探求

Author 小飼 弾（こがい だん）　twitter @dankogai

From AP to Z

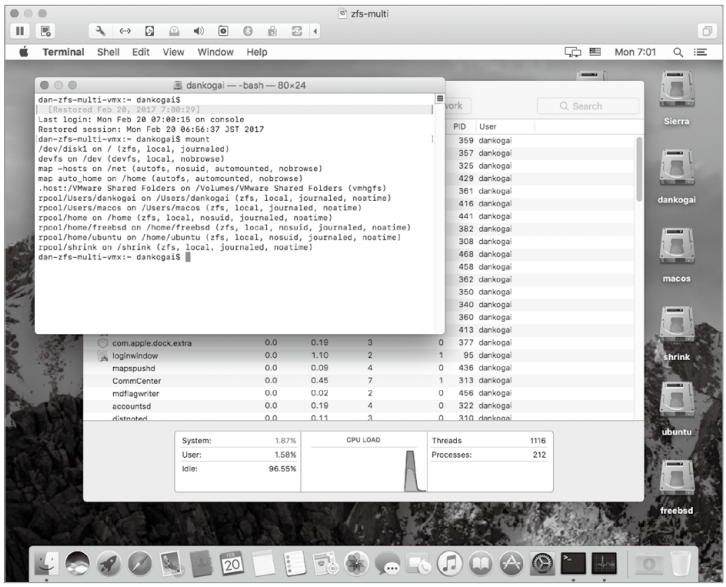
本題に入る前に臨時ニュースを。本連載第18回[注1]でも紹介したAPFSですが、iOS 10.3より正式導入が始まるようです[注2]。本記事執筆現在、iOS 10.3はまだβですが、OSアップデートすると同時にストレージがHFSXからAPFSにアップデートされます。本記事がみなさんのもとに届くころにはiOS 10.3とともにAPFSが我々のiPhoneやiPadに届いているかもしれません。

▼図1　OpenZFS on OS X



それよりずっと地味ながらも見落とせないのは、ZFSのmacOS向けオープンソース実装であるOpenZFS on OS X[注3]のSierra正式対応（**図1**）。FreeBSDやLinuxといったほかのOpenZFS対応OSで作成されたストレージをMacから読み書きできるにとどまらず、なんと同じZFSプールから対応OSをマルチブートできてしまいます（**図2**）。

▼図2　ZFSプールから対応OSをマルチブート



すでにインストーラまでZFSに対応しているFree

注1）URL http://gihyo.jp/dev/serial/01/swift-introduction/0018
注2）URL https://arstechnica.com/apple/2017/01/ios-10-3-will-be-apples-first-update-to-convert-storage-to-apfs/
注3）URL https://openzfsonosx.org

▼図3　Swiftでの配列実験の結果

```
% swift
Welcome to Apple Swift version 3.0.2 (swiftlang-800.0.63 clang-800.0.42.1). Type :help for assistance.
  1> [nil,false,0,"",[],[:]]
error: repl.swift:1:1: error: type of expression is ambiguous without more context
[nil,false,0,"",[],[:]]
```

BSD以外のOSのZFSブート環境構築はまだまだ面倒なのですが、これだけ毛色の異なるOSでファイルシステムがこのレベルで共通して使えることに、オープンソースというものの威力をあらためて実感しています。

オープンソースといえば、Swiftもそうでした。Swiftの父Chris Lattner[注4]のApple退社はSwift界隈で当然話題になりましたが、FreeBSD Projectの父Jordan Hubbard[注5]もAppleに入社しその後退社しています。「父なき」あとも、両プロジェクトとも繁栄していることを鑑みれば、オープンソース化というのはプロジェクトのサバイヴァビリティ向上を大いに改善するのは確かなようです。あのMicrosoftですら、NadellaのCEO就任後はオープンソースへの態度を一変しているのも宜(むべ)なるかな。

型の集まりもまた型

ニュースはこれくらいにして本題に入りましょう。前回に引き続き今回も型の話です。今回は配列(array)や辞書(dictionary)のような、複数の値をまとめて扱うための「まとめ型」について、Arrayを例にとりながら学びます。

Swiftの型は静的。当然配列や辞書も静的な型を持つのですが、動的な型を持つ言語の配列や辞書に慣れていると、面食らうことが1つあります。

まずは、RubyとJavaScriptの配列を見てみましょう。

・Rubyの例

```
% irb
irb(main):001:0> [nil,false,0,"",[],{}].each{|e| p e.class}
NilClass
FalseClass
Fixnum
String
Array
Hash
=> [nil, false, 0, "", [], {}]
```

・JavaScriptの例

```
% node
> [null,false,0,"",[],{}].forEach(function(e){console.log(typeof e)})
object
boolean
number
string
object
object
undefined
>
```

見てのとおり、配列型には異なる型の要素をなんでも入れることができます。なので配列自体の型は1種類で間に合います。

Swiftはどうでしょう？

図3のように文句を言われてしまいました。もう少し見ていきましょう。

```
  1> [false,true]
$R0: [Bool] = 2 values {
  [0] = false
  [1] = true
}
  2> [0,1]
$R1: [Int] = 2 values {
  [0] = 0
  [1] = 1
}
  3> ["","one"]
$R2: [String] = 2 values {
  [0] = ""
  [1] = "one"
}
```

注4）URL https://en.wikipedia.org/wiki/Chris_Lattner
注5）URL https://en.wikipedia.org/wiki/Jordan_Hubbard

そう。Swiftでは配列の要素の型はすべて同じで、そしてElementという要素の型を持つ配列の型は[Element]という型になるのです。

なんて型苦しい？　ところでこのRubyコードを見てくれ。**図4**をどう思う？

すごく……大きなバグを誘発しそうな気がしませんか？

▼図4　Rubyの実行結果

```
% irb
irb(main):001:0> ["answer",42,42.195,[true],false].each{|e| p e + e}
"answeranswer"
84
84.39
[true, true]
NoMethodError: undefined method `+' for false:FalseClass
  from (irb):2:in `block in irb_binding'
  from (irb):2:in `each'
  from (irb):2
  from /usr/bin/irb:12:in `<main>'
```

こういう場合、動的言語ではほとんどの場合、実行時に都度不正な値が入っていないかをチェックする、いわゆるvalidationで解決しています。たとえばこんな感じ。

```
% irb
irb(main):001:0> ["answer",42,42.195,[true],
false].each do |e|
irb(main):002:1*   p e + e if e.respond_to?(:+)
irb(main):003:1> end
"answeranswer"
84
84.39
[true, true]
=> ["answer", 42, 42.195, [true], false]
```

しかしそれはその分コードの量が増えるということでもあり、増えたコードの分だけバグの可能性も増えるということでもあります。配列の要素の型が一定であれば、そうした心配はずっと減ります。

```
  1> [41,42].map{$0+$0}
$R0: [Int] = 2 values {
  [0] = 82
  [1] = 84
}
  2> [41.0,42.195].map{$0+$0}
$R1: [Double] = 2 values {
  [0] = 82
  [1] = 84.39
}
  3> ["answer","universe"].map{$0+$0}
$R2: [String] = 2 values {
  [0] = "answeranswer"
  [1] = "universeuniverse"
}
  4> [false,true].map{$0+$0}
error: repl.swift:4:20: error: binary operator
'+' cannot be applied to two 'Bool' operands
[false,true].map {$0+$0}
```

型がないなら作ればいいのに

しかしそれではJSONのような入れ子があるデータはどうしたらよいのでしょう？　答えは単純。そのような型を作ってしまえばよいのです。たとえば[Int]を入れ子にしたいとしたら、

```
enum NestedArray<Element> {
    case V(Element)
    case A([NestedArray])
}
```

という型を宣言しておいてから、

```
var na = NestedArray.A([
    NestedArray.V(0),
    NestedArray.V(1),
    NestedArray.A([
        NestedArray.V(2),
        NestedArray.V(3)
    ])
])
```

と書けばおk……って何これ長い！

もちろん、もっと短くも書けます。

```
var na:NestedArray = .A([
    .V(0), .V(1), .A([.V(2), .V(3)])
])
```

あらかじめ型を宣言しておけば、その中の要素の種類は省略してこのように書けてしまいます。しかし中の値にどうやってアクセスすれば？

ここから本格的な型作りが始まります。

まずは手始めに中の値にアクセスできるようにしてみましょう。

```
extension NestedArray {
    var value:Element? {
        switch self {
        case let .V(v):
            return v
        default:
            return nil
        }
    }
    var array:[NestedArray]? {
        switch self {
        case let .A(a):
            return a
        default:
            return nil
        }
    }
}
```

これを実行すると次のようになります。

```
na.array?[0].value              // 0
na.array?[1].value              // 1
na.array?[2].value              // nil
na.array?[2].array?[0].value    // 2
```

どうやらうまくいったようです。

でも、本来配列なのに.arrayで中身を取り出さなければならないなんておかしいですよね？

添字で直接中身を読み書きできない？

```
extension NestedArray {
    subscript(i:Int)->NestedArray? {
        get {
        switch self {
        case let .A(a):
            return a[i]
        default:
            return nil
        }
        }
        set {
            switch self {
            case var .A(a):
                a[i] = newValue!
                self = .A(a)
            default:
                return
            }
        }
    }
}
```

これも実行すると、次のような結果が表示されます。

```
na[0]?.value        // 0
na[1]?.value        // 1
na[2]?.value        // nil
na[2]?[0]?.value    // 2
na[0] = .V(42)
na[0]?.value        // 42
na[1]?.value        // 1
```

直接.map()できない？　そのままforループにかけられない？　いっそきちんとしたSequenceになるまで発展させられない？

——あとは読者の宿題ということで。

このような感じで必要な機能をextensionに追加していくのが、Swiftにおける「型作り」になります。

次号予告

ところで今回作ったNestedArrayは、Intだけではなく、DoubleはStringでもいけます。

```
var nad:NestedArray = .A([
    .V(0.0), .V(1.1), .A([.V(2.2), .V(3.3)])
])

var nas:NestedArray = .A([
    .V("zero"), .V("one"), .A([.V("two"), .V("three")])
])
```

とくにそう宣言していないのになぜそれができたのでしょうか？

ところでSwiftにはAnyというなんでも入る型がすでに存在しています。

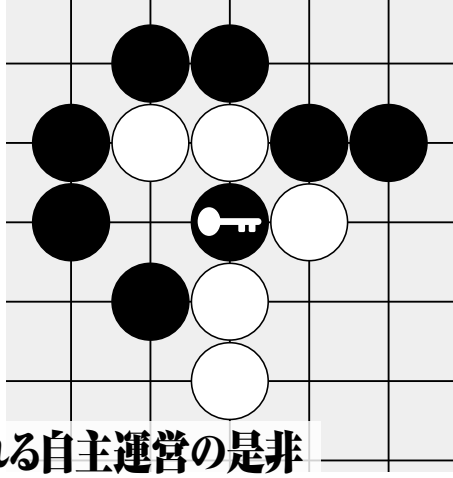
```
var a:Any = [0,1,[2,3]]
```

結論から先に言うと、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。しかし、その理由はいったいなぜなのか？

次回はこれらの疑問に答えつつ、Swiftの型の核心に迫っていきます。SD

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

【第四二回】WordPressサイトの改ざんで問われる自主運営の是非

WordPress 4.7および4.7.1にREST APIに関連する脆弱性が存在していたため、1月27日に修正版4.7.2がリリースされました。2月に入り、この脆弱性対応のアップデートをしていないWordPressサイトをターゲットにした改ざんが多発しました。この事件の背景について解説します。

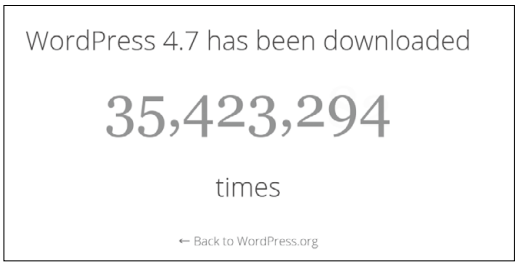
WordPress

WordPressは世界でもっともインストールされているコンテンツ・マネージメント・システムの1つ、簡単に言えば、あちらこちらで使われているブログのシステムです。筆者もまったく独立した4ヵ所のWordPressサイトを管理しています。

また、WordPressをサポートしているレンタルサーバ会社もあり、日本国内でもたいへん多く利用されています。WordPress 4.7がどれくらいダウンロードされているのかをカウントするページでは、2017年2月20日時点で約3,540万回でした（**図1**）。

このWordPressにREST APIに関連する脆弱性があり、2017年2月にその脆弱性をついた攻撃が大量に発生しました。これまでもWordPressをターゲットに、広域に攻撃が発生したことは何度かあります。本体の持つ脆弱性ではなく、プラグインの脆弱性に対する攻撃が主なものでした。WordPressは、管理に注意が必要な部類の1つだと言えます。

◆**図1　WordPress 4.7のダウンロード回数（2017年2月20日時点）**
https://wordpress.org/download/counter/



自動バージョンアップ

2013年10月にリリースされた3.7から、緊急のメンテナンスやセキュリティ・アップデートが必要なときに自動的に適用される機能が入りました。具体的には、マイナーアップデートという形で適用されていきます。筆者のWordPressは自動アップデートの状態にしています。

WordPressのアップデートとバージョン番号の関係ですが、I.J.Kというドット（.）で区切られた3つの整数があると、I.Jがメジャーバージョンを示し、最後のKがマイナーバージョンを示します。ですから、バージョン番号の4.6から4.7へ上がるときがメジャーアップデート、4.7から4.7.1や4.7.2に上がるときがマイナーアップデートとなります。

メジャーバージョンを上げるときは手動で行います。メジャーアップデートは機能のレベルで変更が行われるので、副作用が起きる可能性があるためです。メジャーバージョンが上がると、サイトによっては外観テーマなどに影響が出る場合があります。

マイナーアップデートは、デフォルト設定のままであれば自動的に適用されます。先に示したように、マイナーアップデートという名前ですが、実際

にはメンテナンス／セキュリティ・アップデートです。自動アップデートを停止するなら、マイナーアップデートが発生したときには、直ちに手動で適用する必要があります。

「自動アップデート停止」について書かれた世の中のブログを見ていると、メジャーアップデートとマイナーアップデートの意味が理解されているようには思えず、さらにマイナーアップデートが実質的にセキュリティ・アップデートであると理解されているようにも思えません。自動アップデート停止の副作用として何が起こるのかを考えず、思いつきや漠然とした理由で書かれているものが目につきます。

自動アップデートを停止するときは、マウスでメニューを選択するといったように簡単には行えません。設定をしているPHPプログラムのファイルを手動で編集する必要があります。ですから、「勘違いで」とか「何となくいじっていたら偶然に」とかいう理由で自動アップデートができなくなるということはありません。アップデートしていないサイトは、手間をかけて意図的にそうしているわけです。もちろん自動アップデートが効いていないのはリスクが高いわけですから、常時メンテナンスに目を光らせる必要があります。ですが、その労力を理解したうえで停止しているのかは、たいへん疑問です。

REST APIとは何か

今回の脆弱性の原因となったREST APIとは何なのでしょうか。WordPressは基本的にはWebブラウザを使って読み込み／書き込みをしますが、REST APIを使えば外部からWordPressへの読み書きなどが行えます。具体的には、REST APIを使うことで、スマートフォンのサードパーティ・アプリケーションと連動できます。あるいは、IoT機材から自動的に書き込んだりすることもできます。

REST APIの機能が入ったことにより、REST APIを経由して独立したアプリケーションと情報を簡便にやりとりできるようになりました。REST APIは、これまでの単純なブログシステムのフレームワークから大きく変化するために入った機能だと言えます。最初のREST APIは4.4で入りましたが、本格的な機能は4.7から入っています。REST APIは4.7系からが本番と理解してかまわないと思います。

何が悪かったのか

REST API機能の何が問題だったのか、その詳細な説明は2017年2月1日に公開されたセキュリティベンダのSucuriのブログ[注1]に書かれています。詳細はこのブログを読んでもらうとして、本稿では、実際に筆者のサイトに行われていた攻撃で説明します。以下はその攻撃のリクエストライン（HTTPリクエストの1行目）の内容です。

実際のログに残っていた攻撃のあと

```
POST /wp-json/wp/v2/posts/1213/?id=1213abc HTTP/1.1
```

この攻撃コードでは、ターゲットとなっているWordPressサイト上の記事番号が1213で、そのidが1213abcとなっています。idが正しければ、POSTされる内容（上記のリクエストラインのあとに入っているデータ）で記事番号1213が上書きされます。

本来のidは、1213XXX（XXXは任意のアルファベット）となっていたとします。文字列の比較として1213abcと1213XXXは同じではありませんから、認証は却下されます。

ところが、この値を整数型として処理してしまうバグがあると、1213abcも1213XXXも整数値1213として同等に扱われます。よって、この認証も成立してしまいます。これにより、サイトの内容を書き換えることができます。

これだけを見ると、わりとよく見かける文字列から整数に変換するときのバグです。テストケース漏れと言えるのでしょうが、あっても不思議ではないバグです。ただ、そのありがちな問題が認証回避として現れて、結果的に大きな問題になってしまっています。bashの脆弱性「CVE-2014-6271 "Shellshock"」

注1）Sucuriブログ　https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html

もそうですが、バグの出現頻度とバグが引き起こす結果には関連性がないことをあらためて感じます。

改ざん騒動までの経過

まず、2017年1月26日にWordPress 4.7.2のアップデート版がリリースされました[注2]。この時点では、REST APIの脆弱性に関しては明示されておらず、それ以外の次の3つの問題に対応したのでセキュリティ・アップデートをリリースしたと説明されていました。

①簡易投稿機能「Press This」のタクソノミー（分類）語句を割り当てるユーザインターフェースが、使用権限のないユーザにまで表示される問題
②安全でないデータが渡されたときに発生するWP_QueryのSQLインジェクション（SQLi）の潜在的危険性
③投稿リストテーブルのクロスサイトスクリプティング

筆者のWordPressは自動アップデートされます。アップデートが適用されると「自動的にアップデートした」というメールが届きます。その記録を確認すると、4.7.2へのアップデートは日本時間2017年1月27日午前5時30分に終了しています。

2017年2月1日の情報解禁日

REST APIの脆弱性の情報解禁日は、2017年2月1日でした。先に紹介したSucuriブログと、WordPressのセキュリティ告知[注3]、そしてWordPress 4.7.2アップデートの説明のページの改定が連動して公開されます。

セキュリティの告知のページには、「このREST API脆弱性は、2017年1月20日にSucuriのセキュリティ研究者Marc-Alexandre MontpasによってWordPressのチームに報告されていた」と書かれています。筆者のサイトのアップデート記録と合わせると次のようになります。

- **2017年1月12日　4.7.1アップデート**
- **2017年1月20日　REST API脆弱性発見**
- **2017年1月27日　4.7.2アップデート**
- **2017年2月1日　REST API脆弱性の詳細公開**
- **2017年2月4日　REST API脆弱性を使ったWordPressサイト改ざん相次ぐ**
- **2017年2月6日　テレビや新聞で報道される**
- **2017年2月7日　筆者サイトで初めて攻撃のログを記録**

どこから筆者のサイトに攻撃してきたかを見てみると、フランス、トルコ、ロシア、アメリカ、グルジア、チェコからで、いくつかはVPS（Virtual Private Server）サービスを提供している会社のIPアドレスから来ていました。

被害

いくつかのグループが競い合うようにしてWordPressのサイトを改ざんしています（**図2**）。たとえば、次のような有名なチームのキーワードでGoogle検索すると、改ざんされたと思われるサイトが大量にヒットします（2017年2月20日現在）。

◆図2　改ざんされたWordPressサイトの一例

注2）以下のサイトを見ると、2017年1月26日の内容に、2月1日のREST APIの情報が付け加えられている。
https://wordpress.org/news/2017/01/wordpress-4-7-2-security-release/
注3）WordPressのセキュリティ告知
https://make.wordpress.org/core/2017/02/01/disclosure-of-additional-security-fix-in-wordpress-4-7-2/

Googleで検索したヒット数

- “Hacked By MuhmadEmad” 約935,000件
- “Hacked By SA3D HaCk3D” 約629,000件
- “by w4l3XzY3” 約366,000件

改ざんされたページの正確な数字はわかりませんが、1,000ページとか2,000ページのオーダーではなく、少なくともトータルで100万ページは軽く越えていると見ていいでしょう。

日本国内でも、丸川珠代・五輪担当相のページも被害にあっている関係もあってか、2017年2月7日に新聞で報道されています[注4]。

攻撃はページ単位で起こるので、同じWordPressサイトで“MuhmadEmad”と“w4l3XzY3”の改ざんページが同居していることもしばしばです。そう考えると、一度書き換えられたページがさらに書き換えられているケースも多くあると思います。

今回の攻撃は、たまたま書き換え合戦のようなものでした。執筆の時点では水飲み場攻撃のように巧妙にマルウェアを感染させるリンクなどを置くという被害報告は聞かれません。攻撃手法が簡便だった(ため、いきなりたくさん参入してきて書き換え合戦のようになった)副作用なのかもしれません。

事前に防げたのか

実は4.7.2にアップデートするタイミングで、US CERTのサイトのカレントアクティビティに“WordPress Releases Security Update”ということで掲載されていました[注5]。このカレントアクティビティに掲載される脆弱性は、多々ある脆弱性の中でもセキュリティ・アップデートを“MUST”で行う必要があるものだと考えてください。普段のWordPressのマイナーアップデートはこんなところには出ません。この今回の攻撃が始まる前に、珍しいところにリストアップされているな、と思った記憶があるのですが、自分のサイトはすでにアップデートされていたので内容は気にかけていませんでした。

WordPressのセキュリティプラグイン

筆者はWordPressに、Wordfence社[注6]のセキュリティプラグインを導入しています。まだ有償版にはアップグレードしていませんが、それでも定期的なセキュリティに関するメールマガジンと、プラグインが古くなっていることを自動的にメールしてくれる機能は十分に役に立っています。

今回も、日本時間2017年2月2日23時(米国時間では2月1日)に、Wordfence.comから「至急4.7.2へアップグレードするように」というメールが届いていました。自動アップデートがなかったとしても、このメールを読んで手が空きしだい、すぐに適用していたでしょう。というより、プラグインのアップデートのとき、筆者はいつもそうしています。ギリギリではありますが、これなら大規模な攻撃が起こる前に対応できた可能性は十分にあったでしょう。

4.7に上げない作戦は役にたつのか？

マイナーアップデートを停止させる手順を書いていたサイトでは、マイナーアップデートしない理由として、「アップデートすると、これまでのプラグインなどが動かなくなる」と記載していました。それを言うなら、公開されてすぐのメジャーアップデートを避けるべきではないかと思います。

繰り返しになりますが、メジャーアップデートは新しい枠組みや機能を取り入れるからメジャーアップデートとなるわけで、だからこそ、適用するかどうかの判断は利用者側に任せられているわけです。新しいバージョンが安定したのを見計らってアップデートするという手順でも良いかと思います。

ただし、安定したとはいえ、いつ重大な脆弱性が現れるかは「神のみぞ知る」です。だからこそ、自動でマイナーアップデートをする機能をオフにするの

注4)「丸川大臣HP、改ざんされる　同様の被害450以上」 朝日新聞、2017年2月7日、朝刊、26面
注5) US CERT “WordPress Releases Security Update” Original release date: January 26, 2017 | Last revised: February 01, 2017 **https://www.us-cert.gov/ncas/current-activity/2017/01/26/WordPress-Releases-Security-Update**
注6) WordPressのセキュリティプラグインを提供しているWordfence社 **https://www.wordfence.com/blog/**

は、リスクがあるのです。手動にしていても、脆弱性が発見されればアップデートせざるをえないのですから、脆弱性対応という観点からは、あまり違いがないでしょう。

WordPressを自分で管理しない

筆者は、自分で管理ができないのに無理してWordPressを管理する必要はないと思います。たとえば、WordPress.comでは、WordPressブログをレンタルしてくれますし、また、自分のドメイン名で運用することもできます。そのようなサービスを利用すれば、WordPress本体のメンテナンスからは解放されます。今回のような問題に対しては、ずいぶん改善されると思います。

まとめ

今回のWordPress 4.7のREST API脆弱性を狙った大量ページ書き換えの問題の本質は、きちんと脆弱性対応ができる体制を持っているかどうかに尽きると思います。筆者は職業的にWordPress管理をしているわけではないので即応体制はとれず、そこは自動的に行われるセキュリティ・アップデートに任せています。

セキュリティ・アップデートに即応できるのなら、自動アップデートを停止するのもかまわないと思いますが、対応できないのに自動アップデートを停止するべきではないと筆者は考えます。

WordPressはプラグインまで含めると、独自に運用するのはリスクの高いシステムです。WordPressの使い方は簡単でも、運用はそのほかのWebサービスを管理するのと何一つ変わらないのです。

また、プラグインも含めたWordPressの全体のセキュリティを考えるのなら、「気をつける」というだけでは間に合いません。「セキュリティのプラグインを入れて安全性を高める」、あるいは「情報を収集するためにメールマガジンなどに登録する」など、それなりの手間をかける必要があります。SD

技術評論社

五木田和也 著
A5判／176ページ
定価(本体1,680円+税)
ISBN 978-4-7741-8410-4

大好評発売中!

コンピューターで「脳」がつくれるか

AIが恋に落ちる日

囲碁で人間を打ち負かす人工知能ロボットや自動運転車が進歩を続けても「人間のようなロボット」にはなりません。世間は人工知能ブームに沸いていますが、人間のように自ら学習して成長していく「汎用人工知能」を実現するにはまだ道なかばです。そもそも汎用人工知能とディープラーニングやIBMのWatsonに代表されるようなふつうの人工知能は何が違うのでしょうか。
本書では、脳の各部位がどんな役割を持ち、どうやって情報を処理しているのかを解説し、人間らしい知能を実現するにはどのようなしくみが必要かふれていきます。続いて、身近にある人工知能のなかで動いている機械学習やディープラーニングなどのコンピューター技術のしくみを説明します。人間の脳のしくみと最新の機械学習のアルゴリズムを照らし合わせてみると、なんだか人間のような知能が再現できそうな感じがしてきます!
近い将来、心を持つ人工知能を作ることは現実になるのでしょうか。本書で人類の技術がどこまで人間の「脳」に近づいているのか確かめてください。

こんな方におすすめ
・人工知能について知りたい方
・汎用人工知能に興味のある方

Linuxカーネル「ソースコード」を読み解く

平田 豊 著
B5判／192ページ
2,300円＋税
工学社
ISBN＝978-4-7775-1986-6

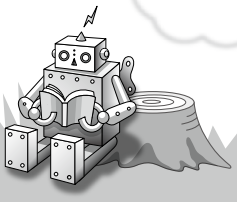
アプリケーションやライブラリがどのようにLinuxカーネルの機能を利用しているかに着目して、Linuxのソースコードを読んでみるという主旨の本。対象OSはFedora 24（Linuxカーネル5.4）。OSのコードリーディングということで身構えてしまうかもしれないが、必要とされるスキルとしては「実装言語であるC言語を読めること」「ソースコードを検索・整形するためのLinuxのコマンドが使えること」の2点だけ。コード読解の“コツ”についても解説しているので初心者に優しい。肝心の本書で読むソースコードは「ユーティリティツール『BusyBox』」「ファイルI/Oの関数」「FIFO（名前付きパイプ）の実装」であり、システムコールやユーザ空間・カーネル空間について学ぶことができる。

プログラミング作法

Brian W. Kernighan、Rob Pike 著／福崎 俊博 訳
A5版／360ページ
2,800円＋税
アスキードワンゴ
ISBN＝978-4-04-893052-9

「スタイル」「アルゴリズムとデータ構造」「設計と実装」「インターフェイス」「デバッグ」「テスト」「性能」「移植性」「記法」という開発における7つのテーマの中で、守るべき作法をそれぞれ解説している。著者は数多のソフトウェアの開発・メンテナンスに関わったベテランのエンジニアで、経験から学んだ実践的なノウハウを教えてくれる。「悪いコードにコメントを付けるな、書き直せ」「バグを再現できるようにしよう」「（文字コードについて）ASCIIを前提にするな」などなど、さまざまな粒度のアドバイスを受け取ることができ、実務に活かせるだろう。ところどころでC/C++/Javaの実例コードが示されるが、本書自体は特定の言語に依存する内容ではなく、プログラミングにかかわる人なら誰にとっても有用な1冊だ。

SD BOOK REVIEW



進化する銀行システム

星野 武史 著／花井 志生 監修
A5判／256ページ
2,580円＋税
技術評論社
ISBN＝978-4-7741-8729-7

本書は銀行システム、とくにメインフレーム（以下、MF）の技術について解説する。MFとIAサーバはまったく別ものに思えるが、本書を読むと、両者はアプローチが違うだけで、同じことを実現しているケースが多いとわかる。「MFは部品やデータの冗長化で、IAサーバはサーバそのものの冗長化で、信頼性を上げる」「MFはスケールアップで、IAサーバはスケールアウトで、拡張性を確保する」など。アプローチが違うのはそれぞれのシステムに求められる要件が違うからだろう。銀行は技術要員も少なく、外部にデータを出すのも難しい。だから「ずっとベンダが面倒を見てくれるサーバに」「サーバは社内に収まる台数に」ということでMFになるのかもしれない。技術を比較しながら読むと、そんなふうに新たな視点で背景が見えてきて興味深い。

プロが教える情報セキュリティの鉄則

香山 哲司、小野寺 匠 著
A5判／176ページ
2,480円＋税
技術評論社
ISBN＝978-4-7741-8815-7

人間はミスをする生き物。これは情報セキュリティを語るときに第一に考えねばならないことだ。世の中には宴会のときに酔っ払ってPCを紛失する可能性があるから別途バイク便で自宅に送るという、笑うに笑えない対策をする会社まであるそうだ。そのような現状のなか、情報セキュリティ対策として何をすべきか。本書では情報流出がなぜ起こるのか、さまざまな事例を起点に誰もが安心して使うことができるシステムのあり方を解説する。各種Webサービスにおける二要素認証の実施など、世の中の流れも事故が発生しにくいように変化してきているが、企業のシステムの場合は基本的にフェールセーフを基準に考えていくべきで、その解として本書ではセキュリティを科学としてとらえなおすこと、そして認証基盤の構築を第一としている。

SOURCES

レッドハット系ソフトウェア最新解説

第8回

OSスキルアップのススメ

コンテナ技術やクラウド技術などを活用したDevOpsが注目されている現在において、OSのスキルが必要な理由や学習方法などを紹介します。

Author 小島 啓史(こじまひろふみ)
mail：hkojima@redhat.com
レッドハット㈱ テクニカルセールス本部 ソリューションアーキテクト

なぜOSのスキルを習得すべきか

日本国内では2013年頃から、DevOpsという言葉が注目を浴びてきています。変化の速いビジネスに合わせたシステムの開発、およびリリースの高速化を実現するための考え方や文化の1つと言えます。AWS、Azure、OpenStackなどのクラウド技術活用によってインフラ導入のハードルが下がったことや、Dockerなどのコンテナ技術活用によるアプリケーション実行環境の統一が容易になったことが追い風となっているように見受けられます。そうした背景により、システムの安定運用を目指す運用側だけでなく、プログラマなどのアプリケーションを開発する側も、簡単な状態確認や障害対応などのために、アプリケーション実行の土台となるOSの知識が求められるようになってきました。多くの場合において、まずOSが出力するリソース利用状況の情報やエラーメッセージなどのログの確認が、状態確認や障害対応の第一歩となることが多いからです。

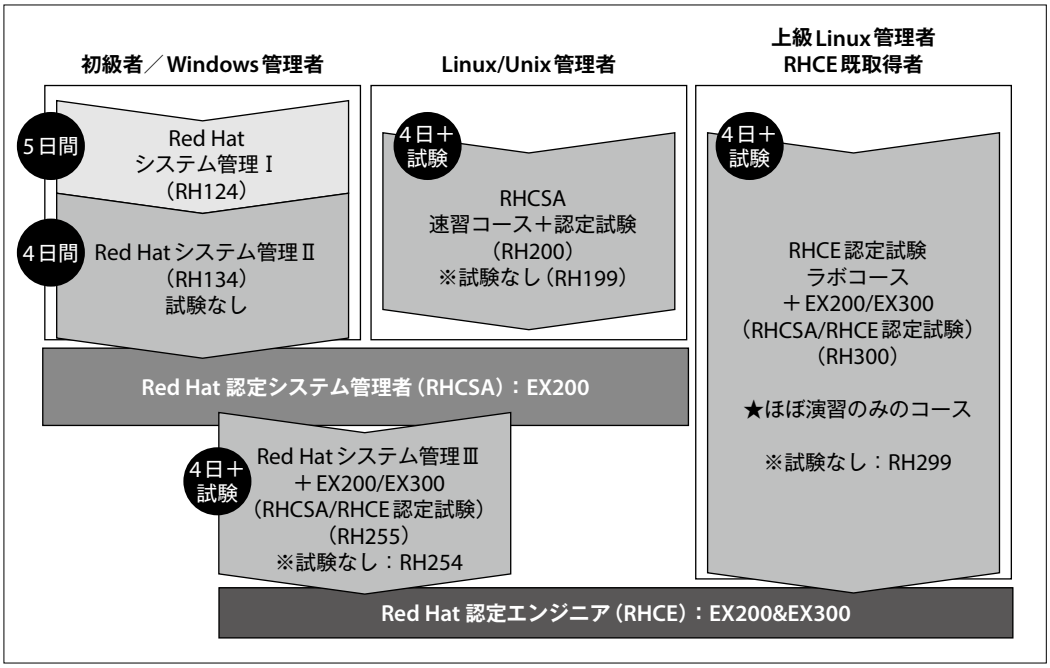
では、どのOSを選択すべきでしょうか。筆者は、個人的にはLinuxを推奨します。上記のOpenStackやDockerはオープンソースであるため、同じオープンソースであるLinuxと相性が良いことや、動作原理や開発過程がWebサイトで公開されているので、開発者の設計思想や想定用途を汲み取ったうえで、利用者にとって最適な構成や設定を調査しながら導入できるといった理由があるからです。

Linuxの学習方法

Linuxにもいろいろ種類はありますが、Red Hat系のLinuxを選択する場合は、Fedora、CentOS、RHEL(Red Hat Enterprise Linux)のいずれかから選択することを推奨します。FedoraはRed Hatがサポートする非営利のコミュニティが開発および提供しているLinuxであり、無償でWebサイトからダウンロードして利用できます。また、RHELはRed Hat社が有償で提供しているLinuxであり、Fedoraで得られたフィードバックをもとに開発されています。CentOSはRHELのソースコードをベースとしたLinuxであり、以前はRed Hatとは独立したコミュニティにより運営されていましたが、2014年1月にRed HatがCentOSプロジェクトを正式に後援することを発表[注1]しました。こうしたFedora/CentOS/RHELについての解説記事が

注1) URL https://www.redhat.com/en/about/press-releases/red-hat-and-centos-join-forces

▼図1　RHEL研修コースの受講パス



Webサイトに掲載[注2]されていますので、興味があれば参照ください。

OSのインストールの実際と参考図書

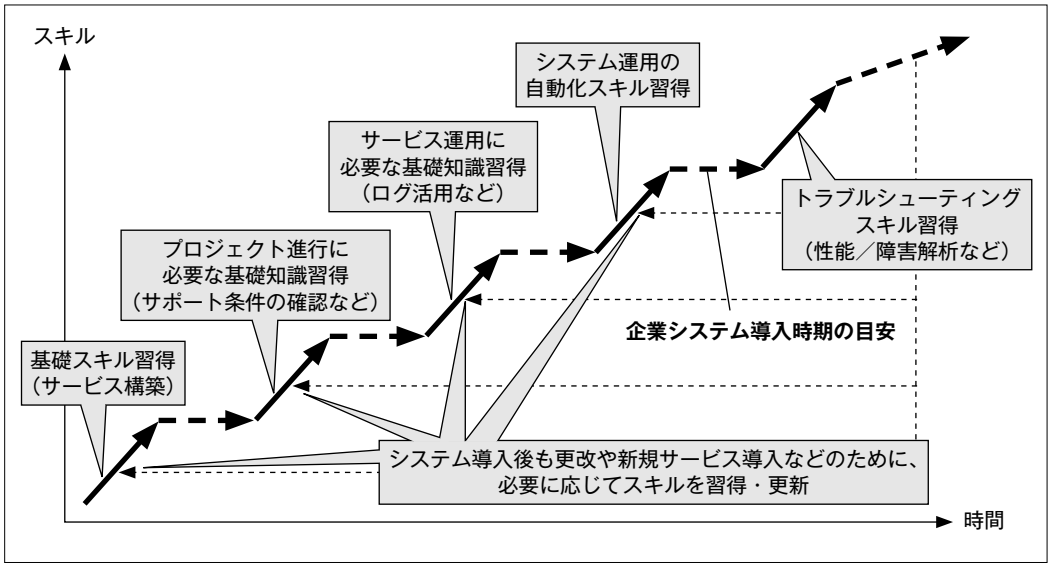
無償でいろいろ学習したい場合は、FedoraかCentOSをWebサイトからダウンロードして利用していくことになります。Linuxのインストール先として物理マシンを用意できる場合は、一般的な家電量販店で販売されているノートPCを用意します。ノートPCにデフォルトで入っていることが多いWindows上の仮想マシンとしてLinuxを入れる場合は、VMware PlayerやVirtualBoxといった無償の仮想マシン構築ソフトウェアを利用します。物理マシンを用意できない場合は、AWSのようなクラウドサービスで用意されている無償の利用枠[注3]を利用します。学習する際の代表的な参考図書には、『独習Linux専科[注4]』、『プロのためのLinuxシステム・10年効く技術[注5]』、『[改訂新版]プロのためのLinuxシステム構築・運用技術[注6]』があります。こうした本やRed Hat社がWebサイトで公開している製品ドキュメント[注7]をもとに設定および構築手順を試していくことが、Linuxスキルの習熟度を上げるための近道かと思います。

Red Hatの研修サービス

ただし、Linuxを独学していくには時間と根気が必要です。根気はともかくとして、時間を短縮したい場合は、Red Hatの研修サービスの受講を推奨します。RHELを熟知した当社インストラクターがサーバ構築や運用に必要不可欠な基礎スキルを、実機による演習を含めながら効率的に紹介していくサービスとなります。RHELに関しては、受講対象者や前提条件を基

注2）URL http://jp-redhat.com/centos_faq/
注3）URL https://aws.amazon.com/jp/partners/redhat/
注4）URL http://gihyo.jp/book/2013/978-4-7741-5937-9
注5）URL http://gihyo.jp/book/2012/978-4-7741-5143-4
注6）URL http://gihyo.jp/book/2016/978-4-7741-8426-5
注7）URL https://access.redhat.com/documentation/ja/red-hat-enterprise-linux/

▼図2　スキル習得および更新イメージ



に複数のコースを提供しています。これらのコースは体系的に設計されており、受講パスは**図1**のようになります。

大まかに解説すると、認定システム管理者系（RHCSA系）のコースでローカルでの設定手順（インストール、簡単なユーザ認証、ファイルシステムの利用方法など）を扱い、認定エンジニア系（RHCE系）のコースでは代表的なサービスの構築手順（ファイル／メール／Webサーバ、DNSのキャッシュ、ファイアウォールなど）を扱います。また、無償のオンラインスキルチェックテスト[注8]がありますので、コース選定の参考にできます。

こうした体系的なコースは高い評価を得て[注9]いますが、システムの運用管理者が企業で使うシステムの安定運用の実現を目指すためには、これらのコースで習得できるスキルに加えて、追加で習得しなくてはいけないスキルがあります。

・サポート条件や推奨構成などに関する知識
・システム監査などに利用するログ活用スキル
・シェルスクリプトなどを用いたシステムの運用自動化スキル
・性能や障害解析に必要な、Linuxカーネル／ネットワークの動作モデルに関する知識

これらのスキルを習得した運用担当者がシステム構成に対する深い理解を持って、アプリケーション実行基盤の最適化に貢献していくことが、DevOps実現につながっていきます。なお、こうしたスキルはシステムの更改や新規サービスの導入に応じて、その都度更新していく必要があります。スキル習得および更新イメージは**図2**のようになります。

こうしたスキルの習得に必要な情報を、当社では製品ドキュメントやナレッジベース[注10]として公開しています。ただし、これらの情報をすべて独学するには難しいと思う場合は、当社コンサルタントにより、ユーザ専用の研修サービスとして提供できます。興味がありましたら、sales-jp@redhat.comまでお問い合わせください。SD

注8）URL https://www.redhat.com/rhtapps/assessment/session/
注9）URL http://jp-redhat.com/training/nri/index.html
注10）URL https://access.redhat.com/search/#/knowledgebase　※閲覧には当社サポート契約が必要な場合があります。

45

Debian Developer　やまねひでき　henrich@debian.org

地道に進むDebian 9に向けたパッケージ更新

Debian Hot Topics

Debian 9リリース作業の進捗は?

Debian 9“Stretch”リリースに向けての進捗は、すでにパッケージの更新についてはレビューを受けないといけない「Full Freeze」期間に入っています。release.debian.org擬似パッケージのページ[注1]には、200を越える大量のunblockリクエストとその処理の跡が見て取れます[注2]。

リリースを阻んでいるリリースクリティカルバグの数は、ジリジリと減って2月中旬時点で約240個程度[注3]です。その詳細を見ると、解決の難しそうな問題ばかりが残っていて、開発はまだ長引きそうです[注4]。全体に影響するような大きな問題はありませんが、個別にひとつひとつを地道に修正する作業が続きます。

Icedoveから Thunderbirdへ

先だって本連載第35回(本誌2016年5月号)でもお伝えしたとおり、IceweaselはFirefoxに名称を戻しました。それに続き、IcedoveもThunderbirdへ改名となりました(**図1**)。

「De-Branding of Icedove, reintroducing Thunderbird packages into Debian」というメール[注5]で案内が出され、この移行でどんな問題が出るか洗い出そうとしているようです。Debian 9“Stretch”リリースの際には、スムーズにDebian 8“Jessie”から変更ができることを期待しましょう。

OpenStack関連パッケージの雲行きは怪しく

OpenStack関連パッケージのメンテナをしているThomas Goirandさんから「The end of OpenStack packages in Debian ?」というメー

注1) URL http://bugs.debian.org/release.debian.org

注2) Full Freezeに入ると、パッケージの更新を行う場合は、Debian BTSに擬似パッケージとして登録を行い、リリースチームにunblockしてもらう(ブロックを解いてもらう)必要があります。

注3) URL https://bugs.debian.org/release-critical/

注4) Bugs search(URL https://udd.debian.org/bugs/)で詳細な分析ができます。

注5) URL http://deb.li/QLUl

▼図1　IcedoveとThunderbird

ル[注6]が送付されました。Thomasさんはこれまで、大手OpenStackベンダのMirantis社と契約してフルタイムでDebianパッケージの作成をしていました。しかし、この契約がなくなってしまい、作業の継続が難しくなったようです。

Mirantisでは、2016年10月に大きなエンジニアのレイオフ(一時解雇)が実施されており[注7]、Thomasさんの契約が打ち切られたのも、その一環でしょう。この動きは、MirantisのOpenStackビジネスが開発・構築主体ではなく、「マネージドプライベートクラウド」と呼ばれているOpenStackの運用へシフトしていることが、大きく影響していると見られます。

AWSをはじめとするパブリッククラウドは業績好調のものばかりですが、プライベートクラウド基盤として使われているOpenStackについては、名前をよく聞くわりにはその周辺ビジネスはあまり順調とは言えない話ばかりですね[注8]。

ほかのOpenStackパッケージチームのメンバーからは「チームでやっているのだから、まるで終わったような表現をしないでよ」とのコメントもありました。ですが、Thomasさんがほとんどの作業をやっていたのは間違いないので、人手が新たに加わらない限り、DebianでのOpenStack関連パッケージの状況は残念ながら悲観的にならざるを得えないでしょう。

GNU Globalパッケージが7年ぶりに更新

少し前の話題になりますが、ソースコードに索引付けをして高速・快適にソースを閲覧できるようにするソースコードタグシステム「GNU Global」[注9]のパッケージが、Debianで7年ぶり(!)に更新されました。あまりにパッケージの更新がされていなかったため、開発元のダウンロードのページでは「It's too old. Please don't use Debian's package.」と赤い太字で書かれるほどでした。いったい何があって停滞し、どのようにして更新が再開されたのでしょう?

BTSを参照すると[注10]、2010年にパッケージメンテナのRon Leeさんがバージョン5.7.1をアップロードして以降、何人もが更新の依頼を投稿していますが、Ronさんは返答をしていません。そんな状況が2013年まで続いていました。すると痺れを切らしたのか、upstreamの山口志義夫さんが登場し「もっと若い人にメンテナを任せたら? パッケージをOrphan(みなしご化=手放してメンテナ不在状態)にしよう」とコメントしました。すると、ようやくRonさんはコメントしたものの「今はDebianはフリーズ中なので変更はしたくないし、新しく導入されたスクリプトがセキュアではないので懸念している」とこれをはねつけます。続けての議論では、

Ron「私は懸念点をすでに表明したし、私やほかの人が以前にした提案はすべて拒否されたよね」
山口「提案を拒否するのも私の仕事だから。GLOBAL Bugメーリングリストで議論しないか?」
Ron「もう議論して合意したとおりだよ」
山口「合意に至った記憶はないよ? メーリングリストでの私のRFC(意見募集)メールに、君はコメントもしていないじゃないか」

というように、「何が問題でどう解決すべきなのか」という技術的な論点が交わされません。経緯がよく見えない周りの人からは「公開の場で議論されていないのでわからないのだけど、何が問題なの? CGIスクリプトだけ?」との問いかけがされるも、スルーされます。

そして、Punit Agrawalさんがパッケージのアップデートをしようと山口さんとやりとりを

注6) URL http://deb.li/i3oSG
注7) URL http://deb.li/TRuC
注8) これについては「OpenStack自体の利用は広まっているが、ユーザ企業が導入から運用まで実施しているため、何かを提案したり導入のサポートを行ったりという周辺ビジネスとしては入り込みにくい」という指摘があります。
注9) URL https://www.tamacom.com/global-j.html
注10) URL https://bugs.debian.org/574947

続けていたところ、ようやくRonさんがコメントをします。「Globalの新しいバージョンでは、静的なスクリプトではなく動的に生成されるCGIスクリプトをrootで動かしており、危険だ」というのが彼の意見で、確かにrootで内容もよくわからないCGIを動かされるのは怖い話ですし、rootで動かす代わりにchmod 777として解決するというのも乱暴過ぎるやり方です。

これに対して山口さんは「(この動作をする)--system-cgiオプションはあまり使われていないし、削除しても問題ないよ」とコメント。Punitさんも「1年ほど使っているけどrootで何か動かすことはなかったけど……？」と疑問を呈しますが、この点についてはRonさんはとくにコメントせず、別の話題が続きます。2015年になって「Global 6.4では--system-cgiオプションはなくなったよ？」という別の人からの指摘コメントにもとくに反応がありません。

時は流れて2016年11月、ついにWookeyさんがNon-maintainer Uploadで6.5.5-0.1をアップロードし[注11]、その数日後にRonさんがパッケージを手放しました[注12]。その後は、Wookeyさんがメンテナを引き継いで、Unstableに更新されたパッケージがアップロードされました。これで、Debian 9"Stretch"では更新されたパッケージが使えるようになりそうです。

今回の件に関して筆者の視点からは、メンテナとして懸念点について説明と調査・対応をしていないRonさんに大いに問題がある、と見えます。もし、私がメンテナだったら次のような対応をするでしょう。

- upstreamで導入されたCGIスクリプトがセキュアではないと思うのであれば、問題点を明示的に列挙してBTSに登録するなどしてほかの人が参照できるようにする
- 問題のありそうな機能をdisableにしたパッケージを更新し、README.Debianファイルに上記の理由と作業内容を追加。更新したパッケージをまずはexperimental(試験版リポジトリ)に入れて意見を募る
- 問題が起こらなければunstableに投入する。変更について、stableでのリリースが適当でないのであれば、testingから削除されるように懸念点をRCバグとして登録しておく
- upstreamとは議論を続けて「よりセキュアにその機能を実現する」方法を模索する

問題のありそうな機能をdisableにしてパッケージを更新することについて、Ronさんは「拒否する。『私はこの機能は欲しくないし、私が望む些細なことを急ぐためにほかの人向けにも削除してしまおう』というのは受け入れがたい解決策だ。もしそれがほしいものならば、自分専用のローカルパッケージならとても簡単に作れるよ」と否定しています。ある意味もっともな話ではあるものの、この判断によりUpstreamが新しいバージョンのリリースで付加した価値[注13]を7年もの間ユーザにまったく届けられていないのですが、そのほうが大きな問題だとはとらえることができていないようです。今回の件は、反面教師として記憶に留めておく価値はあります。

DebConf18は台湾に決定

2018年夏のDebianカンファレンスは投票の結果、台湾(新竹市)に決定しました[注14]。台湾は日本からは安価(1〜2万円程度)かつ短時間で訪れることができるので、今までのカンファレンスと違ってグッと参加しやすいですね。SD

注11) URL https://tracker.debian.org/news/820174

注12) 「このパッケージをOrphan(みなしご化)する。より良くモダンなほかの選択肢があるニッチなパッケージのために、非協力的なupstreamと恩知らずでロ汚いユーザの間で挟まれ続けるのは、楽しくあるべき私の流儀じゃない」と恨み言がchangelogに記載されています。

注13) 機能追加のみならず、バグの修正・改善も「価値」です。BTSでは「(古い)DebianパッケージのGNU Globalでは動かないが、upstreamのは問題なく動く」という話がいくつも報告されていました。このような大きな問題を何年間も放置しているのは、判断ミスとしか言えないでしょう。

注14) URL http://deb.li/3Ho1I

Let's noteにUbuntu GNOME 16.04 LTSをインストールする

Ubuntu Japanese Team　あわしろいくや

はじめに

筆者はこれまでPanasonic Let's note CF-S10を使用していましたが、使用開始から5年を超えてさすがにガタがきはじめたので、別のPCを購入する必要に迫られました。今となってはLet's noteにこだわらなくても軽くてバッテリの持ちのいいうえに安いPCはありますが、今回もLet's note（CF-SZ6）にした理由は、5年間何の故障もなく使えたことと、バッテリが分離できることが大きかったです。Let's noteは大規模なバッテリのリコールがあり、バッテリが内蔵されているといざというときにはPCまるごと送り返さなければいけなくなるので、分離できるのはメリットだと思ったのです。

CF-S10との大きな違いは、メモリの増設もストレージの交換もできないことです。CF-S10ではメモリを4GB増設し、HDDもSSDに交換しました。しかし時代の流れでそれができなくなったのは残念ですが、薄くなったというメリットも一応はあります。となるとメモリは8GB、SSDも256GBモデルにする必要があります。それ以上はBTOモデルとなり、予算を大幅に超過します。そのような理由から、店頭販売モデルの上位2番めのモデルであるCF-SZ6FD3QRになりました。

購入時期は2016年末です。なんとか買えそうな価格だったのと、年末年始の休みのうちに移行作業をやりたかったのが主な理由です。

パーティションの縮小

購入してすぐにするのはリカバリーメディアの作成であることは言うまでもないので解説は省略しますが、16GBのUSBメモリを用意したにもかかわらずリカバリーメディアの作成には使えず、ほかにストックしてあった32GBのUSBメモリを使用したのは誤算でした。16GBのUSBメモリはフォーマットすると14.4GBしか使用できず、必要な領域である15GBを下回ってしまうのです。たしかに筆者が迂闊（うかつ）だったのですが、もう少しわかりやすくてもいいのではないかと思ったものです。実際に作成したリカバリーメディアは8GBちょっとしか使用していないので、腹に据えかねるものがあります。

それはさておき、256GBのSSDだと100GB弱は別のパーティションに割り振っても問題ありません。よって、Windows 10の［ディスクの管理］でパーティションを縮小します。Ubuntuのインストール時にもパーティションを縮小できますが、リスクがあります。Windowsからだと安全確実に縮小できるので、絶対にこの方法で行いましょう。

インストールメディアの準備

インストールメディア（USBメモリ）は別のUbuntu PCで［ブータブルUSBの作成］を使用して行いましたが、Windowsでも作成できます。姉妹連載のUbuntu

Weekly Recipe第415回[注1]が参考になります。

インストールメディアからのブート

インストールメディアを接続し、電源を入れたあと F2 キーを連打するとUEFIメニューが起動するので、[終了]タブで起動するメディアを選択してください。セキュアブートはオフにしなくてもインストールはできます。

インストール

無事にブートできたら、あとは普通にインストールするだけです。しかし無線LANを認識していないことに気づきました。16.04.2以降からインストールする場合はとくに問題ありませんが、16.04.1以前からインストールする場合は有線LANを使用するのがいいでしょう。筆者はとくに何もせずに使用できる無線LANアダプタ[注2]を使用しましたが、お勧めしません。

無線LANを使用する場合はパスフレーズを入力する必要があります。[設定]-[地域と言語]の[入力ソース]に[Japanese]を追加し、これを右上のインジケーターから変更して使用できるようにしておくといいでしょう。

インストーラを起動して進めると、[Ubuntu GNOMEのインストールと準備]がいつもと違うことに気づきました。セキュアブートがオンになっていると、サードパーティのドライバ(カーネルモジュール)が使用できないという問題があるため、ここでオフにできます。とはいえ、素直にUEFIからオフにしたほうが簡単です[注3]。今回は、インストール自体はオンでもできるため、このまま続けることにします。

[インストールの種類]では、当然一番上の[Ubuntu GNOMEをWindows Boot Managerとは別にインストール]を選択します。あとはとくに気をつけることはないので、インストールが完了するのを待ちます。

初回起動

再起動後、Panasonicのロゴが表示され、続いてgrubのメニューが表示されます。無事にデュアルブートに成功しました。

ホイールパッドのタップが効かない

ここからは本題の問題解決に移っていきます。まずは最も解決が簡単なものからです。

ホイールパッド(タッチパッド)をタップしても無反応で、いちいちボタンを押さなくてはいけません。これは面倒ですので[設定]-[マウスとタッチパッド]の[タップでクリックする]にチェックを入れます(**図1**)。

バックライトの調整ができない

Fn + F1 / F2 キーでバックライトの調整ができてほしいのですが、これができません。解決してしまえばなんてことはないので、解説は後回しにします。まずは次のコマンドを実行してください。

```
$ echo "blacklist panasonic_laptop" | sudo tee ↵
 -a /etc/modprobe.d/blacklist-panasonic.conf
```

続いて/etc/default/grubをroot権限で開き、

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

を、

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash acpi_↵
backlight=none"
```

図1　左下に[タップでクリックする]がある

注1) http://gihyo.jp/admin/serial/01/ubuntu-recipe/0415
注2) 本連載第82回(2017年2月号)が参考になります。
注3) そもそもどんなパスワードを入力したらいいのかわかりません。

に変更し、保存します。さらに続けて次のコマンドを実行します。

```
$ sudo update-grub
```

再起動後、Fn + F1/F2 キーでバックライトの調整ができるようになります。

デフォルトではバックライトの調整のためにpanasonic_laptopとintel_backlightという2つのモジュールが読み込まれていますが、実際には後者しか使用できません。というわけで、前者をブラックリストに入れることにより読み込ませないようにし、ブートのパラメータを変更してほかのバックライト関連モジュールを読み込ませないようにしています。

タイミングによって自動的に明るさが変更されることがあります。これを阻止するためには、[設定]-[電源]の[明るさ自動調整]を[オフ]にします。

ホイールパッドでスクロールができない

Let's noteのホイールパッドは、縁をなぞればスクロールできるはずですが、これができません。Let's noteでは右から下になぞるか上になぞるかで回転スクロールができるので、これに沿った設定にします。準備のために**図2**のコマンドを実行してください。

これらのコマンドで設定を変更するファイルのひな形をコピーし、変更します。変更点は**リスト1**のとおりです。

最初の[Section "InputClass"]の中に2行を追加します。[Option "CircularScrolling" "1"]で回転スクロールを有効にし、[Option "CircScrollTrigger" "3"]で右からスクロールを開始します。この値を"0"にするとどこからでもスクロールでき、"1"だと上、"3"だと右、"5"だと下、"7"だと左からになります。偶数はその間ということになります。

変更後、再起動すれば設定が反映されます。

HiDPI

12.1インチで1,920×1,200ドットのディスプレイをHiDPIと言えるのかどうかは微妙なところですが、そのままだと表示が小さ過ぎ、かといってGNOMEが対応しているHiDPI設定だと大きくなり過ぎるので使い物になりません。Windows 10のように150%といった設定ができればいいのですが、GNOMEでは整数でしか指定できず、2倍(200%)になってしまいます。

いくつか対策方法はありますが、一番簡単なのはTweak Tool(パッケージ名はgnome-tweak-tool)を起動し、[フォント]の[倍率]を[1.20]などにすれば、対応しているアプリケーションではフォントが大きくなり読みやすくなります。対応していない場合は個別に指定するしかありません。Firefoxはフォントサイズを絶対的に指定する必要があるようですが、

図2　設定を変更するファイルのひな形をコピーし、変更を行う

```
$ sudo mkdir /etc/X11/xorg.conf.d
$ sudo cp /usr/share/X11/xorg.conf.d/50-synaptics.conf /etc/X11/xorg.conf.d/
$ sudo editor /etc/X11/xorg.conf.d/50-synaptics.conf
```

リスト1　50-synaptics.confの差分ファイル

```
--- /usr/share/X11/xorg.conf.d/50-synaptics.conf    2016-03-03 22:41:40.000000000 +0900
+++ /etc/X11/xorg.conf.d/50-synaptics.conf    2017-01-03 16:52:07.465918852 +0900
@@ -11,6 +11,8 @@
         Identifier "touchpad catchall"
         Driver "synaptics"
         MatchIsTouchpad "on"
+        Option "CircularScrolling" "1"
+        Option "CircScrollTrigger" "3"
 # This option is recommend on all Linux systems using evdev, but cannot be
 # enabled by default. See the following link for details:
 # http://who-t.blogspot.com/2010/11/how-to-ignore-configuration-errors.html
```

Chromiumでは［ページのズーム］で相対的に大きくできます。

Caps Lock キーを Ctrl キーに

Caps Lock キーを Ctrl キーにする方法もいくつかありますが、一番簡単なのはTweak Toolを起動し、［タイピング］を開き［Ctrlキーの位置］を［Caps LockをCtrlとして扱う］にチェックを入れる方法です。

タイムゾーンの統一

デュアルブートでWindowsを起動すると、時間がずれてしまいます。自動的に調整されるものの、それだと不便な場合は、次のコマンドを実行してください。

```
$ sudo timedatectl set-local-rtc 1
```

ただし制限があるようです。詳しくはtimedatectlコマンドを何の引数もつけずに実行して確認してください[注4]。

サスペンド

デフォルトではディスプレイを閉じるとサスペンドしてしまいますが、これを阻止したい場合は/etc/systemd/logind.confの、

```
#HandleLidSwitch=suspend
```

を、

```
HandleLidSwitch=ignore
```

に変更し、再起動します。

Tweak Toolの［電源］-［ラップトップの蓋を閉じたときの挙動］でも設定変更できそうですが、今回は試していません。

HWEのインストール

インストール時点では16.04.1でしたが、これを16.04.2相当にアップグレードするため、次のコマンドを実行し、完了後再起動します。

```
$ sudo apt-get install --install-recommends↵
  xserver-xorg-hwe-16.04
```

これでカーネルとXスタックのバージョンが上がり、内蔵Wi-Fiも使用できます。ただし半年ごとにバージョンアップを行う必要があるので、その点はご注意ください。16.04からはHWEカーネルのサポート期間が短くなったことによる影響です。

テーマの変更

テーマはできればUbuntu(Unity)のAmbienceを使いたいのですが、一部のアプリケーションで困ったことになり、やはりGNOME Shell用に開発されたテーマでないと不都合がありました。ですのでTweak ToolでGTKテーマをNumixに、アイコンテーマのhumanity-icon-themeパッケージをインストールしたうえでUbuntu-mono-darkあるいはUbuntu-monoに変更すると、満足いくものになりました。

問題点

これまでのようにおおむね問題なく使用できていますが、それでもまだいくつか問題があります。VNCサーバであるvinoが自動的に起動しないのと、イヤホンジャックから音が出ません。自動的に起動しないものは手動で起動すればいいですし、［Pulse Audio 音量調節］(パッケージ名はpavucontrol)をインストールすると手動で内蔵スピーカーとイヤホンジャックを切り替えることができるようになります。

最大の問題はやはりHiDPIディスプレイ対応で、これはWaylandに移行すればなんとかなるはずです。Ubuntu GNOME 18.04 LTSのころには悩まなくなっているといいなと思っています。SD

注4) 説明を読む限りでは日本だけで使う分には問題なさそうですが、筆者はタイムゾーンに詳しいわけではないので断言はできません。

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

前回のゲーム企画から半年。ゲームとジョークコマンドを中心に、シェルスクリプト、コマンドの使い方を見ていきます。

第12回 ゲーム&ジョーク特集(その2)

2016年10月号の連載第6回に掲載したRogue(ローグ)とRog-O-Matic(ロゴマチック)の特集に続き、ゲームを題材に、周辺のコマンドを探検します。後半では、シェルスクリプトを駆使したゲームを制作してみましょう。これまでのテキスト処理技術や、シェルスクリプトの記述の仕方、コマンドの連結方法など、さまざまな応用が登場します。ゲームと一緒に、楽しみながら修得してください。

BSDGames(Debian系)

UbuntuなどDebian系のLinuxには、bsdgames、bsdgames-nonfreeというおもにCUIベースのゲームパッケージがあります。インストールは簡単にaptでできます。bsdgamesなのに、FreeBSDの系譜のmacOSにないとはなんともトホホですが……。本稿はUbuntu 16.04で確認した内容です。macOSへの対応は後ほど。

bsdgamesのインストール
```
$ sudo apt install bsdgames bsdgames-nonfree
```

ゲームの実行ファイルは、/usr/gamesに収められます。パッケージには、ローグクローン:**rogue**も見つけられます。ご存じTetris:**tetris-bsd**、Zork[注1]ライクなテキストアドベンチャーゲーム:**adventure**、飛行管制官の仕事を体験するAir Traffic Controller game:**atc**、トランプカードゲーム:**canfield**、欧米では古典ともいえるパーカーブラザースの、車でどちらが先に1,000マイル進めるかを競う人気のカードゲームMille Born:**mille**、数字を拾うたびに長くなる虫のゲーム:**worm**などがあります。CUIベースですが、いずれも楽しめる面白いゲームです。一方、1979年に日本でも翻訳版がアスキーから発売されて、当時ベストセラーとなったBasicゲームプログラムの本『Basic Computer Games』があります。この本にも掲載されていたゲームのUnix版とも言える2つのゲームに注目してみましょう。

TREKkie game – trek

Basic Computer Gamesでは、Super Star Trek[注2]というゲームです。**trek**は、決められた時間内に、銀河系内の敵(クリンゴン)を殲滅(せんめつ)して、帰還するというSFゲームです。壮大なゲームですが、後述の英文のドキュメントにていねいな解説があります(編注:著者による日本語訳を本誌Webサポートページにて提供予定)。トレッキーなら用語の解説もほとんど不要で、すぐにでもゲームにのめり込めるかもしれませんが、少しだけ紹介しておきましょう。

注1) infogram社から発売されたテキストアドベンチャーの大ヒット作。

注2) STAR TREK and related marks are trademarks of CBS Studios Inc.

この世界[注3]のgalaxy（銀河）は、8×8のquadrant（クオドラント）で構成されています。1つのクオドラントは、10×10のsector（セクター）で構成されています。探検隊向きのゲームで、コマンドはコマンドラインで入力していきます。各コマンド名は、他のコマンドと混乱がない範囲で短縮できます。

周囲を確認する

Short Range Scan（ショート・レンジ・スキャン）は、現在のクオドラント内の情報を見るコマンドです。**図1**が実行したゲーム画面です（図中文字の意味は**表1**参照）。左側の図に加えてStatus Report（ステータス・レポート）も表示されます。ショート・レンジ・スキャンが故障したら、statusコマンドで、レポートは確認することができます。また、Visual Scan（目視確認）で特定方向の観測ができます。

Long Range Scan（ロング・レンジ・スキャン）は、自分を中心にした周囲のクオドラントのクリンゴン（100の位）、ベース（10の位）、星（1の位）の数を表示します（**図2**）。

移動

移動は、Warp（ワープ）を使った高速移動か、安全低速なImpulse（インパルス）力によるかの2択です。

ワープ移動では、warp（ワープ・ファクター；1.0～10.0）を設定して、速度を調整します。6.0を超えると、エンジンが故障するリスクが高まります。9.0を上回ると、タイムワープの可能性が生じます。ワープでの移動は、“move <コース><距離>”と指定します。コースは角度（12時を0度、3時を90度として）、距離は1セクター0.1、なので1.0で隣のクオドラントまでが目安。斜めの距離は三平方の定理に従います。

注3）なぜだか2次元世界ですね。

▼図1　ショート・レンジ・スキャン実行（trek）

```
Short range sensor scan
  0 1 2 3 4 5 6 7 8 9
0 . . . . . . . * . * 0   stardate      3702.16
1 . . E . . . . . . . 1   condition     RED
2 . . . . . . . . . * 2   position      0,3/1,2
3 * . . . . # . . . . 3   warp factor   5.0
4 . . . . . . . . . . 4   total energy  4376
5 . . * . * . . . . . 5   torpedoes     9
6 . . . @ . .   . . . 6   shields       down, 78%
7 . . . . . . . . . . 7   Klingons left 3
8 . . . K . . . . . . 8   time left     6.43
9 . . . . . . * . . . 9   life support  damaged, reserves = 2.4
  0 1 2 3 4 5 6 7 8 9
Distressed Starsystem Marcus XII
```

▼表1　ショート・レンジ・スキャンで表示されているオブジェクトの意味

表記	意味
E	エンタープライズ号
K	敵＝クリンゴン
#	スターベース
*	恒星
@	生物のいる恒星系
.	何もない空間
␣	ブラックホール

▼図2　ロング・レンジ・スキャン実行（trek）

```
Long range scan for quadrant 2,5

     4     5     6
   -------------------
1 !   7 !   6 !   4 !
   -------------------
2 !   9 !  18 !   6 !
   -------------------
3 !   4 ! 202 !   6 !
   -------------------
```

インパルスエンジンで移動する場合は、“impulse <コース> <距離>”のコマンドを使います。すごく遅い移動ですが、ワープエンジンが壊れているなら、これを使うしかありません。

ナビゲーションシステムが故障したら、ミスター・スポックが調整してくれますが、完全な正確さを復元するには、スターベースに行ってなおさなければなりません。

艦載コンピュータ

Computer(コンピュータ)を呼び出せば、クエストのために必要なさまざまな情報が得られます。たとえば、chartで開拓済みのクオドラントの情報を表示、courseで方向と距離を計算してくれます。

クリンゴンを攻撃する

武器はtorpedo(光子魚雷)とphasers(フェイザー:エネルギー波)があります。魚雷は10本しか搭載できません。コースを指定して発射します。1発打つごとに減っていきますので、足りなくなったらスターベースに帰還しましょう。フェイザーは周囲にエネルギー波を放つ武器です。自動で周囲に撒き散らす方式と、マニュアルでコースを指定する方法があります。敵までの距離が遠いと、エネルギーが減衰して十分なダメージを与えることができません。

エンタープライズ号を守る

cloak(クローク)でクリンゴンから見えなくなります。クロークしているときは、こちらから攻撃ができませんが、クリンゴンのいるクオドラントにワープして突入するときは、先制攻撃を喰らわないようにするためにクロークしておくべきです。shield(シールド)を使えば、相手の攻撃のダメージを軽減できます。ただし、シールドをしたまま移動するときは、エネルギーを2倍消費するので注意しましょう。

abandonで艦を放棄、terminateでゲームを中断できます[注4]。

ゲームのドキュメント

manで見ると詳しい遊び方は、/usr/share/doc/bsdgames/trek.meを読めとありますが、筆者の手元の環境では、gzipで圧縮されています。また、見なれない.meという拡張子をもつこのファイルは、nroff[注5]で記述されています。中身を見るには、次のようにします。

```
$ zcat /usr/share/doc/bsdgames/trek.me.gz | nroff -t | less -r
```

zcatコマンドは、圧縮されたファイルから解凍ファイルを作らなくても、内容を出力できます。nroffは-tオプションを指定して、表の整形がきちんとできるようにします(筆者の手元のUbuntu 16.04環境では、.meのためのeマクロが導入されていないので-meオプションを指定していません。macOSなどeマクロが存在している環境でドキュメントを見るには、-meオプションも指定してください)。アンダーライン表示などが入るため、lessには、制御文字をそのまま表示する-rオプションが必要です。

STEP UP! nroffとマクロ

nroff/troffは、文章の形式(manpageにはanなど、マクロ名はmXXXというのがほとんどです)に合わせてマクロが用意されています。マクロの指定には、“-m<マクロ名>オプション(-manなど)”で指定します。nroff形式のファイルは、このコンベンション(慣例)に従って拡張子にどのマクロを使うかという命名をします。ですので、trek.meは、nroffでeマクロを使うのだろうと推察できるわけです。

Unix上で提供されているドキュメントは、このようにnroffやTeXなどのものが多くあります。なんだか記号ばかりで、とても読みにくいテキストだと思ったら、行頭‘.’で始まる制御文字がある場合はnroff、‘\xxx’のようなマークアップされたドキュメントの場合はTeX、と推察できます。もちろん、fileコマンドで推察するのも探検隊の流儀です。

hangman

単純だけど、ハマるのが単語当てゲームのhangmanです。7回ミスするまでに、隠された

注4) マニュアルとドキュメントには、ゲームの長さを選択するところでrestartと指定すればセーブデータ(dumpコマンドで作成)のところから再開できるとありますが、きちんと再開できないようです。

注5) 連載第1回(2016年5月号)で登場した、manpagesの記述に使う文書整形ツール

単語を当てなければ絞首刑になるというシリアスなゲーム!?　まあ、**図3**のゲーム画面をご覧ください。はじめのヒントは、何文字の単語か、ということだけです。アルファベットを1文字ずつ入れていくのですが、その文字が単語に含まれていれば、どの位置にあるかもわかります。単語や意味を知らなくても、英単語って、こういう構造のスペルの並びが自然なんだという勘所があれば、成功確率はあがるはずです。

▼図3　hangman

```
    ------
    |    |
    |    0                Guessed:  abcegiknorstuy
    |   /|\
    |    |                Word #:           5
    |   /                 Current Average:  8.400
  __|_____                Overall Average:  9.000
 |       |___
 |___________|

 Word:  backings
Guess:
You got it!
Another word?
```

look – LOOK – 与えられた文字列で始まる行を羅列する

hangmanは、デフォルトで単語を`look`コマンドと共通の辞書から選択します。指定したファイル中の、与えられた文字列で始まる行を抽出するのが`look`です。ファイル名を指定しないときは、`/usr/share/dict/words`から抽出します。「心理学は、phycで始まるはずだけどあとのスペルは……」などというときに便利です。

`look`の標準の辞書は米国英語ですが、英国英語、フランス語などさまざまなLocaleに合わせた辞書もあります。UbuntuなどDebian系のパッケージは、`/usr/share/dict`に導入するwamerican*、wbritish*、wfrenchなどです。

`look`は、`-b`オプションを指定すればあらかじめソートされたファイルから、2分探索によって高速に目的の行を抽出できます。`grep`よりもはるかに高速です（Ubuntu 16.04の日本語manpageは、`-b`オプションについての言及はなく、常に2分探索であるかのように書かれていますが、探索対象のファイルを与えた状態で`-b`なしでは2分探索は使いません）。

STEP UP!　macOSでBSDGamesを動かす

macOSでも、ソースを持ってきてビルドすれば良いはずです。すべてのゲームをビルドするには、何段階かのプロセスが必要かもしれませんが、trekとhangmanは、筆者の環境では、**図4**にある一連のコマンドでmanpageもドキュメントも含めて導入できています。trekはmライブラリ、hangmanはcursesライブラリが必要です。

trekのドキュメントは、次のコマンドで読むことができます。

```
$ nroff -me -t /usr/local/share/doc/games/trek.me | less -r
```

hangmanもどきのbangbanをshellで実現

ここからは、シェルスクリプトでhangmanの実現を目指します。macOSにも`look`はもちろ

▼図4　ソースからビルドする例

```
$ git clone https://github.com/vattam/BSDGames.git

$ cd BSDGames
$ ./configure

$ cd trek
$ gcc -lm -o trek *.c
$ sudo mv trek /usr/local/bin
$ sudo cp trek.6.in /usr/local/share/man/man6/trek.6
$ sudo mkdir /usr/local/share/doc/games
$ sudo cp DOC/trekmanual.nr /usr/local/share/doc/games/trek.me

$ cd ../hangman
$ gcc -lcurses -o hangman *.c
$ sudo mv hangman /usr/local/bin
$ sudo cp hangman.6.in /usr/local/share/man/man6/hangman.6
```

ん、辞書ファイルがありますので、シェルスクリプトで実現すれば、Macでも遊べます。オリジナルをできるだけ踏襲しつつも、失敗していい回数を固定の7回からオプションで指定できるように改善します。そのため、人を吊るすアニメーションは、導火線から爆弾に火がつくまでに正解するという演出にします。だからbangban ;-) プログラミングの基本、必要な部品を用意していきます。

STEP UP! bashの関数、コマンド引数の処理

bashスクリプトでも、他のプログラミング言語同様、再利用されるひとかたまりの処理は関数として定義しておきます。スクリプト言語ですので関数の定義は、呼び出される前に記述さ(読み込ま)れていなければなりません。関数の定義、引数の扱い、局所変数の定義の例として、スクリプト中で定義しているnot_in_listという関数を見てください。bashの関数は、まるでコマンドを実行するように呼び出すことができます。

```
第1引数に文字、第2引数に配列を受け取る関数
not_in_list()
{ # リスト中に文字が「無いこと」を確認
  local c=$1
  shift
  local list=($@)
  for cc in ${list[@]}
  do
    if [ $cc = $c ]
    then
        return 1
    fi
  done
  return 0
}
```

```
not_in_list の呼び出し側
not_in_list "$c" ${guessed[@]}
```

スクリプト実行時などに指定されるオプションの抽出処理は、組み込みコマンドのgetoptsを使うと簡便に記述できます。

辞書から候補の単語を取得する

辞書からアポストロフィ(')を含む単語を除外し、大文字は小文字に変換します。結果重複を避けるためsort|uniqでフィルタします。ゲームが繰り返し遊ばれることを想定して、ゲームで使うこの辞書データは、一時ファイルに書き出しておきます。一時ファイルのファイル名は、mktempコマンドを使って生成します。単語は乱数RANDOM(0〜32767)を読み出し、辞書の行数で余算、1行だけピックアップします。

wc -lで辞書の行数を数えていますが、wcの出力にはむだなスペースとファイル名が付いています。無用な部分はシェルの文字列演算を使って取り除いています。

```
一時ファイルを作って、辞書の大きさを測定
tempdict=$(mktemp /tmp/sdbangban.XXXXXX)
grep -v \' ${dictionary} | tr A-Z a-z | sort ➥
| uniq > $tempdict
dl=$(wc -l ${tempdict}) ; dl=${dl% *} ; ➥
dl=${dl#* }
```

一時辞書ファイルが存在しているうちに、プロセスが中断すると、一時ファイルが残ったままになる可能性があります。bashの組み込みコマンドtrapを使って割り込みをトラップし、きちんと後処理をしてから終了するようにします。

```
割り込み(HUP、INT、TERM)が生じたらgame_exitで後処理をする
trap 'game_exit' 1 2 15
trapをリセットしたい場合は、単純に "trap 2"とする
```

```
単語を1つだけ抜き出す処理
word=$(sed -n "$((RANDOM * RANDOM % dl + 1))➥
p" ${tempdict})
sedコマンドは、別の機会に詳しく解説します
```

1文字ずつ処理する

正解の判定を容易にするために、単語を1文字ずつ配列に分解して入れておきます。選んだ単語からreadに-nオプションを指定して1文字ずつ取り出し、配列wと回答が入る配列ansをそれぞれ初期化します。

```
この書き方は失敗例
echo $word | while read -n 1 c; do if [ ➥
"$c" != "" ] ; then w+=("$c") ans+=("-"); ➥
fi ; done
```

上記の方法では、パイプ(|)でつないだため、サブシェルが起動され、親シェルの配列wには追加できないので失敗します[注6]。bashでは、文字列をコマンドにリダイレクトするしくみ<<<を使えば問題を解決できます。

注6) 詳しくは、2016年12月号 第8回参照。

文字列のリダイレクトでスッキリ解決

```
while read -n 1 c; do if [ "$c" != "" ] ; ↵
then w+=("$c") ans+=("-"); fi; done <<< $word
```

キーボードから、1文字ずつ入力を受け付ける処理にもread -n 1を使います。

表示の演出：unicode、UTF-8の絵文字を表示させる

unicode（ユニコード）が普及して、コンソール環境でも絵文字などさまざまな文字が使えるようになりました。今回は、cursesライブラリを使うなどのキャラクタアニメーションはおこないませんが、文字の演出は頑張ってみましょう。bash 4.3などの新しい環境では、echo、printfでコード指定した文字を表示できます。ユニコードを指定するには、echoでは-eオプションを指定し、文字列中で\Uxxxxxのように記述します。

```
printf "    \\U1F480"
echo -n -e "\U1F525"
```

bash 3.2などでは、\Uを解釈しませんので、スクリプト中では\xによる16進コード指定で表示できるように表記します。ユニコードの文字を16進表記にするといくつになるのかは、次のようにodコマンドで確認します。文字列中で指定するときは、エンディアンを考慮します。

"\Uxxxx"表記で表示できる環境で16進コードを確認する

```
bash-4.3$ echo -e "\U1F480" | od -x↵
0000000      9ff0    8092    000a
0000005
```

特殊文字は端末によっては表示できません。そのような場合には、スクリプト中の変数の内容を表示できる文字列に書き換えてください。

特殊文字を定義している変数

```
# Skull: \U1F480, Fire: \U1F525, Spark: \↵
U1F4A5, Bomb: \U1F4A3
local skull="\xf0\x9f\x92\x80" fire="\xf0\↵
x9f\x94\xa5" spark="\xf0\x9f\x92\xa5" bomb↵
="\xf0\x9f\x92\xa3"
```

ゲームの完成

190行足らずでhangman互換のbangbanができあがりました（図5）。シェルスクリプト、あなどれません。;-) -dで辞書ファイル、-mで最短の文字数を指定できるのはhangmanと同様です。加えて、-rで最大の失敗回数を指定できます。7回では難しいという人は指定してみてください。

全ソースコードの掲載は紙幅の都合でいたしませんが、SDサポートページ[注7]からダウンロードしていただけます。本稿で解説したシェルの技芸以外にも、さまざま小技を使っています。ソースコード中にできるだけ詳しくコメントも記述しましたので、動作を確認しながら読み進めてください。

今回のまとめと次回について

今回は、ゲームを題材に、今までの技の応用を紹介しました。次回は、コマンドライン操作今昔です。SD

注7）URL http://gihyo.jp/magazine/SD/archive/2017/201704/support/

▼図5　bangban

```
    Word : b-----a--------
 Guessed : abcd
    💀( 2/ 7): 🔥___💣
Guess:e

    Word : b-----a--e-----
 Guessed : abcde
    💀( 2/ 7): 🔥___💣
Guess:f
```

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）】
hangman(6), trek(6), look(1), sed(1), mktemp(1), od(1), nroff(1), zcat(1), less(1)
【以下はbashのhelpコマンドを使って確認】
getopts, trap, printf

第60回

Linux 4.4の新機能
NVMe SSDを活用するI/O pollingとLightNVM

Text：青田 直大　AOTA Naohiro

2月19日に今年初めてのカーネルとなるLinux 4.10がリリースされ、その後さっそくLinux 4.11向けのカーネルへの変更が取り入れられつつあります。ですから今回から、Linux 4.4の新機能を紹介していきます。今回はおもにストレージ機能、とくに高速なSSDであるNVMe SSDを活用する機能を紹介していきます。

I/O polling

ハードディスクなどのコンピュータの周辺機器は、CPUやメモリに比べるとたいへん遅いものです。CPUの1サイクルがおよそ0.3ns、メモリへのアクセスが120nsに対して、ハードディスクへのアクセスは、1〜10msかかります[注1]。つまり、1回のメモリへのアクセスでもCPUは400サイクル、ハードディスクにいたっては約2000万サイクルが過ぎてしまいます。もちろんSSDでは、よりアクセスレイテンシは低下しています。それでもSATAベースのSSDの場合、1回のアクセスに0.5ms、すなわち約160万サイクルが必要となります。

このように大きな速度差があるため、ハードディスクに読み書き命令を出してから、その完了をひたすら待っているだけではCPUの時間を無駄に使ってしまいます。したがって、現在のOSのI/Oスタックは、リクエストの完了を割り込みで通知してもらうように作られています。こうして、ディスクにリクエストを送ってから完了するまでの間、CPUはほかの仕事ができるようになります。

しかし、最近ではよりレイテンシの低いデバイスが登場しています。たとえば、2014年のサムスン社のNVMeベースのSSDへのアクセスは110 μsと言われています。新しいNVMe SSDでは、よりレイテンシが低下しています。実際、15μsや6μsのデバイスが出ているようです[注2]。

こうした低レイテンシのデバイスにおいては、現状のI/Oシステムでは、かえって無駄が生じてしまいます。割り込みによってI/Oの終了を待てば、たしかにI/Oが行われている間にCPUはほかの仕事をできます。しかし、割り込みの処理はそれ自体、高コストなものであるため、すぐに処理が完了し、割り込みが来ることがわかっている場合には、割り込みはふさわしくあ

注1）Brendan Gregg "Systems Peformance", Table 2.2より。

注2）https://lwn.net/Articles/663543/

りません。また、スケジューラが間に挟まることで、アプリケーションから見たレイテンシは変動しやすくなります。このようにNVMeに対しては、せっかくのデバイスの性能をI/Oシステムが無駄にしてしまうことがあります。

そこで、Linux 4.4ではI/Oの完了をpollingで待つ機能が追加されました。この機能が使われると、ひたすらループして、I/Oが完了しているかどうかを調べるようになります。これにより、割り込みよりもレイテンシを下げることができます。

実際にサムスンのSSD 950 PRO NVMe M.2 512GBで、I/O pollingを使ってみましょう。このSSDは、/dev/nvme0n1としてシステムに認識されます。まず、通常の割り込みを使った方法で、I/Oレイテンシを計測してみましょう(**図1**の上部)。ダイレクトI/Oで、/dev/nvme0n1を4KBずつ、20万回読んでいきます。全体で4.3秒かかっているので、レイテンシは平均で21.1 μsです。次にI/O pollingを有効にして測ってみましょう(**図1**の中間部)。/sys/block/nvme0n1/queue/io_pollに1を書くことで、/dev/nvme0n1でI/O pollingが有効になります。同様のワークロードで、今度は3.0秒かかっているので、平均でレイテンシは14.8 μsとなります。差分としては6.3 μs程度ですが、I/O pollingによって元のレイテンシの3割程度が削減できたということになります。

/sys/block/nvme0n1/mqの下からI/Oの統計情報などを見てみましょう(**図1**の下部)。0、2、4、6のディレクトリはそれぞれI/Oキューであり、このデバイスの場合4つのI/Oキューを持っているということです。**cpu_list**を見るとそれぞれのCPUについて、1つのキューが割り当てられていることがわかります。これによって、CPU間での干渉を少なくI/Oを行うことができます。また**io_poll**を見ると、I/O pollingの動作状況を見ることができます。それぞれのキューについて、完了しているかどうかを見にいった回数が**invoked**に、そのうち実際に完了していた回数が**success**にカウントされています。ddが1つのCPUでしか動いていないので、6番のキューだけカウンタが増えています。192,740回のI/O完了を受け取るのに、45,656,732回のpollingを行っているので、平均して237回ほどのpollingでI/Oが完了していることがわかります。

Linux 4.4時での実装では、**queue/io_poll**が1になっていれば、どんなI/Oに対してもI/O pollingを行っていました。しかし、現実的にはI/O pollingを使ってレイテンシを低下させたいような、レイテンシに厳しいアプリケーション

▼図1　割り込みの場合とI/O pollingを使ったときの差

```
# uname -r
4.4.28-std490-amd64

（割り込みを使ったI/O）
# echo 0 > /sys/block/nvme0n1/queue/io_poll
# dd if=/dev/nvme0n1 iflag=direct of=/dev/null bs=4k count=200k
204800+0 records in
204800+0 records out
838860800 bytes (839 MB) copied, 4.32489 s, 194 MB/s

（I/O pollingを使ったI/O）
# echo 1 > /sys/block/nvme0n1/queue/io_poll
# dd if=/dev/nvme0n1 iflag=direct of=/dev/null bs=4k count=200k
204800+0 records in
204800+0 records out
838860800 bytes (839 MB) copied, 3.03643 s, 276 MB/s

# ls /sys/block/nvme0n1/mq/
0 2 4 6
# grep . /sys/block/nvme0n1/mq/*/cpu_list
/sys/block/nvme0n1/mq/0/cpu_list:0
/sys/block/nvme0n1/mq/2/cpu_list:1
/sys/block/nvme0n1/mq/4/cpu_list:2
/sys/block/nvme0n1/mq/6/cpu_list:3
# grep . /sys/block/nvme0n1/mq/*/io_poll
/sys/block/nvme0n1/mq/0/io_poll:invoked=0, success=0
/sys/block/nvme0n1/mq/2/io_poll:invoked=0, success=0
/sys/block/nvme0n1/mq/4/io_poll:invoked=0, success=0
/sys/block/nvme0n1/mq/6/io_poll:invoked=45656732, success=192740
```

というものは限られています。それらのアプリケーションのために、その他すべてのI/Oもpollingをして、CPU時間を無駄使いするのはあまり好ましくありません。そこでLinux 4.6からは、preadv2/pwritev2システムコールを使って**RWF_HIPRI**フラグをつけたI/Oリクエストだけが、I/O pollingで処理されるようになっています[注3]。

また、最新版Linux 4.10ではI/O pollingのadaptive modeが追加されています。さきほどの例で見たように、pollingでも14.8μsかかり、約50,000サイクルが費やされています。そこでpollingの前に、しばらくスリープしてからpollingを行えば、CPUの無駄を削減できます。Linux 4.10で**queue/io_poll_delay**というファイルが追加され、この値によってpollingの前のスリープが制御されています。"-1"の場合、これまでと同様にスリープなしにpollingを行います。正の数値の場合は、その指定のナノ秒分スリープしてからpollingを行います。"0"の場合には、それまでのI/O完了時間の平均の半分を、スリープ時間として自動的に決定します。

NVMe SSDや、preadv2/pwritev2というglibcには今のところインターフェースがないシステムコールを必要とはしますが、これまでのI/Oシステムに起因するレイテンシを削ることができるので、うまく活用してみたい機能です。

注3）https://www.spinics.net/lists/linux-fsdevel/msg92878.html

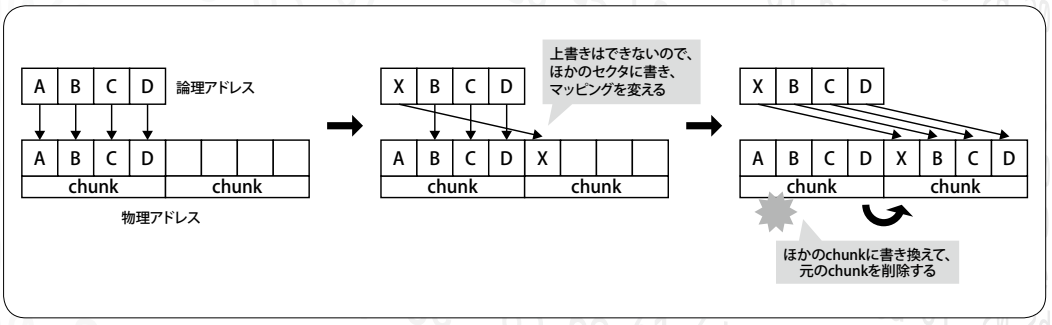
FTLの役割

次のトピックもNVMe SSDを用いるものです。SSDは、OSからはハードディスクのように見えています。しかし、当然ながら、その内部はハードディスクとは大きく異なり、したがってハードディスクとは異なる特性を持っています。たとえば、SSD上のセクタはハードディスクのように単純には書き換えることはできません。あるセクタを書き換えるには、まず、そのセクタを含むより大きなサイズの領域であるチャンクに対して、削除を行う必要があります。書き換えたいセクタだけでなく、その周辺まで削除され、なおかつ削除にはある程度の時間がかかるため、セクタを書き換えるために毎回チャンクを削除するというわけにはいきません。

FTLと呼ばれるSSD上のファームウェアは、こうしたSSDの特性を隠し、既存のI/Oシステムに適用するインターフェースを提供しています。たとえば、書き換えのときに削除が必要であることに対応するため、FTLは「論理アドレス」と「物理アドレス」とのマッピングを行います。

例として、**図2**のようにAと書いてあるセクタを、Xと書き換えることを考えてみましょう。前述のようにFTLは論理アドレスと物理アドレスのマッピングを管理しています。ここでは論理アドレス0番が、物理アドレス0番に対応し、そこに"A"と書いてあります。Xと書き換えるにあたって、そのまま上書きはできません。そ

▼図2　FTLによるセクタのマッピング



こで別のセクタにXと書いて、論理アドレス0番のマッピングを、新しいセクタを指すように更新します。また、前述したように削除はチャンク単位で行われるため、あるチャンク中の有効なセクタをほかのチャンクへと移動し、そのチャンクを削除してチャンクを空けるという処理(ガベージコレクション)も行います。さらには、Flashには書き換え上限があるので、一部のセクタだけが過剰に書かれないようにウェアレベリングも行います。

LightNVM: open channel SSD

このようにFTLは、SSDの面倒な詳細をラップしてハードディスクと変わらないように見せてくれます。しかし、I/O pollingのときと同様に、NVMeのようなデバイスを上手に使うにはFTLが邪魔になることがあります。たとえば、アクセスの激しいタイミングでガベージコレクションが走ったらどうなるでしょうか。普段であればとても小さいレイテンシが、ガベージコレクションのために急に大きくなってしまう可能性もあります。また、ファームウェア内のFTLは一般的なワークロードに向けて調整されているため、ワークロードによっては普通のFTLでは性能を活かしきれない場合も考えられます。

LightNVMは、FTLの機能の一部をOS側に移してしまうことで、FTLの機能をよりアプリケーションと協調できるようにするシステムです。マッピングやガベージコレクションといった機能を、OSやアプリケーション側で実装できるようになります。すると、たとえばアクセスの激しい間はガベージコレクションを止めておいて、その代わりにメンテナンス中や深夜などにガベージコレクションを一気に行うといった細かい制御も可能となるはずです。

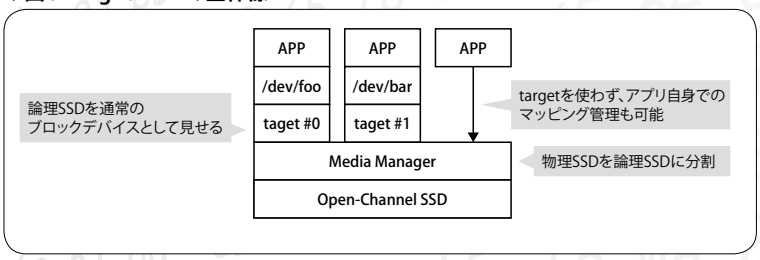
LightNVMのしくみ

まず、LightNVMの全体像を見ていきましょう(**図3**)。LightNVMのデバイスは、下から❶LightNVMに対応したNVMe SSD(Open-Channel SSD)、❷Media Manager、❸targetの階層から成ります。このうちMedia Managerの部分は、現在開発中のLinux 4.11では削除される見込みです。

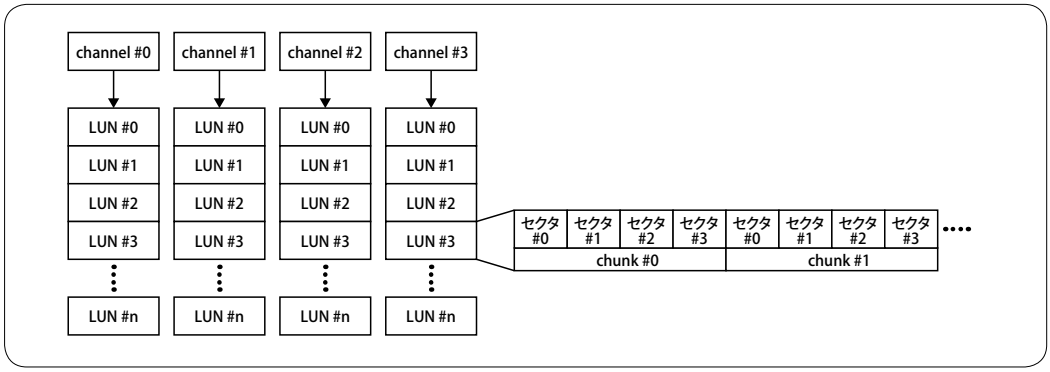
下から見ていきましょう。LightNVMを使うには、対応するNVMeSSD(Open-Channel SSD)が必要です。Open-Channel SSDは、一般のSSDとは違うプロトコルをサポートします。一般のSSDでは、内部の物理的レイアウトを隠し、論理アドレスでのアクセスを受け付けるプロトコルになっています。一方で、Open-Channel SSDは物理構造がどうなっているかの情報を提供し、物理的アドレスで読み書きを行うプロトコルを提供します。

Open-Channel SSDの内部構造を**図4**に示します。セクタは読み書きの最小単位です。セクタがいくつか集まってチャンクを構成します。前述したとおり、チャンクが削除の単位となります。また、チャンクがいくつか集まってLUN(parallelunit)を構成します。異なるLUNの間では、同時にアクセスすることが可能となっています。さらに、LUNがいくつか集まってチャンネルにつながっているという構成になっています。Open-Channel SSDは、たとえば「チャンネル#0、LUN#1、チャンク#2、セクタ#3」といった物理アドレス(Physical Page Address:PPA)

▼図3 LightNVMの全体像



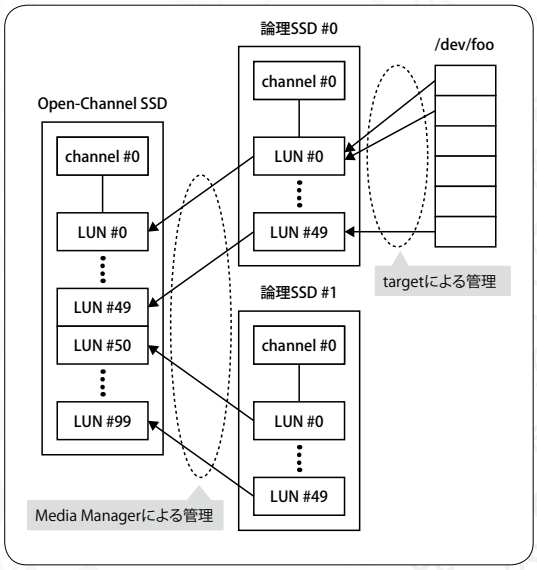
▼図4　Open-Channel SSDの構造



で、読み書きのコマンドを受け付けます。

Media Managerは、1つのSSDにつき1つ動作し、下のSSDのLUNを分割して論理的なSSDを構成し、どのtargetがどのLUNを使うのかを管理します。たとえば、SSDが100個のLUNを持っているとします。これを50個ずつに分割し、2つのtargetにそれぞれ提供します。どちらのtargetからも、50個のLUNを持つSSDを独占しているように見えます。もともとは、この部分はpluginとなっており、Linuxに実装されているgennvm（General Non-Volatile Memory Manager）以外のものを使うことができるようになっていました。しかし、前述したとおりLinux 4.11からは、gennvmがLightNVMのコアにマージされ変更できなくなる見込みです。

▼図5　targetとMedia Managerの動作



Media Managerの上ではtargetが動作します（**図5**）。targetはMedia Managerが、提供する論理Open-Channel SSDを、たとえばブロックデバイスとして見せるなどマッピングやガベージコレクションを行う部分です。アプリケーションに適応したブロックデバイスを作りたい場合には、この部分をいじることになります。

今のところ、Linuxにはrrpc（Round-robin Hybrid Open-Channel SSDtarget）というtargetのみが実装されています。このtargetは、書き込みのたびにラウンドロビンにLUNを変えて、書き込み先を決定し、有効な（マップされている）セクタが最も少ないチャンクを削除するというシンプルなtargetとなっています。なお、ここでHybridとついているのは、target自身が論理アドレスと物理アドレスとのマッピングを保存せず、SSDの方にマッピングの保存を任せているためだと思われます。

LightNVMを試す

では、実際にLightNVMを試してみましょう。実用的には専用のSSDが必要となりますが、動作を見てみる程度であれば実デバイスがなくても大丈夫です。これには2つの方法があります。1つはpatchの当たったQEmuを使って、Open-Channel SSDがつながった仮想マシンを作る方

法です。もう1つは、I/Oシステムのパフォーマンス評価に用いられるnull_blkという何もしない疑似ブロックデバイスを使う方法です。QEmuの方はビルドが必要ですので、今回はとりあえずコマンドの動きを見られればいいということでnull_blkを使います。

まず、nullブロックデバイスをLightNVMに認識させます。モジュールパラメータuse_lightnvm=1を使うことで、nullブロックデバイスが、LightNVMに登録され、/dev/nullb0といった形では出てこなくなります。デフォルトでは、2つのデバイスnullb0とnullb1が作成されます。

Media Managerの設定は、nvme lnvm initで行います(図6)。-dがデバイスを指定します。-mでMedia Managerを指定することもできますが、デフォルトがgennvmですので省略しています。あらためてnvme lnvm listを見ると、Media Managerにgennvmが設定されていることが確認できます。前述したように、Linux 4.11以後では、gennvmがコアに統合されるのでこの作業はいらなくなるでしょう。

最後にnvme lnvm createコマンドでrrpcをtargetとして設定します(図7)。ここでは省略していますが、-bと-eでそれぞれtargetに割り当てるLUNの開始位置と終了位置を設定できます。ここではとくに指定してないので、1つのLUNが割り当てられています。Linux 4.10の時点では、有効なLUNの外を指定するとバグでカーネルハングを起こすようです。試してみるときには注意してください。

これで"/dev/lnvmtest"として通常のデバイスファイルのようにアクセスできるようになります。I/Oリクエストを処理しているように見せかけるだけで、実際にはデータを保存しないので、これ以上とくにできることはありませんが、LightNVMの操作の感じはわかっていただけたのではないかと思います。

targetの削除はnvme lnvm removeコマンドを使います。Linux 4.10時点では、nvme lnvm initされていないディスクがあると、カーネル内のNULL pointerreferenceでコマンドが失敗します。あわてず、すべてのディスクをnvme lnvminitしてからやりなおせば問題なく削除できます。

まとめ

今回は新しいタイプのSSDであるNVMe SSDをより活用するための機能としてI/O pollingと、LightNVMを紹介しました。LightNVMは、バグが残っていたり、Media Managerがコアに統合されたりするなど、まだまだ成長途中といった様子です。しかし、NVMeをより細かく自由にアプリケーションから制御できるようになるわけで、楽しみな機能です。SD

▼図6 Media Managerの設定

```
# modprobe gennvm
# nvme lnvm init -d nullb0
# nvme lnvm list
Number of devices: 2
Device          Block manager   Version
nullb1          none            (0,0,0)
nullb0          gennvm          (0,1,0)
```

▼図7 targetの設定

```
# modprobe rrpc
# nvme lnvm create -n lnvmtest -d nullb0 -t rrpc
# dmesg
...
[232599.325599] nvm: rrpc initialized with 1 luns and 16384000 pages.
# fdisk -l /dev/lnvmtest
Disk /dev/lnvmtest: 56.3 GiB, 60394180608 bytes, 14744673 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
# nvme lnvm remove -n lnvmtest
```

April 2017

NO.66

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
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp

できるエンジニアはクリスマスも旧正月も勉強会

今回は、昨年12月に行ったシェル勉強会と、今年1月に大阪で行った研究会の模様をお伝えします。

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）、
上田 隆一（USP友の会）
【日時】2016年12月25日（日）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8回初心者の方角に向いて講師が喋(しゃべ)るシェル勉強会（初心者向けとは言ってない）／第26回シェル芸勉強会及びエクシェル芸勉強会」を開催しました。参加者は29名でした。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。

午前の部は鳥海さん、午後の部は上田さんが講師を担当されました。その後、LT大会・懇親会が同じ場所で開催されました。

■午前の部

最初に無類のeval好きであるなど講師の自己紹介のあとに、「黒い画面と戯れよう」というタイトルで勉強会が始まりました。黒い画面とは、キーボード入力と文字表示のみでコンピュータの操作を行う画面のことと定義されていました。さまざまなグラフィックとともにマウスやタッチパネルなどを使ったユーザインターフェースとは異なり、文字だけが表示される画面の背景が黒いことから黒い画面と呼ばれます。講師によると、この黒い画面が怖いという初心者もいるそうです。その怖い体験談として、キーボードの操作を間違えると、入力を受け付けなくなること、アプリケーションが中断して戻れなくなることなどが挙げられました。うっかり間違えたキー操作をして戸惑わないようにしたり、機能を知って便利に使えるようにしたり、そういう勉強をするのが今回の趣旨であると説明されました。

そして、bashにリンクされているライブラリの機能について、実際に設定を行ったデモンストレーションを交えて説明が行われました。emacsモードやviモードを使った高度な機能もあり、これらをしっかり覚えればより便利に使えると思いました。

■午後の部

午後は6人ごとに班を作り、参加者同士が互いに協力して問題に取り組める形式にして行われました。

講師からの近況・シェル芸の説明のあと、今回はMicrosoft Office系の文書ファイルに対して、読み書きやファイル生成を行うことがテーマであることが説明されました。Officeの文書ファイルに対して操作ができるようになることにより、UNIX系OSの環境で、自動で処理できることを目指すということでした。なお、今回はかなり難解になるため、ワンライナーだけでなくシェルスクリプトを使った回答もOKになりました。問題に使うファイルはいつものとおり、GitHubで提供されました。

最初に、Office文書はzipで圧縮されたファイルであることが説明されました。このzipファイルを解凍して中身を見て、それを再度圧縮したものをアプリ

ケーションから開けることを確認しました。その後、PowerPoint、Excel、Wordなどの文書ファイルの中身を読み、その中から目的のデータを取り出し、編集し、新しい文書ファイルを生成することなどが問題として出されました。

勉強会を通じて、zipで圧縮されたOfficeの文書の中身の読み方、戻し方、XML文書自体の読み方について理解を深めることができました。また、zip、unzip、tree、xmllint、hxselectなど、Officeの文書ファイルを読み書きするときに便利なコマンドについても、実際に操作しながら解説されました。

■懇親会

さくらインターネットのご好意により同じ会場で、ビア・バッシュ形式の懇親会が開催されました。自己紹介やLT大会が行われ、自由に歓談し、楽しい雰囲気の中で進められました。LT大会では、シェル芸に関係なく、発表者がそれぞれに得意な分野について発表し、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。それぞれが負担にならないように楽しく開催される予定ですので、可能な範囲で協力していきたいと思います。

jus研究会 大阪大会

■ホビーロボット最前線
～オープン環境で創る2足ロボット～
【講師】やまだはるひろ（ホビーロボット研究会）
【日時】2017年1月28日（土）14:00～14:45
【会場】大阪産業創造館 5F 研修室E

昨年11月に続いて大阪での研究会を、オープンソースカンファレンス2017 Osakaの中で実施しました。講師には、やまださんをお迎えし、おもに2足歩行ロボットを中心に、趣味で楽しむロボットの世界を紹介いただきました。参加者は20人でした。

■ホビーロボットの歴史と現状

前半は、約20年におよぶホビーロボットの歴史と現状に関する話でした。ホビーロボットとは、産業用でも家庭用でもない、趣味のロボットの総称です。このうち2足歩行ができるものは、それを操作してサッカーや格闘技などをさせることができます。やまださんは近畿学生2足ロボリーグを運営していて、関西地区の大学対抗による格闘技の試合をしたり、そのための練習会や合宿、試合映像の撮影と公開などを行ったりしています。

ロボットはもともと、金属・樹脂加工、機構学、運動学、センサー技術、制御技術、ヒューマンインターフェースなどの複合技術であり、各分野が連携なく進展していました。それがインターネットの普及により各分野の情報が検索可能になり、また各分野の人が出会える状況が作られたことがホビーロボットの発展につながりました。とくに2足歩行ロボットの格闘競技大会・ROBO-ONEが良いきっかけになったそうです。

■ホビーロボットの作り方

後半はホビーロボットの作り方について解説がありました。ロボットを構成するおもなパーツとしては、外装・フレーム、サーボモーター、マイコン、ファームウェア、操縦器などがあります。これらを個別でそろえることもできますが、初心者はキットを買うのがお勧めです。最近はAndroidのスマートフォンで操作できるキットもあるそうです。ソフトウェアの開発は、かつてはロボットごとに異なる開発環境で、Cで書いたプログラムをコンパイルしてマイコンに書き込んでいましたが、最近はArduinoやRaspberry Piで制御できるようになりました。さらにROS（Robot Operating System）というOSSのフレームワークも登場しています。また、「データ層はオープン」とのことで、ロボットのモーションデータや音声データを共有することもさかんに行われています。

講演に際しては、多数の写真が掲載されたスライドに加えて、ロボットの実機を何台か持ち込み、実際に動かしながら解説されました。見た目にも楽しめる研究会でした。SD

Hack For Japan
エンジニアだからこそできる復興への一歩

Hack For Japan スタッフ
小泉 勝志郎　KOIZUMI katsushiro
Twitter @koi_zoom1
高橋 憲一　TAKAHASHI kenichi
Twitter @ken1_taka

第64回 南相馬市小高の学生生活を楽しくしたい!〜小高ハッカソン

2016年7月に避難指示が解除された南相馬市の小高区。今年の4月から高校も再開されるため、学生生活が楽しくなる街にするには、どんなことができ、どう関わっていけるかをハッカソンで考えてみました。

2017年1月13日から15日にかけて福島県南相馬市小高区にて『「新しい」って、ワクワクする。自分でつくればもっと楽しい。』というキャッチフレーズでハッカソンが開催されました。南相馬市に拠点を置くIT事業者などのメンバーで構成される小高ハッカソン実行委員会の主催で、この小高でのハッカソンは2016年の3月（本連載の2016年6月号でレポートしました）に続いて2回目となります。

開催の経緯

南相馬市の小高区は福島第一原発から10kmから20kmの場所にあり、昨年のハッカソンが開催された際には避難指示の解除を目前に控えた時期でしたが、2016年の7月に避難指示は解除されました。現在はJR常磐線の小高駅も再開されており、2016年の12月からは仙台方面からこの駅まで直通の電車で来ることができるようになっています。

そして今年の4月から高校が再開（小高にあった商業高校と工業高校が1つになり、小高産業技術高校としてスタートします）し、500人の高校生が再びこの地に通うようになるということで、「再開する学校生活に向け、街や通学・学校生活を楽しくする」というのが今回のテーマです。「みんなでおしゃべりする場所、部活の終わりに買い食いして帰るいつものお店。土日に遊ぶデートスポット。まだまだ足りないものばかりですが、だからこそ、エンジニアやデザイナーの皆さんで、ここに付け足すエッセンスをみんなで考え、作り出してほしい」そんな実行委員会の皆さんの思いでこのハッカソンは開催されました。

今回の会場も前回と同じく、小高駅のすぐ近くにある双葉家旅館で、参加者は石巻や仙台、東京から来たエンジニア、南相馬市出身のエンジニア、会津大学の学生、そして今回は三十数名の参加者中10人を超える地元の高校生が参加していました。

街歩き

初日は小高ワーカーズベース[注1]の和田智行さんの案内で街歩きを行いました。この数ヵ月で以前の10％の人口にあたる約1,000人が帰還し、少しずつコンビニなどの商業施設もできてきているものの薬局はないなど、まだまだ十分ではないとのことです。そのような中でも美味しいと評判のお寿司屋、ラーメン屋などが明かりを灯しているのは、少しずつ生活が戻ってきていることを感じさせてくれる風景でした。

ハッカソンの成果

アイデアソン、アイデアピッチを経て8つのチームで開発に取り組み、最後の成果発表は地元の皆さんも多く見学される中で行われました。審査員は小高ワーカーズベースの和田さん、チームラボの床並展和さん、昨年のハッカソンのシェア馬チームを率いた山下敏義さん、Hack For Japanスタッフの及川卓也さんの4名で、及川賞、RESAS賞、最優秀賞の各賞が用意されました。ここではHack For Japanスタッフが関わったチームと各賞の受賞チームについて紹介します。

Lightning Prikura

筆者（高橋）の参加したチームで、地元の高校生、

注1　http://owb.jp/

会津大の学生を含む6人のメンバーで構成されました。初日のアイデアピッチで高校生が「プリクラはあるけど機種が古い」と言っていたこと、地元の企業製による3Dプリンタが会場に設置されていたこと、提供されていた機材の中にIntel RealSenseがあったことで、自分達でもっと面白いもの、3Dプリクラを作ったらどうだろうというのが着想の元です。

RealSenseで人の顔を3Dスキャンし、あらかじめ複数用意してあるものから選んだアクセサリの土台を合成して3Dプリンタで出力するというしくみを開発しました。RealSenseの制御がSDKのとおりに作ったにもかかわらず正しく動作せず、別の手段でスキャンしてあった顔データを使用することにはなりましたが、顔データと土台となるモデルの合成はOpenSCAD[注2]というスクリプト制御可能なCADソフトウェアを使用して自動化し、**写真1**のように実際に3Dプリンタで出力するところまで検証できました。

ぼっちラボ

筆者(小泉)は、南相馬市には2年ほど前までiPhoneアプリ開発講座の講師として毎週行っていたのですが、今回は久々に訪れることとなりました。アイデアピッチの結果1人チームとなり、RESASの説明をしてくださった床並さんの会社であるチームラボさんをもじって「ぼっちラボ」というチーム名にしました。

作ったのは「萌えキャラGO～萌える公園編～」。「それどこかで聞いたことある」と思った方はよく訓練されたSoftware Design読者さんです。本連載の2016年10月号でレポートした、石巻ハッカソンで作成した「萌えキャラGO」に「小高に聖地を作ろう」というコンセプトでRESASを組み込んでみました。

実際に組み込もうとしたRESAS APIは市区町村取得、およびパートナーAPIである駅混雑予報(ナビタイム)と駅天気予報(ヴァル研究所)の3つで、**図1**がこれらを使って作成した画面

◆**図1　RESAS APIを使用した「萌えキャラGO～萌える公園編～」の画面**

注2　http://www.openscad.org/

◆**写真1　3Dプリクラの出力例**

Column　RESASについて

今回のハッカソンではRESAS(Regional Economy Society Analyzing System)[注A]という地域経済分析システムの紹介があり、このAPIを用いたチームに贈られるRESAS賞も用意されました。地方自治体のさまざまな取り組みを情報面から支援するため、内閣官房まち・ひと・しごと創生本部事務局から提供されており、産業構造や人口動態、人の流れなどのさまざまな官民のデータを集約し、可視化するシステムです。

RESASでは国や地域からの数多くの情報がデータ化されており、文字の羅列から情報を分析するのは難しい作業ですが、視覚的に工夫することで地域ごとの情報が直感的にわかりやすくなるようにしてあります。APIも公開されており[注B]、内容は人口の動向から観光、農林水産業と多岐にわたります。これらのAPIはフォームから利用登録した際に発行されるAPIキーを、リクエストヘッダの「X-API-KEY」に入れるだけで手軽に利用できます。そしてこのRESAS APIでは、ナビタイムの駅混雑予報などのパートナーが提供するデータも使うことができます。

説明をしてくれたチームラボのカタリスト、床並展和さんの話は「人から聞いた情報からなんとなく知るのではなく(RESASを用いて)とにかく地域課題を自分ゴトに!!」という熱い言葉で締められました。

注A　https://resas.go.jp/
注B　https://opendata.resas-portal.go.jp/

です。

今回のハッカソンのRESAS賞の要件として、“別に開催されるRESASアプリコンテスト[注3]の「パートナーのものではない純正RESAS APIを用いる」という応募条件に準ずる”というものがありました。筆者が使用した中で純正のRESAS APIは市区町村取得APIで、「南相馬市」という名前を取るために使用しました。これにより仮に南相馬市の名称が変更されても自動的に対応されるというメリットがあります。

今回は「小高区」を指定してデータを取りたかったのですが、小高という単位で取得できるのはパートナーAPIの「駅混雑予報」と「駅天気予報」くらいで、RESAS純正のデータは都道府県単位や市区町村単位で取るものばかりでした。そのため、少々こじつけた感はあるものの純正RESAS APIである市区町村取得を使用しました。RESASのサイトを見ると市区町村や都道府県単位の情報を全国で比較するというものがほとんどで各地域の中を細かく見るものはないため、もっと細かい単位でデータを取得できるとより使い道が増えそうなので今後に期待です。

そして、パートナーAPIであるナビタイムの駅混雑予報は「混んでいるときのみしかレコードが返って来ない」という仕様のようなのですが、ここは時間帯ごとのレコードで返って来てほしかったところです。指定した期間に混雑が予想される時間がないとデータが返ってこないのですが、それだとデータがないのか混雑が予想されないのかの区別がつけられません。小高の場合は火曜日のみデータが返って来て、それが火曜日だけが混んでいるという実際の予想を示しているのか疑問がありました。

また、今回使おうと思ったもうひとつのパートナーAPIの駅天気予報は、登録後に「数日中に弊社担当から連絡します」とのメッセージが返ってきました。このように使おうと思ったその日には使えないものもあるということは覚えておいても良いかと思います。

注3　https://opendata.resas-portal.go.jp/contest/index.html

各賞と受賞チーム

●及川賞：小高ポータル

待ち合わせの場所を作ろうという課題に取り組んだチーム。着想はIngressのポータルのようなかっこいいものを現実に置きたいというところから始まりました。現実世界に作成されたポータルを待ち合わせ場所にするためのアプリを開発し、**写真2**のようなポータルとして設置するオブジェも工作して、実際に小高駅に設置して写真を撮って思い出に残すデモがありました。

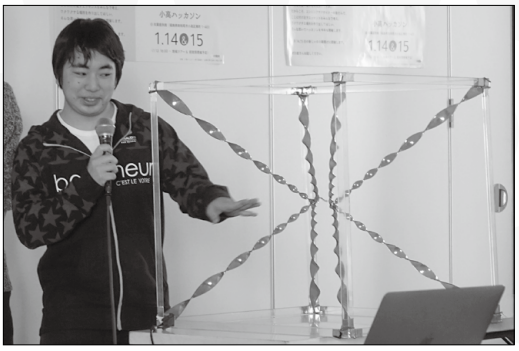
同チームの高校生は「こういう近未来的な新しいものを小高に置くのは新しくて面白い、今後こういう待ち合わせ場所がもっとできればさらに面白い」と言っていました。アーティストや宣伝したい人がポータルを作成して地域振興に役立て、さらにブロックチェーンを使った地域通貨小高ポイントのしくみも作成し、チェックインによるインセンティブなどに使うことができます。発表の最後には「アートと位置情報と地域通貨で新たな街づくりをしていきませんか」という言葉がありました。

審査員の及川さんからは「ポータルが無いなら自分達で作ってしまおうという発想、ブロックチェーンを使って地域ポイントを作るという発想が面白いと思った」というコメントがあり、賞品として及川さんの対談が掲載されている書籍「デベロッパーのキャリアと働き方を語ろう vol.2」が贈られました。

●RESAS賞：＃小高つくってみた

RESAS APIを使ったチームが対象となる賞で、

◆写真2　ポータルのオブジェ



データや思いを見える化して街を自分化するしくみを開発したこのチームに贈られました。

さまざまなアイデアを持つ高校生も(街作りの)経営に参画し、もっと楽しく自分の街にしていくことができるようにしたいと考え、Twitterで「#小高つくってみた」というハッシュタグでつぶやくだけで参加できるようになっています。

地域の小売店のPOSデータとRESAS APIを使って小売に関するデータを取得して表示するアプリを開発し、今売れているものを見える化しました。さらにお店に高校生の企画のための棚を設置し、アプリによって売上や在庫切れが手元でわかるようにすることで、自分で行って補充ができるしくみを想定しました。このしくみを、街を使ったリアルゲームとして発展させ、商品を卸値で購入／販売して大きくしていくことをミッションにした、わらしべ長者的ゲームとして提供することで、継続性を高める構想もしているとのことです。

賞品はチームラボの特製ステッカーと東京土産のお菓子が贈られました。

審査員の床並さんからは「共通のキーワードで小高が頑張っている姿を発信していくというのはバズりそうなリアルな絵がうかんだ。高校生が自分達で何があったら良いのかということをゼロベースで作っていく姿は未来があると思った。」というコメントがありました。

●最優秀賞：OMSBハック

小高にはOdaka Micro Stand Bar(OMSB)という軽ワゴンの移動カフェがあるのですが、それをもっと人が集まるところにして盛り上げていきたいという課題に取り組んだチームです。解決策として、高校生が集まる楽しいところにしたい、カフェに面白いものを置こう、ということで2つのものを作成しました。

1つめはHue[注4]とMaKey MaKey[注5]を使った相性チェッカーで、男女が手をつなぐと相性が良ければピンク、悪ければブルーにHueが光るしくみになっています。

◆写真3　高校生が作成したゲーム「LOVE RAIN」

2つめはLOVE RAINという、降ってくるハートを手で受けて落ちないようにするゲーム(**写真3**)です。今回のハッカソンで初めてプログラミングに触れた高校生がScratchを使って作ったもので、継続的に高校生がプログラミングを学ぶ流れが広がっていくことを想定しています。高校生の感想で「今までプログラミングって黒い画面にカタカタ文字を打ち込んでいるイメージがあったが、試行錯誤して何かができたときの達成感が本当に面白かった」と言っていたのが印象的でした。

審査員の和田さんからは「高校生がやって、しかも実現性が高い、この先継続してつないでいくという点が魅力」というコメントがあり、賞状には「学生生活をワクワクさせる優秀な作品を開発されたのでこれを賞します」という文字もありました。賞品は主催者の森山貴士さんセレクトによる「浪江～相馬まで、それぞれの街のよいもの詰め合わせ」が贈られました。

今後

2回目の開催となった小高ハッカソンは大いに盛り上がった中で無事に終了しました。本連載でも毎回レポートしているとおり、すでに5回の開催を数える夏の石巻ハッカソンに続き、冬の小高ハッカソンとして東北の太平洋側の二大ハッカソンとして恒例のイベントになっていってほしいと思います。SD

注4　ネット経由でON/OFFや色を制御できるLED照明。
注5　身の回りのものをパソコンにつなげて、入力ディバイスやコントローラを自分だけのオリジナルにカスタマイズできる電子創作キット。

温故知新
IT むかしばなし

第64回

MC68000 ～黎明期の高性能16bit CPU～

速水 祐（はやみ ゆう） http://zob.club/ twitter @yyhayami

はじめに

本連載は、ITの歴史を振り返り、未来への糧にすることを目指しています。今回は、「Break away from the past（過去を打ち破る）」のメッセージのもとに、8bitの互換性を捨てて理想を求めて開発された16bit CPU MC68000のお話をしましょう。

8bit CPUから16bit CPUへ

1970年代後半、8bit CPUが16bitに進化しようとしていました。Intel社の8bit CPUであるi8080は人気を得て広く使われ、互換性を維持しながらi8085やZilog社のZ80へと進化し、より利用範囲を広げていきました。そして8bit CPUとのアーキテクチャとしての互換性を目指しながら16bitのCPUとしてIntel 8086が開発され、1978年に登場します。

対してモトローラ社は、8bit CPU MC6800を発表していましたが、i8080の後塵を拝しました。そこで、優れた開発陣により2つのCPUの設計が行われました。1つは究極の8bit CPUと呼ばれたMC6809で、MC6800の互換性をある程度保ちながら、豊富なアドレッシングモードや16bit演算機能などの優れた機能が追加されたものです。もう1つが16bit CPUのMC68000で、MC6800の互換性から逃れて、多くのTTL（Transistor-transistor-logic）で構成されたCPUを持つミニコンであるPDP-11/VAX-11やIBMのメインフレームであるSystem/360のアーキテクチャを参考に設計され、マイコンのレベルを遙かに超えて上位のコンピュータ利用も考えた非常に高機能なCPUであったのです。

MC6809、MC68000ともに1979年に登場しますが、16bit CPUとしてのスタートがi8086に比べて1年遅れだったことが致命的になります。1981年に登場するIBM-PCは、設計の時点ではまだ試作品であったMC68000は採用されず、i8088（i8086のデータバスを8bitにしたCPU）を搭載することになったのです。同様なことがZ80の後継の洗練されバランスよくまとめられたZilog社のZ8000にも当てはまり、そのあとはi8086の独走が始まるのです。

MC68000の機能

MC68000は、16bitの外部データバス、16MBのメモリ空間へのアクセスが可能な24ビット外部アドレスバスがICのピンとして出ており、64ピンの巨大なDIPパッケージ（Dual Inline Package）で登場しました。内部は、トランジスタ数が名前と同じく68,000個以上とi8086の29,000個の2.3倍以上で作られています。内部設計は32ビットになっており、32ビットの8個のデータレジスタ[注1]と8個のアドレスレジスタ（スタックポインタを含める）があります（**図1**）。

当時、高性能なプログラムを作り出すためにはアセンブラを使う頻度が高く、使いやすい命令体系が求められていました。i8080/Z80では、レジスタと命令との組み合わせのルールがあることでレジスタを自由に使えない問題があり、MC6800/6809ではレジスタの少なさで苦労を強いられていました[注2]。それに対

注1） Register。CPUが最速で扱える記憶回路。

注2） レジスタごとに使える命令が限定され、計算専用のものや、カウンタ専用のものなどがあり、その分プログラムが複雑になっていた。

してMC68000は、命令に直行性[注3]があり8個のデータレジスタとアドレスレジスタを自由に使うことができる、アセンブラプログラマからの羨望の的となるCPUでした。

MC68000は、高機能な命令を実現する2段階マイクロプログラム制御を採用したCISC(Complexed Instruction Set Computer)であるため、1命令の実行時間はそれなりに要しました[注4]。それでも、16bit幅の演算処理回路を3組内蔵して並列処理可能な構造により、クロック周波数が8MHzのとき、当時のミニコンであるVAX-11/780とほぼ同じ1MIPS(Million Instructions Per Second)の高速性を実現できていました。

MC68000が搭載されたコンピュータ

MC68000は、広大な16MBのメモリ空間をリニアに利用でき、それまでのミニコンと同様なシステムを組むことが容易であったため、1982年にはサン・マイクロシステムズ社のSUN-1に、ワークステーションのメインプロセッサとして採用され、その後UNIXマシンのCPUとして一定の地位を築いていきました。広大なメモリエリアを扱えるMC68000は、高機能なグラフィックを扱うことができたため、1983年には先進的な機能と操作性を持つApple社のLisaに採用され、続く1984年にはMacintoshにも採用され、大きく販売数を伸ばしていきます。

当時の日本では、MC6809を搭載したFM-7が普及しており、次の富士通の新機種にMC68000が搭載されることををユーザは期待して待っていました。ところが84年に登場したFM-77は搭載CPUがMC 6809のままであり、MC68000搭載マシンの登場は87年まで待つことになります。しかも登場したマシンは富士通ではなくシャープ製のX68000でした。CMOSの日立製10MHzのHD68 HC000を搭載して、高機能なグラフィック機能を持ち、筐体も未来を予感させるツインタワー型でホビーユースマシンとしては理想的なものでした。しかし、87年当時は、高速なi80286が全盛であり、パソコンはビジネスでの利用も広がってきており、ホビーユースがメインのマシンでは、販売を伸ばすことは難しかったようです。しかし、その理想的なマシンへの憧れは現在でも続いており、X68000のユーザグループは活発に活動継続中です。

その後1988年にセガ社からMC68000を搭載したゲーム機のメガドライブが登場します。セガ社製のゲームも同時に発売され、ゲーム画面中をソニックが動くMC68000を最も多く販売したと思われるマシンになりました。

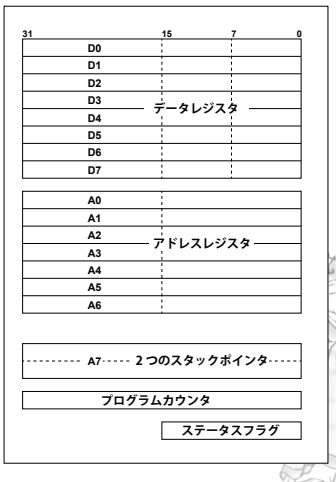
富士通はその後1989年に、32bitのホビーユースと教育用のパソコンを発売しますが、それは微かな期待も裏切ったi80386を搭載するFM TOWNSでした。

MC68000の発展と今

1984年に発表されたMC68020は、外部データバス、アドレスバスともに32bitとなり、本格的な32bit CPUになりました。

ワークステーション用のCPUとして、ソニーの社内ベンチャーとして開発・発売したUNIXワークステーションシリーズNEWSで採用されるなど利用がさらに広がっていきました。その後高速化したMC68030、MC68040、MC68060と進歩は続きましたがライバルとなるx86系CPUに絶えず遅れをとり、その差は広がる一方でありCISCの限界点も顕著になり、ワークステーションとMacintoshもRISCの時代になっていきます。しかし、その優れたアーキテクチャは、組み込み用のCPUとしてFreescale社(現NXP社)のDragon Ball、ColdFireとして使われ、現在でも生き続けています。SD

▼図1　MC68000のレジスタ構成



注3) どのレジスタでも同様に命令が使えること。

注4) 最短のレジスタ間の転送で4クロック。アドレッシングモードによりクロック数が加わり、ワード転送アドレスレジスタ間接モードでは4クロック追加。符号付き割り算命令DIVSでは、158クロック。

うまくいく チーム開発のツール戦略

第7回 もっとJIRAを活用しよう！ いろいろな業務で使ってみる

Author リックソフト㈱ 樋口 晃、大塚 和彦

Atlassian社のJIRAシリーズとは

今回は基本に戻り、Atlassian社が開発するJIRAシリーズを紹介します。JIRA製品の利用目的の約3割が技術系業務以外であることはご存じですか？　その一例を紹介しながら、JIRA製品の可能性を追求してみます。

JIRAはAtlassian社が開発するソフトウェアです。Atlassian社は、オープンソースプロジェクトを支援するために自社ソフトウェアの無償利用を認めているので、オープンソースソフトウェア業界の方はJIRAを利用することも多いのではないでしょうか。

オープンソースソフトウェアの開発では、世界中の技術者が時間や場所に関係なく機能拡張やバグ修正をしています。これらの作業はJIRAにより「誰が」「何を」「いつまでに」を明確にすることができるため、最近の「ワークスタイルの変革」や「新しい働き方」などにも活用する動きが活発になっています。

JIRAは2015年11月に利用形態ごとに「JIRA Core」「JIRA Software」「JIRA Service Desk」という3つのライセンスに分かれ、アドオンとして提供されていたアジャイル開発・スクラム開発に利用する機能がJIRA Softwareの標準機能となりました。旧JIRAの機能をそのまま継承しているのはJIRA Softwareとなります。

基本的な機能を持つJIRA Coreは、営業やマーケティングや総務部門、人事部門などの非技術系の業務プロセスを見える化することで生産性の向上が見込めます。

JIRA Softwareは、「かんばん」「スクラム」などの機能でアジャイル開発にも対応し、Bitbucket Server(Git)やSubversion(SVN)、CIツールであるBamboo、Jenkinsなどと連携させて開発プロセスの生産性の向上や構成管理／変更管理／バグトラッキングなどに活用できます。JIRA Service Deskは、ITIL準拠認定「PinkVERIFY」を取得したITサービスマネジメント(ITSM)ソフトウェアです。ほかのITSMソフトウェアと比べて特徴的なのは、問い合わせ画面のカスタマイズ、Confluenceと連携した強力なFAQシステムの構築、問い合わせ側のワークフローやグループ化など、問い合わせ側に対する機能が充実している点です。当然、SLAによるサービスレベルの管理機能やインシデントの管理機能などもあります。3つのライセンスの違いは**表1**のとおりです。

開発系以外にも使えるJIRAのプロセス管理機能

JIRAシリーズの用途はおもにソフトウェア開発業務系だと思われがちですが、そうではありません。2014年に開発元のAtlassian社が行った調査では、JIRAを利用している企業の約30%がソフトウェア開発業務以外で利用しているという結果が出ました。人事部門、マーケティング部門、財務部門、法務部門などさまざまな業務部門でJIRAが活用されています。ソフトウェア開発業務以外でJIRAを利用する事例は日本国内にもあります。その1つを紹介しましょう。

▼表1　各ライセンスの機能の違い

機能	JIRA Core	JIRA Software	JIRA Service Desk
基本ビジネスプロジェクト	○	○	○
ワークフロー編集	○	○	○
強力な課題検索	○	○	○
ダッシュボード	○	○	○
基本レポート	○	○	○
アジャイルボード		○	
バックログ計画		○	
アジャイルレポート		○	
開発ツール連携		○	
リリースハブ		○	
キュー			○
SLA			○
Confluence KB連携			○
詳細なサービス指標			○
問い合わせ画面のカスタマイズ			○

セミナーやイベント開催のタスク管理に

JIRAはプロジェクト管理ツールとして活用されることが多く、セミナーやイベントの企画から開催／報告までのプロセスを管理できます。ここで紹介する例では、JIRA SoftwareとWBSガントチャートfor JIRAを組み合わせ、セミナーの開催におけるWBS(Work Breakdown Structure)をテンプレート化して登録しています。

親タスクの「FY17セミナー・イベント」の位置でコンテキストメニューを開き、テンプレート内の「無料セミナー業務」を選択すると(**図1**)、テンプレートから自動的にタスクが生成されます。

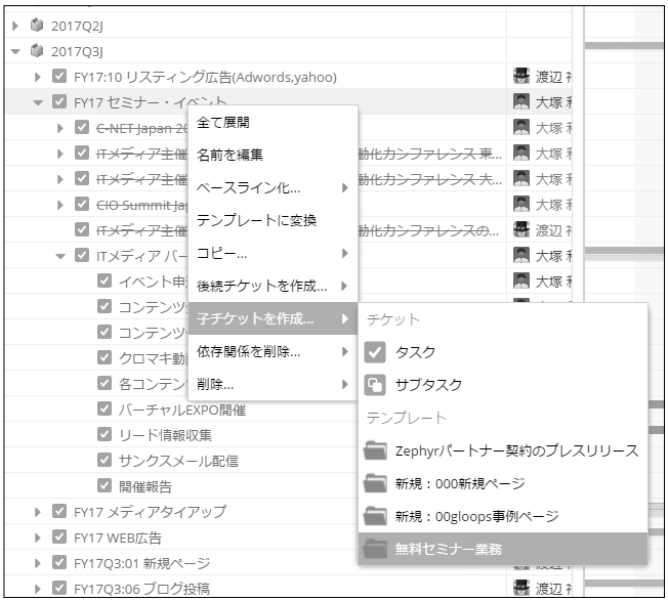
生成されたタスクの名称や期間を修正したり、テンプレートにないタスクを追加したりできます。**図2**のようにガントチャートとして全体スケジュールを作成し、計画時のスケジュールをベースライン化すると、タスクの遅延なども確認できます。

JIRA Softwareのバージョン機能を利用すれば、四半期ごとのタスク数と作業にかかった時間(工数)を集計できます。WBSガントチャートfor JIRAにある四半期表示も利用できますが、この例では四半期ごとのタスク数と工数を集計するためにバージョン機能を利用しています。この方法は、チラシやカタログの作成、ホームページの開設やサイトの更新など多くの業務に応用できます。また、このガントチャートはExcelに出力できるので、JIRA Softwareのアカウント以外でもスケジュールを共有できます。

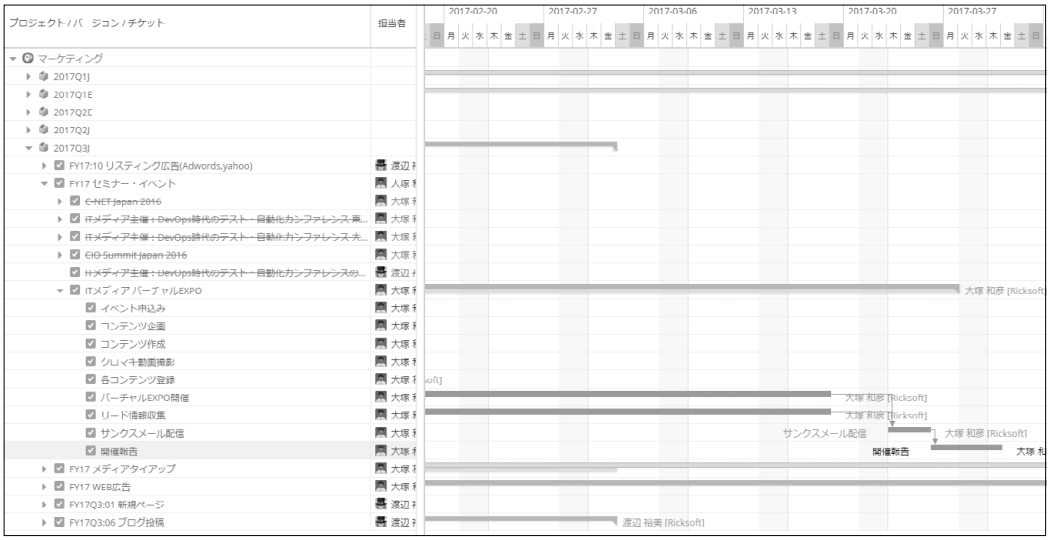
JIRAシリーズおよびAtlassian製品の可能性

Atlassian製品はJIRAシリーズだけではありません(**表2**)。アドオンと組み合わせることで多くの業務用途に対応し、生産性を向上させます。

▼図1　JIRAを利用したタスク管理画面



▼図2　JIRAを利用したガントチャート

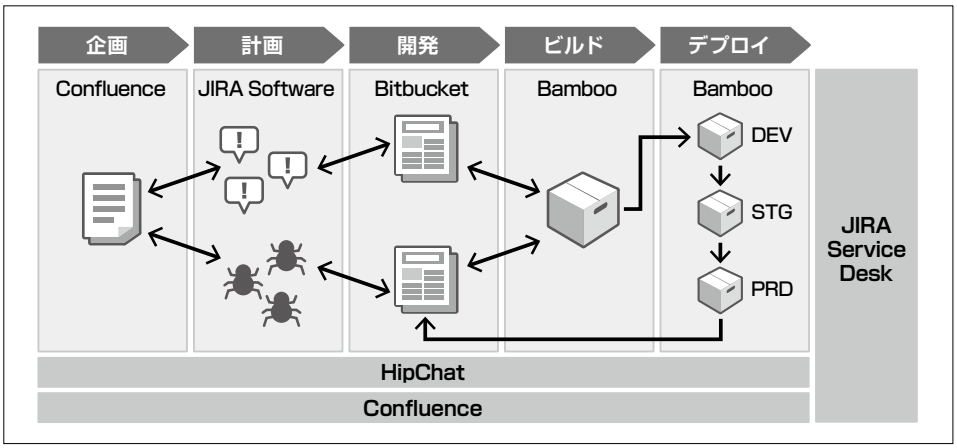


▼表2　Atlassian製品のラインナップ

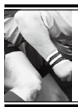
ソフトウェア名称	概要
Confluence	関係者やグループ全体で漏れなく情報を共有するためのツール。「情報を伝える」「情報を探す」という時間を減らし、生産性を向上させる。仕事の整理、ドキュメントの作成、議論などを1ヵ所で行える
Confluence Questions	Confluenceにアドオンするだけで「はてな」や「Yahoo! 知恵袋」のような人力検索やQ&Aツールとして利用できる
Confluence Team Calendars	チームカレンダーをJIRAのプロジェクトと連携し、Confluenceの関連ページに埋め込んでチームメンバーとスケジュールを共有できる
JIRA Core	マーケティングチーム、人事、経理、法務などでドキュメントの承認やレビューなどのビジネスプロジェクトを管理できる
JIRA Software	アジャイルチーム向けのプロジェクト管理ツール。ソフトウェア開発やサポート業務全般に利用できる。また工数／進捗／リリースの管理にも利用できる
JIRA Service Desk	利用者が問い合わせ（課題）を作成しやすい画面UIを作成できる。担当者もプロセスをJIRAできちんと管理でき、サポート業務を進めやすくなる
Bitbucket Server	Gitリポジトリを作成／管理する。JIRA Softwareとの連携で、ブランチ作成などのGit側の動作とJIRAの課題ステータスを連動でき、開発プロセスの生産性を向上させる
FishEye	Subversion（SVN）やCVSなどさまざまなリポジトリとJIRA Softwareを連携させる。コードのブラウズ、コミットのモニター、変更通知の受信などで開発プロセスの生産性を向上させる
Crucible	FishEyeで管理しているソースコードをレビュー。変更箇所を色分け表示し、コメントなどを登録できるので効率良くレビューできる
Bamboo	CIツール。ソースリポジトリからのチェックアウト／ビルド／デプロイを自動化できる。Bambooは、Bitbucket Serverのブランチの作成を検知して追従するので、Gitブランチを追加するたびにBambooのタスクを修正する必要はない
HipChat	Atlassian製品のツールチェーンを強化するために必要なチャットコミュニケーションツール。スマートデバイスやApple Watchなどのデバイスでも利用できる。JIRAの課題を作成でき、ステータス変更をリアルタイムで通知。Confluenceの情報更新を即時通知するなど情報を伝えるツールとして活用できる

開発業務系で使用すれば、企画から計画／開発、ビルド／デプロイまでの連携ができます。DevOpsに必要な要素も備えています（**図3**）。さらに、Salesforce、Slack、Alfresco、Jenkinsなど、Atlassian製品以外のソフトウェア連携も可能です。すでに導入／利用中のソフトウェアを変更することにはさまざまな調整や苦労が伴いますが、Atlassian製品は「連携のしやすさ」「移行のしやすさ」でその障壁を取り除きます。Atlassian社の基準に合格したサードベンダが

▼図3　開発業務プロセスの連携



開発するアドオンをマーケットプレイス[注1]から簡単に追加できるのも魅力です(**図4**)。日本国内からのリクエストをもとに開発されたアドオンも掲載されています(**表3**)。

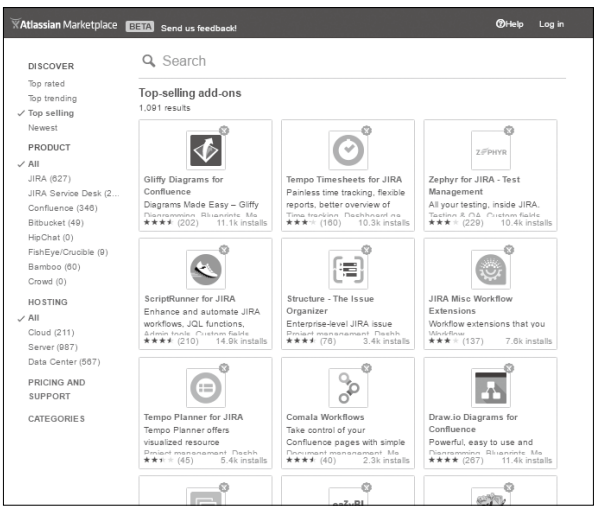


最後に

Atlassian製品の導入を検討する際には、Atlassian社のパートナーが発行する評価ライセンスを利用して評価／検証を行えます。またスモールスタート向けのスターターライセンスを利用すれば、大きな費用負担なく導入できます。ユーザ数が10,000以上の場合は無制限ユーザライセンスもあり、導入しやすいライセンス体系になっています。開発基盤の統合、開発プロセスの生産性向上、ワークスタイルの変革、テレワークの推進、情報共有の改善といったテーマで悩んだときは、Atlassian製品を検討してください。SD

※本連載の過去記事は技術評論社のWebサイト「gihyo.jp」でもご覧になれます。
http://gihyo.jp/ad/01/atlassian

▼図4　Atlassian製品のマーケットプレイス



注1）URL https://marketplace.atlassian.com/

▼表3　Atlassian製品のアドオン

ソフトウェア名称	概要
WBSガントチャート	JIRAシリーズにアドオンすると、ガントチャート表示で全体のスケジュールを把握できる。先行／後続の依存関係やクリティカルパスなど、プロジェクト管理に必要な機能を提供
Excel Like Issue Editor	JIRAシリーズの画面UIをExcelのようなスプレッドシートで表示し、そのままデータを変更できる。コピー& ペーストによる一括変更もできるのでJIRAの操作性が飛躍的に向上する
Alfresco connector	世界中で利用されているオープンソースのECMソフトウェアAlfrescoと連携するアドオン。Alfrescoを利用してJIRAとConfluenceのファイル共有や、JIRAの課題間／プロジェクト間のファイル共有ができる

Special Interview

3年で300%急成長の

本誌で好評連載中の「うまくいくチーム開発のツール戦略」でお馴染みのリックソフト㈱が業務拡大につき各種エンジニアを募集中です。世界40ヵ国54,000社が導入しているAtlassian製品はご存じの読者も多いことでしょう。リックソフトはAPAC（アジア太平洋地域）でナンバー1の販売実績を誇ります。急成長を遂げる同社の戦略や今後の展望、求める人物像について大貫浩代表（**写真**）にお話を伺いました。（編集部）

リックソフト㈱
代表取締役 社長執行役員
大貫 浩氏

独自ソフトの開発をきっかけに成長を加速

――リックソフトの成り立ちを教えていただけますか。

大貫氏　リックソフトを設立する前、個人事業主の立場で金融系のSEとして働いていました。2005年にリックソフトを立ち上げたあとも金融系のSES事業を展開していたのですが、その事業の中で出会ったメンバーと、ボランティアでオープンソースプロダクトの翻訳をしていたんです。そのときにAtlassian製品を使っていて、これはいいなと感じました。それをきっかけにAtlassianの製品を日本でも広めたいと考えるようになり、ビジネス化するために2009年にAtlassianのパートナーになりました。

――どういった点がAtlassian製品の魅力だったのでしょうか。

大貫氏　Atlassianの製品には、メールを書かなくてもメンションの機能を使ってメッセージを送ることができるメンション機能が用意されているなど、チームワーク力を向上させるための工夫が数多く盛り込まれています。また、Atlassian製品はもともと技術者向けでしたが、最近では非技術者の人たちでも使えるように使い勝手が改良されていることも大きなポイントだと感じています。

――リックソフトにおいて、ビジネスの成長につながった取り組みがあれば教えてください。

大貫氏　以前はAtlassian製品の販売とサポートが主だったのですが、その後自社でプロダクトの開発やサービスの提供に乗り出したことですね。具体的なプロダクトやサービスとしては、JIRAのプラグインとして動作し、Webベースでガントチャートが作成できる「WBSガントチャート for JIRA」（図1）や、クラウド上でAtlassian製品が使える「RickCloud」などが挙げられます。

これらをリリースしたことは、私たちの技術力のアピールにもつながりました。たとえばAtlassian製品を導入する際、大規模なお客様になると、数十ある外部システムとの連携や、不足している機能を新規で開発するといった作業も必要になります。その際、お客様としては当然技術力のあるベンダーに依頼したい。このとき、自社でプロダクトを開発していることが技術力の裏書きとなり、大規模導入を支援させていただく機会も増えています。

グローバル展開を目指して海外拠点を設立

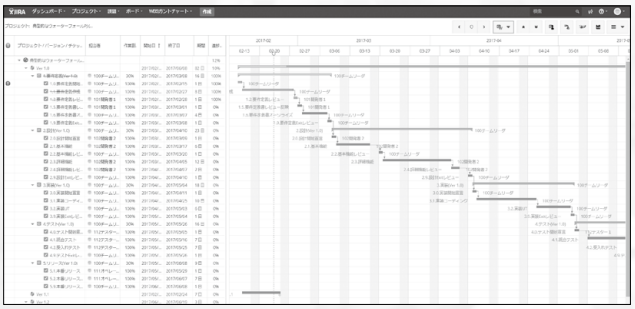
――お客様へ製品を導入する際、どういった点に注意されていますか。

大貫氏　製品だけでなく、それを活用するためのノウハウも一緒に提供しなければならないと考えています。たとえばご提案させていただく

ベンチャーがSE・PGを大募集!!

ときに、この製品を100%使い切るには行動や考え方を改めなければならないといったお話をさせていただくと、「そういうことを実現したいんだ」と共感していただけるお客様もいらっしゃるんですよね。ただ、一緒に考えてくれるパートナーがいなかったので、なかなかやり切れなかった。そこで私たちがアドバイザーとして支援することで、すんなりと導入に進むといったことは珍しくありません。

▼図1　WBSガントチャート for JIRA



――*先ほど技術力についてお話がありましたが、それだけでなく、導入後どうすれば活用できるかについて提案する、コンサルティングの領域にも踏み込まれているわけですね。*

大貫氏　そうですね。そこは他社と最も違う部分だと考えています。

――*2016年12月には米国法人としてRicksoft, Inc.を設立されましたが、どのような目的があるのでしょうか。*

大貫氏　自分たちでもソフトウェアの開発を始めたので、そのグローバル展開を目的としてセールスおよびマーケティングの拠点として位置付けています。またRicksoft, Inc.の拠点はサンフランシスコにあるのですが、ここは新しいテクノロジーの世界的な発信基地でもあるので、それらの最新情報を収集する役割も担っています。

日本全体の生産性向上に貢献したい

――*今回、積極的な人材採用に踏み切られたと伺いました。どのような理由があったのでしょうか。*

大貫氏　ものづくりの現場において、情報共有やコミュニケーションのためのツールの重要性が高まっていると感じています。お客様の製品開発に貢献できる、新しい考え方を採り入れた新しいツールです。そういったものの需要が高まり、それに伴って私たちのビジネスに対する期待も大きくなっていると感じています。そこで新たな人材を採用し、リックソフトとして日本全体の生産性の向上に貢献していきたいというのが大きなねらいです。

――*リックソフトが求める人材像について具体的に教えてください。*

大貫氏　基本的には技術が好きで、常に技術方面にアンテナを立てている人ですね。それで、仲間と技術の話をしていて楽しいと感じられる人。数多くの技術やプロダクトを知っている人でないと、実際の案件が生まれてお客様に提案するときに、「現状ではこれがベストです」と自信を持ってお勧めできないですよね。

それと、深く考えることが好きな人。技術をたくさん知っていても、うわべだけでは意味がない。製品や技術をお客様に提案するとき、「本当にそれで大丈夫なの？」ってチェックを入れるんですけれど、そのときに「これは裏側でこう動いていて、こうなるから大丈夫なんです」と言えるためには、やっぱり深く考えないといけないと思うんです。そのため、何か技術があったとき、これは裏側でどう動いているんだろうとか、あるいはお客様はこう言っていたけれど、従来のビジネスはこうだから、きっとこういうことも望まれているなと、自分の中でシミュレーションしつつ考えられる。そういった人にぜひ来てほしいと考えています。

――*本日はありがとうございました。*SD

募集職種：システムエンジニア／プログラマ ほか

詳しくは　リックソフト　求人　検索

Software

アドバンスソフトウェア、「ExcelCreator for Java」を発売

アドバンスソフトウェア㈱は2月22日、Java EE/SEのプログラムからExcelファイルを生成できるコンポーネント「ExcelCreator for Java」を発売した。Visual Basic .NET、Visual C#、ASP.NETに対応した同社の「ExcelCreator 2016」と同じ機能を実現している。

本コンポーネントを組み込むことで、Excelファイルの新規作成、既存のExcelファイルの上書きや別ファイルへの保存が短いコードで実装できるようになる。また目的のセルに対してデータの設定のほか、セルの書式、計算式、画像、オートシェイプなどの設定、シートのコピーや行、列の挿入／削除なども実現できる。実行環境にExcelをインストールする必要はなく、同社独自の技術でファイルの生成を行うため、たとえば100万セル（10,000行×100列）にランダムな数値データを書き込むのにおよそ4秒程度と、処理速度が高速であるという特徴がある。

価格は1クライアント開発ライセンス40,000円＋税で、本コンポーネントを組み込んで開発したアプリをサーバに配置して使用する場合は、別途サーバライセンス（120,000円＋税）が必要になる。対応環境はJDK7.0/8.0、Excel2007～2016。

CONTACT
アドバンスソフトウェア㈱ URL http://www.adv.co.jp

Software

DataRobot, Inc.、日本ビジネスを本格始動。データ資産を活かしてBIに機械学習の予測モデルを積極導入

米国DataRobot, Inc.は2月28日、2017年度より東京オフィスを新設し、日本市場において本格的にビジネスを展開していくことを発表した。

同社は2012年に米国マサチューセッツ州ボストンで設立され、機械学習による予測モデルを用いたBusiness Intelligence（BI）ツール「DataRobot Cloud」（クラウドベース）と「DataRobot Enterprise」（オンプレミス環境向け）を製品に持つ。日本国内も含め、スポーツ、金融、ヘルスケア、小売ほか、幅広い業界でさまざまなデータ分析に採用されている。日本オフィスは2017年度末までに社員を10名以上に増員する予定。

日本のカントリーマネージャーに就任した原沢滋氏は、DataRobotの役割として「人工知能の民主化」を掲げた。専門家に頼りがちな機械学習の技術を、ビジネスの現場にいる人たちが使えるようにすることこそが革新性を生み出す早道とし、分析したいデータがあれば、3～4ステップで予測モデルを自動作成、評価ができるDataRobotをアピール。同社が誇るデータサイエンティストたちの技術力とパートナー企業とともに、日本での導入を加速していきたいと抱負を語った。

CONTACT
DataRobot, Inc.
URL https://www.datarobot.com/jp

Topic

アカマイ・テクノロジーズ、2016年第4四半期「インターネットの現状／セキュリティレポート」を発表

アカマイ・テクノロジーズ合同会社は2月16日、2016年第4四半期の「インターネットの現状レポート」を発表した。

インターネットの現状レポートでは、世界中に存在するアカマイのエッジサーバから収集した攻撃トラフィック、またそこから分析した脅威の現状などがレポートされる。今回の報告のハイライトは次のとおり。

- 2015年第4四半期に比べ、100Gbpsを超える攻撃が140%増加
- 最大のDDoS攻撃（ピーク時は517Gbps）は、2年以上も前から存在している非IoTボットネットのSpike ボットネットからの攻撃
- 確認されたWebアプリケーション攻撃の95%を、SQLi、LFI、XSSのベクトルが占めた
- トラフィックが100Gbps以上である12の大規模攻撃のうち、7つにMiraiが直接関係
- 25のDDoS攻撃ベクトルのうち、上位3つはUDPフラグメンテーション（27%）、DNS（21%）、NTP（15%）で、攻撃数全体は第3四半期に比べ16%減少

CONTACT
アカマイ・テクノロジーズ合同会社
URL https://www.akamai.com/jp/ja

Report

「ITエンジニアに読んでほしい！技術書・ビジネス書大賞 2017」、大賞決定

2月16日、「ITエンジニアに読んでほしい！技術書・ビジネス書大賞」（㈱翔泳社主催）の授賞式が行われた。Web投票、受賞式での投票を経て選ばれた各賞は次のとおり。

●各賞の受賞書籍

技術書部門大賞	『ゼロから作るDeep Learning――Pythonで学ぶディープラーニングの理論と実装』（オライリー・ジャパン）
ビジネス書部門大賞	『最強の働き方』（東洋経済新聞社）
特別賞（㈱ジュンク堂書店 阿部初音さん選出）	『新装版 達人プログラマー職人から名匠への道』（オーム社）
特別賞（スマートニュース㈱ 藤村厚夫さん選出）	『ITエンジニアが覚えておきたい英語動詞30』（秀和システム）
特別賞（昨年度大賞受賞者 増井敏克さん選出）	『ゼロから作るDeep Learning――Pythonで学ぶディープラーニングの理論と実装』（オライリー・ジャパン）

▲ゲスト、各部門ベスト3の著者・担当編集者の方々

CONTACT
ITエンジニア本大賞 2017
URL https://www.shoeisha.co.jp/campaign/award/2017/

Service

インテル、さらなる成長に向けた戦略発表

インテル㈱は3月2日、同社の成長戦略についてプレスセミナーを開催した。

2020年までに平均的なネットユーザは1日あたり1.5GB、自動運転車にいたっては1日あたり4TBのデータを生み出すという。その分析のために、半導体生産も重要課題になる。そこで同社は、第一に「クラウド、AI（人工知能）、ネットワーク」、第二に「メモリ、FPGA、5G」、第三に「データリッチなモノと機器（IoT）」を重点領域とし、データすべてにかかわる「データカンパニー」への対応宣言を行った。AIに関しては、認識プラットフォームのSaffron社、コンピュータビジョンのMovidius社、AI分野のリーダー企業nervana社の買収を行い、この分野でもリーダーとなることをアピールした。

また小売業界に向けて、インテル・レシポンシブ・リテールネットワークプラットフォームを発表。これは小売店舗をIoT化しインターネットの世界とリアル店舗を結びつけるサービス。そして、2020年までにIoTデバイスの増加でワイヤレス通信の47%が人間を介在しないものになると予測。それに対してインテル5Gモデム（開発コード名Goldbridge）を2017年第2四半期にリリース予定と発表した。

CONTACT
インテル㈱ URL http://www.intel.co.jp

Event

4月9日、技術書限定同人誌即売会「技術書典2」開催

▲公式サイト内のポスター

4月9日、アキバ・スクエア（東京都千代田区）にて技術書限定の同人誌即売会、「技術書典2」が開催される（主催はTechBooster／達人出版会）。去年開催の「技術書典」からは会場の変更があり、参加サークル数は「58」から「189」へ大幅に増加した。一般参加は変わらず無料。

同人誌のラインナップは豊富で、ニキシー管、VR、Android、FPGA、量子コンピュータ、Crystal、mrubyなど、商業誌ではあまり見かけないテーマのものも多い。そのほか、翔泳社、オライリー・ジャパン、インプレス、アスキードワンゴといった出版社の出展もある。

また4月29、30日に幕張メッセ（千葉県千葉市）にて行われる「ニコニコ超会議2017」では、同イベントの出張版「超技術書典」が設けられる。こちらは、一般参加にニコニコ超会議の入場券が必要となる。

CONTACT
技術書典2 URL https://techbookfest.org/event/tbf02

BACKNUMBER バックナンバーのお知らせ

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）

2016年10月号

第1特集
Webサーバはなぜ動くのか？
HTTP、CGI、サーブレット、Node.js、Railsを一挙解説

第2特集
いますぐ始める本格派データベース
新しいPostgreSQLの教科書

一般記事
・CHIRIMENシングルボードコンピュータ入門
Webプログラミングで WoT サイネージ制作

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。

DIGITAL デジタル版のお知らせ

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%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。

ひみつのLinux通信

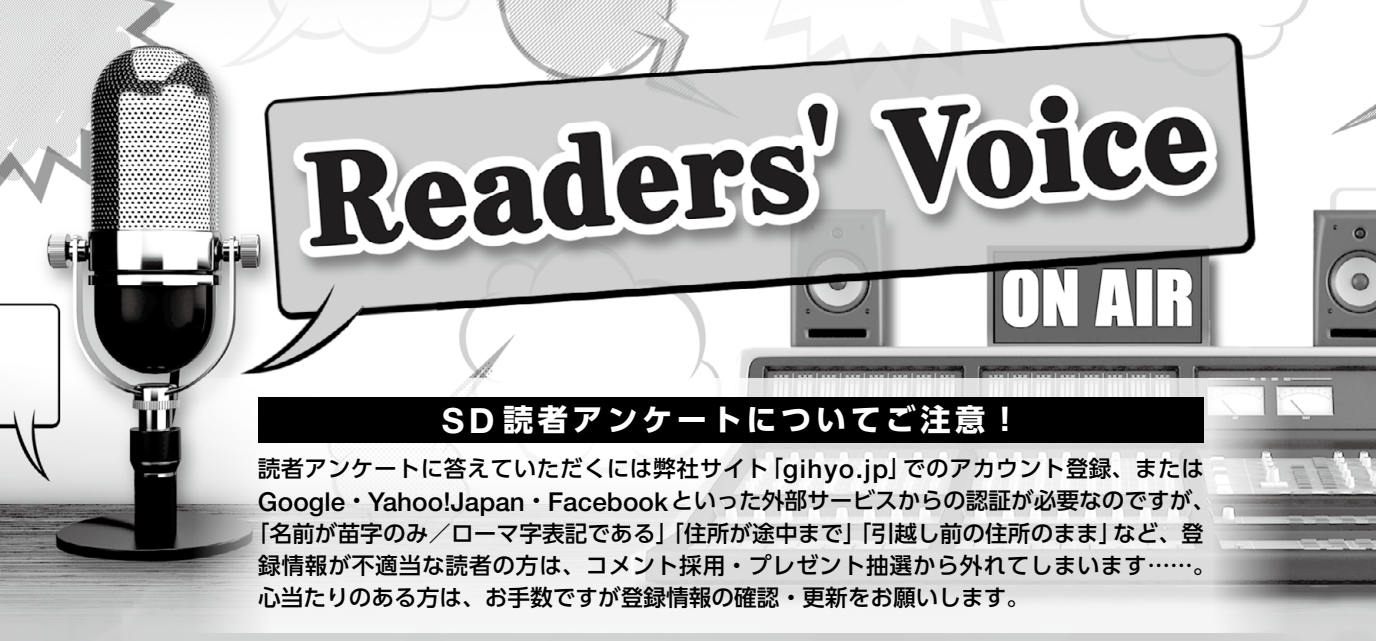
作)くつなりょうすけ
@ryosuke927

第38回 ねんどまつ

初めて閣下を知ったのは、Toi(トーイ)という上條淳士さんのマンガでした——とオッサンほいほいを上回る担当編集の私小説的つぶやきが読めるのは本誌だけ……。

to be Continued

そろそろ桜が咲いた、今年度は新人が何人来るとか、誰々が異動という話の時期ですね。一部では年度末進行真っただ中で大変な方もいらっしゃるでしょう。人は切羽詰まると特異な行動をとるとのこと。年度末のような高ストレス状態では自己破滅型行動をすると聞いたことがあります。定期テスト直前に部屋の掃除をしたくなるのと同じです。ヘンだと思ったら、とりあえず両手を伸ばして深呼吸です。さすれば解決策が見出されます。人にもやさしくできます。稼働始まってからのサポートが、とか考えるのダメです。それでも生きていかざるを得ないのですから！　つうか、こういう衝動って年度末に限らず起き……起きません！　出ませんにょ！(動揺してる)



SD読者アンケートについてご注意！

読者アンケートに答えていただくには弊社サイト「gihyo.jp」でのアカウント登録、またはGoogle・Yahoo!Japan・Facebookといった外部サービスからの認証が必要なのですが、「名前が苗字のみ／ローマ字表記である」「住所が途中まで」「引越し前の住所のまま」など、登録情報が不適当な読者の方は、コメント採用・プレゼント抽選から外れてしまいます……。心当たりのある方は、お手数ですが登録情報の確認・更新をお願いします。

2017年2月号について、たくさんの声が届きました。

第1特集 いまはじめるDocker

「Dockerを使うメリットをプログラマの方に、実際に試すことで理解してもらう」がテーマの特集。基本知識、コマンド、導入事例、Linux Containerの歴史、最新動向と、多方面からDockerを解説しました。

Dockerはいくつも書籍が出ていますが、開発速度も速いのでより新しい情報がほしいのと、新しい技術についてはいろいろな人の意見を聞きたいので、本書は即買いでした。
じゃがりこさん／福岡県

もうDockerは無視できない。
lipgtxさん／東京都

コンテナの歴史の説明がためになった。
勝又さん／埼玉県

会社で導入を検討しようとしていたところだったので、参考になりました。
吉良さん／東京都

ちょうどDockerに手を出そうとしていたので。　hiro_32さん／広島県

Dockerの最新動向を把握せず使用していましたが、swarm modeというのも導入されたんですね。Dockerは、機能の追加、変更が早いので、ぜひ今後も最新動向を記事にしてほしいです。
今井さん／千葉県

寄せられたコメントをみると、使う予定orすでに使っているという声が多数。本誌2015年12月号の特集でDockerを取り上げたころと比べると、研究フェーズから導入・実用フェーズに移っているという印象です。

第2特集 Linuxファイルシステムの教科書

Linuxのファイルシステム（Ext3、Ext4、XFS、F2FS、Btrfs）がどのようにデータを扱うのか、どのように記憶装置とユーザの操作を結び付けるのかを、ジャーナリングファイルシステムのしくみにも触れながら解説しました。

ファイルシステムの細やかな説明にも感動いたしました。Ext3のしくみを調査したことがなかったため、本誌を読んでジャーナリング機能の役割を理解できました。　Naohiro.Cさん／新潟県

低レイヤの情報についてのまとまった情報が、雑誌の特集で得られるのはありがたい。　森崎さん／京都府

ファイルシステムの説明は、なかなかまとまったものがないので、こういう特集はうれしいです。もっとも、自分がこれまで使ってきたのはExt2〜4とXFSくらいなのですが。今ではまた新しいファイルシステムも登場しているようですね。　KKさん／愛知県

知りたかったBtrfsに多くの誌面が割かれていて、勉強になりました。
山下さん／東京都

あとあと勉強しようと思っていた分野の話だったので、楽しく読めた。
山口さん／愛知県

ファイルシステムも重要だけど、マウント・アンマウントという概念がわかっているようでわかっていません。そこで躓いています。今度、教科書シリーズの一部としてわかりやすく説明してもらえるとうれしい。　kmさん／愛知県

日頃より当たり前のように使用しているファイルシステムについて、基本から応用まで一歩踏み込んだ学習ができ、新鮮な気持ちになりました。
オミオさん／宮城県

2月号のプレゼント当選者は、次の皆さまです

①高速無線LANルータ「WNPR2600G」
泊久信様(大阪府)

②ディレクターズ10周年記念バッグ(ベージュ)
中村夏実様(大阪府)

③『ご近所猫の写真集　ねこ舌』
岡本巧様(東京都)、gin様(愛知県)

④『データサイエンティストの秘密ノート』
penpen様(神奈川県)、psi様(東京都)

⑤『やさしく学べるMySQL運用・管理入門』
鈴木浩様(熊本県)、若杉龍郎様(石川県)

⑥『Pythonクローリング&スクレイピング』
バンジョーのジョー様(京都府)、かず様(千葉県)

⑦『はじめよう!プロセス設計』
犬棟梁様(埼玉県)、yasu様(広島県)

※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。

Btrfsについて解説されているのが良いです。　ItSANgoさん／大阪府

重要だけど、なかなか手が出ない分野の特集でした。ファイルシステムの開発に直接関わる読者の方は少ないかと思いますが、実装の理解という点でためになったという声が多く寄せられました。

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】

ITエンジニアの「採用」「評価」をテーマにした小説テイストの特別企画。後編では、会社で自分の能力が正当に評価されず、転職を考えているSI会社新卒4年目のエンジニアが主人公です。

採用担当になったとき、あまり考えていなかった……。
潜在力あると信じたいさん／奈良県

あらためてアウトプットは大切だなと感じました。　橿山さん／埼玉県

数年前に転職をしたのですが、今の会社がどうなのかを比較しながら読んでしまいました。　ほまれさん／千葉県

自ら動かないと何も起こらないのは重々承知していますが、外部との関わり合いはいつも難しく感じます。
最高のあいくさん／神奈川県

以前SIerに所属していたときには同じことを悩みました。今はSNSや勉強会なども活発に行われており、その気になりさえすれば世界は広がるので、若い人にはぜひ外に出てほしいと思います。　tack41さん／愛知県

エンジニアが評価されるようにボトムアップで改善を図ることは難しいと、いつも感じています。記事にあるような転職はなかなか決断しにくいところです。　ともさん／兵庫県

採用する側／される側の両面からエンジニアの働き方について迫った前後編。読者の方も採用する側／される側の立場から、それぞれ現場の声を寄せていただきました。

一般記事　ペネトレーションテストで学ぶ侵入攻撃の手法と対策

「ペネトレーションテスト」とは、システムに脆弱性がないかを、実際にそのシステムへ侵入を試みることで検証するテスト手法です。セキュリティカンファレンス「Black Hat」に参加した著者が、このテスト手法を実践する手順を紹介しました。

守備範囲外の話題でおもしろかった。
西村さん／東京都

毎年外部ベンダにペネトレーションテストを依頼しているため、今後の参考にしたいと思う。　山添さん／東京都

手口、知りたいような知りたくないような……。そんな感じです。
風のピエロさん／長野県

まさにテスト会社へ依頼しようとしているので、選定に役立ちそう。
野﨑さん／東京都

攻撃者の側に立って自分の作ったシステムを試験するということで、開発時には気づかなかった穴を見つけられそうですね。寄せられた声をみると、外部のセキュリティ企業に頼まれる方が多いようです。

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
May 2017

2017年5月号
定価（本体1,220円+税）
184ページ

4月18日発売

［第1特集］新卒歓迎企画【第2弾】試しながら理解がすすむ
Linux入門（ネットワーク編）
システム作りの基礎の基礎を固める

［第2特集］
サービス改善につながるドッグフーディング環境の作り方

［第3特集］
ブロックチェーン入門——いますぐわかりたいFintechの基本

［特別企画］
DDoS攻撃は防げないのか？／ニフティクラウドmobile backend／Lispで関数型を復習

※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。

SD Staff Room

●Suchmosの新アルバムがヘビーローテーション。どういう音かと一言でいえば「90年代の渋谷系といまどきな湘南の雰囲気」などと思っていたら、でも、それはオッサンほいほい系の音だと揶揄するブログが。そんな評価は、もうGood Night！ S.A.T.が急襲してくるよ！（本）

●PENTAX KPを購入。ISO819200の高感度撮影より、感度を上げることにより可能になる高速シャッターや絞り込みの恩恵が大きい。飲み会の料理撮影にも良いかな。久々のミラー機なので、EVFに慣れた老眼には光学ファインダーは手の震えがモロに。SONY RX10M3と一緒に持ち歩いて鳥撮りなどを楽しみたい。（幕）

●ほとんど使わない通話用ケータイをSIMフリースマホにしました。これまでPHSを使っていたのですが、とうとうY!mobileもPHSをおしまいにする気配。NTTパーソナル、DDIポケット、Willcom、そしてY!mobileと渡り歩いてきたPHSともこれでおさらばのようです。ありがとう！ PHS!!（キ）

●私が住む国立市のふるさと納税の特典には、市内の商店で扱っている手作りソーセージや洋菓子が多くあります。普段から地元の商店を応援したいと思っているので、このラインナップは好感が持てます。50万円コースの特典はドローンですが、市内にドローンの販売業者があるなんて初めて知りました。（よし）

●念願のクロスバイクをゲット！ 重量9.8kgとロードバイクほどは軽くないのですが、それでもママチャリや折りたたみと比べると格段に軽快な走り。さっそく友人と40kmほど走ったのですが、首・手首・腰・太ももに重大なエラーが……。体力と装備を整えて、次からは楽しむ余裕も持ちたいです。（な）

●（3月号からの続きで）壊れたガラケーは結局直らずじまいで、ガラホに買い換えました。待ち受け画面がシンプルすぎてちょっと不満ですが、音質もいいし、Wi-Fiが使えてバッテリーの持ちが良いのでこれで正解だったのかな、と思いつつ。2台持ち派としては、新しいSIMフリースマホも欲しいところです。（ま）

本誌に記載の商品名などは、一般に各メーカーの登録商標または商標です。 © 2017 技術評論社

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4月号

発行日
2017年3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。