Special Feature 1 OSS DBを極める手がかり
Special Feature 2 1Gbps超のLAN技術
[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

Software Design

2016年2月18日発行
毎月1回18日発行
通巻370号
（発刊304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

2016 February 2

Special Feature 1

2大OSS
データベースの
勘所を探れ!

MySQLとPostgreSQL

[最新]徹底比較

導入時の
「罠」を避ける
現場ノウハウ

Special Feature 2

インフラエンジニア
の新常識!

1Gbps超ネットワーク高速化時代の適切なLANケーブリングの教科書

Extra Feature

まだEclipseぐせが
残っていませんか?

Android Studioのスタイルで効率アップ

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）

2016 February
2
Software Design

第1特集

2大OSSデータベースの勘所を探れ!

MySQLとPostgreSQL [最新]徹底比較

導入時の「罠」を避ける現場ノウハウ

contents

2016 February
2
Software Design

contents

第2特集

インフラエンジニアの新常識!

1Gbps超ネットワーク高速化時代の適切なLANケーブリングの教科書

一般記事

短期連載

Catch up trend

アラカルト

Column

Development

OS/Network

［ロゴデザイン］
デザイン集合ゼブラ+坂井 哲也
［表紙デザイン］
藤井 耕志（Re:D）
［表紙写真］
Tom Walker /gettyimages
［イラスト］
フクモトミホ
［本文デザイン］
＊岩井 栄子
＊近藤 しのぶ
＊SeaGrape
＊安達 恵美子
＊轟木 亜紀子、阿保 裕美、佐藤 みどり（トップスタジオデザイン室）
＊伊勢 歩、横山 慎昌（BUCH+）
＊森井 一三
＊藤井 耕志（Re:D）
＊石田 昌治（マップス）

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。

技術評論社の 確定申告本

平成28年3月締切分

初めてでも大丈夫！ マネして書くだけ 確定申告 平成28年3月締切分

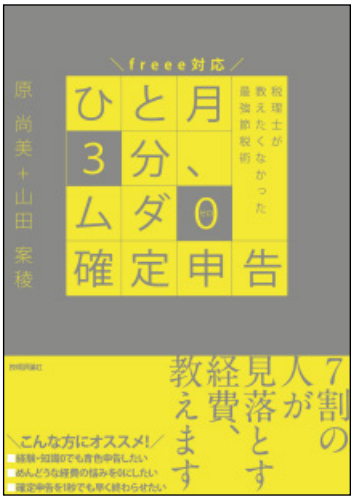
山本宏　監修／A4判／176ページ
定価（本体1,380円＋税）
ISBN978-4-7741-7684-0

家族が働いていたり副業があるために確定申告が必要なサラリーマンのために、確定申告に必要な書類の作成方法を、簡単にわかりやすくまとめました。ほかにもアルバイトやパート、フリーランス、不動産オーナー、年金生活者などが確定申告を行う場合についても、個別のケースごとに詳しく解説。本書を参考にすれば、該当するケースを探して、番号順にマネして書くだけで書類が記入できるので、忙しくて書類作成に時間をとれない方や、初めて確定申告を行う方などにオススメです！

フリーランス&個人事業主 確定申告でお金を残す！元国税調査官のウラ技 第2版

大村大次郎　著
A5判／224ページ
定価（本体1,580円＋税）
ISBN978-4-7741-7664-2

税金には、シロなのかクロなのかはっきり線引きされていないグレーゾーンがいくつもあります。その境界線を賢く活用することで、確定申告は絶好の節税の機会になります。フリーランス・個人事業主の皆さんが確定申告するときにぜひ知っておいてもらいたいことを元国税調査官の著者が厳選しました。はっきりシロと認められていることでも、納税者に有利になる（＝税金が安くなる）確定申告のやり方を税務署がすすんで教えてくれることはありません。自ら知識を仕入れて、確定申告でトクする人になりましょう！

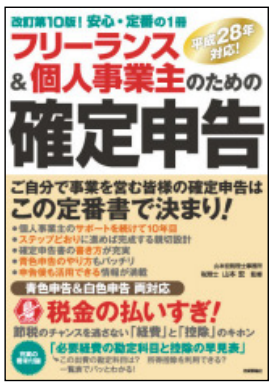


ひと月3分、ムダ0確定申告

税理士が教えたくなかった最強節税術

原尚美・山田案稜　著
A5判／272ページ
定価（本体1,580円＋税）
ISBN978-4-7741-7792-2

「勘定科目なんて知らなくていい」「仕訳なんてしなくてもいい」最高に簡単な確定申告の解説書です。経費で落とせる領収書と落とせない領収書といった悩みどころもしっかり解説。自動で帳簿付けしてくれる話題の全自動クラウド会計ソフト「freee」にも対応したかつてない確定申告本です！



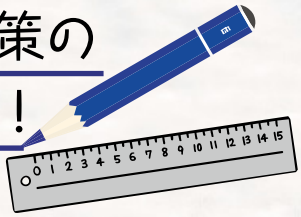
フリーランス&個人事業主のための確定申告 改訂第10版

山本宏　監修
A5判／240ページ
定価（本体1,480円＋税）
ISBN978-4-7741-7665-9

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

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！



金子則彦　著
A5判／536ページ
定価（本体3200円+税）
ISBN978-4-7741-7708-3

岡嶋裕史　著
A5判／424ページ
定価（本体1880円+税）
ISBN978-4-7741-7869-1

大滝みや子・岡嶋裕史　著
A5判／744ページ
定価（本体2980円+税）
ISBN978-4-7741-7821-9

加藤昭・矢野龍王 他　著
B5判／456ページ
定価（本体1880円+税）
ISBN978-4-7741-7823-3

大滝みや子　著
B6判／384ページ
定価（本体1480円+税）
ISBN978-4-7741-6710-7

栢木厚　著
A5判／480ページ
定価（本体1780円+税）
ISBN978-4-7741-7785-4

岡嶋裕史　著
A5判／672ページ
定価（本体2880円+税）
ISBN978-4-7741-7806-6

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）
ISBN978-4-7741-7807-3

岡嶋裕史　著
B6判／352ページ
定価（本体1680円+税）
ISBN978-4-7741-5521-0

きたみりゅうじ　著
A5判／656ページ
定価（本体1980円+税）
ISBN978-4-7741-7833-2

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

第86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。

Accelerated Mobile Pages プロジェクト

モバイル Web の高速化を目指すプロジェクト

Webの高速化に向けてさまざまな施策を繰り出し続けているGoogleが、2015年10月にモバイルWebに焦点を当てた新しいプロジェクトを立ち上げました。それが「Accelerated Mobile Pagesプロジェクト」（以下、AMPプロジェクト）です。AMPプロジェクトが対象とするのはモバイル端末におけるWebページの読み込み速度であり、おもに次のようなことを実現するのが狙いとなっています。

- ニュース記事をはじめとするWebページの表示を高速化する
- モバイル端末におけるWebコンテンツのロード時間を短縮する
- 広告のための適正なしくみを提供する

AMPプロジェクトの設立段階では、GoogleのほかにTwitterやPinterest、Adobe Systemsなどが参加しています。また、大手出版社が多数参加している点も同プロジェクトの特徴です。現在、多くの出版社がオンラインで広告付きのリッチなコンテンツを提供しており、ページの読み込み速度が読者の定着性と、それにともなう広告の売り上げに大きく影響します。AMPプロジェクトでは、そのような広告市場のニーズに応えることも視野に入れています。

広告についての具体的なしくみはまだ検討段階とのことですが、容認可能な広告のパラメータを定め、読み込みの優先度やタイミングを最適化することによって、快適なWeb体験を維持できるような方式を考えているそうです。

AMP による高速化への道

AMPプロジェクトでは、発足と同時にGitHub上に「AMP HTML」と呼ばれるフレームワークを公開しました。AMP HTMLには、モバイル向けWebサイトを高速化するためのさまざまな規定が設けられています。Webサイトの制作者はAMP HTMLの規定に従うことで、ページの読み込み速度を劇的に向上させることができます。ただし、この規定は従来のWebサイトの作り方からするとかなり厳しいものになっています。代表的な規定は次のようなものです。

- 指定されたフォーマットに従う
- head 要素内で AMP JS を読み込む
- AMP JS 以外の JavaScript は使用禁止
- CSS はインラインでの指定が推奨され、外部ファイルでの読み込みは禁止
- img/video/audio/iframe の代わりに、AMP 独自のカスタムエレメントの利用を推奨する
- link 要素は rel:canonical 以外は使用禁止
- object/from/input なども使用禁止
- 広告の表示やアクセス解析には専用のタグを利用する

AMP JSはWebページの高速表示のためのベストプラクティスを詰め込んだJavaScriptライブラリとのことですが、AMP HTMLではAMP JS以外のJavaScriptが禁止されています。したがって、jQueryやAngularJSといった人気のライブラリと併用することはできません。JavaScriptを介した動的なページ構成は、Webページの読み込み速度を保証するうえでは大きな障害となります。そのため、AMPプロジェクトは当初のターゲットを、ある程度静的なコンテンツに絞るという選択をしました。

HTMLの実装と並んでAMPのもう1つの核となるのがサーバ側のキャッシュ技術です。AMP HTMLの仕様に従って作られたWebサイトは、検索サイトのサーバ上にキャッシュされるようになります。このキャッシュにはテキストだけでなく、画像やJavaScriptファイルなどのコンテンツも含まれます。そして、ユーザがモバイル検索の結果一覧からキャッシュされたWebサイトに飛ぼうとすると、Webサイトがホストされているサーバにアクセスする代わりに、キャッシュされた内容が返されます。これによってWebページを読み込む際のオーバーヘッドを最小化して、高速な表示を実現できるというわけです。

Googleでは、2016年2月後半を目処に検索結果に対するAMPによるページ誘導を開始する予定だとしています。そのほか、TwitterやLINE、Viber、TangoなどのサービスもAMPへの対応を表明しています。それと並んで、出版社をはじめとするコンテンツ提供者によるAMP対応も進められています。AMPは、今後のモバイルWebにおいて大きな役割を担う存在になるかもしれません。SD

AMP Project
https://www.ampproject.org/

DIGITAL GADGET vol.206

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

神戸にてSIGGRAPH ASIA 2015開催
躍進するアジアのCG技術

CGの祭典「SIGGRAPH ASIA」

2015年11月2日から11月5日の4日間、コンピュータグラフィックスとインタラクティブ技術に関する世界最大の学会・展示会であるSIGGRAPHのアジア版、「SIGGRAPH ASIA 2015」が神戸で開催されました。

アジア各国で持ち回りで開催されているSIGGRAPH ASIAが日本で開催されるのは、2009年の横浜以来です。今年は世界49ヵ国から7,050人の参加者があり、規模こそ米国で開催されるSIGGRAPHの半分ほどでありながら、アジア各国からの参加者を集め、大盛況のカンファレンスとなりました。

今年のテーマは"[RE]volutionary"「革命的な」でした。REが括弧でくくられている点も含めて意味するところは、過去に研究されていたことや過去のノウハウを再度見直して、新しい革命(レボリューション)を起こそうという、昨今のVRブームを喚起させるものとなっています。

SIGGRAPH ASIA 2015 公式ページ
http://sa2015.siggraph.org/jp/

SIGGRAPH ASIA 2015 論文集一覧
(Ke-Sen Huang 氏がまとめている非公式なもの)
http://kesen.realtimerendering.com/siga2015Papers.htm

論文と先端技術展示

SIGGRAPHに寄せられる論文の最近の傾向は、CG技術が応用された動画像関連技術、3Dプリンタの活用技術など、理論だけでなく現実のクリエイティブな作業に役立つものが目立つようになってきています。また、研究のアプローチ方法も、機械学習によって適切なアルゴリズムを導き出したり、Amazon Merchant Turkにより、多数の人々による評価や判断を研究に活かしたりするようになりました。論文のカテゴリも3Dスキャン、ビデオ処理、ファブリケーション、静止画、デザイン補助など、多岐に広がってきています。

さらに最近の傾向としては、論文の内容をリファレンス実装したサンプルコードや、実際に動かすことのできるツールなどを同時に公開する例も増えてきています。ノートパソコン1台と、クラウドがあれば、さまざまなことが可能になった今の時代を反映しているとも言えます。

それらの採択論文の中から、興味深いものをいくつか紹介しましょう。

Level-Set-Based Partitioning and Packing Optimization of a Printable Model
http://www.miaojunyao.com/projects/

3Dプリンティングした物体を部品ごとに分割し、運送に最適なパッキングをする方法を提案したもの。複雑な形状のオブジェクトも最小限の体積になるように計算され、壊れにくい組み合わせに詰め込むことができる。はみ出すパーツがないよう、きちんと元の形に組み立てられるよう考えられている(図1)。

Interactive design of 3D-printable robotic creatures
http://www.disneyresearch.com/publication/interactive-design-of-3d-printable-robotic-creatures/

ディズニーリサーチによる研究で、生物的な多脚で歩くことのできるロボットを3Dプリンタで作るというもの。


SIGGRAPH ASIA 2015の会場となった神戸国際展示場


レジストレーションの様子

※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。

体のバランスを保ったまま、それぞれの足を動かして前後左右に動くことのできる足を自動設計する。設計された形状は、3Dプリントで出力可能なもので、かつ通常のサーボモータで動かせる。4本足に限らず、5本足、6本足なども自動設計可能。あらかじめコンピュータ画面上で、足の形状や動きの制限、特徴などを教え込むことで、馬や蜘蛛、トカゲなどといったさまざまな生き物の動きを模倣することができる(図2)。

Legolization: Optimizing LEGO Designs

http://www.cmlab.csie.ntu.edu.tw/~forestking/research/SIGA15-Legolization/

ありとあらゆる3D形状を、LEGOブロックで作られたかのような形状に変換できるアルゴリズム。単にブロック形状に分割するだけでなく、色や、実際のブロックを組み合わせたときの安定性も考慮されている。LEGOブロックそのものの接続強度や、重さによるゆがみや不安定さをうまく回避し、安定した形状として組み立てられる。元の形状をボクセル化し、ランダム順でマージしていき、その際の強度を評価していくというもの。従来、達人LEGOビルダーしか作れなかったような複雑な形状も、このツールの助けを借りれば簡単になるかもしれないが、ちょっとしたモデルでも、制作のための手順書(設計図)は数百ページにも及ぶものになるそう(図3)。

WrapIt: Computer-Assisted Crafting of Wire Wrapped Jewelry

http://www-sop.inria.fr/reves/Basilic/2015/ILB15/

ワイヤー素材をもとに、さまざまな形状のアクセサリーを線画から作ることを可能にするアルゴリズム。単なる形状としての設計だけでなく、そのワイヤーを実際に曲げて形状を整えるときに必要な枠も設計される。3Dプリンタで整形したその枠に従ってワイヤーを絡めていけば、目的の形状が完成するというもの。研究にはAdobeも協力しており、将来Adobe製のツールに、このような機能の搭載が期待される。

Interactive Surface Design with Interlocking Elements

http://www.disneyresearch.com/publication/interactive-surface-design/

あらかじめ決められた単純な形状で、かつ組み合わせ可能な1種類の部品から、目的のオブジェクトを生成する技術。折り曲げや接続が可能なパーツを想定しており、菱形や糸巻き型などが使われている。

Computational Design Of Metallophone Contact Sounds

http://people.seas.harvard.edu/~gaurav/papers/cdmcs_sa_2015/

自由な形をとりつつも、音階として成り立つ鉄琴を作るためのアルゴリズム。目的の音から逆解析して形状を導き出している。サンプルでは、さまざまな動物の形をし、正しい音階を持つ鉄琴が披露されている(図4)。

Revealing And Modifying Non-Local Variations In A Single Image

http://people.csail.mit.edu/talidekel/NonLocalVariations.html

静止画像から、ばらついている形状をピッタリそろった画像に整形する手法。デモではトウモロコシの粒をきれいに並べたり、逆にもっとガタガタの粒に変形したりしている。ほかにもレンガやシマウマの縞模様など、1個1個は微妙に形が違うものが並んでいる画像に応用が利く。

Real-Time Pixel Luminance Optimization For Dynamic Multi-Projection Mapping

http://people.mpi-inf.mpg.de/~mzollhoef/Papers/SGASIA2015_PM/page.html

石膏像のような物体へ、2台のプロジェクタで投影するプロジェクションマッピング。単なる映像だけでなく、表面の質感もさまざまな見映えで投影するしくみ。投影先の素材は石膏でも、リアルタイムに環境マッピングを反映させ、金属質感やジェルのような質感の表現もできる。白飛びやテカリといった、1台のプロジェクタでは難しい投影もこなす(図5)。

Autocomplete Hand-Drawn Animations

http://www.liyiwei.org/papers/workflow-siga15/

手描きのための、ありとあらゆる補間をしてくれるドローイングツール。描いているところから、その先に描くものを予想して、次の作業をうながしてくれ

図1 Level-Set-Based Partitioning and Packing Optimization of a Printable Model

図3 Legolization: Optimizing LEGO Designs

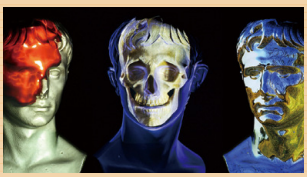


図5 Real-Time Pixel Luminance Optimization For Dynamic Multi-Projection Mapping

図2 Interactive design of 3D-printable robotic creatures

図4 Computational Design Of Metallophone Contact Sounds

図6 Autocomplete Hand-Drawn Animations

る。色や形状はもちろん、時間軸方向の補間アニメーションや、模様を描いたものの繰り返し補間、色の塗り方の補間などもしてくれる。検索用の単語や、カナ漢字変換の先読み補間のような感じで、すいすいと絵が描けるようになるかもしれない（図6）。

CG研究の行方

CG技術の進化の多くは、ハリウッド映画の特殊効果制作を行う現場のニーズによるものと、映像技術や画像処理技術を進化させるもの、それもより現実的な事象とのつながりが重要になってきています。SIGGRAPH ASIA最終日には、CG/VFX業界で古くから活躍するスコット・ロス氏の講演がありました。

今でこそツールや手法が確立してきていますが、CG/VFX制作の黎明期は、道具や手法を考えるところから始め、制作した映像が求めるクオリティに達しないかもしれないというリスクや、コストが見合わないなど、さまざまな苦労を乗り越えてきています。

初期のCG映像制作者は全員が全員プログラムを組める、数学者達、研究者達だったそうです。

今ではCG/VFX技術の進化により演出や映像制作の可能性が大きく広がり、映画監督やクリエーター達が想像するものを思ったとおりに描くことができ、最後の課題は、想像力と、予算だけになってきています。

CG表現が印象的な『ターミネーター2』や『アビス』、『ジュラシックパーク』なども、今見てみるととても簡単でチープで、今のノートパソコン1台で作れるくらいの映像です。

しかし当時は、マシンパワーと映像表現のせめぎ合いのギリギリのところを狙って作られた「初めて」の素晴らしい映像表現だったわけです。

来年夏のSIGGRAPH 2016は米国アナハイムで、次回SIGGRAPH ASIA 2016はマカオで開催される予定です。3D

Gadget 1

FOVE

http://www.getfove.com/jp/

視線追跡ヘッドマウントディスプレイ

FOVEは日本のハードウェアスタートアップによる、先進的なヘッドマウントディスプレイ（HMD）装置。HMDを装着した状態で視線方向を検知でき、今向いている方向をユーザインターフェースの操作に使ったり、これから進みたい方向を目で指示することができます。また通常のHMDとは異なる工夫として、視線方向のみ映像が細かくなり、視点が合っていない周辺部分は解像度を落として表示することで焦点を合わせるような映像作りをしたり、映像生成側のコンピュータの負荷を下げることができます。

Gadget 2

Touchy

http://touchtouchy.com/

触れないと見えないヘッドマウントディスプレイ

アート作品として展示されたTouchyは、誰かに身体を触ってもらわないと前を見ることができないという、究極のコミュニケーションを表現したアート的デジタルデバイス。手をつないでもらえない状態だと、真っ暗で何も見えません。誰かに手を握ってもらったりすると目の前のシャッターが開き、ヘッドマウントディスプレイ（HMD）の映像を見ることができます。ごつくて重たいヘッドマウントディスプレイの後頭部にはディスプレイが搭載されており、本人が見ている映像を、周りにいる人達も共有することができます。

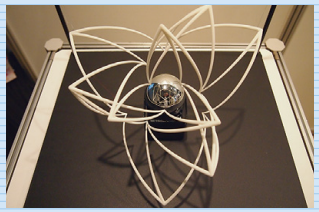
Gadget 3

Three Dimensional Anamorphoses

http://sa2015.siggraph.org/jp/attendees/art-gallery.html

球体への映り込みを利用した3Dプリンタオブジェ

球体状のミラーに映り込んだ映像が、ある地点から覗き込んだときだけ、星形などといった、さまざまな多角形として見えるオブジェ。映り込む外形は、数学的に逆算された形状で、3Dプリンタ出力で制作されている。もともと「Anamorphosis」は、絵画の中央に円筒形の鏡を置き、覗き込んだときに正しい絵が浮かび上がるという古くからある手法で、この作品は二次元のAnamorphosis絵画の世界を、三次元で表現したものです。



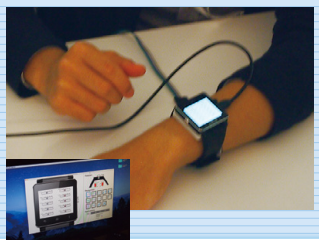
Gadget 4

SkinWatch

http://masaogata.com/

皮膚を使ったガジェット操作

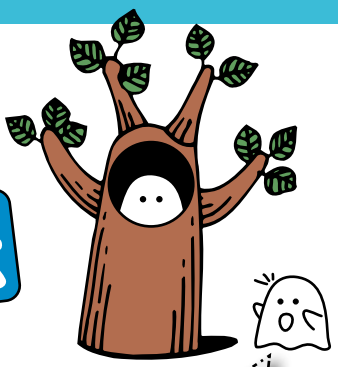
SkinWatchは、スマートウォッチを装着している腕の皮膚を使って、スマートウォッチの細かな操作をするというもの。一般的なスマートウォッチには各種センサー、竜頭タイプのダイヤルやボタン、小さな画面と小さなタッチパネルが搭載されていますが、なにぶん小さいため細かな操作はできませんし、操作中に画面の一部が隠れてしまうこともあります。SkinSketchではスマートウォッチ周辺の皮膚をひっぱったり動かしたりすることで、操作を可能にします。たとえば皮膚の上でピンチイン、ピンチアウトのジェスチャーをして操作できます。



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

33

結城浩の再発見の発想法



Data Compression

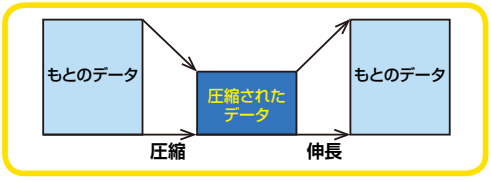
Data Compression ——データ圧縮

データ圧縮とは

データ圧縮(Data Compression)とは、**データの大きさを小さく変換する処理**の一種です。データ圧縮の技術は、あらゆる場面に使われています。ソフトウェアを配布・インストールするときのファイルは必ずデータ圧縮されていますし、私たちがWebページを見るときに行われているHTTPの通信も、多くがgzipでデータ圧縮されています。画像ファイル形式のPNGやJPEGも、音楽データのMP3も、またビデオを試聴するときに使われる各種コーデックにも、データ圧縮の技術が使われています。

データ圧縮で小さくなったデータを、伸長して完全に復元できる場合を「可逆圧縮」と呼びます(**図1**)。それに対して、データ圧縮時に情報が失われるため、もとのデータを完全には復元できない場合を「不可逆圧縮」と呼びます。ファイルアーカイバでは可逆圧縮を行います。画像データや音楽データなどの場合には不可逆圧縮を行うこともあります。

▼図1　データ圧縮と伸長



私が最初にデータ圧縮の話を聞いたのは、ファイルアーカイバのPKARCやLHARCを知ったときです。そのとき、どうしてファイルサイズを小さくするなんてことが可能なのだろうと、非常に驚いたことを記憶しています。ファイルサイズはかっちりと定まっていて、変えることなどできないという錯覚に捕らわれていたのでしょう。

データ圧縮が可能になるのは、多くのデータには**冗長性**があるからです。冗長性というのは、統計的な偏りと考えてもいいし、何らかのパターンと考えてもいいでしょう。たとえば、データの中に「A」という文字が100個並んでいる部分があったとします。そのままでは100バイトの領域を使ってしまいますが、「A100」のような形に変換しておけば、ずっと少ない領域で元のデータを復元するための情報を保存することができます。このように「同じデータの繰り返し」を「そのデータと長さ」の形で表す方法を、**ランレングス符号**と呼びます。ランレングス符号は、データ圧縮で使われるもっとも単純な符号化の1つです。

多くのデータ圧縮アルゴリズムでは、もっと手の込んだ方法でデータの中に含まれているパターンを見つけ出し、より登場頻度の高いパターンをより短いビット列に置き換えることで、圧縮効率を高める工夫がなされています。

データ圧縮を行うと記憶容量の節約になり、また通信量を削減できるというメリットがあります。その一方で、圧縮と伸長を行うための時間がかかってしまうというデメリットもありま

す。通信速度がとても低い場合、通信量の削減効果が高いため、圧縮と伸長に時間がかかってもトータルでの処理時間を短くできます。しかし、通信速度に比して圧縮と伸長を行うCPUが非力なときには、データ圧縮による処理時間の短縮効果は低い場合もあります。ですから、どのようなデータ圧縮をいつ行うかについては、圧縮・通信・伸長にかかる時間の間にトレードオフが存在することになります(**図2**)。

そもそも、データ圧縮が不可能な場合もあります。データ圧縮では冗長性を利用するため、冗長性が低いデータはほとんど圧縮ができないか、むしろ逆にデータが大きくなってしまう場合があるのです。たとえば、データ圧縮後のデータはすでに冗長性が低くなっているので、再圧縮はできません。またランダムなデータは、どんなデータ圧縮アルゴリズムでもサイズを小さくできません。あるいはまた、暗号化されたデータも圧縮することはできません。ですから、暗号ソフトウェアは必ずデータ圧縮後に暗号化を行います。

データ圧縮という処理は、データが持っている冗長性を減らすことになりますので、データのわずかなビットにエラーが起きただけでも、データ全体に悪影響を起こす場合があります。そのため、データ圧縮を用いるファイルアーカイバではCRC符号などを別途付加し、エラー検出ができるようにしてあります。

日常生活におけるデータ圧縮

私たちの日常生活でも、データ圧縮はたくさん見られます。たとえば**指示語**。私たちは、会話の中で直前に出てきたものに言及するとき、「それは中止しよう」や「こっちよりもあっちのほうがいいな」などと、指示語を使います。それによって、会話のスピードを上げ、労力を削減しているのです。

また、会話の相手とコンテキストを共有できている場合には、情報を省くこともよく行われます。学生同士が「レポート出した？」という一言で、どの授業の何のレポートであるか、ちゃんと伝わる場面はよくあります。

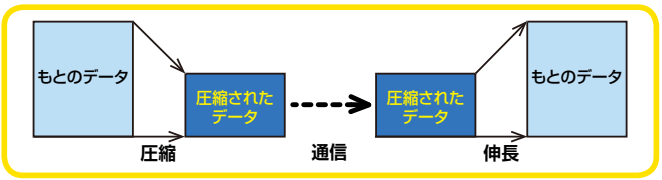
浮き輪を片付けるときに空気を抜くのは、物理的な圧縮ですね。空気を抜くのに手間はかかりますが、空間の無駄がなくなります。その代わり、使うときには空気を入れるための時間がかかります。データ圧縮で、圧縮と伸長に時間がかかるのと同じですね。

何年か前から**ふとん圧縮袋**もよく話を聞きます。ふとんをビニール袋に入れ、掃除機の吸引力を使って、空気を抜き、押し入れのスペースを有効活用する便利グッズです。ほんとうに可逆圧縮できるか不安なので、我が家では使っていませんが……。

考えてみると**名前**というものも、データ圧縮に似ています。ややこしい概念をいちいち文章として表現するのではなく、「名前」という簡潔な数文字で表現するだけで、どれだけ多くの時間が節約されているでしょう。たとえば、現代の私たちはあたりまえのように「メールで送って」や「ネットに公開した」や「クラウドを使えばいい」という言い回しを使います。もしも、「メール」や「ネット」や「クラウド」という名前がなかったら、言いたいことを伝えるのにたいへんな労力がかかるに違いありません。プログラミングの世界ではよく「名前重要」と言われますが、概念に明確な名前を付けることは、重要な知的作業なのです。

あなたの周りに「冗長性があるために無駄な空間や時間を使っているもの」はないでしょうか。データ圧縮のように冗長性を減らすことで、無駄を減らすことはできるでしょうか。ぜひ、考えてみてください。SD

▼図2　圧縮・通信・伸長



増井ラボノート

コロンブス日和

第4回　Gyazo

エンジニアというものは「楽をするためならどんな苦労も厭わない[注1]」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author　増井 俊之（ますい としゆき）　慶應義塾大学

Gyazo

今回は「Gyazo」という画像キャプチャ／アップロードサービスを紹介します（**図1**）。

Gyazoは2015年末の時点で、月間ユニークユーザ1,000万人、1日の画像アップロード数70万件という大規模なWebサービスですが、もともとは2007年ごろに私がコロンブスの卵的な発想で開発した小さなツールが大きく育ったものです。初期のGyazoは、パソコンのデスクトップを領域選択・キャプチャして、アップロードする単純なサービスでしたが、最近は動画キャプチャ機能／画像編集機能／コメント機能／検索機能／ブックマーク機能／チーム機能などが強化され、さまざまな用途に利用できる便利なツールに進化しています。

▼**図1**　Gyazo.comトップページ

Gyazoはもちろん「画像」をもじった名前です。先月までの3回にわたり、「GyaTV」、「Gyump」、「Gyamm」など、「Gy」で始まるシステムを紹介してきましたが、これらはGyazoにちなんだものです。「京都」を「キヨト」と発音する欧米人が多いように、欧米人は「gya」や「kyo」などの発音が苦手らしいので「Gy」で始まるドメインは取得しやすいようです。

Gyazoの基本的な使い方

Gyazoはパソコンのデスクトップ画面の一部をキャプチャして、Webにアップロードするツールですが、従来はこのために次のような3ステップの操作が必要でした。

- アプリを起動してスクリーンキャプチャを画像ファイル(A)としてセーブする
- 画像編集ソフトで(A)を開き、必要部分を切り出して別のファイル(B)にセーブする
- (B)をWebにアップロードする

一方、Gyazoを利用すると次のような1ステップで済むようになります。このような操作を「**Gyazoる**」と呼んでいます。

Gyazoを起動して画面の一部を選択すると、選択部分が自動的にWebにアップロードされて画像URLが割り当てられる

Gyazoった画像のURLはコピーバッファに保存されるので、Gyazoったあとですぐにチャット画面やメールテキストなどにURLを貼り付

注1）http://thinkit.co.jp/free/article/0709/19/

けることができます。

Gyazoにアップロードした画像はWeb上に残るので、あとで資料などとして利用できます。アップロードされた画像のURLは、ハッシュ関数で生成されるので、Gyazoったユーザがこれを公開しない限り、他人から画像が見られることはありません。

Gyazoの拡張機能

Gyazoはもともと私が個人的に使うために開発したもので、前述のようなソースコードをGitHubで公開[注2]していますが、ビジネス化にともなってさまざまな機能を追加したものをGyazo.comで運用しています(いくつかの機能はGyazo Pro[課金版Gyazo]ユーザ限定です)。

編集機能

Gyazoった画像をブラウザ上で編集して文字や図形を追加できます(**図2**)。

コメント機能

アップロードされた画像にブラウザ上でコメントを付けることができます。「EpsonのHMDを試してみてるところ。」というコメントを入力しています(**図3**)。

検索機能

入力したコメントを使って画像を検索できます。前述の画像や編集後の画像が検索結果に現れています(**図4**)。

ブックマーク機能

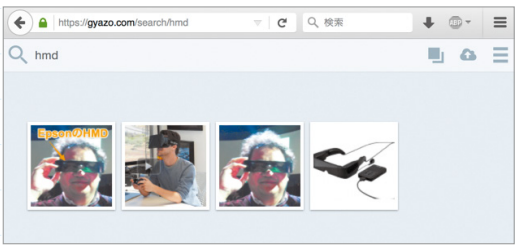
ブラウザで表示されているページをGyazoったときは、Webページのタイトルや URLがコメントに記録されます。また、アプリケーション画面をGyazoったときは「Microsoft Word」のようなアプリケーション名が記録されます。たとえば本誌のWebページをGyazoると、**図5**のようにページタイトル／ページURL／アプリケーション名(Firefox)が、コメントとして記録されます。

このように、Gyazoを画像付きブックマーク

▼ **図2** 画像を編集して矢印とテキストを追加

▼ **図3** コメントの追加

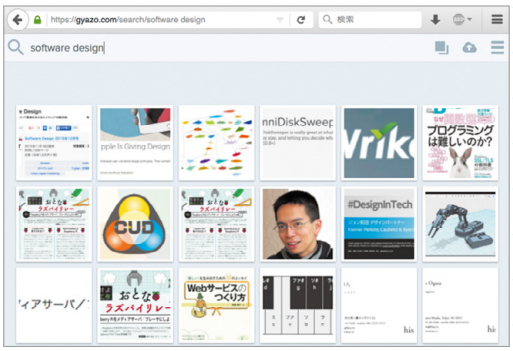
▼ **図4** 「hmd」で検索を行ったところ



▼ **図5** 本誌のページをGyazoった結果

注2) https://github.com/gyazo/Gyazo

▼図6　「Software Design」で検索を行ったところ



として利用できますし、Webページのタイトルや URL をもとに画像を検索することもできます。図6の例では、前回 Gyazoった画像に加え、「Software」「Design」というコメントがついた画像がリストされています。

関連画像検索

Gyazo.comでは、コメントやアップロード日付を利用して、画像の類似検索を行う「IvySearch」という検索機能(図7)を提供しています。

IvySearch とは Gyazoった画像に関連付けて記録される、日付やコメントなどのメタデータを利用して、関連する画像を芋蔓(いもづる)的に検索できるようにしたものです。

図8は「3D プリンタ」というキーワードで検索された画像の1つを選択したものですが、関連画像として別の3D プリンタや Maker Faire の画像などが自動的にリストされています。これらの関連画像はコメントなどから自動的に計算されます。

ここで Maker Faire の画像をクリックすると、選択された画像が表示され、その下にまた関連画像が表示されます(図9)。今回の画像には「Maker Faire」というコメントが記述されているため、Maker Faire に関連する別の画像がリストされています。

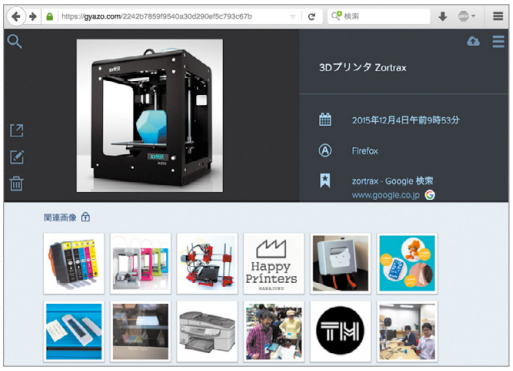
▼図7　IvySearch

このようにして関連画像をたどることにより、昔 Gyazoった画像を効果的に検索していくことができます。思いがけない画像が関連画像として表示されることも多く、古い画像を探索的にブラウズできます。

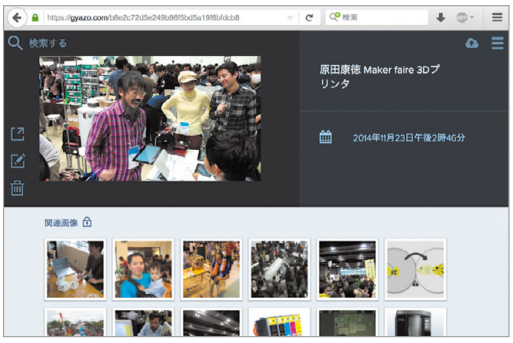
Gyazo な一日

このように、Gyazo を使うと Gyazoったときのアプリケーションや URL を保存して検索できるのに加え、IvySearch を使って内容が近い画像を簡単に検索できるので、広い応用ができます。私は Gyazo の作者ですが信者でもあるので、次のように毎日あらゆる仕事や趣味に Gyazo を活用しています。

▼図8　関連画像検索機能



▼図9　関連画像をたどることができる



・重要メールが来てたのでGmailページをGyazoってコメントに「TODO」と書いておく（後から「TODO」で検索できるようになる）
・新着Facebook記事で見つけた面白いWebページをGyazoって「読み物」というコメントを書いておく
・勉強したい案件のメモをWiki上で作成し、Gyazoった画像を貼り付けておく
・WebページをGyazoってプレゼン資料を作る
・新しいMacが届いたので、昔Gyazoっておいたメール設定データをもとにしてメールソフトを設定
・購入要望のMacの仕様がGyazoで送られてきたのでチェックして注文
・ちょっとしたアイデアを思いついたのでメモ帳に書いてGyazoって「アイデア」と書いておく
・昼の休憩時間に「読み物」とコメントしてあった記事を読む
・ついでにIvySearchで見つかった他の読み物も読む
・Facebookに自分の写真がアップされているのを見つけたのでGyazoってコメントを付けておく
・新しいラーメン屋の評判をGyazoってWikiに貼っておく
・明日の出張のために付近のレストランをチェックしてGyazoっておく
・念のため電車の時刻をGyazoっておく
・Skypeパスワードを忘れたので「Skype パスワード」でGyazo検索したあと、ブックマーク登録されていたEpisoPassページ[注3]を使ってパスワードを思い出す
・請求書作成が必要だったので古い請求書をIvySearchして修正して印刷
・TODO案件を検索してチェックした後、帰宅
・免許証のコピーが必要なサービスに対してデジカメ写真をGyazoってメールで送付
・Peatix[注4]のチケットとして使われるQRコードをGyazoっておく
・美味しかったワインのラベルをGyazoってコメントを書いておく

今回の連載で使っている画像はすべてGyazoでキャプチャしたものですし、資料の整理にもGyazoを活用しています。

Gyazoの歴史と発展

Gyazoは、GyaTVやGyamm、Gyumpなどと同様に、私がおもに自分で使うために作成したサービスなのですが、公開したところ思いのほか人気が出ました。初期のころはGyazo.comを自前のサーバで運用していたのですが、アップロードされる画像の量が増えてきたので、クラウドサービスを利用するようにしたところ、サーバ代が家計を圧迫するようになってしまいました。そういう窮状をみかねてNota Inc.[注5]の洛西一周氏がサービスを引き継いでくれることになったのですが、そのあともユーザ数やアップロード画像の量は指数関数的に伸び続け、サーバ経費もたいへんなことになってきたので、2012年ごろから真剣にビジネス化の検討を始め、機能を強化したり課金モデルを作ったりしてきました。現在は私もNota Inc.の一員となり、Gyazoをグローバルなビジネスとしてさらに発展させるべく、ベンチャーキャピタルから出資を受けて開発を継続しています。

このようなGyazoの発展は、コロンブスの卵的なシステムであっても、大きなビジネスに化ける可能性があることの証明だと考えています。これまで紹介した各種の小さなシステムやこれから紹介するシステムも、ビジネスに発展させられるものがあれば良いなと思っています。SD

注3）http://episopass.com/masui/Skype123456
注4）http://peatix.com/
注5）http://www.notainc.com/ja

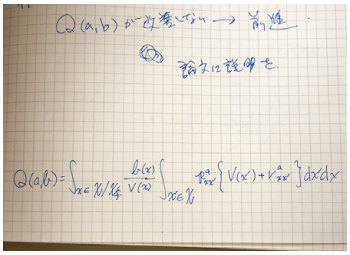
かまぷの部屋 最終回

第19献 ゲスト：上田 隆一さん

㈲ユニバーサル・シェル・プログラミング研究所
鎌田 広子（かまた ひろこ）
Twitter：@kamapu

上田 隆一（うえだりゅういち）さん

富山県出身。千葉工業大学工学部 未来ロボティクス学科准教授。博士（工学）。2004年から2008年まで助手・助教として東京大学大学院に勤務。2009年にUSP研究所に入社。2013年に産業技術大学院大学に助教として復帰。2015年より現職。現在は執筆、研究、教育に従事。「USP友の会」会長。Twitter：@ryuichiueda
URL：https://blog.ueda.asia



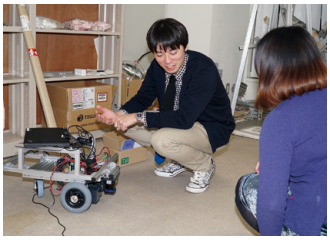
▲写真1

（鎌田）最終回のゲストは上田さんです。本誌上でも、「開眼シェルスクリプト」という連載を2013年の12月号までお持ちでした。上田さんというと、シェル芸のイメージが強いのですが、自己紹介をお願いできますか。

（上田）経歴がめちゃくちゃで、自分でも何をやっている人なのかよくわからないのですが、現在は、千葉工業大学のロボットが専門の学科で教員をやっています。本誌連載は紹介していただいたとおりですが、「シェル芸勉強会」というシェル上でコマンドを組み合わせる、いわゆるワンライナーの勉強会を2ヵ月に1回開催しています。

昔からコンピュータに興味があったのですか？　子供時代はどんな感じだったんですか？

1歳で数字を書き、小学生のときに客先で障子の格子を数えながら無限大について質問して気味悪がられていました。それだけなら神童ですが、落ち着きがなく先生の話も聞かないし、気にくわないことがあると暴れたりする問題児でした。そのうち、知能テストで異常な点数をとり、連絡帳に「あなたは天才なのに、我がままです。社会のために頑張ってください。」というようなことを書かれ、自分だけなんで、と暗い気分で過ごしました。



現在の上田さんからは、想像もできませんね。

救われたのは大学の研究室でした。すぐ成果が出て、英語で予稿を書く必要が出たのですが、英語は受験のときからろくに勉強せず浪人の原因になったくらいで……。6ページ分の字を埋めて大ボスの新井民夫先生（現、芝浦工大教授、東大名誉教授）に持って行ったら「ナンダコレハ！」と叱られまして。それだけなら気も楽だったのですが、土日に缶詰で強烈な個人指導を受けました。ボロクソ言われて泣かされたのですが、先生が単なる一学生にぶつかって来た効果は絶大でした。英語は正直まだ苦手ですが、面倒なことに正面から取り組むように意識が変わりました。また、自分も教育に携わりたいと、考えるようになりました。

大学ではロボットの研究をされているようですが、具体的にはどのような研究ですか？

自律ロボットです。生物は生きていますから、動き続けていますよね。ロボットのプログラムでそれを書くとわかるのですが、これってとても難しいことなのです。どんなふうに書いても、壁の前とか、あるいは人間から見ると何の変哲もないところで、同じところをぐるぐる巡ってしまったり、止まってしまったりするのです。

先に何があるのかわからないところで、動作を考えるのは難しいですね。

そうなんです。私が研究しているのは、情報がないところでロボットがどう動くべきかを決めるアルゴリズムです。たとえば人間は、目をつぶったまま机の上に置いてあるカップとか持てますよね？　真っ暗な場所でも、手探りで移動できますよね？　これは何でなのかと。それを数式で表現することを試みています。たとえば**写真1**のような式です。これは2013年の冬に思いついて今年、投稿論文で公表した式です。この式でロボットが暗闇で出口を探す動き



をします。

式を見てもわけがわかりません……。難しいですね。

別のテーマとして、学習アルゴリズムにも挑戦しています。移動については、脳の海馬のまわりで処理されていることがわかっているのですが、ここは記憶も取り扱っています。記憶と移動は関係があるんですね。だから記憶からどう移動すべきかが直接的に計算できないかということに挑戦しています。やっと最近ロボットが動くようになりました。

AI(人工知能)の1つという理解でしょうか。ところで、シェルに出会ったのはどういういきさつですか？

自身の研究にも猜疑心(さいぎしん)が出て、学生に就職の相談をされても、企業勤めは無経験なのでどうもうまく回答ができず、悩むくらいなら一度出ようと。また、プログラマとして腕を試したいというのもありました。そんなとき、情報系の先生の紹介でUSP研究所と出会いました。そこでシェルスクリプトやプログラムの動きを見せてもらい、「これはおもしろい」と単純に思い、遊びに行くうちに、そのまま居着いてしまいました。自分をけっこう、雑に扱ってくれたところも目的に合っていたんだと思います(笑)。

シェル芸はどんな意図で作ったのですか？

もっとUNIXを使ってもらいたいということですね。本来お堅いものであるUSP研究所のユニケージ(シェルスクリプトで開発をする手法)という開発手法から、一番エンターテイメント感のあるワンライナーを切り出しました。2012年ごろ、ハートビーツさん主催のhb

study(インフラ系勉強会)に呼ばれ、実験的に行ったのが「シェル芸人育成ドリル」でした。「シェル芸(人)」という名前は某TV番組の「〇〇芸人」と、「オタ芸」から取りました。

まさかの「アメトーク」ですか(笑)。

UNIXやLinuxを端末から使いこなすことは極めて重要だと強く信じています。ロボット屋さんもIT屋さんも、これができなくて仕事が極端に遅い人が多過ぎる。一方、きれいなグラフィックのツールや、輸入モノのツールを使って喜んでいたりする人が多い。ある種の消費者感覚です。良い研究室は「消費者、作業者」から「作成者、発信者」への意識の変化を学生に促しますが、当勉強会でもそれを意識しています。UNIXの考え方を理解し、使いこなし、自身が作る側に回ろうよと。

シェルを理解して使いこなすことが、発信側に立つきっかけになるのですね。

シェルは地味な裏方ツールです。エンターテイメントにしていくというのは、自分自身にとって、発信側としての1つの挑戦でした。幸い、周囲におもしろがる人が出てきて独自の活動をするようになったので、シェル芸はそろそろ自分から1人立ちするんではないかなと思っています。

話は変わりますが、ロボカップというものに参加されていますよね。

研究テーマとして大学4年のときから参加していました。ロボットとC++漬けでした。しばらく運営側に回って手伝いをしていましたが、久しぶりに参加者側で出ることになりました。家事ロボットの部門である@ホームリーグに参加します。

それは楽しみですね。最後に、上田さんの夢を教えてください。

制御の教科書に自分で考えた数式を1つ載せて死にたいと考えています。研究者としてはそれ以外の目標はないので、黙々と論文を書いていきます。それ以外のこととなるとおもしろいことをたくさんぶち上げていきたいです。楽しく勉強・学問できるように、学生や社会に対してさまざまなちょっかいを出していこうと考えています。

次の世代へ形となる何かを残したいのですね。偉くなっても今と変わらず仲良くしてくださいね(笑)。今日はどうもありがとうございました。 SD




ツボイのなんでもネットにつなげちまえ道場

第八回 PWMしてみる

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

はじめに

この連載の第2～4回で、LEDを光らせてみました。このLEDの明るさをマイコンからコントロールするにはどうすればよいでしょう。LEDの明るさは、LEDを流れる電流によって変わります。LEDを流れる電流は、LEDの電流制限抵抗の値や、LEDに加える電圧によって変化させることができます。電流と明るさの関係は、LEDのデータシートに記載されています。

しかし、マイコンのピン（入出力端子）から出力する電圧を変化させたり、電流制限抵抗の値をマイコンからコントロールするのは困難です。たとえば、I^2Cでコントロールすることのできる可変抵抗というデバイスが世の中にはありますが、LEDの調光をするだけの目的にはオーバーキルです。同様に、DAC（デジタル-アナログ変換回路）を使えば、マイコンから電圧をコントロールすることもできますが、同様にオーバーキルです。こういったマイコンから出力する電力をコントロールする方法に、オンとオフの繰り返しのスイッチングを行う、PWM（Pulse Width Modulation：パルス幅変調）があります（**図1**）。

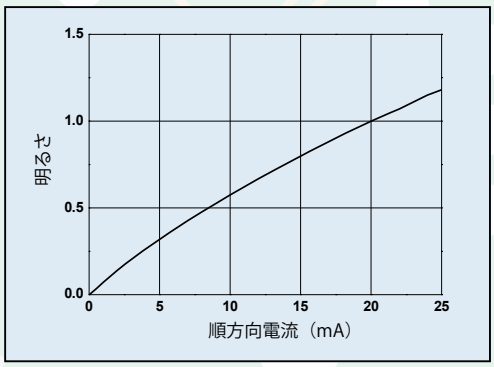
PWMとは

PWMは、先ほども述べたように、オンとオフの繰り返しをすることで、電力を制御します。たとえば、「2ミリ秒[注1]間オンにして、2ミリ秒間オフにする」ということをしてみましょう（**図2**）。

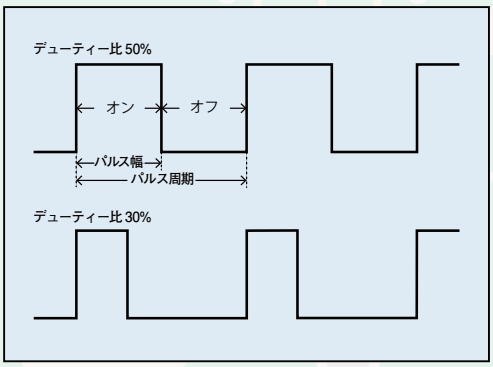
この場合、パルス周期は4ミリ秒です。パルス周期が4ミリ秒のうち、オンになっている時間が2ミリ秒ですので、この場合デューティー比は50%です。**図2**下のように、デューティー比が30%の場合には、オンになっている時間はパルス周期4ミリ秒の30%、つまり1.2ミリ秒間オンで、残りの2.8ミリ秒間がオフです。パルス周期は多くの場合PWM周波数として表現されます。パルス周期が4ミリ秒の場合、1÷0.004=250（Hz）がPWM周波数です。

注1）ミリ秒は、1,000分の1秒です。つまり、1ミリ秒は、0.001秒です。msやmsecと表記したりもします。

▼図1　LEDを流れる電流と明るさの関係（例）



▼図2　PWMの信号



PWMはさまざまな分野で使われていますが、身近な例としてLEDの点灯制御が挙げられます。高速にLEDのオン／オフを繰り返すことで、人の目に映るLEDの明るさを変化させることができます。また、RGBの3色のLEDの明るさをそれぞれ調整して組み合わせて、赤緑青のLEDの標準的な色以外のさまざまな色を出すことができます。

PWMの実験

では、mbedでPWMを使ってLEDの明るさを変えてみましょう。今回は、シンプルに、mbed LPC1768に搭載されているLEDをPWMで点灯させてみたいと思います。

リスト1のように、パルス周期とデューティー比を設定してPWMを使うこともできますし、**リスト2**のように、パルス周期をパルス幅と指定して使うこともできます。

LEDの調光をしてみる

点滅させるだけでは面白くないので、PWMを使ってLEDの調光をしてみましょう。

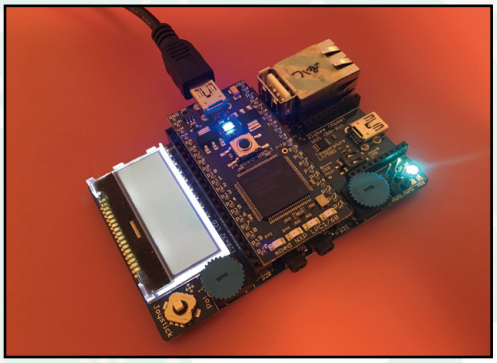
mbedのPWMは、パルス周期を設定しなければ、0.020秒です。**リスト3**ではforループの中でデューティー比を変化させることで、LEDをじわじわと点滅させています。このような点滅は、ノートパソコンのスリープ中の表示などで見かけますね。

LEDの点灯色を変えてみる

次に、先ほど例に挙げた、RGBの3色のLEDをそれぞれPWMで調光して、LEDでいろいろな色を出してみましょう(**リスト4**)。第6回でも紹介したmbedアプリケーションボード[注2]には、このRGB LEDが搭載されていますので、手軽に実験できます(**写真1**)。

PWMはオンとオフの時間を組み合わせるだけですから、ソフトウェアでオンとオ

▼写真1　PWMでLEDの色を変えてみる



▼リスト3　LEDをじわじわ点滅させる例

```
#include "mbed.h"
PwmOut led(LED1);

int main()
{
    float p = 0.0f;

    while(1) {
        for(p = 0.0f; p < 1.0f; p += 0.1f) {
            led = p;
            wait(0.1);
        }

        for(p = 1.0f; p > 0.1f; p -= 0.1f) {
            led = p;
            wait(0.1);
        }
    }
}
```

▼リスト1　パルス周期をデューティー比と設定した例

```
#include "mbed.h"
PwmOut led(LED1);

int main() {
    led.period(1.0f); // パルス周期を1秒に設定
    led.write(0.50f); // デューティー比を50%(0.5)に設定
    while(1);
}
```

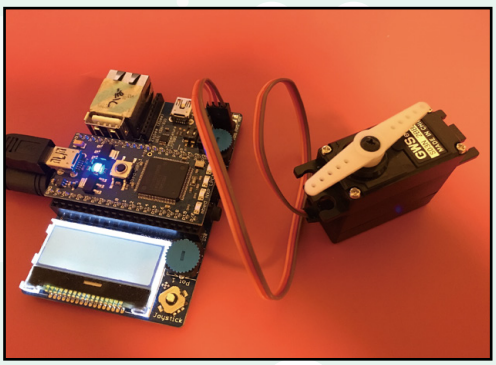
▼リスト2　パルス周期をパルス幅で指定した例

```
#include "mbed.h"
PwmOut led(LED2);

int main() {
    led.period(2.0f);  // パルス周期を2秒に設定
    led.pulsewidth(1); // 1秒間オンを出力するように設定
    while(1);
}
```

注2)　http://ssci.to/1276

フの制御を行って、ソフトウェアPWMを実装することもできます。ただし、ソフトウェアで実装すると、その分マイコンの処理能力がPWMの制御に取られてしまいます。ですので、PWMはハードウェアの機能として組み込まれている場合があります。たとえばmbed LPC1768に搭載されているLPC1768というマイコンは、PWMを同時に6つ使うことができます。使うことのできるPWMの数はマイコンによって異なります。もちろん、ハードウェアとしてPWMを搭載していないマイコンもあります。LPC1768では、PWMを使うことができるピンがあらかじめ決まっています。また、任意のピンでPWMを使うことのできるマイコンもあります。

▼写真2　PWMでサーボをコントロール



▼リスト4　LEDでいろいろな色を出してみる例

```
#include "mbed.h"
PwmOut r (p23);
PwmOut g (p24);
PwmOut b (p25);

int main()
{
    r.period(0.001);
    while(1) {
        for(float i = 0.0; i < 1.0 ; i += 0.001) {
            float p = 3 * i;
            r = 1.0 - ((p < 1.0) ? 1.0 - p : (p > 2.0) ? p - 2.0 : 0.0);
            g = 1.0 - ((p < 1.0) ? p : (p > 2.0) ? 0.0 : 2.0 - p);
            b = 1.0 - ((p < 1.0) ? 0.0 : (p > 2.0) ? 3.0 - p : p - 1.0);
            wait (0.01);
        }
    }
}
```

サーボモーター

ところで、ラジコンなどで使われているサーボモーターというモーターをご存じでしょうか。ラジコンカーのステアリングなどに使われていて、信号に応じた角度に回転するモーターです。サーボモーターは、一周期20ミリ秒のうち、オンになっている時間に応じた角度に回転します。

実際に、mbed LPC1768でサーボモーターをコントロールしてみましょう。今回は筆者の手元にあった、GWSという会社のS03N[注3]というサーボモーターを使ってみました(**写真2**)。このサーボモーターの端子配列は「JRタイプ」と記されています。ほかによく見かける端子配列には「フタバ」というものがあります。どちらも赤色の線が電源＋で、GNDは、JRタイプですと茶色、フタバですと黒色の線です。信号線は、JRタイプですとオレンジ色の線、フタバでは白色の線です。サーボモーターの線の先にあるコネクタを、mbedアプリケーションボードのPWM1という刻印がある端子に挿し込みます。アプリケーションボードの裏面に端子の説明がありますが、基板の断面に最も近いピンがGNDです。

サーボモーターは動かす際に、多くの電力を使います。mbed LPC1768にUSBから給電したのでは電流が足りませんので、mbedアプリケーションボードにACアダプタを接続してサーボへの電力を供給しましょう。mbedアプリケーションボードに接続できるACアダプタは、DC6～9Vで、コネクタの中心が1.3mmで＋のものです。このコネクタの中心が1.3mmのACアダプタはあまり見かけませんので、筆者は「DCプラグ変換プラグ　2.1mmメス⇔1.3mmオス」[注4]を使って、手元の中心が2.1mmのACアダプタのコネクタを

注3）秋月電子通商での通販コードは、M-01793です。
注4）秋月電子通商での通販コードは、C-00088です。

1.3mmに変換して使っています。ACアダプタは、サーボモーターのことを考えて6Vのものを使いましょう。筆者は、6V 2.8AのACアダプタ[注5]を使いました。

mbedアプリケーションボードにサーボモーターとACアダプタ、そしてmbed LPC1768とパソコンをUSBケーブルで接続したところで、ソフトウェアです。

先ほど記したように、サーボモーターをコントロールする信号の周期は20msですので(**図3**)、周期20msのPWM信号を出し、パルス幅を0.001秒(1ミリ秒)から0.01ミリ秒(10マイクロ秒)ずつ増やしてサーボモーターを動かしています(**リスト5**)。パルス幅を1ミリ秒から増やし始めたのには理由があります。サーボモーターには可動範囲があり、360度回転するものもありますが、たいていは180度(中心から±90度)くらいです。可動範囲を超えるとサーボモーターに負荷をかけてしまうため、サーボモーターの可動範囲のだいたい中心位置に来る、パルス幅1,500ミリ秒の近くの値にしました。この可動範囲や、パルス幅と回転角はサーボモーターによって異なります。

PWMと音

PWMで音を鳴らすこともできます。圧電スピーカは、セラミック(お茶碗のような焼き物)の一種である「圧電素子」を金属板に貼り付けた構造です。圧電素子は、電圧を加えると伸びたり縮んだりする性質を持っています。オンオフに合わせて伸び縮みするので、PWMのパルス周期で空気が振動します。圧電素子はそもそも電気を通さないので、圧電スピーカには、電気の消費が少なく、そのためマイコンから直接駆動できるという利点があります。

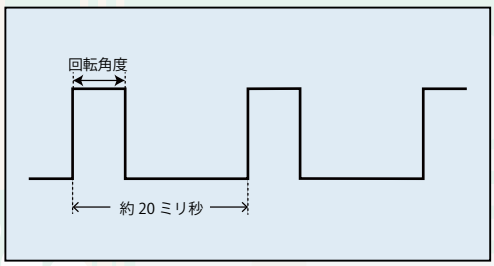
注5) 秋月電子通商での通販コードは、M-02192です。

デューティー比は、オンとオフの時間が均等になる0.5で固定します(**リスト6**)。

まとめ

PWMは調光、モーターの制御、音の再生などさまざまな用途に使うことができます。今回は扱いませんでしたが、サーボモーター以外のモーターの制御でもPWMは頻用されます。PWMによるさまざまなモータの制御は、またあらためて紹介したいと思います。SD

▼図3　サーボモーターの制御信号



▼リスト6　音を鳴らすプログラムの例

```
#include "mbed.h"
PwmOut spkr(p26);

int main() {
    for (float i=2000.0; i<10000.0; i+=100) {
        spkr.period(1.0/i);
        spkr=0.5;
        wait(0.1);
    }
    spkr=0.0;
}
```

▼リスト5　サーボモーターを動かすプログラムの例

```
#include "mbed.h"
PwmOut servo(p21);

int main() {
    servo.period(0.020); // 周期を20msに指定
    while (1) {
        servo.pulsewidth(0.001);
        wait(1);
        for(float offset=0.0; offset<0.0005; offset+=0.00001) {
            servo.pulsewidth(0.001 + offset);
            wait(0.1);
        }
    }
}
```

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**2015年2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 ハンディ洗濯機「COTON」

1名

水を噴射しながら、たたいて汚れを落とす新技術「押し出し洗い」を採用したハンディ洗濯機です。1分間に約700回振動し、押し出された汚れが、衣服の下に敷いたキッチンペーパーなどに移ります。シャツのエリ・そでの黄ばみといった手洗いでは落ちにくい汚れや、もみ洗いで生地を傷めたくない衣服の汚れにお勧めです。単4の乾電池3本で動作します。

提供元 ハイアールアジア　http://haier.co.jp

02 Lexar JumpDrive M20i (32GB)

1名

Lightningコネクタ、USB 3.0コネクタを備えたフラッシュメモリ。読み込み転送速度は最大95MB／秒、書き込みは20MB／秒です。USB 2.0のデバイスでも使用できます。

提供元 マイクロン・テクノロジ　https://www.micron.com

03 ExcelCreator 2016

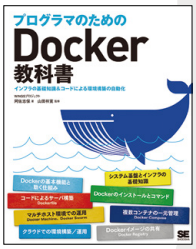
Excelファイル（xlsx/xls形式）をプログラム上で生成できるツール。Excelの環境は必要なく、独自技術によりファイルを高速に生成するので、パフォーマンスに優れたアプリを開発できます（アプリをサーバに配置して使用する場合は別途有料のライセンスが必要です）。

提供元 アドバンスソフトウェア
http://www.adv.co.jp

1名

04 プログラマのためのDocker教科書

WINGSプロジェクト 阿佐 志保、山田 祥寛 著



開発／実行環境といったインフラ構築の「自動化」の経験がない開発者を対象に、システム基盤、インフラの基礎知識に加えて、Dockerによるインフラ構築と管理の方法をやさしく解説しています。

提供元 翔泳社
http://www.shoeisha.co.jp

2名

05 新・明解C言語 実践編

柴田 望洋 著

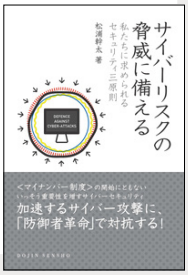
C言語の開発現場で起きた失敗談、疑問点を取り上げ、その解決方法を解説。「見えない／見えにくいエラー」「配列によって実現する線形リスト」など、他書ではなかなか見られないコードサンプルが満載です。

提供元 SBクリエイティブ
http://www.sbcr.jp

2名

06 サイバーリスクの脅威に備える

松浦 幹太 著



サイバー空間を安全・安心に利用するための技術や取り組み方を解説。さらに、専門家と一般ユーザが協力して攻撃者に対抗する「防御者革命」に基づき、安全・安心をいかに実現するかについても考察します。

提供元 化学同人
http://www.kagakudojin.co.jp

2名

07 お金をドブに捨てないシステム開発の教科書

中川 充 著

システムコンサルタント兼公認会計士の著者が“稼げるシステム”の作り方を教える1冊です。システム開発の失敗を防ぎたい情報システム部門、経営企画部担当者、経営者、そしてベンダ企業の方にお勧めです。

提供元 技術評論社
http://gihyo.jp

2名

第1特集

2大OSSデータベースの勘所を探れ！

［最新］MySQLとPostgreSQL徹底比較

導入時の「罠」を避ける現場ノウハウ

Author 曽根 壮大（そね たけとも）フリーランスエンジニア

似て非なるOSS DBの二大巨頭のMySQLとPostgreSQLは、ともにインターネットとOSSをベースとしたシステム開発の流れの中を生き抜き、多くのユーザと開発者を巻き込みながら発展してきました。それぞれに機能的な特徴があるのは当たり前ですが、MySQLを得意とする方、PostgreSQLを得意とする方と大きく 2 つに分かれるのは皆さん承知のことと思います。

本特集では、二大OSS DBの新バージョンのリリースがちょうど重なったこともあり、それぞれの機能の違いを振り返りながら解説し、エンジニアとしてより深い見識を得るために両方の勘所を探ります。コンピュータがかかわるアーキテクチャの違いから、人間がかかわるコミュニティの違いまで、本特集で一気におさえて、OSS DBを自在に使うヒントを得てください。

第1章

利用シーンの違いとアーキテクチャの違い

速度の違いはなにに起因するのか

Author 曽根 壮大（そね たけとも） Twitter @soudai1025

高速・軽量なMySQLと高機能・高可用性のPostgreSQL。それぞれが持つ特徴が生まれる理由はさまざまありますが、本章ではアーキテクチャの違いから探っていくことにしましょう。

はじめに

今、時代はデータベース（以降、DBと略記する場合あり）もオープンソースソフトウェア（以降、OSS）を使って当たり前の時代になっています。その中で選択肢に挙がるのは、二大巨頭と言っても過言ではない、

・MySQL
・PostgreSQL

の2つではないでしょうか。

実際に2011年のデータ[注1]では、Webサービスのアクセス数上位20サイトのうち、世界では18サイト、国内では19サイトで利用しているリレーショナルデータベース（以降、RDB）としてMySQLが挙げられています。それに対し、PostgreSQLは業務系サービスで多く使われてきました。

近年ではその使われ方も双方とも多様化しています。MySQLでレプリケーションを使った業務系サービスの構築も行われていますし、PostgreSQLもRuby on RailsやDjangoなどがデフォルトのDBとしてサポートしていることや多機能であることから、スタートアップを中心にWebサービスでの利用が進んでいます。

そんな歴史ある2つのDBですが、

・2つの違いがわからない
・OSSは怖くて使えない
・困ったときに誰に聞けばいいのかわからない

などの声を多く耳にします。そこでこれらを解消するために、本特集では両者の違いをさまざまな視点から説明します。

なお本文中では、

・MySQL 5.6.23 InnoDB
・PostgreSQL 9.4.5

を対象として説明します。本文中にバージョンの指定なく説明される場合は、すべて上記とお考えください。

アーキテクチャの違い

MySQLとPostgreSQLでは、それぞれのアーキテクチャから、開発の方向性、哲学の違いを感じることができます（**表1**）。MySQLは一般的に、

・高速
・軽量
・シンプル

と言われています。対してPostgreSQLは、

・高機能
・高可用性
・重厚

と言われます。その理由をアーキテクチャの違いから紐解（ひもと）いていきましょう。

注1）出典元：URL http://www.obci.jp/wp-content/uploads/2012/01/20120119_oracle.pdf（16、17ページ）

▼表1　アーキテクチャの違い

項目名	MySQL	PostgreSQL
サーバアーキテクチャ	スレッドタイプ	プロセスタイプ
ストレージアーキテクチャ	更新型	追記型

サーバアーキテクチャ

まず大きな違いとして、MySQLはスレッドタイプ、PostgreSQLはプロセスタイプとなります。これは、

新たなコネクションが発生したときにどのようなタイプで接続を行うか

ということです。

つまり、MySQLは新たにスレッドを作り、コネクションをつなぎます。それに対してPostgreSQLはプロセスを作り、forkした子プロセスとコネクションをつなぎます。このアーキテクチャの違いは、同時に処理できる数や速度に影響します。つまり並列処理能力の差になりやすい違いです。

Linuxユーザの方はマルチスレッドとforkの違いについて理解が深い人も多いのではないかと思います。一般的にスレッドタイプの方がforkのコストが不要なため高速に動作します。これは一般的なアプリケーション全般と同様です。つまり、PostgreSQLよりもMySQLの方がサーバアーキテクチャの点から見ると有利となります。

これがMySQLが高速と言われる要因の1つです。実際、大量の接続に対する処理のベンチマークを行ってみると、MySQLの方が良い結果が出ることが多くあります。PostgreSQLは、UPDATE命令についてはとくに苦手としています。そこでその理由を説明するために、次節のストレージアーキテクチャで紐解いていきます。

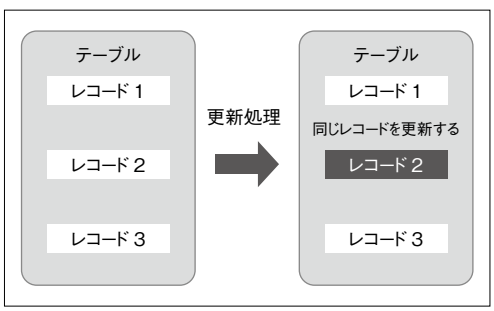
ストレージアーキテクチャ

MySQL、PostgreSQLともにMVCC（Multi Version Concurrency Control：多版型同時実行制御）を採用することによりトランザクションを実現しています。そのストレージアーキテクチャの違いですが、MySQLは更新型[注2]、PostgreSQLは追記型でMVCCを実装しています。この違いについては**図1、2**を見てください。

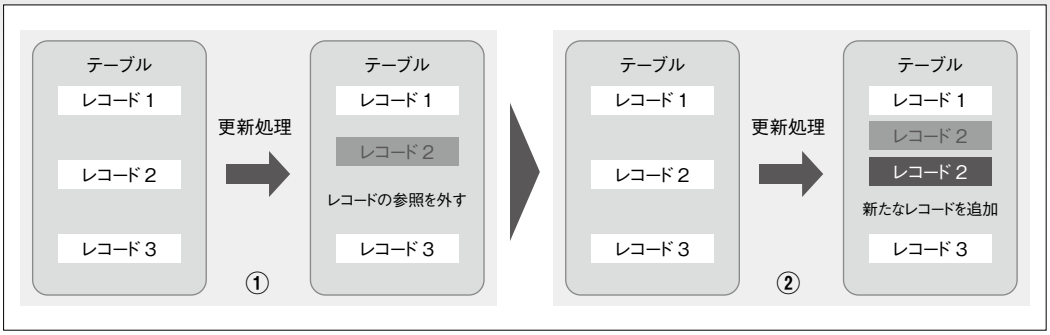
MySQLはUPDATEのときに対象のレコードを更新します。PostgreSQLは既存のレコードは保持し、新たにレコードを追加して参照先を変えます。このため、PostgreSQLはUPDATEでもデータ量が増えます。このレコード追加の更新処理がオーバーヘッドであるため、MySQLに比べてUPDATE実行時に顕著に差が出るのです。

注2） MySQLは更新型と説明しましたが、厳密にはストレージエンジン依存です。たとえばFacebookで開発されたRocksDB Storage Engine for MySQLは追記型です。PostgreSQLのような追記型でMVCCを実現したストレージエンジンを自前で作ることができるのも、MySQLの魅力です。

▼図1　MySQL（更新型の処理）



▼図2　PostgreSQL（更新型の処理）



速度の低下を抑える機能もある

しかしながら、PostgreSQLはHOTとfill factor（どちらもコラム参照）という機能でこのUPDATE時のハンデを補っています。

また、追記時などにDBに溜まるゴミデータを削除する処理、PostgreSQLではVACUUMと呼ばれますが、この処理についても8.1からAUTO VACUUMが実装されており、8.4から本格的に運用されています。そのため現在では手動でVACUUMを管理する必要は、ほぼありません。昔はVACUUMのために運用コストが高いと言われていたPostgreSQLですが、今ではほぼノーメンテナンスでも運用することができます。

一方、MySQLもDELETEのときはPostgreSQLと同じような処理になります。削除された時点では実際に行データは削除されず、該当のレコードには削除済みのstatusが付きます。それと同時にロールバックセグメントに行データが退避され、参照されなくなります。そのためMySQLではDELETEによるフラグメンテーション（断片化）が発生します。この断片化は該当のテーブルにALTER TABLEを実行することで解消できます。

実のところ、MySQLにもPostgreSQLのfillfactorとVACUUM相当の処理があります。更新型であっても更新時にページの断片化は発生します。そのため「ページの中身がある程度スカスカになってきたら、隣のページとマージする」動作が行われます。このパージ（除去）処理は専用スレッドで行われています。パージ処理はPostgreSQLのVACUUMと比べると速度への影響が少ないために運用時に問題とされることはほとんどありません。運用面のコストの少なさもMySQLのメリットといえるでしょう。

Disc I/O アーキテクチャ

MySQLとPostgreSQLの昨今の速度面での違いで重要なものがあります。それはSSDに対する相性です。データベースソフトウェアの今までの歴史は「いかにHDDにアクセスしないか」の戦いでした。しかしSSDはHDDと違い、読み出し／書き込み共に高速なスループットを発揮します。これはDB界のパラダイムシフトと言えるハードウェアの変革でした。

PostgreSQLは「いかにHDDにアクセスしないか」を追求した結果、HDDに特化したチューニングが行われています。とくに書き込み時には、HDDに合わせてシーケンシャルに書き込もうとするため、SSDの場合はそこがオーバーヘッドになります。

MySQLもHDDのシーケンシャルアクセスを意識した作りをしています。そのうえで、SSDでも性能が出るよう、次のようなことを行っています。

column

HOTとは

HOTはPostgreSQLが追記型を採用しているがゆえに生まれた機能です。Heap Only Tupleが語源で「INDEXを持たない、ヒープ(テーブルデータ)のみのタプル」という意味になります。HOTの狙いは2つです。

- 不要なINDEXの更新を行わないことによる更新処理コストの削減
- ゴミデータの自動回収

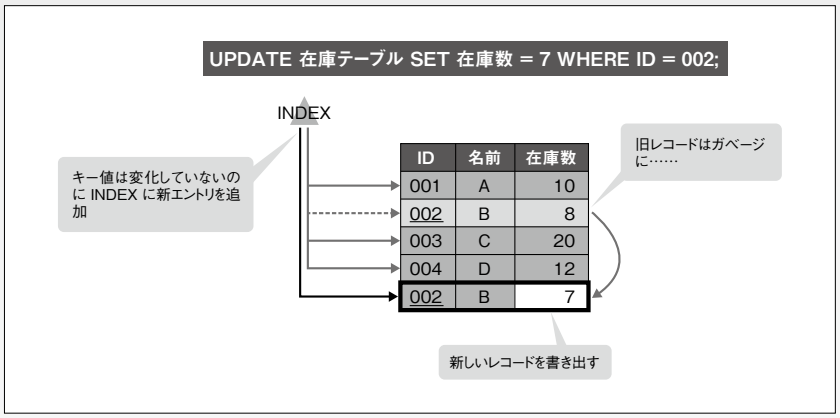
もう少し具体的に説明すると、INDEX以外を対象とした更新の際はINDEX本体の更新は行わず、テーブルの更新のみとゴミデータの回収を行ってくれるというものです。日本PostgreSQLユーザ会が運用している「Let's Postgres」の記事から画像を引用します(**図A、B**)。

注意点として、HOTは次の状況では利用されません。

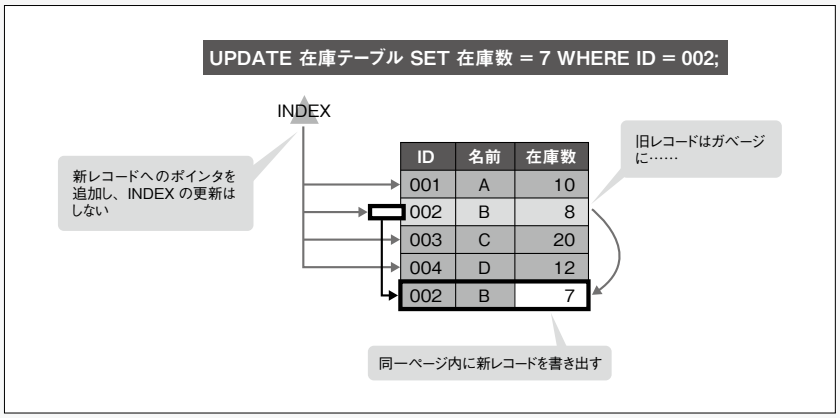
- INDEX列の更新時
- DELETE＋INSERTのような更新シーケンス
- 一度に大量の行を更新するようなUPDATE

HOTはPostgreSQL 8.3からある歴史ある機能ですので、ぜひ一度お試しください。

▼図A　従来の更新処理



▼図B　8.3からのHOT更新処理



引用元：HOTの仕組み(1)
http://lets.postgresql.jp/documents/tutorial/hot_2/hot2_1

column

fillfactorとは

HOTはコラムにあるように、更新時に空きスペースをうまく使うしくみです。一方、fillfactorは新規データ作成時にHOT用の空きスペースを作成する機能です。

たとえばfillfactor=80を設定するとデータファイルに80%を使い、20%は空き領域として確保するということになります。この機能によってPostgreSQLは高速なUPDATE実行とVACUUMのコスト削減を実現しています。デフォルトでは、

- TABLE 100%
- INDEX(Btree) 90%

が指定されています。SELECTとINSERTが主なクエリの場合はデフォルトのままで問題ありません。

勘の良い読者は「fillfactorを使って常に対象レコードの分の空き領域を確保すれば、UPDATEでデータファイルを再利用できる」と考えるかもしれません。確かにそのとおりで、常に空き領域を同量分確保するため交互に更新を行います。しかしこれはファイル領域を2倍使用することになり、INSERTのコストが高くなります。また当然ディスク容量の消費も多くなるため、下限は70%程度までと言われています。逆にまったくUPDATEが行われない場合は、fillfactorを100%にすることでディスク容量を有効に使うことができます。

基本的にデフォルトから無理にチューニングする項目ではありませんが、要件に合わせて変更すると効果的な項目ですので、ぜひ覚えておいてください。

innodb_flush_method=O_DIRECT

innodb_flush_methodはUNIX/Linuxにおいてデータファイル、ログファイルの読み書き方式を指定するためのものです。このパラメータにはおもに次の4種類[注3]の設定が可能です。

- fsync
- O_DSYNC
- O_DIRECT
- O_DIRECT_NO_FSYNC

通常はfsyncがデフォルトで設定されています。O_DIRECTはDirect I/Oを使うことを意味し、OSのページキャッシュを無視して書き込みを行います。このO_DIRECTがSSDと相性が良いです。

PostgreSQLもwal_sync_methodでO_DIRECTを利用するように設定できますが、影響範囲がWAL(Write Ahead Logging：ログ先行書き込み)のみとなっています。

NVMFSに特化している

MySQLはFusion-ioのNVMFSに特化したチューニングが行われています。そのためとくにFusion-io利用時は通常のSSDよりもパフォーマンスが発揮されます。

これらのことから、HDDで同じパフォーマンスを発揮していた環境をSSDにした場合、PostgreSQLとMySQLでは差が出ます。とくに前述のFusion-ioを使った場合には、数十倍の差が出ることもあります。

◆ ◆ ◆

以上見てきたとおり、ハードウェアにかかわる速度面では圧倒的にMySQLの方が有利です。これが冒頭で触れた、アクセス上位のサイトでMySQLが利用される理由の1つと言えるでしょう。

しかし、一概にどんなシーンでもMySQLが高速だとは限りません。その理由については第3章や第4章で詳しく説明します。SD

注3) innodb_flush_methodの公式ドキュメント
URL http://dev.mysql.com/doc/refman/5.6/ja/innodb-parameters.html#sysvar_innodb_flush_method

第2章 利用時の違い

ライセンスの検討、インストールとクライアントツールを知る

Author 曽根 壮大(そね たけとも) Twitter @soudai1025

第1章ではアーキテクチャを中心に違いを説明しました。本章では利用時のライセンス、インストール方法、クライアントからの接続方法の違いについて説明します。

利用時のライセンスの違い

オープンソースソフトウェア(以降、OSS)を実際に利用するとき、重要な点にライセンスがあります。MySQLとPostgreSQLは同じOSSといっても、ライセンスに違いがあるため利用方法に差が出てきます。まずはメジャーなライセンスについて、**表1〜4**にまとめます。

まずPostgreSQLについてですが、ライセンスはBSDを元にしたPostgreSQL Licenseという独自ライセンスとなっています。内容はライセンス条件の記述をソースファイルやドキュメント上に残すことで、再頒布や再利用を許可しています。この条件を守っている場合は商用・非商用関係なくPostgreSQLの改良や再配布、またソースコードを非公開にできます。そのため、PostgreSQLをベースにした商用製品などを作ることができます。実際に商用利用された例ですと、

- EnterpriseDB「Postgres Plus Advanced Server」
- SRA OSS, Inc. 日本支社「PowerGres」
- Greenplum「Greenplum Database」

などの改良した製品が販売されています。

それに対してMySQLですが、GPL v2と商用ライセンスのダブルライセンスです。このように複数のライセンスを持つOSSは、ほかにもFirefox[注1]などがあります。商用ライセンスは商用版MySQLを利用する際に適用されます。Webサービスなどで使われているMySQLはOSS版であるため、GPL v2が適用されます。GPL v2ですので、MySQLを改良してソフトウェアとして利用・販売する場合はコードの明示などが必要になります。ただし、Webサービスのようにプロトコルを通して利用するだけの場合は、ソースコードの開示は必要ありません。

OSSのライセンスはたくさんありますが、利用するときは必ず確認し、ライセンスを遵守しましょう。

インストール

CentOSを例に取ると、MySQLはCentOS 7からデフォルトでbaseにRPMが用意されなくなりました。PostgreSQLはデフォルトでは9.2がインストールされています。しかし、両DBとも最新版のRPMが用意されているため、インストールについては問題なく行うことができます。**図1〜4**に、CentOS 7でのインストール例を記載します。

このような簡単なコマンド実行だけで、どちらもすぐに使い始めることができます。最近はAnsibleやChefでの自動環境構築の方法が公開されていますので、うまく利用すればもっと簡単に構築・設定できるでしょう。また、

注1) FirefoxはMPL 2.0になるまで、MPL1.1、GPL、LGPLのトリプルライセンスで運用されていました。

MySQLは公式にDockerのコンテナを提供しています[注2](執筆時はまだ準備中)。こちらを使うとより簡単に環境構築が行え、お試しできると思います。

クライアントからの接続

クライアントからの接続はGUIベースとCUIベースそれぞれが用意されています。

まずGUIですが、MySQLもPostgreSQLも無料でツールが公開されています。

・**MySQL Workbench**
https://www-jp.mysql.com/products/workbench/

・**pgadmin3**
http://www.pgadmin.org/

MySQL Workbench

MySQL Workbenchは名前のとおりMySQL専用のクライアントソフトです。MacでもWindowsでも動作し、非常に優秀でいろいろな機能が無料で使えます。ごく簡単に、検索と実行計画の表示機能を紹介します。

MySQL Workbenchはテーブルの表示をすると**図5**のように該当のtableのSELECT文を発行します。GUI上でレコードの内容を変更することもできますし、発行されたSQLを修正してORDER BYやWHERE句を追加することもできます。

さらにSQLの実行計画を見たいときは、**図**

注2) URL https://blogs.oracle.com/MySQL/entry/oracle_s_mysql_image_coming

▼表1　オープンソースライセンス一覧

ライセンス	Required(必須)	Permitted(許可)	Forbidden(禁止)
GPL v2	著作権の表示、変更箇所の明示、ソースの明示	商用利用、修正、配布、特許許可	責任免除、サブライセンス
GPL v3	著作権の表示、変更箇所の明示、ソースの明示	商用利用、修正、配布、特許許可	責任免除、サブライセンス
LGPL v3	著作権の表示、ライブラリの使用、ソースの明示	商用利用、修正、配布、サブライセンス、特許許可	責任免除
BSD 2-Clause	著作権の表示	商用利用、修正、配布、サブライセンス	責任免除

▼表2　Required(必須)の補足説明

項目名	説明
License and copyright notice(著作権の表示)	ライセンスと著作権のコピーをコードに含めなくてはいけません
Library Usage(ライブラリの使用)	ライブラリは非オープンソースのアプリケーションで使用されるかもしれません
State Changes(変更箇所の明示)	コード内の重要な変更箇所を明示してください
Disclose Source(ソースの明示)	ライブラリのソースを明らかにしてください

▼表3　Permitted(許可)の補足説明

項目名	説明
Sublicense(サブライセンス)	このソフトウェアはサブライセンスを与えてもよいです
Modifications(修正)	このソフトウェアは修正してもよいです
Commercial Use(商用利用)	このソフトウェアは商用目的のために使用してもよいです
Distribution(配布)	このソフトウェアを配布してもよいです
Patent Grant(特許許可)	このライセンスは個人または企業(コントリビュータ)をはじめ、すべての利用者に特許権の特別な許可を提供します

▼表4　Forbidden(禁止)の補足説明

項目名	説明
Hold Liable(責任免除)	ソフトウェアは保証なしで提供されます。ソフトウェアによる損害賠償金などの法的責任は負わされません
No Sublicense(サブライセンス禁止)	ライセンスに含まれない変更をして、ソフトウェアにサブライセンスを与えてはいけません

表1〜4の引用元：たくさんあるオープンソースライセンスのそれぞれの特徴のまとめ(一部抜粋)
http://coliss.com/articles/build-websites/operation/work/choose-a-license-by-github.html

5-❶のアイコンをクリックすると、図5-❷のようにグラフィカルに表示できます。図5-❷では、遅延の原因になりやすい「Full Table Scan」が実行されているため、該当個所が赤く表示されます。INDEXを利用した高速な実行計画の場合は青色で表示されます。JOINやサブクエリを行ったときは対象のテーブルのオブジェクトがそれぞれ作られます。そのため、複雑なSQLでも実行計画の赤色の部分を確認することで、簡単にボトルネックとなる場所を見つけられます。

◆ ◆ ◆

MySQL Workbenchはここでは紹介しきれないほど多機能です。残念ながら日本語版がありませんが比較的簡単な英語ばかりなので、ぜひ使ってみてください。

▼図1　MySQLのインストール

```
# yum -y install http://dev.mysql.com/get/mysql57-community-release-el7-7.noarch.rpm
# yum -y install mysql-community-client mysql-community-devel mysql-community-server
# service mysqld start
```

▼図2　MySQLのセットアップ

```
# /usr/bin/mysql_secure_installation
```

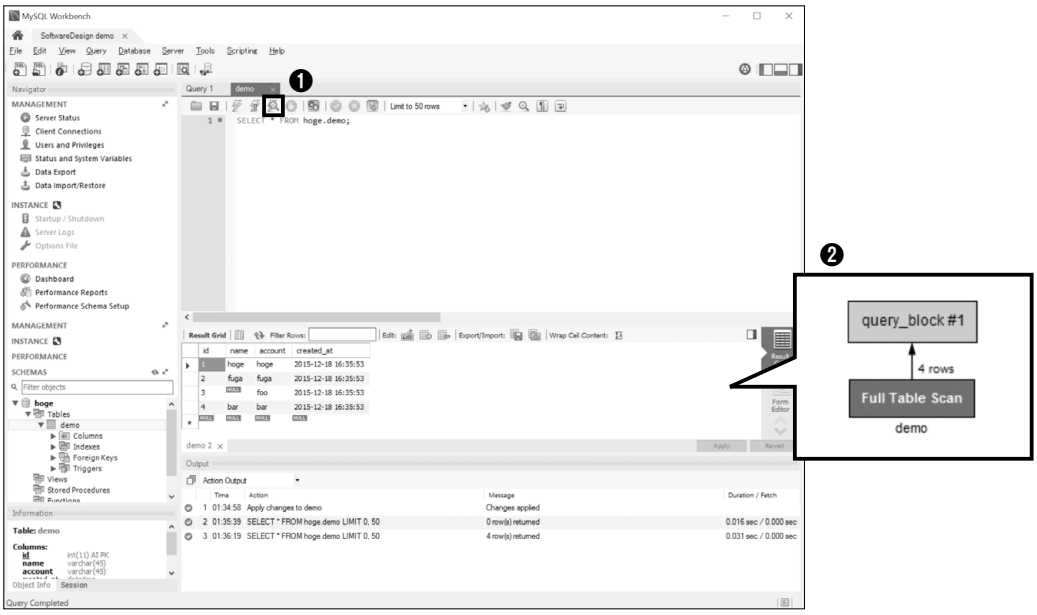
▼図3　PostgreSQLのインストール

```
# yum -y install http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/pgdg-centos94-9.4-1.noarch.rpm
# yum install postgresql94-server postgresql94-contrib
```

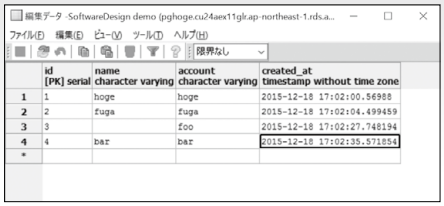
▼図4　PostgreSQLのセットアップ

```
# /usr/pgsql-9.4/bin/postgresql94-setup initdb --no-locale
 ↑デフォルトは0Sのロケールを使い、ソートに問題が発生しやすいためno-localeを指定しましょう
# systemctl start postgresql-9.4
```

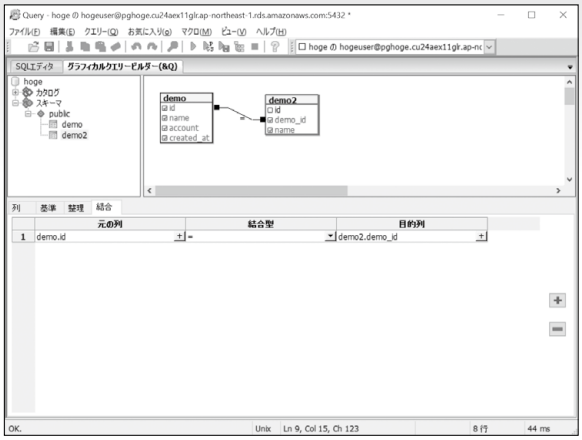
▼図5　MySQL Workbench：検索と実行計画の表示



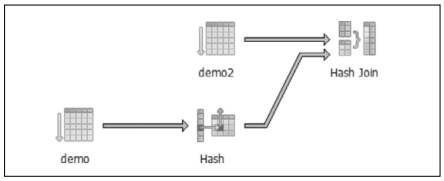
▼図6　pgadmin3：テーブル表示



▼図7　pgadmin3：グラフィカルクエリービルダー



▼図8　pgadmin3：実行計画の表示



pgadmin3

PostgreSQL専用のクライアントソフトであるpgadmin3は、MacでもWindowsでも動作し、無料で、さらに日本語版があります。MySQL Workbenchと同様に検索と実行計画を紹介します。

pgadmin3でテーブルの表示を選ぶと、**図6**のように表示されます。このGUI上で直接レコードの編集ができますし、ソート条件の追加なども可能です。ただしMySQL Workbenchとは違い、SQLを直接修正することはできません。そのためSELECT文を使うときは「グラフィカルクエリービルダー」を利用します。

もちろん直接SQLを書くこともできますが、pgadmin3の大きな特徴の1つであるグラフィカルクエリービルダーを利用すると、**図7**のようにGUI上で条件を選択するだけでSQLを作成できます。発行されたSQLの表示画面では、直接SQLを修正することもできます。またこのSQLの実行計画を確認すると、**図8**のようにグラフィカルに表示できます。

◆　◆　◆

MySQL Workbenchと同様pgadmin3も高機能で、PostgreSQLの運用を助けてくれる頼もしいソフトウェアです。しかし、次のような翻訳が散見されます。

DROP TABLE→削除／抹消
確認メッセージ→「テーブル"hoge"を抹消しますか？」

TRUNCATE→抹消
確認メッセージ→「このテーブルを切除しますか？警告：この操作でテーブル内の全てのデータが削除されます！」

ほかにも「external table」と「foreign table」が両方「外部テーブル」と表現されているなど、残念な翻訳が目立ちます。そのためpgadmin3を利用される場合は英語版をお勧めします。

CUI

最後に、CUIからの接続です。CUIはTerminalからそれぞれ次のコマンドで利用できます。

・mysql
・psql

mysqlでは、SELECTなどのSQLは大文字で入力しないと補完されないなどの注意点があります。

PostgreSQLを使う方はGUIツールよりもpsqlを使う方が多いようです。psqlはデフォルトで補完が効きますが、.psqlrcを設定することで自分好みにカスタマイズできます。

自分にあったアクセス方法を選んでみてください。SD

第3章 SQLの違い

実務で気をつけるべき構文とその挙動

Author 曽根 壮大（そね たけとも） Twitter @soudai1025

MySQLとPostgreSQLはともにSQLの標準規格に則っており、SQLの構文に大きな違いはありません。しかしDBによって、同じ機能でも書くべきSQLが違う、同じSQLでも実行すると挙動が違うなど、注意点がいくつかあります。本章では、MERGE文、Windows関数、ALTER文での両DBのSQLの違いについて見ていきます。

文法や機能の違い

リレーショナルデータベースに対するアクセスは、SQLで行うのが一般的です。例に漏れず、MySQLもPostgreSQLも、その名前のとおりSQLを利用します。そこで本章では、実務で一番のメインになるであろうSQLの違いについて説明します。

SQLには**表1**のような標準規格があります。この標準はSQLの機能、動作、文法、リレーショナルデータベースの標準的な動作を定義しています。PostgreSQLは良くも悪くも、この標準になるべく沿うように開発されています。また、MySQLも基本的にはこの標準を準拠しており、どちらのデータベースを利用しても基本的なSQLの構文に違いはありません。

たとえば、

```
SELECT * FROM table_name;
CREATE DATABASE database_name;
```

のような構文はMySQL、PostgreSQLどちらでも動作します。そのため、リレーショナルデータベースを触った経験がある方であれば、すぐに利用できます。

しかし、両DBとも利便性の向上から独自の機能を用意していたり、逆にサポートしていない機能があったりします。その点を説明します。

MERGE文をどのように実現するか

MERGE文はSQL標準にも規定されている機能です。動作は次のとおりです。

- 対象のtableに対して、primary keyまたはunique keyを確認
- primary keyまたはunique keyの衝突がなければ、指定のデータをINSERT
- primary keyまたはunique keyが衝突した場合は、指定のデータでUPDATE

つまり、対象のデータがない場合はINSERT、ある場合はUPDATEを行ってくれます。MERGE文は一度SELECT文を流して確認する必要がないため、非常に便利な機能です。たとえば、ゲームデータを登録するSAVE機能などを実装するようなとき、

- 新規ユーザによる初めてのSAVEはINSERT
- 既存ユーザによるSAVEの場合はUPDATE

を自動的に実施してくれます。実際に筆者もAndroidアプリのゲーム内ランキングに利用するクリアデータの保存などで利用しています。

しかし、この機能は残念ながら両DBともサポートしていません。そこで本節では、両DBでMERGE文に相当する機能をどのように実現するのか、それぞれのアプローチを紹介したいと思います。

▼表1　SQLの標準規格（参考：https://ja.wikipedia.org/wiki/SQL）

年	規格名称	説明
1986	SQL86	ANSIによって発表された最初の規約。1987年にISOによって批准された。データ操作言語（DML）仕様策定：COBOL、FORTRAN、PL/Iなど、親言語（母言語、ホスト言語とも言う）への埋め込みSQL文仕様策定
1989	SQL89	マイナーバージョン。データ定義言語（DDL）仕様策定（CREATE TABLE文、CREATE VIEW文、GRANT文。ただし、DROP文、ALTER文、REVOKE文はなし）／制約および整合性機能を追加（DEFAULT、UNIQUE制約、NOT NULL制約、PRIMARY KEY制約、CHECK制約、参照整合性制約）／C言語への埋め込みSQL文仕様の追加
1992	SQL92	メジャーバージョン。直交性の改善（表式）／データ型の拡張（可変長文字列、ビット、文字集合、日付・時刻・時間間隔（DATE、TIME、TIMESTAMP、INTERVAL））／外部結合（OUTER JOIN）／定義域（DOMAIN）／表明（ASSERTION）／一時表（TEMPORARY TABLE：永続化しないデータを格納）／DDL仕様追加（DROP文、ALTER文）／動的SQL仕様前方・後方スクロール可能なカーソルサポート／クライアント、サーバシステムのためのCONNECT/DISCONNECT文
1995	SQL/CLI	コールレベルインターフェース（Call Level Interface）。業界標準になったODBC APIのインターフェースに相当する機能を国際標準化した規格
1996	SQL/PSM	永続格納モジュール（Persistent Storage Module）。一般的にストアドプロシージャと呼ばれる機能を国際標準化した規格
1999	SQL:1999（SQL99）	RDBMSのための完全な言語になることを目指した仕様。正規表現による値照合／共通表式（WITH句）／再帰クエリOLAP（ROLLUP、CUBE、GROUPING SETS）／ユニオン（UNION）、結合経由の更新／カーソル操作の機能強化（トランザクション完了後のオープン状態保持）・ユーザ定義権限（ROLE）・トランザクション管理の新機能（SAVEPOINT）／SQL/PSM強化（制御構文（IF、WHILEなど）サポートなど）／SQLJ（Javaを親言語とする埋め込みSQL規格）／データベーストリガ／ユーザ定義関数（ストアドファンクション）／非スカラー型の新しいデータ型：真理値（BOOLEAN）型と配列（ARRAY）型、LOB（Large Object）、ユーザ定義型、構造型／上位表と下位表（スーパーテーブルとサブテーブル）／オブジェクト指向の考え方を取り入れたオブジェクトリレーショナルデータベース技術（ORDB）。配列型やユーザ定義型、ユーザ定義関数と上位表、下位表仕様により実現されている
2003	SQL:2003	SQL/MM（マルチメディア：フレームワーク、全文検索、空間データ（Spatial）、静止画像）／SQL/MED（外部データ管理：非リレーショナルデータ（順編成ファイルや階層型データベースなど）や他社のリレーショナルデータをSQLでアクセスするための規格）／SQL/OLB（オブジェクト言語バインディング：SQLJを標準化する。Javaプログラムに埋め込むSQL文）／XML関連の機能ウィンドウ関数／順序（シーケンス）の標準化と識別キー列に対する値の自動生成を行う列仕様の導入（ID型）
2008	SQL:2008	INSTEAD OFトリガ／TRUNCATE TABLEステートメント／配列型の集約と展開（array_agg, unnest）

▼リスト1　INSERT ON DUPLICATE KEY UPDATE構文

```
INSERT INTO table名 (column1, column2, column3) values (value1, value2, value3)
    ON DUPLICATE KEY
       UPDATE column2=value2, column3=value3;
```

MySQLの場合

まずMySQLですが、MERGE文に相当するものとして、

・INSERT ON DUPLICATE KEY UPDATE構文
・REPLACE文

の2つが用意されています。

INSERT ON DUPLICATE KEY UPDATE構文

まず、INSERT ON DUPLICATE KEY UPDATE構文ですが、MERGE文に非常に近い動作をします。SQLは**リスト1**となります。実際に流したときの動作は**図1**のとおりです。id 3とid 4は通常のINSERTとなり、id 2はUPDATEが行われています。注目は実行結果です。

```
Query OK, 4 rows affected (0,00 sec)
Records: 3  Duplicates: 1  Warnings: 0
```

これは、「4件のSQLを実行したが、3件はそのまま実行され、1件は一意性制約で弾かれた」ことを表しています。つまり、実際にはid 2に対してもINSERTは実行されており、それが失敗したためにUPDATEが行われたことを表しています。このことからわかるとおり、この構文を使ううえでの注意点は、

▼図1　INSERT ON DUPLICATE KEY UPDATEを実行

```
mysql> SELECT * FROM fuga;
+----+------+---------------------+---------------------+
| id | name | created_at          | updated_at          |
+----+------+---------------------+---------------------+
|  1 | hoge | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
|  2 | fuga | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
+----+------+---------------------+---------------------+
2 rows in set (0,00 sec)

mysql> INSERT INTO fuga (id, name) values (null, 'bar'),(null, 'foo'),(2, 'titi')
    ->     ON DUPLICATE KEY
    ->        UPDATE name='update';
Query OK, 4 rows affected (0,00 sec)
Records: 3  Duplicates: 1  Warnings: 0

mysql> SELECT * FROM fuga;
+----+--------+---------------------+---------------------+
| id | name   | created_at          | updated_at          |
+----+--------+---------------------+---------------------+
|  1 | hoge   | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
|  2 | update | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
|  3 | bar    | 2015-11-24 17:55:45 | 2015-11-24 17:55:45 |
|  4 | foo    | 2015-11-24 17:55:45 | 2015-11-24 17:55:45 |
+----+--------+---------------------+---------------------+
4 rows in set (0,00 sec)
```

▼図2　INSERTに失敗しても値が進み、1つ飛ばしでINSERTされる

```
mysql> INSERT INTO fuga (id, name) values (null, 'bar'),(null, 'foo'),(2, 'titi')
    ->     ON DUPLICATE KEY
    ->        UPDATE name='update';
Query OK, 2 rows affected (0,01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM fuga;
+----+--------+---------------------+---------------------+
| id | name   | created_at          | updated_at          |
+----+--------+---------------------+---------------------+
|  1 | hoge   | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
|  2 | update | 2015-11-24 17:52:48 | 2015-11-24 17:52:48 |
|  3 | bar    | 2015-11-24 17:55:45 | 2015-11-24 17:55:45 |
|  4 | foo    | 2015-11-24 17:55:45 | 2015-11-24 17:55:45 |
|  6 | bar    | 2015-11-24 18:00:25 | 2015-11-24 18:00:25 |
|  7 | foo    | 2015-11-24 18:00:25 | 2015-11-24 18:00:25 |
+----+--------+---------------------+---------------------+
6 rows in set (0,00 sec)
```

・INSERTをまず行う
・UPDATEはINSERTが終了後に行われる

となります。これは、バルクインサート[注1]やINSERT SELECTなどを行ったときに影響します。

具体例は**図2**のとおりです。**図1**と同じSQLを再度実行しました。設定によっては**図2**のようにオートインクリメントを利用していた場合、INSERTに失敗しても値が進みます。そのため、2回目の新たなINSERTはid 5を飛ばして、id 6とid 7を生成しています。

“設定によっては”と前置きしましたが、innodb_autoinc_lock_modeのパラメータを変えることで防げます。デフォルトでは、innodb_autoinc_lock_mode=1が設定されていますが、このパラメータに0を指定することで挙動を変えることができます。また、innodb_autoinc_

注1） 一度のINSERT文で複数レコードをDBにINSERTするもの。

lock_mode=0は1に比べて性能劣化が激しいので注意が必要です。挙動についてはたとえば、

```
+-----+------+
| c1  | c2   |
+-----+------+
|   1 | a    |
| 101 | b    |
|   5 | c    |
| 102 | d    |
+-----+------+
```

のようなテーブルに対して、

```
INSERT INTO t1 (c1,c2) VALUES (1,'a'), ➥
(NULL,'b'), (5,'c'), (NULL,'d');
```

のようなSQLを実行したときに、innodb_autoinc_lock_modeが、

・「0」のとき、次の自動インクリメント値は「103」
・「1」のとき、次の自動インクリメント値は「105」

という違いが出ます。アプリケーションの仕様にも関わる挙動の違いですので、よくよくご検討のうえで設定してください。本構文は、シンプルな動作かつINSERT文の構文に近いため、目にする機会がよくあるでしょう。

REPLACE文

もう1つのアプローチがREPLACE文です。OracleDBには同名で正規表現を利用できる関数がありますが、それとはまったく別の機能となります。動作としましては、

・指定されたprimary keyまたはunique keyの対象データを削除
・指定されたデータをINSERT

という動作になります。名前のとおり、すべて置き換えるわけです。注意点は文字どおり、いったん削除するということです。構文は次のように、INSERT文に非常に似ています。

```
REPLACE INTO table名 (column1, column2, ➥
column3) values (value1, value2, value3);
```

頭のINSERTをREPLACEに変更するとそのまま使えます。既存のコードからもし置き換える必要があるときに、簡易に対応できるところがメリットの1つです。

しかし前述のとおり、対象のレコードをいったんすべて削除してしまいます。たとえばunique keyで指定しており、別途サロゲートキーをprimary keyとしてオートインクリメントで発行した場合などは、そのIDが変更されてしまいます。前述のINSERT ON DUPLICATE KEY UPDATE構文は削除をしないため、外部Key制約に対する影響は少ないと言えます。しかし、REPLACE文は該当のデータをいったん削除するため、もし外部Key制約を利用していた場合には影響を受けます。CASCADE ON DELETEを指定していた場合には、関連するデータもすべて消えてしまいます。

これら影響については、きちんと認識して利用するようにしましょう。もし、どちらでも目的を達成できる場合はINSERT ON DUPLICATE KEY UPDATE構文をお勧めします。

また、トリガーを使っている場合も影響を受けます。しかも、トリガーの挙動が直感的ではありません。

・REPLACE INTOは行を消してから新しい行を書く
・トリガーは、INSERTトリガーが先に引かれて、DELETEトリガーが後から引かれる

となります。具体的には、

①before insert
②before delete
③after delete
④after insert

の順でイベントが実行されます。トリガー利用時のREPLACE文にはご注意ください。

PostgreSQLの場合

ここまではMySQLの説明でしたが、続いてPostgreSQLです。PostgreSQLには、MERGE文相当の機能は現在ありません。そのため、同様の機能を実現する場合は、トリガーを利用するかWITH句＋RETURNING句を利用します。

今回は、WITH句+RETURNING句を説明します。構文は**リスト2**のとおりです。こちらは、

・対象のデータベースからUPDATEの対象を抽出
・UPDATEの対象を更新
・更新予定の一覧からUPDATE外のデータを作成し、INSERT

という動作になります。処理の概要をイメージ図にすると**図3**のとおりです。非常に多くの処理を行っているのがわかります。

通常は、PHPなどのプログラミング言語で中間結果を変数に受け取るのが一般的な処理かと思います。また、データが大きくなると速度の影響も大きいため、高速な処理とは言い難いです。そのためPostgreSQLで同様の機能を実現したい場合は、それ相当のトリガーやストアドプロシジャを実装することになります。

ちなみに、MySQLはWITH句もRETURNING句もサポートしていません。MERGE文相当の構文こそないものの、SQLの構文の多さはPostgreSQLに利があります。

Window関数

まず結論から述べると、MySQLにはWindow関数はありません。いろいろなところで渇望の声を聞きますが、最新版のMySQL 5.7でも

▼リスト2　WITH句+RETURNING句でMERGE文を再現

```
CREATE TABLE tbl (id integer PRIMARY KEY, v integer);

WITH val AS (SELECT ((1, 1235)::tbl).*),
     upd AS (UPDATE tbl SET v = val.v FROM val
                WHERE tbl.id = val.id RETURNING tbl.id)

INSERT INTO tbl SELECT * FROM val
    WHERE id NOT IN (SELECT id FROM upd);
```

column

救いの手？　PostgreSQL 9.5

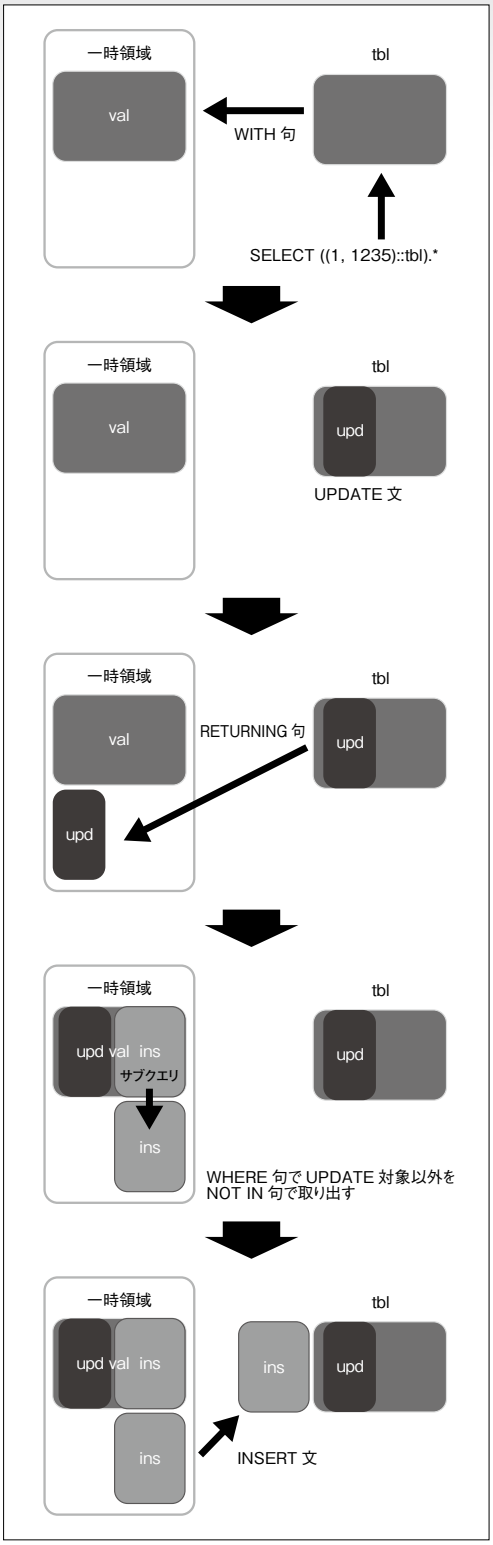
PostgreSQL 9.5からINSERT ON CONFLICT構文という機能が追加になりました。これはMySQLの、INSERT ON DUPLICATE KEY UPDATE構文相当の機能になります。SQLも非常に似ており、**リストA**のように書くことで同様の処理を実行できます。

PostgreSQLのコミュニティでは長年MERGE文相当の機能が渇望されていましたが、次のバージョンからサポートされます。この機能だけでも新バージョンを使う価値があると思います。本誌を読まれているころにはリリースされる予定ですので、ぜひ使ってみてください。

▼リストA　INSERT ON CONFLICT構文

```
INSERT INTO table名 (column1, column2, column3) values (value1, value2, value3)
   ON CONFLICT 制約名
   DO UPDATE SET ( カラム名 = 値 )
```

▼図3　WITH句+RETURNING句の動作イメージ



実装されませんでした。それに対して、PostgreSQLにはWindow関数があります。

Window関数はテーブルをパーテーション(ひとまとまり)ごとに集計する機能です。検索した行に対して、パーテーションを指定して全般の計算を行えます。Window関数は集約関数(GROUP BY)に似ていますが、Window関数では複数の行がまとめられることはなく、行それぞれが返却されます。また、処理中の行以外の行の値を読み取ることもできます。Window関数は筆者が非常によく使う機能の1つです。次のような場合に1つのSQLで実現できます。

・ゲームの状況別(クリアマップ別など)のランキング集計
・月別の出庫管理台帳や日別の金銭台帳
・店舗別の売上集計

このように、グループ分けした対象ごとの集計で力を発揮します。実際に実行結果を見たほうがわかりやすいので、PostgreSQLの公式ドキュメント[注2]から例を引用します(**図4**)。PARTITION BY カラム名を渡すことで区切りとなるパーテーションを指定できます。そのパーテーションに対して行う処理を、OVER ()の前にある関数で指定します。

PostgreSQLでWindow関数で指定できる関数は**表2**のとおりです。

もちろん同じようなことを、MySQLでもサブクエリやJOINを使えば実現できます。しかし、残念ながらMySQLは相関サブクエリを非常に苦手としており、高速とは言えません。また、JOINを利用する場合もクエリ対象がJOINの回数分増えるわけですから、Window関数より高速とは言えません。単純なSELECT文ではMySQLに利がありますが、このような集計などのSELECT文ではPostgreSQLに利があり

注2) URL https://www.postgresql.jp/document/9.3/html/tutorial-window.html

ます。これが、もともと業務系でPostgreSQLが利用されていた背景の1つです。

近年ではスタートアップのWebサービスでも多くのデータを扱います。そのため複雑なSQLを必要とする場面が多くあり、スタートアップでもPostgreSQLを利用するシーンが増えています。筆者個人の意見としても、高速で成長していくスタートアップでは、まずPostgreSQLを使うことをお勧めします。最近では、AWSのAmazon RDSを始めとするDBaaSでもPostgreSQLをサポートしているで、すぐ使い始められます。

ALTER文

TABLEの情報を更新するときは、MySQL

▼表2　利用可能なWindow関数

関数	説明
row_number()	行番号
rank()	ランキング（同率で番号を飛ばす）
dense_rank()	ランキング（同率で番号を飛ばさない）
percent_rank()	ランキング（%で表示）：(rank-1)／(全行数-1)
cume_dist()	percent_rankに類似：(現在の行の位置)／(全行数)
ntile(N)	ランキング（1..Nに分割）
lag(value, offset, default)	ソート状態での前の行の値
lead(value, offset, default)	ソート状態での後の行の値
first_value(value)	最初の値
last_value(value)	最後の値
nth_value(value, N)	N番目の値（1から数える）

▼図4　Window関数の例

・ある部署の平均給与とそれぞれの従業員の給与をどのように比較するかを示した例

```
postgres=# SELECT depname, empno, salary,
                  avg(salary) OVER (PARTITION BY depname)
FROM empsalary;

  depname  | empno | salary |          avg
-----------+-------+--------+----------------------
 develop   |    11 |   5200 | 5020.0000000000000000
 develop   |     7 |   4200 | 5020.0000000000000000
 develop   |     9 |   4500 | 5020.0000000000000000
 develop   |     8 |   6000 | 5020.0000000000000000
 develop   |    10 |   5200 | 5020.0000000000000000
 personnel |     5 |   3500 | 3700.0000000000000000
 personnel |     2 |   3900 | 3700.0000000000000000
 sales     |     3 |   4800 | 4866.6666666666666667
 sales     |     1 |   5000 | 4866.6666666666666667
 sales     |     4 |   4800 | 4866.6666666666666667
(10 rows)
```

・部署ごとの給料順のランキングを示した例

```
postgres=# SELECT depname, empno, salary,
                  rank() OVER (PARTITION BY depname ORDER BY salary DESC)
FROM empsalary;

  depname  | empno | salary | rank
-----------+-------+--------+------
 develop   |     8 |   6000 |    1
 develop   |    10 |   5200 |    2
 develop   |    11 |   5200 |    2
 develop   |     9 |   4500 |    4
 develop   |     7 |   4200 |    5
 personnel |     2 |   3900 |    1
 personnel |     5 |   3500 |    2
 sales     |     1 |   5000 |    1
 sales     |     4 |   4800 |    2
 sales     |     3 |   4800 |    2
(10 rows)
```

でもPostgreSQLでもALTER文を利用します。標準的なSQLですので、みなさんにも馴染みが深いのではないでしょうか。

このALTER文に関しても両DBで違いがあります。それは、MySQL 5.6のInnoDBにはオンラインALTER文(**表3**)があるということです。ALTER文の実行時はPostgreSQLではロックを取得するのに対し、MySQLは一部の条件ではALTER文の実行中でも更新できます。具体的には、

・カラムの追加、削除、並び替え
・カラムをNULL許可にする
・カラムをNOT NULLにする
・INDEXの追加、削除
・カラムのデフォルト値の設定
・オートインクリメント値の変更
・外部キー制約の追加、削除
・テーブルスペースファイルのデフラグ

は、ALTER文の実行中でもロックを取得しません。ただし、次の条件の場合はロックを取得します。

・カラムのデータ型変更
・全文検索用INDEXの追加
・プライマリキーの削除
・文字コードの変換、指定

このオンラインALTER文は、Webサービスのように常に更新が走るDBにはうれしい機能です。また、DBを24時間365日止められないようなミッションクリティカルなサービスでも、MySQLのALTER文は助かります。第4章に出てくるレプリケーションも含め、運用時にも止めずにメンテナンスができる方向性と、前章のような並列処理の強さがMySQLの強みです。そのため、ミッションクリティカルな大規模業務系サービスでもMySQLは使われています。

このような細かい違いでも、MySQLとPostgreSQLの方向性の違いが見受けられます。

関数の挙動の違い

date_formatとto_char

MySQLでもPostgreSQLでも、SQLで指定する関数が多くあります。たとえば、NULLの置き換えでよく使われるCOALESCE()はどちらのDBにもあり、同じような動作をします。しかし、一見同じように見えても挙動が違う関数があります。

ここでは日付を曜日に対応した数字に変える関数を見てみましょう。両DBで該当の関数は次のとおりです。

・MySQL：date_format(time, "%W")
・PostgreSQL：to_char(time, 'D')

どちらも一見すると、timeとformatの指定という同じような引数を取ります。しかし、返す値に違いがあります。timeに該当する日付が日曜日の場合、

・date_format(time, "%W")　→　0を返す
・to_char(time, 'D')　→　1を返す

となります。つまり、PostgreSQLからMySQLに置き換えるようなときに上記の関数に置き換えると、「アプリケーションはエラーなく動くが曜日がズレる」というバグを生むことになります。

このような関数の挙動の違いは多くあります。そのため、必ず公式ドキュメントで確認するようにしましょう。

CHECK制約

前述のように、SQLは通るが挙動が違うというものはほかにもあります。その中でハマりやすいのがCHECK制約です。CHECK制約はカラムの中に入るデータを制限できる機能です。

結論から言うと、MySQLにはこの機能があ

りません。PostgreSQLにはCHECK制約があり、CHECK(条件式)と書くことで条件式に該当しない場合はエラーにできます。たとえば、

```
CREATE TABLE table_name
   (column_name integer CHECK(column_name>
 < 10));
```

とすると、column_nameは9以下の数字のみ受け付けるようになります。

このSQLをMySQLで実行すると、なんと実行されてしまいます。しかし、MySQLにはCHECK制約はありませんので"CHECK制約の部分は無視されて"実行されます。これは、SQL標準に対応するための、MySQLの方針です。

同じくINDEXの昇降順についても、MySQLでは混在していてもすべて昇順となります。例を挙げますと、

```
CREATE INDEX
    test_index ON table_name (column1 >
ASC,column2 DESC);
```

というSQLは実行できますが、実際にはcolumn2のINDEXもASCで作成されます。

こういった、エラーにはならず、しかし予想とは違う挙動をする関数もあるので注意してください。SD

▼表3　ALTER文によるおもなオンラインステータス
（参考：https://dev.mysql.com/doc/refman/5.6/ja/innodb-create-index-overview.html）

操作	インプレース？	テーブルをコピー？	並列DMLを許可？	並列クエリを許可？	注意事項
CREATE INDEX、ADD INDEX	はい	いいえ	はい	はい	FULLTEXTインデックスにはいくつかの制限があります(1つ下の行を参照してください)。現在は、作成対象の同じインデックスも同じALTER TABLEステートメント内の前の句によって削除された場合、この操作はインプレースではありません(つまり、テーブルをコピーします)
ADD FULLTEXT INDEX	はい	いいえ	いいえ	はい	ユーザが指定したFTS_DOC_IDカラムがない限り、テーブルの最初のFULLTEXTインデックスの作成にはテーブルコピーが必要です。同じテーブル上の以降のFULLTEXTインデックスは、インプレースで作成できます
カラムのデフォルト値を設定する	はい	いいえ	はい	はい	データファイルではなく、.frmファイルのみを変更します
外部キー制約を追加する	はい	いいえ	はい	はい	テーブルのコピーを行わないようにするには、制約の作成中にforeign_key_checksを無効にします
外部キー制約を削除する	はい	いいえ	はい	はい	foreign_key_checksオプションを有効または無効にできます
カラムを追加する	はい	はい	はい	はい	自動インクリメントカラムを追加する場合は、並列DMLが許可されません。ALGORITHM=INPLACEは許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です
カラムを削除する	はい	はい	はい	はい	ALGORITHM=INPLACEは許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です
カラムNULLを作成する	はい	はい	はい	はい	ALGORITHM=INPLACEは許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です
カラムNOT NULLを作成する	はい	はい	はい	はい	SQL_MODEにstrict_all_tablesまたはstrict_all_tablesが含まれている場合は、カラムにNullが含まれていると操作は失敗します。ALGORITHM=INPLACEは許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です
主キーを追加する	はい	はい	はい	はい	ALGORITHM=INPLACEは許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。カラムをNOT NULLに変換する必要がある場合は、特定の状況ではALGORITHM=INPLACEが許可されません

第4章 機能性の違い

速度、パフォーマンスについて考える

Author 曽根 壮大（そね たけとも） Twitter @soudai1025

本章ではMySQL、PostgreSQLのより内部に焦点を当て、機能性の違いについて見ていきます。JOINの種類・アルゴリズムに対するサポート、INDEXの張り方、データ型の細かな挙動について、両DBには少しずつ差異があります。終盤では、Viewへの問い合わせにおける遅延の解決の仕方についても解説します。

JOIN

種類

リレーショナルデータベースとJOINは、切っても切れない関係です。JOINの種類は大きく分けて次のとおりです。

- CROSS JOIN
- INNER JOIN
- （LEFT or RIGHT）OUTER JOIN
- FULL OUTER JOIN

PostgreSQLはすべてに対応していますが、MySQLはFULL OUTER JOINをサポートしていません。しかし、MySQLでは“RIGHT JOINとLEFT JOINをUNIONする”ことでFULL OUTER JOINを実現できます。実際のSQLを例にしますと、

```
SELECT * FROM test1
  LEFT JOIN test2 ON test1.id = test2.id
UNION
SELECT * FROM test1
  RIGHT JOIN test2 ON test1.id = test2.id
```

とすることで、FULL OUTER JOINと等価になります。

しかし、FULL OUTER JOINはデータがとても大きくなり、パーフォーマンス遅延の要因になりやすいです。積極的に使うのではなく必要最低限に留めるようにしましょう。

OUTER JOINとINNER JOINは目的が違いますが、OUTER JOINは大は小を兼ねるため、WHERE句と合わせることでINNER JOINと同じ結果を得ることもできます。しかし、INNER JOINで解決できる場合はそちらのほうが高速に動作することが多いので、適切に使い分けるようにしましょう。これはMySQLもPostgreSQLでも同じです。

またJOINは処理コストが非常に高いSQL文の1つです。基本的にJOINするTABLE同士の掛け算だと思ってください。100行と100行のTABLEでJOINする場合は、10,000行のTABLEを参照しているようなものです。10,000行と10,000行の場合は100,000,000行です。ですのでJOINを行う際は、

- できるだけデータを小さくしてJOINをする
- 不要なJOINは避ける
- INDEXを利用したJOINをする

の3つに気をつけることが、ハイパフォーマンスを維持するコツです。

アルゴリズム

JOINのアルゴリズムは一般的に次の3種類です。

- Nested Loop Join
- Sort Merge Join

・Hash Join

MySQLにはNested Loop Joinしかなく、大きなデータのJOINにはとくに弱いので注意してください。PostgreSQLは3種類ともサポートしていますので、MySQLよりもJOINが得意と言えますが、それにも限度があります。前述のとおり大きなTABLEのJOINを複数回することはパフォーマンスの遅延原因の1つです。

この問題を話題にした際に、JOINを悪として非正規化を推奨する人をまれに見かけます。RDBの本来の責務は、データを正しく守ることです。非正規化はデータの整合性を崩す大きな原因の1つですので、正規化を前提としてハイパフォーマンスを目指すことを忘れないでください。2度目になりますが、JOINをする際は“小さく・最小限に・INDEXを利用して”が大切です。

INDEXの種類と違い

ハイパフォーマンスを実現するためには、INDEXは必要不可欠です。前述のJOINも、INDEXが利用できるかどうかで実行速度に大きな差が生まれます。ですのでINDEXを知ることは、リレーショナルデータベースにとって重要なことなのです。

ここではMySQLとPostgreSQLの、INDEXの違いについて説明します。まず、INDEXには複数の種類があります。そこで代表的な3種類を紹介します。

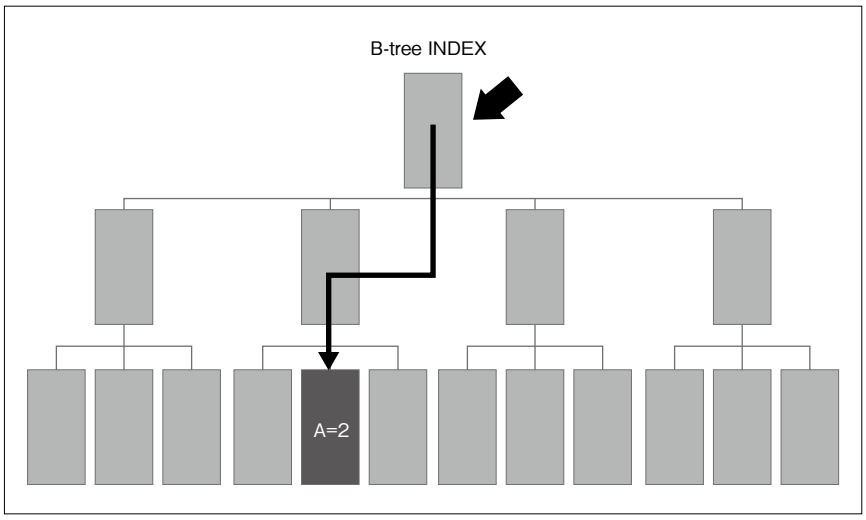
・B-tree
・GiST
・GIN

PostgreSQLは3つともサポートしていますが、MySQLはB-treeしかサポートしていません。ですが普段、ユニークな値に対してINDEXを利用することが多いと思いますので、B-treeをベースに考えても問題ないでしょう。そのため、ここではB-tree INDEXにフォーカスして説明します。

まずB-treeですが、**図1**のように木構造でINDEXを保持する手法を言います。そのため、ユニークな値に非常に強いのですが、同じ値が複数ある場合は順番に確認するため、逆に速度が遅延する可能性があります。

また、SQLの結果がTABLEの全体の10%～30%程度よりも大きな行数になる場合には、INDEXを使わずTABLEシーケンススキャン

▼図1　B-tree



が選ばれることが多いです。これは、INDEXを順番にそれぞれ精査するよりも、TABLE全体を順番に確認して必要な行数だけ取得するほうが速いためです。

以上の性質をしっかりと理解してINDEXを利用するようにしましょう。

また、PostgreSQLには式INDEXと呼ばれる、式の結果をINDEXとして保持する手法があります。これはフルTABLEスキャンの原因になりやすい関数を高速化するための有効な手段ですので、ぜひ覚えておいてください。

だからと言ってINDEXをいくつも張ってはINSERTやUPDATEの遅延の原因になります。これはインデックスショットガン[注1]と呼ばれるアンチパターンです。インデックスショットガンについてはMySQL、PostgreSQL関係なくDBMSで起こりやすいアンチパターンの1つですのでご注意ください。

INDEX設計はDB設計の1つです。とくに、実際にどのようなSQLが実行されるか意識してDB設計をすることが、無駄のないINDEXの利用につながります。また、INDEXが効かないSQLは遅延の原因になります。SQLを実行するときはぜひ実行計画を確認してみてください。

注1)『SQLアンチパターン』、Bill Karwin 著、和田卓人、和田省二 監訳、児島修 訳、オライリー・ジャパン、2013

column

MySQL 5.7とR-Tree INDEX

MySQL 5.7から、innoDBが位置情報などで利用される空間INDEXをサポートするようになりました。実は、MyISAMは以前から空間INDEXに使われるR-Tree INDEXをサポートしていました。R-Tree INDEXはPostgreSQLのGiST相当です。PostgreSQLは昔から空間INDEXをサポートしており、拡張であるPostGISは地図系の処理に利用されてきました。MySQLは、5.7から大幅にGIS機能を強化されています。しかし、PostGISに比べるとMySQL 5.7は経緯度データから面積を算出できないなど、機能不足がまだまだあります。

それでも、点での緯度経度の検索はPostGISをはるかに上回る速度で検索できるなどメリットも大きく、GIS(Geographic Information System:地理情報システム)においてMySQLが新たな選択肢になったと言えるでしょう。今後のMySQLに期待したい機能です。

レプリケーション

参照の負荷分散やクラスタ構成の実現などで、レプリケーションが使われるのは当たり前になっています。このレプリケーションについては、MySQLでは古くから実装されていました。

それに対して、PostgreSQLは9.0系からのサポートですので最近のことです。そのためMySQLのMHA(mysql-master-ha)のような確立されたクラスタ構成がなく、PostgreSQLクラスタは群雄割拠と言えるでしょう。

ただし、PostgreSQLも特定の条件下であればPG-REGやHAProxyなどのツールでクラスタ化ができます。ケースバイケースで選びましょう。

レプリケーションには更新情報を利用します。その更新情報の入ったログを、MySQLではバイナリログ、PostgreSQLではWALと呼びます。

MySQLでは、バイナリログのコネクション要求はスレーブから送られますが、コネクション後はマスタのBinlog Dumpスレッドから自発的にsendtoされます。

それに対して、PostgreSQLはマスタがスレーブに対してWALを直接pushします。また、PostgreSQLは本体がサポートするまで、slony-Iやpgpool-Ⅱなどクラスタリングするためのミドルウェアがありました。そのため現在でも、pgpool-Ⅱを使ったクラスタ構成が日常的に行われています。これは、pgpool-Ⅱを使うことでアクティブ×アクティブの構成

▼表1　MySQLとPostgreSQLのデータ型

DB／型	数値型	文字列型	日付／時刻型
MySQL	INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT、DECIMAL、FLOAT、DOUBLE	VARCHAR、CHAR、TEXT	DATE、DATETIME、TIMESTAMP、TIME、YEAR
PostgreSQL	bigint、integer、smallint、numericreal、double、boolean	character varying、character、text	timestamp、date、time、interval

が組めるなど、本体だけでは実現できない機能があるためです。pgpool-Ⅱは高性能なPostgreSQL専用のProxyやロードバランサと言えます。

しかし、関連するソフトウェアが増えれば増えるほど、障害点が増えるのは事実です。MySQLはシンプルな構成になる分、運用面や安定性では有利と言えます。現在Webにある大手サイトの大半がMySQLを利用している現状には、このような背景もあります。しかし今後、PostgreSQLはよりレプリケーションに力を入れていくのは間違いありません。クラスタリングについては両DBとも積極的に開発が進んでいるので、今後の楽しみな要素の1つです。

データ型

MySQLとPostgreSQLは、データ型のそれぞれを見ても特徴があります(**表1**)。普段使う型については、名称に多少の差異はありますが両者で用意されています。しかし、次から説明するように、詳細の挙動に違いがあります。

文字列型の挙動の違い

最初に、文字列型に注目します。次のとおり、文字列型には大きな違いがあります。

・MySQLはデフォルトで、大文字小文字を区別しないCollationを選ぶ
・PostgreSQLは大文字小文字を区別する

この仕様の違いは、ユーザにとって大きな違いです。

MySQLは文字列型にbinary属性を付けることで大文字小文字を区別できます。Collation(設定)依存ですが、デフォルトでは大文字小文字を区別しないので注意が必要です。また、設定はカラム単位で指定できるため、仕様に合わせて柔軟に設定できます。

標準として、PostgreSQLの文字列型自体に文字列の大文字小文字を区別しない機能はありません。その代わり、citextモジュールを追加することで大文字小文字の区別がない文字列型を提供します。citext型は値の比較の際、基本的に内部でlowerを呼び出します。そのため、INDEXには基本的に利用せず、PRIMARY KEYやUNIQUE KEYに利用したときは、INDEXは大文字小文字を区別します。完璧ではないのでご注意ください。

このほかの方法では、PostgreSQLの機能であるユーザ独自型を利用して専用の型を作成するか、大文字小文字を区別せずに検索するILIKEなどを利用して検索条件側で同様の仕様にする必要があります。

ちなみに、筆者はPostgreSQLで大文字小文字を区別せずにアカウントIDなどを区別する必要がある場合は、DBにすべて小文字で保存します。その状態で、

```
SELECT * FROM users WHERE account_id = ➢
LOWER(アカウントID);
```

として、すべてを小文字にして検索するようにしています。これは、citext型を導入したときの動作と同じになります。

また、文字の結合方法にも違いがあります。両DBともにCONCAT()という関数で文字結

合できますが、PostgreSQLはOracleDBのように、「||」でも文字結合できます。MySQLでは、「||」はデフォルトではOR演算子を指すため使えません。ただし、MySQLにはSQLモードという動作を設定する機能があり、そこで次の設定を行えば「||」での文字結合ができます。

```
SET sql_mode= CONCAT_WS(',', @@sql_mode,➥
'PIPES_AS_CONCAT');
```

以上のとおり、文字結合や検索に違いがあり、とくにMySQLは環境や設定によっても動作が違うので注意が必要です。

日付／日時型とtimezone

世界規模で使うようなアプリケーションの場合、timezoneは意識すべき重要な要素の1つです。

timezoneの保存について、MySQLではできませんが、PostgreSQLでは、timezoneを保存する日付／日時型と保存しない日付／日時型があります。timezoneを保存する場合はUTC(協定世界時間)でDBには登録し、timezoneに合わせて表示する時間を調節します。

timezoneの変更について、MySQLではDB自体でtimezoneを持つので次のような方法などで変更できます。

```
SET time_zone = timezone;
```

MySQLでは、timezoneを設定するとTIMESTAMP型は影響を受けますが、DATETIME型は影響を受けません。またMySQLには、0000-00-00 00:00:00という実在しない時間を登録できます。PostgreSQLはこのような時間は登録できません。PostgreSQLはその代わりに、最小を表す- INFINITEと最大を表すINFINITEがあります。もし検索条件などで必要な場合は、こちらを利用しましょう。

PostgreSQLの独自型

PostgreSQLには、ほかのRDBにはない独自の型が多くあります。たとえば、P.38のコラム「MySQL 5.7とR-Tree INDEX」で説明したGISに使われる幾何学型などがその例です。そのほかにも、配列型や第6章で出てくるJSON型などがあります。ここでは筆者が実装時に非常に助けられた2つの型を紹介します。

ネットワークアドレス型

ネットワークアドレス型は、その名のとおりネットワークアドレスを保存する型です。この型は、アドレス単体も、サブネットマスクで指定されたネットワーク本体も登録できます。具体的には、「IPv4、およびIPv6ネットワーク」を登録するinet型と、「IPv4もしくはIPv6ホスト、およびネットワーク」を登録するcidr型があります。さらに、MACアドレスを登録できるmacaddr型もあります。

この型を利用するメリットは3つあります。

1つめは、保存されるIPアドレスやネットワークの値が正しいことを担保できることです。たとえば、192.168.0.255/24のような存在しないIPアドレスは登録できません。また、サブネットマスクで指定した範囲外のIPアドレスも登録できません。ネットワークアドレス型は、サブネットマスクも含む場合の複雑なIPアドレスのバリデーションを、正しく行ってくれるのです。

2つめはソートと検索です。IPアドレスを文字列としてソートした場合、10.2.1.1と10.11.1.1であれば後者が優先されてしまいます。また、検索で192.168.1/24の範囲内のIPアドレスを検索する場合なども難しい処理が必要です。しかし、PostgreSQLでは、内包は次のような構文で表現できます。

```
SELECT inet '192.168.1.5' << inet ➥
'192.168.1/24';
```

この構文で検索することで、指定の範囲内のIPアドレスを検索できます。この機能は非常に便利ですので、覚えておいて損はありません。

そして3つめは、IPv6対応がされていることです。たとえば、IPv6のバリデーションもたいへんですが、IPv6は128bitまであるため多くの環境で桁数の問題からintにできません。実際に、PHPではintにできないためip2long()はIPv4しかありません。しかし、PostgreSQLはIPv6対応されているため、既存のIPにインクリメントして次のIPアドレスがほしいときは次のSQLで対応できます。

```
postgres=# SELECT inet '2001:0db8:bd05:
01d2:288a:1fc0:0001:10ee' + 1 AS net;
                 net
-------------------------------------
 2001:db8:bd05:1d2:288a:1fc0:1:10ef
(1 rows)
```

プログラム側で処理することがたいへんなことも、PostgreSQLの適切な型を使うことで解決する例です。

範囲型

範囲型は、PostgreSQL 9.2から入った機能です。次の型に対して範囲を指定できるようになります。

- int4range：integerの範囲
- int8range：bigintの範囲
- numrange：numericの範囲
- tsrange：timestamp without time zoneの範囲
- tstzrange：timestamp with time zoneの範囲
- daterange：dateの範囲

範囲を指定できるというのは、fromとtoをそれぞれ、または片方のみ登録できるというものです。たとえば次のように利用できます。

・含有

```
postgres=# SELECT int4range(10, 20) @>
11 AS result;
 result
--------
 t
(1 rows)

postgres=# SELECT int4range(10, 20) @>
3 AS result;
 result
--------
 f
(1 rows)
```

・重なり

```
postgres=# SELECT numrange(11.1, 22.2)
&& numrange(20.0, 30.0) AS result;
 result
----------
 t
(1 rows)

postgres=# SELECT numrange(11.1, 22.2)
&& numrange(23.0, 30.0) AS result;
 result
----------
 f
(1 rows)
```

・上限の取得

```
postgres=# SELECT upper(int8range(15,
25));
 upper
-------
    25
(1 rows)
```

・共通部分の計算

```
postgres=# SELECT int4range(10, 20) *
int4range(15, 25) AS result;
 result
---------
 [15,20)
(1 rows)
```

・差分部分の計算

```
postgres=# SELECT int8range(5,15) -
int8range(10,20) AS result;
 result
--------
 [5,10)
(1 rows)
```

・範囲は空か

```
postgres=# SELECT isempty(numrange(0, 0));
 isempty
---------
 t
(1 rows)

postgres=# SELECT isempty(numrange(1, 5));
 isempty
---------
 f
(1 rows)
```

MySQLで範囲を表現する場合は、toとendをそれぞれ別のカラムとして登録することで表現します。しかし、その場合SQLでは重なりの表現や共通部分の計算などを柔軟に行うことはできません。

このように、RDBは点でデータを保存するため、連続性の必要なデータは苦手な傾向があります。しかし、範囲型を使うことで検索面において柔軟な検索を可能にします。実際に筆者は、例のような場合以外でも商品の有効期限やお知らせの公開時間などの範囲で日時を指定する場合に、よく利用します。アルゴリズムを明確に簡潔にすることはバグの抑制にもつながるので、みなさんもぜひ使ってみてください。

Viewの問い合わせ遅延をいかに解決するか

SQLが複雑になってきたときなどに、計算結果をViewとして、新たにTABLE化した経験があるのではないでしょうか。Viewの進化はMySQLとPostgreSQLの両DBともすばらしく、今ではSimpleなViewに対しては更新処理などもできるようになっています。

Viewは問い合わせがあるたびに設定されたSQLを実行します。そのため、実行されるSQLが速いときは良いのですが、データが大きくなったなどの理由から遅延し始めると、View全体が影響を受けます。

その問題を解決するための機能が、Materialized View(以下マテビュー)です。マテビューは実行されるViewの結果を実体化することで、キャッシュのような動きをします。実行結果を実体化するので表領域を消費しますが、INDEXを新たに張ることもできます。ただし、元となるTABLEが更新された場合は、マテビューをリフレッシュして更新する必要があります。つまり、実行結果のTABLEが存在するため、大きなTABLEを参照することで遅延していたSQLの問題を解決できます。

しかし、残念ながらMySQLにはマテビューがありません。PostgreSQLもマテビューは9.3からの対応であり、最近実装された機能です。では、それまではどうやってこの問題に対応していたのでしょうか。

それには、Temporary Tableを使っていました。Temporary Tableはその名のとおり、同じSESSION内でのみ有効な、一時的なTABLEのことを指します。そのため、SESSIONを抜けると自動的にDROPされるので、永続的には保存できません。マテビューとはこの点で大きく違うので注意が必要です。

また、MySQLは相関サブクエリが高速に動作しないため、サブクエリではなくTemporary Tableに一時的に保存して再利用することがたびたび見受けられます。これは、Temporary Tableが小さいTABLEであれば非常に有効な手段です。ただし、MySQLのTemporary Tableは自己結合と自分自身をUNIONするようなケースでは使えませんのでご注意ください。SD

第5章 拡張性の違い

一歩進んだ使い方を知る

Author 曽根 壮大(そね たけとも) Twitter @soudai1025

DBの機能や速度を高めたいとき、MySQLではストレージエンジンとプラグインを使って、PostgreSQLではForeign Data Wrappers(FDW)とエクステンションを使って拡張を行います。本章では、それらの概要について解説します。

拡張方法にも違いあり

サーバアーキテクチャ、ストレージアーキテクチャの違いによって、MySQLには速度的に有利な面があることがわかりました。それに対して、SQLや機能面ではそれぞれ特徴がありながらも、PostgreSQLがより多機能であることがわかりました。ただし、機能面や速度面においては拡張を行うことで不利な点を補えるケースがあります。そこで、この章では拡張について説明します。

MySQL——ストレージエンジンとプラグイン

MySQLの拡張と言えば、特徴的な機能がストレージエンジンです。MySQL 5.5からInnoDBがデフォルトになったことで、一般的にストレージエンジンにはInnoDBが使われているかと思います。ほかに有名なストレージエンジンと言えばMySIAMですが、トランザクションがないため今後は積極的な理由で使う機会はないと思います。

MySQLの全体像は**図1**のとおりです。最下層にあるストレージエンジンは、MySQLの“データに対するアプローチ”を決めるものと言っても過言ではありません。そのため、自由に設計・拡張できますが、それには多くの実装が必要になります。

そこで普段はInnoDBをストレージエンジンとして使い、少しの拡張だけしたい場合にはプラグインを利用します。MySQLのプラグインAPIは現在8つ公開されています。

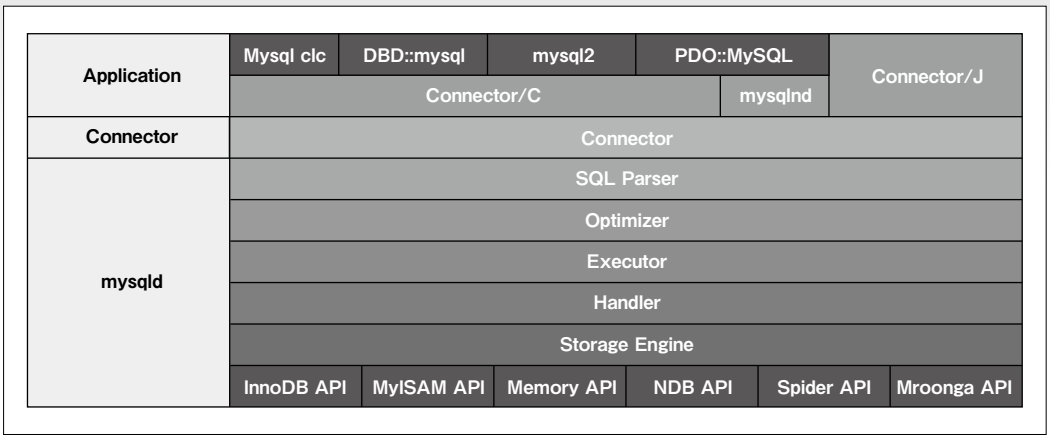
- Storage engines
- Full-text parsers
- Daemons
- INFORMATION_SCHEMA tables
- Semisynchronous replication
- Auditing
- Authentication
- Password validation and strength checking

これらAPIを使えば、ストレージエンジンのようにすべてのルールを制約することこそできませんが、いろいろなアプローチが採れます。たとえばHandler Socket Plugin(**図2**)を導入すれば、高速に、NoSQLのようにデータを取得できるようになります。

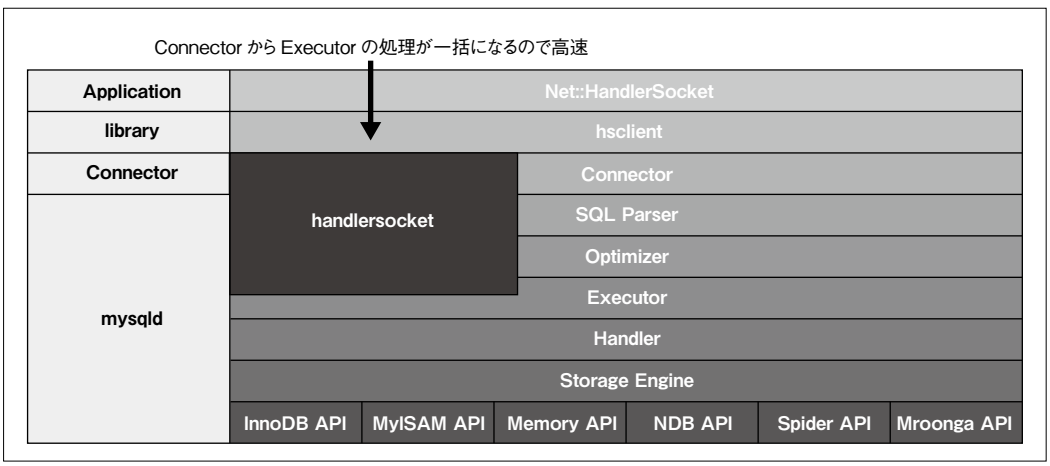
PostgreSQL——FDWとエクステンション

先ほどはMySQLの拡張について、ストレージエンジンとプラグインの説明をしました。それに対してPostgreSQLの拡張には、Foreign Data Wrappers(以下FDW)と呼ばれる外部データラッパーとエクステンションを使います(**表1**)。FDWがMySQLにおけるストレージエンジン、エクステンションがMySQLにおけるプラグインという位置づけになります。

▼図1　MySQLの全体像



▼図2　Handler Socket Plugin



まずFDWについてですが、こちらはPostgreSQLのテーブルにアクセスするのと同じように、別のデータ層へアクセスできるようにするしくみです。データ層はFDWのルールに沿って定義してあればどんなデータでもかまいません。つまり、MySQLのテーブルやOracleDBのテーブルにもアクセスできます。また、RDBに縛られることはなく、CSVファイルやIMAP、Web APIの結果などにも柔軟に対応できます。もちろん、NoSQLのようにデータ構造が違うデータでも問題ありません。**図3**に、MySQLにアクセスする際の例を示します。

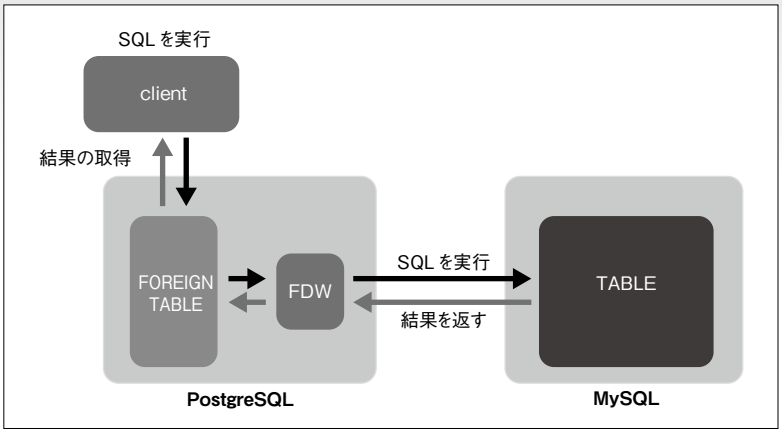
エクステンションはPostgreSQLの拡張です。PostgreSQLのバックエンドレイヤであれば自由にアクセスできます。このエクステンションには、PostgreSQL本体が提供し、PostgreSQLのソースコードに同梱されているcontrib[注1]モジュールと呼ばれるものがあります。その中には、暗号化機能を追加するpgcryptoなど、豊富な種類が用意されています。

現在インストールしているEXTENSIONは次のSQLで確認できます。

```
select * FROM pg_available_extensions;
```

注1）追加で提供されるモジュールの一覧
URL http://www.postgresql.jp/document/9.4/html/contrib.html

▼図3　Foreign Data Wrappersのしくみ



▼表1　公開されているおもなFDWの一覧（参考：https://wiki.postgresql.org/wiki/Foreign_data_wrappers）

SQL Databases Wrappers
postgres
oracle
mysql
SQLite
Firebird
MS SQL Server
odbc
jdbc
NoSQL Databases Wrappers
Cassandra
couchdb
MonetDB
mongo
redis
Neo4j
Tycoon
DynamoDB
Hadoop
Elastic Search

File Wrappers
CSV
JSON
XML
ZIP Files
TAR Files
pg_dump
Others
git
RSS
ldap
IMAP
s3
www
OS
Google Spreadsheets
Google
Twitter
Facebook

PostgreSQL本体以外が開発しているエクステンションも数多く、OSSとして公開されています。たとえば、全文検索機能を追加するpg_bigmなどが便利です。

◆　◆　◆

MySQLもPostgreSQLも、標準的な機能も重要ですが、拡張によってさらにデータベースの世界が広がります。その拡張機能の多くは、OSSとして公開されています。いろいろと調べてみると、困ったときや新たな設計をするときの引き出しが増えます。また、自分で拡張機能を実装をすると、内部構造を知る良い機会になります。ソフトウェアの内部構造を知ることはチューニングや設計時に非常に役に立つ知識です。ぜひ、拡張機能を自分なりに作ってみてください。SD

第6章 MySQL 5.7とPostgreSQL 9.5 新機能比較

JSON対応をはじめとした期待の新機能

Author 曽根 壮大（そね たけとも） Twitter @soudai1025

MySQL 5.7は2015年10月19日にリリースされました。 PostgreSQL 9.5は11月23日現在beta2がリリースされており、近日中にリリースが予想されます(皆さんが読まれているころにはリリースされているかもしれません)。そこで本章では執筆時点での最新版の情報を紹介します。

MySQL 5.7期待の新機能

MySQL 5.7では大幅な性能向上と新機能追加が行われ、おもな改善点は200項目近くに上ります。新機能についてはMySQLコミュニティの有志によって日本語に訳されています[注1]。その中で筆者が注目している新機能は次の2つです。

・1テーブル1イベントに複数のトリガーを設定可能
・データ型としてJSON型のサポート

それぞれ見ていきましょう。

1テーブル1イベントに複数のトリガーを設定可能

MySQL 5.6までは、1テーブル1イベントに1つのトリガーしか設定できませんでした。これにより、どうしてもDB側で実装できずにアプリケーション化したり、ストアドプロシージャを作成したりする必要がありました。その問題が5.7から解決しました。

ただし、トリガーの対象となるイベントが発生した際に行われる処理があったとして、それをサポートしているのは、1行ごとに実行されるオプションの`FOR EACH ROW`だけです。そのためSQL文ごとに実行させる`FOR EACH STATEMENT`を指定できません。この違いは1回のUPDATE文で複数の行に影響を与えるような場面などで、その処理に大きな差が発生します。MySQLのトリガーは、対象となるイベントが発生した場合に、1行ごとに1回実行されるので注意が必要です。

データ型としてJSON型をサポート

PostgreSQLでは9.2からサポートされているJSON型ですが、ついにMySQLでもサポートされました。これによりドキュメント指向でかつスキーマレスな設計を実現できます。MySQLのJSON型はPostgreSQLのJSONB型相当で、バイナリ形式で保存されます。バイナリで保存することで効率よくデータが保存されます。またJSON型専用の組込み関数が追加され、保存・検索・更新・操作することが可能になりました。組込み関数については**表1**のとおりです。

MySQL 5.7でのJSON対応について、参考までに`JSON_SEARCH()`の例を**図1**で挙げます。それでは公式ドキュメントで紹介している例を見てみましょう。

図1のようにあいまい検索も含め、JSONの中を検索して該当のKeyを返してくれます。これにより柔軟な設計が可能になり、RDBの可能性が広がります。

またMySQL 5.7からの新機能の1つであるGenerated Columnsを利用してINDEXを利用した検索もできます。これにより、柔軟で高速なテーブル設計を可能にしました。もちろんJSON型には外部Key制約が使えないなど

注1） URL https://yakst.com/ja/posts/3037

▼表1　MySQL 5.7でサポートされたJSON型専用の組込み関数
（参考：https://dev.mysql.com/doc/refman/5.7/en/json-function-reference.html）

関数名	説明
JSON_ARRAY_APPEND()	JSONドキュメントの最後に要素を追加してその結果を返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_ARRAY_INSERT()	JSON配列に要素を追加して更新後のJSONドキュメントを返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_ARRAY()	空または値入りのJSONの配列を返す
->	JSON_EXTRACT()と同じ挙動をする。カラム->JSON階層と書くと結果を表示するカラムに配列から該当する値を取り出して返す
JSON_CONTAINS_PATH()	JSONドキュメントに指定した値が含まれているかどうかチェックし、0か1を返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でないとエラーを返す。JSON_CONTAINS()との違いは引数を複数指定できるかどうかであり、oneを指定すると最低1つ含まれていれば1を返し、allを指定すると複数指定した階層が含まれていれば1を返す
JSON_CONTAINS()	JSONドキュメントに指定した値が含まれているかどうかチェックし、0か1を返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_DEPTH()	JSONドキュメントの階層の深さを返す。引数にNULLが指定されたときNULLを返す。空の配列やオブジェクトやスカラ変数のとき、1を返す。JSONとして正しい形式ではないとエラーを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_EXTRACT()	JSONドキュメントの何番目を取り出すかを指定し、そこに格納されている値を返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でないとエラーを返す
JSON_INSERT()	JSONドキュメントへ値を挿入する。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_KEYS()	JSONドキュメントから配列の一番上の階層のキーを返す。階層を未指定時は一番上の階層から、指定時はネストした構造であるとき、その階層の一番上のキーを返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_LENGTH()	JSONドキュメントの要素数を返す。要素として数えられる単位としては変数、配列、オブジェクト。ネスト化された配列やオブジェクトはカウントされない。引数にNULLが指定されたときまたは指定されたパスがドキュメントに見つからないとき、NULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_MERGE()	2つまたはそれ以上のJSONドキュメントをマージして返す。隣接した配列は1つの配列に、隣接したオブジェクトは1つのオブジェクトにマージされる。変数は配列としてオートラップされ、配列としてマージされる。隣接した配列とオブジェクトはオブジェクトを配列としてオートラップし、2つの配列をマージした扱いとなる。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式でない場合、エラーを返す
JSON_OBJECT()	キーと値のペアを引数として、JSONオブジェクトを作る。キーがNULLのときや引数が不正なとき、エラーを返す
JSON_QUOTE()	JSONドキュメントの引数をダブルクオテーションでくくる。'"hoge"'が指定されたときはエスケープしてutf8mb4のstringを返す。JSONstringとして正しい文字列をJSONドキュメント内で作るときに使用する。引数にNULLが指定されたときNULLを返す
JSON_REMOVE()	JSONドキュメントから、指定した階層のデータを消して返す。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す

（次ページへ続く）

(前ページから続く)

関数名	説明
JSON_REPLACE()	JSONドキュメントから、指定した階層に置換する値を入れて返す。指定された階層や値のペアがJSONドキュメントに存在しない場合はとくに何もされず無視される。引数にNULLが指定されたときNULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_SEARCH()	JSONドキュメントから指定した文字列を含む階層を返す。ネストされた構造の場合にoneを引数として指定した場合は一番上の階層のみから探し出し、allを指定した場合はネストの中まで探しに行く。searchstrには%やがLIKEと同じように使える。引数のどれかがNULLのときや階層が存在しないなどした場合、NULLを返す。JSONとして正しい形式や階層でないとエラーを返す
JSON_SET()	JSONドキュメントへインサートまたはアップデートを行い結果を返す。引数のどれかがNULLのときや階層が存在しないなどした場合、NULLを返す。JSONとして正しい形式や階層でなかったり、*や**などワイルドカードが含まれているとエラーを返す
JSON_TYPE()	引数に指定したJSONの値の型をutf8mb4 stringで返す
JSON_UNQUOTE()	JSONの値のクオートを外してutf8mb4 stringで返す。クオートがダブルクオテーションで括られ、かつJSONのストリングとして正しくない場合、エラーが返ってくる。エスケープシーケンスとして存在するものに限り、エスケープするためのバックスラッシュが認識されてしまう(¥bなど)
JSON_VALID()	JSONドキュメントとして正しいかどうか判別し、0か1で返す。引数にNULLが指定されたときNULLを返す

▼図1　公式ドキュメントにおけるMySQL 5.7のJSON対応例

```
MySQLの独自機能のユーザ定義変数を利用して@jにJSONデータを設定する。これはテーブルの代わりになる
mysql> SET @j = '["abc", [{"k": "10"}, "def"], {"x":"abc"}, {"y":"bcd"}]';
1階層目を指定しているので"abc", [{"k": "10"}, "def"]がマッチして配列のkeyである0が返る
mysql> SELECT JSON_SEARCH(@j, 'one', 'abc');

+--------------------------------+
| JSON_SEARCH(@j, 'one', 'abc') |
+--------------------------------+
| "$[0]"                         |
+--------------------------------+
すべての階層を指定しているので、さらに2階層目の{"x":"abc"}がマッチしている
mysql> SELECT JSON_SEARCH(@j, 'all', 'abc');
+--------------------------------+
| JSON_SEARCH(@j, 'all', 'abc') |
+--------------------------------+
| ["$[0]", "$[2].x"]             |
+--------------------------------+

何も該当しない場合はnullを返す
mysql> SELECT JSON_SEARCH(@j, 'all', 'ghi');
+--------------------------------+
| JSON_SEARCH(@j, 'all', 'ghi') |
+--------------------------------+
| NULL                           |
+--------------------------------+

あいまい検索
mysql> SELECT JSON_SEARCH(@j, 'all', '%a%');
+--------------------------------+
| JSON_SEARCH(@j, 'all', '%a%') |
+--------------------------------+
| ["$[0]", "$[2].x"]             |
+--------------------------------+
```

の課題はPostgreSQLと同様にあります。ですが使い方しだいで選択肢が広がる機能なだけに、今後も期待が膨らむ機能です。

PostgreSQL 9.5 期待の新機能

現在beta2をリリースされているPostgreSQL 9.5ですが、こちらも目玉機能が多数あります。PostgreSQL 9.5は開発時や運用時に便利になる機能が多数追加されています。前述したINSERT ON CONFLICT以外にも次の機能に注目しています。

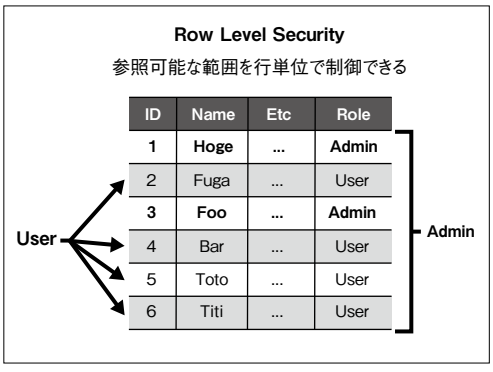
・Row Level Security
・IMPORT FOREIGN SCHEMA

Row Level Security

Row Level Security(以降RLS)は、ユーザが操作・閲覧できる行を指定できる機能です(**図2**)。tableよりも細かい行単位のアクセス制限をすることでよりセキュアな運用が可能になります。今までこのようなケースではViewを使われることが多くありました。PostgreSQLはViewだけそれぞれ別に参照権限を分けることができます。ただし、行単位ではできないためケースごとにViewを作る必要がありました。そこでRLSを使うことで特定のtableの特定の行だけをアクセスするような制限ができるようになったのです。実際の例を**図3**に示します。

図3のように非常に簡単にアクセス制限ができます。マイナンバーをはじめ、個人情報保護について厳しい昨今、これは重要な機能です。

▼図2　Row Level Securityの概念図



IMPORT FOREIGN SCHEMA

PostgreSQLは拡張機能の第5章で説明したとおり、FDWという外部tableを利用する機能があります。しかしこの機能は、参照元で参照先と同様のtableを定義する必要がありました。この作業をPostgreSQL to PostgreSQLの場合は省略できるようになりました。これによりFDWの運用がより一層楽になりました。またFDWは参照先のtableを元に継承もできるようになり、PostgreSQL 9.5では大きな機能強化対象になっています。

このほかにもMySQLのJSON型と同等機能であるJSONB型に対する強化や関数の追加も行われています。正式版のリリースが待ち遠しいところです。SD

▼図3　RLSの使用例

```
CREATE POLICY emp_foo_policy
    ON emp FOR SELECT TO foo USING (role = 'User');

ALTER TABLE emp ENABLE ROW LEVEL SECURITY; SELECT * FROM tmp;
```

第7章 MySQLとPostgreSQLコミュニティの違い

仲間作りと情報交換の場

Author 曽根 壮大（そね たけとも） Twitter @soudai1025

ソフトウェア開発にあたり、コミュニティの存在は欠かすことができません。とくにデータベースは、一度導入すると、何年もメンテナンスを続けることになります。エラーが起きたときの情報交換の場として重要な場です。本章では、まずMySQLとPostgreSQLのソフトウェアの開発元の違いについて説明し、それぞれのコミュニティの特徴について紹介します。

MySQLとコミュニティ

MySQLは1995年にMySQL AB社が開発し、2008年にMySQL AB社をSun Microsystems社が買収しました。買収後は同社が開発・管理していましたが、2010年にSun Microsystems社はOracle社に買収され、それ以降今に至るまでMySQLはOracle社が管理することになりました。そのためMySQLの開発主体はあくまで企業であり、MySQL 5.6も5.7もOracle社が開発しています。社外プログラマはオープンになっているソースコードをもとにパッチをOracle社に提供することで、開発に参加する形になります。そのためコミッターにはOracle社員である必要があります。しかし所定の手続きは必要ですが、コントリビュータには社外のエンジニアもいます。また、MySQLへのパッチはFacebookをはじめとする多くのエンジニアや企業から提供されマージされています。このようにMySQL自体はOracle社の管理下にあるものですが、MySQLユーザのためのコミュニティも存在します。国内では、次の2つの団体があります。

・日本MySQLユーザ会（以下MyNA）
・MySQL Casual

MyNA[注1]はメーリングリストに登録することで参加でき、メーリングリストを中心に活動されています。とみたまさひろ氏が代表を務め2000年から活動しています。

メーリングリストでは、

・MySQLにかかわるイベントやニュースの告知
・MySQLにかかわる質疑応答

などが行われており、多くのユーザが参加しています。

もう一方のMySQL Casual[注2]は比較的新しいコミュニティで若い世代を中心に活動がされています。MyNAとの違いはSlackを中心に活動をしているところです。参加は注2のURLからメールを送ることで招待状が届き、Slackのチャンネルに参加することできます。

双方の違いは、

・MyNAは比較的固定メンバーが少なく、業務としてMySQLを使っている人たちが多い
・MyNAのイベントの登壇メンバーは割とクローズドに決まる
・MySQL Casualは固定メンバーが多く、彼らはWeb界隈で名が知れている人が多い
・MySQL Casualのイベント登壇は公募制で、内容は運用に関することが多い

注1） URL http://www.mysql.gr.jp/
注2） URL http://mysql-casual-slackin.herokuapp.com/

などがあります。

PostgreSQLとコミュニティ

PostgreSQLは、コミュニティにより開発されているコミュニティ主体のOSSです。コミュニティの中からコミッターが選任され、コミッター以外はパッチをコミュニティに送り、マージしてもらいます。また年に一度PGConがオタワで行われており、コミッターが一堂に会する場所となっています。日本人にもコミッターはおり、若い人ではFujii Masao(@fujii_masao)さんが活動しています。

日本のコミュニティとしては、日本PostgreSQLユーザ会(以下 JPUG)があります。参加はメーリングリストまたはSlackの参加登録サイト[注3]から行えます。JPUGはMyNAとは違いNPO法人として組織化されて活動しています。協賛企業から協賛金の提供を受けて活動しており、コミュニティ主催のイベントなどを積極的に開催しています。またJPUG主催以外のイベントにもスピーカーの交通費や会場費の提供を行うなどソフトウェア業界全体へのサポートを行っています。

筆者はMyNA、JPUGともに参加しており、JPUGでは中国支部長として理事も務めています。企業が開発することという制約のあるMySQLがコミュニティはあえて組織化せず、ユーザ会は自由なコミュニティを目指しているのに対し、コミュニティが主体のPostgreSQLはユーザ会は組織化して厳格に運営を目指しているのは対局的で面白い関係です。

また一見MySQLとPostgreSQLはライバル関係に思われるためユーザ会も敵対しそうですが非常にフレンドリーな関係をしています。実際にMyNAとJPUGで合同勉強会[注4]が開催されています。

興味がある方は、ぜひコミュニティに参加して最新情報をチェックしてみてください。また、両コミュニティのメーリングリストやSlackに参加しているメンバーは人助けや議論に飢えています。ですのでコミュニティに参加して、気軽に質問や疑問を投稿してみてください。コミュニティの方々が喜々として回答してくれます。

公式ドキュメントの活用のしかた

MySQLもPostgreSQLもWeb上で公式ドキュメントが提供されています。

・MySQL 公式ドキュメント
・PostgreSQL 公式ドキュメント

膨大な量のマニュアルがありますが、MySQLはOracle、PostgreSQLはJPUGが主導となって提供されています。ただしMySQL 5.7は英語版[注5]しかなく、日本語版は提供されていません。PostgreSQLはJPUGが翻訳を行っており、PostgreSQL 9.4まで公開されています。現在は田中 健一朗氏(文書・書籍関連分科会座長)を中心にPostgreSQL公式ドキュメントの翻訳活動が行われており、GitHub[注6]で管理されています。

GitHubで管理されているため誰でもプルリクエストを送ることができます。また直接プルリクエストが難しい場合はissuesを作ることでコミュニティに依頼できます。issuesを始めとしたやりとりは日本語で行われているため英語が苦手な方でも安心です。誤字・脱字・誤訳・誤変換などを見つけた場合は積極的にプルリクエストやissuesを作ってみてください。

公式ドキュメントの貢献はコミュニティに対する重要な貢献です。コードが書けなくて

注3) メーリングリスト(URL https://ml.postgresql.jp/mailman/listinfo/jpug-users)、slack(URL https://postgresql-hackers-jp.herokuapp.com/)
注4) 第1回 MyNA・JPUG合同勉強会 URL http://dbstudychugoku.github.io/events/tokyo-001.html
注5) URL https://dev.mysql.com/doc/relnotes/mysql/5.7/en/
注6) URL https://github.com/pgsql-jp/jpug-doc

も気軽に始めることができる大きな貢献の1つですので、これを機会にみなさんもぜひ参加してみてください。

最後に

ここまでMySQLとPostgreSQLの違いを紹介しました。次にまとめます。

・MySQL
- □ スピード重視
- □ シンプルだが機能が少ない
- □ レプリケーションやシャーディングの知見が多い

・PostgreSQL
- □ 機能性と堅牢性重視
- □ 高機能だが限界値がMySQLよりも低い
- □ 設計や仕様の問題などを機能でカバーしやすい

さらに、次のように利用シーンをもう少し細かく分けて考えます。

・MySQL
- □ 大量のUPDATEが走る場合
- □ 単純なクエリを大量に捌く必要がある場合
- □ レプリケーションやパーティションを利用した負荷分散が必要な場合
- □ 大規模データ・高負荷サービス向き

・PostgreSQL
- □ 仕様変更が多い場合
- □ 集計や分析が必要な場合
- □ データの整合性や保護が重要な場合
- □ 仕様変更の多いスタートアップや集計が複雑なサービス向き

と筆者は考えてます。

つまりWeb系も業務系も関係なく、データの利用方法でソフトウェアを選んだほうがよいでしょう。その理由としては、実際にWebサービスもデータが複雑化していることが挙げられます。さらに業務系でもビックデータと呼ばれるサイズのデータを扱うようになっているからです。

今後もMySQLとPostgreSQLはライバルではなく、お互いの方向性の違いからサポートし合う関係として進んでいくと、筆者は思っています。この住み分けがある限り、どちらのソフトウェアも形は、変われど続いていきます。そのため、すべてのケースで同じソフトウェアを使うのではなく、ケースバイケースで選択していくことが非常に大切です。

また、エンジニアとして複数の選択肢を知ることで、自分の中の評価軸が増えます。その結果、選択肢から対応できる発想が広がり、エンジニアリングの幅が広がり、しかも設計力も上がります。

ソフトウェアの選択肢が増えることは仕事の幅も広がることにつながります。現在のDB業界ではMySQLもPostgreSQLも詳しいと言える人はまだまだ多くありません。その中でも今後も間違いなくOSSの利用は増えるため、仕事の幅を広げることは生き残り戦略としても重要な手法の1つです。

ところで筆者の持論ですが、データベースの寿命はコードより長いと思っています。現在のソフトウェア開発でリレーショナルデータベースは必須です。そのなかでサーバやコードのリプレースを行うことがあってもデータベース内に構築されたデータをリプレースすることはほとんどありません。ですから5年、10年と一緒に付き合い続けねばならないデータベースの知識は重要なのです。しかしながら、二兎追うものは一兎をも得ずという諺(ことわざ)にもあるとおり、変化の激しいIT業界で知識を増やすことはたいへんです。しかも知るだけでなく、技術として研鑽せねばなりません。そこでオットー・フォン・ビスマルクの次の言葉を引用します。

「愚者は経験に学び、賢者は歴史に学ぶ」

この考え方をうまく活用してください。

歴史は、過去の事例や周囲の経験談です。コミュニティにはそれがたくさんあります。これらを有効活用し、さらに情報交換することで他者の歴史を学びましょう。より効率的に知識の研鑽が行えます。

本章を読んで興味を持った方は、DBのコミュニティにぜひ参加しましょう。MySQLのコミュニティもPostgreSQLのコミュニティも暖かく迎え入れてくれます。皆さんの参加を心からお待ちしています。

最後になりましたが、本原稿の執筆に際して、レビュアを快く引き受け、そして多くの指摘をくださった @yoku0825、@nuko_yokohama、@kasa_zip、@daiti0804、@tyokkin3rd の皆さんに感謝します。SD

column

筆者がコミュニティを勧める理由

OSSにはユーザ主導のコミュニティが必ずと言っていいほど存在しています。このコミュニティにかかわることで多くのメリットを得られます。ここでは筆者の経験したいくつかのメリットを紹介します。

良質な情報をキャッチアップ

コミュニティにはコミッターを始め、関連するソフトウェアのトップエンジニアがたくさんいます。その人達のアウトプットをチェックするだけで良質の情報を得ることができます。とくに最新情報については新たな機能を知ることでも、今ある問題を解決する術が増えます。手法を多く知ることは選択肢の幅を広げるだけでなく、発想も広がるので非常に有益です。また、機能だけではなく、運用方法や脆弱性のセキュリティ情報なども発信されています。とくにセキュリティ問題は非常にクリティカルですので重要です。迅速にパッチを当てることで情報漏えいを事前に防ぎ、正しい運用を行うことができます。

とくにDBとは、長く付き合っていく必要がありますから常に良質の情報をキャッチアップできることの有無は数年後に大きな差となります。

自分の立ち位置を知る

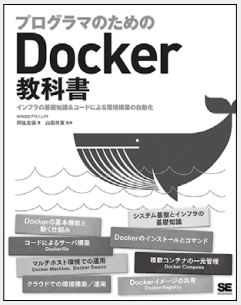
今やエンジニアが転職しながら流動していくのは当たり前の時代です。しかしコミュニティに関わらずにいるとどうしても視野が仕事の範囲に限られてしまいます。自分がどのような市場価値があるのかわからないとキャリアプランを練るのも難しくなります。また、会社の中での立ち位置だけで考え出すと、どうしても外に出る不安を持ってしまいます。

そんな中、コミュニティを通して多くのエンジニアとかかわることで自分の立脚点を知ることができます。また外の世界を知ることで自分の中で常識と思っていたことが非常識だと知ることも多々あります。その結果、自分が目指すキャリアにとって足りないことを気づくことができます。逆に自分の環境の良さを感じることも多々あります。もし、今の環境に不満があるなら、コミュニティにぜひ参加してみてください。自分の中で新たな視点ができ、答えが見つかることでしょう。

ギブ・アンド・ギブ

筆者がコミュニティで一番好きなところです。もしエラーログを読んでも解決できないような問題があったときにSlackなどでエラーログを添付して質問すると多くのエンジニアが親身になって相談に乗ってくれます。ここには見返りや損得の考えはありません。純粋に同じコミュニティの仲間が困っているから助け合うという目的だけです。またドキュメントやWebの情報で疑問点があったときにも気軽に聞いていただければ、わかる人が回答してくれます。時にはドキュメントの著者本人が回答してくれることもあります。このように損得勘定なく交流できる場所は豊かな人間関係を作り出します。筆者はそういうところがOSSコミュニティの好きなところです。

本特集記事を読んで興味を持った方は何らかのOSSコミュニティに、ぜひ参加してみてください。どんなコミュニティを選んだとしても暖かくあなたを迎え入れてくれるはずです。もちろん、ここ紹介したMyNAやJPUGも一緒です。それはどこのコミュニティも変わらない素晴らしいところです。筆者も皆さんとコミュニティの場でお会いできるのを楽しみにしています。



プログラマのための Docker教科書

WINGSプロジェクト　阿佐 志保、山田 祥寛 著
B5変形判／312ページ
3,000円＋税
翔泳社
ISBN＝978-4-7981-4102-2

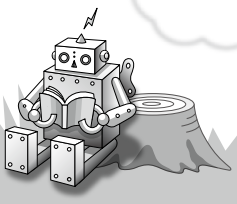
本書は3部構成となっており、インフラの基礎知識、Dockerの導入と使い方や環境構築、その応用という構成になっている。導入編として、2章を割いて、1章ではネットワークやOSなどの基礎知識を扱い、2章ではDockerを含んだ仮想化の概念を解説している。この部分は初心者向けの内容で、Dockerのためというより基礎のおさらい的な内容だ。その後、インストール、基本コマンドと進み、Dockerfileを使ったサーバ構築などの基本編を経て、コンテナ管理や運用についての応用編と続く。豆知識的なコラムもあり、役立つ情報を補っている。「プログラマのための」という書名だが、解説イラストも多くコマンドや設定など丁寧に解説されているので、普通にDockerを理解するのに役立つだろう。

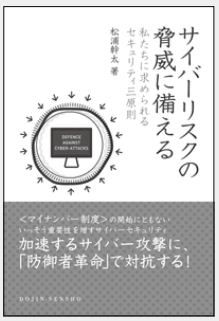
新・明解C言語 実践編

柴田 望洋 著
B5変形判／360ページ
2,300円＋税
SBクリエイティブ
ISBN＝978-4-7973-8410-9

C言語ではまりやすいエラー、理解しにくい動作、より良いプログラムのためのTipsを広く紹介している。読者としては、C言語の基礎は押さえたが次のステップに進みたいという人が対象だ。「ヘッダファイルの最後の行に改行文字を付けないとエラーになる」といった見えにくいエラー、「自動記憶域期間を持つオブジェクトは、プログラムの流れがその宣言を通過するときに生成・初期化される」といった内部動作の詳細、「構造体に型名を与えるには、マクロではなくtypedefを利用すること」といった落とし穴の回避方法のほか、型変換や文字列の扱いにおける注意事項も扱っている。C言語においての大きな壁「ポインタ」については、アドレス、間接演算子、配列との関係を整理しながら、サンプルコードと図表で丁寧に解説している。

SD BOOK REVIEW





サイバーリスクの脅威に備える

松浦 幹太 著
B6判／228ページ
1,700円＋税
化学同人
ISBN＝978-4-7598-1668-6

安全なインターネットの実現のためにどのような技術が使われているのか、インターネットを安心して利用するためにどのような知識を持つべきなのかを解説した1冊。それぞれの技術・知識の概要を押さえるための読みものではあるが、話を単純化したりはせず、適宜数式や擬似プログラミングなども使って踏み込んだ説明をしている。本書では、インターネットを活用したサービスが次々に生まれていく一方、攻撃者がいち早くそれらを分析して、ソフトウェア的な攻撃で実世界にも大きな被害を引き起こせるようになったという「攻撃者革命」の危険性を指摘している。それに対抗する術が「防御者革命」であり、研究者・実務家・ユーザが協働してセキュリティ技術を高める大きなPDCAサイクルを回す環境を整えるべきだと提言している。

お金をドブに捨てないシステム開発の教科書

中川 充 著
A5判／192ページ
1,880円＋税
技術評論社
ISBN＝978-4-7741-7817-2

企業のトップから末端の業務担当者までが満足できるシステムとは何だろう。本書はその答えとして「稼げるシステム」を挙げる。単なる業務改善レベルではなく、企業に利益をもたらすシステムを作るには、要件定義以前の「システム構想づくり」が重要だという。そこで、新システムの方向性を考えるための4つの視点（経営・会計・業務・システム）について解説する。「システム構想の基本方針は経営や事業の戦略そのもの。経営者が作ること」「システム構想はシステム部門ではなく、経営の単独プロジェクトで作れ」など、随所で経営陣の積極的な働きかけが強調されているのも印象的だ。読後はシステム刷新の意義を再認識し、「どうせ作るなら、稼げるシステムを目指したい」そんな前向きな気持ちになれるのではないか。

第2特集

インフラエンジニアの新常識！

1Gbps超 ネットワーク高速化時代の 適切なLANケーブリングの 教科書

少し前まで、有線LANの速度は100Mbps～1Gbps程度で済んでいました。しかし、最近のデータセンターやオフィスでもネットワークの高速化が進んできており、1Gbps超の速度を取り扱う機会が増えてきました。サーバでは10GbEをサポートしたり、オフィスでも10Gbps～100Gbpsをサポートしたりというケースもあります。

通信速度の高速化は、利用する通信ケーブルの多様化につながります。多種多様な通信ケーブルを選択し、そのシステムシステムごとに適切な通信ケーブルを選ぶ時代になりました。

また、ラック内の配線にもいろいろな技があります。たとえば、ケーブルを抜けにくく、保守しやすくするノウハウや道具があります。

本特集では、インフラエンジニアに求められる、最近のケーブリング事情について解説します。

Author 佐伯 尊子（さえき たかこ） ㈱ブロードバンドタワー

図表協力：㈱ハイアーネット、フルーク・ネットワークス、ザ・シーモン・カンパニー、バンドウィットコーポレーション日本支社、㈱ブロードバンドタワー

第1章

ネットワーク／サーバエンジニアに求められる ケーブルと配線の知識

最近では、ネットワーク機器が10Gbps、40Gbps、場合によっては100Gbpsを必要とするような時代になってきています。サーバでも、10Gbps（10GbE）をサポートするなど1Gbps超の速度を取り扱う機会が増えてきています。これに合わせて、利用する通信ケーブルの選択も考えなければなりません。本章では、通信ケーブルの知識や、配線方法、そして管理方法について解説します。

Author 佐伯 尊子（さえき たかこ） ㈱ブロードバンドタワー

はじめに

ここ10年くらい、オフィスの有線LANの速度はずっと100Mbps～1Gbpsをサポートしていれば、まったく問題がありませんでした。また、ネットワーク機器やサーバなども1Gbpsであれば、たいていのものが動いていました。しかし、ここ2～3年、これらの機器のポート速度が加速し始めました。ネットワーク機器が10Gbps、40Gbps場合によっては100Gbpsを必要としてきました。またサーバも10Gbps（10GbE）をサポートしたり、さらにストレージは、Fibre Channel の 8Gbps 超、FCoE（Fibre Channel over Ethernet）の10Gbps、またInfinibandで高速伝送を考えている方もいらっしゃるかと思います。このように、私たちのまわりで1Gbps超の速度を取り扱う機会が増えてきました。

通信速度の高速化は、利用する通信ケーブルの多様化につながります。多種多様な通信ケーブルを選択し、そのシステムシステムごとに適切な通信ケーブルを選ぶ時代に突入したと言えます。また、今までは1Gbps程度だったので、先輩やまわりの見よう見まねで配線してもリンクアップ[注1]していましたが、1Gbps超時代はそれだけでは安定した回線品質を保つことが難しくなってきました。ケーブル特性の基本に沿って配線することで、今までと同じ安定的な稼働が望めます。

このように、ネットワーク／サーバエンジニアのみなさんに求められる能力は、単に機器設定だけでなく、ケーブル1本の配線にまでおよぶようになりました。通信ケーブルの選択／配線はけっして難しいものではありません。本特集では、インフラエンジニアに必要な、通信ケーブルを中心としたケーブルやラック、電力などの選択方法・配線方法、そして管理方法について説明します。

注1）通信可能になること。

通信ケーブルと規格

最初に、ネットワーク／サーバ機器と通信ケーブルの関係をおさらいしましょう。「OSI階層モデル」の概念から、それぞれの階層ごとに通信のやりとり（プロトコル）を取り決めているものが規格になります。規格に則ることで、ケーブル同士、機器同士、そしてその上で動作するアプリケーション同士が、それぞれ自分たちの役割を適切に果たすことができます。

現在有線LANの規格のうち、利用者の大半を占めるものがEthernetになります。ちょっと前は、ATM（Asynchronous Transfer Mode）や、Apple Talk、FDDI（Fiber-distributed data

interface）など、いくつかのLANの規格がありましたが、現在はEthernetがほとんどです。そして、そのEthernetをベースとして、その上のレイヤでTCP/IP、HTTPなどを使って私たちはインターネットと接続しています。

Ethernet規格

まずは、そのEthernetの規格について、確認しましょう。Ethernetは、IEEE802.3で決められている規格です。IEEE（Institute of Electrical and Electronics Engineers, Inc.）は、アメリカ合衆国に本部を持つ電気工学・電子工学技術学会の略号で、ここの802部会の3分科会で、Ethernetについて審議、規格を制定しています。

表1に、Ethernet規格のうち、速度と利用する通信ケーブルについて、現在制定済み、もしくは制定中の規格の主だったものを示します。

通信ケーブルを選ぶ場合、利用する機器やその機器が則っている規格から一意にケーブルは決まります[注2]。Ethernet規格は、通信ケーブルに対する要求仕様を決定しています。まずこれらの情報から必要な通信ケーブル、使用上の制限を選び出していきます。

通信ケーブル規格

IEEE802.3にて、機器側から「こういう通信ケーブルの仕様が必要」という条件が出たところで、次にケーブルの規格を探します。通信ケーブルの詳細規格の主なもの[注3]は次のようになります。

・TIA-568 Commercial Building Telecommunications Cabling Standard「商用ビル通信配線規格」2015年9月にD版（最新版）が発行されました。
TIA（The Telecommunications Industry Association）米国通信機器工業会

・ISO/IEC11801 Information technology --

注2） 本来のスタートは「こういうデータをやりとりしたい」というコンテンツの部分からになりますが、今回は省略します。

注3） 米国の規格であるTIA、国際規格であるISO/IEC、そして国際規格を日本語化したJISとなります。したがって、ISO/IEC ≒ JISとなります（本来ならば、＝なのですが、改版作業が遅れており、現在JISは2004年版が最新となります）。

▼表1　Ethernetの速度と線種（抜粋）

Ethernet規格	規格で定められている速度（Ethernet名称）	通信ケーブル	制定年
IEEE802.3	10Mbps（10BASE-5）	同軸ケーブル	1983年
IEEE802.3i	10Mbps（10BASE-T）	ツイストペアケーブル	1990年
IEEE802.3u	100Mbps（100BASE-TX）	ツイストペアケーブル	1995年
IEEE802.3z	1Gbps（1000BASE-SX/LX）	光ケーブル	1998年
IEEE802.3ab	1Gbps（1000BASE-T）	ツイストペアケーブル	1999年
IEEE802.3ae	10Gbps（10GBASE- SR/LR/ER/他）	光ケーブル	2002年
IEEE802.3an	10Gbps（10GBASE-T）	ツイストペアケーブル	2006年
IEEE802.3ba	40Gbps（40GBASE-SR4/LR4）	光ケーブル	2010年
	100Gbps　（100GBASE-SR10/LR4/ER4）		
IEEE802.3bg	40Gbps（40GBASE-FR）	光ケーブル	2010年
IEEE802.3bm	40Gbps（40GBASE-ER4）	光ケーブル	2015年
	100Gbps（100GBASE-SR4/FR）		
IEEE802.3bq*	25Gbps（25GBASE-T）	ツイストペアケーブル	—
	40Gbps（40GBASE-T）		
IEEE802.3bs*	400Gbps（400GBASE-**）	光ケーブル	—
IEEE802.3by*	25Gbps（25GBASE-**）	光ケーブル	—
IEEE802.3bz*	2.5Gbps（2.5GBASE-T）	ツイストペアケーブル	—
	5Gbps（5GBASE-T）		

＊）規格制定中

Generic cabling for customer premises「構内情報配線システム規格」2010年版（最新版）
ISO（International Organization for Standardization）国際標準化機構
IEC（International Electrotechnical Commission）国際電気標準会議

・JIS X 5150「構内情報配線システム」 2004年（最新版）
JIS（Japanese Industrial Standards）日本工業規格

これらの規格は、通信ケーブルのうちの、ツイストペアケーブルや光ケーブルの仕様、それぞれのコネクタの仕様、配線経路の仕様（TIAを除く）、利用できるアプリケーション（EthernetやFibre Channelなどのプロトコル）とその距離について定めています。

ツイストペアケーブルの配線規格

それでは、具体的に機器に最適な通信ケーブルを探してみましょう。各機器のケーブルに関する情報は、たとえば**図1**のようになっています。

各社表現方法がバラバラです。機器仕様書に記載されているものだけでなく、「特徴」として解説している機器もあるため、一概に**図1**の抜粋だけで1つの通信ケーブルが決定するわけではありません。最終的には利用する速度などを鑑みながら、通信ケーブルを決定します。

ケーブル規格

図1の中で**例4**が、規格と規格名が一番わかりやすく書かれていることがわかります。基準規格がIEEE802.3→Ethernetであること、通信速度は「1,000Mbps」「100Mbps」「10Mbps」の3つがあることがわかります。さらにIEEE802.3の、どの規格がどの速度に対する仕様なのかもわかります。**表1**から、それぞれの速度で利用するケーブルについて確認できます。それぞれツイストペアケーブル[注4]だということがわかります。

次にツイストペアケーブルの規格を確認しましょう。TIA-568もしくはISO/IEC11081が該当するケーブルに関する規格となります。今回は、TIA規格で確認していきましょう。

TIAでは、ツイストペアケーブルの種類ごとの最大周波数や、利用するケーブル長が示されています。**表2**から、今回の利用に適したケーブルは「カテゴリ5e」「カテゴリ6」「カテゴリ6A」のどれでも対応することがわかります。

対応するケーブルが多い場合は、その状況に

注4）より対線。2本の電線をより合わせることによって、平行な線よりも電波や磁気の干渉によるノイズを抑えるようにしたもの。種類については、後で詳しく解説。

▼図1　機器スペック例（抜粋）

・例1

```
ポート
  24個の10/100ポート
```

・例2

```
標準ポート属性
  10 GbE SFP+自動検出（10Gb/1Gb）固定ポート×24
```

・例3

```
コネクタオプション
    1 GbEカッパーSFPオプション
    1000Base-SX and 1000Base-LX
    10 Gbps SFP+ オプション
```

・例4

```
基準規格      IEEE802.3ab   1000BASE-T
              IEEE802.3u    100BASE-TX
              IEEE802.3     10BASE-T
通信速度      1000Mbps/100Mbps/10Mbps
ポート        1000BASE-T/100BASE-TX/10BASE-T（RJ-45コネクタ）×24
              オートネゴシエーション、MDI/MDI-X自動認識
使用ケーブル  1000BASE-T   UTPエンハンスド・カテゴリ5以上
              100BASE-TX   UTPカテゴリ5以上
              10BASE-T     UTPカテゴリ3以上
```

応じて適切な線種を選びます。「適切」に含まれる条件は、伝送速度のほか、コストや納期、入手しやすさなどがあります。今回は1,000Mps/100Mbps/10Mbpsですから、カテゴリ5eを選択することが適切であると言えるでしょう。また、**図1**の**例4**の機器側スペックに「コネクタはRJ-45」と記載があり(**写真1**)、またTIA-568規格では、このコネクタにツイストペアケーブルを取り付ける際の色や配置について詳しく記載がありますので、このコネクタを選択して問題ないことがわかります。

ここまで確認して、やっとこの機器に必要なケーブルとコネクタの当たりがつきました。

ケーブル成端作業

次に、ケーブルの成端(結線)方法について確認します。ケーブルとコネクタの当たりが付いただけでは、最終的な配線はできません。コネクタ付ケーブルを購入したり、工事業者に配線工事を依頼したり、もしくは自分でケーブルにコネクタを取り付けるときに気をつけなければならないことがあります。コネクタの取り付け(成端もしくは結線という)方法についても知っておく必要があります。これもTIA-568の規格に記載があります。**図2**の2種類の成端方法があります。

ケーブル内のツイストペアの1組の対は、それぞれ「色線：色に白の混ざった線」の組み合わせになっています。ケーブル内には4組のツイストペアがあり、伝送するときには色同士の組み合わせにも注意する必要があります。その色同士の組み合わせは、T568AとT568Bの2つがあります。

接続する機器の間にいくつかの接続点がある場合は、T568A成端とT568B成端が混在することは望ましくありません。どちらかで統一するようにしましょう。

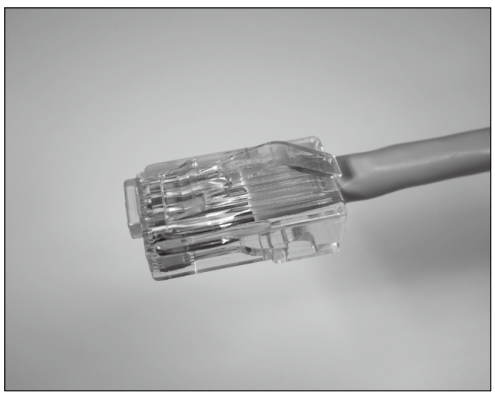
T568Aは、官公庁の通信設備や公共施設のLANなどで使われていることが多いようです。T568Bは北米やヨーロッパ、日本でもネットワークサービスを行っている事業者が利用していることが多いようです。一般的なオフィスLANでは、どちらが多いというような傾向はありません。普段から「当社はすべてB成端」など、共通ルールを取り決めておくことが必要です。そして、工事業者に発注する際、もしくは購入や自分たちで製作をする際には必ず、「成端方法はT568A(またはT568B)で」と、その旨伝えることが大切です。

今まではT568AとT568Bの混在した配線であっても、とくに気

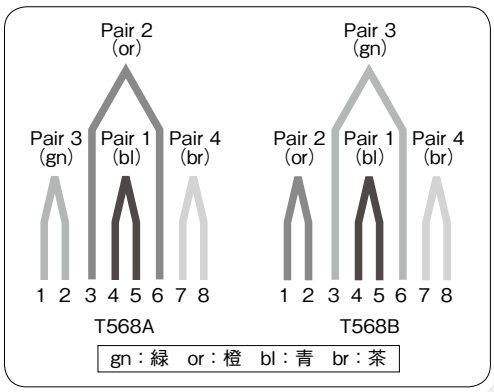
▼表2　TIA-568(ツイストペアケーブル部分の抜粋)

線種	保証周波数帯域	速度			
		10MbE	100MbE	GbE	10GbE
カテゴリ5e	100MHz	100m	100m	100m	—
カテゴリ6	250MHz	100m	100m	100m	最大37m
カテゴリ6A	500MHz	100m	100m	100m	100m

▼写真1　カテゴリ5eケーブルとRJ-45



▼図2　T568A成端とT568B成端



にせずに利用してきたかと思います。しかし今後、より高速通信を考えていくならば、成端方法の統一を図ることで、不要な通信エラーの要因を排除できます。

モジュラープラグとモジュラージャック

今までは簡便的に「コネクタ」と表記していましたが、正確には「モジュラープラグ(オス)」と「モジュラージャック(メス)」という表現になります(**写真2**)。以降では、モジュラープラグ／モジュラージャックという表現をしていきます。

単線とより線

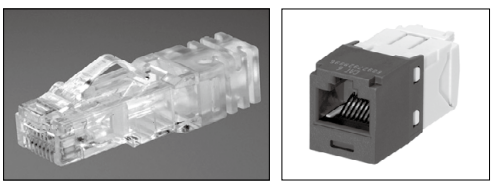
また、ツイストペアケーブルと一口に言ってもケーブルには2種類があります。「単線ケーブル」と「より線ケーブル」です(**図3**)。

1本の銅線に被覆(周りを絶縁体で包んだ状態)したものを単線ケーブルと呼びます。フロア間など比較的長い距離で用います。また断面積が単線と同等になるように細い単線を束ねた、より線ケーブルは、パッチパネル[注5]と機器間、島Hub[注6]とPC間など比較的短い長さで、取り回しの自由度を求められる場所に使います。それぞれの特徴を**表3**に示します。

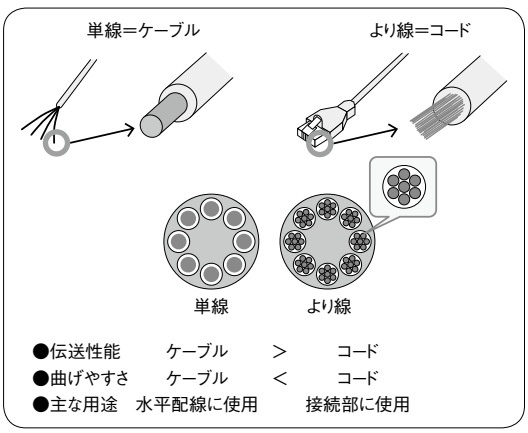
気をつけたいのは、モジュラージャック(メス)成端には単線ケーブルが用いられ、より線ケーブルを使ったものは基本的にはありません。しかし、モジュラープラグ(オス)はより線用が基本ですが、単線用のもの(ワークエリアコード[注7]で利用することがあります)もあります。自分でケーブルを成端する場合は、それぞれケーブルに見合ったモジュラープラグ／モジュラージャックを用意する必要があります。

なぜこのように2種類のケーブルが必要なのでしょうか？　それについては、のちほど説明します。

▼写真2　モジュラープラグ(左)とモジュラージャック(右) [1]



▼図3　単線ケーブルとより線ケーブル



シールドなしケーブルと、スクリーンシールドつきケーブル

別な角度から、ツイストペアケーブルについて考えてみましょう(**図4**)。私たちが通常、よく利用している通信ケーブルは、UTP(Unshielded twisted Pair cable)と言われ、スクリーンシールド[注8]がついていないものになります。ケーブルによっては、スクリーンシールドがついたタイプScTP(Screened Twisted Pair cable)があります。

組み合わせだけで考えるとたくさんありますが、一般に用いられているものは、おもに次の3種類です。

注5) ラックなどで使用される、モジュラージャックが一列に並んだパネル。
注6) オフィスなどでデスクの集合ごとに配置されるHubの呼称。
注7) 会議室のLAN配線や、ローゼットから無線LANのアクセスポイントまでの配線などのエリア内配線。
注8) ケーブルの周りのシールドとして、銅やアルミの箔を巻いたり、金網で覆ったもの。

▼表3　単線とより線の特徴

線種	特徴
単線	曲げにくい、(パーマネントリンク(後述)など)比較的距離の長い部分を配線
より線	曲げやすい、機器間もしくは、機器とパッチパネル間(パッチケーブル)などの配線に利用

・U/UTP

従来のシールドなしケーブルになります。

・F/UTP

ケーブル外皮の内側に薄い金属膜(スクリーン)で中のケーブルを覆う構造です。ただし、中のケーブル1対1対は単に被覆が掛かっているだけで、被覆に金属はありません。

・S/FTP

各対を金属膜で覆い、さらに4対すべてを編組(金網)で覆う構造です。

これらのスクリーンシールドが必要な理由は、EMI(電磁妨害)からケーブルを守り、伝送するデータの品質を保つためです。病院や工場など、電磁波にさらされている環境では、効力を発揮します。また、ケーブルの各対間を遮蔽することで、ほかの対から漏れ出すノイズを防ぎ、伝送品質の劣化を防ぐことができるためです。

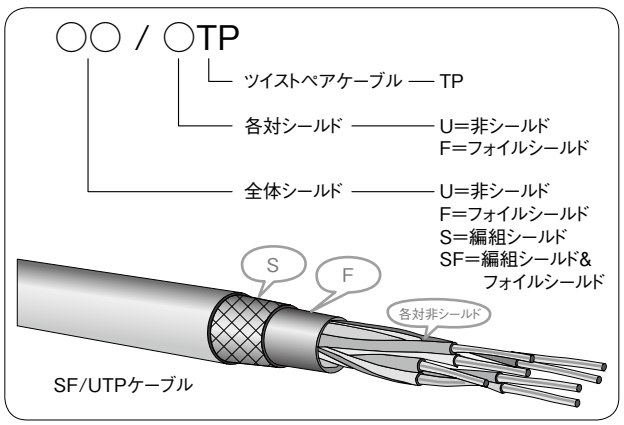
細径線と、通常線

同じカテゴリ5eでも、通常の太さとは別に「細径ケーブル」と呼ばれる細いケーブルも多々利用されています。これは、パッチパネルに集まる大量の機器ケーブルをうまく整理するため、また機器ケーブルがラック内の空気の流れを妨げるのを抑えるために開発されました。具体的に通常のケーブルとどこが違うのでしょうか？

表4に例として、カテゴリ5eの細径線と通常線の仕様を比較します。

その違いは、見てわかるとおりケーブル外径にあります。**写真3**に示す右が細径線、左が通常線になります。数字で見ると、2mm程度の差しかありませんが、実際に現場で利用すると、その違いが実感できるかと思います。

▼図4　ツイストペアケーブル構造と名称



▼表4　カテゴリ5eの細径線と通常線

項目	細径線	通常線
規格	TIA-568準拠	TIA-568に合致
導線サイズ	AWG28～AWG30	AWG24
ケーブル外径[mm]	3.2～3.8	5.1～5.8
最大チャネル長	60m程度	100m
その他	PoEの使えないベンダ有り	―

▼写真3　細径線(右)と通常線(左) [1]

パーマネントリンクとチャネル

パーマネントリンクはラックからローゼット間、チャネルはパーマネントリンクに機器ケーブルを含めた機器間のことを言います(詳しくは次項「構造化配線」を参照)。

TIA-568では、水平ケーブルと、幹線ケーブル、機器ケーブルの3つが示されています(**図5**)。それぞれ、特徴に見合ったケーブルを選ぶことが大切です。先ほど説明した単線や、より線もそれぞれ適材適所に配置することで、その特性を存分に活かすことができるのです。

これらの配線ルートで、水平ケーブルと幹線ケーブル、一部ワークエリアコードは、**図3**で示した単線を用います。機器ケーブルは、より線を用います。水平ケーブルや幹線ケーブルは、一度ケーブルを張ったら(敷設といいます)、トラブルがない限り固定して利用する部分ですので、しなやかさが不要な単線ケーブルを選定します。また機器ケーブルは、パッチパネルと機器の間、ローゼットと端末の間を接続するケーブルですので、しなやかで曲げ癖などが少ないものの方が、扱いやすいことがわかります。

構造化配線

図5のように、パッチパネルを用いて、機器間を接続する方法を「構造化配線」と言います。先ほど示した規格すべてが、この「構造化配線」をもとに設定されています。そもそも、この構造化配線がどうしてできたのか？というと、構内(ビル丸ごと一棟や、大学などのいくつも建物がある専有された敷地内)で通信配線をするときに、機器間を一対一で配線していたのではケーブルの配線経路がぐちゃぐちゃになって、保守性・拡張性が望めなくなるため、もっとすっきり管理しやすい方法を考えた結果、このような形がよいとされました。この形になったのが1980年代、ちょうど10MbpsのEthernetが同軸からツイストペアケーブルに変わったときです。

それまでは、同軸ケーブルを用いたLANでした。同軸ケーブルでは、その途中途中にトランシーバもしくは同軸のT型コネクタを接続し、別のケーブルと接続するバス型配線でした(**図6**)。バス型配線は、オフィスのレイアウトに制限を与える配線でしたが、その後ツイストペアケーブルによるスター型配線は、オフィスのレイアウトがバス型と比べ自由度が高いため、ツイストペアによるLAN配線が急速に広まりました。

自由度が高くなった分、理路整然とした配線を行いにくくなったため、**図5**に示す構造化配線を規格化し、それに則った配線を構築することを業界団体として推奨しました。このようにすることで、次の2つに分離することが可能になりました。

・パーマネントリンクもしくは単にリンク

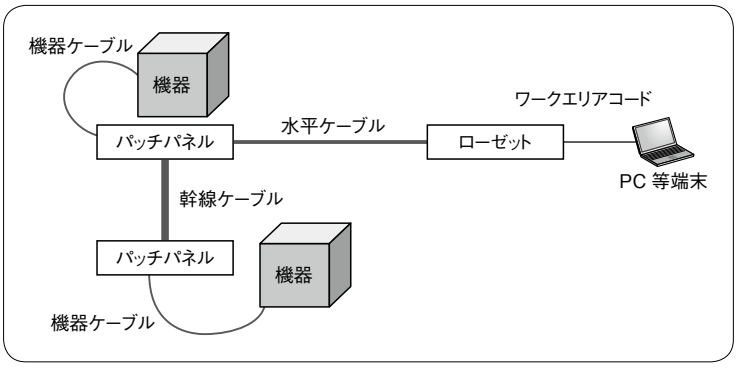
図5のパッチパネル＝水平配線(もしくは幹線)＝パッチパネルやローゼット間

・チャネル

図5のパーマネントリンクに機器ケーブルを含めた機器間

パーマネントリンク部分を工事会社に依頼し、機器

▼図5　水平ケーブルと幹線ケーブル、パッチケーブルの配線例



ケーブルは自分たちで用意します。すると、担当範囲を明確に分けることが可能になります。この構造化配線は、現在に至るまで欧米を主として、利用されています。

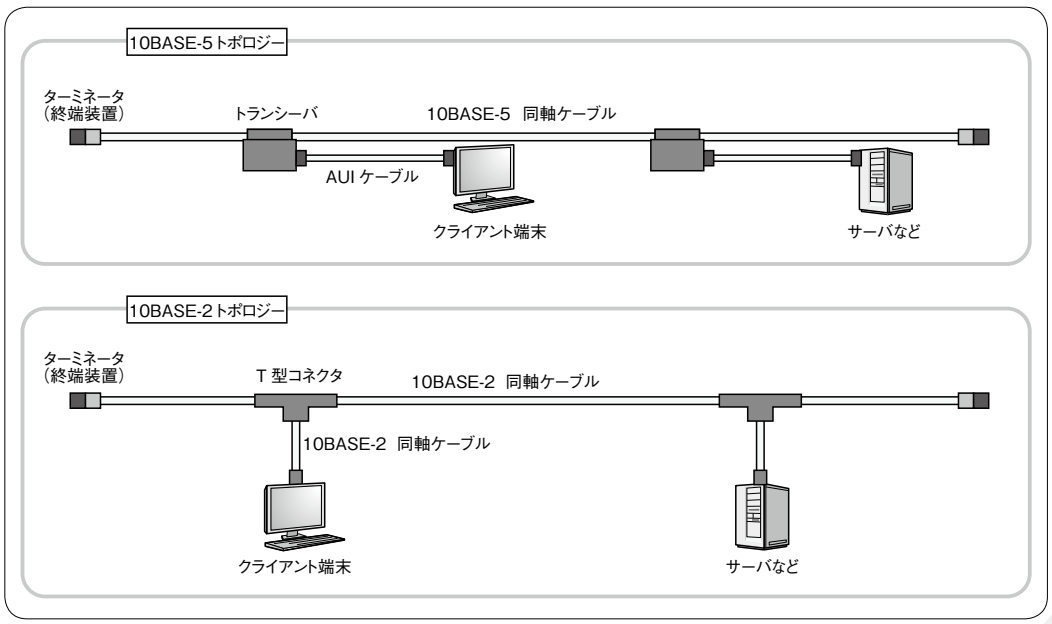
日本の場合はどうでしょうか？　実は日本では「構造化配線」の規格を踏襲しつつ独自路線が主流なようです。オフィス内で「分岐点」に相当するモジュラージャックは、会議室の壁のコンセント、床やテーブルの中に仕込まれているローゼットと呼ばれる四角い箱に相当します（**写真4**）。しかしオフィス執務室では、島Hubに直接モジュラープラグを接続するパターンも多く見受けられます。また、配線の大元側は「パッチパネル」を用いて、オフィス機器と配線をしているところが多いですが、100％ではないようです。

日本で構造化配線が定着していない理由の1つに、パッチパネルを設置する必要性、フロア内の通信ケーブル工事に対する時間とコストについて、十分な理解が得られていないように感じます。とくに今までは「電話工事」があったので、そのついでで通信工事も実施していましたが「（規格に合致した）LAN工事」というより、最低限必要なものを必要なところに配線する工事でしかなかったようです。さらに近年は、固定電話から携帯（無線）電話へ、LANも「有線LAN」から「無線LAN」にシフトしているオフィスが多いようです。そのため、さらに電話工事、有線LAN工事にかけるコストや時間が減っています。

これはまさに、先ほど**図6**で説明したバス型配線だとオフィスレイアウトの自由度がないため、スター型配線が広まったときと同じように、「スター型配線など物理的配線をしなくても、無線という媒体を使えば、配線経路を考慮せずに済む」と考えるトポロジーの変化の時代だと考え

▼写真4　ローゼット例[1]

▼図6　10BASE-5、10BASE-2の配線トポロジー



られます。

ただ、無線LANの場合は、無線LANのAP（アクセスポイント）までは必ず有線のLANケーブルが必要になりますので、有線LANの配線の考え方は必ず残ります。そのためにも「構造化配線」の考え方は、LAN配線をするうえで、押さえておきたいポイントの1つです。

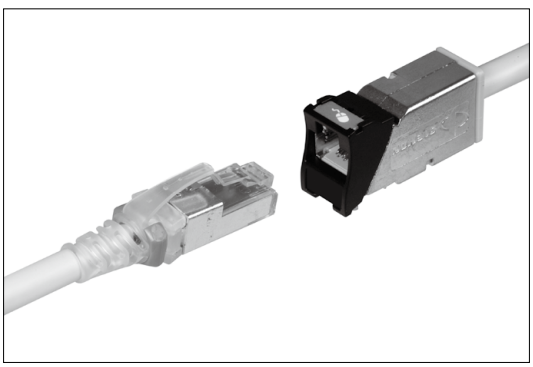
ScTPケーブルの接地

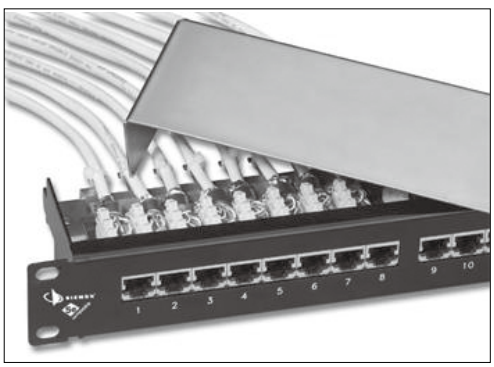
スクリーンシールドケーブルは、システム全体に渡って金属箔に電気が流れる状態を作らないと（途中で寸断されると）まったく意味をなさないだけでなく、より伝送が不安定になることが知られています。そのためには、単にケーブルだけをScTPにするだけでは不完全です。モジュラープラグやモジュラージャックもScTP用の金属の付いたものが必要になります（**写真5**）。さらにパッチパネルもScTP用とし、パッチパネルごとに接地が必要になります（**図7**）。

日本では、ScTPによる通信配線工事は、病院や工場など特殊な事例を除き、一般的なオフィスや自宅などでの工事では、ほとんど要望がありません。そのため、LAN配線用のケーブルは、シールドなし以外をご存じない読者も多いと思います。しかし最近は、一概に特殊事例のみの配線方法ではなくなってきました。というのも、オフィスやデータセンターでも10Gbps以上の高速伝送が始まりました。家庭でも10GbEを謳うブロードバンド回線が提供され始めています。このような高速伝送の用途で着実にScTPが使われ始めています。では、このような高速配線で、なぜシールド付ケーブルが必要になるのでしょうか？　それは「エイリアンクロストーク[注9]」など1Gbps時代にはとくに気に

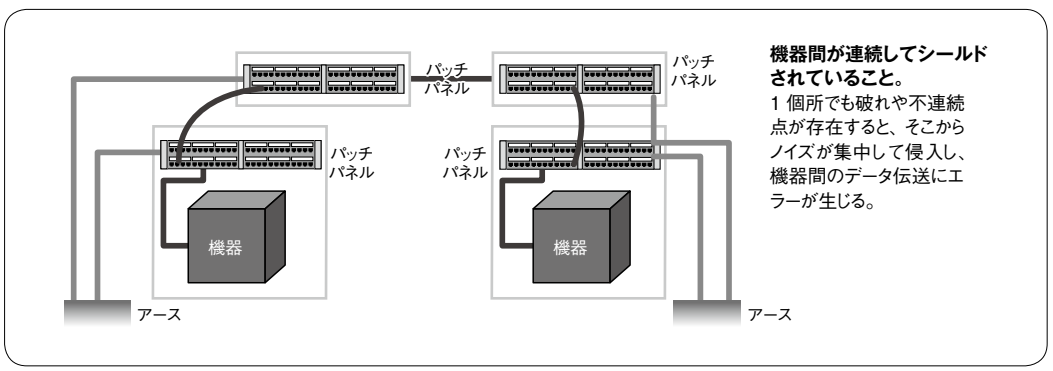
注9） 隣り合ったUTPケーブル間で伝わるノイズのこと。

▼写真5　専用RJ-45（左）およびパッチパネル（右）例[2]





▼図7　ScTP配線のシールドアース



ならなかった伝送特性が影響して、伝送品質が劣化することがわかっているからです。そのため、10Gbps以上の高速伝送を考える場合は、敷設する環境のほか、将来も含めて利用する伝送速度にも考慮してケーブルのグレードやシールドの有無、そして配線長を選ぶ必要があります。

ツイストペアケーブルの品質

ツイストペアケーブルの品質についても確認しておきましょう。

購入

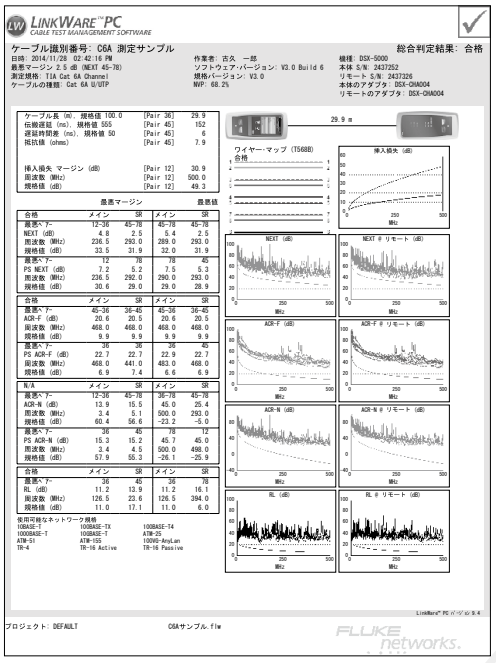
みなさんはどのようにしてツイストペアケーブルを購入していますか？　ケーブル専門メーカーからの購入のほか、会社であれば、法人向けネット通販サイトや家電量販店でもツイストペアケーブル、モジュールプラグのパーツ、両端コネクタ付ケーブルが販売されています。個人の方では100円ショップで販売しているケーブルを使う方もいらっしゃるかと思います。ツイストペアケーブルはどこで購入しても特性が大きく変わるものではないと思い、納期や価格、手軽に発注できてかつ会社のお墨付きのあるルートから購入している方がほとんどかと思います。通販サイトや家電量販店、100円ショップのケーブル類は、製造元メーカーが多種多様なため、品質のばらつきが大きいようです。同じ規格を謳っていても、各製品ごとに品質にばらつきがあり、中には規格を満足していないものもある旨の記事が掲載されているのを目にされた方もいらっしゃるかと思います。どうしてこのように製品にばらつきが出るのでしょうか？　それは、このお手軽な販売元は出荷にあたって伝送特性などの品質チェックが簡単に済まされているケースが多いためです。

専門メーカーは、製造したケーブルの段階で品質チェックしていますし、端末の加工を施したものであれば、成端した状態で測定して出荷しています。そのため規格に合致したものだけが販売されています。品質が安定しているため、安心して利用できます。確かに量販店などと比べると「高い」イメージがありますし、さらに「今すぐほしい」に対応するのは難しいですが、専門の小売店であれば在庫を用意しているケースもあります。予測がつくのであれば、事前にケーブルを発注しておけばよいですし、どうしてもすぐ必要であれば、小売店の場所を確認しておくといいでしょう。

成端の確認

自分で設定する機器類を接続するため、ご自身でツイストペアケーブルにモジュラープラグを取り付ける作業をされる方もいると思います。しかし、自分で成端作業を実施した際、その機器ケーブルなりをどのように保証しますか？　導通試験機（ケーブルチェッカー）で「結線されている」「結線の間違いがない」ことを確認していますよ、というお答えをいただくかもしれません。しかし、本当にそれだけでよいのでしょうか？　導通試験器は本当に簡単なチェックしか

▼図8　ツイストペアケーブル測定結果例[3]



できません。ケーブルの特性を測定するには、それ相応の測定器が必要になります。

パーマネントリンク施工

図5のパーマネントリンクの部分を工事会社に発注、施工後の完成図書を受け取った際に**図8**のような資料を見たことがあるのではないでしょうか？

難しいグラフがいくつも並んでいますが、最終的には右上の「合格」欄に「レ」点が入っていれば、そのケーブルは特性を満足しているという印になります。先ほどの導通試験器と比べ、1つの表と9つのグラフ表示があると、「いろいろな項目を測定して、最終的にOKが出ている」と、項目ひとつひとつの意味はわからなくとも「きちんと検査している」ことがわかります。工事業者は、単に敷設するだけでなく、敷設したケーブルがちゃんと規格を満足しているかどうか、必ず全数チェックしているのです。そのための試験器の例を示します。

写真6は、けっして安価な測定器ではありませんが、規格で制定されている「現場レベルで必ず測定しなければならない項目」が測れる測定器です。

このような測定器を用いて、1本ずつパーマネントリンクの保証をした水平配線・幹線配線であるにもかかわらず、最後の機器ケーブルを自作したり特性の安定しない安価なケーブルを用いると、その結果、機器間すべての通信が不安定になってしまいます。それを避けるためにも機器ケーブルは自作したり、安価な方法で入手せず、必ず専門メーカーのケーブルを購入しましょう。今までは1Gbps程度だったので、たまたま量販店のケーブルや自作のケーブルでも動作していたかもしれませんが、これからは機器ケーブル1本でも、システム全体の不良を引き起こす可能性も出てきます。

まとめ

このように、機器のスペックなどに記載されている情報の裏には、これだけ調べないと、正確なケーブルやコネクタを定めることができないことがおわかりいただけたと思います。さらに、配線するルートや環境も加味して、最終的にケーブルを決定します。今まではツイストペアケーブルの配線であれば、「カテゴリ5eのツイストペアケーブルとRJ-45のコネクタ」で大概のことは済んでいました。今までは、それは間違いではありませんでした。しかし、新しい速度や規格に対応したケーブルやコネクタを導入するときには、何をもって最適なのかを見極める力を育てることが大切です。そのためにも規格や原理を理解しておくことは重要なことだと言えます。SD

▼写真6　ツイストペアケーブル測定器例（DSX-5000）[3]

■参考

[1]パンドウイットコーポレーション日本支社
http://www.panduit.co.jp/
[2]ザ・シーモン・カンパニー
http://www.siemon.co.jp/jp/index.html
[3]フルークネットワークス
http://www.flukenetworks.com/
・IEEE802.3
http://ieee802.org/3/
・ISO / IEC 11801
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=65423

Column

スムーズなIEEE802.11ac無線LAN移行のために ――「NBASE-T」「MGBASE-T」の取り組み

Author 佐伯 尊子（さえき たかこ） ㈱ブロードバンドタワー

今やオフィス内のネットワークは、無線LANが主流になっているのではないでしょうか。本コラムでは、無線LANをめぐる有線技術の最新トピックについて述べます。

無線LAN規格

無線LANもEthernetと同様、IEEE802委員会によって規格化されています。IEEE802.11で最初の規格が制定されました。最新の規格はIEEE802.11acで、最大6.9Gbpsの通信を制定しています。

この規格をサポートするためには、速度に見合う線種を選び、無線LANのアクセスポイント（以下、AP）とネットワーク機器間を接続しなければなりません。6.9Gbps以上の速度で通信することができるツイストペアは、カテゴリ6（伝送距離：37m）、カテゴリ6A（伝送距離：100m）となります。

10GbEのツイストペアケーブル規格

前述のとおり、無線で6.9Gbpsの速度をサポートするには、有線は6.9Gbps以上の速度をサポートするものを選びます。TIAでもISO/IECでも規定しているカテゴリ6、もしくはカテゴリ6Aが、10GbEを伝送できる仕様です。

次に、APとネットワーク機器間の距離について考えてみます。Ethernetの最大離隔距離100mを考慮すると、選択できる線種はカテゴリ6Aのみとなります。

したがって、伝送速度および機器間の距離から、カテゴリ6Aを選択することが望ましいことがわかります。

ここまでは、一般的に考えられている無線LANと有線LANの規格に沿った考え方です。しかしよく考えてみると、無線LANのAPを今までのIEEE 802.11bやIEEE802.11n対応のものから、IEEE802.11ac対応のものに替えることで、すでに利用しているカテゴリ5eやカテゴリ6の通信ケーブルをすべて張り替えなければならないことがわかります。

この手間がIEEE802.11ac普及の阻害要因となることを懸念して、次の2つのアライアンスが発足しました。どちらのアライアンスも、「無線LANのAPや機器が更新された際に、いかにして現在、利用されているカテゴリ5eやカテゴリ6の既存配線を有効利用し、スムーズに移行できるようにするか」を検討する団体です。

- NBASE-Tアライアンス[注1]
  シスコ社やインテル社を中心にケーブルメーカーや測定器メーカーが参加
- MGBASE-Tアライアンス[注2]
  ブロケード社やブロードコム社を中心に、キャリアなどが参加

この2つのアライアンスそれぞれで、伝送方式のしくみを検討しており、まずNBASE-Tアライアンスが2014年秋にNBASE-T 1.0を制定しました。NBASE-T 1.0では、既存のカテゴリ5eやカテゴリ6で、100mの2.5Gbpsや5Gbpsを伝送させる仕様としました。

また、無線LANのAPの高速化に対応することを主目的としているため、PoE（Power over Ethernet）[注3]による15W、30W、60Wの3種類の給電のしくみもサポートしています。

標準化規格「IEEE802.3bz」の発足

これらのアライアンスだけでは、どうしてもそのアライアンスに加盟しているメーカー間の機器による制限がかかってしまい、そのほかのメーカーの機器を利用することができません。ユーザの立場に立った場合、せっかく規格が制定されても、十分な恩恵を受けることができないことが想定されます。

そこで、2015年5月のIEEE802.3委員会にて、正式に規格化することが決定しました。初回の会合が2015年6月に実施され、これから規格の制定が始まります。今後はアライアンスメンバーだけでなく、IEEEに参加しているメンバー間で幅広く意見の交流が実施され、より良い規格が制定されるのではないかと思われます。SD

注1） http://www.nbaset.org/
注2） http://www.mgbasetalliance.org/
注3） Ethernetのカテゴリ5以上のUTPケーブルを使って電力を供給する技術。詳細は、本特集のAppendix 2を参照のこと。

第2章

勝負は機器設定、マウント時から始まっている

保守性・耐障害性に優れたラック内配線

ケーブルを選択し、ラックに搭載する機器がそろったら、いよいよラック内の配線です。ケーブルが抜けにくく、かつ、構築後も保守しやすくするには、さまざまなノウハウや便利な小道具があります。適切な通信配線経路について具体的に考えていきましょう。

Author 佐伯 尊子（さえき たかこ）　（株）ブロードバンドタワー

通信配線経路

オフィスにおける通信配線経路は、大きく2種類に分かれます。

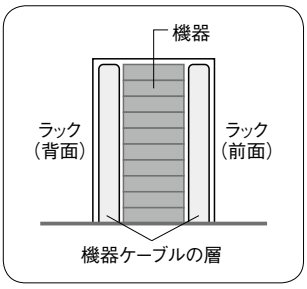
①ラック内もしくは近接ラックの機器と機器を接続する場合
②ラック内機器と、オフィスフロア内（ラック外）機器を接続する場合

今回は、①の配線の注意点について、説明していきます。

ラック内の配線経路を考える

機器間は直線距離で、最大でも3m程度と仮定します。一番簡単なのは、機器のポート（コネクタの差込口）間を直線で結ぶルートで接続した場合です。3mの機器ケーブルさえあれば、2つの機器は接続できます。この接続方法を選んだ場合、機器ケーブル長は最短距離を結ぶことができますが、本当にこの配線方法でいいのでしょうか？

▼図1　機器ケーブルの積層化（ラックを横から見た図）

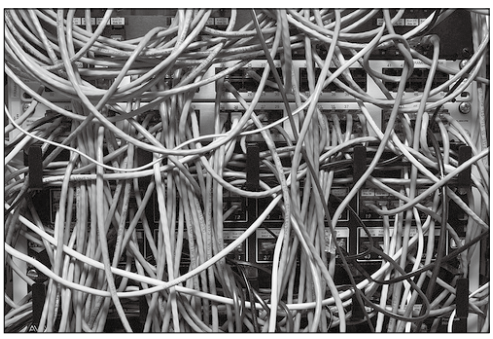


というのも、1本や2本くらいでは気がつきませんが、何十本も縦横無尽に最短経路で接続すると、ラックの中のケーブルの広がり方が面になってしまうため、ラック前面もしくは背面一面に機器ケーブルの層ができてしまいます（**写真1**、**図1**）。そうなると、1本1本のケーブルを識別することはとても困難になってしまいます。

また、空きポートにケーブルを接続しようとしても、機器ケーブルに遮られて、空きポートを外から見つけるのが難しくなります。無理にケーブルをかき分けて、空きポートを探すことで、既存の接続されているケーブルが外れてしまう恐れもあります。

また、サーバ機器などはメンテナンスを行う際、機器を前面に引き出して作業することが多

▼写真1　面配線の例（TechVirtuoso, LLC.）[1]



いかと思います。その際も、ラックのドアを開けると、まず通信ケーブルの層があって、その次にサーバが設置されており、サーバを引き出すどころか、触ることもできなくなる可能性が高いです。そうすると、保守(メンテナンス)ができません。機器ケーブルの積層化は、保守・運用するうえでとても危険な配線と言えます。

この積層化を防ぐには、ラック内の通信経路の考え方を面から線に変更することが必要です。具体的には、ラックの中に配線のための経路(道路)を設け、その経路からはみ出さないように配線します。それでは、具体的にどのようにして経路を作るかを考えてみましょう。

配線経路を作るときのポイントは次の2点だけです。

・最短経路で接続しない
・一度決めた経路は機器の大幅変更など、レイアウト変更がない限り必ず守る

「1本くらい経路を外れてもいいや」と思って適当に配線すると、それが無法地帯への入口になり、ケーブルが積層化してしまいます。「リンクアップなどの確認をするためにとりあえず配線しておいて、あとで適切な長さのケーブルに差し替え、正規のルートで配線しよう」とか、「適切な長さのケーブルは在庫がなかったので、間に合わせのケーブルを利用しよう」と思って何気なく接続した機器ケーブルは、経験則上、あとで差し替えることはまずありません。

詳しくは後ほど説明しますが、配線経路をしっかり定めておくと、いくつかの決まった長さのケーブルだけを利用しても、きれいな配線を保つことができるのです。

先述の2点をふまえ、配線経路を作ってみましょう。上下方向と左右方向に経路を作ります。**写真2**に例を示します。

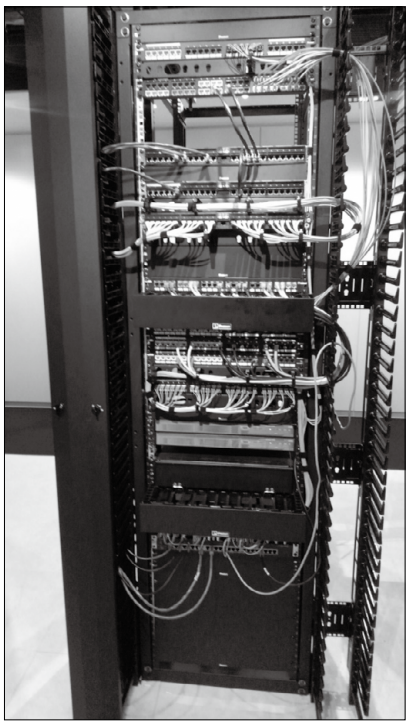
上下方向は、マウントフレームの外、側板の内側を利用します。左右方向は、ラック内の適切な場所に整線パネルを用いて、横配線用の経路を設けます。左右方向は、ラックマウントに必要なU数を使うことになるため、もったいないと思うかもしれませんが、配線経路を必要とするラックは、実はフルに機器を実装することのほうが少ないため、ブランクパネルなどで塞ぐことを考えるよりも、このように左右配線の経路として整線パネルを利用したほうが、無駄なくラック内を利用することができます。

整理すると、次のようになります。

・ラックの上下方向は、マウントフレームの外側を利用する
・ラックの左右方向は、整線パネル(後述)を用いて通路を作る
・ラックの前後方向は、上下と同様マウントフレームの外側を利用する方法と、マウントフレーム内側の空きスペースを利用する方法の2種類ある(マウントフレームの外側を利用する方法を推奨する)

それぞれの配線方法についてもう少し詳しく見ていきましょう。まずは上下配線についてで

▼写真2　経路を考えた配線例[2]



す。ToR(Top of Rack)配線[注1]で、ラック上部のスイッチと下部のサーバを接続する場合には、通信ケーブルはそのまま下に垂らして、ポートに接続します。

このとき、複数台あるサーバ機器のポートの位置は、基本的に同じ場所になるため、スイッチ直下に整線パネルを設けて、いったん整線パネルで受けてから、左右に振り分けて、サーバと接続することで、スイッチに接続する大量のケーブルの入線個所を分散させることができます(**図2、写真3**)。

次に左右配線についてです。先に挙げた**写真2**を見るとわかるように、整線パネルを用いてケーブルの左右の経路を確保しています。またこの左右の経路はラック内の適切な位置に複数設けることが望ましいです。ケーブルの量や、ケーブルが左右を行き来する場所を考慮し、適切に整線パネルを設けることが大切になります。

最後に前後の配線についても考えてみましょう。前後の配線は、なるべく少なくしましょう。どうしても配線しなければならないもののみ吟味して、1ヵ所もしくは2ヵ所程度、前後用の配線位置を決めます。上下左右と違って、配線の難度が高いため、日ごろの保守運用ではなるべく追加／変更をしないで済むように設計時に心がけることがポイントです。

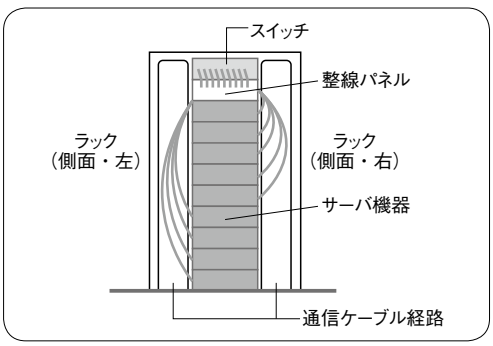
サーバ類は基本的に、背面に電源ケーブルや通信ケーブルのポートがあります。そのため、まずは機器を接続する場所は、前面なのか、背面なのかを吟味します。たとえば、ToRスイッチのポートは前面、サーバのポートは背面に多く存在します。このとき、慣習だけでスイッチを前面に付ける必要があるでしょうか？　エアフローや機器のメンテナンス性を考慮したうえで、マウント位置を決定することが重要です。

背面マウントした場合でも、スイッチのエアフローを前面背面で切り替えられるのであれば、スイッチ自体は背面に設置するほうが配線するうえではメンテナンス性が良くなります。背面にマウントすることで、前面から背面へのケーブルの行き来がなくて済みますし、前後の行き来がない分、ケーブル長を短く抑えることができます。ケーブル長が短くなれば、接続されている機器同士の視認性が上がることと、メンテナンス性に優れることが予想されます。

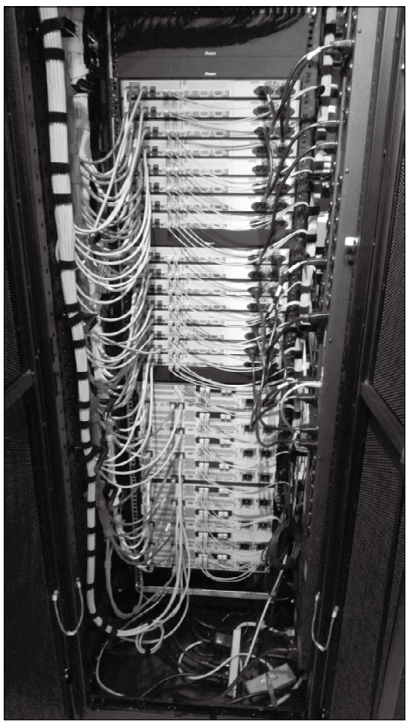
また、前面と背面に接続ポートが分かれる場合は、ケーブルの配線経路はマウントフレームの外側を確保しましょう。このとき、間違って

注1) サーバを収容するスイッチの設置方法の1つで、ラックごとに、ラック上部にスイッチを配置する方法のことを指す。

▼図2　上下配線経路例



▼写真3　上下配線経路例[2]



もマウントフレームの内側からケーブルをくぐらせてはいけません。ケーブルをマウントフレームの内側にくぐらせることで、機器を設置するための区画(ユニット)がケーブルの配線のために削られてしまいます。機器を設置できなくなるだけでなく、ブランクパネルなども取り付けられずエアフローが確保できなくなるため、NGな配線と言えます。

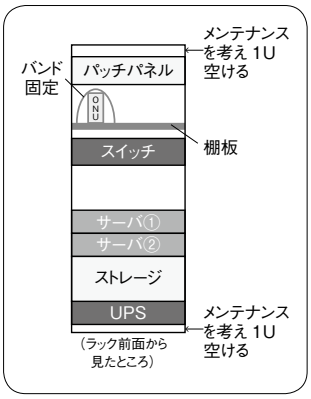
機器のマウント方法

最初に機器のマウント(設置)方法についておさらいしましょう。当たり前ですが、重いもの、奥行きがかさばるものを下から順に搭載します。ラック自体の転倒を防止するという理由もありますが、重い機材をラック上部に設置すると作業者の腰や首、腕を痛める原因にもなるので、それを防ぐ意味もあります。マウント例を**図3**に示します。下からUPS、ストレージ、サーバ(複数)、スイッチ、パッチパネル、となります。

機器ケーブルの配線方法

機器ケーブル類は1本1本それぞれ丸めるのではなく、全体をそうめん束のように配線経路内に垂らします(**写真4**)。あとから1本追加／撤去する場合は、機器ケーブルの束をいったんほどいて、機器ケーブルを追加／撤去します。束にする際は、ケーブル自重(ケーブル自身の重さ)で余長(余った長さ)やコネクタプラグ部が垂れ下がらないように、接続点根元付近は必ず重さを受ける工夫をしましょう。具体的には整線パネルで左右方向の流れを作る、もしくは、大きく曲げを作ってから垂らすなどがあります。

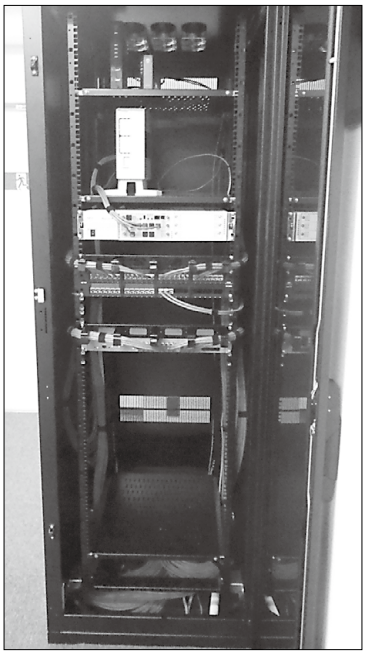
▼図3　ラック内機器マウント例



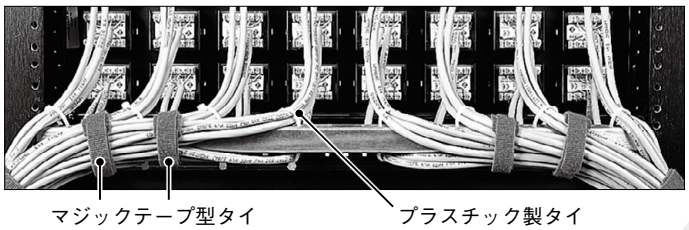
このとき、垂らしたコードは空気の流れを妨げないようにマジックテープ状のケーブルタイで、ところどころ留めます(**写真5**)。通信ケーブルを固定するときには、プラスチック製の使い切りのケーブルタイを使うことは望ましくありません。

プラスチック製のケーブルタイの場合、縛るために力を入れ過ぎて、ケーブル外被が変形しやすいのが難点です。また、変形させそうになっても、速やかにやりなおしがききません。新しいケーブルタイを用いることになるため、結果的にケーブルタイの無駄使いになったり、ケーブルタイを切るときにケーブル外被を傷つけた

▼写真4　ラックのマウントフレーム外でコードを垂らす[3]



▼写真5　マジックテープ型タイとプラスチック製タイによるケーブル捕縛[4]



りします。そもそも縛ること自体をやりなおそうという気力が起きにくいのも確かです[注2]。

したがって、外被が変形しないように押さえるには、マジックテープ型のケーブルタイを利用したほうが良いのです[注3]。縛るためのツールはいくつか種類や色をそろえ、機器用、ケーブル用など適材を適所に選択できると良いですね。

通信ケーブルをきつく締めつけてはいけないのはなぜでしょう？　外被が変形するほど締めつけた状態にすると、ツイストペアケーブルの場合は、内部の銅線の導体の特性が変化し、機器間の伝送品質の低下の要因となります。光ファイバの場合は、きつく締めつけることで、内部の光ファイバに微細な曲げが発生し、光が全反射できず損失が増加することが考えられます。

また、ツイストペアケーブル、光ケーブルともに締めつけられた状態が長く続くと内部の導体もしくは光ファイバに圧力がかかり続けることになるため、最悪、破断してしまう恐れもあります。このような通信障害を引き起こす可能性のある配線はしてはいけません。また、無理な曲げやひねった状態での固定も特性を劣化させる要因となるので、避けてください。

機器ケーブルの選び方のコツ

通信ケーブルの配線にあたって、ちょっとしたコツがあります。自分たちの配線規模に合わせて、選択できるとベストですね。

ケーブル質感（硬さ）

第1章でも説明しましたが、機器間や、機器とパッチパネル間は「より線」の機器ケーブルを使って配線します。この機器ケーブルはしなやかで柔らかい質感を持ちます。そのため、曲げ癖がつきにくく、ラック内の配線には適しています。

さらに少し高度な考え方ですが、同じより線の機器ケーブルであっても、メーカーによってその硬さは違います。細径線と通常線でも硬さが変わってきます。硬めの機器ケーブルは、サーバとスイッチ間、サーバとストレージ間などを配線するときに利用すると、ケーブル類が垂れ下がらず、メンテナンス性が向上する場合があります。また、柔らかめの機器ケーブルは、集線用パッチパネルのように配線が密集している場所では、配線の型が作りやすいため、ラック内できれいに収めることができます。

色分け

利用用途ごとにコードの色やコネクタに付ける印などで色分けする場合があるかと思います。見た目にもわかりやすいため、保守運用性に優れているかと思われます。しかし、用途ごとにあまりにも細かく色分けした場合、構築当初はきれいに見えても、運用が始まると「ケーブルの在庫量が多い」、「構築当初と用途が変わってしまったポートであるにもかかわらず、ケーブル色は構築当初の用途のままで放置されている」という問題が起きることが少なくありません。そうなると、用途ごとに色分けしたことが、かえって運用上のリスクに変わってきてしまう恐れもあります。ここでは、次のような考え方で分類する必要があります（詳しくは後述しますが、ラック内にケーブルを固定するときも同じような考え方で分類します）。

- 用途の変わらない（もしくは用途が変わるときはシステム運用を再度見直す必要がある）「固定ケーブル」
- 都度用途が変わる、もしくは臨機応変に対応できる「変更ケーブル」
- この2つに属さない、もしくはどちらかの用途をさらに詳しく分ける

都合、ケーブルの色は3系統の色（赤系／黄系／青系）に絞り、その中で用途を定めることが望

注2）プラスチック製のケーブルタイは、通信ケーブルを縛ることには向きませんが、機器や金物を固定させるときには、しっかり固定できるため、非常に便利です。
注3）マジックテープ型であっても、外被が変形するほど縛れば、プラスチック製のケーブルタイと同様の事故が発生するので注意しましょう。

ましいようです。なぜ3系統に絞るかというと、3系統程度であれば、どのケーブルメーカーであっても提供してもらえる色だからです。ケーブルメーカーを自由にしておくことで、納期どおりに入手したり、都度、各社の価格を確認しながら一番安価なケーブルを購入したりできます。

配線のコツ

ラック内で同じ経路を配線する機器ケーブルであっても「一度配線したら、ラック内大改造がない限り接続したままにする機器ケーブル(固定束)」と、「増加／撤去が見込まれる機器ケーブル(変化束)」に分け、それぞれ別束にして束ねると、より安全に運用することが可能となります。このとき束ねるケーブルタイの色を変えると、運用上よりわかりやすくなります。

どちらの場合も、それぞれがレイヤになるように配線することです。話は飛びますが、みなさんはプレゼン資料などを作成する際、ベースのスタイルの上に、図形のレイヤ、文字のレイヤなど、分けて作成していらっしゃるかと思います。それと同様に、ラック本体のレイヤ、設置する機器のレイヤ、電源ケーブルのレイヤ、通信ケーブルのレイヤと分けます。これは、それぞれのレイヤが独立していることを示します。

さらに、通信ケーブルのレイヤは「固定束」と「変更束」に分かれるように配線します。接続するポートの位置によっては、どうしてもケーブルが絡む場合もありますが、そのときも極力絡まないように意識しながら配線します。このときに、ケーブルごとの色で分けられるようにしておくことが望ましいです。たとえば、「固定束」は「上位のスイッチとの接続用のケーブル」や「いったん配線したら変更しないと想定しているケーブル」とします。「変更束」は「今後、増減が発生する可能性のあるケーブル」あるいは「機器マネジメント用のケーブル」とします。それぞれの色が独立したレイヤになるように配線を分けます。

そして、「固定束」のケーブルはより機器に近い側に配線、「変更束」はラック扉に近い側に配線するときれいにレイヤを分けることができます。さらに、それぞれ独立して色の違うマジックテープで捕縛しておけば、お互いが絡まず、増減するケーブルだけ、スムーズに施工することができるかと思います。

この2種類の機器ケーブルは、ラックの中の配線設計だけでなく、構築しようとしているネットワーク、サーバ、ストレージの設計にも深くかかわってくる部分です。

ケーブリングは、機器設定、マウント時点から始まる

レイヤを分けて配線できるようにするには、じつは「スイッチのどのポートに何を設定するのか？」ということにも配慮する必要があります。アップリンク[注4]のポートとダウンリンク[注5]のポート、その位置。単に機器をラックに適当に搭載するのではなく、(できればラック内の拡張性までも考慮した)配線を考えた設置位置を決めることが望ましいのです。そうすることで、単にきれいな配線になるだけでなく、メンテナンス性にも優れ、耐障害特性も向上します。SD

■参考

[1] TechVirtuoso, LLC.
http://techvirtuoso.com/2010/11/08/cable-management-and-you/
[2] Panduit社ショールーム
http://www.panduit.co.jp/ebc/
[3] HigherNet社工事事例
http://higher.co.jp/
[4] ザ・シーモン・カンパニー
http://www.siemon.co.jp/jp/
[5] LANケーブル配線.COM
http://www.lan-cabling.com/

注4) 通信回線の上り方向(サーバやストレージから、スイッチ・WANへ向かう方向)のこと。
注5) 通信回線の下り方向(スイッチ・WANから、サーバやストレージへ向かう方向)のこと。

Appendix 1

膨大なケーブルを効率よく管理するために

配線管理と誤抜防止に役立つツール

ラックでサーバを構築すると、ケーブルの数も膨大なものになります。数が多いと、構成を管理するのはたいへんですし、ケーブルの抜き差しをするだけでもリスクが伴います。本稿で紹介する便利なツールを活用して、効率よく管理／運用を行いましょう。

Author 佐伯 尊子（さえき たかこ）　（株）ブロードバンドタワー

配線管理の課題

パッチパネルと機器、機器間を接続する通信ケーブルは、どうしても本数が多くなります。その1本1本を管理することは、簡単なようで実は面倒くさい作業でもあります。機器間の接続は、いくつかのパッチパネルを介して両端の機器のポートに接続しています。しかし、それを管理するときには、途中のパッチパネルと機器のポートをそれぞれ輪切りの情報としてExcelで管理していることが多いようです。

パッチパネルや機器のポート数が少ない場合は、それでも問題ないのですが、パッチパネルの数、機器数、ポート数が増えてくると、ポート1つから両端の機器までどのポートを介して接続されているかを確認するのは難儀します。

さらに、光ファイバになると、2芯一括で送受信するため、さらに困難を極めます。また極性を十分管理できていないと、どちらが送信側でどちらが受信側か、もうわからなくなってしまいます。

配線／ラック管理ツール「openDCIM」

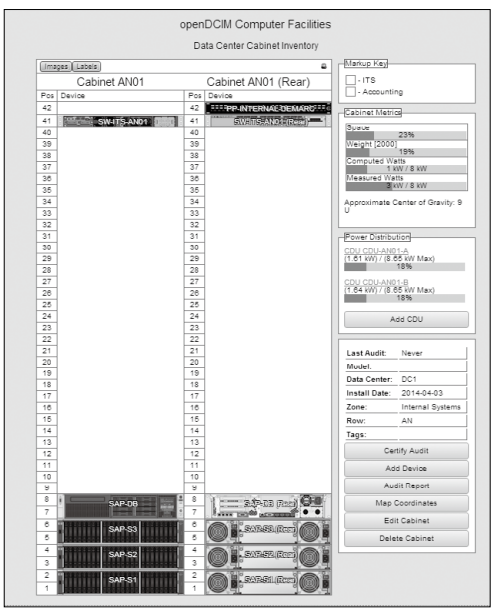
そこで、ポート1つから両端の機器がわかり、さらにどのポートと接続されているかを一目で管理できるツールを紹介します。「openDCIM」というOSS（オープンソースソフトウェア）です（**図1、2、3**）[注1]。

このツールの優れているところは、ポート図（**図2**）、結線図（**図3**）が充実していることです。これによって、機器のどのポートからどのポートまで、どのパッチパネルのどのポートを通って配線されているかがわかります。

また、入力／編集にExcelを利用されている方も多いかと思います。ただ、Excelを複数の

注1）http://www.opendcim.org/

▼図1　openDCIMのラック図



方が管理／運用されている場合、「元データをサーバに保管し、編集作業は手元PCで実施」したために、「(アップロードし忘れて)最新版がどこにあるのかわからない」「(複数の人が)同時進行で作業してそれぞれ最新版ができてしまった」など、履歴管理をしっかりしていないと、うまく運用できません。本ツールを使えば、そのような手間も省けます。さらに、インポート／エクスポート機能でExcelをサポートしているため、Excelとの親和性が高いところもポイントです。

このように、誰でも簡単に更新できる管理ツールを使うことで、「あとでやろう」「まとめてやろう」「完成したらやろう」という行為を防ぎ、最新の状態を常に管理することができるようになります。この「(誰が見ても)いつでも最新の状態」に一元管理しておくことが、配線管理(を含むラック管理)で大切なことです。

誤抜防止

また、このように管理された配線であっても、パッチパネルや機器のポートは有限資源です。この資源を有効に活用するためには、どうしてもケーブルを抜去(ばっきょ)する必要が出てくるかと思います。すでに一部のデータセンターでは、機器コードやフロア配線のケーブルも撤去している例が増えてきました。

その際に、間違って利用しているケーブルを抜いてしまう事故も少なくありません。管理不十分で指示した図面と、実際のポートが違っている場合も多いのですが、そのときは抜去する前にもう一度確認ができます。しかし、指示図面どおりに抜去したつもりで、別のポートを抜いてしまう恐れがあります。それを防止するために、専用のツールがあります(**写真1**)。

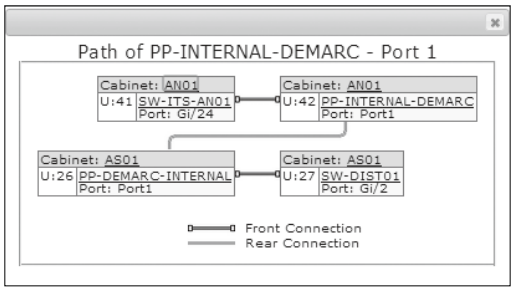
このツールを挿入して機器コードを接続しておけば、「指示図面を見ながらいったんツールを外して、再度確認し、問題なければ本当に抜去する」というように、ステップを踏むことによって、誤抜去を防止します。わざわざツールを抜くステップがあるのは面倒な気もしますが、このようなひと手間をかけることで、より信頼度の高い配線システムを構築することができます。SD

▼図2　openDCIMのポート図

Connections

#	Port Name	Device	Device Port	Notes	Status	Media Type	Color Code
1	Gi/1	SAP-DB	ILO				
2	Gi/2	SAP-DB	Net1				
3	Gi/3	SAP-DB	Net2				
4	Gi/4						
5	Gi/5						
6	Gi/6						
7	Gi/7						
8	Gi/8						
9	Gi/9						
10	Gi/10						
11	Gi/11						
12	Gi/12						
13	Gi/13						
14	Gi/14						
15	Gi/15						
16	Gi/16						
17	Gi/17						
18	Gi/18						
19	Gi/19						
20	Gi/20						
21	Gi/21						
22	Gi/22						
23	Gi/23						
24	Gi/24	PP-INTERNAL-DEMARC	Port1				
	SW-ITS-AN01[Gi/24] -> PP-INTERNAL-DEMARC[1] -> PP-DEMARC-INTERNAL[Port1] -> SW-DIST01[Gi/2]						
25	Te/1						
26	Te/2						

▼図3　openDCIMの結線図



▼写真1　誤抜去防止ツール(Panduit製)

Appendix 2

通信とともに給電も行える

PoEのしくみと機器選定の注意点

1本のケーブルで通信と給電の両方を行えるPoE。便利ではありますが、接続する機器同士の対応がきちんととれている必要があります。機器選定の際には、どこに注意すべきか、本稿で整理しましょう。

Author 佐伯 尊子（さえき たかこ）　（株）ブロードバンドタワー

PoE、PoE+

PoE（Power over Ethernet）技術は、電力供給機器（PSE：Power Sourcing Equipment）と電力需給機器（PD：Power Device）の間をツイストペアケーブルで接続し、ケーブルにデータと電圧を送信するしくみです。

この方式はIEEE802.3at「PoE+」（ピーオーイープラス）にて、2009年に規格化されています。もともと2003年に、IEEE802.3af「PoE」（ピーオーイー）として規格化されていたのですが、そのときの規格値を見直した形でPoE+が制定されました。PoEの規格は、今はPoE+の一部として残っています。

そもそも、なぜ通信線に電力も併せて送信しようと考えたのでしょうか？　それは、ネットワーク機器が至るところに利用され始めたことから、電源の供給方法が難しくなってきたためです。とくに無線LANのアクセスポイント（以下、AP）やIPカメラなどは、「最適な場所に設置したい」「電源ケーブルの取り回しで制限されたくない」というニーズから生まれました。また、IP電話などは、1つの端末からツイストペアケーブルも電源線も両方必要になると、机上で邪魔になったり、見栄えが悪かったりと、見た目の問題もありました。

そこで、給電（電気を供給すること）できるデータ通信を考える必要が出てきました。それがPoEとなりました。そして、当初は15.4Wという供給能力があれば大丈夫と思っていたPoEですが、それでは駆動できない機器が出てきたため、さらに大電力が伝送できるようPoE+が規格化されました。これは、無線LANの速度の進化がEthernet以上に早いため、その速度を供給する無線LANのAPには大電力を要してしまうからです。そのため、PoEの普及が本格化した2007年以降、あっという間にPoE+の規格が制定されました。

PoEユースケース

最初に、PoEのユースケースを確認しましょう。

PoEの給電方法は、大きく2つに分けられます。1つは「エンドポイント型」、もう1つが「ミッドスパン型」です（**図1**）。

エンドポイント型は、両端の機器がPoEに対応している場合を示します。現在は、ほとんどがこの形になっています。

ミッドスパン型は端末側のみPDに対応していますが、ハブやスイッチ側はPoEに対応していない場合のケースです。最近はPoE対応のスイッチ製品が増えたため、機能面から選択でき

るようになってきましたが、PoEの提供当初（2003～2007年頃）は、既存のPoE非対応スイッチを利用することも多かったため、場合によってはこのミッドスパン型も利用されていました。

それでは、このPoEでどのくらいの電力がまかなえるのか？　具体的に見ていきましょう。

PoEの規格内容

表1に、PoE規格の内容を示します。

PoE時代は、カテゴリ3のケーブルも利用できましたが、最大で15.5Wまでしか PSE側で電力を消費できませんでした。

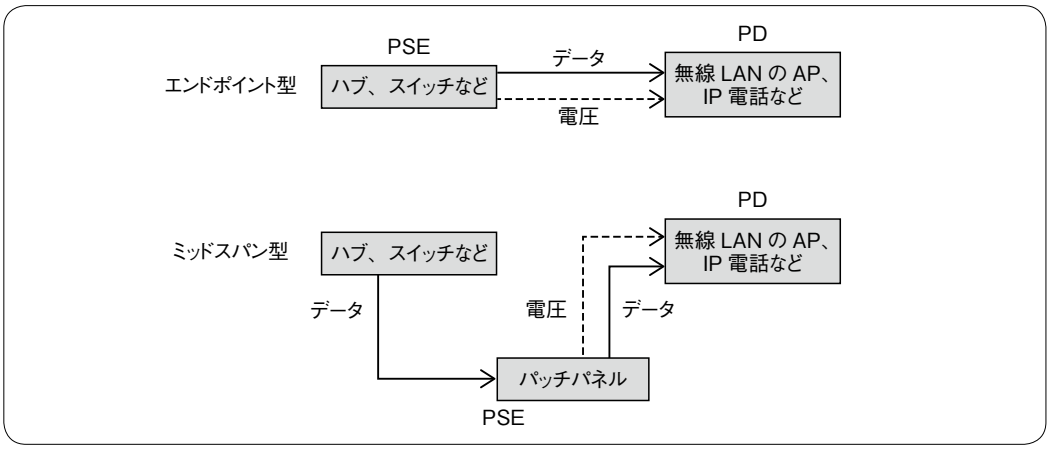
しかし、PoE+では、ケーブルをカテゴリ5e以上と制限したことによって、PSE側で最大で30Wまで電力を使えるようになりました。2009年には、すでにカテゴリ3を使ってデータ伝送しているLANはほとんどありませんでしたので、実害はありませんでした。

PoEのしくみ

利用するケーブルは、データ伝送で利用しているツイストペアケーブルです。ツイストペアケーブルは4対8芯からなりますが、PoEではそれぞれがどのような役割をしているかを**図2**に示します。

線はツイストペアケーブルのそれぞれの対を示します。10M/100MのEthernetでは、4対のうち、ペア2とペア3を使ってデータ伝送しています。最初にPSE側に戻ってくる電流値を測定して、PD側がPoEに対応しているかなどを

▼図1　PoE給電方法



▼表1　PoE規格

規格		IEEE802.3at（Type 1） IEEE802.3af （PoE）	IEEE802.3at（Type 2） （PoE+）
PSE（給電）	電圧	44～57V	50～57V
	最大消費電力	15.5W	30W
PD（受電）	電圧	37～57V	42.5～57V
	最大消費電力	13W	25.5W
利用可能ケーブル		カテゴリ3、カテゴリ5e、カテゴリ6	カテゴリ5e、カテゴリ6
サポートモード		Mode A（エンドスパン）、 Mode B（ミッドスパン）	Mode A（エンドスパン）、 Mode B（ミッドスパン）
使用するツイストペア		2対	2対

把握してから、最適な電圧を供給するしくみです。

データ伝送の対に電圧も負荷させる方法を「Mode A」、データ伝送の対を使わず、空き対に電圧を負荷させる方法を「Mode B」と呼びます。「最初にPSE/PDの機器であるかどうか確認する」作業のときに併せてMode AもしくはMode Bも確認します。メーカーによっては、クロスケーブルとストレートケーブルを判別し、適切にModeの統一が取れるよう切り替えスイッチをPSE側に内蔵しているタイプもあります。

また、GbE伝送する場合は、4対すべてをデータ伝送に利用しているので、Mode AであってもMode Bであっても、常に2対はデータ＋電圧がかかるしくみになります。

さらに、PoEを利用している状態でケーブルが抜けた場合は、ただちに給電を停止するしくみも有しています。

PoE機器選定の注意点

このように、PoEの規格に則った機器間であれば、私たちが細かいところまで気にしなくても自動で最適な電圧をかけるしくみを有しているため、急速に利用が広まりました。

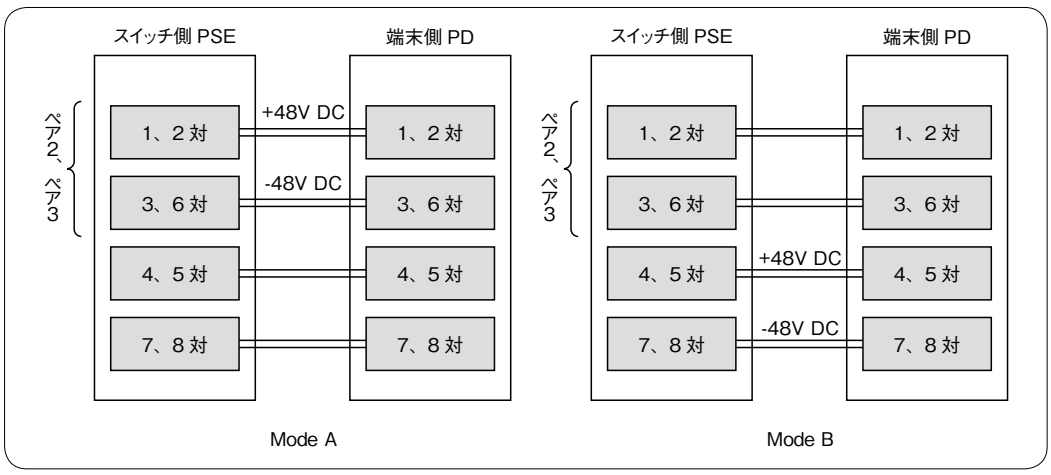
ただ、実際に販売されている機器を見ると「規格に準拠」しているものの、「規格どおり」の機器だけではないようです。自社独自仕様を盛り込むことで、さらに大電力に対応できるなどのメリットを享受できることもありますが、「規格に準拠」した無線LANのAP（PD）を購入すると、「実は自社のスイッチ（PSE）を使わないと通信できません」ということもあります。

ベンダロックがかかった状態になると、「自社のスイッチではネットワークの別の仕様が自分たちの要求を満足しない」など、あちらが立てばこちらが立たずになる可能性もあります。もちろん満足のいく仕様であれば、ベンダロックがあっても良いのですが、自由度が高く機器を利用したい場合は、まずカタログベースで規格との整合性をしっかり確認してから購入することをお勧めします。SD

■参考

- IEEE P802.3af DTE Power via MDI Task Force
  http://www.ieee802.org/3/af/
- IEEE P802.3at DTE Power Enhancements Task Force
  http://www.ieee802.org/3/at/

▼図2　ツイストペアケーブルの信号と電圧



バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

本誌バックナンバー（紙版）はお近くの書店に注文されるか、下記のバックナンバー常備取り扱い店にてお求めいただけます。また、「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）や、e-hon（http://www.e-hon.ne.jp）にて、Webから注文し、ご自宅やお近くの書店へお届けするサービスもございます。

2016年1月号

第1特集
はじまっています。ChatOps
導入を決めた7社の成功パターン

第2特集
手軽さとコード化しやすさが人気!
Ansibleでサーバ管理構成を省力化

新連載
・Androidで広がるエンジニアの愉しみ

定価（本体1,220円+税）

2015年12月号

第1特集
［決定版］Docker自由自在
実用期に入ったLinuxコンテナ技術

第2特集
ネットワーク・システム管理の定石
SNMPの教科書

短期連載
・クラウド時代のWebサービス負荷試験再入門

定価（本体1,220円+税）

2015年11月号

第1特集
すいすいわかるHTTP/2
HTTP/1.1から変わること・変わらないこと

第2特集
攻撃を最前線で防ぐ
ファイアウォールの教科書

特別企画
・SMB実装をめぐる冒険　File System for Windowsの作り方

定価（本体1,220円+税）

2015年10月号

第1特集
多層防御や感染後対策を汎用サーバに実装
攻撃に強いネットワークの作り方

第2特集
Webメールの教科書
クラウドサービス利用か？　自社で構築か？

特別付録
・創刊300号記念　Vim&Emacsチートシート

定価（本体1,220円+税）

2015年9月号

第1特集
特講
正規表現・SQL・オブジェクト指向
苦手克服のベストプラクティス

第2特集
メールシステムの教科書
日本語もバイナリもちゃんと届くのはなぜか

特別企画
・なぜ俺の提案は通らないのか?

定価（本体1,220円+税）

2015年8月号

第1特集
Lispより始めよ、されば救われん!
なぜ関数型プログラミングは難しいのか？

第2特集
安全な通信を確保する
SSL/TLSの教科書

短期連載
・AWSで始めよう!　モダンなJavaアプリケーション開発

定価（本体1,220円+税）

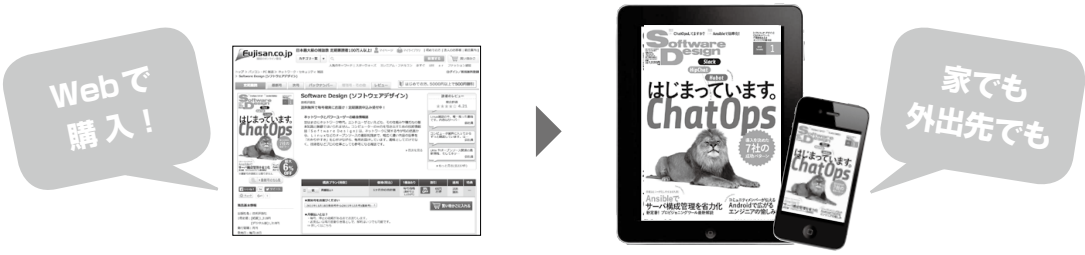
Software Design　バックナンバー常備取り扱い店							
北海道	札幌市中央区	MARUZEN＆ジュンク堂書店　札幌店	011-223-1911	神奈川県	川崎市高津区	文教堂書店　溝の口本店	044-812-0063
	札幌市中央区	紀伊國屋書店　札幌本店	011-231-2131	静岡県	静岡市葵区	戸田書店　静岡本店	054-205-6111
東京都	豊島区	ジュンク堂書店　池袋本店	03-5956-6111	愛知県	名古屋市中区	三洋堂書店　上前津店	052-251-8334
	新宿区	紀伊國屋書店　新宿本店	03-3354-0131	大阪府	大阪市北区	ジュンク堂書店　大阪本店	06-4799-1090
	渋谷区	紀伊國屋書店　新宿南店	03-5361-3315	兵庫県	神戸市中央区	ジュンク堂書店　三宮店	078-392-1001
	千代田区	書泉ブックタワー	03-5296-0051	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	千代田区	丸善　丸の内本店	03-5288-8881		広島市中区	丸善　広島店	082-504-6210
	中央区	八重洲ブックセンター本店	03-3281-1811	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322
	渋谷区	MARUZEN＆ジュンク堂書店　渋谷店	03-5456-2111				

※店舗によってバックナンバー取り扱い期間などが異なります。在庫などは各書店にご確認ください。

デジタル版のお知らせ　DIGITAL

デジタル版Software Design 好評発売中！紙版で在庫切れのバックナンバーも購入できます

本誌デジタル版は「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）と、「雑誌オンライン.com」（http://www.zasshi-online.com/）で購入できます。最新号、バックナンバーとも1冊のみの購入はもちろん、定期購読も対応。1年間定期購読なら5%強の割引になります。デジタル版はPCのほかにiPad／iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく、読むことができます。



まだEclipseぐせが残ってませんか?

Android Studioのスタイルで効率アップ!

Author 有山 圭二(ありやま けいじ) ㈲シーリス マンガ 根雪れい

2015年12月でAndroid Developer Tools(ADT)のサポートが正式に終了しました。ADTから移行したばかりだと操作に戸惑うばかりで「Android Studioの良さがわからない」と思う人も多いのではないでしょうか。本稿では最近ADTから移行したばかりの人に向けて、アプリ開発を効率良く行うためのAndroid Studioの活用方法を解説します。

ADTとAndroid Studio

はじめに、ADTとAndroid Studioについて軽くおさらいをしておきましょう。

ADTは、2007年にAndroidとSDKが発表されたときから使われていた**統合開発環境**(IDE)です。当初はEclipse用のプラグインをインストールするという形式で配布されていましたが、2010年ごろからは「ADT」というEclipseをベースにした(プラグインをインストール済の)ソフトウェアとして配布されるようになりました。Android Studioのバージョンが1.0になったと同時に、ADTの配布は停止されました。また、サポートも2015年12月末で終了しています。

一方、Android Studioは2013年にプレビュー版が発表された新しいIDEです。2015年1月にバージョンが1.0になり、現在はAndroidアプリ開発用のIDEとして公式に推奨されています。

Android Studioのベースは、JetBrains社が開発しているIntelliJ IDEAのオープンソース版(Community Edition)です。ADTとはキーマップ(ショートカット)がまったく違うので、移行した直後は戸惑うかもしれません。一方、IntelliJ IDEAを使っていた方はショートカットがほぼそのまま使えます[注1]。

注1) https://www.jetbrains.com/idea/docs/IntelliJIDEA_ReferenceCard.pdf
https://www.jetbrains.com/idea/docs/IntelliJIDEA_ReferenceCard_Mac.pdf

Android Studioで開発の効率アップ

Android Studioは、ADTの代わりというわけではありません。さまざまな便利な機能が組み込まれていて、使いこなせば開発の効率をアップできます。ここからは、ADTにはなかった(ADTより優れた)、アプリ開発の効率をアップするAndroid Studioの機能について紹介します。

開発環境を共通/最新に保つ

ADTでライブラリを使う場合、jarファイルをプロジェクト内に置く必要がありました。Android Studioが採用しているビルドシステム「Gradle」には、ライブラリの名前とバージョンを指定するだけで、ビルド時に必要なライブラリをダウンロードするしくみがあります。プロジェクト内にjarファイルを持たず、さらにバージョンを固定することができるので、複数人数で開発する場合に便利です。

直接指定する

アプリが使うライブラリをGradleのビルドファイル「build.gradle」に指定します。build.gradleファイルは、Android Viewの場合はGradle Scriptsの下に(**図1**)、Project Viewの場合はアプリのモジュールの下に(**図2**)それぞれ表示されます。

build.gradleを開いて、後半にあるdependenciesがライブラリの指定です(**リスト1**)。たと

えば、Jake Warton氏の開発しているライブラリButterKnifeを使う場合、

```
com.jakewharton:butterknife:6.1.0
```

のように記載します。

build.gradleの変更をAndroid Studioが検知すると「Sync(同期)」の確認を上部バーに表示します。ここで同期しないとライブラリは使えないので注意してください。初めて使うライブラリの場合、インターネット上のリポジトリからファイルをダウンロードするため時間がかかることがあります。

なお、例ではバージョンに6.1.0を指定していますが、ButterKnifeの最新バージョンは7.0.1です。ButterKnifeはバージョン6と7で大幅な変更が加わり、最新バージョンを使うには既存のコードを変える必要があります。このように、コードの互換性を維持するため、あえて古いバージョンのライブラリを指定することもできます。

検索して追加する

ライブラリの名前を正確に覚えていなくても、検索して追加できます。Android Studioの左側に表示されているモジュール(app)を右クリックして[Open Module Settings]を開き、上にある「Dependencies」タブをクリックすると、現在のライブラリの一覧が表示されます(**図3**)。

下の「+」記号をクリックして「Library Dependencies」を選択すると、検索画面が開きます(**図4**)。

テキストボックスにライブラリの名前を入力します。「butterknife」と入力して検索ボタンを押すと「com.jakewharton:butterknife:7.0.1」が該当します。目的のライブラリを選択・決定すると、ライブラリがモジュールで利用可能になります。

build.gradleから設定した場合と同じで、インターネット上のリポジトリからライブラリをダウンロードするため、ビルドに時間がかかることがあります。

サポートライブラリのバージョンアップの通知

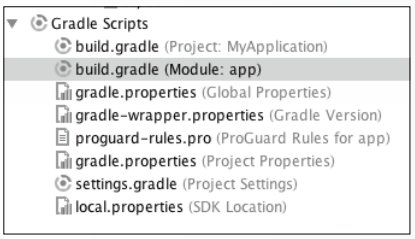
Androidのバージョン間の差を埋めるサポートライブラリは、今やAndroidアプリ開発には欠かせない存在となっています。プロジェクトの新規作成時や、あらかじめ用意されているテンプレートを使う場合でも、サポートライブラリがdependeciesに追加されます。

このサポートライブラリは更新されることがあります。理由はさまざまですが、バグが修正されたり新しい機能が追加されたりします。ADTでは、サポートライブラリの更新はSDK Managerを立ち上げてチェックしなければわかりませんでした。

Android Studioは、SDK Managerと同等の機能が統合されており、サポートライブラリの更新を自動でチェックして、更新があった場合は通知します(**図5**)。

またプロジェクトやモジュールが使っているサポートライブラリ

▼図1　Android Viewの場合

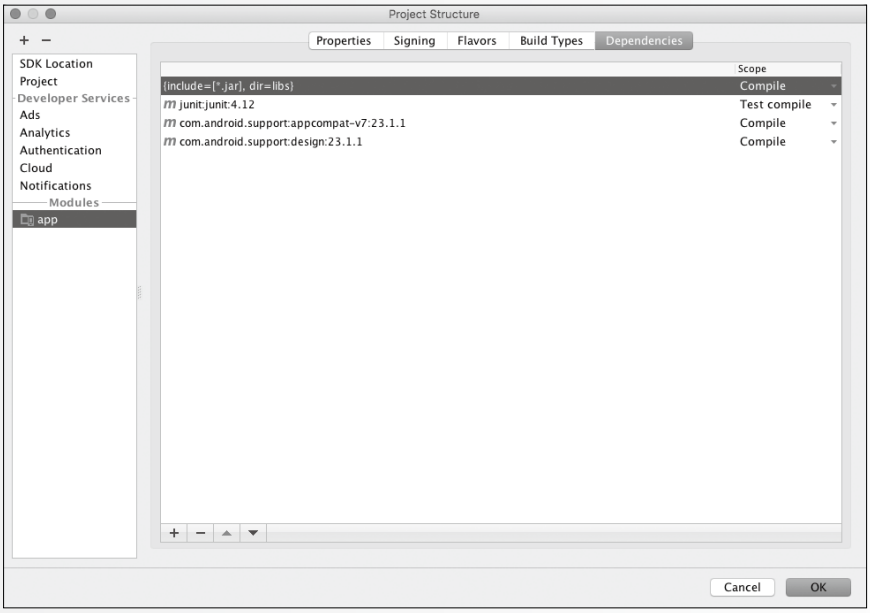


▼図2　Project Viewの場合

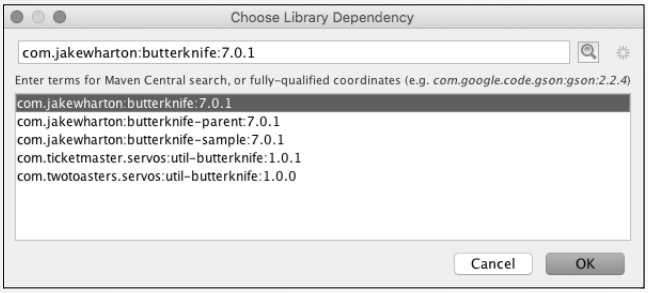
▼リスト1　ライブラリの指定個所

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
}
```

▼図3　現在のライブラリー覧



▼図4　検索画面



▼図5　サポートライブラリ更新の通知(Android Studio内)

やビルドツールのバージョンが古い場合にも、Android Studioが指摘します(もちろん、ほかのライブラリ同様、古いバージョンをそのまま使い続けることもできます)。

文字列リソースの翻訳管理

文字列リソースと翻訳の管理は頭の痛い問題です。最初のうちは文字列リソースのファイルを細かく分割したり、すべての言語の文字列リソースの並び順を同じにしたりして対応するのですが、変更を繰り返すうちに秩序を保つことが難しくなってきます。

また多言語に翻訳する場合を考えてみると、どの文字列を翻訳してどれが翻訳されていないのか。ファイルを見比べて比較するには多大な労力を使います。

Android Studioでは、文字列リソースを管理する専用のエディタが組み込まれています。文字列リソースを開いた状態で[Open Editor]をクリックすると「Translation Editor」が表示されます(**図6**)。

左上の地球のマークをクリックして、追加したい言語を選択します。リストに表示される言語の種類が多い場合、言語の名前をキーボードから入力して絞り込むことができます。

Translation Editorは、翻訳が必要な文字列リソースを赤字で表示します(**図7**。誌面ではKey列の薄くなっている文字列が該当)。文字列リソースを含むファイルが複数あっても統合し

た状態で表示するので見逃す心配はありません。

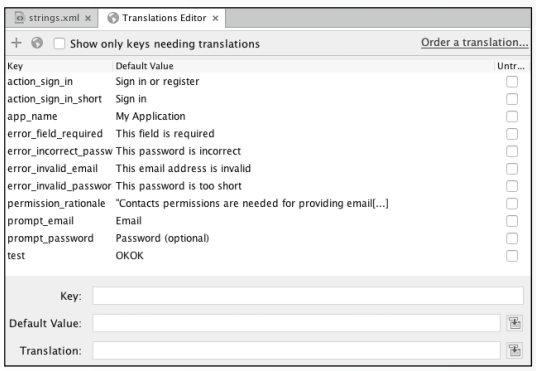
また、翻訳をしない場合は「Untranslatable」のチェックボックスをチェックすることで、翻訳の対象から除外できます。

Parcelableの実装

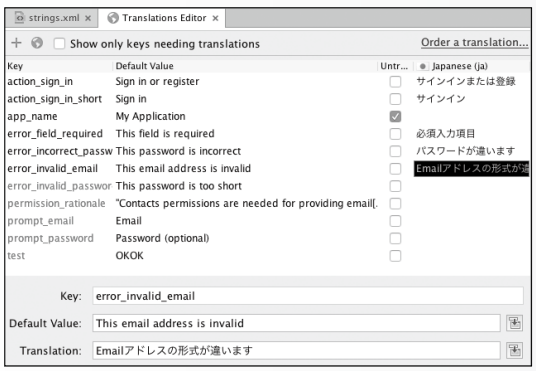
ServiceなどでAIDLを使う場合に、やり取りするクラスにParcelableインターフェースを実装する必要があります。しかし、Parcelableでやり取りするには、インターフェースにより強制されるメソッドだけでなく、Parcelable.Creatorをpublic staticで保持しなければならないなどの規則があります。このようなボイラープレートなコードも、Android Studioなら自動で生成します。

まずクラスUserDataにParcelableをimplementsして、必要なメソッドを自動生成します。この時点では、メソッドは空で構いません。

▼図6　Translation Editor



▼図7　文字列リソースの翻訳状態を表示



次にマウスカーソルをUserDataに合わせて右クリックし、表示されるメニューから[Add Parcelable Implementation]を選択すると、Android Studioが現在のクラスのフィールドに応じて必要な処理を実装します(**図8**)。

リスト2が生成されたParcelableです。UserDataクラスのフィールドemail、password、authorizedの3つについて、きちんと書き込みと読み込みの処理が追加されています。このように、Android Studioを使えば、AIDLを使ううえで煩雑なコードの入力を省略できます。

利用していないリソースを自動的に削除する

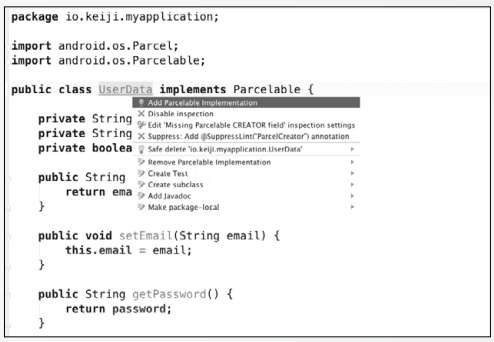
アプリの変更を繰り返すと、使っていないリソースが増えてきます。とくに画像リソースは数が多く、APKのサイズを増大させるので、利用していない画像リソースは消す必要があります。しかし、リソースはさまざまな形で参照される可能性があり、本当に使っていないか、1つ1つ確認していくのは大がかりで根気のいる作業です。

Android Studio(Android plugin for Gradle)には、使われていないリソースファイルをビルド時に除外する機能があります。

リスト3のようにbuild.gradleにminifyEnabledとshrinkResourcesをtrueに設定すると、ビルド時に利用していないリソースを検知してAPKから除外します[注2]。

注2) 正確には画像は1×1の最小サイズに、そのほかの容量の大きなファイルは0バイトの空ファイルとしてリソースに残ります。文字列やレイアウトなど画像と比べてサイズの小さいリソースには影響ありません。

▼図8　Parcelableの生成をメニューから選択



▼リスト2　自動生成されたParcelableの例

```
@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(email);
    dest.writeString(password);
    dest.writeByte((byte) (authorized ? 1 : 0));
}

protected UserData(Parcel in) {
    email = in.readString();
    password = in.readString();
    authorized = in.readByte() != 0;
}

public static final Creator<UserData> CREATOR = new Creator<UserData>() {
    @Override
    public UserData createFromParcel(Parcel in) {
        return new UserData(in);
    }

    @Override
    public UserData[] newArray(int size) {
        return new UserData[size];
    }
};
```

▼リスト3　使われていないリソースファイルを除外する設定

```
buildTypes {
    debug {
        minifyEnabled true
        shrinkResources true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

▼表1　代表的なビルドタスク

Build Task	解説
assembleDebug	デバッグ版のビルドをする
assembleRelease	リリースビルドをビルドする。「Lint」によるコードチェックなど、リリースに必要な処理が追加されている
installDebug	デバッグビルドしたAPKを接続しているAndroid端末(エミュレータ)にインストールする
connectedAndroidTest	接続しているAndroid端末(エミュレータ)でプロジェクトのユニットテストを実行する
clean	ビルドに関係するファイルが出力されるbuildディレクトリを削除する
tasks	利用可能なビルドタスクを一覧表示する

コマンドラインからビルドを実行する

ADTのビルドシステムはADT(Eclipse)独自のものでした。オプションでAntのビルドファイルを生成することでコマンドラインからビルドすることもできましたが、ADTとAntのビルド設定はそれぞれ独立していて、両者を共通化するには非常な手間がかかっていました。

一方、Android Studioはビルドシステムに「Gradle」を採用しているので、標準でコマンドラインからアプリをビルドすることができます。試しに、コマンドラインからビルドを実行

してみましょう。

コマンドプロンプトからプロジェクトのディレクトリに移動して、ビルドを実行します(**図9**)。

これだけで、アプリケーションのAPKファイルがapp/build/outputs/apk/の下に生成されます。gradlewは、gradle wrapperと呼ばれるもので、Windows/Mac OS X/Linuxの環境の差を埋めるためにあります。続くassembleDebugは「ビルドタスク」で、デバッグビルドを実行するという意味になります。

ビルドタスクを指定することでさまざまなビルドを実行できます。指定できるビルドタスクはbuild.gradleの記述によって変わりますが、代表的なものを**表1**に掲載します。

コマンドラインから標準でビルドできると、今では当たり前になったCI(Continuous Integration)の導入が非常に簡単になります。また、Android Studioのビルドとコマンドラインからのビルドは共通のビルドファイル(build.gradle)で定義できるので、ADTとAntのように、それぞれで固有のビルドの設定をメンテナンスする必要もありません。

最新のAndroid Studioを使ってみよう

ここからは少し趣を変えて、Androidの開発バージョンについて解説していきます。

読者の皆さんは今、どのバージョンのAndroid Studioを使っていますか？　Android Studioは、大きく2種類のチャンネルで提供されています。

Stableチャンネル

その名のとおり最も安定したバージョンです。通常のサイト[注3]からダウンロードすると、このチャンネルのAndroid Studioになります。

▼図9　コマンドプロンプトからのビルド実行例

```
$ ./gradlew assembleDebug↵
Parallel execution is an incubating feature.
:app:preBuild UP-TO-DATE
:app:preDebugBuild UP-TO-DATE
:app:checkDebugManifest
:app:preReleaseBuild UP-TO-DATE
(省略)
:app:packageDebug
:app:zipalignDebug
:app:assembleDebug

BUILD SUCCESSFUL

Total time: 19.408 secs
```

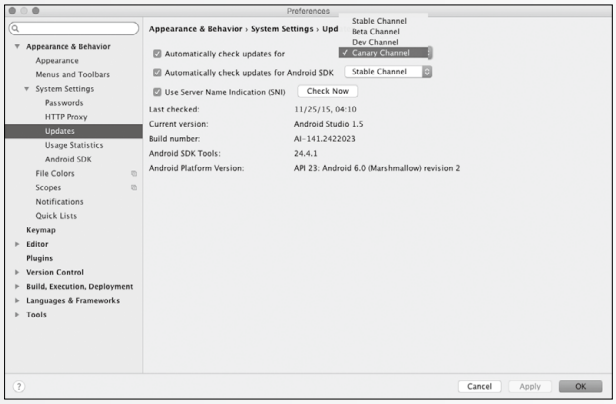
Canaryチャンネル

開発中のAndroid Studioが配布されるチャンネルです。だいたい週に一度のペースで更新されます。Canaryという名前は、昔の炭鉱では敏感なカナリア(Canary)をかごに入れてもぐることで、いち早く有毒ガスの発生を検知したことに由来します。

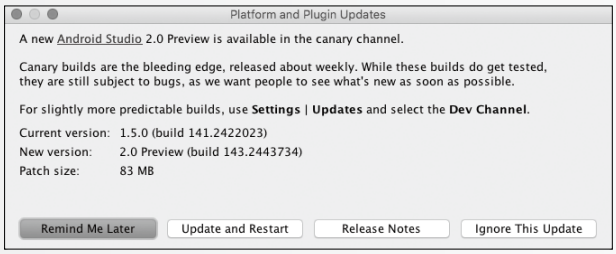
CanaryチャンネルからAndroid Studioを手に入れると、新しい機能を追加したAndroid Studioを試すことができる一方、Canaryはあくまで開発中という位置づけなので不具合がある可能性が高く、実務で使うには注意が必要です。しかし、最新のAndroid Studioに追加される便利な機能を使えるという魅力は非常に強

注3）https://developer.android.com/intl/ja/sdk/

▼図10　アップデートチャンネルの切り替え



▼図11　アップデートがあった場合のダイアログ



く、安定性と引き替えに常にCanaryを使い続けているアプリ開発者もいます。

Canaryチャンネルを使う

CanaryチャンネルのAndroid Studioは、Android Tools Projectのサイト[注4]からダウンロードできます。

また、Stableチャンネルの Android Studioを、Canaryチャンネルに切り替えることもできます。Android Studioの[Preference]を開いて、[Updates]を選択して、[Automatically check updates for]の項目でチャンネルを[Canary Channel]に切り替えます(**図10**)。

下にある[Check Now]のボタンをクリックしてから、[OK]ボタンをクリックするなどして設定画面を閉じると、アップデートがあった場合はダイアログが表示されます(**図11**。設定画面を閉じないと、アップデートのダイアログは表示されないので注意してください)。[Update and Restart]をクリックすると、Canary版のAndroid Studioへのアップデートが始まります。

Android Studio 2.0 Preview

本稿執筆時点で、Canaryチャンネルで配信されている最新バージョンは2.0 Preview 3bです。この目玉機能「Instant Run」を使ってみましょう。

Instant Run

「Instant Run」を使えば、Android端末やエミュレータ上で実行中のコードを動的に置き換えることで、ビルド時間やインストール時間を大幅に短縮できます。ADTからAndroid Studioに移行した人の感想で多いのは「ビルドの時間が長い」というものです。Android Studioがビルドシステムに採用している「Gradle」には便利な機能がたくさんありますが、反面、ADTと比べて1回のビルドにかかる時間は長くなります。

Googleはこの課題を、現在開発中の新しいビルドシステム「Bazel」で解決しようとしていますが、まだ実用には至っていません[注5]。

Instant Runを試すために、Android Studio 2.0 Previewでプロジェクトを作成します。以前のバージョンのAndroid Studioで作成したプロジェクトでInstant Runを試すには、設定の書き換えが必要になります。ここはあくまで試したいだけなので、新しいプロジェクトを作成して実験します。

リスト4は、ボタンを押すたびに画面にメッセージを表示するアプリです(**図12**)。

注4) http://tools.android.com/download/studio

注5) Googleが発表したオープンソースの機械学習フレームワーク「Tensor Flow」のビルドシステムにはBazelが採用されています。

▼リスト4　Instant Runを試すサンプルコード

```
public class MainActivity extends AppCompatActivity {

    private TextView mStatus;
    private Button mButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mStatus = (TextView) findViewById(R.id.status);
        mButton = (Button) findViewById(R.id.button);
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mStatus.append("さようならADT\n");
            }
        });
    }
}
```

mStatus.append("さようならADT\n");に設定してある文字列を"こんにちはAndroid Studio\n"に変更して再度実行します。

これまでは、ビルド後にアプリをインストールし直すため、Activityはそのたびに再起動していました。しかし、Instant Runを使うとActivityの再起動は発生せず、そのまま変更が反映されます。**図13**のように、ボタンを押して表示されるメッセージが途中から変化していることから、再起動せずに変更が反映されたことがわかります。

Instant Runは、すべての場合に使えるのではありません。たとえば、定数として宣言した文字列を変更した場合はInstant Runの対象とならず、Activityが再起動しました。

現時点ではいくつかの制約はあるものの、Instant Runの登場はAndroid Studioの抱える速度の問題の解決に向けた大きな前進と言えるでしょう。

まとめ

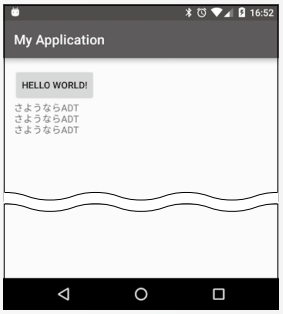
ここまで、Android Studioの活用方法について解説しました。いかがでしたか？

Android StudioはただADTを代替するものではなく、もっと機能が強化された開発環境であることがおわかりいただけたと思います。また、Canaryチャンネルで配布されているAndroid Studioを利用して、不安定ではあるものの、最新の機能を試すこともできました。

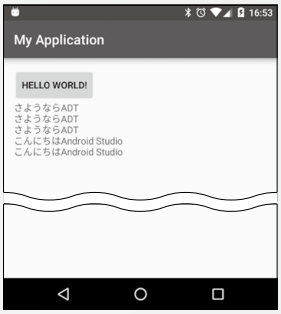
ADTのサポートがなくなった今、公式にはAndroid Studio以外の選択肢はありません。これからAndroidアプリ開発に挑戦したくなったら、拙著の『[改訂版]Android Studioではじめる 簡単Androidアプリ開発』(技術評論社刊)を参考にしていただければ幸いです。

短い期間でバージョンアップを繰り返し、より便利になるAndroid Studioの機能を使いこなすことで、Androidアプリ開発の効率をアップしましょう！SD

▼図12　リスト4のアプリ画面



▼図13　変更後のアプリ画面



短期集中連載

開発に効く数字の測り方

クラウド時代のWebサービス負荷試験再入門

Author 仲川 樽八（なかがわ たるはち） ㈱ゆめみ Twitter@tarupachi

第3回

段取りに従った負荷試験の進め方（前編）

はじめに

第2回の記事では、負荷試験を行うためのツールの紹介および、負荷試験実施にあたっての全体的な考え方を紹介しました。今回からはいよいよ実際の負荷試験の進め方を紹介します。

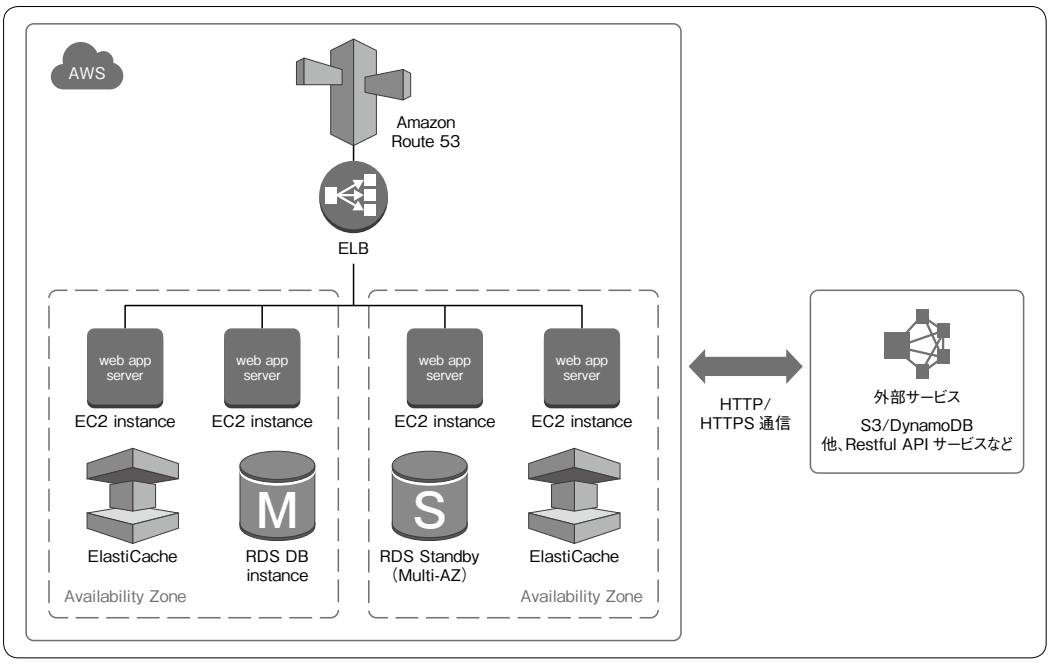
負荷試験対象システム

AWS上に、スケール可能かつ、単一障害点を作らないWebサービスを構成するものとします（**図1**）。システムの概要は次のようになります[注1]。

- MySQLはRDSというAWSの提供するサービスを利用
- 上記MySQLにはMulti-AZオプションを付けて、ホットスタンバイ方式による冗長化を行っておく
- DNSとしてAWSの提供するRoute53サービスを利用
- ロードバランサとしてAWSの提供するElastic Load Balancing（以降ELB）を利用

注1） 今回は説明をシンプルにするために、MySQLのスケール対応はRDSのスケールアップだけで対応することにしていますが基本は同じです。

▼図1 負荷試験の対象となるシステム概要



・Webサーバはデータセンターに相当するAvailability Zone（以降AZ）をまたがる形で複数台利用することで、冗長化とスケールアップ性能を担保
・キャッシュとしてAWSがサービスとして提供するElastiCacheを、Memcacheプロトコルで AZをまたぐ形で利用
・外部のサービスとhttpまたはhttps経由で連携している

負荷試験の段取りについて

前回記事の最後に、より効率的な負荷試験をするためには、負荷試験の過程として次の段階に従って試験を進めていくべきだと説明しました。今回はこのうち、step1.～step5.までを紹介します。

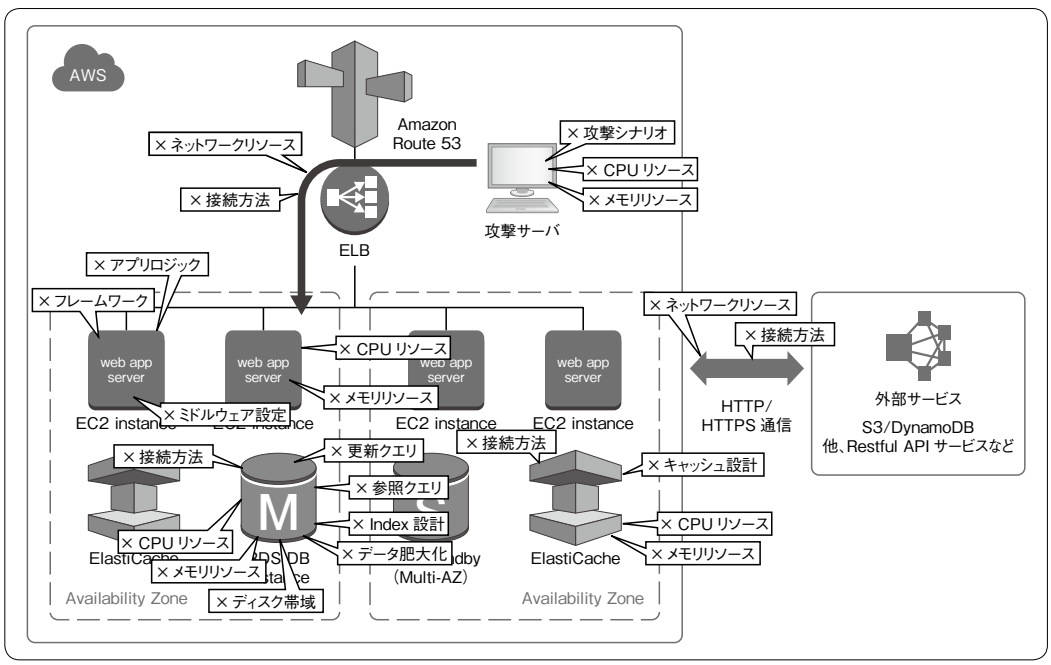
・step1. 静的ファイルを叩くことで利用する負荷試験ツールや設定の試験を行う
・step2. HelloWorldを叩く
・step3. 参照系のページ、APIを叩く
・step4. 更新の発生するページ、APIを叩く
・step5. 外部サービスとの結合を含むページ、APIを叩く
・step6. シナリオを組んで試験を行う
・step7. 離れたネットワークから試験を行う
・step8. 各リソースのスケールアップ・スケールアウトをして試験を行う
・step9. より強い負荷を与えてみる

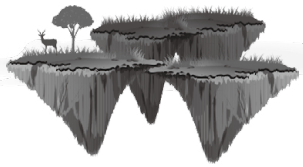
負荷試験に段取りが必要な理由

負荷試験を行った際に、良い結果が出ない原因は、たとえば**図2**に示したものが考えられますが、やみくもに負荷試験を行ってしまうと、問題の特定がたいへんになります。たとえすべてのリソースの負荷をモニタリングしていたとしても、対象のリソースがすでにボトルネック状態になっているのかどうかを判別できないこともあります[注2]。段取りに従った試験を行うことで、これらの問題の切り分けを行いやすくします。

注2）たとえばCPU利用状況でいうと100%付近が上限ということはわかりやすいのですが、ネットワーク帯域やストレージのI/Oがある時点で、上限になっているかどうかはネットワークI/Oなどを監視していても判定が非常に困難です。

▼図2　負荷試験で特定できないさまざまな要因

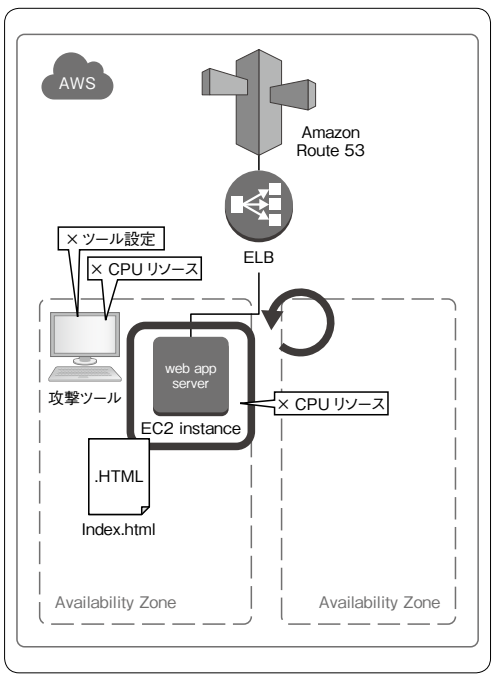




負荷試験の結果が悪くなる例を、次にまとめます。

・負荷試験内容・方法
・インフラの問題
　→ロードバランサの性能不足
　→ネットワーク設定不備／帯域不足
・ミドルウェア設定／カーネルパラメータ設定の不備
・アプリケーションの問題
　→不適切なフレームワーク
　→キャッシュ設計不足
　→不要な問い合わせの発生
　→不適切なロジック
　→CPU／メモリの非効率なコード記載
・WebサーバのCPU／メモリリソース不足
・DBの問題
　→非効率な参照／更新SQL
　→不適切な実行計画
　→不適切なindex
　→CPU／メモリリソース不足
　→ロックの発生

▼図3　試験対象のシステム概要



[Step1]静的ファイルを叩くことで利用する負荷試験ツールや設定の試験を行う

試験対象のシステム概要

最初に肩慣らしです。まずは負荷試験攻撃ツールの扱いに慣れましょう。ここでは、インフラの問題や構築されたアプリケーション、DBの問題などを切り分けるために、Webサーバのうちの1台に対してのみ試験を行います(**図3**)。攻撃ツールを負荷試験対象のWebサーバにインストールし、そこからlocalhost指定で自サーバに対する負荷試験をかけます[注3]。負荷試験をかける対象のURLは静的なファイルとしてください。その理由は、負荷試験対象のシステムにボトルネックを作らないようにし、負荷試験ツールが利用可能なリソースを増やすためです。

実際にユーザがアクセスするときは、システムと同一のデータセンターではなく、ネットワーク的に距離のある場所から発生するということを考えると、負荷試験も実際のユーザのリクエストをシミュレートして遠くからかけたくなるかもしれません。しかしながら、攻撃サーバがネットワーク的に遠いときにはシステムへの負荷が適切にかかりません。、サーバリソースの使用状況は非常に低いにもかかわらずシステムのスループットはまったく出ず、レイテンシが大きくタイムアウトも頻発するといった状況になることがあります。

ネットワーク由来のレイテンシがある状態で、それを適切に評価したうえで負荷量を調節することは非常に困難です。負荷試験に関してはで

注3) 一般的には負荷をかけるツールと、攻撃対象のシステムを同一のサーバ上に構築することは推奨されていません。これは、攻撃ツールがシステムリソースを利用することにより、システムの応答特性が変わってしまうことが大きな原因ではあるのですが、過去の試験においては攻撃サーバが同居することによるシステムスループットの低下は軽微であり、全体の中では大きな問題ではないことが多いです。ここではネットワークの問題を切り分けて考えることができるようになるメリットを優先します。また、ツールの関係で、Webサーバ上にツールをインストールしたくない場合は同一ネットワークセグメントにある攻撃サーバからPrivate IPを指定して負荷をかけることで、ネットワークの影響を最小限に抑えるようにします。

きるだけ近いネットワークからの攻撃をかけることが重要です。前回の負荷試験を卓球のラリーにたとえた例で言えば高速ラリーで相手にプレッシャーをかけるためには、できるだけ相手の近くに陣取る必要があるというイメージになります。

負荷試験の目的

[Step1]では、負荷試験ツールにフォーカスした試験を行います。対象のシステムに対して負荷をかけることができる、その上限を測定します。

次へ進む条件

この試験の結果として、WebサーバのCPUリソースがほぼ100%となり、「数千リクエスト/秒」以上のスループットが出れば、次のStepに進んでください。この段階に限りませんが、できれば複数の攻撃ツールを利用して、スループットの値がほぼ同じ数値となることを確かめたいところです。

負荷がうまくかからない原因の例

試験結果からわかる原因の例を次に挙げます。

・負荷試験ツールの設定不備
・同時アクセス数が過小もしくは過剰

CPUコア数が多い場合、クライアントの同時アクセス数の設定を上げてかまいません。Webサーバの設定として受け付けることができる数を超えるとエラーとなります。詳細過ぎる結果レポートを利用していることも、たとえばJmeterの結果をツリー表示するなども原因のひとつです。そして攻撃ツールが限界に達していることも考えられます。静的なファイルへのリクエストはWebサーバの機能からすると、本来非常に負荷の低いリクエストです。負荷試験ツールによってはツールの限界がWebサーバの限界より先におとずれ、本来のスループットを計測できないこともあります。ただし、計測された結果が最終的なシステムのスループットの目標値を上回っていた場合には、そのまま次の段階に進んでしまってかまいません。システムの目標スループットが今回計測された数値よりも上である場合は、負荷試験ツールの変更や、台数変更などが必要になってくるかもしれません。

[Step2] HelloWorldを叩く

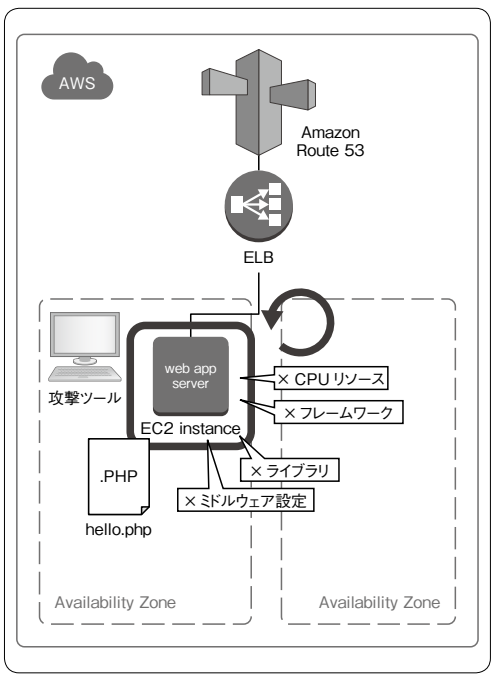
試験対象のシステム概要

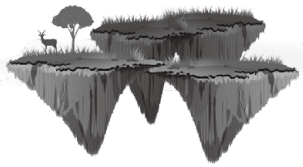
[Step1]の静的ファイルへの攻撃と同様の試験です。今度は静的ファイルではなく、フレームワークの機能を利用して動的に生成されるHellowWorldに対して試験をします(**図4**)。

負荷試験の目的

このあとのロジックを含めた試験結果と比較するため、利用するフレームワークでの最速値を計測します。

▼図4 HelloWorldを叩く、そのための試験構成例





次へ進む条件

この試験の結果として、WebサーバのCPUリソースがほぼ100%となり、期待されるスループットが出れば、次に進んでください。利用するフレームワークにもよりますが、静的なファイルへのアクセスと比較して1/10以下のスループットとなることもあります。

経験上、ここまでの試験においては、Webサーバのメモリリソースやディスク I/Oリソースの枯渇はあまり心配する必要がありません。ほぼCPUがボトルネックとなります[注4]。

負荷がうまくかからない原因の例

この段階で問題に気が付いた場合は、アプリケーション依存のロジックの部分を疑うことなく、フレームワーク利用部分のチューニングを行うことができます。最悪のケースでは、この時点でフレームワークの選定からやり直さないといけない可能性もあります。負荷試験はシステムの構築が終わった最後のリリース直前にスケジューリングされがちですが、それはアンチパターンです。負荷試験の結果、問題が判明したときに対処する時間がなくなりますし、対策のために構成を変えたときには、すでに完了したほかの試験も再実行する必要が生じます。負荷試験に関しては、システムが完全に組み上がる前でかまわないので、かなり早い段階で行う必要があります。次に原因をまとめます。

・フレームワーク選定の間違い
・不適切なフレームワークの利用方法
・apc-cacheなどPHPアクセラレータの未導入
・不適切なライブラリ利用
・不適切ミドルウェアの設定

注4）メモリ使用量と引き換えに、Apacheの同時リクエスト数の受付上限数を上げることはできますが、DBなどへの永続的接続を利用する際にそちらの同時接続数が上限に達してしまうことがあります。そのためApacheの同時接続数の設定は、実際には増やさないことが少なくありません。結果として、メモリリソースが余剰になる傾向があります。

また、負荷試験環境としては商用環境の構成に準ずるものでなければなりません。負荷試験のシナリオの検証をするため、開発用サーバに対して負荷試験をかけて、そのうえでアプリケーションのチューニングを進めようとしたことがあります。その開発サーバは、同一のサーバ上にWebサーバ、DBサーバ、キャッシュサーバなどをすべてインストールしたミニマム構成として構築したものでした。結果としてその環境上でのプロファイリング結果には何の意味もなく、その試験結果を元に無意味なリファクタリングに時間を消費してしまいました。これは同一のサーバ上で動作中に必要リソースが競合してしまって、実際の商用環境での挙動とはまったく異なるものとなったためです。物理構成の異なるサーバ上では、負荷試験シナリオの確認まではできますが、試験結果は利用できません。

[Step3]参照系のページ、APIを叩く

試験対象のシステム概要

ここで初めて、Webサーバの外部リソースを利用する試験を行います(**図5**)。ただし、ここではDBに対する更新を行いません。対象のページの選定は最初は試験を行いやすいページで問題ありませんが、個別でチューニングしたいページがあれば追加でしてください。参照先のテーブルには、あらかじめ運用時のボリュームを想定したダミーデータの登録が必要です。この試験では、参照されるデータの範囲が同一のキーに集中しないようにコントロールする必要がありますので、動的にパラメータを変更して攻撃することが難しいツールの場合には、あらかじめアプリケーション側でパラメータの変更をさせる機構を埋め込んでおいてください。

負荷試験の目的

キャッシュ利用やDBの参照部分にフォーカスした試験を行います。

次へ進む条件

この試験の結果として、Webサーバもしくは DBサーバなどの利用中のリソースのいずれかが逼迫することが確認できれば、次に進んでください。この試験におけるスループットは先のHelloWorld時の試験結果より速くなることはありません。この時点で明らかなレイテンシの悪化およびスループットの低下が認められる場合は先に進む前に個別のチューニングが必要です。

負荷がうまくかからない原因の例

DB接続、Memcache接続ともに永続的接続の利用の有無でスループットは大きく変動しますので、高スループットが要求されるサービスでは永続的接続はほぼ必須になると考えて良いでしょう。こちらはMaxConnectionsの設定を確認しながら調整することが重要です。また、永続的接続を利用することで軽減はされるのですが、AWS利用時には高負荷になるとDB接続の失敗は頻発しますので、DB接続のリトライ機構は組み込んでおく必要があります。次に原因をまとめます。

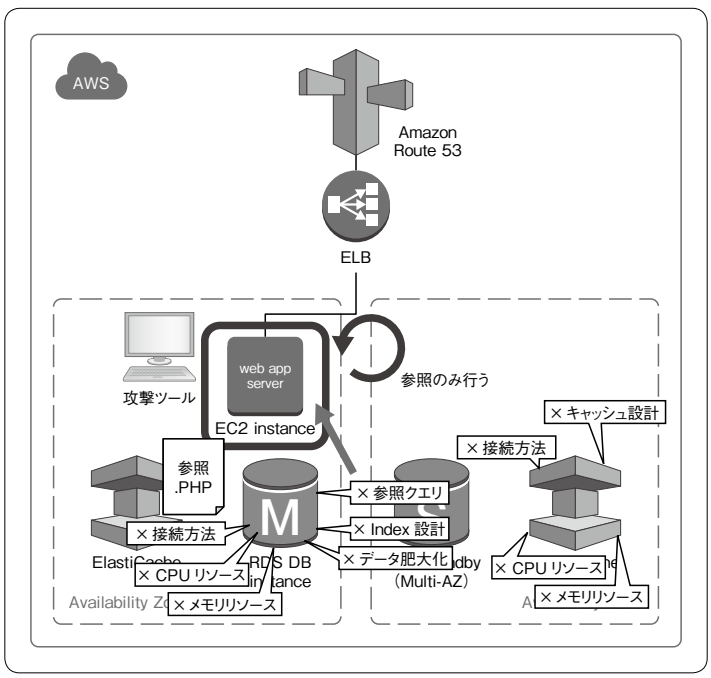
・不適切なMemcache接続方法
・不適切なDB接続方法
・DBの欠陥
　→DBの選定ミス
　→参照SQLの不備(不必要な問い合わせ、不適切なQueryなど)
　→不適切な実行計画
　→不適切なIndex
　→CPU／メモリリソース不足

[Step4]更新の発生するページ、APIを叩く

試験対象のシステム概要

[Step3]と同じですが、対象が更新の発生する部分となります。こちらも、対象のページは試験をしやすい1つのページで問題ありませんが、個別でチューニングしたいページがある場合は、追加試験してください(図6)。

▼図5 参照系のページ、APIを叩く試験構成例

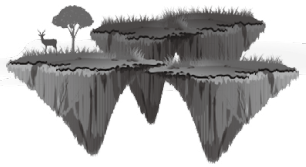


負荷試験の目的

DBの更新処理にフォーカスした試験を行います。DBに限らず、共有リソースに対する更新処理はボトルネックの発生原因となりやすい部分ですので、参照系とは別に試験します。

次へ進む条件

この試験の結果として、Webサーバもしくは DBサーバのリソースのいずれかが逼迫することが確認できれ



ば、次に進んでください。参照系ページへの試験と同様に、この時点で明らかなレイテンシの悪化およびスループットの低下が認められる場合は先に進む前に個別のチューニングが必要です。

負荷がうまくかからない原因の例

負荷試験のやり方が不適切な場合に発生する例としては、実際には別ユーザとしてアクセスする複数の更新リクエストがすべて同じユーザとして発生した場合にはロックが発生し更新待ちとなることがあります。

これまで同様、他の原因を次にまとめます。

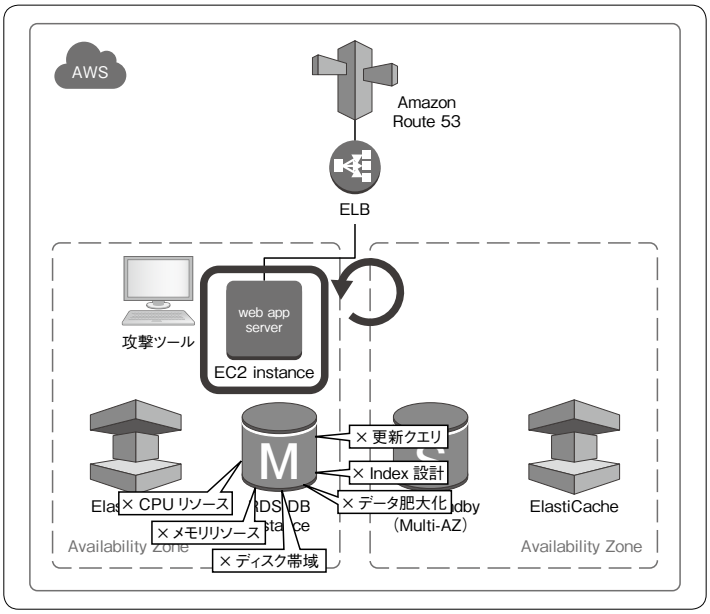
・DBの問題
→更新SQLの不備(不必要な問い合わせ、不適切なQueryなど)
→更新クエリの同一のレコードへの集中[注5]
→不適切な実行計画
→更新ロック範囲不備
→CPU／メモリリソース不足
→ディスクI/Oボトルネック

[step5]外部サービスとの結合を含むページ、APIを叩く

試験対象のシステム概要

外部システムとの結合をAPIなどでサーバ間通信として行う場合、そちらを含めた試験を行います。**図7**に示す対象のシステムとしては、アプリケーション依存の独自のRestful APIサービスへの連携だけにとどまらず、ストレージサービスとしてのS3や、KVSサービスのDynamo DBなどの利用も対象となります。前提条件として、対象の外部システム単体では目的のスループットを上回る応答性能を持っている必要があります。ここでは、ネットワークを中心に試験ができれば良いですので、対象の外部のシステムに負荷をかけて良い試験環境が存在しない場合には、スタブサーバを構築してそちらに静的なファイルを置くなどして対応してください。

▼図6　更新の発生するページ、APIを叩く試験構成例



負荷試験の目的

外部システムとの結合にフォーカスした試験を行います。外部システム利用時のオーバーヘッドを見積もり、より効率的な呼び出しがされていることを確認します。

次へ進む条件

外部システムとの結合をした結果として、明確なボトルネックを観測できなくなることがよくあります。これは次に挙げる負荷がうまくかけられない例に当てはまっても、なおかつそれを解消するための手段が採れないことがあるた

注5) 複数のユーザのアクションを同一のレコード上のカラムをインクリメントしてカウントするテーブルなどがあると行ロックが発生します。そのため全体のスループットを大きく落とす原因となります。この部分が問題になった場合などは、DB設計レベルから見直す必要があります。

めです。ですから外部結合部分のレイテンシを観測して、十分なチューニングの達成を判断できたときに先に進んでください。

負荷がうまくかからない原因の例

これまで同様、原因を次にまとめます。

・結合先外部システムがネットワーク的に離れているためにレイテンシが非常に大きい
・SSL接続（固定のサーバからのサーバ間通信ゆえ、負荷試験攻撃サーバからの攻撃と同じくSSLの影響が非常に高い）
・Keep-Aliveの有無
・NATサーバのボトルネック（VPC外部への接続目的）

対策例

前述の原因に対して解決方法を次に挙げます。

・対象のシステムを同一のVPC内に構築することで、対象のシステムのエンドポイントとして内部ELBを別途構築し、ネットワーク的に近いシステムにする
・SSL接続が必須でない場合には利用しないことを検討する
・Keep-Alive可能な手法を検討する（コラム参照）
・NATの増強

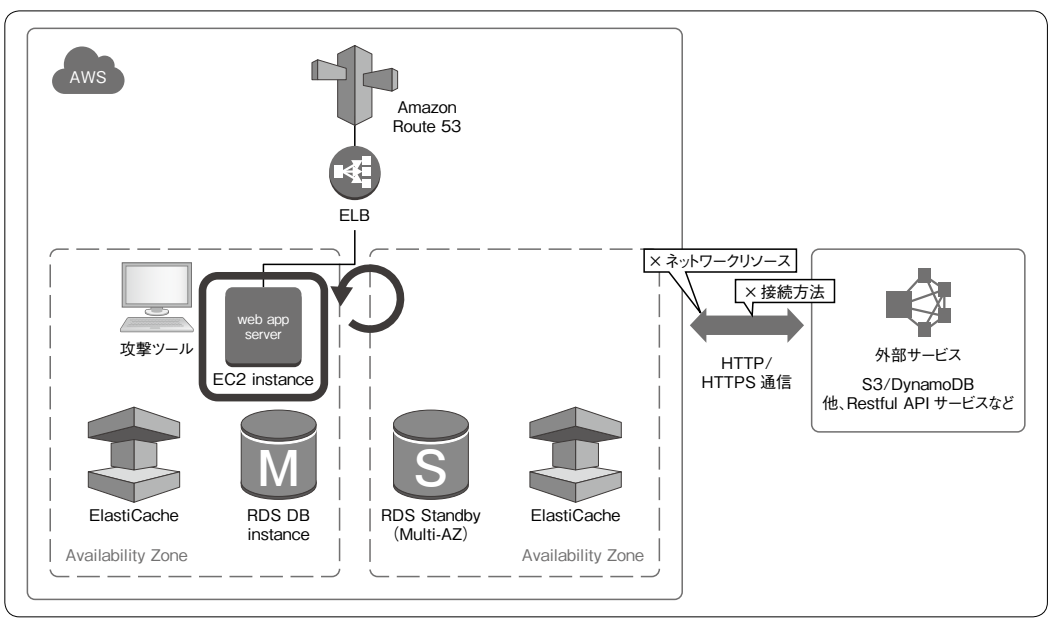
次回予告

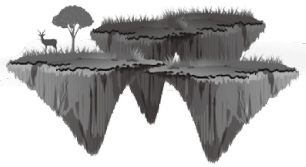
ここまでで紹介したように、WebサーバにCPU負荷がかかっていないのに、レイテンシが遅い、スループットが低い、といった状況が発生したとき、それらの原因の多くはWebサーバの内部にはなく、次のようなWebサーバ外部との結合部分にあります。

・cache接続部分
・DB接続部分
・外部サービス結合部分

このような状態になったときには、Webサーバをスケールアップ、スケールアウトしてもシステム全体のスループットはほとんど改善しないという状況となりますので、次へのStepへ進むことは無意味です。まずは今回紹介したStepまでを1つずつ見なおしてください。次

▼図7　外部サービスとの結合を含むページ、APIを叩く試験構成例





回は最終回ですが、後編として、スケール可能なシステムに対する、より実践的な負荷試験をかける方法を紹介します。SD

COLUMN

PHPのサーバ間通信でKeep-Aliveを行う方法について

Apache＋mod-phpという構成だけでサーバ間通信のKeep-Aliveを行うことは厳密にはできません。しかしながら、同一リクエスト内で複数回同一の外部APIに対して通信を行う場合に限り、CURLを利用した通信で実現できます。

■同一リクエスト内でサーバ間のKeep-Aliveをする方法

CURLOPT_FORBID_REUSEの値はデフォルトではfalseであるため、**リスト1**のように、curl_init()の結果をsingletonにすることで同一リクエスト内に限りKeep-Aliveを行えるようになります。ですが、curlオブジェクトをそのつど再生成しないため、以前設定したcurl_setoptの値を後続の処理が引き継いでしまうことに注意してください（これをcurl::init()の中でリセットしています）。

ただし、この手法では複数のリクエストをまたいだKeep-Aliveを行うことはできませんので、Webサービスとして利用する場合の効果はかなり限定的です（バッチ内で大量にリクエストを発行する場合には、そのつどcurl_init()を呼び出す場合と比較して数倍の速度となります）。

▼リスト1　mod-phpのサンプル

```
<?php
class curl {
    protected static $curl = null;

    public static function init($path){
        if(!self::$curl){
            self::$curl = curl_init();
        }
        curl_setopt(self::$curl, CURLOPT_URL, $path);
        curl_setopt(self::$curl, CURLOPT_POSTFIELDS, '');
        curl_setopt(self::$curl, CURLOPT_TIMEOUT , CURLOPT_RESTAPI_TIMEOUT);
        curl_setopt(self::$curl, CURLOPT_RETURNTRANSFER  , true);
        curl_setopt(self::$curl, CURLOPT_HEADER, true);
        curl_setopt(self::$curl, CURLOPT_SSL_VERIFYPEER  , false);
        return self::$curl;
    }
}

$curl = curl::init(API_PATH);
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'GET');
$get_response = curl_exec($curl);

$curl2 = curl::init(API_PATH);
curl_setopt($curl2, CURLOPT_CUSTOMREQUEST, 'POST');
curl_setopt($curl2, CURLOPT_POSTFIELDS, $post_fields);
$post_response = curl_exec($curl);
```

■複数のリクエストをまたいでサーバ間のKeep-Aliveをする方法

Nginxなどのツールを用いてローカルにフォワードプロキシサーバを立てて、そちらと外部サーバ間でKeep-Aliveをさせるという方法もあります（**図A**）。検証環境および検証コードではこの手法を取ることで、PHPから単独での外部APIとのサーバ間通信を行っていた場合の10倍近いスループットを出すことができました（**リスト2**）。古い情報ですと、NginxでのSSLフォワードは対応していないと記載してあるのですが、現在はSSLも対応しているようです。**リスト2**の設定の場合、

```
http://localhost/elb/～
http://localhost/sslelb/～
```

へのリクエストがそれぞれ、

```
http://test-elb-endpoint.elb.amazonaws.com/～
https://test-elb-endpoint.elb.amazonaws.com/～
```

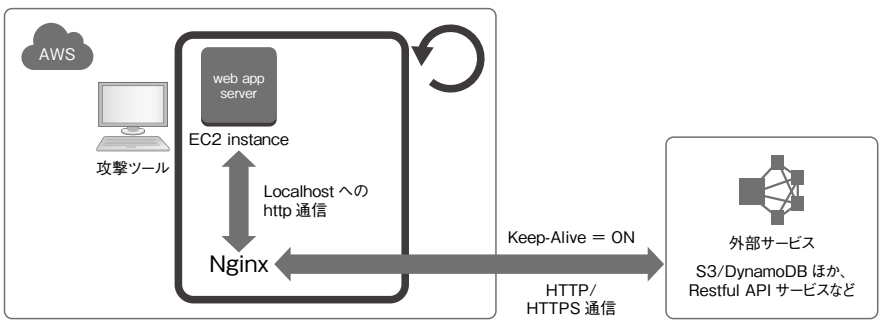
へKeep-Aliveされた状態でフォワードされたサーバ間通信となります。この手法をとった場合には利

用するシステムやミドルウェアに依存せずにKeep-Aliveを利用できるメリットがあります。また、ローカルのWebサーバ上にNginxインストールが難しい場合であっても、Nginxをローカルネットワーク上のサービスとして立ちあげて、そちらとの通信を行うという構成を取ることも考えられます。この場合はローカルネットワーク上の通信になるので、外部のサービスとの通信と比較すると非常に高速に応答が可能になります。

▼リスト2　nginx.conf設定サンプルから一部抜粋

```
……省略……
upstream elb {
    server test-elb-endpoint.elb.amazonaws.com:80;
    keepalive 32;
}

upstream sslelb {
    server test-elb-endpoint.elb.amazonaws.com:443;
    keepalive 32;
}

server {
    listen       8000;
    server_name  localhost;
    root         /usr/share/nginx/html;

    #charset koi8-r;

    #access_log  /var/log/nginx/host.access.log  main;

    location /elb {
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        rewrite /elb/(.*) $1 break;
        proxy_pass http://elb/$1;
    }

    location /sslelb {
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host        "test-elb-endpoint.elb.amazonaws.com";
        proxy_ssl_session_reuse off;
        rewrite /sslelb/(.*) $1 break;
        proxy_pass https://sslelb/$1;
    }
……省略……
```

▼図A　Nginxを使用したKeep-Alive生成例



手がかりを探せ！

SMB実装をめぐる冒険

第4回 最終回

File System for Windowsの作り方

探す、調べる、ソフトを作る喜び

こんにちは。よういちろうです。「Windows共有フォルダをChromeOSのファイルアプリにマウントする」ことができるChromeアプリを開発してリリースしました。これを開発するためには、SMB（Server Message Block）と呼ばれるプロトコルを理解し、SMBプロトコルを話すクライアントコードをJavaScriptで書くことが必要でした。これは「File System for Windows」という名前でChromeウェブストアにて無料で公開していますので、Chromebookを持っている方はぜひ使ってみてください。今回は大団円の最終回です。

Author 田中 洋一郎（たなか よういちろう） Blog https://www.eisbahn.jp/yoichiro Twitter @yoichiro

第6部 資産を我が手にしろ

ネゴシエーション、ユーザ認証、そして共有リソース一覧取得と、非常に複雑な手順を一つずつ紐解いてきました。これら必要な前処理はすべて終わり、敵のアジトに潜入して資産をいじり放題できるところまで来ました。つまり、やっとディレクトリやファイルという言葉を使い始めることができる段階までたどり着いたわけです。

SMBプロトコルにおける「複雑」と言われている個所は、もうすべてクリアしました。苦行は終わりです。あとは自分がやりたいと思うファイル操作を実現していくだけです。

共有リソースへの接続とtree_idの入手

SMBプロトコルでのファイル操作は、共有リソース単位で行われます。共有リソース一覧の取得処理の中で、SMB_COM_TREE_CONNECT_ANDXメッセージを使ってIPC$共有リソースに接続を行いました。これと同じ手順で、扱いたい共有リソースに接続を行い、その結果のtree_idを入手します。その際に使う共有リソース名は、共有リソース一覧で取得した結果に含まれるnameを使えば良いです。

基本的に本連載第3回の**リスト1**で説明した構造を使うのですが、SMB_DATAの指定内容が少しだけ異なっています。具体的には、**リスト1**のようになります。

仮に接続したい共有リソース名が“share”だった場合は、pathの指定は“¥¥[server_name]¥share”となります。serviceに関しては、前回と同じように“?????”というワイルドカード指定で大丈夫です。

このレスポンスとして新しいtree_id値がサーバから返ってきます。次のメッセージ送信からは、ヘッダのtree_idに入手した値をセットします。もちろん、user_idも忘れずに指定しましょう。

読み解ける仕様書

tcpdumpやWiresharkを使ってパケットキャプチャすることで実際の通信内容を解析しながら今まで前に進んできました。SMB1/CIFS、

▼リスト1 ファイルアクセスのためのSMB_COM_TREE_CONNECT_ANDXメッセージのSMB_DATA構造

```
SMB_DATA {
  USHORT byte_length;
  bytes {
    ANY password;           // 0x00
    UCHAR[0 or 1] padding; // この場合はなしでOK
    ANY path;               // "¥¥[server_name]¥[共有リソース名]"
    ANY service;            // "?????"
  }
}
```

NTLMSSP、DCE/RPCという各種プロトコルの内容を解析してきた今であれば、「実はCIFSの仕様書を読んで理解できるのではないか？」とふと思いました。パケットやヘッダ、パラメータ、データ、コマンド、user_id、そしてtree_idといった基本構造や作法は十分に身についていて、今やパケットキャプチャされた16進数のダンプリストを見て直接理解できるレベルまで来ています。ここまで来れば、仕様書を読むための前提知識は自然と身についた気がしました。

Microsoft社は、CIFSの仕様を文書化して公開していますが、2種類あります。

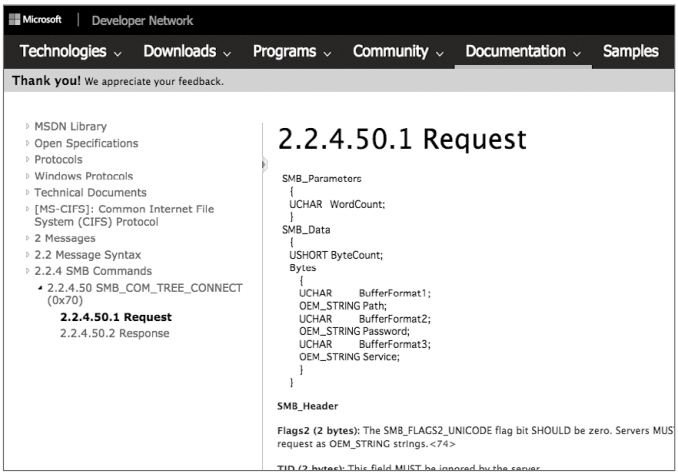
- IETFに提出された仕様書[注1]
- Microsoft Developer Networkにて公開された仕様書[注2]

この2つめのMS-CIFS文書を読んでみると、やはり「読める！読めるぞ！」となりました。たとえば「2.2.3 SMB Message Structure」で説明されているヘッダ、パラメータ、そしてデータの構造は、今まで解析してきた内容がほぼそのまま掲載されています。「なんだ、最初から読めばよかった」と思うのは、いまだから言えることです。最初にこれを読んだとしても、きっと理解できなかったことでしょう。試行錯誤してきた結果として理解できるようになった、と思うことにします。

この気づきによって、パケットキャプチャの目的が「プロトコルの内容をキャプチャ結果から知ること」ではなく、「仕様書に書かれた内容どおりの通信内容かどうか」の確認やデバッグするための情報収集に変わりました。ここからは、行いたい操作を実現するためのコマンドを探すために、まずはMS-CIFS仕様書の「2.2.4 SMB Commands」に掲載されたコマンド一覧に頻繁に訪れることになりました。各コマンドのページには、**図1**のように、リクエストとレスポンスそれぞれの構造が説明されています。

ここから先の開発がとても楽しくなったことを、今でも覚えています。

▼図1　MS-CIFS文書のコマンド説明ページ



あるディレクトリ内のファイル一覧取得

すべてのファイル操作を本誌面上で紹介することはできませんので、いくつか代表的な操作について取り上げてみたいと思います。最初は、あるディレクトリ内のファイル一覧を取得するための方法です。

SMB1/CIFSプロトコルにおけるファイル一覧の取得処理は、SMB_COM_TRANSACTION2(0x32)メッセージと、そのサブメッセージであるTRANS2_FIND_FIRST2(0x0001)とTRANS2_FIND_NEXT2(0x0002)を使います。「またサブプロトコルかよ！」と思ったかもしれませんが、それに近いです。厳密に言うと、これらのサブコマンドはSMB1/CIFSプロトコル内で規定さ

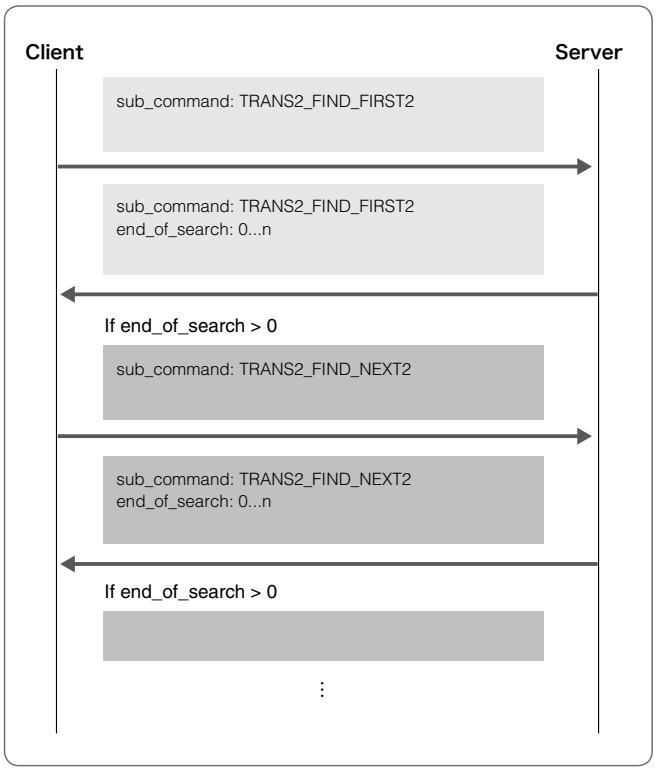
注1）https://tools.ietf.org/html/draft-heizer-cifs-v1-spec-00
注2）https://msdn.microsoft.com/en-us/library/ee442092.aspx

れていますので、別のプロトコルというわけではありません。

実はSMB_COM_TRANSACTION2メッセージの構成は、コマンドが0x32になるだけで、SMB_COM_TRANSACTIONメッセージとまったく同じです。つまり、trans_setup、trans_parameter、そしてtrans_dataの3つのペイロードを持っています。TRANS2_FIND_FIRST2およびTRANS2_FIND_NEXT2では、trans_setupおよびtrans_parameterの2つのペイロードを使います。

TRANS2_FIND_FIRST2とTRANS2_FIND_NEXT2の使い方は、**図2**のようになります。最初にTRANS2_FIND_FIRST2リクエストをサーバに送信し、その結果のファイル一覧を取得します。そのレスポンスの中に、end_of_searchという値が含まれています。この値が0ではなかった場合、まだ取得すべきファイル一覧がサーバに残っているという状況を示しています。残りのファイル一覧を取得するために、今後はTRANS2_FIND_NEXT2リクエストをサーバに送信します。そしてサーバからファイル一覧を得た後、再度end_of_search値が0かどうかを見て、0になるまでTRANS_FIND_NEXT2を送り続けます。

▼図2　TRANS2_FIND_FIRST2とTRANS2_FIND_NEXT2の使い分け



▼リスト2　TRANS2_FIND_FIRST2リクエストの構造

```
trans_setup {
  USHORT function;           // TRANS2_FIND_FIRST2 (0x0001)
}
trans_parameter {
  USHORT search_attributes; // HIDDEN(0x0002) | SYSTEM(0x0004) | ↵
DIRECTORY(0x0010)
  USHORT search_count;       // 1リクエストあたりの取得件数
  USHORT flags; // 0
  USHORT information_level; // BOTH_DIRECTORY_INFO (0x0104)
  UINT search_storage_type; // 0
  ANY file_name;             // Null-terminated UNICODE文字列
}
```

リスト2は、TRANS2_FIND_FIRST2リクエストの構造を説明したものです。SMB_COM_TRANSACTION2リクエストのtrans_setupおよびtrans_parameterに対してバイト列がセットされます。

あるディレクトリ内のファイル一覧を取得したい場合は、file_nameの末尾に"¥*"というようにワイルドカードを指定します。たとえば、UNIX形式で"/foo/bar"というパスで示されるディレクトリを対象にしたければ、file_nameは"¥foo¥bar¥*"と指定します。このfile_nameに指定するパスは、接続した共有リソースをルートとしたパスになります。

サーバは、ファイル一覧をTRANS2_FIND_FIRST2レスポンスにてクライアントに返します。TRANS2_FIND_FIRST2レスポンスでは、SMB_COM_TRANSACTION2レスポンスのtrans_parameterおよ

びtrans_dataペイロードが使われます。**リスト3**は、TRANS2_FIND_FIRST2レスポンスの構造です。

各ファイルに関する情報が、TRANS2_FIND_FIRST2レスポンス内で列挙されます。もしfile_attributesにFILE_ATTRIBUTE_DIRECTORY(0x0010)が含まれていたら、それはディレクトリです。

trans_parameterに含まれるend_of_search値が0ではなかった場合は、TRANS2_FIND_NEXT2リクエストを送信します。TRANS2_FIND_NEXT2の構造は、TRANS2_FIND_FIRST2とほとんど変わりません。TRANS2_FIND_FIRST2レスポンスの内容と、end_of_search値が0になるまで繰り返し取得したTRANS_FIND_NEXT2レスポンスの内容を合わせて、ファイル一覧取得結果とします。

▼リスト3　TRANS2_FIND_FIRST2レスポンスの構造

```
trans_parameter {
  USHORT search_id;
  USHORT search_count;          // レスポンスに含まれるファイル数
  USHORT end_of_search;
  USHORT ea_error_offset;
  USHORT last_name_offset;
}
trans_data {
  FILE_INFO[search_count] files;
}
---
struct {
  UINT next_entry_offset;
  UINT file_index;
  ULONG creation_time;          // SMBタイムスタンプ値
  ULONG last_access_time;       // SMBタイムスタンプ値
  ULONG last_write_time;        // SMBタイムスタンプ値
  ULONG last_change_time;       // SMBタイムスタンプ値
  ULONG end_of_file;            // 実際のファイルサイズ
  ULONG allocation_size;        // ディスク上のサイズ
  UINT file_attributes;         // ファイル属性のフラグ値
  UINT file_name_length;
  UINT ea_size;
  UCHAR short_name_length;
  UCHAR reserved;
  UCHAR[24] short_name;
  ANY file_name;                // UNICODE文字列
} FILE_INFO;
```

ファイルの読み込み

ディレクトリをトラバースしながらファイル一覧を自由自在に取得できたあとは、各ファイルの内容を読み込みたくなってきます。あるファイルの内容を読み込むためには、次の手順を踏みます。

① ファイル一覧取得結果から、対象のファイルのファイルサイズ(end_of_file値)を得ておく
② SMB_COM_NT_CREATE_ANDXメッセージを使って、対象のファイルをオープンし、その結果のFID値を入手する
③ SMB_COM_READ_ANDXメッセージを使って、ファイル内のある位置からある長さ分のバイト列を読み込む
④ ③で読み込んだバイト数の合計が①のファイルサイズに達するまで、読み込み開始位置を変えていきながら③を繰り返す
⑤ SMB_COM_CLOSE(0x04)メッセージを使って、対象のファイルをクローズする

最初にネゴシエートした際に、SMBメッセージの1回あたりの最大バイト数が決定していますので、その最大バイト数を超える大きさのファイルは、1回のレスポンスだけでは読み込むことができません。そのため、ファイル内の読み込み開始位置をずらしていきながら、最大バイト数の範囲内でファイルの内容を繰り返し読み込んでいきます。

FID値を入手するための方法は、共有リソース一覧を取得したときに行った"¥¥[server_name]¥IPC$¥srvsvc"のFID値の入手方法とまったく同じです。つまり、SMB_COM_NT_CREATE_ANDXメッセージを使います。その際、create Disposition値にFILE_OPEN(0x01)を、file_nameとして読み込みたいファイルのパスを指定します。たとえば、UNIX形式で"/foo/bar/hello.txt"というパスのファイルを読み込みたい場合は、file_nameには"¥foo¥bar¥hello.

txt"と指定します。そして、SMB_COM_NT_CREATE_ANDXレスポンスに含まれるFID値を入手します。

その際に書き込みを行うことができないようにしたい場合は、desired_accessフラグ値から書き込み処理関連のビットを0にしておくと良いでしょう。

ファイルの読み込みは、SMB_COM_READ_ANDX(0x2e)メッセージを使います。このSMB_COM_READ_ANDXは、指定されたFID値で示されるファイルの内容について、指定位置から指定バイト数を読み込む能力があります。**リスト4**は、SMB_COM_READ_ANDXリクエストの構造です。パラメータだけ使用します。

このメッセージの仕様があとから拡張されたためなのか、読み込み開始位置の指定が4バイトずつ離れた個所で指定するようになっています。max_countおよびmin_countについては、基本的には特別な理由がない限り、1回のレスポンスで受信可能なバイト数をどちらにも指定しておけば良いでしょう。

SMB_COM_READ_ANDXレスポンスの構造は、**リスト5**のようになります。先ほどと違い、パラメータとデータの両方が使われます。

筆者の書いたコードでは、必ずリクエスト時に要求したバイト数がレスポンスで返ってくる保証はない、という前提で開発をしました。具体的には、期待する総読み込みバイト数から、data_length値を引いていって、その数が0になるまでSMB_COM_READ_ANDXリクエストを繰り返し送信する、ということをしています。

あとで仕様書を読んだところ、max_count_of_bytes_to_return値で指定したバイト数よりもdata_length値が小さかった場合は、「ファイルの終端(End Of the File)に達した」とみなして良いと書かれていました。EOFを検出するためには、自分で計算せずに、max_count_of_bytes_to_return値とdata_length値の比較で良いかもしれません。

後始末として、SMB_COM_CLOSE(0x04)メッセージを使って、対象のファイルをクローズします。SMB_COM_CLOSEリクエストの構造は、**リスト6**のよ

▼リスト4　SMB_COM_READ_ANDXリクエストの構造

```
SMB_PARAMETER {
  UCHAR word_count;
  words {
    struct {
      UCHAR command;                        // 0xff
      UCHAR reserved;                       // 0
      USHORT offset;                        // 0
    } ANDX;
    USHORT fid;                             // FID
    UINT offset_low;                        // 読み込み開始位置の下位4バイト
    USHORT max_count_of_bytes_to_return;    // 読み込む最大バイト数
    USHORT min_count_of_bytes_to_return;    // 読み込む最小バイト数
    UINT timeout;                           // INFINITE (0)
    USHORT remaining;                       // 0
    UINT offset_high;                       // 読み込み開始位置の上位4バイト
  }
}
```

▼リスト5　SMB_COM_READ_ANDXレスポンスの構造

```
SMB_PARAMETER {
  UCHAR word_count;
  words {
    USHORT available;             // 0
    USHORT data_compaction_mode; // 0
    UCHAR[2] reserved;            // 0
    USHORT data_length;           // このレスポンスが持つファイル内容のバイト数
    USHORT data_offset;           // ファイル内容があるSMBメッセージ先頭から
の位置
    UCHAR[10] reserved;           // 0
  }
}
SMB_DATA {
  USHORT byte_count;
  bytes {
    UCHAR[0 or 1] padding;        // 2バイト境界に合わせるため
    ANY data;                     // ファイル内容のバイト列
  }
}
```

▼リスト6　SMB_COM_CLOSEリクエストの構造

```
SMB_PARAMETER {
  UCHAR word_count;
  words {
    USHORT fid; // FID
    UINT last_modified_time; // UNIXタイムスタンプ(s)
  }
}
```

うに単純です。

サーバからのレスポンスのヘッダに含まれるnt_status値が0であれば、対象のファイルをクローズできたことになります。おもしろい機能としては、last_modified_timeを使うことで、ファイルをクローズする際に最終変更日時を更新できます。変更したくない場合は、0を指定しておけば良いです。

ファイルの書き込み

ファイル操作の最後に、ファイルの書き込み方法について紹介したいと思います。ファイル一覧が取得できて、ファイルが読み込めて、そしてファイルを作成し内容を書き込むことができれば、共有ファイルに対する基本的かつ最も重要な操作ができるようになったと言うことができるでしょう。つまり、敵を完全に攻略した、と言って良さそうです。

ファイルの読み込み方法がわかってしまえば、実はファイルの書き込み方法はそう難しくありません。なぜなら、読み込みと逆のことをしてあげれば良いだけだからです。手順としては次になります。

①SMB_COM_NT_CREATE_ANDXメッセージを使って、書き込みモードで対象のファイルをオープンし、その結果のFID値を入手する
②SMB_COM_WRITE_ANDX(0x2f)メッセージを使って、ファイルの書き込み位置と書き込みたいバイト列をサーバに送信する
③書き込みたいすべてのバイト列をサーバに送信し終わるまで、②を繰り返す
④SMB_COM_CLOSE(0x04)メッセージを使って、対象のファイルをクローズする

1回のリクエストでのSMBパケットの大きさは制限されますので、書き込みたい内容を1回のリクエストですべて送信できない可能性があります。そのため、書き込みたい内容を細切れにして、複数回サーバにリクエストを送信します。

FID値を入手するための方法は、共有リソース一覧を取得したとき、そしてファイルの読み込みを行ったときと同じで良いです。その際に、もし指定したfile_nameで示されるファイルがまだ存在していなかった場合は、create_dispositionとしてFILE_OPEN_IF(0x00000003)を指定することで、自動的にファイルを作成してくれます。その後すぐにクローズしてしまえば、ファイルサイズが0のファイルを作成できます。

書き込みたい内容をサーバに送るためのSMB_COM_WRITE_ANDXリクエストの構造は**リスト7**になります。

SMB_COM_READ_ANDXレスポンスの内容と似ていることがわかると思います。このリクエストを受け取ったサーバは、書き込み処理を行い、その結果をクライアントに返します。**リスト8**は、SMB_COM_WRITE_ANDXレスポンスの構造です。

念のため書き込まれたバイト数をcountから入手して、次に書き込む位置とバイト列を決定して、再度SMB_COM_WRITE_ANDXリクエストを送信します。それを繰り返すことで、書き込みたいすべての内容をサーバに送信します。

最後に、対象のファイルをSMB_COM_CLOSEメッセージを使ってクローズします。これはファイルの読み込みのときに行った内容とまったく同じです。

その他の操作と対応するコマンド

今回は、代表的な3つの操作(ファイル一覧取得、ファイルの読み込み、ファイルの書き込み)の方法について紹介しました。ほかにも

SMB1/CIFSプロトコルで規定された操作はたくさんあります。ChromeOS向けのアプリケーションで実装した経験があるそのほかのコマンドを、**表1**に示します。

ネゴシエートからファイルの書き込みまで、非常に複雑な手順や送受信内容の上にSMB1/CIFSプロトコルが成り立っていることがわかっていただけたと思います。しかし、もう敵は丸裸です。Windowsであれ、OS Xであれ、Sambaであれ、今まで紹介してきた手順を使えば、SMBプロトコルを使ったファイル操作が可能になります。また、サーバの設定内容にも依存しますが、NTLMSSP認証方式とLMv2 Response、NTLMv2 Responseを組み合わせて、ドメイン環境下にあるファイル共有サーバにもアクセスできます。

▼リスト7　SMB_COM_WRITE_ANDXリクエストの構造

```
SMB_PARAMETER {
  UCHAR word_count;
  words {
    struct {
      UCHAR command;      // 0xff
      UCHAR reserved;     // 0
      USHORT offset;      // 0
    } ANDX;
    USHORT fid;           // FID
    UINT offset_low;      // ファイルの書き込み位置の上位4バイト
    UINT timeout;         // INFINITE (0)
    USHORT write_mode;    // 0
    USHORT remaining;     // 0
    USHORT data_length;   // 書き込むバイト列の大きさ
    USHORT data_offset;   // 書き込むバイト列のSMBメッセージ先頭からの位置
    UINT offset_high;     // ファイルの書き込み位置の下位4バイト
  }
}
SMB_DATA {
  USHORT byte_count;
  bytes {
    UCHAR padding;        // 0
    ANY data;             // data_lengthで指定した長さのバイト列
  }
}
```

▼リスト8　SMB_COM_WRITE_ANDXレスポンスの構造

```
SMB_PARAMETER {
  UCHAR word_count;
  words {
    USHORT count;         // 書き込まれたバイト数
    USHORT available;
  }
}
```

謎はすべて解けました。世界中からの期待にも、これで答えることができそうです。実際にここまで調べてきた知識をもとにして、JavaScriptでSMBクライアントを実装し、File System for Windowsという名前でアプリケーションをChromeウェブストアに公開しました。より多くの人々に使ってもらうべく、日々改善に努めています。

▼表1　SMB1/CIFSプロトコルで規定されたコマンド（抜粋）

コマンド	値	意味
SMB_COM_ECHO	0x2b	クライアントからの送信内容をそのまま返す
SMB_COM_SEEK	0x12	指定された位置まで進める
SMB_COM_DELETE	0x06	ファイルを削除する
SMB_COM_DELETE_DIRECTORY	0x01	ディレクトリを削除する
SMB_COM_RENAME	0x07	ファイルやディレクトリの名前を変更する
SMB_COM_FIND_CLOSE2	0x34	検索結果を閉じる
SMB_COM_TREE_DISCONNECT	0x71	共有リソースへの接続を解除する
SMB_COM_LOGOFF_ANDX	0x74	ログオフする

第7部　調査は続く

目的だったSMB1/CIFSプロトコルの理解と実装は、これで達成しました。ほぼ攻略したと言っても良いと思っていますが、残念ながら日々ユーザからの不具合報告があります。たとえば、特定のサーバでは動作しない、共有リソースの個数が多い場合にエラーになる、といった改善すべき点やもっと深く理解する必要がある処理など、毎日SMBプロトコルの奥の深さを痛感しています。

しかし、SMB1/CIFSプロトコルを把握したからといって、SMBプロトコルの世界の理解度はまだまだ低いと言わざるを得ません。

時代はもうSMB2,3である

パケットキャプチャを駆使して理解したSMB1/CIFSプロトコルは、登場から20年以上経過している非常に古いプロトコルです。DOS時代に策定されたあとに、ロングファイルネーム形式やUnicodeサポートが追加されたために、結果として継ぎはぎだらけのプロトコルに仕上がってしまったという印象を強く持ちました。

ではそんな状況が今まで放置されていたのか？というとそんなことはなく、もうSMB1/CIFSプロトコルは捨てよう、というタイミングの手前まで来ています。その代わり、今ではSMBプロトコルはバージョン3まで進化しています。その前のSMBバージョン2(SMB2)の策定時には、Unicodeを前提とし、コマンドの数も10分の1まで減らすなど、抜本的な改善が行われました。つまり、SMB1/CIFSとSMB2以降では、互換性はありません。

そろそろSMB1/CIFSプロトコルでは会話ができないサーバが増えてきそうですので、できるだけ早くSMB2以降のバージョンを理解して実装する必要があります。

時代はもうKerberosである

本記事では、ユーザ認証方式として、LMv2 ResponseおよびNTLMv2 Responseの作り方を解説しました。もちろん現在でもこの方法は多くのサーバで利用可能なのですが、すでにWindowsネットワークの世界では、Kerberos認証方式が主流となっています。

Windows Server 2008や2012では、NTLMv2方式でのユーザ認証を無効にできます。セキュリティの向上を強く意識している管理者であればあるほど、NTLMv2方式を無効にして、Kerberos認証のみを許可している可能性が高まります。そうなってしまうと、NTLMv2による認証方式しかサポートしていないクライアントは役に立たないものとなってしまいます。

より多くの環境で動作するクライアントとするためには、Kerberosによるユーザ認証方式のサポートは必須と言えます。

おわりに

以上でSMB1/CIFSプロトコルの説明は終わりとなります。SMB2、3やKerberos以外にも、暗号化や署名といった機能には今回触れませんでした。近年セキュリティに関する懸念事項は増す一方であり、セキュリティ的にどんなに安全だったとしても足りないと言われてしまうほど、ファイル共有はシビアな分野です。SMBプロトコルもそういったニーズに応えるべく、今も進化を続けています。

今後もそういった進化に追随していきながら、もしまとまった知識を得ることができたなら、再度今回のように文書化して読者の方々にお伝えします！SD

presented by
Japan Android Group
http://www.
android-group.jp/

コミュニティメンバーが伝える Androidで広がるエンジニアの愉しみ

第2回 Android 6.0の新しいセキュリティモデル

谷口 岳(たにぐち がく)
リスクファインダー(株)

Androidは世界で出荷される8割のスマートフォンに搭載※される、事実上スマートフォンの標準OSです。このOS上で動くアプリケーションを開発することは、自分のアイデアを世界に発信し、最も多くのスマートフォン上で動かしてもらえる可能性がある行為です。このAndroidの最新情報と、それを取り巻くコミュニティを知って、魅力的な開発をはじめませんか?

※IDC Worldwide Mobile Phone Tracker, August 7, 2013

はじめに

Google I/O 2015でAndroid 6.0が発表されました。これにあわせて初のAndroid 6.0端末となるNexus 5X、Nexus 6Pが日本でも発売されました。一部の旧Nexusにもアップデートが来ているため、手にされている方も増えていると思います。Android 6.0は外から見ると大きな変更点はないように見えますが、セキュリティ的な観点からはかなり大きな変更がありました。

従来の携帯電話(いわゆるガラケー)では、プライバシーにかかわる情報をキャリアがすべて考えてくれていました。スマートフォンが主流になり、自由にアプリケーション(以降、アプリと省略する場合があります)を入れることができて便利にはなりましたが、「自分のプライバシーは自分で責任を持つ」ルールになってしまいました。

外部アプリの利用には、ユーザのプライバシー情報を盗み取る悪いアプリ、提供したプライバシーデータの漏えい、プライバシーデータの二次利用・販売、などといった負の側面が存在します。一方、正当な理由によりプライバシー情報を要求し、正しい運用をしているアプリもあります。しかしながら、良いアプリか、悪いアプリかの判断は難しく、従来のAndroidのセキュリティモデルだと、ユーザが「自分のプライバシー情報を適切に管理する」ことは非常に難しいことでした。

最新のAndroid 6.0では、ランタイムパーミッションというセキュリティモデルが導入されました。一言で言えば、「アプリインストール時にパーミッション(後述)を確認していたのが、実行時にパーミッションの許可/不許可を選択できるようになった」ということです。これによりユーザによる、プライバシーデータのコントロールがずいぶんとやりやすくなりました。

Android 6.0が主流のOSとなるのはまだ先のことですが、すでに対応しているアプリが続々と出てきています。いちユーザとしても、開発者としても、このランタイムパーミッションの良さを知っていただき、安全なスマートフォンライフを送ってほしいと思います。本記事では、このAndroid 6.0で導入されたランタイムパーミッションについて解説します。

パーミッション

Androidのアプリケーションは、OSが提供するデータや重要な機能(以降リソースと呼びます)にアクセスすることは基本的にできません。これらのリソースを使用するためには、アクセス権限(パーミッション)の使用許可を得る必要があります。たとえば、アプリケーションが位置情報を利用したいときには位置情報への

パーミッションを、電話帳のデータにアクセスしたいときは電話帳へのパーミッションを取得します。このようなしくみにより、アプリケーションがどのリソースにアクセスするかがわかるようになっています。

プロテクションレベル

パーミッションにはどれぐらいセンシティブな情報を扱うかのレベル(プロテクションレベル)が規定されています。プロテクションレベル「Normal」はManifestファイルに利用宣言をするだけで利用可能になります。プロテクションレベル「Dangerous」はユーザに使用許可を得る必要のあるパーミッションです。

Hint Android 6.0では多くのパーミッションにおいて、プロテクションレベルがDangerousからNormalに変更されています。開発経験のある方は再度見直されることをお勧めします。

インストールタイムパーミッションモデル

Android 6.0で導入されたランタイムパーミッションモデルに対し、従来のモデルをインストールタイムパーミッションモデルと呼びます。インストールタイムパーミッションモデルでは、アプリインストール時にパーミッション確認ダイアログ(**図1**)が表示されます。

ユーザはアプリが連絡先データ(電話帳)にアクセスするなど、パーミッションを通して、どのリソースにアクセスするか確認をしてインストールします。アクセスされたくないリソースがある場合は、インストールをしないという選択を取ることで自分のプライバシーデータを守ります。

しかしながら、このモデルには次のような問題があります。

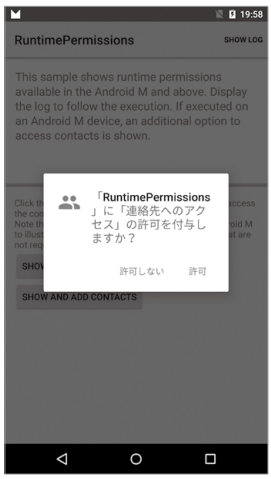
- 画面を見ても何のことだかわからない、または確認せずにインストールしてしまうユーザが多い
- アプリを使いたいのであれば「同意する」しかない。アプリを探してインストールする段階まできているため、同意してしまうことが多い
- 位置情報はアクセスしても良いが、連絡先にはアクセスしてほしくないといった選択ができない
- 何のためにプライバシーデータを使用するのかの利用目的がわからない
- アプリが取得したデータを第三者(他の会社など)に提供するのか、しないのかがわからない

ランタイムパーミッションモデル

Android 6.0で導入されたランタイムパーミッションに対応したアプリケーションでは、インストール時に**図1**のようなパーミッションの確認ダイアログは表示されません。つまり、インストール時に権限はまったく与えられません。代わりに、実際に権限が必要になったタイミングでダイアログを表示してユーザに許可を求め

▼図1 インストール時のパーミッション確認

▼図2 権限許可ダイアログ



ます。ここでユーザは許可／不許可を選択できます(**図2**)。許可されなかった場合、アプリは許可が必要なリソースへはアクセスできません[注1]。

また、Android 6.0以上では一度許可したパーミッションを後から不許可にすることも可能です。後から設定を変更するには、[設定]→[アプリ]→[アプリ情報]→[許可]から個別に許可／不許可の設定を行います(**図3**)。

ランタイムパーミッションモデルには次の長所があります。

・アプリのインストール時に権限問題を気にする必要がなく、手軽にインストールできる
・位置情報はアクセスしても良いが、電話帳へのアクセスはしないなどの選択が可能
・許可／不許可の判断を通して、ユーザがプライバシーデータの取り扱いに注意するようになる
・ユーザの許可率を上げるために利用目的や第三者提供の有無など、データの取い扱いに対して開発者側がダイアログやチュートリアルで説明をするようになる

このような利点はありますが、従来作成したアプリケーションが自動的にこのような動作になるわけではありません。開発者がランタイムパーミッションに対応したコードを書く必要があります。

注1) インストールはされているので、制限された状態でアプリが使えます。

▼図3　アプリ情報画面(左)から権限設定画面(右)へ

パーミッショングループ

ここまで、パーミッションを取得する流れを説明してきましたが、1点注意する必要があります。それは、開発者はパーミッション単位で許可／不許可の指示を行うのに対し、ユーザはパーミッショングループ単位での許可／不許可を行う点です。プロテクションレベルDangerous

▼表1　パーミッションとパーミッショングループの関係

パーミッショングループ	パーミッション
CALENDAR	READ_CALENDAR
	WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS
	WRITE_CONTACTS
	GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION
	ACCESS_COARSE_LOCATION
	CAR_SPEED
PHONE	READ_PHONE_STATE
	CALL_PHONE
	READ_CALL_LOG
	WRITE_CALL_LOG
	ADD_VOICEMAIL
	USE_SIP
	PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
	USE_FINGERPRINT※
SMS	SEND_SMS
	RECEIVE_SMS
	READ_SMS
	RECEIVE_WAP_PUSH
	RECEIVE_MMS
	READ_CELL_BROADCASTS
STORAGE	READ_EXTERNAL_STORAGE
	WRITE_EXTERNAL_STORAGE
MICROPHONE	RECORD_AUDIO
CAR_INFORMATION	CAR_VENDOR_EXTENSION
	CAR_MILEAGE
	CAR_FUEL

※USE_FINGERPRINTは、プロテクションレベルNormalです。

▼図4　連絡先へのアクセス許可ダイアログ

のパーミッションは、すべてパーミッショングループに所属しています(**表1**)。

たとえば、CONTACTSグループに所属する連絡先への読み込み権限(READ_CONTACTS)をユーザにリクエストをした場合、**図4**のダイアログが表示されます。読み込み権限のリクエストをしたのですが、「連絡先へのアクセス」の許可と表示されています。このリクエストにユーザが「許可」をした場合、連絡先への読み込み権限(READ_CONTACTS)と書き込み権限(WRITE_CONTACTS)の両方の権限が許可されます(ManifestにWRITE_CONTACTSの利用宣言がされている場合)。このダイアログは書き込みと読み込みを区別せず、連絡先に関するすべての権限の許可を求めています。この仕様はAndroidに慣れた人は少し戸惑うかもしれません。

また、開発者はこの仕様をもとに"読み込み権限があるときは書き込み権限もある"という前提でアプリを実装するのは危険です。グループのカテゴライズは今後変更になることもありえます。そのため個々のパーミッションを取得しているかを必ず確認するようにしてください。

プログラミング

次に、どのようにしてランタイムパーミッションに対応したアプリケーションを作成すれば良いのかを解説します。

Manifestへの記載

アプリケーションがパーミッションを必要とする場合は、Manifestファイルにパーミッションの利用宣言をする必要があります。ランタイムパーミッションモデルでもManifest上での利用宣言が必要です(**リスト1**)。

利用宣言したパーミッションのうち、プロテクションレベルがDangerousのものは実行時にユーザに許可を求める必要があり、許可されてはじめて利用可能になります。一方、プロテクションレベルNormalのパーミッションは、アプリインストール時に利用可能になります。

パーミッションのチェック

Android 6.0ではアプリケーションが指定したパーミッションを持っているかを確認するためのメソッドが追加されました。**リスト2**の例では、電話帳読み込み権限があるかを確認しています。

パーミッションの要求

パーミッションを取得していない場合、ユーザに許可を求めるためにrequestPermissionsメソッドを使用します。**リスト3**では、権限を取得したいREAD_CONTACTSパーミッショ

▼リスト1　Manifestファイルへの利用宣言例

```
〈uses-permission android:name="android.permission.READ_CONTACTS"/〉
```

▼リスト2　READ_CONTACTS権限の確認

```
if (checkSelfPermission(thisActivity,Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {
    //パーミッション取得していない
}
```

ンと結果を受け取るときに使用する値MY_PERMISSIONS_REQUEST_READ_CONTACTSを指定しています(この値はプログラム内で区別できる値であれば何でも構いません)。requestPermissionsメソッドを呼び出すと、OSが前述した**図4**のような許可ダイアログを表示します(ダイアログをプログラム側で作成する必要はありません)。

ユーザが許可／不許可の選択をした結果は、onRequestPermissionsResultメソッドで受け取ります(**リスト4**)。requestPermissionsで指定したMY_PERMISSIONS_REQUEST_READ_CONTACTSがrequestCodeに入ってきたとき、grantResultsの値がPERMISSION_GRANTEDの場合はユーザが許可を、それ以外の場合は不許可を選択したことを表します。

許可のボタンが押された場合は指定した権限が取得されていますので、そのまま処理を進めることができます。不許可の場合は権限を取得していませんので、そのまま処理を進めるとSecurity Exceptionが発生し、アプリケーションはハングアップします。「許可をしないと使用できない」など、ユーザに表示することが必要になるでしょう。

互換性

さて、Android 6.0でランタイムパーミッションが導入されましたが、Android 6.0端末を持っているのは、まだほんの一握りのユーザです。ランタイムパーミッションに対応したアプリケーションの下位互換性について見ていきたいと思います。

実行環境とtargetSdkVersion

ランタイムパーミッションに対応したアプリを作成したとしても、Android 6.0未満の端末ではOSが新しいパーミッションモデルに対応していないため、従来のインストールタイムパーミッションモデルで動作します。Android 6.0以上の端末で、アプリがランタイムパーミッションに対応している場合はランタイムパーミッションモデルとなり、対応していない場合はインストールパーミッションモデルで動作します(**表2**)。

Hint アプリがランタイムパーミッションに対応している場合は、Manifestファイルにtargetsdkversion=23を指定します(23はAPI Level23を表し、Android 6.0ということを表現します)。

▼リスト3　READ_CONTACTS権限の取得ダイアログの表示

```
requestPermissions(thisActivity,
    new String[]{Manifest.permission.READ_CONTACTS},
    MY_PERMISSIONS_REQUEST_READ_CONTACTS);
```

▼リスト4　許可・不許可の取得

```
@Override
public void onRequestPermissionsResult(int requestCode,
        String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
            // If request is cancelled, the result arrays
are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.
PERMISSION_GRANTED) {
                // 許可が押された
            } else {
                // 不許可が押された
            }
            return;
        }
    }
}
```

▼表2　実行環境とtargertSdkVersion

実行環境	targetSdkVersion=23以上	targetSdkVersion=23未満
Android 6.0未満	Install-Time model	Install-Time model
Android 6.0以上	Run-Time model	Install-Time model

問題のあるパターン

このような状態でひとつ問題となるのは、Android 6.0以上の端末かつインストールタイムパーミッションモデルで動作する場合です(**表2の右下のセル**)。Manifestファイルで記載されたパーミッションはすべて取得された状態で動作するのですが、アプリの権限画面(**図3の右画面**)を使用してアプリが持っている権限をはく奪することが可能です。

この場合、権限が必要な機能を実行したときにSecurityExceptionは発生せず、電話帳なら0件のデータを返すなどダミーの動作をします。必ずハングアップをするということはありませんが、今まで想定しなかった値が返ってくることがありますので別の理由でハングアップするかもしれません。

サポートライブラリ

Android 6.0で新規に作成された、checkSelfPermissionやrequestPermissionsメソッドはAndroid 6.0未満の端末では動作しません。これらのメソッドを使用してしまうと、Android 6.0以上でしか動作しないアプリケーションになってしまいます。これではユーザ数が少なく困ってしまいます。

この問題を解決するために、サポートライブラリが用意されています。このライブラリを使用することで**リスト2**や**リスト3**と同じようなコードが記載でき、6.0未満の端末ではインストールタイムパーミッションモデルで、6.0以上の端末ではランタイムパーミッションモデルで動作するようになります。しばらくは、サポートライブラリを使用して、checkSelfPermissionやrequestPermissionsなどの互換メソッドを利用して実装していくことになるかと思います。

最後に

Android 6.0で導入されたランタイムパーミッションは、ユーザにとってより良い機能です。すでに対応済みのアプリケーションも見受けられます。ランタイムパーミッションの対応は多少の学習コストはかかりますが、ユーザ目線で考えてみてください。インストールするときに権限を聞いてくるアプリと聞いてこないアプリのどちらを選択するでしょうか。おそらく、権限を聞いてこないアプリをインストールするでしょう。つまり、新しいランタイムパーミッションモデルへの対応はアプリ選択のインセンティブになります。

この記事によって、ランタイムパーミッションに対応したアプリが1つでも増えれば幸いです。SD

COLUMN

Androidのセキュリティ向上を目指して

本誌に寄稿するのはこれで4回目となります。1年に1回のペースでタオソフトウェア(株)の谷口として、Androidのセキュリティ関連の記事を書かせていただきました。現在セキュリティ業界では、サイバーセキュリティ基本法設立、個人情報保護法の改正、マイナンバー導入、オリンピック開催に向けてのセキュリティ強化といったように、国家的な情報セキュリティが注目されています。しかし、それらの規模が大きすぎてスマートフォンセキュリティの分野が積極展開されていないのを常々不安に感じていました。スマートフォンは子供からお年寄りまで使用する機器です。セキュリティをおろそかにしていいはずがありません。

このため、リスクファインダー(株)というスマートフォンセキュリティの会社をタオソフトウェアから分離する形で新たに立ち上げました。まずはアプリケーションの脆弱性診断を行いながら、スマートフォンセキュリティに関する記事やニュースをブログなどを通じて発信し、安心・安全な社会の実現に向けて少しでも手助けができたらと思っています。ご興味のある方は弊社ブログ(http://blog.riskfinder.co.jp/)をご覧ください。

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22回 auto-insert-modeでファイル新規作成を迅速に

特定の拡張子やファイル名のファイルを新規作成したとき、テンプレートが自動で挿入されると便利ですよね。今回はそれを可能にするauto-insert-modeを紹介します。既存のauto-insert-alistを変更する方法、yatemplateを使って新しく設定を追加する方法、2つのアプローチを解説します。

2016年！

ども、るびきちです。新年を迎えましたが、あなたのEmacsの調子はいいですか？　おかげさまで本連載ももうすぐ2年になろうとしています。Emacsの世界はとてつもなく広大で、これまで書いてきたことは氷山の一角にすぎません。まだまだ、あなたに伝えなければならないことは、山ほど残っています。

しかも、Emacs自身も絶え間なく進化していて、今年中にEmacs25がリリースされるのは間違いありません。Emacs25は、WebKit埋め込み機能とダイナミックリンク（拡張ライブラリ）がサポートされます。前者は、たとえばグラフィカルなWebブラウザをEmacsのバッファで表示させることができます。後者は、ユーザがC言語でEmacsの関数を記述できるようになり、elispだと遅くて実用的でなかったことが実現できるようになります。将来的には、「標準拡張ライブラリ」という形で標準関数が充実するでしょう。また、パッケージシステムも拡張ライブラリ対応になり、高速な拡張ライブラリとelispの混成パッケージも登場することでしょう。PerlのCPANや、RubyのRubyGemsと同じような方向性になるのではないでしょうか。Emacs24では、パッケージシステムにより、開発者・ユーザともに一気にelispの世界が開けましたが、Emacs25も大きな動きとなるのは間違いありません。楽しみですね！

ファイル作成前に内容を用意しておく

テンプレートとスニペット

今回は新規ファイルの作成を迅速に行う方法をお話します。

ここで、用語を明確に定義させてください。yasnippetの解説において「テンプレート」と「スニペット」を同じ意味の言葉として使ってきましたが、今回は明確に区別することにします。どちらも定型文、とくに穴埋めが含まれる文字列の雛型を意味しますが、対象が異なります。

テンプレートとは、「ファイル全体の雛型で新規作成時にあらかじめ展開（挿入）されるべき内容」を意味することとします。今回はこの「テンプレート」が話題の中心になります。

スニペットとは、「任意のタイミングで展開される定型文」を意味することとします。当然、yasnippetのスニペットは「スニペット」であり、スケルトンの類も「スニペット」になります。テンプレートはスニペットを使って定義されるので、スニペットの使用例の1つと言えます。

▼リスト1　elispのヘッダのテンプレート

```
;;; ファイル名 --- 説明

;; Copyright (C) 年 著者

;; Author: 著者 <メールアドレス>
;; Keywords: キーワード

;; This program is free software; you can redistribute it and/or modify ;; (略)

;;; Commentary:

;;; Code:

(provide 'ファイル名から拡張子を取り除いた文字列)
;;; ファイル名 ends here
```

新規ファイルのテンプレート

新規ファイルのテンプレートを扱うには、Emacs標準のauto-insert-modeを使います。グローバルマイナーモードですので、次の1行を設定に加えればテンプレートが使用できるようになります。

```
(auto-insert-mode 1)
```

テンプレートを定義する変数auto-insert-alistには、あらかじめ次のテンプレートが用意されています。

・C/C++ヘッダ
・C/C++コード
・HTML
・LaTeX
・Ada
・Man page
・elisp
・Texinfo

Emacsユーザにとって一番身近な例は、elispのヘッダではないでしょうか。elispを公開する場合、セミコロンの数も合わせて**リスト1**のテンプレートに従うよう規約に定められているのですが、毎回手作業で入力するのは馬鹿げています。そういう決まり文句は、*.elを新規作成したときにauto-insert-modeによって自動的に入力させるのが一番です。

auto-insert-alistを詳しく見てみる

auto-insert-alistは、各要素が、

```
(条件 . アクション)
```

あるいは、

```
((条件 . 説明) . アクション)
```

の形をしたリストです。条件に入るのは、ファイル名の正規表現かメジャーモードです。また、自動挿入される内容に対して説明文を付けることもできます。デフォルトの設定における条件、説明を抜き出すと、**リスト2**のようになります。

アクションはスケルトン、自動挿入されるファイルのファイル名、あるいは実行される関数を指定します。ベクタ(配列)を指定すれば、立て続けにアクションを実行できます。

auto-insert-alistを変更する

auto-insert-modeを愛用するようになると、当然auto-insert-alistを変更せざるを得なくなります。多くの場合、auto-insert-alistに新しい設定を追加することになると思いますが、それは次節に回します。ここでは、auto-insert-alistにすでに登録されている内容を変更する方法を紹介します。

実際、デフォルトのauto-insert-alistに登録

▼リスト2　デフォルトのauto-insert-alist（抜粋）

```
(("\\.\\([Hh]\\|hh\\|hpp\\)\\'" . "C / C++ header")
 ("\\.\\([Cc]\\|cc\\|cpp\\)\\'" . "C / C++ program")
 ("[Mm]akefile\\'" . "Makefile")
 html-mode plain-tex-mode bibtex-mode latex-mode
 ("/bin/.*[^/]\\'" . "Shell-Script mode magic number")
 ada-mode
 ("\\.[1-9]\\'" . "Man page skeleton")
 ("\\.el\\'" . "Emacs Lisp header")
 ("\\.texi\\(nfo\\)?\\'" . "Texinfo file skeleton"))
```

▼リスト3　個人設定ファイルの作成をシンプルに

```
(require 'autoinsert)
(dolist (x auto-insert-alist)
  (when (equal "\\.el\\'" (car-safe (car x)))
    (setcar (car x) "/src/.+\\.el\\'")))
```

されているEmacs Lisp Headerは、スケルトンでありながらもできが良いからです。しかしながら、*.elを新規作成するたびに、わざわざ「Emacs Lisp Headerを展開しますか」と聞かれるのは煩わしいものです。ヘッダが要求されるのはあくまでも公開用elispのみであり、個人設定では不要です。多くの場合、新規作成されるのは個人設定ファイルです。

そこで、Emacs Lisp Headerを展開する条件を*.elからsrc/*.elに変更し、公開用elispはsrcディレクトリに置くようにすることで、この問題を解決します（**リスト3**）。この設定内容を詳しく説明するのは少し難しいですが、setcar関数によって、auto-insert-alist中の「"\\.el\\'"」をピンポイントで、「"/src/.+\\.el\\'"」に変更しています。

ファイル作成時にスニペットを展開させる

扱いづらいauto-insert-alist

<f1> v auto-insert-alistでその値を表示してみると、誌面に掲載しきれないほど、とてつもなく長いものになります。そのため、前節では概略を説明するにとどめました。

auto-insert-modeを実用化するにあたっては、auto-insert-alistを直接変更するのは賢明ではありません。auto-insert-alistで指定できるのは、スケルトン、穴埋めなしテンプレートファイル、関数のいずれかであり、どれも中途半端です。

スケルトンは表現力こそあれど可読性が低く、スニペット展開機能としてはyasnippetに取って代わられました。

テンプレートファイルを指定しても、そのファイルの内容を字面どおり挿入することしかできません。

関数定義は一般ユーザには荷が重過ぎます。そのままでは、せいぜい穴埋めなしテンプレートファイルで我慢するのが精一杯でしょう。

yasnippetをテンプレートにできないか!?

もしyasnippetファイルをテンプレートにできれば、とてもうれしいのではないでしょうか。auto-insert-alistでアクションにベクタを指定すれば、立て続けにアクションを実行できるので、yasnippetファイルの内容を挿入してから、それをyasnippetスニペットとして展開するということが可能になります。ファイル名に規約を設ければ、auto-insert-alistの設定も自動化できるでしょう。ユーザ側がauto-insert-alistをいじらないで、yasnippetをテンプレートにしてくれるパッケージはないでしょうか？

それを行うのがMELPAにあるyatemplateパッケージです。M-x package-install yatemplateでインストールしてください。

yatemplateファイル名規約

yatemplateをインストールしたあとは、~/.emacs.d/templates以下に配置しているyasnippetファイルを、テンプレートファイルとしてauto-insert-alistに登録します。ファイル名は「**数字：正規表現**」のように指定します。

たとえばPythonのテンプレートですと、次のようになります。

```
00:test_.*.py
01:.*.py
```

数字が若いほうが優先度は高くなります。たとえば、「test_foo.py」を新規作成したときには、「~/.emacs.d/templates/00:test_.*.py」ファイルに書かれたテンプレートが採用されます。

yatemplateを使う

yatemplateを使うには次の設定をします。

```
(yatemplate-fill-alist)
(auto-insert-mode 1)
```

新しくyatemplateファイルを追加すると、自動的にauto-insert-alistに反映されます。これでauto-insert-alistをいじることなく、手軽に高機能テンプレートを設定できるようになりました。

おわりに

筆者のサイト「日刊Emacs」は日本語版Emacs辞典を目指すべく日々更新しています。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

一歩進んだ使い方のためのイロハ

Vimの細道

第5回

mattn
twitter:@mattn_jp

Vimで何でも読み書きする

今回はVimでのファイル読み書きに焦点を当てます。前半では、開くファイルを判別して、インデントや文字コードを操作するコマンドなどを自動で実行できる「autocmd」について解説します。後半では、そのautocmdが応用された、さまざまな外部コンテンツを読み書きできるプラグイン「vim-metarw」を紹介します。

Vimの魔法

Vimが便利だと言われる理由に、数多くのオプションによるカスタマイズが挙げられますが、実際はそれだけではありません。一般的なテキストエディタが行う仕事は「ファイルを読み込むこと」と「ファイルへ書き込むこと」ですが、その仕事をユーザが期待する以上に行える機能をVimは持ち合わせています。

たとえばGo言語(以下golang)で開発する際に便利なのが**vim-go**[注1]です。vim-goではgolangのソースコードを開き、:wで保存すると、統一されていないインデントであったりズレた括弧の位置であったりが、きれいに整形されるようになっています。

golangには、gofmtというソースコードを整形するプログラムが同梱されており、コマンドラインから、

```
$ go fmt main.go
```

と実行するだけで、ソースファイルmain.goがgolangで推奨される形式に整形されます。golangに慣れていない人からすると、一見インデントがハードタブに置き換えられたり括弧の位置が強制されたりと理不尽に感じることもあるかもしれません。しかし、しばらく使っているととても心地良くなっていき、慣れてしまうと、この機能がないと逆に気持ち悪く感じてしまうことさえあります。

autocmdでイベント発行

vim-goはどのように動くのか

Vimには、ファイルへの書き込みに際してイベントを発行できる機能があります。vim-goのplugin/go.vimを見ると次のコードがあります。

```
autocmd BufWritePre *.go call go#fmt#Format(-1)
```

これは*.goで表されるバッファに対して書き込み操作が行われる直前に、call go#fmt#Format(-1)を実行するという意味になります。Vimには**表1**のようなファイルI/Oに関するイベントが用意されています。このほかにもテキストが変更された際に呼び出されるTextChangedや、バッファが閉じられた際に呼び出されるBufLeave、Vimが終了する前に呼び出されるVimLeavePreなどの便利なイベントも用意されています。

注1) URL https://github.com/fatih/vim-go

autocmdとは何か

ここでautocmdの構文を見てみましょう。

```
autocmd BufNewFile,BufRead *.html echo "HTMLが読み込まれました"
```

この例では、拡張子htmlのファイルがバッファで作成された、もしくは読み込まれた際に「HTMLが読み込まれました」というメッセージを表示するものです。実際には:e foo.htmlや:new foo.htmlを実行した際にメッセージが表示されます。応用すれば、バッファだけに作用するオプションを設定したり、ファイルが書き込まれた際に特定のコマンドを実行したりもできます。

リスト1の例は、Javaのファイルを開いた際に前々回(2015年12月号)の記事で紹介したjavacomplete2の補完機能を有効にする設定です。また、ファイルを保存した際にCheckStyleというコマンドを実行し、ソースファイルの検証を行っています。

autocmdはとても便利なのですが、vimrcなどに記述して使う場合、設定の反映のために何度も:so ~/.vimrcを実行すると、複数回イベントが登録されてしまいます。

そこで、**リスト2**のように書くことが推奨されています。2行目のautocmd!は、それを内包するイベントグループ名MyFileTypeJavaで登録されたイベントを削除する命令です。こうすることで、前述の複数回イベントが実行されてしまう問題を回避できます。

コマンドイベントとは何か

表1で説明した中に、コマンドイベント(Cmd-event)というものがあります。これは、実はファイル自体は存在せず、指定のコマンドで読み込みや保存を代行するものです。

たとえば**リスト3**を見てください。このイベントを登録しておき:e edit://fooを実行すると、fooという内容のバッファが開かれます。そして、buftype=nofileが設定されているので、:wで保存できなくなっています。Vimには標準でnetrwという機能が備わっており、「http://」で始まるURLを指定してバッ

▼表1　Vimのバッファ、ファイルI/Oに関するイベント

イベント名	説明
BufReadPre	バッファにファイルが読み込まれる前に呼ばれる
BufReadPost	バッファにファイルが読み込まれた後に呼ばれる
BufReadCmd	BufReadPreのコマンドイベント
BufWritePre	バッファ全体がファイルに書き込まれる前に呼ばれる
BufWritePost	バッファ全体がファイルに書き込まれた後に呼ばれる
BufWriteCmd	BufWritePreのコマンドイベント
FilterWritePre	%コマンドなどでバッファの一部がファイルに出力される前に呼ばれる
FilterWritePost	%コマンドなどでバッファの一部がファイルに出力された後に呼ばれる
FilterWriteCmd	FilterWritePreのコマンドイベント
FileAppendPre	rコマンドなどでファイルへ追加が行われる前に呼ばれる
FileAppendPost	rコマンドなどでファイルへ追加が行われた後に呼ばれる
FileAppendCmd	FileAppendPreのコマンドイベント
FileWritePre	バッファの一部がファイルに書き込まれる前に呼ばれる
FileWritePost	バッファの一部がファイルに書き込まれた後に呼ばれる
FileWriteCmd	FileWritePreのコマンドイベント
FileType	指定のファイルタイプが設定された際に呼ばれる

▼リスト1　Javaファイルを開くと、javacomplete2が有効になる設定

```
autocmd FileType java setlocal omnifunc=javacomplete#Complete
autocmd BufWritePost *.java Checkstyle
```

▼リスト2　autocmdを使ううえでの推奨設定

```
augroup MyFileTypeJava
  autocmd!
  autocmd FileType java setlocal omnifunc=javacomplete#Complete
  autocmd BufWritePost *.java Checkstyle
augroup END
```

▼リスト3　Cmd-eventの例

```
autocmd BufReadCmd edit://foo :call setline(".", "foo")|setlocal buftype=nofile
```

ファを開けますが、それはこのBufReadCmdを使って実現されています。

読み込みは良しとして書き込みはどうでしょうか？　BufWritePreやBufWritePostなどは、フックはできても実際にファイルを作ってしまいます。そこで必要になるのが、buftype=acwriteというオプションです。このオプションをバッファに設定しておくと、BufWriteCmdで設定されるコマンド以外ではバッファが保存しなくなります。

コマンドイベントを使ったプラグインを作る

これを応用して、Javaのpropertiesファイルを編集できるプラグインを作ってみましょう。

まず、Javaのpropertiesファイルは ISO-8859-1でエンコードされ、Unicodeを含ませたい場合には「\uXXXX」というUnicode表記でエスケープして記述する必要があります。Javaには専用にnative2asciiコマンドが付属しており、このコマンドを使用することでエスケープされたファイルから実際のファイルへ相互変換できるようになっています。ただし、native2asciiはWindows上では現在のコードページでエンコード／デコードを行うので、iconv()関数を使ってシステムロケールが使うエンコーディングに変換する必要があります(**リスト4**)。

▼リスト4　Javaのpropertiesファイルを編集できるプラグイン

```
function! s:native2ascii(...) abort
  " native2ascii -reverse コマンドを使用してデコードする。
  " ただし native2ascii の出力はシステムロケール依存なので
  " バッファのエンコーディングに合わせて iconv() で変換する。
  let r = split(iconv(
  \  system('native2ascii -reverse ' . shellescape(expand('%:p'))),
  \    "default", &encoding), "¥n")

  " 出力された結果でバッファを更新する。
  call setline('.', r)

  " :w 時に s:ascii2nativeが呼ばれるように設定する
  setlocal buftype=acwrite
endfunction

function! s:ascii2native(...) abort
  " バッファの内容をnative2asciiコマンドを使用して
  " ISO-8859-1 形式に変換する。
  " ただし native2ascii が期待する入力はシステムロケール依存なので
  " iconv() で変換する。
  let r = split(system('native2ascii',
  \  map(getline(1, '$'), 'iconv(v:val, &encoding, "default")')), "\n")

  " 出力された結果をファイルに書き込む
  call writefile(r, expand('%:p'))

  " 変更済みとする
  setlocal nomodified
endfunction

augroup JavaProperties
  au!
  autocmd BufReadCmd *.properties call <SID>native2ascii()
  autocmd BufWriteCmd *.properties call <SID>ascii2native()
augroup END
```

今回はsystem()関数とiconv()関数を使いましたが、native2asciiコマンドを使い、

```
:%!native2ascii➥
-reverse
```

を実行することでも同じ結果が得られます。ただしこの場合、file encodings(ファイル読み込み時の文字コードの設定)の内容に依存してしまうため、場合に

▼表2　vim-metarwでできること

metarwプラグイン	機能
mattn/vim-metarw-gdrive	Google Drive上のファイルを読み書き
mattn/vim-metarw-simplenote	メモ取りサービスSimpleNote上のコンテンツを編集
mattn/vim-metarw-webdav	WebDAV上のコンテンツを編集
emonkak/vim-metarw-gist	GitHubのGistを編集
kana/vim-metarw-git	Git上のファイルを閲覧
ujihisa/blogger.vim	ブログサービスBloggerのコンテンツを編集
joker1007/vim-metarw-qiita	情報共有サービスQiitaの記事を編集
hara/vim-metarw-tumblr	Tumblrのテキストを編集
joker1007/vim-metarw-github-issues	GitHub IssuesをVimから編集
mattn/vim-metarw-redmine	VimからRedmineのチケットを投稿／編集

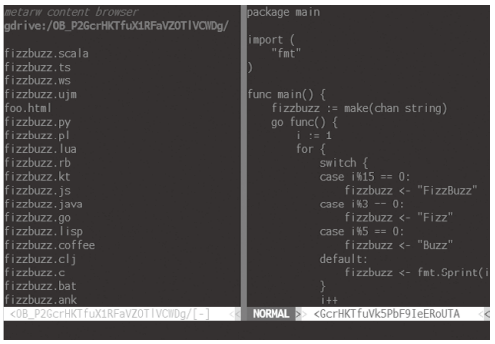
よっては誤変換が発生する可能性があります。

また、**リスト4**を使うことで、Windowsでもほぼ期待どおりに動くのですが、機種依存文字を使ってしまったり、元のpropertiesファイルに、Unicodeにしか存在しない文字が含まれていたりすると、正しく変換できません。ちゃんとした変換を期待する場合は、**vim-edit-properties**[注2]を使うことをお勧めします。

このように、autocmdがあれば実際にファイルが存在していなくても、もしくは実際のファイルの内容と異なっていても、Vim上で可視化された状態で開くことができ、保存と同時に元のファイルへ還元できます。筆者が思うに、このファイルのI/Oに特化したautocmdを一番うまく応用しているのが、**vim-metarw**[注3]です。

vim-metarwは単体では何も機能せず、vim-metarwの拡張プラグインとともに動作します。vim-metarwはファイルシステムのURL(っぽい)スキーマと、そのI/Oの受け渡しを担うフレームワークです。また、ファイル一覧を表示するUIも提供しており、拡張プラグイン側が得たコンテンツやファイル一覧を引き渡すことで、あたかもそこにファイルが存在するかのようにさまざまなコンテンツを扱えるようになります。

▼図1　vim-metarw-gdriveでGoogle Drive上のファイルを読み書き



たとえば、vim-metarwとその拡張プラグインを使うことで、**表2**のようなことが実現できます。これらプラグインは、すべてGitHub上のプラグインです。たとえば、**mattn/vim-metarw-gdrive**(**図1**)をインストールしたい場合、vim-plugをお使いであれば、

```
Plug 'kana/vim-metarw'
Plug 'mattn/vim-metarw-gdrive'
```

のように設定して、:PlugInstallとすることでインストールできます。

vim-metarwが拡張プラグイン側に要求するのは、**リスト5**の3つのAPIのみです。この場合スクリプトファイルは、autoload/metarw/foo.vimに格納するだけで、metarwが自動で読み込んでくれます。

vim-metarwの拡張のしくみは、実はそれほど難しくありません。たとえば、vim-metarw-webdav

注2) URL https://github.com/kamichidu/vim-edit-properties
注3) URL https://github.com/kana/vim-metarw

▼リスト5　vim-metarwが拡張プラグイン側に要求するAPI

```
function! metarw#foo#complete(arglead, cmdline, cursorpos)
  " 補完候補を返す
endfunction

function! metarw#foo#read(fakepath)
  " 引数fakepathに対してディレクトリもしくはファイルのコンテンツを返す
endfunction

function! metarw#foo#write(fakepath, line1, line2, append_p)
  " 引数fakepathに対してバッファのline1からline2までを書き込む
  " append_pが指定されている場合は追加書き込み
endfunction
```

であれば、WebDAVとのI/Oはすべてdavcというクライアントソフトが受け持つため、vim-metarw-webdavのソースコードはわずか100行程度しかありません。あとは、metarwが拡張プラグインの名前をスキーマ名として扱い、

```
:e foo:/
```

を実行した際に前述の関数を呼び出すことで、実在しないファイルを取り扱えるようになっています[注4]。

◆ ◆ ◆

今回はautocmdイベントの基本的な使い方と、metarwを使ったVimの拡張方法を紹介しました。metarw拡張の実装はそれほど難しくないので、ぜひおもしろいものを作ってブログなどで紹介してください。また、autocmdイベントを使った新しいアイデアが見つかれば、ぜひおもしろいプラグインを実装してみてください。SD

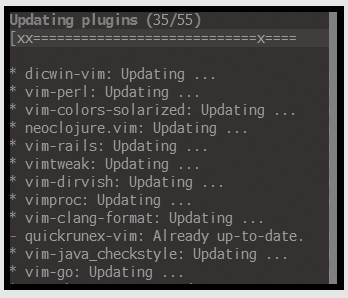
注4）vim-metarw-webdavであれば`:e webdav:/`

Vim自報

プラグインマネージャの乱立

昔、Vimプラグインはtar ballやzipファイル、もしくはvim ball（vba）と呼ばれる独自の配布形態でユーザに提供されてきました。しかし、そのあとTim Pope氏のpathogenの登場、GitHubやGitの発展により、Vim plugin導入の敷居がぐっと下がりました。巷で有名なものとしては**表A**のものがあります。筆者もつい数ヵ月前まではsunaku/vim-unbundleを使っていたのですが、プラグイン更新の速さに魅かれてvim-plug（**図A**）を使うようになりました。新しい環境を構築する際は、プラグインを一括でたくさんインストールすることになるのでとても重宝しています。

▼図A　vim-plugのUI

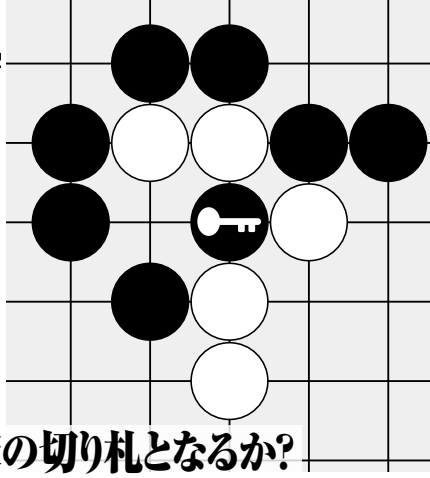


▼表A　おもなプラグインマネージャ

プラグイン名	入手先	遅延読み込み	特徴
vim-pathogen	tpope/vim-pathogen	×	runtimepathを使った基本的機能を提供
Vundle.vim	VundleVim/Vundle.vim	○	世界的に有名な高機能プラグインマネージャ
Neobundle	Shougo/neobundle.vim	○	日本で有名な拡張性の高いプラグインマネージャ
vim-plug	junegunn/vim-plug	○	高速なプラグイン更新と斬新なUI

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

【第二九回】DNSシンクホールはマルウェア対策の切り札となるか?

今回は一般のユーザにはあまり耳慣れない「DNSシンクホール」を取り上げます。ボットネット／標的型攻撃対策として、数年前から使われている手法です。C&Cサーバへの接続を阻止する、C&Cサーバへの接続を観測することでマルウェア感染を発見する、ということが可能になります。実際に設置／運用するのは、かなり上流(上級)のネットワーク管理者でしょうが、このようなしくみを知っておくと、いろいろなことが見えてくるかもしれません。また最後に、安全なDNSサービスについても紹介します。

DNSの役目

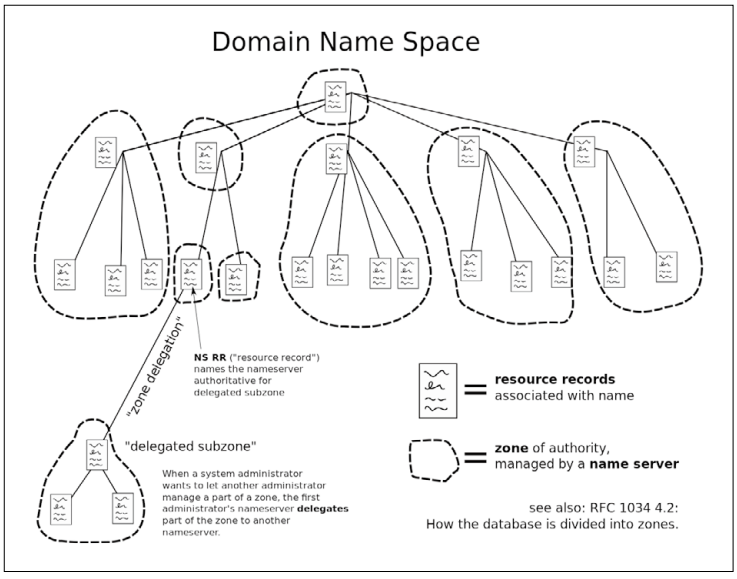
「DNSシンクホール」の議論を進める前に、あらためてDNSのしくみと役目を確認してみます。DNS(Domain Name System)の最大の役割は、クライアントからの「このドメイン名／ホスト名に対応するIPアドレスは何か?」という問い合わせに対して、正しいIPアドレスを返答することです。たとえば、DNSサーバにwww.example.comというホスト名を問い合わせると93.184.216.34というIPアドレスを返答してくれます。

問い合わせのことを「DNSクエリ(query)」、返答のことを「DNSクエリレスポンス(query response)」と言います。ドメイン名／ホスト名に対応するIPアドレスを得ることを一般的には「名前を解決(resolve)する」と呼んでいます。

DNSは問い合わせに対して回答を返すわけですが、1つのDNSサーバがインターネット全体のホスト名とIPアドレスの組み合わせを、すべて知っているというわけではありません。名前空間全体がツリー構造になっている分散型データベースだと言えます(**図1**)。

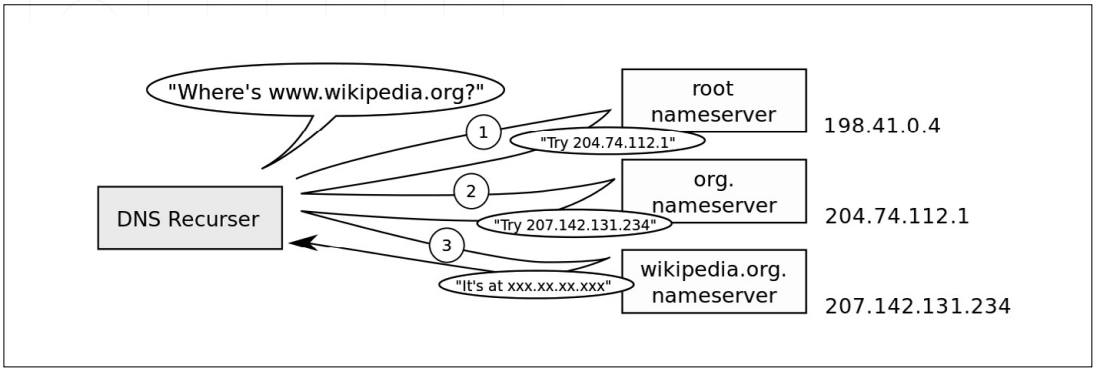
名前解決の流れを説明します。クライアントは、まず自分の直接使っている(指定している)DNSサーバにクエリを発行し、名前を解決(IPアドレスを問い合わせ)しようとします。そのDNSサーバに情報がなければ、rootネームサーバに問い合わせ、

◆**図1　ドメイン名空間**



概念的にはrootネームサーバを頂点としたツリー構造になっている。
(出典:Wikipedia英語版 Domain name space.svg(02:43, 7 October 2014) **https://en.wikipedia.org/wiki/Domain_Name_System#/media/File:Domain_name_space.svg**)

◆図2　DNSへの問い合わせの流れ



概念的な説明ではあるが、このような流れになってホスト名が解決される。
（出典：Wikipedia英語版 An_example_of_theoretical_DNS_recursion.svg (05:07, 25 October 2005)　**https://en.wikipedia.org/wiki/Domain_Name_System#/media/File:An_example_of_theoretical_DNS_recursion.svg**

そこから情報を得て、さらに下位のネームサーバへと問い合わせを行います。

たとえば、www.wikipedia.orgの名前を解決する例を考えてみます（**図2**）、クライアントは、まずrootネームサーバに問い合わせをします（**図2-①**）。すると、orgドメインを管理しているネームサーバの情報が返ってくるので、今度はorgネームサーバに問い合わせをします（**図2-②**）。そこからはwikipedia.orgドメインを管理しているネームサーバの情報が返ってくるので、さらにwikipedia.orgネームサーバに問い合わせをします（**図2-③**）。そこでwww.wikipedia.orgの情報が管理されているため、やっとIPアドレスがわかります。

日本語でのDNSのしくみの概要説明は、JPNICのサイト[注1]にわかりやすいものがありますので、そちらも参照してみてください。

DNSシンクホール

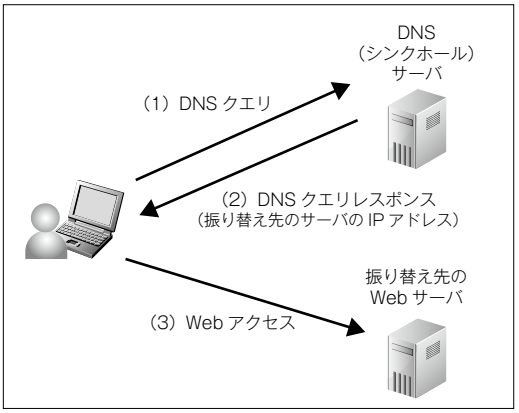
「シンクホール（sinkhole）」というキーワードでGoogleの画像検索を行うと、地面に大きな穴があいた不気味な画像がたくさん出てくると思います。

DNSシンクホールをブラックホールDNSと呼ぶ人もいます。これは特定ホスト名のDNSクエリに対して反応しない、あるいは違うIPアドレスを返す設定をした、DNSクエリの落とし穴となる特殊なDNSサーバです。

たとえば、evil.example.comというホストにボットをコントロールするためのC&C（Command-and-Control）サーバが用意されていると仮定します。さらに、そのホスト名がC&Cサーバであることは、すでに判明しているとします。そして、マルウェアがISPレベルや、あるいは国の単位で蔓延（まんえん）しているということにします。

その前提の場合、マルウェアが感染したクライアントは、C&Cサーバのホスト名を解決するためにDNSサーバにDNSクエリを発行します。このと

◆図3　DNSシンクホールのしくみ



（1）DNSクエリ、（2）DNSクエリレスポンス、（3）Webアクセス（ホストへのアクセス）、という流れになる。

注1）「インターネット10分講座：DNS」　**https://www.nic.ad.jp/ja/newsletter/No22/080.html**

き、DNS(シンクホール)サーバは、C&Cサーバの IPアドレスではなく、別途、振り替え先として用意しておいたサーバのIPアドレスを、DNSクエリレスポンスで返します。クライアントには振り替え先サーバのIPアドレスが返ってくるわけですから、当然、次にクライアントがアクセスするサーバは、その振り替え先のWebサーバになります(**図3**)。

これでevil.example.comをアクセスしようとしたクライアントはすべて、DNSシンクホールが導いた振り替え先のサーバにアクセスすることになり、そこでマルウェアの存在をキャッチすることが可能になります。

マルウェアの解析が行われ、C&Cサーバがどこにあるか知られていて、すでにテイクダウンしている(セキュリティ対応してサーバを停止している)場合でも、マルウェアそのものの感染が続いていることはよくあります。あるいは、標的型メールが送られて、それが読まれ感染するタイミングより、分析やテイクダウンのほうが早いような場合も考えられるでしょう。そのような状況のときに、DNSシンクホールを活用すれば、マルウェアに感染しているクライアントが捕捉可能となります。

また、クライアントがマルウェアに感染しボットが組み込まれ、DDoSの攻撃ノードとなってしまっているような場合には、DNSシンクホールサーバが、攻撃先のアドレスを127.0.0.1(自分のコンピュータ)と返すことで、攻撃パケットを封じ込める(感染したクライアントが自らを攻撃し始める)ことも可能となります。

あと、セキュリティのためだけではなく、アクセス制御に応用することも可能です。たとえば、学校からポルノサイトなどを覗(のぞ)くことができないよう(そして、DNSクエリ記録が残るよう)に、このDNSシンクホールを使うこともできます。

DNSシンクホールの基本的な考え方は参考文献[1][注2]がわかりやすいかと思います。

HOSTS.TXT

ドメイン名解決のための分散データベースとも言えるDNSサーバができる前は、どうしていたかというと、SRI(Stanford Research Institute)のサーバ上に、「HOSTS.TXT」というファイルが用意されていました。そのファイルには、ホスト名とIPアドレスの組み合わせのリストが入っており、利用者は定期的にHOSTS.TXTをダウンロードして使っていました。今日のインターネットと呼ばれる広域ネットワークでDNSが動き始めたのは、1984年以降です。

今でもHOSTS.TXTの機能は生きています。UNIX系では、

```
/etc/hosts
```

に、Windows系では、

```
%SystemRoot%\System32\drivers\etc\hosts
```

に、用意されています。

スマートフォンなどのiOSもAndroidも両方とも

◉今でも/etc/hostsを使う意義はある

筆者のTCP/IPの利用は、1985年にEthernetにつながれTCP/IPで通信する機能を持ったVAX DEC 780上の4.2BSDからです。そのころは、まだDNSを用意するといった時代ではなく、社内標準のhostsファイルを定期的に自分でダウンロードしてきて、/etc/hostsを更新する方法を採っていました。

今でも筆者のオフィス環境では、常用するすべてのGNU/Linuxマシンは固定IPアドレスです。DHCPを使って、クライアントのMACアドレスをもとに固定でIPアドレスを割り当てる場合でも、/etc/hostsに明示的に書いています。これは、使っているDNSリゾルバやDHCPサーバが死んでも、相互に通信できるようにしているためです。

注2) 参考文献[1] Rick Wanner, "DNS Sinkhole", SANS Institute InfoSec Reading Room, August 7, 2010
https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523

UNIXの子孫ですので、

```
/etc/hosts
  ↑ /private/etc/hosts へのシンボリックリンク
```

で参照することが可能です。

hostsを使ったファーミング

名前解決する際には、DNSよりも、hostsのファイル中にあるホスト名とIPアドレスの組み合わせが優先されます。

ホスト名でアクセスしようとして、正しくないIPアドレスをつかまされ不正サイトへ誘導されることを、「ファーミング(Pharming)」と呼びます。DNSを乗っ取る、あるいは、DNSの脆弱性を使って誤った情報を送り込み、DNSクエリに対して偽のIPアドレスを返答させることでファーミングは実現可能となります。

しかし、ある種のコンピュータには、もっと簡単な方法があります。それはマルウェアでhostsファイルの書き換えを行い、ユーザを不正なサイトへ誘導することです。近年まで、パソコンのアクセス管理は、管理者権限も一般ユーザ権限もあいまいで、システム上の任意のファイルにクリック1つで書き込んでしまえる程度のものでした。そのような環境では、マルウェアにより簡単にhostsファイルが書き換えられ、ファーミングができてしまいます。

DNSシンクホール活用のケーススタディ

韓国

韓国のナショナルCSIRTであるKrCERT/CC[注3]が、2005年にDNSシンクホールのプロジェクトを立ち上げています[注4]。国全体をカバーするDNSシンクホールの運用に関しては、10年を越える経験を積んでいます。

2009年7月7日と2011年3月4日に、韓国国内で発生した2つの大規模なDDoS攻撃において、DNSシンクホールがずいぶん役にたったようです。ISPと協力し、クライアントとC&Cサーバを接続させないためにDNSシンクホールを活用しました。また、ボット数(ゾンビコンピュータ数)も計測しており、2009年のDDoSの際のボット数は約11万5,000台、2011年の際は約11万6,000台とあります。2009年と2011年でほぼ同じ台数を計測したのは、興味深いところだと思います。

韓国ではメジャーなISPを含む60組織と協力して、DNSシンクホールのスキームを運用しているとのことですが、これは以前[注5]に説明した韓国の特徴が関係していると筆者は考えます。その特徴とは、韓国では少数のDNSサーバで全体をカバーするスタイルを採っていることです。ですから、60組織分のDNSサーバなんていうのは日本の感覚から言えば、ごく少数のような気がしますが、韓国の場合、かなりのカバー率であろうと推察します。

ポーランド

ポーランドのCERT Polskaも韓国と同様に、ボットネット対策としてDNSシンクホールの手法を取り入れ大きな成果[注6]を挙げています。

少し話は変わりますが、CERT Polskaは、ポーランドの研究ネットワーク管理組織NASKの中に作られています。初期のインターネットが研究分野で発達したこともあり、CSIRTの草分けであるCERT/CCができたのも、カーネギーメロン大学の研究所であるSoftware Engineering Instituteの中です。ヨーロッパでも研究ネットワーク管理組織や大学が、その国の最初のCSIRTを立ち上げる

注3) 日本のナショナルCSIRTであるJPCERT/CCは独立した組織ですが、KrCERT/CCはKISA(Korea Information Security Agency)の1部門として存在しています。

注4) 現在は、KISAがDNSシンクホールのプロジェクトの主催者となっています。
(1) Heung Youl YOUM, "Korea's experience of massive DDoS attacks from Botnet."
http://www.itu.int/en/ITU-T/studygroups/com17/Documents/tutorials/2011/ITU-T-ddos-tutorial-20110412-hyyoum.pdf
(2) DNSシンクホールの申し込みページ http://www.boho.or.kr/webprotect/dnsSinkhole.do

注5) 本誌2014年10月号、本連載第14回「現実の脅威となったDNSサーバへのDDoS攻撃」を参照。

注6) CERT Polska REport 2013 http://www.cert.pl/PDF/Report_CP_2013.pdf

ケースが多かった歴史的経緯があります。ドイツで最初のCSIRTであるDFN-CERTはハンブルク大学の組織の1つですし、オランダのSURFnet(オランダの研究教育ネットワーク組織)、スペインのRedIRIS(スペインの学術/研究ネットワーク)も、その国で最初のCSIRTを立ち上げています。

さて、話をもとに戻します。

「Virut」として名前が知られているボットネットのインフラがあります。これは2006年以来、知られている最悪なボットネットのインフラの1つです。2013年1月、CERT PolskaはVirutを壊滅状態に追い込むという快挙を成し遂げました[注7]。もちろん、DNSシンクホールだけで壊滅状態にさせたわけではありません。しかし、DNSシンクホールはボットネットと戦うための強力な武器として使えたのは言うまでもありません。

日本

日本のJPCERT/CCでも、DNSシンクホールを運用[注8]しています。しかし、こちらのほうはレジストラから攻撃者が放棄したドメインを購入し、ネームサーバをJPCERT/CCが管理するサーバへ向けるように設定しています。このようにしてマルウェア感染し、ボット化しているコンピュータからの通信をキャッチすることが可能になります。それらのデータはJPCERT/CCの分析システムで処理を行って、マルウェア感染の実態の把握などに役立てています。

オープンに使えるDNSシンクホール

これはパソコンを使う一般ユーザでも利用できる有用なお話です。DNSシンクホールと銘打つわけではありませんが、DNSシンクホールのような機能も入っている、トータルで安全かつオープンなDNSのサービスがあります。

ここでは2つ紹介したいと思います。1つはSymantec社の「Norton ConnectSafe」です。

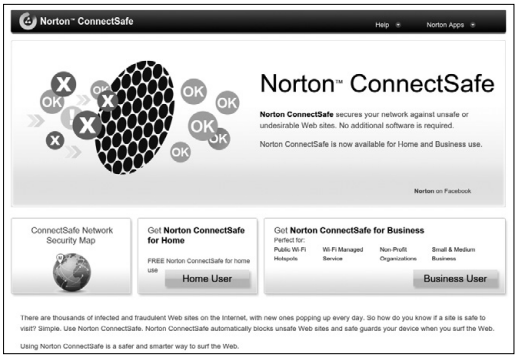
- Norton ConnectSafe(図4)
  https://dns.norton.com/

もう1つはOpenDNSです。OpenDNSは、今はCisco社の傘下に入った組織になっています。

- OpenDNS(図5)
  https://www.opendns.com

両方とも、個人で使う範囲では無料で使うことができます。企業や組織向けの有料サービスもあります。DNSは止めることのできないサービスですから、ビジネスでの利用を検討してみるのも良い選択肢かもしれません。SD

◆図4　Norton ConnectSafeのWebサイト



Symantec社の基本的なセキュリティ機能が含まれた、無料で公開されているDNSサーバ。現状、安全なDNSサーバの管理は難しくなってきており、安全なDNSを手軽に使うというニーズは高まっている。

◆図5　OpenDNSのWebサイト

OpenDNS社が提供するフリーなDNSサービス。セキュリティとしては、フィッシングフィルター機能、ドメインブロック機能などを備えている。

注7) Thomas Morrison, "Cooperative Efforts To Shut Down Virut Botnet" http://www.spamhaus.org/news/article/690/
注8) ログを活用した高度サイバー攻撃の早期発見と分析 https://www.jpcert.or.jp/research/apt-loganalysis.html

Erlangで学ぶ並行プログラミング

Author 力武健次技術士事務所 所長 力武 健次（りきたけ けんじ） http://rikitake.jp/

第11回 NIFによる外部プログラムやライブラリとの連携

この連載ではプログラミング言語Erlangとその並行プログラミングについて紹介していきます。今回はErlangとNative Interface Function（NIF）による外部プログラムやライブラリとの連携の方法について紹介します。

OTP最新版の状況

本稿の脱稿時のErlang/OTPの最新版は11月27日のパッチリリースである18.1.5です注1[1]。sshモジュールでPuTTYとerlとの交信をできるようにするなどの機能拡張とバグ修正が行われています。

Erlang仮想マシンと外部とのコミュニケーションの方法

Erlang/OTPはファイルの読み書きやネットワーク通信など多くの入出力機能を持っていますが、現実の問題解決には外部プログラムとの連携が不可欠です。ほかの言語で書いたほうが高機能あるいは速いプログラムもあるでしょうし、もともと存在するプログラムを呼び出したほうが全体の作業工数を短縮できます。

Erlang/OTPで最も簡単に外部プログラムを呼び出すには、`os:cmd/1`という関数を使います。これはOSのシェルを起動してコマンドを渡し、その結果を戻すといういたって単純な関数です。システムの初期化や設定など逐次実行の局面では、この関数で十分な場合が多いでしょう。しかし、実際の運用でOSのシェルを介さずに直接プログラムを起動したい場合も多いでしょうし、速度を重視するためにErlangの仮想マシンBEAMにライブラリをリンクしたいという場合もあるかもしれません。そこでErlangでは外部プログラムに対し、次のコミュニケーションの方法を用意しています。

ポート[2]

外部のプログラムあるいはファイルディスクリプタに対する接続口です。ポートにはErlangのメッセージパッシングを使い情報の送受を行うことができます。前述の`os:cmd/1`も内部ではポートで実現されています。

リンクドインドライバ（linked-in driver）[3]

これはポートのコミュニケーションの対象を、外部プログラムではなく、共有ライブラリとしてErlangに読み込むことでより高速な動作を可能にするものです。ただし、共有ライブラリですから、誤動作が発生するとBEAM全体に影響してしまうという問題があります。

Native Interface Function (NIF)[4]

これは外部の（おもにC言語で書かれた）コードに対して直接Erlangの関数として呼べるようにすることで、オーバーヘッドを最小にして実行するものです。算術計算の高速化など、Erlangの不得意な分野での逐次実行の速度を

注1）最新版はGitHubリポジトリを使いタグを指定することでビルドできます。詳細は「kerlでGitHub版のErlangをインストールする」（http://qiita.com/jj1bdx/items/4f7d7b5a53fcec32ab8d）を参照してください。

上げるのに適しています。コードは共有ライブラリとしてロードする必要があり、リンクドインドライバ同様に、誤動作が発生すると最悪の場合BEAM全体が動作停止するという問題が発生します(**図1**)。

Cノード[5]

BEAM同士の分散プログラミングと同じプロトコルを使うプログラムを(おもにC言語で)書くことで、分散ノードの1つとして外部のプログラムを扱うことができます。

◆ ◆ ◆

これらの手法については、Erlang/OTPのマニュアルにガイド[6]が用意されています。今回は上に紹介した4つの方法のうち、筆者が過去開発したコードの高速化に使ってきたNIFについて紹介します。

疑似乱数Xorshift64*を例にしたNIFの作り方

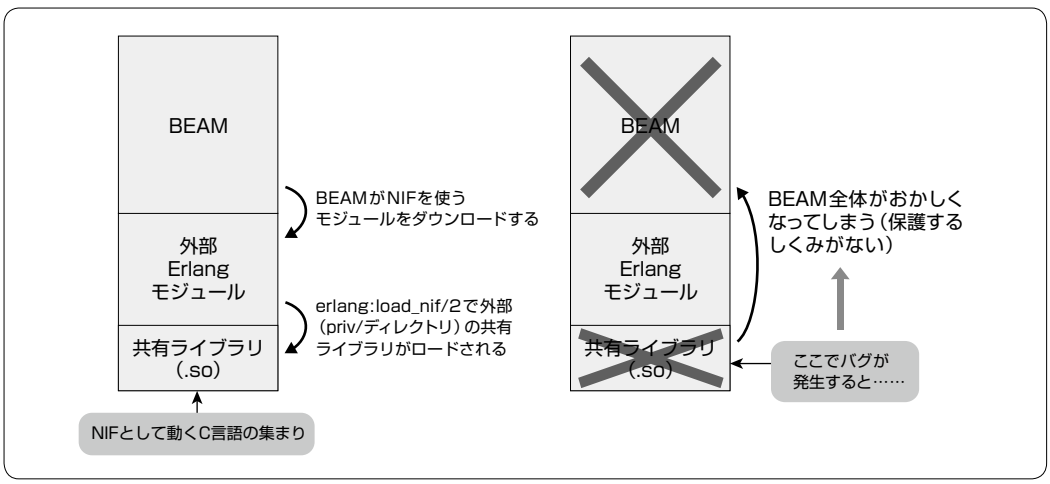
Erlang/OTPではNIFを書くためのC言語のライブラリとして、`erl_nif`[7]が用意されています。`erl_nif`のおもな機能は、Erlangの各種データ構造の型の判定やC言語の各種変数からのErlangの項やバイナリの作成、処理に必要な動的メモリ管理、排他制御やC言語のスレッド制御、スケジューラへの状態通知など、多岐にわたります。別の言い方をするなら、Erlangを普段書くときにBEAMやコンパイラがやってくれていることを自分でやるための関数が一式用意してあると考えることもできます。

NIFは一般的にはC言語あるいは同等のAPIを提供する言語で書かれ、コンパイル時に`erl_nif`をリンクしたC言語の共有ライブラリ(`.so`)となります。これを実行時に`erlang:load_nif/2`でロードすることで、使用可能になります。NIFを使うためには、対応するErlangで書かれたモジュールに、スタブ(仮の置き場)となる関数を定義する必要があります。これらのスタブをNIFの共有ライブラリで上書きすることで、NIFが使えるようになります。実際には上書きのために、「`-on_load(モジュール内関数名).`」という指示を与えて、モジュールロード時にプロセスとして実行する関数を指定します。

NIFの例として、Xorshift64*[8]という疑似乱数(以下「乱数」)の実装例の一部を紹介します[注2]。この乱数は64ビットの符号なし整数で表現される内部状態を持ち、一度呼ぶごとに64

注2) このアルゴリズムはOTPのrandモジュールで選択できます。筆者はOTP 17.x以前でも動作する独立した実装を公開しています(https://github.com/jj1bdx/exs64/を参照)。

▼図1 NIFでバグが発生したときの問題



ビットの整数を生成します。比較的単純なアルゴリズムですが、周期は$(2^{64}-1)$と、内部状態の数が少ないという制約を考えれば実用上遜色のない性質を持っています。

リスト1にErlangによるXorshift64*実装例、またNIFのスタブ関数を示します。Erlangの整数は多倍長のため、出力が64ビットを越えないようにビット演算子bandでマスクする必要があり、かつ内部表現は64ビットのBEAMでは60ビットまでしか表現できない[9]ため、実行速度が遅くなるという問題があります。この差は多くの要素を持つ乱数列を一度に生成するときにとくに顕著になります。そこで**リスト1**ではexs64m:next_list/2という乱数列を生成する関数を用意し、かつこれをNIFとして実装したexs64m:nif_next_list/2というスタブ関数を用意しました。

リスト2に、exs64m:nif_next_list/2のC言語で書いたNIFによる実装を示します。次に箇条書きで構成の概要を示します。

▼リスト1　Xorshift64* アルゴリズムのErlangによる実装exs64mモジュールの一部

```
疑似乱数Xorshift64*のモジュールexs64mの抜粋
-module(exs64m).
モジュールのロード時に実行する関数。NIFの共有ライブラリを先にロードする際使用する
-on_load(load_nif/0).
export宣言は省略、以下は型宣言
-type uint64() :: 0..16#ffffffffffffffff.
-opaque state() :: uint64().
-define(UINT39MASK, 16#0000007fffffffff).
-define(UINT64MASK, 16#ffffffffffffffff).
Xorshift64* アルゴリズムの計算部
-spec next(state()) -> {uint64(), state()}.
next(R) ->
    R1 = R bxor (R bsr 12),
    R2 = R1 bxor ((R1 band ?UINT39MASK) bsl 25),
    R3 = R2 bxor (R2 bsr 27),
    {(R3 * 2685821657736338717) band ?UINT64MASK, R3}.
末尾再帰によるループで指定された個数の要素を持つ疑似乱数のリストを生成する。結果のリストと更新された内部状態を返す
-spec next_list(pos_integer(), state()) -> {[uint64()], state()}.
next_list(N, S) when is_integer(N), N > 0 -> next_list(N, S, []).
リストAの先頭に結果を蓄積し、最後に反転して結果を得る
next_list(0, S, A) -> {lists:reverse(A), S};
next_list(N, S, A) ->
    {V, S2} = next(S),
    next_list(N-1, S2, [V|A]).
NIFのスタブ（仮の関数）とバージョン番号のマクロ
-define(nif_stub, nif_stub_error(?LINE)).
-define(NIF_LOAD_INFO, 101).
未定義の場合のエラーを返す
nif_stub_error(Line) ->
        erlang:nif_error({nif_not_loaded, module, ?MODULE, line, Line}).
NIFで定義した関数のスタブ部分（マクロ）
-spec nif_next_list(pos_integer(), state()) -> {[uint64()], state()}.
nif_next_list(_, _) -> ?nif_stub.
このモジュールをロードするときにNIFを先にロードするための関数。priv/という名前のディレクトリに共有ライブラリを置く
load_nif() ->
    PrivDir =
        Erlangライブラリのpriv/ を探す
        case code:priv_dir(?MODULE) of
            {error, _} ->
                見つからない場合はモジュールのある場所から相対パスでpriv/の位置を設定する
                EbinDir = filename:dirname(code:which(?MODULE)),
                AppPath = filename:dirname(EbinDir),
                filename:join(AppPath, "priv");
            Path -> Path
        end,
        見つけたpriv/のディレクトリから共有ライブラリをロードする
    erlang:load_nif(
        filename:join(PrivDir, "exs64m_nif"), ?NIF_LOAD_INFO).
```

▼リスト2　exs64mモジュールのNIFのC言語によるコードexs64m_nif.cの一部

```
/* 誌面に限りがあるため関数は抜粋しています */
/* またC言語のヘッダーファイルは省略しています */
/* Xorshift64*生成関数で使う構造体と型の定義です */
struct EXS64_RETVAL {
    uint64_t x;     /* 乱数の返り値 */
    uint64_t state; /* 乱数の内部状態 */
};
typedef struct EXS64_RETVAL exs64_retval;
/* モジュール中の名前がどのCの関数に対応するかのリストです */
/* nif_next_list/2に対応するのがexs64m_nif_nif_next_list()です */
static ErlNifFunc nif_funcs[] = {
    {"nif_next_list", 2, exs64m_nif_nif_next_list}
};
/* NIFの初期設定を行います */
ERL_NIF_INIT(exs64m, nif_funcs, load, NULL, upgrade, unload)
/* exs64m:nif_next_list/2として呼ばれる関数です */
static ERL_NIF_TERM
exs64m_nif_nif_next_list(ErlNifEnv *env, int argc, const ERL_NIF_TERM argv[])
{ /* argv[0]のErlangの型: uint64(),argv[1]のErlangの型: state() (uint64()と同値)*/
    ErlNifUInt64 len, state;
    uint64_t i;
    ERL_NIF_TERM *terms, list;
    exs64_retval new;
    /* 引数を取得し、値の正当性をチェックします */
    /* ここでは長さと内部状態がゼロでないかの確認をします */
    if (!enif_get_uint64(env, argv[0], &len)
        || len == 0LL
        || !enif_get_uint64(env, argv[1], &state)
        || state == 0LL ) {
        /* 問題があった場合はbadargというアトムを返して終了します */
        return enif_make_badarg(env);
    }
    /* 結果を構成するErlangの項のポインタの配列を確保します */
    terms = (ERL_NIF_TERM *) enif_alloc(len * sizeof(ERL_NIF_TERM *));
    if (NULL == terms) {
        /* 領域確保ができなかった場合はbadargを返して終了します */
        return enif_make_badarg(env);
    }
    /* 乱数をまとめて逐次計算し、Erlangの項のリストを作ります */
    for (i = 0LL; i < (uint64_t)len; i++) {
        /* 現在の内部状態から乱数の値と次の内部状態を得ます */
        new = next((uint64_t)state);
        /* enif_make_uint64()でErlangの項を作り、そのアドレスをポインタの配列に入れます */
        terms[i] = enif_make_uint64(env, new.x);
        /* 逐次計算のたびに乱数の内部状態を更新します */
        state = (ErlNifUInt64)new.state;
    }
    /* 計算したErlangの項から結果として返すリストを作ります */
    list = enif_make_list_from_array(env, terms, len);
    /* この時点でenif_allocで確保した領域は不要になるので解放します */
    enif_free(terms);
    /* 結果のリストと内部状態を要素とするErlangのタプルを返します */
    /* enig_make_tuple2()は要素2つのタプルを返します */
    return enif_make_tuple2(
             env,
             list,
             enif_make_uint64(env, state));
}
/* Xorshift64*アルゴリズムのCのコードです */
static inline exs64_retval next(uint64_t s) {
    exs64_retval new;
    /* 内部状態と結果の乱数を両方returnの際に返しています */
    s ^= s >> 12;
    s ^= s << 25;
    s ^= s >> 27;
    new.x = s * 2685821657736338717LL;
    new.state = s;
    return new;
}
```

・最初に構造体ErlNifFuncで、モジュール内関数名とアリティの組に対応するC言語の関数を定義している。またマクロERL_NIF_INIT()で、NIFの初期設定を行う

・構造体ERL_NIF_TERMは、Erlangの項すべてに対応する型。NIFのC言語による関数はこの型を返す必要がある

・NIFとして呼ばれる関数は、実行環境として構造体ErlNifEnvへのポインタ、引数の数argc、そしてErlangの項として構造体ERL_NIF_TERMの配列であるargvを引数として持つ

・引数の内容に問題がある場合は、enif_make_badarg(env)という関数を返すことで、エラーコードのアトムbadargを返すようにしている

・C言語ではデータを書き込む実体を自動的に確保してくれるわけではない。確保するためには、あらかじめenif_alloc()を使ってメモリ領域を確保する

・ErlangのリストはC言語でもERL_NIF_TERMのポインタの配列として表現する。これはErlangのリストの定義と符合する

・Erlangの項とC言語のデータとは相互変換が必要で、かつErlangの項は動的に作られる。よって項の実体を参照するときはERL_NIF_TERMのポインタを使って操作する必要がある。これらの処理は“enif_make_”で始まる一連の関数として定義されている

・erl_nifライブラリでの64ビット符号なし整数はErlNifUInt64という型で示されるが、C言語の関数の中ではuint64_tと同義

NIFの効果と注意点

このコードを実際にコンパイルし、1,000個の乱数列を10万回生成するという作業を、NIFを使わない場合とNIFを使った場合で比較してみました[注3]。手もとのMac miniでOS X 10.11.1とErlang/OTP 18.1.5の組み合わせ(詳細は連載第10回を参照)で実行したところ、NIFなしで約38秒、一方NIFありでは1.5秒と、約25倍の高速化を図ることができました。BEAMにとって負荷の大きな作業をC言語で高速化するという形で効果が得られています。OTPでもOpenSSLによる暗号化ライブラリを提供するcryptoモジュールは、NIFとして作られており、計算速度の高速化が図られています。

しかし、NIFの利用は良いことばかりではありません。NIFはBEAMを直接C言語で拡張しているため、NIFの中でメモリリークや共有資源の誤操作[注4]が発生した場合、BEAMの異常動作が発生し得ます。この理由だけのために、ErlangではNIFを使わないという開発者もいます。品質の低いライブラリを安易にリンクされて全体の信頼性が下がってしまったという結果を避けるためには、NIFを極力避けるというのはやむを得ない判断であろうと筆者も思います。

また、NIFの動作中はBEAMのスケジューラがNIF内部での実行時間を正確に把握できないという問題が起こります。NIFの関数を実行している間は、BEAMはいわばNIF実行に集中しており、スケジューラの制御下にはないからです[注5](**図2**)。

このスケジューラに与える影響を最小限に抑えるためには、NIFの関数が消費している実行時間を可能な限り短くする必要があります。

注3) 使用したコードはGitHubにて公開しています(本文最後のソースコードの欄を参照)。

注4) Erlangのバイナリは、大きさは64バイトを越えると「参照カウントバイナリ」(reference-counted binaries)とされ、単一代入原則(連載第2回を参照)の意味を失わない範囲で高速化とコピーを避けるための実体の共有が行われます(http://erlang.org/doc/efficiency_guide/binaryhandling.html#id67212)。この参照カウントバイナリの実体をコピーを取らずに直接C言語で操作してしまった場合、予期せぬ参照元に影響が及び、修正の困難なバグの原因となり得ます。

注5) 複数のCPUコアが一般的になった現在、NIFによるコア独占の影響はそれほど重大ではないともいえます。しかし、シングルコアのCPUで並行動作を行う際は、単独のコアが一瞬とはいえNIFの動作で(OSによる割り込みやスケジューリングによる操作を除けば)目一杯使われてしまうため、軽視できない影響が発生する可能性があります。

今回紹介した乱数の例で、1,000個の乱数列を作る作業の繰り返しとしたのは、NIF内での実行時間を高々数十マイクロ秒に抑えることでスケジューラへの悪影響を防ぐという意図もあります。実際に汎用のNIFライブラリを作る際は、各関数が実行に使う時間をfprof[10]プロファイリングツールで関数ごとに詳細に計測して見極める必要があります[11]。

まとめ

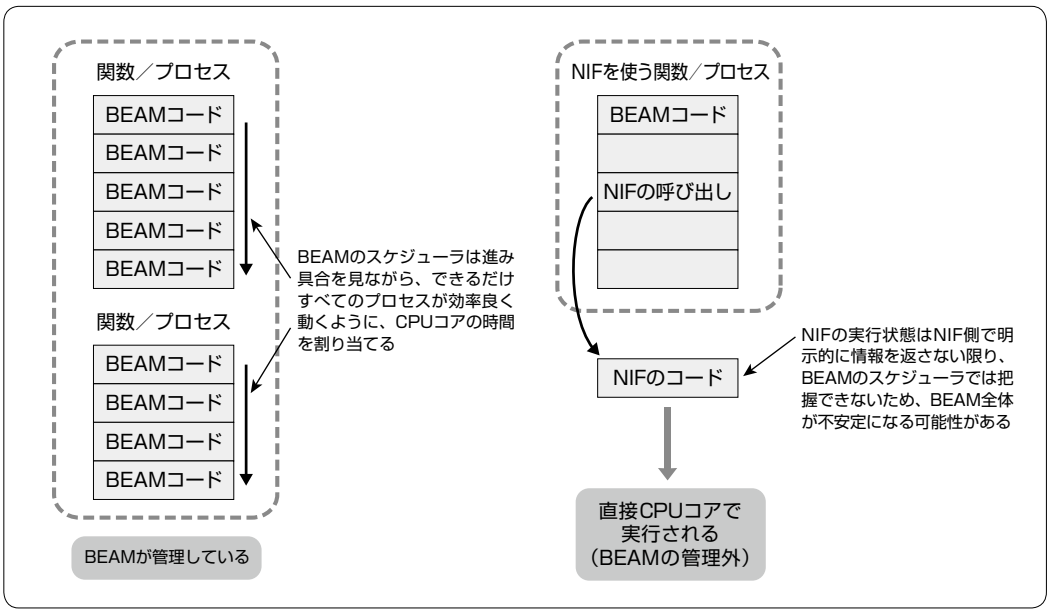
今回はErlang/OTPで外部プログラムやライブラリを利用する方法の一例として、NIFについて紹介しました。

次回はErlangによるWebサーバの構築と運用、そしてプログラミングについて紹介します。

ソースコードとサポートページ

今回の例に使用したNIFによる乱数の実装はGitHubのリポジトリで公開しています(https://github.com/jj1bdx/exs64m/)。また、連載の記事で紹介したソースコードなどは、GitHubのリポジトリに置いています(https://github.com/jj1bdx/sd-erlang-public/)。どうぞご活用ください。SD

▼図2 NIFとBEAMスケジューラの問題



参考文献

[1] http://erlang.org/pipermail/erlang-questions/2015-November/086897.html
[2] http://www.erlang.org/doc/tutorial/c_port.html
[3] http://www.erlang.org/doc/tutorial/c_portdriver.html
[4] http://www.erlang.org/doc/tutorial/nif.html
[5] http://www.erlang.org/doc/tutorial/cnode.html
[6] http://www.erlang.org/doc/tutorial/users_guide.html
[7] http://www.erlang.org/doc/man/erl_nif.html
[8] http://xorshift.di.unimi.it/xorshift64star.c
[9] http://erlang.org/doc/efficiency_guide/advanced.html#id68923
[10] http://erlang.org/doc/man/fprof.html
[11] Kenji Rikitake. 2011. SFMT pseudo random number generator for Erlang. In Proceedings of the 10th ACM SIGPLAN workshop on Erlang(Erlang '11). ACM, New York, NY, USA, 78-83. DOI=http://dx.doi.org/10.1145/2034654.2034669

書いて覚える Swift 入門

第11回　Swiftのオープンソース化

Writer 小飼 弾（こがい だん）　twitter @dankogai

swift.isOpenSource == true

今回はProtocol Oriented Programmingを紹介する予定でしたが、ここで緊急のお知らせです。と言ってもこれが記事に反映されるのは1ヵ月以上先ではあるのですが、それを考慮しても予定を変更するだけの価値があるでしょう。Swiftにとってそれ自体のリリースの次に重要なニュースなのですから。

2015年12月3日（日本時間では翌4日）、Swiftはオープンソースとして公開されました。

> *[Swift is Open Source[注1]]*
> - *[Swift.org] – a site dedicated to the open source Swift community*
> - *Public source code repositories at [github.com/apple]*
> - *A new Swift package manager project for easily sharing and building code*
> - *A Swift-native core libraries project with higher-level functionality above the standard library*
> - *Platform support for all Apple platforms as well as Linux*

抄訳すると、次のとおりです。

- オープンソースSwift専用サイト、[Swift.org]の開設（**図1**）
- GitHubにおけるソースコード公開（**図2**）
- Swiftパッケージマネージャープロジェクト開始
- Swiftネイティブ標準ライブラリ以上の高機能ライブラリプロジェクト開始
- すべてのAppleデバイスに加え、Linuxのサポート

クリスマスを待たずして、公約は果たされたわけです。

Swift on Linux: Getting Started

オープンソースとなった意義はこの後じっくり吟味するとして、まずは実際に試してみましょう。もちろんオープンソースだけあって、**図2**のWebページの解説に従ってソースからビルドしても良いのですが、引数なしのデフォルトの`utils/build-script`をそのまま実行すると、16GBのメモリ、64GB程度の空き容量が必要でした。仮想マシンだとちょっと荷が重い。`-R`をつけてデバッギングシンボルなしのリリースビルドだとそこまでリソースは食わないのですが、幸いにしてビルド済みのbinary snapshotをAppleが用意してくれているので今回はそれを利用することにします。

注1） Swift if Open Source（https://developer.apple.com/swift/blog/?id=34）

用意するもの

64-bit版のUbuntu 14.04 LTSまたはUbuntu 15.10

いずれはもっと多くのプラットフォームでサポートさせるはずですが、執筆現在、Mac以外のプラットフォームで正式サポートされているのはLinuxそれもUbuntuだけです。とはいえオープンソースなプラットフォームとしては最も普及しているものでもあり、導入の敷居は低いでしょう。GUIは含まれていないのでデスクトップ版ではなくサーバ版でもかまいませんし、仮想マシンでもかまいません。筆者はVMware FusionでRAM 2GB、仮想ディスク16GBの仮想マシンで動かしています。

clangのアップデート（14.04 LTSのみ）

Ubuntu 15.10では不要のようです。

```
$ sudo apt-get install clang-3.6
$ sudo update-alternatives --install /usr/bin/clang clang /usr/bin/clang-3.6 100
$ sudo update-alternatives --install /usr/bin/clang++ clang++ /usr/bin/clang++-3.6 100
```

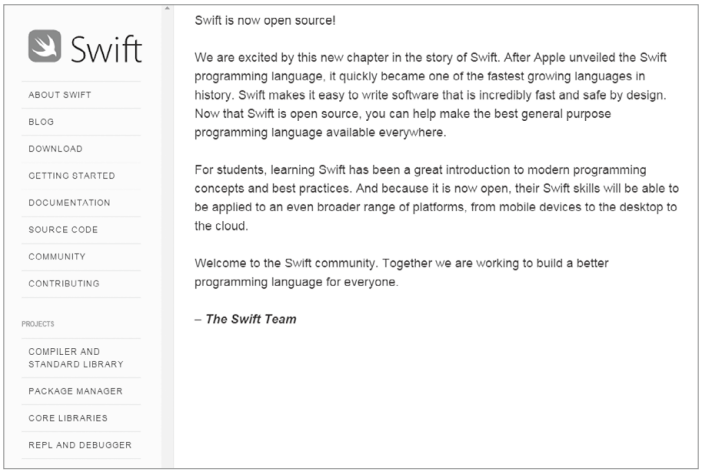
ダウンロード

あとは、**図2**のWebページからLatest Development Snapshotをダウンロードし、適当なところに解凍すれば準備完了です。ここでは配布物全体を~/swiftに置いています。全部で90MB弱。clangなどが含まれていないとはいえ、意外とコンパクトです。

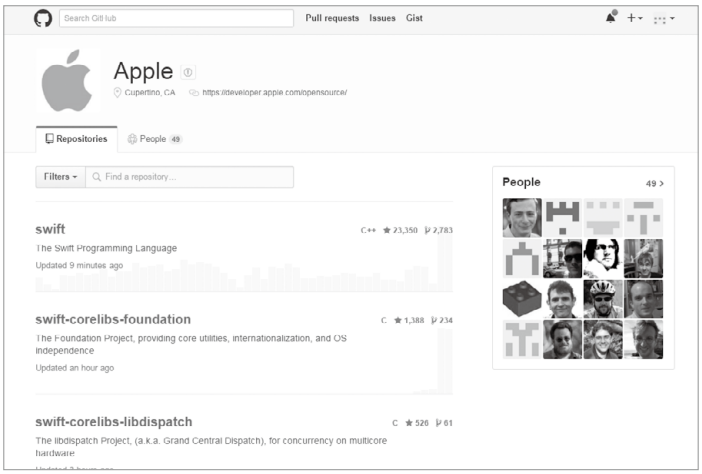
```
$ wget https://swift.org/builds/ubuntu1404/swift-2.2-SNAPSHOT-2015-12-01-b/swift-2.2-SNAPSHOT-2015-12-01-b-ubuntu14.04.tar.gz
$ tar zxvf swift-2.2-SNAPSHOT-2015-12-01-b-ubuntu14.04.tar.gz
$ mv swift-2.2-SNAPSHOT-2015-12-01-b-ubuntu14.04 ~/swift
```

なお、Tarballは執筆現在のものであり、本稿が読者の皆さんに届くころには変わっている可能性があるのでご注意を。

▼図1　Swift.org（https://swift.org/）



▼図2　github.com/apple（https://github.com/apple）



REPL

それでは、早速ターミナルから、

```
$ ~/swift/usr/bin/swift
```

と叩けば、REPLが起動します。フルパスが面倒なら、

```
$ export PATH=$HOME/swift/usr/bin:$PATH
```

などで~/swift/usr/binをパスに追加しておけば、swiftだけでOKです。

Rubyにおけるirbや引数なしのpythonを実行したのと同様に、対話的にSwiftを使うことができます(**図3**)。

Tips

このREPL、irbやインタラクティブモードのpythonと比べると、少しモダンになっています。

関数／メソッド補完

そのひとつは、関数やメソッドを補完してくれることです。たとえば1.と打った後でTabを打つと……、

```
  1> 1.
Available completions:
  advancedBy(n: Distance) -> Int
  advancedBy(n: Int, limit: Int) -> Int
  bigEndian: Int
  byteSwapped: Int
  description: String
  distanceTo(other: Int) -> Distance
  hashValue: Int
  littleEndian: Int
  predecessor() -> Int
  stride(through: Int, by: Distance) ->
StrideThrough<Int>
  stride(to: Int, by: Distance) ->
StrideTo<Int>
  successor() -> Int
                              ※右上へ続く
```

```
※左下からの続き
  toIntMax() -> IntMax
  1> 1.successor()
$R0: Int = 2
```

...Intのインスタンスメソッドが表示されますし、さらにsuと打ってTabを打つと、.successor()まで補完してくれます。

ブロック編集のサポート

SwiftのREPLのヒストリーは、行単位ではなくブロック単位です。たとえば、

```
  1> (1...10).reduce(0) {
  2.     $0 + $1
  3. }
$R0: Int = 55
```

この状態で↑キーを押すと、3行に渡るこのブロックが丸ごと再表示されます。reduce(0)をreduce(1)に、$0 + $1を$0 * $1に編集して、最後に}の後ろまでカーソルを移動してからEnterキーを押すと……、

```
  4> (1...10).reduce(1) {
  5.     $0 * $1
  6. }
$R1: Int = 3628800
```

となります。ブロックの終了、つまり}以前にリターンした場合、そのまま行挿入もできます。

import Glibc

Swiftは強力な言語です。しかしPerlやPythonやRubyなどのスクリプト言語と異なり、生のSwiftは三角関数を1つサポートしていません。

```
  7> let pi = -2 * atan2(-1, 0)
repl.swift:7:15: error: use of unresolved
identifier 'atan2'
let pi = -2 * atan2(-1, 0)
```

新規playgroundで、iOSならimport UIKit、

▼図3　Linux上で動くSwift

OS Xならimport Cocoaという「呪文」が最初から入っているのは、そのためです。残念ながらLinux版のSwiftにはまだplaygroundはないのですが、Linuxでは何をインポートするのがそれに相当するのでしょうか？

import Glibcだそうです。連載第7回で筆者が予想したimport POSIXではなく。

```
  7> import Glibc
  8> let pi = -2 * atan2(-1, 0)
pi: Double = 3.1415926535897931
```

これを#ifと組み合わせると、クロスプラットフォームなSwiftコードが書けそうです。**リスト1**のコードは、Linuxと OS X双方でchmod +xしたうえでスクリプトとして実行可能で、swiftc pi.swiftでコンパイルしても動くことを確認しました。

あらためて、オープンソースであるということ

公約どおり、Swiftはオープンソースとなりました。ここで2つの疑問が湧いてきます。

・なぜ、オープンソースにしたのか？
・なぜ、はじめからオープンソースにしなかったのか？

この2つの疑問に対し、連載第7回時点の筆者はこう答えています。

▼リスト1　pi.swift

```
#!/usr/bin/env swift
#if     os(OSX)
import Cocoa
#elseif os(iOS)
import UIKit
#elseif os(Linux)
import Glibc
#endif

let π = -2 * atan2(-1.0, 0.0)
print("π = \(π)")
```

> ***食えなきゃ誰も食ってくれない。***
> ***オープンでなければ、誰も食いつづけてくれない。***

WWDC2014におけるデビューからわずか1年半、iOSとOS Xのリリースサイクルわずか1回分で、Swiftはi OSアプリ開発における第一言語となっています。オープンソースとなる前から、「食える言語」という地位は、すでに確立したわけです。しかし、Xcode以外の実装を持たなかったSwiftは、真の意味での汎用言語ではありませんでした。「どんなプログラムでも書ける」では汎用言語としては十分ではありません。「どんなプラットフォーム上でも」も成立して、はじめて汎用と呼べるのです。オープンソース化は、そのための最短距離でもあります。Apple自身はLinux、それもUbuntuしか現時点でサポートしていませんが、リリースから1日も経たずして、すでにGitHubでは他のプラットフォームへの移植が雨後の筍のように始まっています。「言語の普及競争において、Swiftほど高いオッズを持つ言語が見当たりません」と連載第7回時点で筆者は言いましたが、オープンソース化の公約を果たした今、オッズはさらに高まったのは確かでしょう。

Appleプラットフォームとは無縁だった読者も、今後本連載はスルーできなくなったのではないでしょうか。SD

Sphinxで始めるドキュメント作成術

打田 智子 Uchida Tomoko Twitter @moco_beta ㈱ロンウイット URL http://www.rondhuit.com/

第11回 HTMLドキュメントを検索しよう

今回のテーマ

今回のテーマは「全文検索」です。ドキュメントが大きくなり、複数ページに渡るようになると、目的のページを探すのに時間がかかるようになりがちです。その場合に、検索ができると便利です。SphinxがHTMLおよびJavaScriptには、転置インデックス[注1]を用いた全文検索機能が付いています。一般的なWebブラウザに付属するページ内検索とは異なり、ドキュメントが複数ページに分かれていても横断で検索ができ、また、多言語にも対応しています。

今回は、デフォルトで使える検索機能および、日本語ドキュメントの検索精度を上げるための設定を紹介します。なお本稿では以下、(とくに問題がない限り)全文検索機能のことを、単に検索機能と記載します。

検索機能の紹介

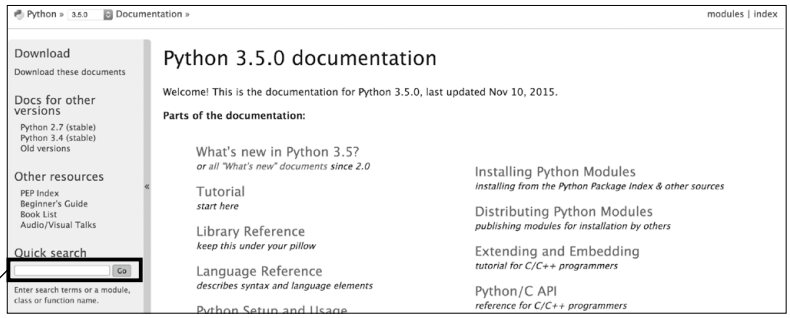
Pythonの公式ドキュメンテーション[注2]やSphinx-Users.jpのサイト[注3]には、検索フォーム(**図1、2**)が付いていて、探したい関数や機能の名称などを入力して検索ができます。**図3**に、Sphinx-Users.jpで「テーブル」という文字列で検索したときの検索結果画面を示します。

なお、検索フォームが付いているかどうかはテーマによります。Sphinx 1.3.3現在の組み込みのHTMLテーマ[注4]のうち、検索フォームが付いているのは、alabaster、sphinx_rtd_theme、classic、sphinxdoc、agogo、traditional、nature、

注1) 全文検索で広く使われるデータ構造の一種で、ドキュメント中に出現する単語の一覧と、各単語がどこに出現するかの位置情報を保持したものです。grepに代表される逐次検索方式(ドキュメントの先頭から文字列探索し、検索キーワードとのマッチングを行う)と比較すると、事前のインデックス構築が必要となる代わりに、ドキュメントが大きくなった場合でも高速に検索できます。本の巻末にある索引をイメージするとわかりやすいでしょう。

注2) https://docs.python.org/
注3) http://sphinx-users.jp/
注4) http://docs.sphinx-users.jp/theming.html

▼図1 Python公式ドキュメンテーションの検索フォーム(画面左下)



pyramid、bizstyleです。

AND検索、NOT検索

検索フォームに、スペースで区切って複数の単語を入れるとAND検索になります。たとえば「テーブル 画像」と入力して検索すると、「テーブル」と「画像」の両方を含むドキュメントがヒットします。

単語の冒頭に"-"を付けると、NOT検索(否定)になります。たとえば、「テーブル -改行」と入力して検索すると、「テーブル」を含み「改行」を含まないドキュメントがヒットします。

日本語のドキュメントを検索しよう

サンプルプロジェクトの作成

それではさっそく、日本語で書いたドキュメントを検索してみましょう。例として、**リスト1、2、3**のようなドキュメントを含むプロジェクトを作成します。

また、conf.pyのlanguage設定(sphinx-quickstart時に指定する"Project Language")で`'en'`など、`'ja'`以外のものを指定しているけれど、日本語検索用の転置インデックスを生成したいという場合には、conf.pyの設定を次のように書き換えて保存します。

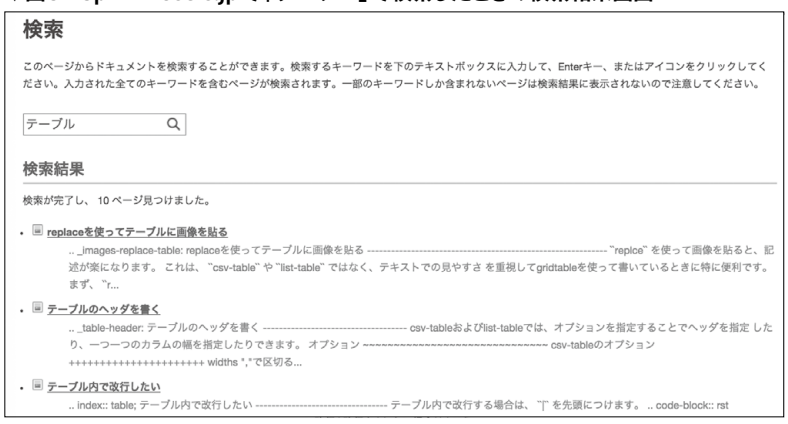
日本語の転置インデックスを生成する設定
変更前
```
#html_search_language = 'en'
```
変更後
```
html_search_language = 'ja'
```

languageが`'ja'`の場合は、デフォルトでhtml_search_languageも`'ja'`が選択されるため、この設定変更は不要です。

`make html`を実行したときに、**図4**のようなログが出力されていれば日本語用の転置インデックスが生成されています。

▼図2 Sphinx-Users.jpトップページの検索フォーム(画面右上)

▼図3 Sphinx-Users.jpで「テーブル」で検索したときの検索結果画面



▼リスト1 mimei.rst

```
小川未明
========

* 赤い蝋燭と人魚
* 時計のない村
* 雪の上のおじいさん
* 手袋を買いに
```

▼リスト2 kyouka.rst

```
泉　鏡花
========

* 金時計
* 外科室
* 星あかり
```

▼リスト3 nankichi.rst

```
新見南吉
========

* おじいさんのランプ
* うた時計
```

日本語検索の確認

HTMLを作成し、「人魚」「時計」「手袋」といったキーワードで検索をしてみましょう。そのキーワードを含むドキュメントがヒットするはずです。**図5**に、検索結果の例を示します。

検索結果では、検索キーワードを含む周辺の文字列がスニペット表示され、さらにキーワードがハイライトされます。スニペット&ハイライト表示は、ドキュメントの中でどこにキーワードが出現するかがぱっと見てわかるため、長いドキュメントを検索するときにとても便利な機能です。

転置インデックスのしくみと注意点

転置インデックスでは、一般に、インデックス構築時(make html実行時)にあらかじめドキュメントを単語に分割し、一覧にして保存しておきます。検索実行時には、その単語一覧を使ってマッチングを行うしくみです。日本語の場合、単語分割には、分かち書きや形態素解析[注5]を行う専用ソフトが使用されます。生成された単語リストに含まれないキーワードで検索された場合は、たとえ検索キーワード文字列が検索対象のドキュメントに含まれていたとしても、ゼロ件ヒットとなってしまいます。

たとえば、Sphinx本体に含まれる日本語の単語分割モジュール(JavaScriptで書かれたコンパクトな分かち書きソフト「TinySegmenter」[注6]をPythonに移植したもの)では、「おじいさん」という単語は生成されず「おじい」と「さん」に分かれます。

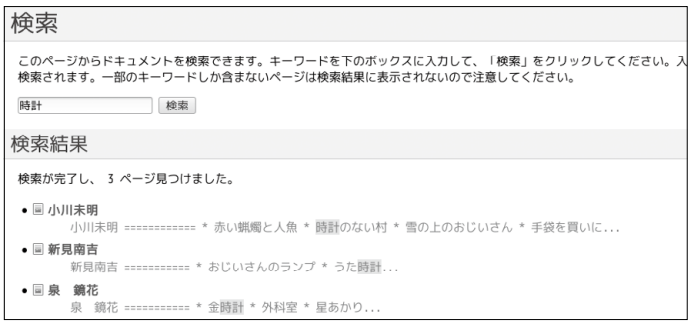
このため、本節のサンプルmimei.rst(**リスト1**)とnankichi.rst(**リスト3**)には「おじいさん」という文字列が含まれているにもかかわらず、**図6**のように「おじいさん」で検索してもヒットしないという、直感に反する結果になります(「おじい」と「さん」が分割されて別々の単語として保持されているため、「おじい　さん」のように、「おじい」と「さん」をスペースで区切ってAND検索を実行するとヒットします。しかし、通常のユースケースでは「おじいさん」という語でヒットするのが望ましいでしょう)。

このように、単語分割の精度によっては、意図するページが見つからないことがあります。次節では、インデックス作成時の単語分割に日本語形態素解析器「MeCab」を使うことで、この問題を改善する設定を紹介します。

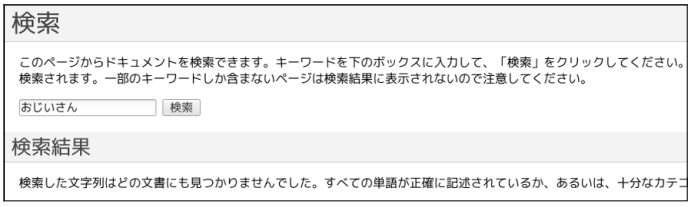
▼図4　転置インデックスが生成されたときに出力されるログ

```
$ make html
（略）
dumping search index in Japanese (code: ja) ... done
```

▼図5　サンプルデータを「時計」で検索した例



▼図6　「おじいさん」で検索してもヒットしない



1文字からなる単語の検索について

Sphinxの日本語検索では、単語分割したあとで、1文字のみからなる単語をインデックスに含めないように対象から外す実装になっています。そのため、「星」「猫」と

注5) 文章を意味のある言語の最小単位(単語)に区切り、さらに品詞などを判別すること。

注6) http://chasen.org/~taku/software/TinySegmenter/

いった、1文字からなる単語で検索しても、残念ながらヒットしません。おそらく、JavaScript中に全インデックスを保持しなければならないため、インデックスサイズを肥大化させないための配慮だと思われます。

MeCabを使ったインデックス作成

前節では、デフォルトの日本語検索では「おじいさん」という単語がヒットしない問題に触れました。これはおもに、TinySegmenter(および、そのPython移植)が辞書を使わずに単語の境界を推定していることが原因です。辞書を使わないため、モジュールが軽量になるという利点があります。新聞記事のような文ならうまくいきやすいのですが、ひらがな混じりの文などには弱いようです。

Sphinxは、インデックス作成に使用される単語分割モジュールを変更できます。たとえば、日本語検索ではTinySegmenterがデフォルトで使用されますが、これをC/C++で書かれた日本語形態素解析器「MeCab」[注7]を使用するように変更できます。MeCabは辞書(MeCab-IPADIC)を使って単語境界を推定するため、TinySegmenterよりも精度の良い分割となることが多いです。

MeCab、MeCab-Pythonのインストール

MeCabはCライブラリのため、別途インストールが必要です。公式サイトの案内[注8]にしたがい、インストールしておいてください。インストールにはgccとiconvが必要です。また、Pythonバインディングがインストールされていると、Sphinxはバインディング経由でMeCabを使います。2015年12月現在、MeCabおよびPython 2系のバインディングmecab-pythonは、Google Drive[注9]上で配布されています。Python 3系の対応状況については次節を参照してください。

図7、8のコマンドが実行できれば、MeCab

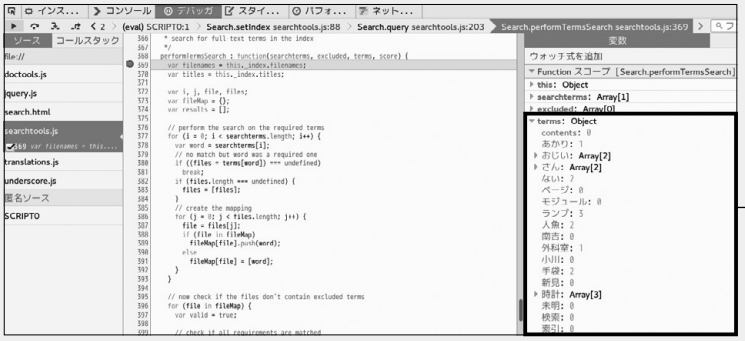
注7) http://taku910.github.io/mecab/
注8) http://taku910.github.io/mecab/#install
注9) https://goo.gl/42LpXS

COLUMN

転置インデックスの単語リストを確認する

Sphinxの転置インデックスは、JavaScriptのオブジェクトとして格納されていて、ブラウザのデバッガを使うとインデックスの一覧を見ることができます。たとえば、Firefoxの「開発ツール」→「デバッガ」で、searchtools.jsのperformTermsSearchメソッド(370行目付近)にブレークポイントを入れて検索を実行すると、メソッドのterms引数に単語の一覧が格納されていることが確認できます(**図A**の右欄参照)。

▼図A　リスト1、2、3から生成された転置インデックス(単語リスト)



およびmecab-pythonが正しくインストールされています。

Python 3系の対応

MeCab公式で配布されているバインディングはPython 2系にのみ対応しています。Python 3系でMeCabを使う際は、有志の方が作成、PyPIで配布しているmecab-python3[注10]などをご利用ください。

また、Sphinx 1.3.1以前は、SphinxからMeCabバインディングを呼び出すモジュールがPython 3系で動作しない不具合がありました。不具合を修正するパッチを筆者が投稿[注11]し、1.3.2で取り込まれたため、1.3.2以降のSphinxをご利用ください。

MeCabを使うconf.pyの設定

MeCab(および、オプションでmecab-python)をインストールしたら、conf.pyのhtml_search_optionsを**リスト4**のように書き換えます。

conf.pyを書き換えて保存し、`make html`を実行すると、MeCabを使って作成した転置インデックスを含むHTMLファイルができあがります。確認のため、先ほどはヒットしなかった「おじいさん」という検索キーワードで検索してみましょう。2件のドキュメントがヒットするはずです(**図9**)。

このように、転置インデックスの作成にMeCabを使用することで、日本語検索の使い勝手が向上します。分量の多い日本語ドキュメントを書く際は、ぜひ利用を検討してみてください。

日本語以外のドキュメントの検索

Sphinxは日本語以外にも、英語、フランス語、ドイツ語、ロシア語、といった多くの言語の検索をサポートしています。対応言語についてはリファレンス[注12]のhtml_search_languageの説明を参照してください。

次回予告

今回は、大きなプロジェクトにおいて、目的のドキュメントをすばやく見つけるための全文検索機能について紹介しました。

次回は、Sphinxを活用した本や雑誌記事の執筆について取り上げます。SD

注10) https://pypi.python.org/pypi/mecab-python3
注11) https://github.com/sphinx-doc/sphinx/pull/2127
注12) http://docs.sphinx-users.jp/config.html

▼図7　MeCabのインストール確認

```
$ echo 本日は晴天なり | mecab
本日    名詞,副詞可能,*,*,*,*,本日,ホンジツ,ホンジツ
は      助詞,係助詞,*,*,*,*,は,ハ,ワ
晴天    名詞,一般,*,*,*,*,晴天,セイテン,セイテン
なり    助動詞,*,*,*,文語・ナリ,基本形,なり,ナリ,ナリ
EOS
```

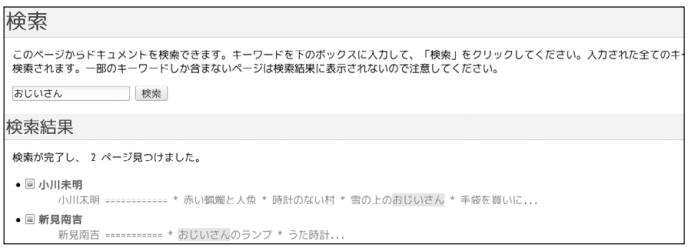
▼図8　mecab-pythonのインストール確認

```
$ python
>>> import MeCab
>>> m = MeCab.Tagger("-Owakati")
>>> print(m.parse("本日は晴天なり"))
本日 は 晴天 なり
```

▼リスト4　MeCab、mecab-pythonを使う場合のconf.pyの設定

```
変更前
  #html_search_options = {'type': 'default'}
変更後
  html_search_options = {'type': 'mecab'}
```

▼図9　「おじいさん」で検索すると2件ヒットする



COLUMN

ctypes経由でMeCabを呼び出す

Python 2系に限定されますが、バインディングを経由せず、ctypes[注A]を使って直接MeCabを呼び出すこともできます。何らかの理由でバインディングをインストールしない場合は、**リストA**のようにMeCabのライブラリパスを併せて指定してください。

注A）Cで書かれたライブラリを、Pythonから利用するためのライブラリ。

▼リストA　ctypes経由で使う場合のライブラリパス設定例

```
html_search_options = {'type': 'mecab', 'lib': '/usr/lib64/libmecab.so'}
```

COLUMN

SphinxCon JP 2015／Sphinx-1.3.3 リリース

Author 清水川 貴之

■SphinxCon JP 2015

2015年11月24日に、Sphinxのカンファレンス、SphinxCon JP 2015[注B]を開催しました(**写真A**)。2012年から毎年開催しているSphinxConですが、今年のSphinxCon JPのテーマは「色々やってみよう」です。

当日は平日夜の開催にもかかわらず、約50名が集まり非常に盛況でした。基調講演は「ドキュメントシステムはこれを使え2015年版」というタイトルで鹿野桂一郎さんにお話しいただきました。鹿野さんはオーム社在籍中に書籍制作の自動化に取り組まれてきた編集者です。今回のSphinxConは発表中心で、基調講演のほかに6名の発表者からいろいろなテーマで発表していただきました。

- 「ドキュメントシステムはこれを使え2015年版」鹿野 桂一郎さん
- 「Sphinxで電子書籍を書こうと色々やってみた」@momijiameさん
- 「SphinxでMarkdownを使う(仮)」@r_rudiさん
- 「Sphinxで手軽に作るドキュメント」@usaturnさん
- 「Sphinxで社内勉強会(Git)の資料を作ってみた」@takuan_oshoさん
- 「APIドキュメントのはなし」@tk0miyaさん
- 「All docs lead to Sphinx」洪 民憙(ホンミンニ)さん

ホンミンニさんは、このイベントのために韓国から来日してくれました。Sphinxおよびドキュメンテーションツールの盛り上がりを感じられるイベントだったと思います。

▼写真A　SphinxCon JP 2015の様子

すべての発表スライド、当日の様子などをサイト[注B]に掲載していますので、ぜひご参照ください。

■Sphinx 1.3.3 リリース

2015年11月30日と12月2日に、Sphinxの新しいバージョン1.3.2と1.3.3をリリースしました[注C]。1.3.3は1.3系のマイナーバージョンアップで、1.3.1から45個の不具合を修正しています。動作について気になる点がありましたら、Sphinx-users.jpメーリングリスト[注D]、またはSphinxのGitHub[注E]までご連絡ください。

開発は今後、1.4系に注力していきます。どのような機能が1.4に実装されていくのか興味のある方は、GitHubをご確認ください。

注B）http://sphinx-users.jp/event/20151124_sphinxconjp/index.html

注C）https://pypi.python.org/pypi/Sphinx/1.3.3

注D）http://sphinx-users.jp/howtojoin.html

注E）https://github.com/sphinx-doc/sphinx

Red Hat Enterprise Linuxを極める・使いこなすヒント

.SPECs ドット・スペックス

第18回 RHEL 7.2リリース

米国時間・2015年11月19日にRed Hat Enterprise Linux（RHEL）7.2がリリースされました。今回はRHEL 7.2でとくに注目すべき点について紹介します。

Author レッドハット(株)サービス事業統括本部
プラットフォームソリューション統括部ソリューションアーキテクト部長 藤田 稜（ふじた りょう）
Twitter @rioriost

コンテナ関連の機能強化

コンテナ関連の機能強化がRHEL 7.2では引き続き行われており、docker-1.8.2-8.el7、kubernetes-1.0.3-0.2.gitb9a88a7.el7が採用されています[注1]。またRed Hat Container Development Kit 2の提供が開始され、コンテナベースのアプリケーションの開発を簡易化するツールが利用可能になりました。

セキュリティの強化

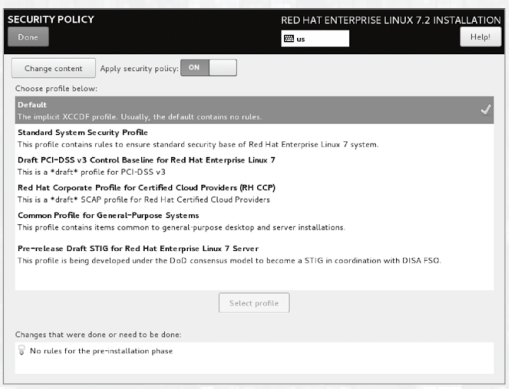
セキュリティ面では、各種セキュリティ規格やガイドラインに沿ったシステムの設定などが非常に煩雑になるなか、セキュリティ設定共通化手順(SCAP：Security Content Automation Protocol)を策定することで、自動化と標準化を目指そうという流れに対応し、RHEL 7.2ではSCAPのオープンな実装であるOpenSCAPの設定がインストーラのAnacondaで可能になりました(**図1**、**図2**)。

Relax and Recoverの追加

システムのバックアップツールとして、Relax-and-Recover(rear-1.17.2-1.el7)が追加されました。NFSなどを経由してISOなどの各種の起動可能なイメージファイルにバックアップし、そのメディアから起動することでシステムをリカバリしたり、ディザスタリカバリ環境を構築できます。使いやすいことを目標に開発

▼図1 Anacondaに追加された[SECURITY POLICY]の設定

▼図2 SECURITY POLICYの選択画面



注1）subscription-manager repos --enable=rhel-7-server-extras-rpms でリポジトリを有効にすることで利用可能。

されたこともあり、rearのインストール後にman rearやrear dumpコマンドを実行してみると、初期設定を少しだけ変更すればバックアップが適切に取得されることがわかると思いますので、ぜひ試してみてください。従来、rearはEPEL(Extra Packages for Enterprise Linux)で提供されており[注2]、RHELのリポジトリに「昇格」した例[注3]と言えます。

AMCが提供される初めてのRHEL 7

RHELには、Self Support(日本国内未提供)、Standard、Premiumの3つのサポートレベル[注4]があり、Premiumであれば24時間×週7日[注5]のサポートが提供されます。PremiumサポートではEUS(Extended Update Support)add-onが含まれるため、最長で1年半程度のマイナーリリースサポートを提供しています。しかし、社会インフラなどに用いるシステムではより長期に渡って特定のマイナーリリースがサポートされ安定して運用できることや、日本固有の事情として文字コード・Shift-JISのサポートが必要となることがあり、これらの要件を満たすべく提供されているのがAMC(Advanced Mission Critical)サポート[注6]です。RHEL 7.0および7.1で未提供であったAMCが初めて提供されるのが7.2ということになります。

TCP/IPスタックは3.18

RHEL 7で採用されているLinuxカーネルは3.10ですが、RHEL 7.2ではNFV(Network Function Virtualization)やSDN(Software Defined Network)におけるTCP/IPの性能向上のため、

▼表1 RHEL 7.1と7.2のカーネルの/net/以下のサブディレクトリを比較(上位10位)

	サブディレクトリ	行数
1	ipv4	4520
2	mac80211	3456
3	core	1416
4	ipv6	1233
5	wireless	1098
6	openvswitch	952
7	bluetooth	238
8	ceph	238
9	sunrpc	237
10	xfrm	211

TCP/IPスタックはLinuxカーネル3.18にリベースしています。実際にRHEL 7.1と7.2のカーネルのソースコードを比較すると[注7]、/net/ipv4/以下などが大きく書き換えられていることがわかります(**表1**)。

基本的なことですが、RHELのソースコードをダウンロードしパッケージを展開する手順について紹介しておきましょう。

まず、ソースリポジトリを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-source-rpms
```

次にyum-utilsをインストールします。

```
# yum -y install yum-utils
```

/tmpディレクトリにソースrpmパッケージをダウンロードします。

```
# cd /tmp
# yumdownloader --source kernel-3.10.0-229.el7 kernel-3.10.0-327.el7
```

作業用ディレクトリを作成し、展開します。

注2) http://dl.fedoraproject.org/pub/epel/7/x86_64/repoview/rear.html
注3) mikutterなどにも頑張ってほしいところ。
注4) https://access.redhat.com/ja/support/offerings/production/sla
注5) 24×7(Twenty four by Seven)は、うるう年であっても有効な表記。Red Hatは「365日」という表記を原則的に用いない。
注6) サポートの内容や初期レスポンスタイムなどの定義に関しては各OEMに委ねられているため、執筆時点で提供している富士通、日立、NEC、日本HPの各社に確認のこと。
注7) linux-3.10.0-229.el7/net/*とlinux-3.10.0-327.el7/net/*をdiffで比較して、7.1から書き換えられた行数をカウント。

```
# mkdir 7.1 7.2
# cd 7.1
# rpm2cpio ../kernel-3.10.0-229.el7.
src.rpm | cpio -id
# tar Jxvf linux-3.10.0-229.el7.tar.xz
# mv linux-3.10.0-229.el7 /tmp/
# cd ../7.2
# rpm2cpio ../kernel-3.10.0-327.el7.
src.rpm | cpio -id
# tar Jxvf linux-3.10.0-327.el7.tar.xz
# mv linux-3.10.0-327.el7 /tmp/
```

これで/tmpに比較したいソースコードが用意できたので、単純に/net/以下を比較するのであれば次のようにコマンドを実行してみましょう。

```
# cd /tmp
# diff -r linux-3.10.0-229.el7/net/
 linux-3.10.0-327.el7/net/
```

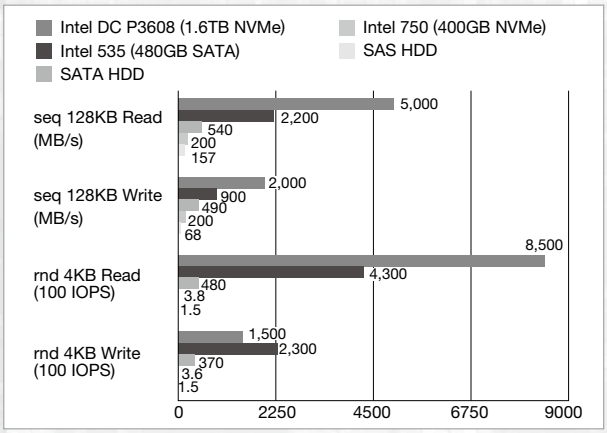
NVMe + blk-mq

NVMe(Non Volatile Memory Express)は、従来のAHCI(Advanced Host Controller Interface)がSATAインターフェイスの仕様であるのと同様に、PCIe(Peripheral Component Interconnect Express)に接続されたSSDインターフェースの仕様です。NVMeで接続できる物理的なインターフェースとして、PCIeスロット、およびU.2(旧名称：SFF-8639)コネクタがあります。また、SATA ExpressやM.2コネクタでもNVMeを論理的なインターフェースとして扱うことで接続できますが、NVMe SSDの性能が十分に発揮できないため、前者の物理的なインターフェースを利用することをお勧めします。

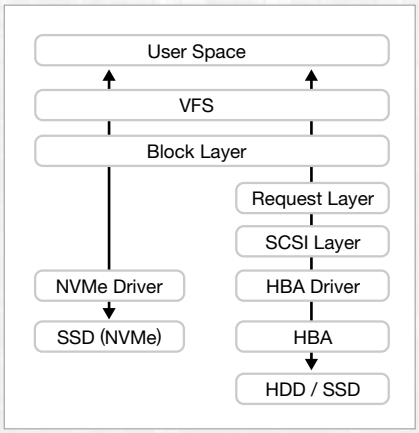
AHCIで1つしかなかったコマンドキューがNVMeでは65,536になっていることや、単一の割り込みしかないAHCIに対してNVMeでは2048のMSI-X[注8]が実装されている点など、低レイテンシおよび並列化による帯域の拡大を実現しています。結果として、**図3**のグラフに示すようにNVMe接続のSSDは、HDDはもちろんSATA接続のSSDと比較しても大きな性能改善を見せており、Gバイト単価が下げ止まっているHDDに対しハイペースで下がっているSSDがストレージの主役にとって代わる日は遠くなさそう[注9]です。

ハードウェアの改善に対しソフトウェアでも追随する動きが加速しています。Linuxではカーネル3.3にNVMeドライバがマージされたため、Linuxカーネル3.10を採用するRHEL 7では最初のマイナーリリース・7.0からNVMe SSDを利用可能[注10]です。NVMeとAHCIのLinuxにおける実装を比較すると(**図4**)、NVMeではSCSI

▼図3　HDDとSSDの性能比較



▼図4　NVMe(左)とAHCI(右)の実装の違い



注8) Message Signaled Interrupts、MSI-XはPCI 3.0規格から利用可能で、割り込みは最大2,048。
注9) COMPUTERWORLD "Consumer SSDs and hard drive prices are nearing parity"など(http://bit.ly/1RqPSeS)。
注10) BIOSやマザーボード上のコネクタなど、ハードウェアがNVMeに対応していることも必須条件。

レイヤーやリクエストレイヤーが存在しないためソフトウェアによるオーバーヘッドも少なくなっています。

また、この実装の違いにより長年慣れ親しんできた/dev/sdxというデバイスノードを目にする機会は減るでしょう。**図5**はIntel SSD 750を搭載したRHEL 7.2環境でコマンドを実行したものです。

▼図5　NVMe SSD環境でのデバイスの見え方

```
# ls /dev/sd*
ls: cannot access /dev/sd*: No such file or directory
# ls /dev/nvme*
/dev/nvme0  /dev/nvme0n1  /dev/nvme0n1p1  /dev/nvme0n1p2  /dev/nvme0n1p3
# lsblk
NAME              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
nvme0n1           259:0    0 372.6G  0 disk
|-nvme0n1p1       259:1    0   200M  0 part /boot/efi
|-nvme0n1p2       259:2    0   500M  0 part /boot
`-nvme0n1p3       259:3    0 371.9G  0 part
  |-rhel_www-root 253:0    0 356.2G  0 lvm  /
  `-rhel_www-swap 253:1    0  15.7G  0 lvm  [SWAP]
```

Linuxカーネルでは NVMe SSDに対する性能改善が継続的に行われており、RHEL 7.2ではblk-mqがnvmeドライバとともに利用できるようになりました[注11]。blk-mqはLinuxカーネルにおけるネットワークスタックのマルチキューイングの実装をブロックI/Oに応用したもので、マルチコアCPUかつNUMA環境でのブロックI/Oのスケーラビリティを大きく改善するものです(**図6**、**図7**)。

したがって、NVMe SSDを利用するのであれば、RHEL 7.2あるいはCentOS 7.2以降を利用することをお勧めします。

NVMe SSDの管理・運用について、RHELのマイナーリリース以外にも注意すべき点があります。従来であればsmartctlコマンド[注12]によってブロックデバイスのステータスを収集しトラブル回避に利用していたと思いますが、NVMe SSDは

▼図6　旧来のシングルキュー

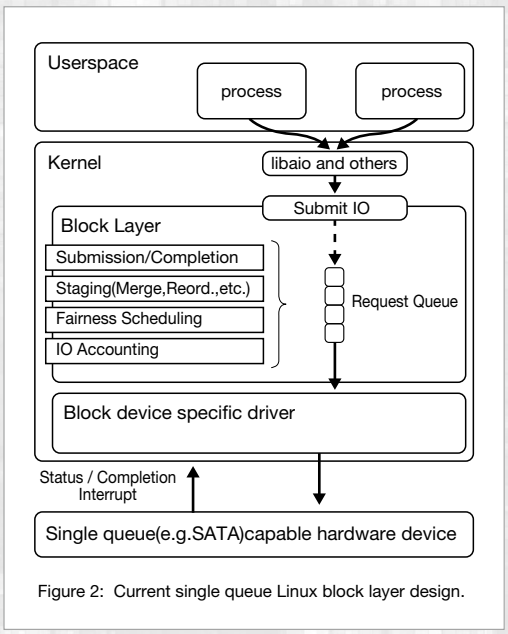


Figure 2: Current single queue Linux block layer design.

Matias Bjørling, Jens Axboe, David Nellans and Philippe Bonnet. Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems. page 2

▼図7　新しいキューのしくみ blk-mq

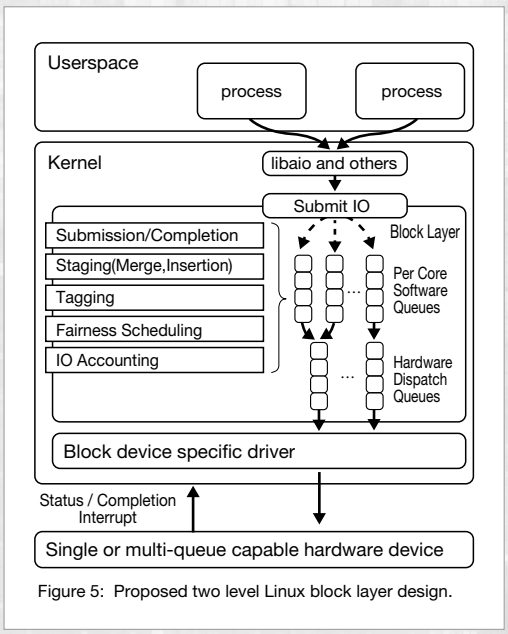


Figure 5: Proposed two level Linux block layer design.

Matias Bjørling, Jens Axboe, David Nellans and Philippe Bonnet. Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems. page 5

注11) blk-mkは、Linuxカーネル3.13でマージされた。
注12) S.M.A.R.T. : Self-Monitoring, Analysis and Reporting Technology System

▼図8　S.M.A.R.T.ではエラーロギングなどが利用できない

```
# smartctl -d scsi --all /dev/nvme0n1p1
smartctl 6.2 2013-07-26 r3841 [x86_64-linux-3.10.0-327.el7.x86_64] (local build)
Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Vendor:               NVMe
Product:              INTEL SSDPEDMW40
Revision:             0135
User Capacity:        400,088,457,216 bytes [400 GB]
Logical block size:   512 bytes
Rotation Rate:        Solid State Device
Logical Unit id:      8086INTEL SSDPEDMW400G4                    1000CVCQ514600SX400AGN
Serial number:        CVCQ514600SX400AGN
Device type:          disk
Local Time is:        Mon Dec  7 14:39:33 2015 JST
SMART support is:     Unavailable - device lacks SMART capability.

=== START OF READ SMART DATA SECTION ===

Current Drive Temperature:     35 C
Drive Trip Temperature:        85 C

Error Counter logging not supported

[GLTSD (Global Logging Target Save Disable) set. Enable Save with '-S on']
Device does not support Self Test logging
```

S.M.A.R.T.を完全にサポートしていません(**図8**)。

写真1のようなIntel製のSSDであれば、isdct(Intel SSD Data Center Tool)[注13]を利用して各種ステータスを収集することになります(**図9**)。32bit/64bitのRPMパッケージも用意されているので、RHEL 7/CentOS 7でのインストールも簡単に行えます。isdctで取得できるステータスの意味や読み方についてはisdctのドキュメントを参照してください。

高いパフォーマンスを発揮するSSDにも弱点がないわけではなく、SSDは上書きができません。このため実データを書き込もうとすると「消去」と「実データの書き込み」という2回の書き込みによって性能が劣化するという問題があります。実データの書き込み前にSSDのセルを消去させる余地をSSDに与えるのがTRIMコマンドであり、SSDの黎明期には「TRIMコマンドのサポートの有無」がよく取り沙汰されました。TRIMはATAのコマンドですが、NVMeでは同様の機能がData-set ManagementとしてLinuxカーネル3.9で実装され、SATA/SAS接続のSSDと同様にマウント時のdiscardオプションや、fstrimコマン

▼図9　isdctコマンドの実行例

```
# isdct dump -o text -intelssd 0↵
DataType=Nvmelog LogId=202
- CVCQ514600SX400AGN.AB -
ID: AB
Description: Program Fail Count
Normalized: 100
Raw: 0x0

- CVCQ514600SX400AGN.AC -
ID: AC
Description: Erase Fail Count
Normalized: 100
Raw: 0x0

- CVCQ514600SX400AGN.AD -
ID: AD
Description: Wear Leveling Count
Normalized: 100
Raw: 0x8000A0006

<< snip >>

- CVCQ514600SX400AGN.F3 -
ID: F3
Description: PLI Lock Loss Count
Normalized: 100
Raw: 0x0
```

注13) インテル Solid-State Drive Data Center Tool http://intel.ly/1IyQcq8

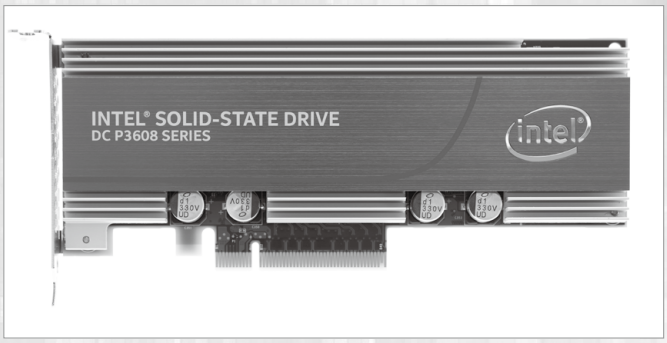
ドを利用可能です。図10のスクリプトを定期的に実行すれば簡単に性能を維持できます。

まとめ

本稿では誌面の都合で触れなかった機能強化や改善点がRHEL 7.2には数多く含まれます。また日本国内ではメジャーリリースからしばらくの間は様子見で、“0.2”ぐらいのマイナーリリースから本格的に導入する傾向が強く、RHEL 7.2以降をプロダクションシステムに採用することが今後は増加すると思われます。

次回は本誌・2015年12月号で予告したIdentity Managementの認証について紹介する予定です。SD

▼写真1　Intel SSD DC P3608



▼図10　fstrimコマンドを実行するスクリプトをcronで実行する例

```
# cat /etc/cron.daily/fstrim
#!/bin/bash
logger -t fstrim -i `fstrim -v /`
logger -t fstrim -i `fstrim -v /boot`
# chmod +x /etc/cron.daily/fstrim
# ll /etc/cron.daily/fstrim
-rwxr-xr-x. 1 root root 84 Dec  8 16:55 /etc/cron.daily/fstrim
```

Be familiar with FreeBSD.

チャーリー・ルートからの手紙

第28回 ❖bhyveでOpenBSDファイアウォール on FreeBSDを構築(その3)

◉著者プロフィール
後藤 大地(ごとう だいち)
BSDコンサルティング(株)　取締役／(有)オングス　代表取締役／FreeBSD committer
エンタープライズシステムの設計、開発、運用、保守からIT系ニュースの執筆、IT系雑誌や書籍における執筆まで幅広く手がける。カーネルのカスタマイズから業務に特化したディストリビューションの構築、自動デプロイ、ネットワークの構築など、FreeBSD/Linuxを含め対応。BSDコンサルティングでは企業レベルで求められるFreeBSDの要求に柔軟に対応。

OpenBSD on FreeBSD、三度

前回はOpenBSDのファイアウォールpf(4)について紹介しました。FreeBSDのハイパーバイザbhyveでOpenBSDを動作させ、OpenBSDネイティブの最新pf(4)を活用するというシナリオの中でpf(4)を取り上げました。

設定ファイルpf.conf(5)と制御コマンドpfctl(8)、それにファイアウォールルールセットの基本機能であるマクロ、リスト、テーブルについては前回紹介しました。マクロ、リスト、テーブルは最初に説明しておくほうが、後のシンタックスの説明がスムーズになるからです。今回はpf(4)の基本的なシンタックスを説明します。

ルールシンタックス

パケットフィルタリングではネットワークインターフェース、レイヤ3(IPv4/IPv6)、レイヤ4(TCP/UDP/ICMP/ICMPv6)、送信元アドレス、送信先アドレス、送信元ポート番号、送信先ポート番号などを条件指定してパケットを特定し、一致したパケットに対して通過を許可するかブロックするかを指定します。**リスト1**のシンタックスが指定の基本です。

適用されるルールは“最後に一致した”ルールです。quickが指定されている場合には、例外的に一致した段階で適用されます。しかし、デフォルトでは最初のルールから最後のルールまで一致するかどうかがチェックされ、最後に一致したフィルタリングルールが適用されます。

設定の最初は、暗黙のうちにすべてのパケットに対してpassのアクションが適用されるしくみになっています。そのためフィルタリングルールにまったく一致しなかった場合には、パケットはそのまま素通りすることになります。次に、それぞれの項目で指定する内容を説明します。

action

actionにはpassまたはblockが指定できます。passが指定された場合、一致したパケットはカーネルに戻され次の処理に進むことになります。blockが指定された場合、一致したパケットは破棄されます。このデフォルトの動作はblock dropと同じです。block returnやblock-policyがreturnに設定されている場合には、ブロックしたTCPパケットに対してはTCP RSTを返す動きをし、それ以外のパ

▼リスト1　pf(4)シンタックスの基本

```
action [direction] [log] [quick] [on interface] [af] [proto protocol] \
   [from src_addr [port src_port]] [to dst_addr [port dst_port]] [flags tcp_flags] [state]
```


各コマンドのうしろについている括弧書きの数字は、manコマンドで見ることができるマニュアルに記載されている章番号を表しています。

ケットに対してはICMP (Internet Control Message Protocol) のUnreachableパケットが返されます。

direction

directionには、指定したインターフェースにおいてパケットが、入ってくる方向の動きをしているか、出て行く方向の動きをしているかを指定します。指定はinかoutで行います。

log

logはpflogd(8)デーモン経由でパケットフィルタリングのログを取るかどうかの指定です。pflogd(8)はパケットフィルタリングのログを取るための専用のデーモンで、pf(4)経由でログを取るために使われています。

quick

quickが指定された場合、ルールはその場所に記載されているというよりも、ルールの一番最後に記載されているかのように振る舞うようになります。つまり、このルールに一致した段階でフィルタリング動作が実施され、それ以降にルールがあった場合でも無視されます。

interface

interfaceにはパケットが通過するネットワークインターフェース名、またはネットワークインターフェースのグループ名を指定します。ネットワークインターフェースグループはifconfig(8)コマンドで作成でき、任意のネットワークインターフェースを追加することができます。

OpenBSDではカーネルに、egress、ppp、carpなどのネットワークインターフェースグループが自動で作成されるしくみになっています。egressはデフォルトルートを保持しているインターフェースを含むグループです。pppとcarpはクローンインターフェースのためのファミリーグループです。

af

afにはアドレスファミリーを指定します。IPv4にはinet、IPv6にはinet6を指定します。ただし、pf(4)は指定されたアドレスなどをベースに自動的にafを決定しますので、アドレスでどちらを使うかが明確な場合には指定しなくても動作します。

protocol

protocolに指定するのはレイヤ4のプロトコルです。指定できるプロトコルは/etc/protocolsに記載されているプロトコルのうち、プロトコル番号が0から255までのものです。代表的なところではtcp、udp、icmp、icmp6などを指定します。複数のプロトコルを指定する場合にはリストを使います。

src_addr、dst_addr

src_addrまたはdst_addrにはIPヘッダのアドレス(送信元または送信先)に相当するものを指定します。アドレスとしては**表1**のものを指定できます。**表1**の指定に対して先頭に「!」を指定すると、指定した対象以外の対象に一致するようになります。

このようにpf(4)のアドレス指定は柔軟で強力です。割り当てられたIPアドレスが変更される場合にも設定をアップデートできるなど、現実の状況に応じた設定が可能です。

src_port、dst_port

src_portまたはdst_portにはレイヤ4パケットヘッダのポート番号を指定します。次のような条件または範囲指定ができます。

- 指定できるポート番号は1から65535
- 指定できるサービス名は/etc/servicesに記載されている必要がある

▼表1　src_addr、dst_addrに指定できるアドレス一覧

指定書式	説明
単一のIPv4アドレス	—
単一のIPv6アドレス	—
単一のCIDRネットワークブロック	—
DNS経由で名前解決できる完全修飾ドメイン名	ルールセット読み込み時にDNSで名前解決できる必要がある
単一のネットワークインターフェース名	ネットワークインターフェースに割り当てられている1つ以上のIPアドレスに置換される
単一のネットワークインターフェースグループ名	ネットワークインターフェースに割り当てられている1つ以上のIPアドレスに置換される
単一のネットワークインターフェース名/ネットマスク	ネットワークインターフェースに割り当てられている1つ以上のIPアドレスが持つ、それぞれのCIDRネットワークブロックのアドレスに展開され置換される
(単一のネットワークインターフェース名)	ネットワークインターフェースに割り当てられたIPアドレスが変更になった場合にルールをアップデートする指定。DHCPやダイヤルアップなどでの使用を想定
(単一のネットワークインターフェースグループ名)	ネットワークインターフェースに割り当てられたIPアドレスが変更になった場合にルールをアップデートする指定。DHCPやダイヤルアップなどでの使用を想定
単一のネットワークインターフェース名:network	CIDRネットワークブロックに置換される
単一のネットワークインターフェース名:broadcast	ネットワークブロードキャストアドレスに置換される
単一のネットワークインターフェース名:peer	ピアツーピアリンクのピアIPアドレスに置換される
(単一のネットワークインターフェース名:0)	置換にエイリアスIPアドレスを含めない
(単一のネットワークインターフェースグループ名:0)	置換にエイリアスIPアドレスを含めない
単一のネットワークインターフェース名:network:0	置換にエイリアスIPアドレスを含めない
単一のネットワークインターフェース名:broadcast:0	置換にエイリアスIPアドレスを含めない
単一のネットワークインターフェース名:peer:0	置換にエイリアスIPアドレスを含めない
単一のテーブル	—
単一のリスト	—
any	すべてのアドレス
all	any to anyに同じ
urpf-failed	ソースアドレスがUnicast Reverse Path Forwardingチェック経由で使われることを指定

- **!=**……不等
- **<** ……よりも小さい
- **>** ……よりも大きい
- **<=** ……同じまたはそれよりも小さい
- **>=** ……同じまたはそれよりも大きい
- **><** ……二項演算子での範囲指定(範囲は含まない)
- **<>** ……二項演算子での範囲指定以外に一致(範囲は含まない)
- **:** ……二項演算子での範囲指定(範囲を含む)
- 単一のリスト

tcp_flags

tcp_flagsにはprotocolにtcpを指定した場合に使える指定で、TCPヘッダに指定されるフラグを指定します。フラグの指定は、flags S/SAのように「flags チェック／マスク」で指定します。たとえばflags S/SAの指定であれば、SYNフラグがonになったSYNおよびACKが一致の対象になります。flags anyが指定された場合にはpf(4)はフラグのチェックは行いません。

state

stateにはルールにマッチしたパケットのステート情報をどのように処理するかを指定します。**表2**の指定が可能です。

作り方❶：デフォルトブロック

pf(4)の基本的な使い方がわかったら目的とする

▼表2　stateに指定できる処理一覧

指定書式	説明
no state	ステートを保持した状態ではトラッキングしない(TCP、UDP、ICMPに適用可能)
keep state	デフォルトの挙動(TCP、UDP、ICMPに適用可能)
modulate state	パケットマッチングのためにTCPの初期シーケンス番号を生成する(TCPにのみ適用可能)
synproxy state	TCPなりすましSYN flood攻撃からサーバを保護するために使用する機能。プロキシのインカミングTCPコネクションに適用される。機能的にはkeep stateおよびmodulate stateの双方の機能も備えている

パケットフィルタリングルールを記述していくわけですが、推奨される作り方があります。次のように、最初はすべてのアクセスを拒否する設定をスタートポイントとし、ここに必要となるアクセスを許可していくというものです。

```
block in all
block out all
```

このアプローチは石橋を叩いて渡るような方式で安全方向に設定が振られることになるほか、ルールの記述がシンプルになる効果があるため、作業方法として推奨されています。

作り方❷：一致ルールは可能な限り狭く

すべてのパケットをブロックする設定をしたあとは、使用したいパケットに関して通過の規則を書いていきます。このとき一致させるパケットを可能な限り厳密に、対象となるパケットだけになるように記述するのが1つのポイントです。

たとえば対象のホストのIPアドレスが192.168.1.1、所属しているネットワークが192.168.1.0/24だとします。fxp0のインターフェースを通じて、ネットワークとこのホストの間の通信のみを許可するのであれば、**リスト2**のようにフィルタリングルールを記述します。

Webサーバのように外部にサービスを提供しているのであれば、**リスト3**のようにポート番号を指定してフィルタリングルールを記述します。対象のインターフェースを指定していることと、送付先としてもこのインターフェースを指定しているあたりがポイントです。

最初にすべてブロックする設定を有効にしておくと、ここでのルールの記述がより厳密になりやすくなります。

作り方❸：quickに注意

すべてをブロックする設定から作りはじめた場合には大丈夫なのですが、ルールに一致しなかったものはすべて通過させる設定になっている場合には注意が必要です。たとえば**リスト4**の設定はsshdへの接続を拒否する「気持ち」のこもったルールですが、これは気持ちどおりには動作しません。

pf(4)ではいちばん最後に一致したルールが適用されます。**リスト4**の例ですと、最後のルールにすべて一致してしまいますので、sshdへのアクセスはブロックされることなく動いてしまいます。この場合には**リスト5**のようにquickを指定して、即座にルールが適用されるようにします。quickの適用されたルールセットが実行された場合、それ以降の行は無視されるようになるためです。

次回はステート

今回はpf(4)の基本的なシンタックスと、その使い方を説明しました。前回のマクロ、リスト、テーブルと組み合わせると、すでにかなりの規則が記述できるようになっていることに気がつくと思います。記述もシンプルですし、pf(4)に人気がある理由もわかるような気がしますね。

ファイアウォールではステートを加味した処理も重要になってきます。次回はpf(4)におけるステートの考え方や記述方法などを説明します。SD

▼リスト2　記述例その1

```
pass in on fxp0 from 192.168.1.0/24 to 192.168.1.1
pass out on fxp0 from 192.168.1.1 to 192.168.1.0/24
```

▼リスト3　記述例その2

```
pass in on fxp0 proto tcp from any to fxp0 port www
```

▼リスト4　sshdへの接続をブロックしたい(が、ブロックされない)

```
block in on fxp0 proto tcp to port ssh
pass in all
```

▼リスト5　sshdへの接続をブロック

```
block in quick on fxp0 proto tcp to port ssh
pass in all
```

32

Debian Developer　やまねひでき　henrich@debian.org

DebConf15レポート（後編）と、Debianの近況

Debian Hot Topics

DebConf15レポート（つづき）

この原稿を書いているときは年末進行真っ盛りですが、今回もまずは、この夏のDebConfレポート（後編）をお届けします。

Linux以外の移植版の話――kFreeBSDとHurd

Debianと言えば、Linuxカーネル以外にもFreeBSD、そしてHurdを使うバージョンもあることをご存じの方も多いかと思います。今回のDebianカンファレンスでも関連した発表がありましたので、ご紹介しましょう。

「GNU/kFreeBSD explained」のセッションでは、FreeBSDカーネルを使うkFreeBSD[注1]移植版についての説明がされました。まず、これまでの歴史について**表1**のような流れが説明されました。そして、現状のkFreeBSDで安定して400日以上連続稼働しているプロダクション環境としてgit.gnupg.orgが挙げられました（ほかの例として、筆者のブログからDebian勉強会の参加者のスクリーンショットが紹介されていたのにはちょっと驚きました）。

表1にあるように、Debian 8（Jessie）からはkFreeBSDが外されてリリースされました。しかし、これとは別にjessie-kfreebsdリリースを作成しており、セキュリティアップデートも安定版同様に受け取れるように関係者間で調整したため、Debian 7（Wheezy）からアップグレードが可能です。**リスト1**のapt lineを指定すれば良い、とのことです。

ただ、kFreeBSDでのリリースブロッカー[注2]や移植作業の注意点の話が聴けるかと思ったのですが、そういう話ではなかったので、若干肩透かしでした。

「Debian GNU/Hurd status update」というセッションでは、Hurdの特徴として「マイクロカーネルで独立性が高いため、たとえドライバの品

注1）kFreeBSDのkは「kernel」を意味します。

注2）修正しないかぎり、リリースができないような致命的なバグや問題のこと。「Show stopper」とも呼ばれることがある（Windows NT開発を題材にした書籍『闘うプログラマー』の原題でもありますね）。

▼リスト1　kFreeBSDのDebian 8「Jessie」でのapt line

```
deb http://httpredir.debian.org/debian jessie-kfreebsd main
deb http://security.debian.org/ jessie-kfreebsd/updates main
```

▼表1　Debian GNU/kFreeBSD関連タイムライン

年	出来事
1999年	取り組みが開始
2001年	NetBSDのchrootでDebianが動作（だが、当時はサーバのディスク容量などが問題に）
2003年	i386でのchrootでのkFreeBSDがリリース
2006年	amd64に移植
2011年	テクノロジプレビューとしてDebian 6でリリース
2013年	89%のパッケージがビルド完了
2015年	Debian 8にて、サポートするリソース不足の観点から、公式リリースからは除外

質が悪くクラッシュを引き起こしたとしても影響範囲が小さいこと」や「Linuxでいうところのunionfs[注3]のような『remap』機能の存在」を説明し、内部構造などの紹介を行いました。

そして、Hurdの移植作業で面倒なこととして、アプリケーション作者がHurdの存在を知らない、あるいは、移植性を無視しているために、次のような「決め打ち」されているビルドスクリプトがあるという問題が挙げられました。

- 「make -j $(grep … /proc/cpuinfo)」のように、「/proc/cpuinfoが存在すること」が前提となっている
- 「LinuxでもBSDでもない、ではWindowsだな」と、「#include <windows.h>」を指定している
- ヘッダファイルmach.hが存在するならMacに違いないと判定している(実際はHurdも)
- Linuxを前提とした「#include <linux/limits.h>」という記述がある
- ハードコードされたerrno値がある(Hurdでは違う値が割り振られている)
- ビルドに際して-lpthread、-ldl、-lX11などのライブラリ指定が抜けている

とはいえ、移植作業は順調に進捗しており、

- 81%のパッケージがビルドできている
- initとしてsysvinitが利用可能になった[注4]
- jessie/sid snapshot CDが2015年5月にリリースされた
- 64bitサポートの作業が始まった(起動はするものの、RPC周りがまだ完了していない)

という状況です。これからも地道に活動が進んでいくことでしょう。

GnuPG関連セッション

「GnuPG: Past, Present and Future」というセッションでは、「こんなに重要なツールなのに、開発者が1人しかいない!?」と、話題になったGnuPGのメイン開発者であるWerner Kochさんが、在住しているドイツ開催ということでゲストスピーカーとして招かれました[注5]。

Kochさんは、GnuPGの歴史として、1991年のPGP2(Pretty Good Privacy 2)の話から始め、暗号にかかわる特許や輸出規制との戦い、それから商用サポート企業が出てきたものの自由なPGP実装はまだなかった様子などを説明しました。そこにGnuPG(当時はg10という名称)が1997年に現れて状況が変わり、DebianへGnuPG関連パッケージが導入されることに……という「過去」の出来事を話しました。

続く「現在」の話では、関連RFCのアップデートや各種の楕円曲線アルゴリズムの採用について考慮をしている旨を説明。

そして「未来」については、「スノーデン事件でGPGの重要性を気づかせてもらったが、GPGのセキュリティモデルである"Web-of-Trust"[注6]は一般の人々にはなかなか難しいもので、どのように大衆の利用を浸透させていけばいいのか検討中である」と述べました。新しいモデルとして"TOFU"[注7]を採用しようとしているとのことです。

また、多くの人が気にかけている資金面や開発の状況についても、どのような手助けがあったのかという話や、現状では複数の開発者が作業しているという話が行われました。

注3) 複数のファイルシステムのファイルやディレクトリを透過的に重ね、単一の一貫したファイルシステムとして見えるようにする機能。Linuxでの実装としてはaufsやoverlayfsなどがある。

注4) Linuxではsystemdで騒いでいる人がいましたが、Hurdでは自作の謎スクリプトからsysvinitへの移行がようやく実施された、ということです。

注5) 彼のラップトップPCの天板には大きなDebianのロゴが描かれていたのを申し添えておきます :-)

注6) 「友達は信用できる。友達の友達も信用できる」という形で、信頼の輪を広げていく手法。ここで注意したいのは信用/信頼というのは「本人である」ことを信じるのであって、相手が行うことを何でも信用する、というわけではありません。このモデルの問題は「最初に信用する友達をどうやって見つけるか」という点と「きちんと信頼できる相手であると認識できるか」という点にあります。

注7) Trust On First Useの略。「初回に接続/交換した相手は、変更がなければそれ以降も信用に足るものとして扱う」という手法で、SSHで利用されています(が、日本人には「豆腐」しか思い浮かばないですね……)。

そしてもう1人、日本で唯一のGnuPGの開発者であるg新部さんによる「More Entropy, Please」のセッションは、おそらくDebConf初の家族での発表でした。内容が気になる方はYouTubeで検索してみてください。

NEXT➡アフリカ

次回のDebConfは2016年7月3～9日に、南アフリカで行われることが発表されました。会場は南アフリカ最古の大学、ケープタウン大学です。アフリカ大陸初の開催であり、運営チームも張り切っているようで期待できますね。

日本からの場合、ビザの問題はとくにありません。ですが、パスポートは未使用査証欄が南アフリカ入国時に2ページ以上必要で、余白が足りないと入国時にトラブルとなる場合があるので参加を検討する方は注意してください。

また、開催に向けて資金調達が始まりました。スポンサーとなっていただけそうな団体に心当たりがありましたら、「sponsors@debconf.org」宛にご連絡ください。

Debian開発の近況

ライブラリの「transition」

さて、ここからは現状のDebianの開発について見ていきましょう。

おおまかには、比較的淡々と進んでいると言えます。ライブラリのバージョン変更に伴う大規模なリビルド(2015年4月リリースのgcc5への移行と付随するlibstdc++のバージョンアップのためのリビルド、2015年7月リリースのOpenSSL 1.0.2に対するリビルド、そして2015年12月リリースのPerl 5.22に対するリビルドなど)がいくつか行われていますが、特段の不具合もなく進んでいます。libstdc++については登録されているバグも9割方完了しており[注8]、規模のわりには順調なようです。

ライブラリのABI[注9]が変わると、非互換性を表すため「soname」[注10]のバージョンが上がります(例：fooパッケージがアップデートされて、/usr/lib/foo.so.0が、/usr/lib/foo.so.1となる)。これをパッケージマネージャーがうまく取り扱うために、libfooソースパッケージから生成されるバイナリパッケージには「libfoo0」という名前を付けておき、sonameのバージョンが上がったときには「libfoo1」というパッケージを提供するなどして明示するようにします。

この際、libfooに依存しているすべてのパッケージはlibfoo0ではなくlibfoo1を使うようにリビルドされないと、依存関係が破損してインストールできなくなります。このリビルドの作業を「transition」と呼んでいます[注11]。

transition作業の流れ

①まず、新しいライブラリパッケージを含むソースを、experimentalリポジトリにアップロードする。これにより、ビルドが必要なパッケージをリストアップ／再ビルドを行うtransitionスクリプトが自動的に準備される

②transitionスクリプトを確認後、逆依存関係(reverse dependency)にあるパッケージを手元でリビルドして問題ないことを確認する

③release.debian.org擬似パッケージへバグ報告[注12]をして、リリースチームにtransitionを要求する

④リリースチームから返信があったら、unstableリポジトリにアップロードしなおし、transitionが開始される

⑤各ソフトウェアが、各アーキテクチャのbuild daemonで再ビルドされ、バージョン番号に

注8) URL https://goo.gl/OHpO5U を参照。

注9) 「Application Binary Interface」の略。この場合はアプリケーションとライブラリのインターフェースを指す。

注10) 「Shared Object Name」の略で、共有ライブラリファイル中で提供している機能のこと。sonameの数字は、互換性がどのレベルで保たれているのかを表すために使われる。

注11) 詳細は URL https://wiki.debian.org/Teams/ReleaseTeam/Transitions を参照。

注12) Debian BTSはパッケージ単位での管理のため、パッケージ外のissueについては擬似パッケージ(pseudo package)としてパッケージのように取り扱います。

は「+b1」のように付けられる(リスト2)

なお、この様子は「Transition tracker」でおおまかな状態が確認できますので、興味を持った方はたまに覗(のぞ)くなどしてみてください[注13]。

Debian LTSに初のプラチナスポンサー

喜ばしいニュースです。Raphael Hertzogさんのブログ[注14]によると、東芝がToshiba Software Development(Vietnam)を通じて、Debian LTSのサポート(**図1**)に初のプラチナスポンサーとして加わりました。これにより、フルタイムサポートまであと35%の資金調達のところまで到達しています。どのぐらいの期間、スポンサーをしてもらえるのかは、不透明ですが、長くなることを期待したいところです。

そうそう、東芝は日本国内でも改札機にDebianを採用したことが「信頼性と拡張性を備えた新型自動改札機EG-5000」という論文[注15]で公表されています。読者のみなさんが普段利用している改札機も、もしかしたらDebianが動いているのかもしれませんよ。

インストール「CD」の削除

debian-installerはバージョン9 alpha4がリリースされました。これに伴い、通常のインストール用CDが削除されました。Debianのインストールについては、小さなメディアサイズのものを選びたい人は、今どきネットワークインストールを選択するでしょうし、光学メディアを使うのであれば、DVDがかなり普及しています[注16]。CDイメージを大量に作成して配布するのは現状の利用に見合わない、という判断です。すべてを用意しようとすると1アーキテクチャあたり数十枚という規模でビルドすることになりますし、良い判断ではないでしょうか。

このほかにも議論されているクラウド用のイメージについて、Debian 9(Stretch)では進展があることを願いたいところです。

小さなことからコツコツと

最後に小ネタを。先日aptのChangeLogを見ていたら次のような修正が入っていました。

```
* select kernels to protect from ↵
autoremove based on Debian version
    (Closes: 787827)
```

「apt-get autoremoveの際、今現在稼働しているカーネルパッケージは削除しないようにするよ」という内容です。いやはや、今までこんなバグが直っていなかったのですね。

とはいえ、このような細かいけれどもちょっと気になるというような挙動でも、キチンとレポートを行えば対処されるという良い例のように思います[注17]。

みなさんも日ごろ、Debianを使っていて気になることがあれば、手元でメモを作っておき、メーリングリスト、勉強会、イベントなどで、ほかのユーザと話し合ってみてください。もしかしたら、バグ報告までたどり着いてうまくFixできる「Contribution」の機会かもしれませんよ。SD

注13) URL https://release.debian.org/transitions/ 参照。
注14) URL https://raphaelhertzog.com/2015/11/13/freexians-report-about-debian-long-term-support-october-2015/
注15) URL https://www.toshiba.co.jp/tech/review/2010/10/65_10pdf/f05.pdf
注16) 日本に住んでいるとあまり意識しませんが、世界には最近になってようやくDVDドライブが広まってきた、という地域もあるのです。
注17) 逆に言うと報告しないと、バグであっても存在しないも同然、ということでもありますが。

▼リスト2 再ビルドされたパッケージのChangeLogの例

```
apache2 (2.4.17-2+b1) sid; urgency=low, binary-only=yes
                    ↑バージョン番号に「+b1」と付いている
  * Binary-only non-maintainer upload for amd64; no source changes.
  * Rebuild against libssl1.0.2.

 -- amd64 Build Daemon (binet) <buildd-binet@buildd.debian.org>  ↵
Sat, 31 Oct 2015 23:17:11 +0100
```

▼図1 Debian LTSのロゴ

LibreOffice 5.1の新機能

Ubuntu Japanese Team　あわしろいくや

今回は、本誌発売後にリリース予定のLibreOffice 5.1の新機能や変更点について解説します。

LibreOffice 5.1概要

LibreOffice 5.1は、タイムベースリリースの方針に従って、半年に一度のリリースが行われているLibreOfficeの新バージョンです。2月第1週のリリースを目指して開発が進んでいます。

半年に一度のリリースのうち、年の前半リリース分（すなわち今回）は割に新機能が多め、後半リリース分は控えめというのが伝統になっていました。なぜなら、Google Summer of Codeで開発された機能が取り込まれるのが前半リリース分だからです。とはいえ、今回は大きな新機能はさほど多くありませんでした。そのぶんUIの見直しが多いです。これは4.xのころから続いている傾向です。

今回は諸事情によりベータ2で検証するため、メニューの未訳が多いです。英語になっているものも、日本語に翻訳されているかのように紹介していますが、実際にリリースされる翻訳とは一部異なる可能性があることをご承知おきください。

全般

メニューの追加

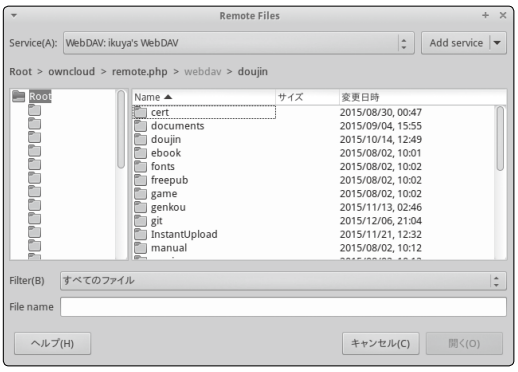
WriterとCalcとImpressでは、それぞれ［スタイル］、［シート］、［スライド］というメニューがメニューバーに追加されました。これらのメニューはそれぞれサイドバーにもあるものですが、解像度が低い環境だと常に表示しておくと不都合なことがありますし、かといって表示をオンオフするのも手間ですので、メニューバーに追加されたのは歓迎するべきでしょう。

リモートファイルを開く

これまでも、実際に開けるかどうかはさておき、Google DriveやOneDriveなどのリモートにあるファイルを開いたり保存したりする機能自体はありましたが、ダイアログをデフォルトからLibreOffice独自のものに変更する必要があり、あまり使い勝手がよくありませんでした。

5.1からは［リモートファイルを開く］という機能が

図1　WebDAVを開いたところ



追加され(**図1**)、ダイアログを変更しなくてもよくなりましたが、やはりGoogle DriveやOneDriveにあるファイルにはアクセスできません。ただ、WebDAVやSSHサーバにも対応していますし、前者に関しては改善点もあるため、たとえばownCloudを使用している場合にはWebDAVでアクセスしてもいいのではないでしょうか。

サイドバーの項目追加／変更

サイドバーもさまざまな変更がありますが、1つ例として挙げると、Calc/Impress/Drawのオブジェクトで影が付けられるようになりました(**図2**)。もちろん影を付ける機能自体はあったのですが、サイドバーに追加されたことにより簡単に細かな設定ができるようになったのがメリットです。

常に訂正

スペルチェックなどのオートコレクト機能の右クリックに、[常に訂正]機能が追加されました。この機能は、スペルミスがあった時点で即座に訂正するように登録する機能です。実際にUIを見てみるのがわかりやすいと思います(**図3**)。

日本語用の辞書はありませんが、英語など辞書がある言語を入力する際で、かつ手グセでついついtypoしてしまう単語がある場合には便利でしょう[注1]。

図2　サイドバーからオブジェクトに影が付けられるようになった

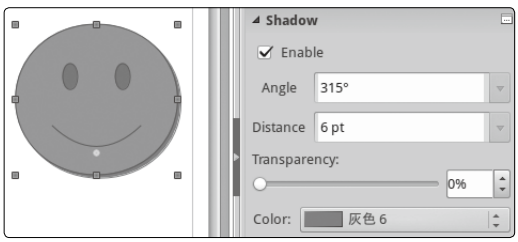
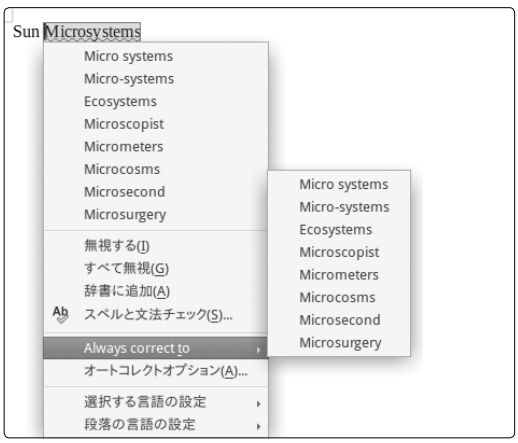


図3　入力した時点で修正できるようになった



[スペルと文法チェック]ダイアログに貼り付けボタンと特殊文字ボタンの追加

[スペルと文法チェック]ダイアログに、貼り付けボタンと特殊文字ボタンが追加されました。とはいえ、どういう用途で便利なのかがよくわかりません。マルチリンガルユーザで、かつその言語の特殊文字の入力方法を知らないユーザ向けとのことですが、筆者もろくに中国語の読み書きができないにもかかわらず、たまに扱うことがあります。しかし、そのような必要にかられたことはありません。そもそも右クリックから貼り付けができますし。というわけで、どのように便利なのかは筆者にはわからないものの、便利に思うユーザはいるのでしょう。

すべての描画にOpenGLを使用

[ツール]-[オプション]-[表示]の[図の出力]の下に[すべての描画にOpenGLを使用(再起動後)[注2]]が追加されました(**図4**)。テスト中にそのような場面

図4　OpenGLでレンダリングするオプションが追加された

注1) ちなみに筆者は"VirtualBox"のtypoが多いです。
注2) 原文はUse OpenGL for all rendering (on restart)

には遭遇しませんでしたが、描画に問題がある場合などはここを確認してみるといいでしょう。

改ページ前後の行数

Writerの[書式]-[段落]-[体裁]の[改ページの前に残す行数]と[改ページ後の行数]にデフォルトでチェックが入るようになりました(**図5**)。チェックがないと、1行だけの段落が改ページの前後に発生することになり、これは見栄えが悪いということのようです。ちなみに英語では"Widow/Orphan Control"で、日本語に直訳すると「寡婦／孤児コントロール」になります。Wikipedinaには由来[注3]が書いてあり、興味深いです。もちろん日本語には該当する表現がないので、改ページ前後としてあるわけです。

余白の非表示

Microsoft Wordをお使いの方であればピンとくると思うのですが、ページ余白とページ余白の間をダブルクリックすると、その余白が非表示になってページがつながっているように表示する機能があります。このたび、これがWriterにも実装されました(**図6**)。Wordにはこの機能に名称はとくについていないようですが、Writerでは余白の非表示[注4]です。

余白を非表示にすると[表示]-[余白の非表示]にチェックが入るので、このチェックを外すか、あるいは同じく余白があったところをダブルクリックすると再び表示できます。

それにしたがって[表示]-[印刷レイアウト]が[表示]-[通常レイアウト]に変更されています。たしかに余白を非表示にしたら印刷レイアウトではなくなるので正しい変更ですが、Wordは印刷レイアウトのままだったりします。

文章を連続して読みたい場合にはこの機能はとても便利で、これだけでもバージョンアップに値するのではないでしょうか。

アウトラインのプリセット

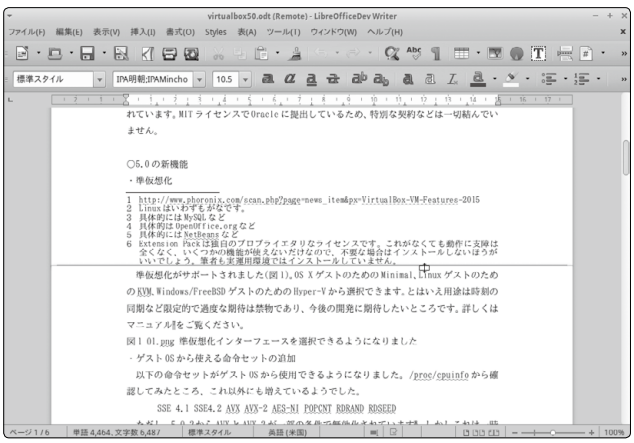
[書式]-[箇条書きと番号付け]を見ると、この機能は[箇条書き][番号付け][アウトライン][画像]の4つから選択できます。しかし、ツールバーにアイコンがあるのは前者2つまででした。このたび[アウトライン]もツールバーにアイコンを表示できるようになりましたが、デフォルトでは非表示です。[表示]-[ツールバー]-[カスタマイズ]-[ツールバー]タブを開き、[ツールバー]を[書式設定]にすると[Outline Presets]があるので、これにチェックを入れて[OK]をクリックするとツールバーに表示されるようにな

注3) https://en.wikipedia.org/wiki/Widows_and_orphans

注4) Hide Whitespace

図5　改ページのオプションがデフォルトで有効になった

図6　ページ間の余白を非表示にできるようになった



ります。現状ではまだアイコンがないらしく、テキストになってしまいます。

Calc

[下の行に挿入]と[右の列に挿入]の追加

これまでは[挿入]-[行]と[挿入]-[列]で、それぞれ現在がカーソルのある行の上と左の列に挿入していましたが、[シート]-[行の挿入]に[上の行][下の行]、同じく[シート]-[列の挿入]に[左の列][右の列]となりました。すなわち下の行と右の列に挿入できるようになったということです。

少なくともMicrosoft Excel 2010/2013にはこのような機能はありませんでしたが、たしかに挿入したい列や行の位置を変更したいことはあるように思います。

PNGでのエクスポート

[ファイル]-[エクスポート]でこれまでのXHTMLとPDFに加えてPNGでもエクスポートできるようになりました。どのような場面で便利なのかはよくわかりません。開発者としてはドキュメントのサムネイルに使えるということですので、将来的にそのような機能が実装されるのを楽しみにしましょう。もっとも、現状でスタート画面ではファイルの内容のプレビューができているのですが……。

書式設定済みの値でも検索と置換が可能に

たとえば1万円という表記をしたい場合、セルに[10000]と入力して右クリック-[セルの書式設定]-[数値]タブ-[カテゴリー]の[通貨]で[-￥1,234]を選択し、[￥10,000]とします。この[￥10,000]を検索したい場合、[10000]で検索することはできます。しかし、[￥10,000]で検索することはできません。よって、今回[書式設定した表記で検索[注5]]にチェックを入れることによって検索ができるようになりました(**図7**)。

WEEKNUMとWEEKNUM_ADD関数の変更

WEEKNUM関数は、指定した日がその年の何週目かを計算する関数です。WEEKNUM_ADDも同様ですが、実際に値を入力してみると、結果が違うことがあります。

たとえば2015年12月6日を例にとると、WEEKNUMは49、WEEKNUM_ADDは50を返します。WEEKNUMはISO 8601[注6]に基づき、1月4日がある週を第1週としています。2015年1月4日は日曜日であり、1月1日から起算すると2週目となります。すなわちWEEKNUMで1週少ない結果になります。Microsoft ExcelのWEEKNUM関数ではWEEKNUM_ADDと同じ結果になります。

一見とくに何の問題もなさそうですし、日本では何週目かを気にすることはあまりないため、余計に何が問題なのかわかりくいのですが、ODF 1.2の関数の定義とは異なるということで、修正されることになりました。

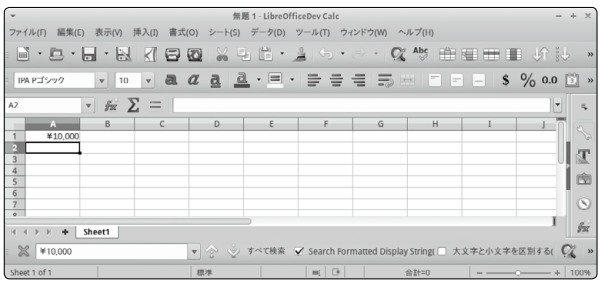
まず、ISO 8601に定義された値を返すISOWEEKNUMという関数が追加されました。オプションはとくになく、日付を入れればいいだけのシンプルな関数です。これがODF 1.2の仕様書にはなかったものの、LibreOffice Calcには実装されていなかった関数です。

互換性を維持するため、WEEKNUM関数はとくに

注5) 原文はSearch Formatted Display Stringで、これは参考訳です。

注6) https://ja.Wikipedia.org/Wiki/ISO_8601

図7 この条件で検索ができるようになった



変更はありません。また、ODF 1.2の仕様書との齟齬もなさそうですので、修正する必要もなかったのでしょう。

WEEKNUM_ADDはWEEKNUM_EXCEL2003に名称変更されました。ドサクサに紛れてNETWORKDAYS_ADDもNETWORKDAYS_XLSに名称変更されています。

Impress/Draw

スライドナビゲーション

新設された[スライド]メニューに、[最初のページへ][前のページへ][次のページへ][最後のページへ]が追加されました。意外なことに、スライドペインでのページ移動のためのUIは今までありませんでした。

また、Ctrl+Shift+Homeキーで現在のスライドを一番上に、Ctrl+Shift+↑キーで現在のスライドを1つ上に、Ctrl+Shift+↓キーで現在のスライドを1つ下に、Ctrl+Shift+Endキーで現在のスライドを一番下に移動させることができるようにもなりました。

マスタスライドの背景

マスタスライドの編集中の背景が、従来よりもより濃い灰色になり、区別がつきやすくなりました。とはいえ、気をつけて比べてみないとわかりませんし、そもそも今編集しているのがマスタスライドなのかどうかの区別がつかないこともあまりないと思われるので、あまり気にしなくてもいい気はします。

スクリーンセーバーの停止

KDEとXFCEとMATEを使用している場合でも、スライド実行中にスクリーンセーバーが確実に停止されるようになったとのことですが、筆者がXubuntu 14.04 LTSで確認したところだとやはりスクリーンセーバーが起動してしまいました。

編集モードの切り替え

Impressで編集モードがツールバーから変更できるようになりました(**図8**)。これも今まではサイドバーからしかできませんでした。

幅／高さを合わせる

複数のオブジェクトを選択し、[右クリックシェイプ]に[幅を合わせる[注7]][高さを合わせる[注8]]が追加されました(**図9**)。これは微妙に幅や高さが合わなくて困った場合には便利な機能です。

その他：フィルタ

Gnumericのインポートフィルタは以前からありま

図8　ツールバーから編集モードを変更できるようになった

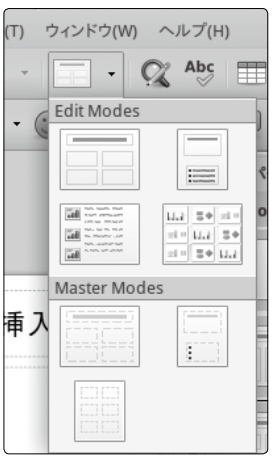
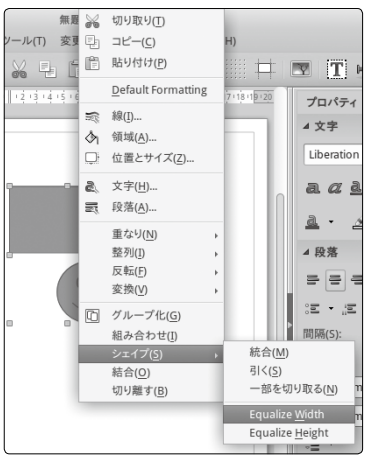


図9　オブジェクトの幅や高さを合わせることができるようになった



注7）原文はEqualize Width
注8）原文はEqualize Height

したが、開発途中で限定的な用途でした。5.1からは普通にサポートされるようになりました。新しめのGnumericはODFもOOXMLも対応しているため必須の機能ではありませんが、Xubuntuは15.10からGnumericに代わってCalcがデフォルトの表計算ソフトになったので、うれしい人はいるかもしれません。

あとは、ほぼ人々の記憶から消えているであろうMicrosoft Write（.wri）形式のインポートにも対応しました。Windows 3.1のころにお目にかかった記憶はあるものの、それ以降はワードパットに置き換えられてソフト自体は目にしなくなりました。しかし、このファイル形式自体はWindows XPのワードパットまでは対応していたそうです。

最後に、一番影響するユーザが多いであろう、Keynote 6のインポートにも対応しました。

開発スケジュールと翻訳

まずは**表1**をご覧ください。

これが5.1のリリーススケジュール[注7]です。RC1のリリースとともにUIの文字変更が禁止となるので、それまでに翻訳を行った場合は無駄になる可能性が若干は存在するということです。そして、RC2に間に合うように翻訳しようとすると、翻訳期間はちょうど1ヵ月となります。RC2に間に合うように終わらせ、RC3は最終確認で、どうしても修正しなければいけないものだけ修正ということになります。もっとも、LibreOfficeの場合はメンテナンスリリースでも翻訳がアップデートされるので、そんなに厳密なものでもありませんが。

翻訳のシステムも見直されており、今はGitのmasterに対する翻訳もできるようになっています。しかし、前述のようにせっかく訳したにもかかわらず文言が修正されれば未訳に逆戻りです。かといってその1ヵ月間に翻訳が集中すると、今度は翻訳のWebインターフェースであるPootle[注8]がダウンし、翻訳ができなくなるということも起きます。

日本語への翻訳状況も芳しいとはいえず、未訳が散見します。翻訳者は筆者を含めて必ずしもオフィススイートに詳しいわけではないので、専門知識がある方を常に求めています。

Ubuntuで5.1を使用する

4月にリリース予定のUbuntu 16.04はLTSであり、もちろん5.1がインストールされるので、最長で5年間お世話になる可能性があります。14.04 LTSからアップグレードした場合は4.2から5.1ですので、大きな違いを感じそうです。

それ以前のUbuntuでは、PPA[注9]からパッケージをインストールしてください。SD

注7）https://Wiki.documentfoundation.org/ReleasePlan/5.1#5.1.0_release

注8）https://translations.documentfoundation.org/ja/

注9）https://launchpad.net/~libreoffice

表1　5.1のリリーススケジュール

event	date
Alpha1	Week 42 , Oct 12, 2015 - Oct 18, 2015
Hard feature freezed & branched libreoffice-5-1 (initially week 47)	Week 48 , Nov 23, 2015 - Nov 29, 2015
Beta1 (initially week 47)	Week 48 , Nov 23, 2015 - Nov 29, 2015
Beta2 (optionally)	Week 49 , Nov 30, 2015 - Dec 6, 2015
Hard English string & UI freeze	Week 51 , Dec 14, 2015 - Dec 20, 2015
RC1	Week 51 , Dec 14, 2015 - Dec 20, 2015
Hard code freeze & branch libreoffice-5-1-0	Week 2 , Jan 11, 2016 - Jan 17, 2016
RC2	Week 2 , Jan 11, 2016 - Jan 17, 2016
RC3	Week 4 , Jan 25, 2016 - Jan 31, 2016
Release 5.1.0	Week 5 , Feb 1, 2016 - Feb 7, 2016

第47回

Linux 4.1の新機能
mdをクラスタに対応する md-cluster

Text：青田 直大　AOTA Naohiro

今月はLinux 4.1の新機能から、md(RAID1)をクラスタに対応するmd-clusterについて紹介します。

mdadmによるRAID構成

md driverはMultiple Device driverのことで、おもにLinuxでストレージのソフトウェアRAIDを構成するために用いられています。md-clusterに入る前に、まずはmdについて簡単に見ていきます。

Linux mdはLINEAR、RAID(0、1、4、5、6、10)、MULTIPATH、FAULTY、CONTAINERの構成をとることができます。

LINEARは複数のデバイスをリニアにつなげて、1つの大きなデバイスとして使用する構成です。

MULTIPATHでは、1つの物理デバイスに対して複数のファイバチャネルなどのアクセスパスを用意し、それらから1つのmdデバイスが構成されます。1つのアクセスパスに障害が発生しても、ほかのアクセスパスを用いて物理デバイスにアクセスできるようにするものです。しかし、この機能は現在開発・テストされていないので、代わりにLVMのMultipathターゲットを使うことが推奨されています。

FAULTYは実用的に使われるものではなく、テスト用に使われる構成です。この構成は読み書きのI/Oが失敗するディスクをシミュレートし、そのうえでファイルシステムやほかのmdデバイスが正しく振る舞うかどうかをテストするために使うことができます。

CONTAINERはディスクをグループ化し、「外部メタデータ」をサポートするための構成です。ここでいう「外部メタデータ」とは、Linuxの外部から操作可能という意味になり、外部のツールやハードウェアから認識できるということになります。現在はCommon RAID Disk Data Format(DDF)[注1]と、Intel Matrix Storage Manager(IMSM)のフォーマットをサポートしています。mdコンテナを作り、さらにその上にRAIDを構成することで、指定したフォーマットでメタデータが記録されます。すなわちIMSMで作成した場合、BIOSからもRAIDデバイスとして見えるようになるということです。

注1) http://www.snia.org/tech_activities/standards/curr_standards/ddf

mdデバイスの作成

ではmdデバイスを作成し、その中身を見てみましょう。ここでは/dev/sdaと/dev/sdbをRAID1の構成で/dev/md0というデバイスを作成します。また、Linux mdのメタデータフォーマットには0.90、1.0、1.1、1.2の4つがありますが、ここでは後の都合上1.2を使います。同じく都合上、bitmapを作成するように設定します。

mdadmコマンドを**図1**のように使うとdmesgにいろいろとログが表示され/dev/md0が作成されます。ここで"resync of RAID array md0"と出ていることに注目してください。RAID1デバイス作成時点では、その構成デバイスの内容は多くの場合一致していません。そこでresyncというプロセスが走ります。これはmasterである/dev/sdaから/dev/sdbへとデータを同期するものです。/proc/mdstatを見ると、resyncの進行状況を見ることができます。resyncが完了するとメタデータ以外の部分では、/dev/sdaと/dev/sdbの内容が一致します。また/proc/mdstatを見るとresyncの進行状況が表示されていた行が消えています。

▼図1　mdadmコマンドでRAID1デバイスを作成

```
（メタデータがわかりやすいように、ディスクの先頭をクリア）
# dd if=/dev/zero of=/dev/sda bs=4096 count=16
（md デバイスの作成）
# mdadm --verbose --create /dev/md0 --metadata 1.2 --bitmap internal --level raid1 -n 2 /dev/sda /dev/sdb
mdadm: size set to 16760832K
mdadm: array /dev/md0 started.
# dmesg
[56573.613743] md: bind<sda>
[56573.616719] md: bind<sdb>
[56573.675646] md/raid1:md0: not clean -- starting background reconstruction
[56573.675649] md/raid1:md0: active with 2 out of 2 mirrors
[56573.676236] created bitmap (1 pages) for device md0
[56573.676891] md0: bitmap initialized from disk: read 1 pages, set 256 of 256 bits
[56573.678354] md0: detected capacity change from 0 to 17163091968
[56573.681827] md: resync of RAID array md0  ←resync
[56573.681829] md: minimum _guaranteed_  speed: 1000 KB/sec/disk.
[56573.681830] md: using maximum available idle IO bandwidth (but not more than 200000 KB/sec) for resync.
[56573.681832] md: using 128k window, over a total of 16760832k.
（resync 中）
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb[1] sda[0]
      16760832 blocks super 1.2 [2/2] [UU]
      [=========>...........]  resync = 45.5% (7642688/16760832) finish=5.4min speed=28044K/sec
      bitmap: 1/1 pages [4KB], 65536KB chunk

unused devices: <none>
（resync 完了）
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb[1] sda[0]
      16760832 blocks super 1.2 [2/2] [UU]
      bitmap: 0/1 pages [0KB], 65536KB chunk

unused devices: <none>
（メタデータ部分に不一致がある）
# cmp  /dev/sda /dev/sdb
/dev/sda /dev/sdb differ: byte 4257, line 1
（メタデータをスキップすると完全に一致する）
# cmp -i $(( 4096 * 3 )) /dev/sda /dev/sdb
```

mdデバイスのメタデータ

resyncが完了したところでmdデバイスのメタデータを見ていきましょう。前述したとおり、mdadmには0.90、1.0、1.1、1.2の4つのフォーマットがあります。0.90は、かつてはデフォルトで使われていた古いフォーマットです。このフォーマットではデバイスのサイズや数といった制限があるため、現在ではより新しい1.xのフォーマットがLinux 3.1.1からはデフォルトとなっています。1.0、1.1、1.2間の違いはmd super blockの位置にあります。1.0ではデバイスの末尾に、1.1では先頭に、1.2では先頭から4KBの位置にそれぞれsuper blockが置かれます。

/dev/sdaをダンプしてメタデータを見てみましょう(**図2**)。前述したように最初の4KBはスキップされるので、初めのゼロ埋めがそのまま出ています。次の4KBがsuper blockになり、このblock内にRAID arrayおよびディスクのメ

▼図2　/dev/sdaのダンプ

```
# hexdump -C /dev/sda
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|  # 先頭4KBは空白
*
00001000  fc 4e 2b a9 01 00 00 00  01 00 00 00 00 00 00 00  |.N+.............|  # 次の4KB は super block
00001010  6f 3b 19 0e 8e 1e 19 6d  5a 30 9d da a9 21 7c 7a  |o;.....mZ0...!|z|  # 0x1080までは array 固有
00001020  6e 61 6f 74 61 2d 73 65  72 76 30 30 31 3a 30 00  |naota-serv001:0.|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001040  6c 07 6f 56 00 00 00 00  01 00 00 00 00 00 00 00  |l.oV............|
00001050  00 80 ff 01 00 00 00 00  00 00 00 00 02 00 00 00  |................|
00001060  08 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001080  00 80 00 00 00 00 00 00  00 80 ff 01 00 00 00 00  |................|  # 0x10c0まではデバイス固有
00001090  08 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000010a0  00 00 00 00 00 00 00 00  45 21 d4 73 da 35 ec 4b  |........E!.s.5.K|
000010b0  03 a9 95 cb a8 d9 d5 45  00 00 08 00 48 00 00 00  |.......E....H...|
000010c0  13 0a 6f 56 00 00 00 00  89 00 00 00 00 00 00 00  |..oV............|  # 0x1100までは arrayの状態を保持
000010d0  ff ff ff ff ff ff ff ff  e4 a6 8d 05 80 00 00 00  |................|
000010e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001100  00 00 01 00 fe ff fe ff  fe ff fe ff fe ff fe ff  |................|  # 残りのデバイスの役割を示す配列
00001110  fe ff fe ff fe ff fe ff  fe ff fe ff fe ff fe ff  |................|
*
00001200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00002000  62 69 74 6d 04 00 00 00  6f 3b 19 0e 8e 1e 19 6d  |bitm....o;.....m|  # bitmap super(0x2100まで)
00002010  5a 30 9d da a9 21 7c 7a  89 00 00 00 00 00 00 00  |Z0...!|z........|
00002020  00 00 00 00 00 00 00 00  00 80 ff 01 00 00 00 00  |................|
00002030  00 00 00 00 00 00 00 04  05 00 00 00 00 00 00 00  |................|
00002040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*                                                          # bitmap(0x2100-0x2120)
00002120  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00003000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
# hexdump -C /dev/sdb
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
...
00001080  00 80 00 00 00 00 00 00  00 80 ff 01 00 00 00 00  |................|
00001090  08 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000010a0  01 00 00 00 00 00 00 00  89 6f d9 a8 64 98 bd ce  |.........o..d...|
000010b0  6e 10 df ba c1 fd a3 b3  00 00 08 00 48 00 00 00  |n...........H...|
000010c0  6c 07 6f 56 00 00 00 00  00 00 00 00 00 00 00 00  |l.oV............|
000010d0  00 00 00 00 00 00 00 00  47 e0 ba 1a 80 00 00 00  |........G.......|
000010e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

タデータが保存されます。先頭の128byteの0x1080までは、RAID array固有の情報が入っており、RAIDを構成するすべてのデバイスで同一のデータが保存されています。0x1080から64byteの0x10c0までには、構成するデバイス固有のデータが入っています。たとえば、/dev/sdbのほうを見てみると0x10a0 byte目が/dev/sdaでは0x00で、/dev/sdbでは0x01となっています。これはsdaのデバイス番号が0であり、sdbのデバイス番号が1であることに対応しています。同様に0x10a8から0x10b7までのデバイスUUIDも異なっています。0x10c0から0x1100までは、arrayの状態が保持されています。ここにはsuper blockの更新時刻やchecksumが保存されています。そのあとにはディスクの役割を示す2byteの配列が続きます。

Write-Intent Bitmap

メタデータのあとにはWrite-Intent Bitmapと呼ばれるbitmapが続いています。先頭256byteはbitmap superであり、RAID arrayと一致するUUIDなどが保存されています。そのあとには0x2100から0x2120までbitmapが続いています。このbitmapはWrite-Intent Bitmapと呼ばれるもので、resyncを効率的に行うために使われます。

たとえばRAID1構成の場合には、データがmdデバイスに書き込まれたあと、マスタとなるデバイスに書き込まれ、ほかのデバイスにミラーリングを行います。この途中でマシンが落ちてしまった場合、デバイス間でデータの不一致ができているかもしれないのでresyncが必要になります。デバイスのサイズが大きくなるとそれだけresyncに時間がかかります。resyncは2つのデバイスからデータを読み出し、不一致があれば書き出すという重たい作業ですので、あまりに時間がかかるのは好ましくありません。ここでWrite-Intent Bitmapを使ってresync範囲の削減を図ります。

bitmapを使う場合、mdデバイスに書き込みが実行される前に、書き込み先に対応するbitを立ててbitmapを更新します。そのあと、実際の書き込みを行います。立てたbitはある程度の時間、対応するblockへの書き込みがなければクリアされます。デバイスの更新中にマシンが落ちた場合、bitが立っている部分だけをresyncすればデバイス間で同期がとれることになります。

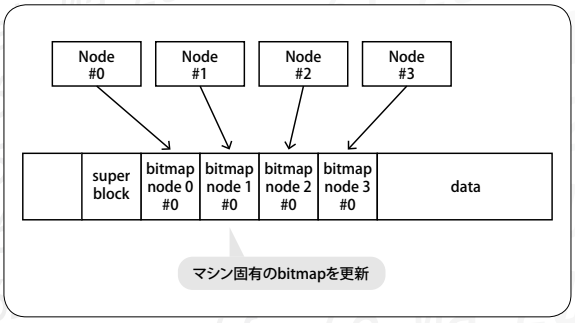
クラスタでのmd構成

では次に、md-clusterのターゲットであるクラスタ環境でのmdデバイスの使用について考えてみましょう。

クラスタ内の4つのマシン間でmd RAID1の共通のストレージを使用する状況を想定します。これを実現するために、md構成用のマシンを1つ用意して、そこでRAID1を構成し、ほかのマシンに見せるという方法が考えられます。しかし、これではmd構成用マシンがSPOF(Single Point of Failure：単一障害点)となってしまいます。

それではクラスタ内のマシンに直接デバイスを見せて、それぞれのマシンでmd RAID1を構成するとどうなるでしょうか。この方法ではbitmapの更新に問題が出てきます。近接するbitに対応するブロックを複数のマシンが同時に更新する場合、一方で立てたbitが消えてしまうことがありえます。もちろんマシン間でロックをとれば正しくbitmapを更新することはでき

▼図3　md-cluster



ますが、書き込みごとにロックをとるのでパフォーマンスが低下してしまうでしょう。

そこでmd-clusterでは、bitmap領域を拡張しクラスタ内の各マシンが個別のbitmapに書き込むことができるようにします(**図3**)。そうすると、ブロックレベルでは、ほかのマシンのことを気にせずに書き込むことができます。

クラスタ間の通信

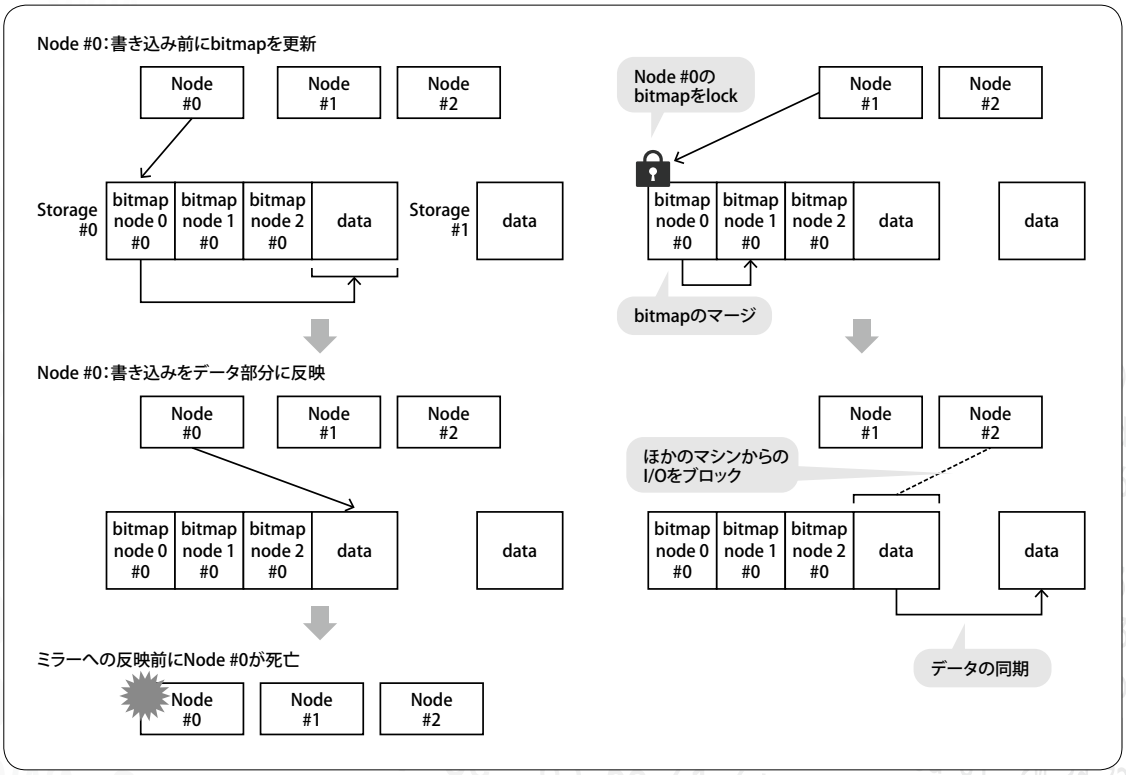
もちろんbitmapを分割するだけですべてうまくいくわけではありません。デバイスの追加や、resyncの実行という部分ではマシン間の協調が必要となります。

例として、マシンが落ちたときにどうなるかを見ていきましょう(**図4**)。マシンが落ちたことは、クラスタ内のほかのマシンに通知されます。通知されたマシンは リカバリスレッドを開始して、落ちたマシンで行われていた書き込みを最後まで引き継ぎます。リカバリスレッドは、まず"bitmap<num>"(numは落ちたマシンの番号)という名前のロックを獲得し、ほかのマシンが同時にリカバリを行わないようにします。

次に、落ちたマシンに対応するWrite-Intent bitmapを開き、自分のbitmapへとマージし、落ちたマシンのbitmapをクリアします。ここで"bitmap<num>"ロックを解放し、マージしたbitmapのresyncを開始します。あるディスク領域にresyncを開始する前に、リカバリスレッドはそのことをほかのマシンに通知し、ほかのマシンからのその領域への読み書きを一時的に停止します。これによって、ほかのマシンのことを考えずに該当領域のresyncを行うことができます。

ここで、①落ちたマシンの通知、②bitmapロックの獲得、③resync領域の通知、とマシン間でのロック管理や通信が必要になっています。最後に、これらの機能を担当するDLMについて

▼図4　マシンが落ちたときの動作



見ていきます。

DLMを使った通信

ロック管理も通信もDistributed Lock Manager(DLM)というサブシステムが担当しています。DLMはOCFS2やGFS2でも使われる分散ロックシステムで、複数のマシン間での共通して用いるリリースへのアドバイザリロック機能を提供します。ここではmd-clusterがどうやってDLMを使い、メッセージをやりとりするかを見ていきます。

まずDLMのロックの特徴について見ていきます。DLMのロックは、ロックモードとLock Value Block(LVB)という特徴的なデータを持ちます。ロックモードは、そのロックを持つプロセスがロックで保護されたリソースに対して何ができ、ほかのプロセスからは何ができるかを決定するためのものです。

ロックモードには、Null(NL)、Concurrent Read(CR)、Concurrent Write(CW)、Protected Read(PR)、Protected Write(PW)、Exclusive(EX)の6つのモードがあります。

NLはロック保持者に何の読み書きも許可しないモードです。NLでのロックは必ずとることができるので、とりあえずNLでロックをとっておいて後にロックモードをアップグレードするといった使い方をします。CRとCWはどちらもほかのプロセスの読み書き両方を許可しつつ、CRは自分に読み込みのみを、CWは自分に読み書き両方を許可するモードです。PRとPWはほかのプロセスからは読み込みのみを許可し、それぞれ読み込みのみ、読み書き両方を自分に許可するモードです。EXは最も排他的なモードでほかのプロセスからのアクセスをいっさい許可しません。LVBはロックを使うアプリケーションが自由に使うことができる16byteの領域です。

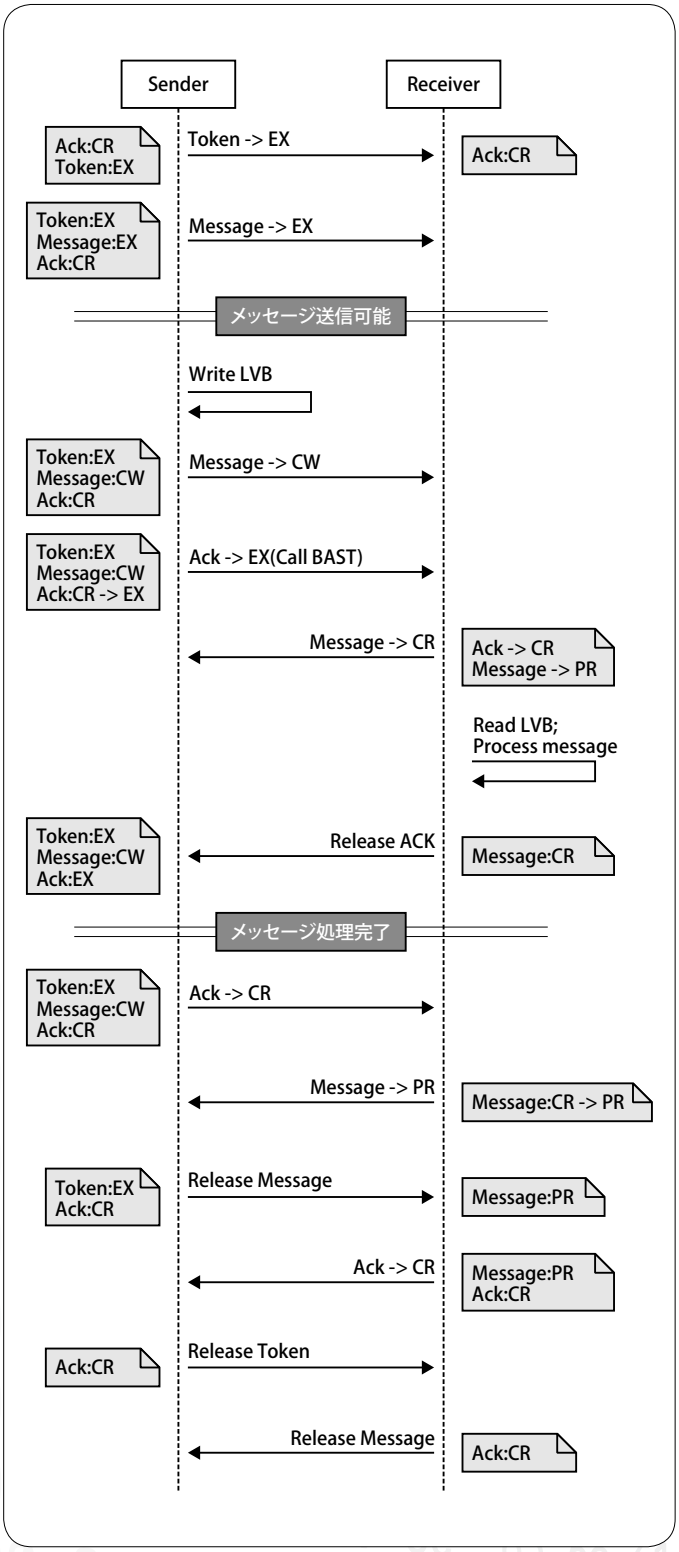
md-clusterでのDLMを使った通信には、“Token”、“Message”、“Ack”という3つのロックが関わってきます。Tokenはメッセージシステムを保護するロックで、このロックの保持者のみがメッセージを送信できます。Messageはデータを送信するためのロックで、このロックのLVBを用いてデータを送受信します。Ackはメッセージを受信したこと、およびメッセージの到着を知らせるためのロックです。

ロックによる通信は**図5**のような流れで行われます。まず、初期状態では送受信者すべてが、AckのCRをとっています。そこから送信側がまずToken、次にMessageの排他的ロックEXを取得します。この時点では誰も同じロックをとろうとしていないとして、どちらもすぐに取得できます。

ここでメッセージの送信準備が整います。MessageのLVBに送信する内容を書き、受信者からメッセージが読めるようにMessageをCWへと制限を緩めます。次にAckを排他ロックEXへアップグレードすることを試みます。この試みによって受信者側でAckに設定されたコールバック(BAST)が呼び出され、受信者にメッセージが来ていることが伝わります。受信者はまずMessageをCRモードでロックをとります。先ほどMessageがCWに変わっているので、このロックはすぐに取得されます。

Messageロックをとれれば、そのLVBを読みだしそのデータに対応する処理(たとえばresync対象領域のサスペンドなど)を行います。メッセージの処理が終われば、Ackのロックを解放します。すべての受信者がAckを解放すれば、送信者のAckの排他ロック獲得が成功し、送ったメッセージがすべての受信者で処理されたことがわかります。送信者はAckをCRへダウングレード、受信者はMessageをPRへとアップグレードします。送信者がさらにMessageを解放したところで、受信者のPRへのアップグレードが完了し、送信者がメッセージの受信完了を認識したことがわかります。ここから先は後片付けの処理になります。受信者はAckをCRへとダウングレードし、Messageを解放します。送信者はToken

▼図5　ロックによる通信



を解放し、両者ともAckをCRモードでとっているだけになり、メッセージの送受信プロセスが完了します。

md-cluster構成の作成

最後に簡単にmd-clusterデバイスの作成方法について紹介します。基本的にはpatchの作者のメール[注2]のとおりに行えばうまくいきます。crosyncおよびpacemakerが設定されたclusterにDLMのリソースを設定しておきます。そして、mdadmコマンドに"--bitmap=clustered"を指定してmdデバイスを作成します。この引数を使うには作者のmdadmリポジトリ[注3]のcluster-mdブランチからビルドしたmdadmコマンドが必要です。ただしブランチの最新版はLinux4.4以後でしか動かないようですので、Linux 4.1から4.3であればcommit 4a3d29edc時点のものを使う必要があります。

まとめ

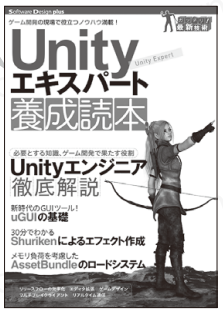
今月はLinux 4.1の新機能であるmd-clusterを中心に、mdおよびDLMについて紹介しました。来月からはLinux 4.2の機能を見ていきたいと思います。SD

注2）http://marc.info/?l=linux-raid&m=141935561418770&w=2
注3）https://github.com/goldwynr/mdadm

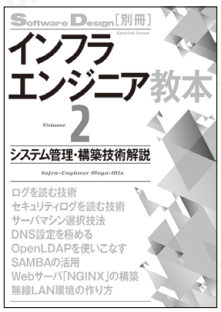
Software Design plus

Software Design plusシリーズは、OSとネットワーク、IT環境を支えるエンジニアの総合誌『Software Design』編集部が自信を持ってお届けする書籍シリーズです。

最新刊！



養成読本編集部 編
B5判・192ページ
定価2,480円（本体）+税
ISBN 978-4-7741-7858-5



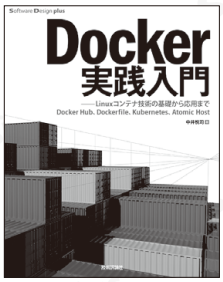
Software Design 編集部 編
B5判・344ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-7782-3

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

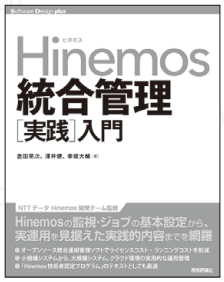
中村行宏、横田翔 著
A5判・320ページ
定価 2,480円（本体）+税
ISBN 978-4-7741-7114-2

川本安武 著
A5判・400ページ
定価 2,980円（本体）+税
ISBN 978-4-7741-6807-4

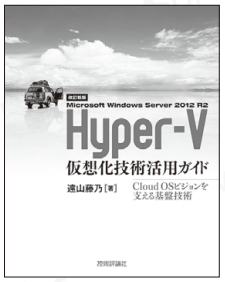
勝俣智成、佐伯昌樹、原田登志 著
A5判・288ページ
定価 3,300円（本体）+税
ISBN 978-4-7741-6709-1



中井悦司 著
B5変形判・200ページ
定価 2,680円（本体）+税
ISBN 978-4-7741-7654-3



倉田晃次、澤井健、幸坂大輔 著
B5変形判・520ページ
定価 3,700円（本体）+税
ISBN 978-4-7741-6984-2



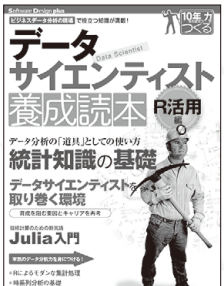
遠山藤乃 著
B5変形判・392ページ
定価 3,500円（本体）+税
ISBN 978-4-7741-6571-4

寺島広大 著
B5変形判・440ページ
定価 3,500円（本体）+税
ISBN 978-4-7741-6543-1

養成読本編集部 編
B5判・192ページ
定価 2,280円（本体）+税
ISBN 978-4-7741-7631-4

養成読本編集部 編
B5判・128ページ
定価 1,980円（本体）+税
ISBN 978-4-7741-7607-9

養成読本編集部 編
B5判・128ページ
定価 1,980円（本体）+税
ISBN 978-4-7741-7320-7



養成読本編集部 編
B5判・164ページ
定価 1,980円（本体）+税
ISBN 978-4-7741-7057-2

Vyatta仮想ルータ活用ガイド
松本直人、さくらインターネット研究所（日本Vyattaユーザー会）著
定価 3,300円+税　ISBN 978-4-7741-6553-0

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

Javaエンジニア養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6931-6

JavaScriptエンジニア養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6797-8

WordPressプロフェッショナル養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6787-9

サーバ／インフラエンジニア養成読本 ログ収集～可視化編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6983-5

フロントエンドエンジニア養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6578-3

PHPライブラリ&サンプル実践活用［厳選100］
WINGSプロジェクト 著
定価 2,480円+税　ISBN 978-4-7741-6566-0

アドテクノロジー　プロフェッショナル養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6429-8

【改訂新版】サーバ／インフラエンジニア養成読本 管理／監視編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6424-3

【改訂新版】サーバ／インフラエンジニア養成読本 仮想化活用編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6425-0

February 2016

NO.52

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp
榎 真治　ENOKI Shinji　enoki-s@imail.plala.or.jp

地域は違えど悩みは同じ？　コミュニティ運営を考える

最近、jusでは「ITコミュニティの運営を考える」と題するセッションを各地で行っています。今回はjus研究会の福岡大会と徳島大会の模様をお届けします。

jus研究会　福岡大会

■ITコミュニティの運営を考える
【講師】江頭 竜二（baserCMSユーザー会）、
谷崎 文義（QUNOG）、
榎 真治（日本UNIXユーザ会／LibreOffice日本語チーム）、
法林 浩之（日本UNIXユーザ会）
【日時】2015年10月3日（土）16:15～17:00
【会場】九州産業大学 2号館 3階 2E309教室

福岡大会は、九州在住の、ITコミュニティ運営で活躍されている江頭さん、谷崎さんをお招きしました。オープンソースカンファレンス2015 Fukuokaの中での開催で、参加者は25名でした。

まず、QUNOGの谷崎さんから、「会則などは必要か？」とのお題が出されました。QUNOGは九州と沖縄のネットワークオペレーターのグループで、最近立ち上げられて会合はしているものの組織はなく、今後どうするかを考えているそうです。baserCMSユーザー会は会則のない緩い集まりです。CMS（Contents Management System）の開発を支えるNPOが別途あり、そちらでは定款を定め、協賛金も集めているそうです。NPOでは年会費があるので、非アクティブな人は会費未払いとなるため、更新されないそうです。LibreOffice日本語チームは規則が定められており、重要な意思決定では投票も行います。最初に規則を厳しくし過ぎたために、規則を改正したくても、それに必要な投票数が集まらず改正できないという事態に陥りそうになりました。jusからは、会則と年会費があること、総会開催は手間がかかること、年会費の払い忘れが発生するケースもあることが紹介されました。

続いて筆者（榎）から、「コミュニティからお金を出す際のルールはどのようにしているか」を伺いました。LibreOffice日本語チーム自体では少額のお金はありますが、お金を集めたり扱ったりするしくみが十分に整えられてはいません。みんなが納得できるところで支出しようとすると意外に難しいと感じています。baserCMSユーザー会では、会の通帳を作って管理し、3ヵ月ごとに開催される定例会で通帳の残高を確認しているそうです。今ではNPOで協賛金を集めて年間予算を組んでおり、チラシ代金や交通費に支出しているとのこと。jusは会計を担当する事務局長が置かれ、税務署とのやりとりも行っています。ほかに会計をできる幹事がおらず、ずっと留任になっているという課題があること、承認は月1回の幹事会で行われていることも説明がありました。

3つめのお題は、江頭さんから出された「メンバーが各地に別れている中で、スタッフ・ミーティングをどうしていますか？」というものでした。baserCMSユーザー会はGoogleハングアウトで音声のミーティングをしているそうです。QUNOGはメーリングリストで議論するとのこと。LibreOffice日本語チームは

Googleハングアウトでのチャット、jusはSkypeでのチャットです。法林さんからSlackが増えてきたという話があり、baserCMSユーザー会からも、開発ではSlackが使われているという報告がありました。jusでは、以前は月に1回、金曜の夜にオフラインで集まっていましたが、遠方からの移動がたいへんなためにオンラインミーティングに移行しました。今は、3ヵ月に1回程度オフラインで行っています。

4つめのお題は法林さんからで、「福岡と東京との違い」でした。飲み会に関しては「メニューが違う」、「東京の人は終電で帰るが、福岡の人は最初からタクシー覚悟」という意見が出ました。コミュニティ関連では、九州には複数のCMSコミュニティが存在するが、メンバーがかぶっていることが多いそうです。一方、LibreOfficeは九州のコミュニティは残念ながらまだ立ち上がりきれていない状況です。福岡は熱いイメージがあるという話題も出ました。

会場からは、「コミュニティに興味はあるがなかなか入れない人に対してはどうするか？」というお題も出ました。オープンソースカンファレンスや勉強会などで話しかけるようにする、など直接的な交流を図るというコメントが多かったです。

かなり具体的な内容で、すぐに参考にできそうな話題が多かったように思います。会場からも活発に質問が出るなど、短い時間でしたが盛り上がった研究会でした。

jus研究会　徳島大会

■ITコミュニティの運営を考える
【講師】辰濱 健一（tokushima.app）、
野原 直一（tokushima.rb）、
榎 真治（日本UNIXユーザ会／LibreOffice日本語チーム）、
法林 浩之（日本UNIXユーザ会）
【日時】2015年11月14日（土）16:15～17:00
【会場】とくぎんトモニプラザ 3階 大会議室

2年半ぶりに徳島での研究会が実現しました。こちらも地元で活動されている辰濱さんと野原さんを講師にお迎えし、オープンソースカンファレンス2015 Tokushimaの中で開催しました。参加者は28人でした。

榎さんからのお題は「徳島でもLibreOfficeの勉強会をやっているが、地元の人々になかなかリーチしないのはなぜか」で、これに対しては「徳島では業務以外に勉強する場があることが、まだ知られていないのではないか」というコメントがありました。

辰濱さんからは「定期的に開催するにはどうすれば良いか」というお題が提示され、回答として「開催日を毎月第4土曜日などのように固定する」、「常連参加者に運営メンバー入りしてもらう」などが示されました。

野原さんからは「世代の違う仲間を増やす方法」というお題が出されました。これは多くのコミュニティも課題と感じている難しいテーマです。「若い世代の人を増やしたいなら、学校向けの取り組みを増やすのが良いだろう」とか、「活動を長期に渡り継続することで少しずつ広がっていく」という意見がありました。

筆者（法林）からは、徳島県の「VS 東京」というキャンペーンに引っ掛けて「ITコミュニティの世界でも、VS 東京はあるのか」というお題を出しました。これに対しては「VSではなく、東京で得たものをどうやって徳島に伝えるかという意味で意識している」、「ネットが発達しても、ネット越しに得られる情報とオフラインで得られる情報には差がある」などのコメントが得られました。

最後に参加者から「成果物を残すことに対する苦労はないか」という質問があり、これに対して「成果物を残すことを目的とする集会と、コミュニティを広げることを目的とする集会を分けて開催したほうが良い」という回答がありました。

これまでに開催した同テーマのセッションとは異なる話題が提示され、コミュニティの運営についてさらに知見を増やすことができてよかったです。今後も継続的にこのテーマのセッションを実施したいと思います。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

●Hack For Japanスタッフ
及川 卓也　OIKAWA Takuya
Twitter @takoratta
佐伯 幸治　SAEKI koji
Twitter @widesilverz
鎌田 篤慎　KAMATA Shigenori
Twitter @4niruddha
高橋 憲一　TAKAHASHI Kenichi
Twitter @ken1_taka

第50回

第3回 IT×災害会議で考えた、エンジニアができる貢献とは

2015年11月21日に、3回目となるIT×災害会議が開催されました。防災・減災への取り組みを考えるすべての人々がゆるくつながれる、そんな場になることで、各々の活動を継続する力に変えてもらいたいと考えています。

第3回 IT×災害会議開催！

2013年開催のIT×災害会議「つながる」、2014年開催の第2回 IT×災害会議「つながる×うごく」、これらに続き、第3回目のIT×災害会議が開催されました。今回は「つながり×ひろげる」がテーマです。災害に関していろいろな想いを持ち、いろいろな活動に携わっている方々がゆるくつながり、ひろがること、さらには「新しい化学反応を生み出す」ことで新たな活動へ結びつけられるよう、さまざまなプログラムが組まれました。

午前中には、初めて会議に参加する方に向けたオリエンテーションや名刺交換タイムを設けて、より積極的に会議に参加してもらう環境づくりを行いました。その後、会議から派生したプロジェクトである「IT DART」「減災インフォ」「減災ソフトウェア開発に関わる一日会議」についての活動紹介、東日本大震災における「ふんばろう東日本支援プロジェクト」の物資支援についての話、9月に起こった関東・東北豪雨災害での支援活動の話、災害が起こったときの心構え、などといったプログラムで、午前の全体セッションを終えました。

全日本芋煮会同好会と炊き出し訓練

ランチの時間には第1回でも好評だった芋煮会を、災害時を想定した炊き出し訓練として実施しました（**写真1**）。全日本芋煮会同好会の皆さんの協力を得て、炊き出し訓練に先立って同好会代表の黒沼篤さんより芋煮会の意義をお話いただきました。

東北地方の風習として一般的に行われている芋煮会は、参加者が材料を持ち寄り、自分達で調理し、分け合います。この作業の流れの中にコミュニケーションの重要な部分が集約されているということに、東日本大震災がきっかけで気付いたそうです。そこで復興支援の一環になればと思い、人と人をつなぐ芋煮会を通して地域に元気を取り戻し、ひいては日本を元気にしていくという気持ちで活動されているとのことでした。

また、シンプルな手順と材料で、ゴミも少なく衛生的に作れる芋煮はコミュニケーションを深めるだけでなく、災害時の炊き出しという観点から見ても有用な手段と言えます。今回の炊き出しでは山形村山風の牛肉を使った芋煮に加え、その出汁を使ったカレーうどんも提供され、非常に美味しく、楽しい炊き出し訓練となりました。

午後のセッション

午後は個別セッションとして18のセッションを用意しました。大きなくくりとしては「支援システ

◆写真1　炊き出し訓練の様子

ムとしくみ」「国・自治体・市民のIT連携」「災害時のTwitter活用」「災害支援ツールの体験」「IT支援活動の現場から」「エンジニアができる貢献」というグループ分けになりました。それぞれのテーマについて学んだり、議論したり、体験できるように、講義形式だけでなくワークショップ・ライトニングトーク・アンカンファレンスなど多様な方法で参加できるよう工夫されていました。

実際の会議の様子はレポートなどにまとめられる予定ですので、詳しく知りたい方は第3回 IT×災害会議のサイト[注1]からご覧ください。こちらの誌面では読者の皆様の関心が深いであろう「エンジニアができる貢献」について詳しくお伝えします。

エンジニアができる貢献

IT×災害コミュニティは、ITという冠が付いているにもかかわらず、実際に手を動かすエンジニアが少なく、「IT×災害[注2]」も「情報支援レスキュー隊[注3]」(IT DART)も「減災インフォ[注4]」も、そのサイトの構築と運用はごく数人のエンジニアによって行われています。技術偏重にならないように、ITでも“I”、すなわち情報を大事にしようとスタッフ間でも話しているのですが、それにしてもエンジニアの参加が少ないので、どうにかしようと企画されたのが、このセッションでした。

セッション1：技術者・研究者と社会貢献

エンジニアに興味を持ってもらえるように、今までの防災や減災というテーマでは登壇しないような人に話してもらおうということで、思い浮かんだのが“村上総裁”こと村上福之さんです。国会議員にFAXで意見を送信できるJapan Changerというサービスや、Twitterなどで募金を募るソーシャル募金といった社会貢献活動を行っていることでも知られています。また一方、研究者は社会貢献についてどのように考えているのかという観点から、今回会場を提供いただき、共催にもなっていただいた統計数理研究所の教授である丸山宏さんにも登壇いただきました。

まず、村上総裁からは「ぼくと、貢献して魔法ギークになってよ！」と題して話していただきました。Japan Changerはテレビでも紹介されるほど世間の注目を集めたのですが、通信費などで結局は持ち出し(赤字)だったことなどが明かされました。また、ソーシャル募金ではPaypalを利用して募金を集めていたのですが、Paypalが吐き出すコードをそのまま使っており、開発らしい開発はしていないそうです。ソーシャル募金は一般の募金で手数料が多いことに疑問を持ったことから始まったとのことで、このような活動の目的は“技術による手続きの中抜き”だそうです。

次に、丸山教授による「研究者・技術者の社会貢献」についてのお話では、19世紀まで科学は主要な職業には成り得ず、公共による支援が必要なものだったが、現代ではオープンサイエンスやオープンソースで民主化が起きている(Civic Science)ということが解説されました。

お二人による個別の話の後は、筆者(及川)がモデレーターとなり、会場の参加者も加わってパネルディスカッションが行われました。やはりメインになったのは、エンジニアや研究者が社会貢献的な活動にかかわるモチベーションについてです。マズローの欲求5段階説も持ちだされ、基本的な生活が充足された状態で余裕がないと、やはりボランティアはできないのではないかという意見も出ました。しかし、金銭的な満足だけが人間の行動を決めるものではないだろうと村上総裁は言っていました。また丸山教授が、研究者はレピュテーション(世評、評判)により動機づけられることが多いとお話しされたのに対して、村上総裁は、研究者として極めれば極めるほどその研究は世間一般からは理解されないくらい専門性が高いものとなり、レピュテーションは得られなくなるのではないかと質問しました。丸山教授は、全世界の研究者を見れば、同じ分野の研究をしている人は必ずいるのではと回答されていました。パネルディスカッションを通じて、筆者が

注1　http://2015.itxsaigai.org/
注2　http://www.itxsaigai.org
注3　http://itdart.org
注4　http://www.gensaiinfo.com

一番心に残ったのは村上総裁が言われた、「なぜこのような活動をするかは、Linuxを作ったライナス・トーバルズが言った『Just for Fun』。つまり、楽しめるから」という言葉でした。このような活動は使命感や時にはつい義務感で行いがちですが、やはりやっている本人が楽しめているかというのが一番大事でないかと思います。

セッション2：ライトニングトーク

2つめのセッションでは、実際にIT×災害として活用されている事案や、活用できそうなトピックを拾い上げる目的で、技術寄りの内容でのライトニングトークが行われました。

●今夜から快眠生活！ 低コストで非常通知受信端末を作った話

➡ ゲヒルン 石森大貴さん

「気象庁、総務省、IIJ緊急地震速報を防災情報配信システムで受けるインフラを作り、災害時にも強いSMSで配信するようにしました。しかし、受信端末としてスマホを使うとほかのアプリの通知がうるさいので、非常通知だけを受け取れるようキッズケータイを選択しました。電池も2週間持ちます。」

●宇宙からデータで探す災害

➡ 下農淳司（@himorin）さん

「宇宙から衛星を使ったリモートセンシングによるデータ活用で、波長ごとにその特性により得られる情報が異なることを活かして、さまざまな情報を得ることができます。JAXAの地球観測センターのデータ提供サイト[注5]では、実際にさまざまな衛星の観測データを取得できます。」

●人工知能の災害対応への活用 ——ソーシャル分析から遺伝アルゴリズムまで

➡ 村上明子さん

「発災期のデータ分析において、具体的な位置を示すジオタグは付加されていなくても、たとえば『XX橋で水位が上がっている』などとつぶやいている人が多いため、それを使って市民が怖いと思っている場所を推定することができます。さらに遺伝的アルゴリズムで避難シミュレーションを行い、車で出るべきかそうでないかを判定し、渋滞による避難の遅れを避けることを考えるといったこともできます。」

●減災インフォにおけるBigQueryを利用した自治体ツイートの収集

➡ 森下泰光さん

「2014年10月のHack For Japanのハッカソンをきっかけに減災インフォのコミュニティにかかわり、情報収集のしくみの開発を担当しています。TwitterのstreamAPIで情報を取得し、fluentdでBigQueryに転送し、災害情報を抽出するしくみを作成しました（写真2）。」

●災害IT支援ネットワークでのツールの裏側

➡ 柴田哲史さん

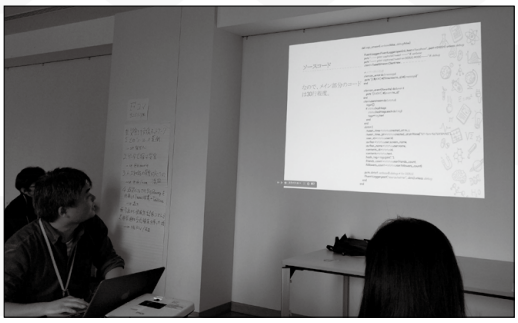
「常総などの災害ボランティアセンターでIT支援をしてきました。災害発生後は問い合わせが殺到するのですが、公式サイトで『よくある質問』の内容を充実させると電話の数は半分まで減らせます。今後やりたいこととして、『手書きで送られてくる被災者ニーズのデータ化が簡単に違和感なくできないか』、『数千名にもなるボランティア受け付けをスムーズにしたい』、『数十台規模の送迎バスが今どこにいるか知りたい（Uberのようにさっと見えるように）』ということを考えています。また、活動していくうえで、『最新技術の導入を目的にしてはいけない』、『現場に負担をかけない』、『普段使用していないものは現場では使えない』ということに気をつけなければいけません。」

セッション3：アンカンファレンス

3つめのセッションではそれまでの話を踏まえて、「エンジニアにいかに参加してもらうか」「災害とデータ」「ボランティア活動サポート」の3つのテーマに分かれて議論が交わされました。

注5　http://www.eorc.jaxa.jp/about/distribution/

◆写真2　エンジニアの集まるセッションらしくコードを解説する場面も



●エンジニアにいかに参加してもらうか

1つは、この会議の参加者にさえ、IT×災害やその派生プロジェクトのサイトや、さまざまなITツールを開発しているのがわずか数人だということが知られていなかった反省を踏まえて、もっと活動を可視化していくというアイデアです。いわゆる「中の人」は別に技術的にすごいことを駆使しているのではなく、サイトはWordPressだったり、Twitter Bootstrapをベタに使ったりしていたわけなので、本当は活動に加われる人はもっと多いはずです。活動の内容や時には窮状も積極的に表に出していくことが大事という指摘がありました。

もう1つが、自分達のイベントへの参加を呼びかけるだけではなく、逆に自分達から各種イベントや勉強会などに参加し、そこで呼びかけるのが良いのではないかというアイデアです。

そして最後が、活動をもっとPRしていくというアイデアです。エンジニアとしても「かっこいい」ことだと思われるように、可能ならばメディアなどと組んで積極的にアピールしていくことを検討することになりました。

ここでの提案を受けて、すでにいくつか動き始めています。たとえば、今までは今いる人達だけで解決を図ろうとしていたのですが、もっと状況をオープンにし、GitHubのリポジトリにIssueを立てて、「誰か拾ってください[注6]」とするのはどうだろうなどとも話し合っています。

そのほか、「災害とデータ」をテーマに話し合ったグループからは必要なこととして、データの収集と標準化、既存データの利用と公開の推進、リスクと責任追及の軽減、データ利用の訓練、といったことが挙げられました。「ボランティア活動サポート」について話し合ったグループからは、ボランティア受け付け時の集中化を解決する案として、スマホを活用してColorSync[注7]を使うことや、サーバ運用資金などの捻出方法として、カンパも良いが企業のサポートも得られると助かる、といった話が出されました。

第3回会議のまとめと次回開催予告

「つながり×ひろげる」という今回のテーマは、これまでの会議では東日本大震災を軸としてそれぞれ活動していた支援団体同士がつながり、そこから動きはじめた多くの活動を、さらに世代や地域を越えて拡げていくというものでした。活動内容の報告から始まるアンカンファレンスを軸にしていたこれまでと比べ、個別のセッションに分かれてのセミナーやワークショップといったアンカンファレンス以外の試みによって、これまで「IT×災害」会議に参加したことのない人達を惹きつける結果にもつながりました。

それぞれのセッションで行われたディスカッションや四面会議などのワークショップから導かれた、災害時の支援活動に関する意見なども、会議の最後に各セッションの代表者からそれぞれ発表してもらいました。そうした発表の内容からも、参加した人達の間に次のアクションにつながるものを感じ取れ、つながり、うごき、ひろがっていく様子がうかがえました。そして会の最後に、防災や減災の取り組みはいかに継続していくかが大切だという話を結びの言葉とし、次回第4回「IT×災害」会議の開催が2016年11月と予告され、会議は閉会となりました。SD

注6　さらには、基本的なフレームワークだけインストールしたリポジトリを作って、Pull Requestを期待しても良いかもしれません。

注7　http://about.peatix.com/colorsync.html

温故知新
IT むかしばなし

第51回

Pascal～プログラミング教育に最適な言語～

速水 祐（HAYAMI You）　http://zob.club/　Twitter：@yyhayami

はじめに

将来のICT社会を担う、子どもたちに対するプログラミング教育が大きく注目されています。プログラミング教育において、初めて触れる言語は、その後のプログラミング技術の進歩に大きく影響を与えます。1980年代前半、最初に使用する言語はパソコンではBASICだったと思います。当時、次に使う言語としてBASICより学びやすく、より正確に動作するプログラムを作成できるということでプログラミング言語「Pascal[注1]」が注目を浴びていました。今回は、そのPascal言語についてです。

Pascalとは

1970年にチューリッヒ工科大学のニクラウス・ヴィルト教授によって教育用プログラミング言語として開発され、70年代後半にはパソコンで動作するUCSD Pascal（カリフォルニア大学サンディエゴ校）が登場しました。UCSD Pascalは、Pマシンという仮想マシンのためのPコードを一度作成し、そのPコードをマシン語に変換して実行します。この2ステージのコンパイル作業は当時のパソコンの能力に合っており、異なるCPUのパソコンにも容易に移植できるものでした。APPLE IIで動作するUCSD Pascalは大いに普及し、その膨大で複雑な処理を扱える利点によりRPGゲームの原点と言われるウィザードリィなども作成されました。

Pascalの文法

Pascalは構造化言語であるため、当時BASICに慣れ親しんだユーザには、言語仕様の違いに戸惑いがありました。しかし、実際プログラムすると、ソースプログラムの見通しがよくなり、バグが少ない正確なプログラミングができたのです。

筆者が最初に使ったPascalは，シャープMZ-2000用のカセットテープによりロードするPascalインタプリタでした。インタプリタですので実行速度は遅いのですが、ソースリストを打ち込んで即実行の環境は言語の学習には適していました。BASIC言語と比べて言語仕様の違いで戸惑った点は次の4つです。

①代入には":="、比較等号には"="を使う

BASICは、両方とも"="、C言語では"="と"=="でありバグの発生しやすい原因の1つです。

②"begin"、"end"で囲むブロック構造でプログラミングを行う

C言語では"{"、"}"で囲む形で同様ですが、BASICにはブロック構造の概念がありませんでした。

③行番号がなく、goto文もない

一般的なPascalには、ブロックからの脱出のためのgoto文がありますが、MZのPascalにはgoto文がありません。BASICではgoto文を駆使してプログラミングを行うため、初めはかなり違和感がありました。

④定数・変数宣言、手続き・関数の宣言は必ず使用前に行う

定数や変数を必ず定義しなければならない点は忘れがちで、あとから利用することになった変数を前に戻って書き加える作

注1）機械式計算機を発明した、17世紀のフランスの哲学者ブレーズ・パスカルにちなんで命名されています。気圧の単位「ヘクト・パスカル」も同人物が発見した「パスカルの原理」から。

業は面倒だった記憶があります。

また、BASICでは行番号順にメインルーチン、サブルーチンと作るのに慣れていたので、メインルーチンが最後にあるPascalはわかりにくいと思ったこともありました。しかし、これらはプログラムのミスを少なくし、コンパイラの効率を高めるための構文の基本であり、教育用の言語としての優れた設計だとわかり、感心しました。

アルゴリズムを理解して記述するには、非常に適しており、『コンピュータアルゴリズム辞典[注2]』ではPascalによって、多くのアルゴリズムが的確に理解しやすく解説されています。その後、同著者によるC言語とJavaによるコンピュータアルゴリズム辞典が出版されましたが、最初のPascal版が最もわかりやすいと思います。

プログラムを組んで感動したのは、自分を自分で呼び出す再帰呼び出しにより、記述が簡単にできることでした。ハノイの塔[注3]の解法プログラムを**リスト1**に示します。この短いプログラムで複雑なパズルが簡単に解けるのです。

Turbo Pascal

Pascalの衝撃といえば、1983年に発売されたTurbo Pascal[注4]だと思います。CP/M上で動作するTurbo Pascalは、当時の高級スポーツカーに搭載されたターボチャージャーの名を得て、高速なコンパイラ、フルスクリーンエディタを含む使いやすい統合開発環境を提供しました。当時のCP/Mのフルスクリーンエディタであったワードスターの編集操作用キーボードショートカットをそのまま利用できたことでエディタ操作の学習もできました。

筆者は、NEC PC-8801と富士通FM-7+Z80ボード上のCP/MでTurbo Pascalを動かしていました。とくにFM-7+Z80カードは高速な文字表示とスクロールにより、プログラミング学習開発環境としては最適なものだと感じていました。

Pascalの開発者のヴィルト教授に計算機科学を学んだフィリップ・カーンは、Pascalのすばらしさをアンダース・ヘルスバーグに説き、彼はその天才性を発揮しオールアセンブラで高速なワンパスコンパイラを開発したのです。Turbo Pascalは、その後MS-DOS版、Windows版として機能が追加されていきました。そして言語仕様を大きく拡張したObject PascalであるDelphiに発展し、Windowsプログラミング環境では最適なものと考えられたのですが……。その後、アンダース・ヘルスバーグは、Microsoftに移籍し、.NET FrameworkとC#のチーフアーキテクトとして活躍を続けました。

おわりに

Pascalには、アプリケーションを本格的に記述するには、問題も多くありますが[注5]、教育用プログラミング言語としてみたとき、1つの選択肢になると思います。現在、子どもたちにビジュアルプログラミングによる教育が行われ始めましたが、学習の発展には疑問が残ります。リストの再帰呼び出しの内容を理解できたときの感動を伝えられるような、プログラミング教育が行われてほしいものです。SD

注2) 『コンピュータアルゴリズム辞典』、奥村晴彦著、技術評論社、1987年

注3) 左端の杭のすべての円盤を次のルールで右端の杭に移動させれば完成(**図1**)。●円盤を一回に一枚ずつどれかの杭に移動させる。●小さな円盤の上に大きな円盤を乗せることはできない。**リスト1**は、4枚の円盤を移動するものです。Move関数：第1引数は円盤の総枚数、第2引数は初めの杭の番号、第3引数は移動すべき杭の番号。

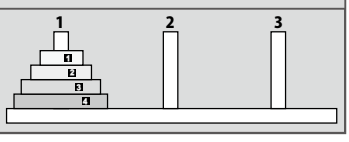
注4) 機能は限定的だがTurbo Pascal的な動作をWeb上で実現。http://www.teamten.com/lawrence/projects/turbo_pascal_compiler/（**リスト1**程度のプログラムを実行するには最適です）

注5) Pascalの批判としてC言語の開発者のカーニハンによる1981年の論文「Why Pascal is Not My Favorite Programming Language」などがあります。http://www.lysator.liu.se/c/bwk-on-pascal.html

▼リスト1　Pascalで書かれた4枚の円盤を動かすハノイの塔のプログラム

```
program Hanoi;
    procedure Move(N, A, B: integer);
    begin
        if N > 1 then Move(N - 1, A, 6 - A - B);
        writeln('Move disk ', N, ' from ', A, ' to ', B);
        if N > 1 then Move(N - 1, 6 - A - B, B);
    end;

begin
    Move(4, 1, 3)
end.
```

▶図1



開発のボトルネックはどこだ?

迷えるマネージャのための プロジェクト管理ツール再入門

第11回 クラウド時代だからこそIT運用部門の負担が増大! JIRA Service Deskで改善しよう

Author リックソフト㈱ 樋口 晃(ひぐち あきら)、南澤 華代(みなみさわ はなよ)、大塚 和彦(おおつか かずひこ)

ITサービス運用者の叫び

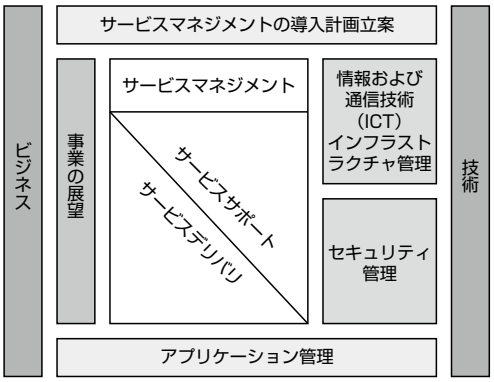
クラウドサービスに移行したからといって、ITサービスの運用が楽になるわけではありません。むしろ運用工数は増えていると感じる担当者は多いのではないでしょうか。

IaaSの場合、サーバ調達が早くなり、可用性などはサービス側に任せることもできます。SaaSの場合は、契約すればスグにサービスを使えるようになります。簡単に導入できるというメリットは、裏を返せば、運用担当者のサポート対象を増加させているということに気づいているでしょうか。クラウドサービスを利用する場合でも、「サービスサポート」や「サービスデリバリ」をきちんと考えることは必要です。

「サービスサポート」「サービスデリバリ」は、国際的なフレームワークであるITILで「ITサービスマネジメント(以下、ITSM)」として定義されています。クラウド時代にそれらを支えるのはIT運用部門です。クラウドサービスの利用が増加すればするほど、IT運用部門において「問い合わせ管理」「FAQの整備」「運用分析」を行うためのITSM導入は急務になると言わざるをえません。

ITサービス運用者の叫びが悲鳴に変わる前に、ITSM導入の一手を講じていただきたいと思います。

▼図1 ITILの構成要素



失敗しないITサービスマネジメントとは

ITSMは、ITILの構成要素(**図1**)のうち、「サービスマネジメント」部分を担います。ITSMの導入に失敗するケースで多いのは、いきなり難易度の高い要素(「構成管理」「変更管理」「CMDB:Configuration Management Database」など)に導入することだと聞いています。

ソースコードの構成管理やバージョン管理(版管理)などはプロセスと連動してスムーズに行うことができますが、OS、ソフトウェア、メモリ、ディスクなどの細かいインベントリ情報をCMDBに登録してしまうと、メンテナンスがたいへんになってしまいます。

ITSMの導入に失敗しないために「インシデント管理」「SLA:Service Level Agreement」「報告・分析レポート」からスタートすることをお勧めします。

表計算ソフトによるインシデント管理の問題

さまざまな企業で「インシデント管理」「SLA」「報告・分析レポート」を表計算ソフトで行っていると聞きます。表計算ソフトでの管理を否定するわけではありませんが、そのような企業の多くは、次のような問題を抱えています。

●転記(コピー&ペースト)が多い

メールでの問い合わせの内容や、障害などのインシデントの発生状況を表計算ソフトへ転記し、さらに報告書へも記入するなど転記が多く、業務効率がよくありません。

●情報共有ができない

インシデントがタイムリーに共有されないため、同じようなインシデントにそのつど最初から対応することになります。

●分析がスムーズにできない

表計算ソフトのグラフ機能などを利用したインシデントの推移やカテゴリの分析には、マクロやグラフ範囲など手作業を要するところが多くあります。

●レポート作成の手間

お客様や上司への報告のため、経緯などを示した報告書を表計算のシートとは別に作成しています。

◆　◆　◆

これらの問題点はITSMツールを導入することで解決できます。

市場にあるITSMツールはオンプレミス型、クラウド型(SaaSなど)といったものが提供されていますが、今回は2015年にBest ITSM Pro Productsにも選ばれた「JIRA Service Desk」を紹介します。

JIRA Service Deskとは

JIRA Service Deskは、インシデント管理、SLA、ワークフロー、ダッシュボードなど、さまざまな機能を提供するソフトウェアです。利用者は直感的にわかりやすいユーザインターフェースで問い合わせ(リクエスト)ができます。問い合わせ窓口を1つにすることで、担当者もサポート業務を進めやすくなります。

利用者からのリクエスト(タスク)は、カスタマイズ可能なサービス水準合意(SLA)、ライブキュー、リアルタイムレポートなどのサービスデスク管理ツールで効率良く仕分けされ、IT運用部門やソフトウェア開発部門は、慣れ親しんだ強力なタスク管理ツールのJIRAを使い、問題解決に向かってスムーズに作業を進めることができます。

JIRA Service Deskの特長

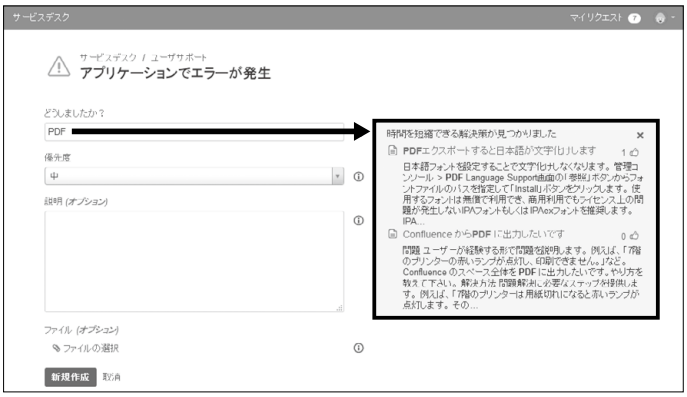
JIRA Service Deskには次のような特長があります。

●わかりやすいインターフェース

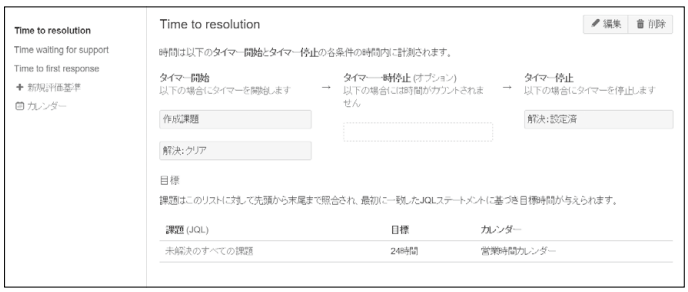
わかりやすいメニュー画面が簡単に作れます。JIRAの各項目をユーザにわかりやすい名前で、説明文とともに表示できます(**図2**)。

▼図2　わかりやすい項目名と説明文を表示

▼図3　関連するFAQをリアルタイム検索



▼図4　課題解決までのSLA制限登録



▼図5　各課題の状況をSLAと比較して表示

▼図6　SLAの達成状況を示すグラフ

●FAQと連動した問い合わせフォーム(リアルタイム検索)

入力された文字列でFAQを検索し、関連するものをユーザに提示することができるため、問い合わせ件数を減らす効果があります(**図3**)。

●SLA

解決までの時間やユーザを待たせる時間について、SLA制限を登録できます(**図4**)。営業カレンダーや営業時間を設定できますので、時間は営業日で計算されます(例:24時間 → 1日の営業時間が8時間の場合、3営業日)。

●状況の表示

各課題の状況を、緑はOK、赤はNGといったようにSLAと比較して表示します(**図5**)。

●レポーティング

SLAの達成状況をグラフで確認できます(**図6**)。

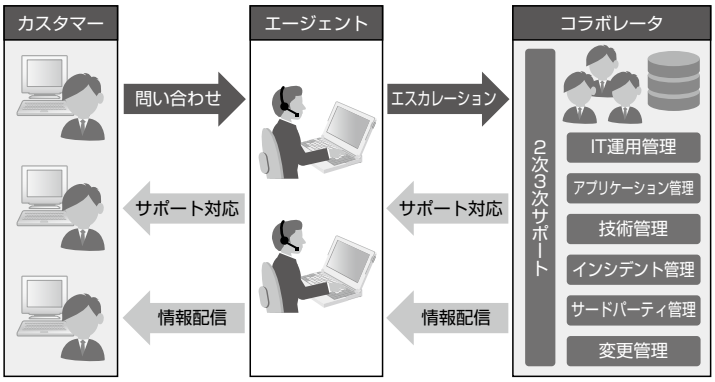
●JIRA製品との統合

JIRA Service Deskは、ビジネスプロセス、課題管理、プロジェクト管理、バグトラッキングとして多く利用されているJIRAの基本機能を利用できます。JIRAの機能やアドオンと組み合わせることで、アカウント管理、チャット、ナレッジ共有、ストレージ、文書管理などと連携できます。APIも公開されており、アトラシアン製品以外との連携も可能です。

●導入しやすいライセンス形態

JIRA Service Deskのライセンスは、問い合わせの窓口であるエージェントのみが課金対象となります(**図7**)。したがって、全アカウント数分の契約が必要なソフトウェアや、ユーザ数無制限であってもサポート費用が高いソフトウェアと比べて低コストで導入できます。

▼図7　課金対象となるのはエージェントのみ



※コラボレータはJIRA Softwareが必要

航空業界でも活用されるJIRA

JIRAを構成管理システムとして活用している事例として、1日約800便、搭乗者10万人以上の国内線の予約・発券から搭乗までのエアライン業務を扱う中枢システムがあります。

このシステムでは、JIRAの導入によって次のような効果が見られました。

- 他社製品からJIRAへの移行でライセンスコストが約10分の1になった
- 誰が、いつ、何を変更したかという履歴管理(トラッキング)ができるようになった
- 利用者からの操作に関する問い合わせが少なく、スムーズに導入できた

このほかにも大手SNS企業のヘルプデスクなど多くの導入事例があります。

IT業務、ソフトウェア開発の統合業務基盤として

JIRA Service Deskを含むJIRA製品はDevOps(デブオプス)への活用もできます。DevOpsには、「測定・分析」「共有」「自動化」「コラボレーション(コミュニケーション)」の要素が必要です。

異なるカルチャーを持つ「開発部門」と「運用部門」が効率良く連携するために必要な機能を、JIRA製品を含むアトラシアン製品は持っています。

たとえば、運用部門からのユーザニーズをとらえた変更要求に対して、開発部門はソースコードを修正し、ビルド・デプロイを行います。それに対して運用部門がサポートを行う、サポートを行っていく中で新たなユーザニーズを把握していく。その情報をプロセスとして管理でき、履歴(トラッキング)まで行うことができます。このサイクルこそが継続的ITシステム運用の姿ではないでしょうか。

今後、IT部門やソフトウェア開発部門の統合業務基盤として、IT運用のフレームワークであるITILやDevOpsに対応することは必要な要素だと思われます。SD

日本だけでなく、アジア圏でもアトラシアン製品販売のトップエキスパートであるリックソフトのWebサイトでは、各アトラシアン製品の体験版を提供しているほか、アトラシアン製品専用のコミュニティも運営しています。まずはアクセスしてみては！

https://www.ricksoft.jp/

Atlassian Expertsの盾には2015年のアジアパシフィック市場においてトップセラーを証す刻印が……。

Mackerelではじめるサーバ管理

Writer 高谷 雄貴(こうや ゆうき) GMOペパボ㈱
Twitter @buty4649

第12回 Mackerel活用事例——GMOペパボの場合

今回は趣向を変えて、Mackerelのユーザ企業「GMOペパボ」のエンジニアに、開発の現場でMackerelをどのように活用しているのかを伺いました。クラウドへの移行に合わせてMackerelを利用するようになった同社の、Mackerel運用Tipsを紹介します。

クラウドに便利なMackerel

GMOペパボ技術部インフラグループの高谷と申します。

弊社では、『minne』や『カラーミー』といったASP事業と『ロリポップ』や『ヘテムル』などのホスティング事業の2つの業態のサービスを運営しています。オンプレミスでサーバの運用を行っていますが、今年4月からOpenStackを基盤としたプライベートクラウドの導入が開始され、ASP事業の各サービスはそちらに順次移行し始めています。監視システムには今まで、NagiosやMuninを使っていましたが、プライベートクラウドへの移行を契機に、Mackerelに置き換えるサービスが増えていきました。初期は監視やメトリック取得のみにしか使っていませんでしたが、利用者が増えるに従って、MackerelのAPIを駆使したさまざまな使い方が考案されました。そこで、今回は弊社でのMackerelの活用事例を紹介したいと思います。

fabricとの連携

弊社ではサーバの構成管理ツールとして、おもにPuppet[注1]を利用しています。一部のサービスでは、Puppetのデプロイにfabric[注2]を使用しています。fabricはParallel SSHのように使え、大量のサーバへの一括オペレーションが容易にできるので以前から使用していましたが、そのままデプロイツールとしても利用しています。

デプロイを行うときには必ず、デプロイ先のホストを指定する必要があります。今まではロールごとにホスト一覧を定義したファイルを用意し、それを利用してデプロイを行っていました。インフラがクラウドアーキテクチャに変わったことで、ホストの増減が著しくなってしまい、この方法では運用が間に合わなくなってしまいました。結果、ホストの追加漏れや削除忘れが多発してしまったのです。

そこで、新たに利用することになったのがMackerel[注3]です。fabricでは実行先ホストの管理にロールが使用でき、Mackerelもホストにロール付けが行えるため連携が容易だと考えました。そこで、MackerelのAPIからホスト情報を取得し、fabricへロール情報を受け渡す**リスト1**のようなfabricのタスクを追加しました。実際にweb/app/dbロールに所属するホストに対してデプロイを行う場合は、次のようなコマンドを実行します。

注1） URL https://puppetlabs.com
注2） URL https://get.fabric.io
注3） URL https://mackerel.io

▼リスト1　Mackerelとfabricの連携

```
import json, urllib2

def role(*roles):
  roledefs = {}
  service = "foo"

  if "MACKEREL_APIKEY" not in os.environ:
    abort(red("please set $MACKEREL_APIKEY"))

  url = "https://mackerel.io/api/v0/hosts.json?service=%s" % service
  headers = {"X-Api-Key": os.environ["MACKEREL_APIKEY"]}
  request = urllib2.Request(url, headers=headers)
  response= urllib2.urlopen(request)

  for entry in json.loads(response.read())["hosts"]:
    hostname = entry["name"]
    for role in entry["roles"][service]:
      if role not in roledefs:
        roledefs[role] = []
      roledefs[role].append(hostname)

  for r in roledefs.keys():
    roledefs[r].sort()

  env.roledefs = roledefs
  env.roles = roles
```

```
$ fab role:web,app,db deploy
```

これで、ホストが増減してもデプロイ漏れが発生することはなくなりました。

hostsファイルの生成

サーバの名前解決にはDNSを使いますが、LAN側のIPアドレスにも名前解決を行いたい場合があります。しかし、小規模なサービスの場合、DNSを構築し、運用するのは負担が大きい割にはあまりメリットがありません。そういった場合、hostsファイルに直接IPアドレスを書いてしまったほうが手っ取り早いこともあります。

このとき問題になるのがhostsファイルの更新です。サーバのIPが変わったとき、hostsの更新を忘れると間違ったアドレスへ通信してしまい、エラーとなってしまいます。そこで、Mackerelからホスト名とIPアドレスを取得し、定期的に更新するしくみを作ることにしました。**リスト2**は、弊社若手インフラエンジニアの@hfm[注4]が作ったhost-gen.rbです。このスクリプトをcronに登録し、定期的に実行しています。

Mackerel運用に役立つmalsh

“確認漏れ”を防ぐツール

弊社イケメンエンジニア@pyama86[注5]が作っているmalsh[注6]について紹介します。

Mackerelを運用していると、次のような問

注4）URL http://twitter.com/hfm
注5）URL https://twitter.com/pyama86
注6）URL https://github.com/pyama86/malsh

▼リスト2　hostsファイルを更新するRubyスクリプト

```
#!/usr/bin/env ruby

require 'net/http'
require 'uri'
require 'json'

service = 'foo'
url = URI.parse('https://mackerel.io/api/v0/hosts.json?service=#{service}')
https = Net::HTTP.new(url.host, url.port)
https.use_ssl = true

conf = '/etc/mackerel-agent/mackerel-agent.conf'
key = `grep '^apikey' #{conf} | sed -r 's/apikey¥¥s+=¥¥s+"(.*)"/¥¥1/'`
header = { 'X-Api-Key' => key }

res = https.get("#{url.path}?#{url.query}", header)
fail(res.body) unless res.is_a?(Net::HTTPOK)

ips_hosts = JSON.parse(res.body)['hosts'].map do |host|
  host['interfaces'].map do |ifc|
    case ifc['name']
    when 'eth0'
      "#{ifc['ipAddress']}¥t#{host['name']}"
    when 'eth1'
      "#{ifc['ipAddress']}¥t#{host['name'].gsub('jp','lan')}"
    end
  end
end

hosts = ""
File.open('/etc/hosts.base', 'r') do |hosts_base|
  hosts_base.each_line {|l| hosts << l }
end
hosts << "¥n# From mackerel.io¥n"
hosts << ips_hosts.flatten.join("¥n")

require 'optparse'
opt = ARGV.getopts("", "w", "write")
if opt['w'] or opt['write']
  require 'tempfile'
  Tempfile.open('hosts') do |f|
    f.puts hosts
    f.rewind

    FileUtils.cp f, '/etc/hosts'
  end
else
  puts hosts
end
```

題が発生します。

❶退役忘れのホストが存在し、無駄にライセンスを消費している
❷ホストがロールに紐付いていないために、リストから漏れてしまった

malshを使えば、このような問題を解決できます。利用方法は簡単で、MACKEREL_APIKEY環境変数にMackerelのAPIキーを登録して実行するだけです。次はインストールと実行例です。

```
$ gem install malsh
$ export MACKEREL_APIKEY=<API Key>

退役忘れのホスト一覧の検索
$ malsh retire

ロールに紐付いていないホストの検索
$ malsh maverick
```

退役は、過去5分間にロードアベレージのメトリックの投稿があるかどうかで判定しています。これは、Mackerelのホスト数のカウント方法が、「ホストのステータスにかかわらず、メトリック投稿APIにアクセスしたユニークなホスト数を計上している」という方法であるためです。メトリックの投稿がないということは、すなわち退役忘れのホストであるということになります。

▼図1　退役漏れホストのSlack通知

▼図2　ロールに紐付いていないホストのSlack通知

この例では、実行結果は標準出力に取得されますが、次のように環境変数を設定することで、Slackにも通知できます。

```
$ export SLACK_WEBHOOK=<Slack WebHook URI>
$ export SLACK_CHANNEL=<Slack Channel>
$ export SLACK_USER=<Slack User>
```

弊社では、Slack通知を有効にして退役忘れのホスト一覧(**図1**)とロールに紐付いていないホスト一覧(**図2**)を毎日通知しています。

特定条件によるホストの検索

malshには、前述した機能のほかに、ホストの検索機能があります。検索条件には次を指定できます。

・ホスト名
・過去7日間のCPUの最大使用率
・過去7日間のメモリの使用率

この機能を利用し、CPUやメモリの空き率が多いホストを検索することで、オーバースペックなホストを検知できます。プライベートクラウドとはいえ、コストは発生します。過剰なスペックだった場合、適切なスペックのVMに作りなおすことでコストを削減できます。

まとめ

今回はGMOペパボでのMackerelの活用事例を紹介させていただきました。

Mackerelは、使いやすいAPIが提供されていて、いろいろなことに簡単に応用できます。また、毎週新しい機能をリリースしているため、今後はもっと便利で使いやすいツールになっていくと期待しています。Mackerelを監視やメトリック取得だけに使うのはもったいないと思いますので、ぜひともMackerelを活用し、日々のシステム運用を楽にしましょう！SD

Software

ネオジャパン、オンプレミス型ビジネスチャットシステム「ChatLuck」を提供開始

㈱ネオジャパンは2015年12月3日、オンプレミス型ビジネスチャットシステム「ChatLuck」を提供開始した。

同製品は、メールに代わる新たな企業内メッセージング基盤となるWebチャットシステム。Webブラウザや、スマートフォン向け専用アプリから利用できる。プロジェクトやテーマごとに「ルーム」と呼ばれる専用の部屋を作り、メンバー間でリアルタイムな情報共有を行える。

同社はこれまでグループウェア「desknet's NEO」を開発／提供してきており、そのノウハウがChatLuckにも活かされている。具体的には、ファイル共有、タスク／スケジュール管理、アンケート収集／集計など、日々の業務と共同作業を効率化するための機能を搭載する。

また、個人間でチャットを行う「コンタクト」機能では、WebRTCを用いたビデオ通話／音声通話が可能なほか、PC画面の共有も可能となっている。

企業で安全に利用するために必要なセキュリティ機能としては、ユーザや組織の一括管理、社内外ユーザのアクセス権限管理、パスワードポリシーの適用、ファイルのダウンロード制限などの機能を備えている。

ライセンス形態は、買い切り型の2種類が用意されている。1つめは利用するユーザ数に応じた「ユーザーライセンス」で、100ユーザなら36万円から、1,000ユーザなら360万円から利用できる。もう1つは、最大10ルームまで作成できる「ルームライセンス」で、15万円で利用できる（価格はすべて税別。初年度サポート費を含む）。

▲ChatLuck（Webブラウザでの利用イメージ）

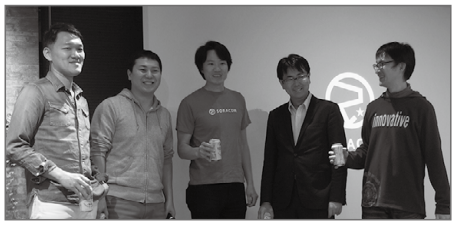
CONTACT
㈱ネオジャパン URL http://www.neo.co.jp

Report

ソラコム、「Developers Conference#0」開催

㈱ソラコムが発表したIoTプラットフォーム「SORACOM」のユーザグループが立ち上げられ、その第0回の会合が2015年12月11日、イベント＆コミュニティスペース「dots.」（東京都渋谷区）にて行われた。

イベントの最初、ユーザグループのキックオフでは、「ハードウェアをお持ちのメーカーさま、また持っていなくてもこんなハードを知っているよという方、どんどん紹介してほしい」「ソフトウェア、ハードウェアの相互の組み合わせで問題解決ができたらなと考えている」「大きな学びの場にしていきたい。ビジネスの話もどんどんしてほしい」などと今後の展望が語られた。



▲ユーザ会立ち上げメンバーと㈱ソラコムの玉川 憲氏（左から後藤 和貴氏、横田 聡氏、玉川 憲氏、竹之下 航洋氏、松下 享平氏）

次にソラコムから発表されたのは、SORACOMのアップデート情報。

- SORACOM Beam：「AWS IoTとの連携機能追加」「UDP→HTTP（S）のプロトコル変換が可能に」
- SORACOM Air：「メタデータサービス機能追加」「カスタムDND機能追加」
- イベントハンドラーが、ステータス／速度変更に対応
- 契約数・期間を事前に申請することで基本料金が安くなるプランの追加

その後は、次のような、SORACOMを活用した計11のライトニング・トークが行われた。

- SORACOM Airを操作するNode-RED FlowをAWS Lambdaで動かす
- SORACOM AirをつないだRaspberryPiをlittleBits＋Milkcocoaで遠隔OFFするボタンをつくった話

CONTACT
㈱ソラコム URL https://soracom.jp/

Topic

アカマイ・テクノロジーズ合同会社、2015年第3四半期の「インターネットの現状」レポートを発表

アカマイ・テクノロジーズ合同会社は2015年12月15日、世界中にある同社のエッジサーバから2015年第3四半期に収集したデータに基づいたレポートを発表した。そのうち、サイバー攻撃に関するレポートを紹介する。

2015年第3四半期のDDoS攻撃は合計1,510件。昨年の同期間と比べると、件数は2倍近く増加した一方、平均攻撃時間・平均ピーク帯域・平均ピークパケット数はすべて低下しており、“小粒の攻撃”が増えたと言える。また、HTTP GETやHTTP POSTを利用したアプリケーション層への攻撃から、UDPを使ったリフレクション攻撃といったインフラ層への攻撃件数が増えた。このうち、UDPリフレクション攻撃の手法としては、NTPやSSDPの脆弱性を使ったものが増えているという。そのほかのトピックは次のようなもの。

- DDoS代行業者の台頭。パブリッククラウドを使ってマシンパワーを稼いで攻撃に用いるケースも
- DDoS攻撃の発信元のランキングにおいて、中国やアメリカを退け、初めて英国がトップへ
- 攻撃先企業としては、ゲーム業界、ソフトウェア業界が依然として多く、金融業への攻撃は減少傾向

CONTACT
アカマイ・テクノロジーズ合同会社
URL https://www.akamai.com/jp/ja

Report

「Ruby biz Grand prix 2015」開催

2015年12月17日、Rubyを活用してビジネスで新たな価値を創造し、イノベーションを起こしたサービスや商品を表彰するグランプリ「Ruby biz Grand prix 2015」が開催された（主催：Ruby bizグランプリ実行委員会）。

表彰式には、溝口善兵衛島根県知事、まつもとゆきひろ氏らが参加した。国内外の30事例から大賞2点と特別賞3点、エンタープライズパイオニア賞2点が発表された。受賞企業とおもなプロダクトは次のとおり。

○大賞

トレジャーデータ㈱：fluentd、Embulk
㈱ユビレジ：iPadで使えるPOSレジシステム「ユビレジ」

○特別賞

HipByte：RubyでiOS、Androidアプリを開発できる「RUBYMOTION」
GMOペパボ㈱：ハンドメイドマーケット「minne」
㈱マネーフォワード：自動家計簿・資産管理サービス「MoneyForward」

▲トレジャーデータ㈱　古橋 貞之氏

▲㈱ユビレジ　木戸 啓太氏

○エンタープライズパイオニア賞

ベニックソリューション㈱：Rubyを活用した製造業向けシステム構築
㈱テクノプロジェクト：ベトナムにおける医療情報ネットワーク「Mame-NET」

大賞を受賞したトレジャーデータの古橋氏は「Rubyによって、世界中で開発されているプラグインを柔軟に取り込めるfluentdが開発できた」と語り、ユビレジの木戸氏は「Ruby言語の柔軟性が開発スピードに貢献した」と語った。授賞式の最後、審査員長を務めるまつもとゆきひろ氏は、「プログラミングは魔法などではなく実に人間的な活動で、プログラマが楽しんでコードを書くことが何よりも大事。Rubyは早いうちから、プログラミングする楽しさを提供してきた言語。これからも、Rubyを愛して使ってくれている人たちとともに未来を作っていきたい」と述べた。

▲審査委員、運営委員、受賞者のみなさん

CONTACT
Ruby biz Grand prix　URL http://rubybiz.jp

Topic

ファイア・アイ、2016年セキュリティ動向予測を発表

ファイア・アイ㈱は2015年12月21日、2016年のサイバーセキュリティ環境の変化についての説明会を実施した。説明会では、10点のセキュリティ予測が挙げられた。

①経営者レベルでのセキュリティへの取り組みが必須に
②取締役会のセキュリティ対策への理解の必要性
③ハッカーはクラウドに潜んでいる：攻撃者がクラウド上の仮想マシンをC&Cサーバとして利用するように
④より多くのものがハッカーの標的となる：IoTのスタートアップ企業の大量参入により、脆弱性を残したネットワーク製品が氾濫する。
⑤Appleへの攻撃に方向を変える攻撃者：Masque Attack（2014年）、XcodeGhost（2015年）に続き、Appleへのサイバー攻撃が増加する
⑥Security as a Serviceの高まり
⑦M&Aにおけるサーバ審査：財政状況とともに、セキュリティ対策のレベルが査定の対象に
⑧サイバー保険がビジネス基準に
⑨より多くの平和的なサーバ犯罪条約が締結される
⑩依然残るアトリビューション問題：サイバー犯罪の情報は秘匿性が高く、情報共有が進まない

CONTACT
ファイア・アイ㈱ URL https://www.fireeye.jp

Report

ギットハブ・ジャパン、「Swiftのオープンソース化とイノベーション」開催

ギットハブ・ジャパン合同会社は2015年12月18日、Apple Store銀座（東京都中央区）にてエンジニア向けイベント「Swiftのオープンソース化とイノベーション」を開催した。イベントには、GitHub社の共同創業者Scott Chacon氏、次世代モバイル向けデータベース「Realm」を提供するRealm社の岸川克己氏が登壇した。

○大企業がOSSに貢献――Scott Chacon氏

Chacon氏が語ったのは、OSSを取り巻く昨今の情勢。2014～2015年にかけて、Microsoft、Facebook、IBMが続々と自社のプロダクトの一部をOSS化していった。そして12月3日、AppleもSwiftのOSS化によって、この大きな流れに続いた。Swiftには現在までに500以上のプルリクエストが集まり、そのうち350近くがマージされるなど、GitHubでも1、2を争うほど開発が活発だ。また、AppleはSwiftという言語自体に加え、Swiftのこれからの方向性を議論する場「Swift Evolution」もOSS化した。これは、AppleのSwiftコアメンバーだけではなく、外部のユーザも開発の議論に参加できることを意味する。

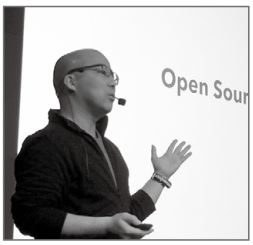
同じOSSでも、プロジェクトによってはコミッタが冷たい態度を取り、マージに対して消極的な場合があるが、Appleはまったく逆で、SwiftがOSSの中心になっていくのは間違いないだろう、とのこと。

○AppleのOSSへの貢献に、我々はどう応えるか――岸川克己氏

岸川氏は、当初SwiftがOSS化されるというニュースを聞いたとき、“どのようにOSS化されるのか”心配だったという。自社のページで圧縮ファイルがアップされ、ビルドの方法も説明されないような形でのOSSもあり得たからだ。結果から言うと、SwiftのOSS化は、これ以上ないほど歓迎される形で成された。非常にオープンな場である「GitHub」で、比較的ゆるく、利用しやすいライセンス「Apache Lisense 2.0」で公開されたからである。今回公開されたSwiftのリリースのうち、氏が注目しているのは、「Swift Core Libraries」「Swift Evolution」「Swift Package Manager」の3つ。このうち、Package Managerの公開は非常に重要で、コードを簡単に共有・再利用することが可能になり、言語の進化が加速する。

このように、AppleはSwiftのユーザコミュニティに対して最大限の開かれた姿勢を示した。氏は、この貢献に応える形で我々もコミュニティを一層盛り上げていきたいと語り、2016年3月2～4日に渋谷で行われるSwiftの大規模イベント「try! Swift Conference」が告知された。

▲GitHub社　Scott Chacon氏



▲Realm社　岸川 克己

CONTACT
ギットハブ・ジャパン合同会社 URL http://github.co.jp

ひみつのLinux通信

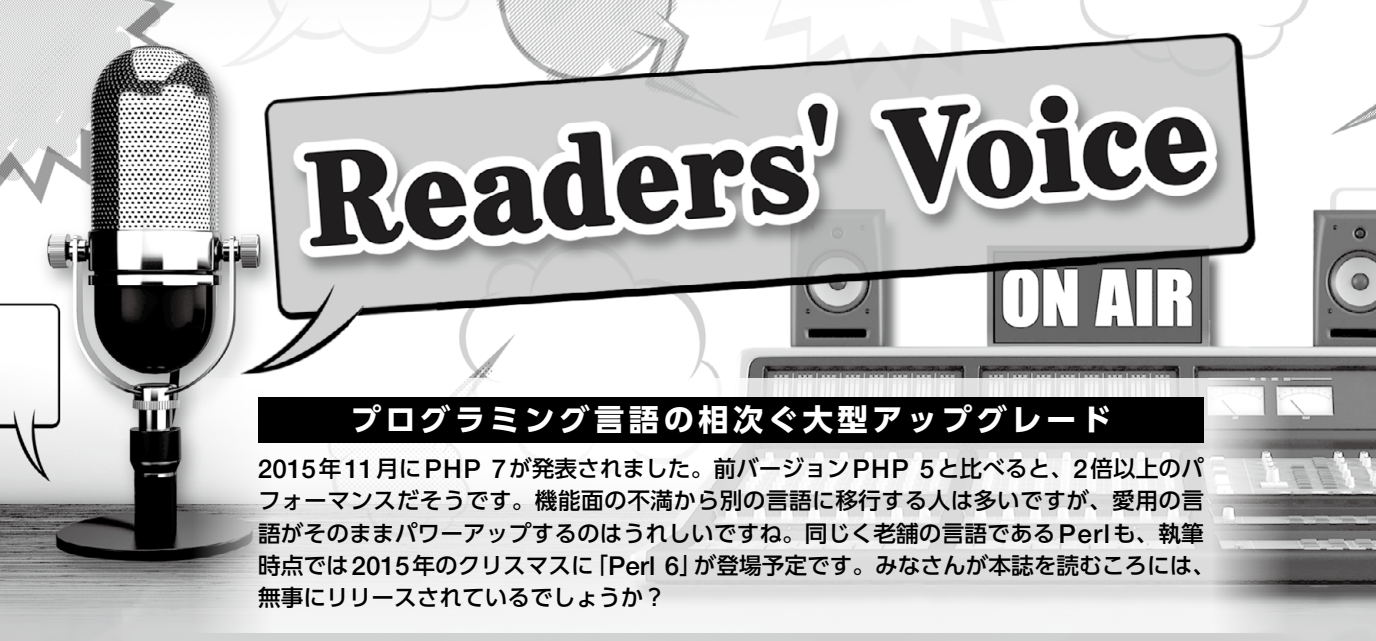
作)くつなりょうすけ
@ryosuke927

第24回 エリート語

殿下という言葉がいくつになっても似合うあの方ですが、アレでナニという言葉が似合うのはくつな先生だけ！

to be Continued

このごろ改竄されたWebサイトに書かれるのは隣の大陸文字が多く、10年ほど前によく見た「leet語」は珍しくなりました。若い人は「ギャル文字」に流れたのでしょうか。いまだに使ってるのは「痛い人」のイメージすらありますね。セキュリティスキャナとして有名なツール「nmap」には、この「leet語」を出力するオプションがあります。「nmap -A サーバ名 -oS -」で「leet語」に変換された結果を標準出力に出せます。対象サーバは、他人の迷惑にならないあなたのサーバなどを指定して試してみてください。ただし、SSHデーモンで使っているホスト鍵のフィンガープリントも「e」が「3」などになって意味がなくなるので本当に戯れ程度で使いましょう。



プログラミング言語の相次ぐ大型アップグレード

2015年11月にPHP 7が発表されました。前バージョンPHP 5と比べると、2倍以上のパフォーマンスだそうです。機能面の不満から別の言語に移行する人は多いですが、愛用の言語がそのままパワーアップするのはうれしいですね。同じく老舗の言語であるPerlも、執筆時点では2015年のクリスマスに「Perl 6」が登場予定です。みなさんが本誌を読むころには、無事にリリースされているでしょうか？

2015年12月号について、たくさんの声が届きました。

第1特集 [決定版]Docker自由自在

2013年に登場したコンテナ型仮想化技術「Docker」ですが、ここにきて多くの企業で導入事例を聞くようになりました。もはや実用期に入ったと言えるDockerについて、基本操作から各種ツールを使った自動化まで、詳細に解説しました。

自動化、省力化まで特集されていて良かった。 さくらますさん／東京都

HashiCorp社のソフトの使い分けは気になっていたので、たいへん参考になりました。 コメットさん／兵庫県

Dockerの基礎から実践的な内容までまとまっていてよかったです。 橿山さん／埼玉県

今とっても困っているところなので、助かります。 tekitoizmさん／東京都

Dockerについて検討している最中に本誌で特集が組まれたので、思わず購入しました。内容も調べていたことが1冊にまとまっていたため、購入して良かったと思えるものでした。 片野さん／埼玉県

この特集目当てに、本誌を買っていただいた人が多かったようです。実際に社内で導入、検討をしている最中の読者から「参考になった」という声が多く寄せられました。

第2特集 SNMPの教科書

「SNMP」は、ネットワークにつながったあらゆる機器を同じルールで管理できるプロトコル。本特集ではそのSNMPについて、入門から応用まで、実運用でのハマりどころも折り混ぜながら幅広く解説しました。

初心者にも優しく、詳しく随所まで勉強できた。 宮嵜さん／大阪府

システム運用管理において有用な機能だと思うが、社内では上の理解を得られず浸透しない。 隼さん／岩手県

MIBは敬遠していたので、しっかりやりたいと思います。 小林さん／東京都

SNMPがあるおかげでネットワーク機器の監視ができて運用業務が助かっていることを、あらためて実感しました。 永作さん／東京都

妙に懐かしく感じられて、書棚から『The Simple Book An Introduction to Management of TCP/IP-based Internets(Prentice-Hall 1991)』を引き出してきて読みなおしています。 鈴木さん／熊本県

ネットワークにつながった機器というと流行のIoTが連想されます。IoTが一般的になっていくなか、より重要なプロトコルになっていくかもしれませんね。これを機に勉強を始めてみては？

短期連載　クラウド時代のWebサービス負荷試験再入門【1】

ITインフラの中心がオンプレミスからクラウドへ移るに伴って、システムに対する負荷試験も、それに合わせたものを用意する必要があります。本連載ではクラウドに載せたWebサービスの「スケーラビリティ」を担保するための負荷試験について見ていきます。第1回では、負荷試験全般について解説しました。

負荷試験をすることの意義が、具体的な例などとともに紹介されていて読みやすかった。 村橋さん／北海道

負荷試験は重要なプロセスであり、これを基本的なところから学び直すこと

12月号のプレゼント当選者は、次の皆さまです

①Olasonic『TW-S9』
宇佐見幸光様（東京都）

②D456 USB Desktop アクアリウム
酒井花乃子（静岡県）

③ウイルスバスターモバイル
尾崎俊一郎様（神奈川県）

④『人工知能入門』
陰山善行様（神奈川県）、Tayu様（千葉県）

⑤『実践JUnit』
丸山貴之様（神奈川県）、くまーーー様（神奈川県）

⑥『あなたの知らない超絶技巧プログラミングの世界』
かふぇのわーる様（東京都）、岡雅善様（東京都）

⑦『Docker実践入門』
ken.saka様（大阪府）、ほまれ様（千葉県）

※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。

ができ、とても良い機会をいただきました。 オミオさん／宮城県

負荷試験というスキルの大切さがわかった。 蛸倉 健さん／長崎県

体験談を交えての解説が、ホントに身に染みる感じでした。負荷試験はまともにやったことがなかったですが、今後は検討します。 NGC2068さん／愛知県

負荷試験という内容に興味が持てました。良い連載です。 Shimizusさん／東京都

Webサービスなどの「会社の商品」となるシステムのダウンは、大きな損失につながります。事前の負荷試験の重要性を知れて良かったという声がいくつか寄せられました。今後の連載ではその具体的な方法が扱われるので、必見です。

短期連載 SMB実装をめぐる冒険【2】

Windows共有フォルダをChromeOSのファイルにマウントするアプリ。その開発には「SMBプロトコル」の理解とJavaScriptでの実装が必要でした。連載第2回では、安全なファイル共有を実現するために、どのような「ユーザ認証」が行われているのかを探りました。

昔からあるSMBをJavaScriptで実装するという発想に驚いた。 山添さん／東京都

知らないことが多かったので良かった。 嶋田さん／三重県

ドキュメントがそろっていない開発の苦労が見えるようで……。 とーふやさん／神奈川県

「やっかいなPadding」についてのコラムが良い。 psiさん／東京都

開発の苦労に共感するような声がいくつか寄せられました。本連載ではドキュメントに頼れないために実際の通信を解析してプロトコルを理解しようとしています。後続の開発者のためにも、ドキュメントを残しておくというのは大事なことだとわかりますね。

連載

「Sphinxで始めるドキュメント作成術」で紹介された方法で、ドキュメント管理がおもしろいように改善された。 raiさん／東京都

プログラマのあいだで、じわじわと人気が出てきたSphinx。その開発言語であるPythonの人気と相まって、さらに広まっていくかもしれません。

「なんでもネットにつなげちまえ道場」の拡大版が読みたいです。 進藤さん／東京都

IoT時代を迎えるにあたり、こちらの連載も注目が高まっているようです。

「セキュリティ実践の基本定石」、とってもよかったです。 kmさん／愛知県

セキュリティに関する情報は、読者からの需要が非常に高いです。本連載では、実際に起きたセキュリティ事件を基に、その背景から要因まで詳細に分析しており、本誌でも人気の連載記事です。

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
March 2016

2016年3月号
定価（本体1,220円+税）
192ページ

2月18日発売

［第1特集］これが定番!
Web開発の新しい教科書
～開発手法、テスト手法、継続的インテグレーションのヒント～

ソフトウェア／サービスのリリース速度の向上、品質の向上に必須である開発サイクルを回すためには、どういったことを行えば良いのでしょうか? Webサービス系企業（リクルート住まいカンパニー、Retty、Increments）によるソフトウェア開発「秘伝」技術紹介と各種ツール群（JIRA、Qiita:Team、HipChat、Jenkins、CircleCI、JaCoCo、Git）の使い方を解説します!

［第2特集］社会／経済を支える技術
あなたの知らないCOBOLの実力
好き嫌いで判断していませんか?

春先に決まって売れるIT書籍あり。その名は「COBOL」。古い、レガシーだ、今どきじゃないと思われていますが、金融システムを支え、社会の基盤となっているのは言うまでもありません。そんなCOBOL今をさまざまな観点から紹介します。

［特別企画］
Webサイトオーナーが改ざん被害に気づいたときにとるべき行動

※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

●メガネの度が合わず、ついに新しいものを作ってもらった。老眼は少し進んだ程度だったので、手もとの度数を下げ、遠くを見やすくしたレンズにしてもらった。結果、よく見えて非常に楽になった。ストレスフリー。本当の老眼鏡のレンズだと1枚3万だそうだ。目の話は老化現象だよなと思いつつ、今年1年を振り返る。（本）

●「かまぶの部屋」が最終回。ゲストの皆様ありがとうございました。敬称略（奥谷泉／永淵恭子／ブヤン／戸倉彩／平初&愛美／長田絵理子／藤崎正範／谷嶋佑里恵／下農淳司／多田歩美／前島有貴／Paul McMahon／鹿野恵子／篠田佳奈／前佛雅人／角田千佳／吉岡弘隆／タナカユカ／上田隆一）また飲みましょう!（幕）

●縁あって応用動物行動学を聴講。アジア系遊牧民がなぜヒツジの群れの中でヤギを飼うのか。数ある理由の中でも、外敵から逃げるとき、ヒツジと外敵との間にヤギが必ず入るという話にはときめいた。「おメェたちは先に逃げな」、ヤギかっこいい! でも詳しく聴くと違う事実が見えてくる。研究って面白い!（キ）

●2016年の三が日は土日と重なっているせいで仕事始めが早い。きっと正月休みは不完全燃焼に違いありません（これを書いているのは2015年末です）。未練がましく2016～2017年の年末年始を調べたところ、大みそかと元日が土日なので、仕事始めはやはり1月4日。今からモチベーションが下がる。（よし）

●仕事中の眠気覚ましに、よくグミを買って食べています。自分は硬いグミが好きで、買うかどうかの判断基準もそこにあったのですが、最近になって、果肉の食感を再現した「コロロ」というやわらかめのグミに出会い、パラダイムシフトが起きました。お菓子メーカーの研究開発力はすごいですね。（な）

●編集部が2階から3階へお引っ越ししたら、キッチンコーナーに置いてある紅茶のティーバックの種類が増えました（嬉しい!）。冷蔵庫に牛乳を常備して、生姜やシナモンパウダーをちょい足ししたスパイスミルクティーを楽しんでいます。このスパイス、ちょっとした気分転換や風邪の予防にもなるのでおススメですよ。（ま）

ご案内
編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［E-mail］
sd@gihyo.co.jp

［FAX］
03-3513-6173

※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

Software Design
2016年2月号

発行日
2016年2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。