Special Feature 1
→基礎のWeb技術
Special Feature 2
→プロはマルチ画面で仕事する!

Software Design

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

9
September / 2017

2017年9月18日発行
毎月1回18日発行
通巻389号
（発刊323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

いま一度振り返るWebのしくみと開発方法

根底から理解

Web技術 超入門

Special Feature ⟼ 1

OSSで開発!
ApacheとNginx比較!
スマホゲームの舞台裏!
喰わず嫌いIIS!

TelnetでWebサーバと通信したことありますか?

Extra Feature ⟼ 1
不正アクセス・なりすましを防ぐ
『認証を支える技術』

Extra Feature ⟼ 2
Eject職人の朝は早い!
夏休みだ、Ejectコマンドで遊ぼう!!
——スマートコンセントで扇風機を回そうの巻

Extra Feature ⟼ 3
［短期集中連載］最終回
人工知能時代のLispのススメ

Special Feature ⟼ 2
tmux＆Byobu
開発効率アップのターミナル改造術
ひとつの画面で
満足していませんか?

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

目次

Contents

Software Design

September, 2017

9

目次

Contents

Software Design

September, 2017

9

Column

Development

OS/Network

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

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

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

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

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

第105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。

Istio

マイクロサービスのためのサービスメッシュ

ITシステムのインフラとしてクラウドの利用が当たり前になりつつある中で、アプリケーションを小さなサービスの組み合わせとして定義するマイクロサービスアーキテクチャが急速に普及しています。業界を牽引するクラウドベンダであるGoogleとIBM、そしてライドシェアリングサービス大手のLyftが新たに開始した「Istio」は、そのようなマイクロサービスに最適化されたインフラソフトウェアの開発を目指すオープンソースプロジェクトです。

マイクロサービスアーキテクチャでは、大規模なアプリケーションを小さなサービスの集合体として定義し、それぞれのサービスがAPIを通じて互いに通信し協調することで動作します。各サービスが独立して実行されるため、変化に対応するための柔軟性が高いという点が大きな特徴で、クラウド時代のアプリケーションの形に適しているというのが人気の高まっている理由です。

しかし実際にこのようなアーキテクチャを安定的かつ効率的に運用するためには、それに適した形のインフラを構築することが必要になります。マイクロサービスによるアプリケーションの運用では、従来のアーキテクチャに比べて極めて大量のサービスが同じ基盤の上で同時に実行されることもあるため、そこには新しい課題も発生します。その代表的な例がトラフィック管理です。マイクロサービスによるアプリケーション内部では、各サービスの間で大量のトラフィックが複雑に飛び交うことになるため、これを適切に管理することが安定的な運用には不可欠だからです。

Istioは、そのようなマイクロサービスの運用に関わる問題を解決し、安全なアプリケーション運用基盤を提供することを目的として立ち上げられました。Istioでは、マイクロサービスの運用に必要となるインフラ機能を「サービスメッシュ」と呼ばれるレイヤとして提供します。サービスメッシュは、サービスとネットワークインフラの間に挟み込まれるレイヤであり、その上で動作するマイクロサービスに対して高い安定性と適切な運用管理機能を提供するとのことです。

Istioで提供される機能

Istioは、クラウドやオンプレミスの区別なくさまざまな環境で利用できる点や、既存のマイクロサービスアプリケーションに特別な修正を加えることなく導入できる点などが大きな強みとなっています。Istioのサービスメッシュでは、おもに次のような機能が提供されます。

- HTTP、gRPC、TCPトラフィックに対する自動ロードバランシング
- 豊富なルーティングルールによるきめ細かいトラフィックコントロール
- トラフィックの暗号化、サービス間の認証、強力な同一性証明などのセキュリティ機能
- 全体に対するポリシーの強制
- 詳細な測定とレポート機能

Istioは、内部的にはLyftが開発してオープンソース化した「Envoy」というサービスプロキシソフトウェアをベースとして開発されているそうです。Envoyはトラフィックのフィルタリングやルーティング、サービスディスカバリ、ロードバランシングのためのヘルスチェック、TLSサポートなどといった機能を提供するプロキシです。Istio上のマイクロサービスは、Envoyを経由して前述のさまざまな管理機能を利用できます。

Istioを利用することで、アプリケーションの運用管理者は、マイクロサービスの稼働状況を適切に把握し、セキュリティや可用性を高めることができます。一方で開発者としては、運用管理に関連する特別な機能をサービス本体に直接組み込む必要がなくなります。また、A/Bテストや先行リリース、障害シミュレーションなどを、システム全体に影響を与えることなく容易に実施できるようになるとのことです。

本稿執筆時点では、オーケストレーションツールとしてKubernetesだけがサポートされていますが、将来的にはCloud FoundryやMesosなどをはじめとする各種ツールや環境への対応も進めていく計画とのことです。SD

Istio
https://istio.io/

DIGITAL GADGET vol.225

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

広告とテクノロジの新しい関係

カンヌ国際クリエイティビティフェスティバル「Cannes Lions 2017」より

2017年6月に、広告を中心としたクリエイティブ作品を讃える国際クリエイティブフェスティバル「Cannes Lions 2017」が開催されました。最近の傾向としては、VR技術や人工知能技術を活用した広告から、FacebookやTwitter、InstagramなどのSNSを最大限に活用した広告展開が目立ちます。また、プロ中のプロが、予算と時間をじっくりかけて作り上げた広告作品、社会的問題や課題に真っ正面から取り組んだもの、ちょっとしたアイデアで予算をそれほどかけずに口コミを重視したものなどなど、さまざまな展開がみてとれます。たとえばイギリスの調査会社Stylus社が今年の特徴として挙げたのは、

- automation：自動制御
- inclusivity：すべてを包括
- layered-identity：階層化された個性

の3つでした。一見、よくわからない言葉が並んでいますが、よくよく考えると、人工知能や機械学習によって、いろいろなものが自動的に制御されるようになり、ある特定の人たちのためではなく、多種多様な人たちのために

※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。

サービスが多様性をもって展開しており、さらに人々の個性もある特定の一種類というわけではなく、さまざまな個性が多層的に重なっている感じが伝わってきます。この3つが消費者のライフスタイルの変化や、これからのマーケティング戦略、ビジネスの創造性の視点で見た現状を象徴しているのでしょう。

また、広告制作のデジタルエージェンシーの中には、顧客から広告制作を受託して作成するだけではなく、自らスタートアップ支援などのしくみを作り、新しいプロダクトや、新しいサービスをともに開発しつつ、広告宣伝も適切に展開するといった「共創」の流れも生まれてきています。単にモノを買ってもらい、どんどん消費していけばよいという時代から、体験や経験、共感といったことが重視される時代になってきたのだとも言えます。

それでは、デジタル視点、ガジェット視点で、いくつかの作品を紹介していきましょう。

サイバー部門（Webサイトやデジタル分野の広告）より

Google Sheep View
http://www.googlesheepview.com/

道路周辺のパノラマ画像を提供するGoogle Street Viewのパロディで、Google Street Viewに羊（Sheep）が映っているものだけを集めたサービス（**pic.1**）。Googleの公式サービスではなく、オランダの人が、観光目的で始めたアートプロジェクト。

Samsung Official TVC: Ostrich
https://www.youtube.com/watch?v=hjKd24UCPYY

「ダチョウにVRゴーグルをかぶせて驚かせたら?」という作品で、GearVR

デジタルカメラの広告。スマートフォンのカメラだと、パスワードロックを外している間にシャッターチャンスを逃してしまいますよ!という状況を示している広告


レゴブロックの「Build The Future（未来を築く）」という新製品シリーズの広告。プリント（ポスター）部門の賞を受賞

というスマートフォン用のVRゴーグルの広告映像。VR映像ですばらしい映像を見て飛び立とうとするダチョウを描いたもので（ダチョウは空を飛ばない鳥）、「#DoWhatYouCan't（できないことをやってみよう）」というキャンペーンが展開された（**pic.2**）。

モバイル部門（スマートフォン、タブレット端末向けの広告）より

glicode
http://cp.glico.jp/glicode/

グリコのお菓子をプログラミングコードに見立てて、簡単なプログラミングを学び、楽しむことのできるしくみ（**pic.3**）。スマートフォンアプリで規則的に並べられたお菓子を撮影することで、プログラミング言語でプログラミングしたのと同じように、繰り返しや、振る舞いを定義することができる。

Chat Yourself
https://www.youtube.com/watch?v=eE6BO5ANyaI

認知症による物忘れをサポートするためのチャットボット（**pic.4**）。自分が何をしようとしていたのか、自分は今どこにいるのか、ある意味過去の自分自身とのチャットにより、思い出すためのしくみ。

デザイン部門（デザインを優れた使い方で利用した広告）より

Hijacked Highway
https://www.youtube.com/watch?v=M66nX2uxAK0

退屈な高速道路での移動を、VRゴーグルで楽しむ解決策（**pic.5**）。高速道路での移動中に巨大なオブジェクトやあり得ない車などが登場し、同乗者の子供たち（大人も）が長時間の移動を楽しみながら過ごすことができる。もちろん運転者は使えない。

The Lenz App
http://saatchi.co.uk/en-gb/work/deutsche-telekom-and-saatchi-saatchi-create-new-media-channel-using-colour-as-the-medium

マゼンタ（ドイツテレコムのブランドカラー）色のものをスマートフォンのカメラを通してみると、何か新しい映像に差し変わって見えるという、音楽バンドGorillazと電話会社ドイツテレコムとのキャンペーン（**pic.6**）。現実空間にCG映像を重ねて表示するARは数多いが、ある意味逆の表現方法がネットの世界とリアル（現実）の世界をあいまいにしている。音楽バンドGorillazの活動自体も仮想的であることから、表現としてもピッタリあっている。

デジタルクラフト部門（ブランドと消費者との優れた体験広告）より

NBA2K
https://www.nba2k.com/

バスケットボールのテレビゲームとウェアラブルデバイスのFitBitが連携し、1日の運動量が1万歩を越えるとゲーム内でのポイントがアップし、ゲーム内キャラクタのパフォーマンスがアップする（**pic.7**）。

Possession bEGINS
https://www.possessionbegins.com/

音だけのインタラクティブ予告編（**pic.8**）。パソコンのWebCamと顔認識技術を活用し、目をつぶっているときだけストーリーが進む。ホラー映画ならではの演出。

この先の広告とテクノロジの進化

広告の分野だけでなく、あらゆる分野でテクノロジを活用するのは当然のこととしながら、とくに広告分野では新しい技術、新しい体験には積極的に乗っていこうという流れを感じます。それとともに新しいテクノロジを活用し続ける挑戦者であろう、一番最初に活用して目にものを見せてやろうという気概が感じられる分野でもあります。

テクノロジの進化によって、必ずしも莫大な予算や、リソース、特別な舞台などがなくとも、話題をさらったり、注目を浴びたりすることは、アイデアや工夫で可能な事柄になりました。

サイバー部門より

pic.1 Google Sheep View

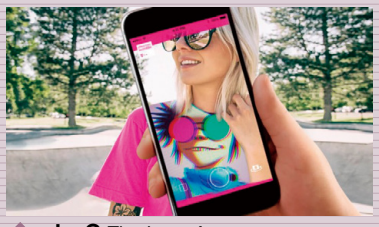
pic.2 Samsung Official TVC: Ostrich

モバイル部門より

pic.3 glicode

デザイン部門より

pic.5 Hijacked Highway



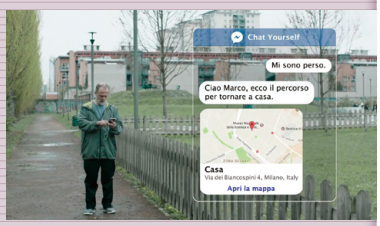
pic.6 The Lenz App

デジタルクラフト部門より

pic.7 NBA2K

たとえば「Google Home of the Whopper」というハンバーガーのキャンペーンは、前回8月号の本連載でも紹介したGoogleのスマートスピーカー「Google Home」のしくみを逆手にとったアイデアです。それは、テレビCMの音声でGoogle Homeが反応するような会話を流し「Ok, Google」で勝手にGoogle Homeが反応するようにしてしまったのです。Google Homeは音声で投げかけられた質問に該当する回答が見つからない場合、オンライン辞書であるWikipediaに書かれている内容を返答します。このしくみを利用しました。「広告のためにWikipediaを書き換えてもいいのか?」という物議を醸し出しましたが「テクノロジの悪用としては最高だ!」との評価もあり、なるほど!と膝を打った人も多かったかもしれません。

この広告の作戦は、実際には数時間で対策が施されたり、数日のうちにGoogle Homeにあらかじめ設定しておいた6名までの音声にしか反応しないような対処がなされました。この広告の例は計算づくの攻防戦ですが、これからの私たちを取り巻く環境では、意図せず、テクノロジのおかげではなく「テクノロジのせい」で変なことが起こってしまうのかもしれませんね。SD



pic.4 Chat Yourself

pic.8 Possession bEGINS

Gadget 1

Magic Wallpaper

http://creativity-online.com/work/castorama-the-magic-wallpaper/51901

魔法の壁紙

フランス企業Castoramaが提供する子供部屋用の壁紙は、壁紙に描かれた模様がデジタルデバイスのARマーカーとして機能し、楽しむことのできる未来的な壁紙です。壁に描かれたキャラクタがスマートフォンやタブレットの中で動き出したり、お絵描きや塗り絵ができたり、ただの壁紙から一気に世界が広がるアイデアです。Castoramaは日曜大工の道具や材料を提供するホームセンター企業であり、この壁紙をきっかけに日曜大工にも興味を持ってもらおうという壁紙広告です。

Gadget 2

hovding

https://hovding.com/

自転車用ヘルメット

だいぶ前からコンセプトデザインだけは先行しており、本当に実現するかどうか皆が見守っていたプロジェクト。219ポンド(約3万円)で販売が開始されました。自転車に乗っている際に自動車に追突された場合など、急な衝撃があった場合にセンサーによって首回りのエアーバッグがふくらみ、頭部を覆って守るしくみになっています。センサーの駆動のために充電の必要があります。機能性だけでなくファッショナブルでコンパクトな作りも注目です。

Gadget 3

WEIGHT Tag by Samsonite

https://www.youtube.com/watch?v=ucsFJ36DyR8

トランクの制限重量を計ることのできる簡易タグ

旅行鞄、トランクメーカーのサムソナイトのアイデア商品。航空機、とくに国際線の飛行機に乗る場合は、預ける荷物の重量制限が厳しく、超過した場合には、思った以上の追加料金を支払わなければいけない場合があります。電子的に軽量する重量計はありますが、比較的高価ですし、わざわざ旅行に持ち歩くのも面倒です。このWEIGHT Tagは、タグが伸びて変形・切断したなら重量超過という、簡易的なしくみながら、必要十分な役目を果たしています。

Gadget 4

TINY1

http://tinymos.com/

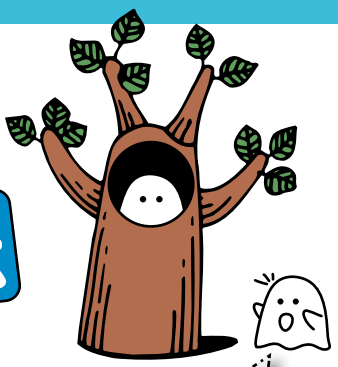
星空専用デジカメ

きれいに星空を撮影するのは、最新のデジタルカメラやスマートフォンでも難しい撮影技術の1つです。TINY1は星空の撮影専用にカスタマイズされたスマートフォンカメラのようなもので、星空や星座を液晶画面で確認したり、撮影した写真をネットで共有したりできます。479ドルで、2017年11月に発売の予定です。解像度は2,688×1,520ピクセル。暗闇の撮影に適したノイズリダクション機能も予定されています。

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

52

結城浩の再発見の発想法



Promise ── プロミス

Promiseとは

プロミス(Promise)とは、**すぐには結果が得られない処理を非同期に実行するためのパターン**です。プログラミング言語によってはフューチャー(Future)と呼ばれることもあります。

通常のプログラムでは、コンピュータは書かれた順序で処理を実行します(逐次処理)。たとえば**リスト1**では、プログラムに書かれた順序で実行が行われています。最初にSTARTと表示し、途中にABC、ABCDEF、ABCDEFGHIという文字列の連結を行い、最後にENDと表示しています(**リスト2**)。

逐次処理は自然でわかりやすいですが、もし途中の処理に時間がかかる場合、それ以降の処理はすべて遅れてしまいます。その結果、応答が返るまでの時間(レスポンスタイム)が長くなり、ユーザに「処理が重い」と感じさせてしまうことになります。

スレッドやコールバックのような**非同期処理**のしくみを使うと、1つの処理の実行終了を待たずに次の処理を実行開始することができ、レスポンスタイムを短くすることができます。たとえば、Webアプリでは、JavaScriptのコールバックがあちこちで効果的に用いられ、ユーザに「重い」と感じさせない工夫がなされています。

JavaScriptのコールバックでは、非同期処理が完了すると、その実行結果を引数に与えてコールバック関数が呼び出されます。それはいいのですが、コールバック関数の中でさらに別の非同期的な処理を行おうとすると、コールバック関数が入れ子状態になり、プログラムが複雑になってしまいます。

プロミスはこのような問題を解決するために用いられます。すぐには結果が得られない処理を実行開始するときに、実行結果の代わりにプロミスを返します。1つのプロミスは実行結果ではなく、実行結果の代替物(プロキシー)として機能します。非同期処理が終了したあとには、プロミスを経由して値を取得できます。また、

▼リスト1　逐次処理

```
console.log('START');

var value = '';

value = 'ABC';
console.log(value); // => ABC
value = value + 'DEF';
console.log(value); // => ABCDEF
value = value + 'GHI';
console.log(value); // => ABCDEFGHI

console.log('END');
```

▼リスト2　出力1

```
START
ABC
ABCDEF
ABCDEFGHI
END
```

プロミスにコールバック関数を与えることで見通しよくプログラムを作ることもできます。

リスト3は、JavaScriptでプロミスを使ったプログラムの例です。ここでは、処理に時間がかかり、結果がすぐに得られない様子をシミュレートするため、1,000ミリ秒のタイムアウトを用いています。先ほどの**リスト2**(出力1)では、ENDが最後に表示されましたが、**リスト4**(出力2)では、ENDが表示されたあとになってから、ABC、ABCDEF、ABCDEFGHIという文字列が表示されていることに注目してください。

日常生活と非同期処理

私たちは日常生活で非同期処理をよく行います。たとえば**料理を作る**ときを考えましょう。卵をゆでながら、トースターでパンを焼きながら、野菜を切るという具合に、複数の作業を並行して行いますね。

▼リスト3　プロミス

```
console.log('START');

new Promise((resolved, rejected) => {
  setTimeout(resolved, 1000, 'ABC');
}).then((value) => {
  console.log(value); // => ABC
  return value + 'DEF';
}).then((value) => {
  console.log(value); // => ABCDEF
  return value + 'GHI';
}).then((value) => {
  console.log(value); // => ABCDEFGHI
});

console.log('END');
```

▼リスト4　出力2

```
START
END
ABC
ABCDEF
ABCDEFGHI
```

そこで大事なのは、同期を取る方法です。私たちは、卵がゆであがるまでその前でボーッと待つということはしません。キッチンタイマーをセットして、火を止める時間がきたらベルが鳴るようにするのは、いわば、自分をコールバックしているのですね。また、パンが飛び出すトースターを使うのは焼き過ぎないようにするためです。非同期処理ができないと効率が悪くなり、きちんと同期が取れないと成果物が正しく得られません。

日常生活でプロミスに近いものは**引換券**です。たとえば、大きなケーキを買いに行ったとしましょう。代金を支払ってもすぐにはケーキはできません。ケーキの代わりに店員は引換券をくれました。あとでその引換券を持って行けばケーキと引き換えてくれるので、ケーキができるまで店頭で待っている必要はなくなります。引換券が購入した印となり、ケーキという成果物を正しく受け取れるのです。**約束手形**や**整理券**などもプロミスに似ていますね。

複数人で**共同作業する**ときも、効率と同期を考える必要があります。複数人いるメリットはそれぞれが独立に作業できるからです。つまり、非同期性を高めれば高めるほど効率が上がるでしょう。しかし、うまく同期を取らないと混乱が生じてしまいます。

共同作業においてプロミスに近い発想は、作業を誰かに依頼するときに、その成果物を置いておく共有フォルダを定めておくようなものでしょうか。人数が多くなればなるほど、成果物をどのように受け渡すか、作業の完了をどうやって連絡し合うかは重要になりますね。

◆ ◆ ◆

あなたの周りを見回して、並行して行っている作業を探してみてください。そのとき、作業の結果をどのようにして同期していますか。あるいはまた、本来なら並行して進められるのに逐次処理をしたり、無用な待ちが発生したりしている作業はありませんか。

ぜひ、考えてみてください。SD

及川卓也の プロダクト開発の道しるべ

品質を高めるプロダクトマネージャーの仕事とは？

第11回 Mind The Productカンファレンス

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

今回は趣向を変えて、6月にサンフランシスコで行われたMind The Productカンファレンスの模様をお届けします。

Mind The Productとは

Mind The Product[注1]は、2010年にスタートした地域ミートアップコミュニティのProductTank[注2]を母体とする世界的なプロダクトマネージャーのためのコミュニティです。ProductTankが100を超える都市で5万名以上のメンバーを持つミートアップを行い、Mind The Productは年次の大規模カンファレンスを開催するという分担になっているようですが、どちらもほぼ同じメンバーにより構成されているなど、この両者はとても近い関係にあります。

筆者は昨年このMind The Productカンファレンスをふとしたきっかけで知ったのですが、それ以来、参加する機会を探していました。今回、所属していた会社を退職したタイミングを活かし、単独で参加してきました[注3]。

Mind The Product サンフランシスコ

もともとはロンドンで開始されたMind The Productカンファレンスですが、昨年からはサンフランシスコでも開催されるようになっています。今年は6月13日にサンフランシスコのデイビスシンフォニーホールというコンサートホールで行われました(**写真1**)。ここはダウンタウンからは少し離れた場所にあり、近くにはTwitterの本社などがあります。

中はコンサートホールそのものなので、大きなホールがあるだけです。ですが、Mind The Productカンファレンスの構成は最初から最後まで1つのトラックしかないというシンプルなものなので、とくに問題ありません。運営からの発表では、世界20ヵ国および米国からは36の州から1,500名ほど集まったそうですが、参加者はこのホールで1つのトラックを聞き、休憩時間には外の通路で交流したり、スポンサーとなったベンダの展示などを訪れたりしていま

▼写真1　会場となったデイビスシンフォニーホール。サンフランシスコ市庁舎やTwitter本社があるものの、ダウンタウンからここまでの間は若干治安の悪いところもある

注1）URL http://www.mindtheproduct.com/
注2）URL http://www.producttank.com/
注3）現地で2人の日本人にもお会いしました。

した。

セッション

10時から開始されたカンファレンスはオープニングとクロージング以外に10のセッションから構成されていました。途中に40分と45分の休憩時間がある以外は、ひたすらセッションを聞かされます。こう書くと、かなり忙しいカンファレンスだったのではと思われるかもしれませんが、実際そうでした。ほとんどのセッションは25分という短い時間なのですが、話すスピードは速く、とても内容の濃いものでした。仕事の関係で、前日の夜に現地入りした筆者は終わるころにはくたくたになっていました。

セッション自体はいわゆるTEDスタイル。スピーカーは壇上で身振り手振りを交えながら参加者に訴えかけます。プロダクトマネージャーの要件の1つに、人々の心に訴えかけられるようなプレゼンテーション能力というのも挙げられると思いますが、今回のカンファレンスはその意味でもとても参考になりました。

オープニング

オープニングはProductTankおよびMind The Productの創設者のMartin Eriksson氏[注4]によるものでした(**写真2**)。カンファレンスの歩みなどを簡単に説明したあと、彼が言った“Product management has almost nothing to do with product, and everything to do with people.”(プロダクトマネージメントはプロダクトに関係するものはほとんどなく、人々に関係するものがすべてだ)という言葉はこのカンファレンスのみならず、コミュニティの精神として心に響きました。実際、このあとのセッションでも何度も「人」に関係する話が出てきました。

注4) URL https://twitter.com/bfgmartin

心に残ったセッション

さて、すでに書いたように、セッションは全部で10あったのですが、誌面の関係もあり、ここですべてを紹介することはできません。そこでいくつか心に残ったセッションをかいつまんで紹介してみたいと思います。

Building Products AI-first (AIファーストのプロダクト開発)

GoogleでAI製品のプロダクトマネージメントのディレクターを務めるAparna Chennapragada氏がAI関連プロダクトでのプロダクトマネージメントのあり方を話しました。彼女はAI時代の4ヵ条として、1)人がコンピュータに合わせるのではなく、コンピュータが人に合わせる、2)どのような(AIが提供する)高度な能力をユーザに与えるかをしっかりと考える、3)プロダクトを作るという発想からプロダクトをトレーニングするという発想への転換、4)新しいプロダクトは新しい機能から生まれると考える、を参加者に訴えました。また、機械学習のアルゴリズムやライブラリ、フレームワークはオープンであるものがほとんどのため、競合との優位性はじきに出しにくくなる。そのときにプロダクトマネージャーがやるべきことは、機械学習により解決すべき正しい課題を見つけ出すことであるとの話も、とても説得力がありました。

▼写真2　オープニングを飾ったMartin Eriksson氏。Mind The Productの創設者の1人。フィナンシャルタイムズなどで勤務した経験がある

少し脱線しますが、AI時代への移り変わりということで彼女が話したエピソードもおもしろかったので、紹介します。モバイルネイティブ世代の子どもたちはスマホやタブレットでないデバイスや、ましては紙でも、ピンチアウトして拡大しようとしたり、スワイプして次に移ることを期待するというのはすでに知られるところですが、Googleアシスタントなどに慣れた子どもたちは道端にある機械に話しかけて、返答を期待するというのがあるそうです。これからのAIファースト時代のプロダクト開発はそのような世代に向けてのものとなっていくのかもしれません。

Pluralsight CXO Nate Walkingshaw氏

技術系オンライントレーニングを行うPluralsight[注5]のCXO Nate Walkingshaw氏の話は、今回のカンファレンスの中で個人的にもっともおもしろかったものです(**写真3**)。EMS(緊急医療サービス)のスタッフとしてキャリアをスタートさせたという彼の話は、さまざまな状況の人間の心電図の波形を見せるところから始まりました。どう関係するのかと思っていたら、緊急医療における心構えとプロダクトマネージャーの心構えが同じであるとの話に結びつけ、そして最後は心電図の波形とプロダクトの成長の様子を結びつけ、参加者を唸らせました(**写真4**)。こう書くと、こじつけのように聞こえるかもしれませんが、そんなことはまったくなく、その話の展開は見事なものでした。

Escaping the Build Trap(ものづくりの罠からの脱出)

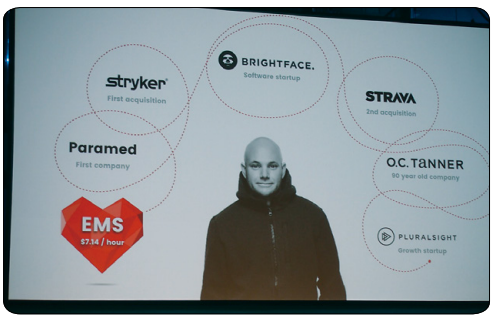
Produx Labs[注6] CEOであるMelissa Perri氏は開発するということが目的になり、良いプロダクトを開発するという視点が失われてしまう“Build Trap”の危険性を訴えました。彼女のセッションでは、正しい時期に正しいプロセスを適切に用いることで、プロダクト開発の失敗のリスクを軽減させられることを説明しました。セッションの中では、いくつも筆者の心にグッとくる言葉があったのですが、その1つが「スクラムはどのように作るかは教えてくれるが、何を作るかは教えてくれない」というものです。これが先ほどの正しい時期に正しいプロセスを用いるという例で、スクラムは作るものが決まったあとのプロセスであり、顧客が必要とするプロダクトを決定する時期に適したプロセスはまた別にあるのです。

それぞれの時期に適したプロセスの根幹にあるのが、ミッションとビジョン、そしてゴールです。彼女はこれを全社レベルのものから、機能レベルのものまでブレークダウンしていくことを勧めています(**写真5**)。ほかにも、戦略とプロセスと文化の深い関係も述べるなど、このセッションもとても内容が濃いものでした。

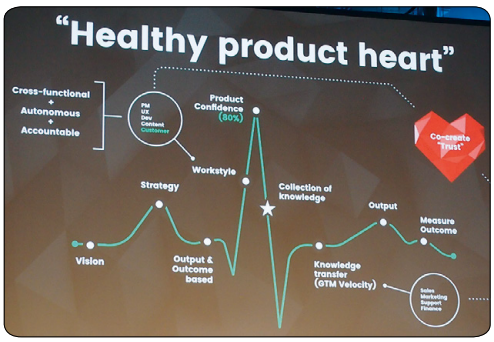
注5) URL https://www.pluralsight.com/

注6) URL http://www.produxlabs.com/

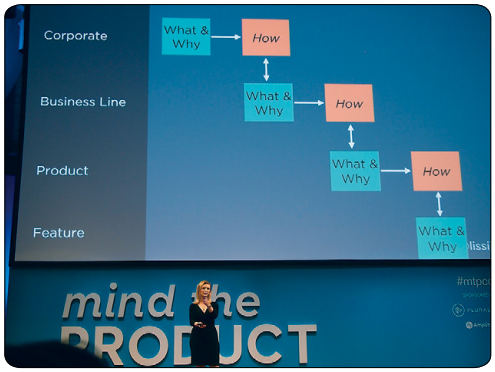
▼写真3　Nate Walkingshaw氏の経歴。EMS(緊急医療サービス)スタッフがキャリアのスタートという異色の人物



▼写真4　心電図の波形と同じ波形でプロダクトを成功に導くまでの導線を解説



▼写真5　What & Why → Howをブレークダウンさせていくアプローチを説明するMelissa Perri氏



そのほかのセッション

紹介した以外にも、メトリックスについてフォーカスしたセッションやOculusのプロダクトデザインのトップを務めるCaitlin Kalinowski氏のプロトタイピングに関してのセッションなど、とても興味深いセッションが多くありました。プロトタイプはできるだけ醜いものにしろという話など、いつか機会があったら、各セッションの中で述べられていることをこの連載の中で触れていきたいと思います。

会場の様子

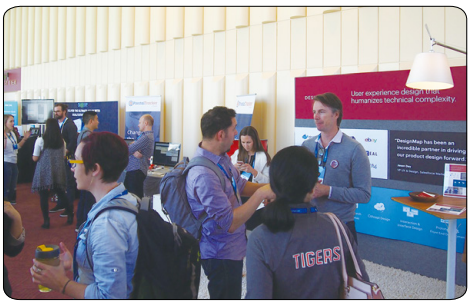
すでに説明したように、会場は通常のコンサートホールなのですが、ホール外の通路は広めにできており、そこで休憩時間になると参加者同士が交流できるようになっています。そのため、休憩時間などには熱心に意見交換する参加者の姿が多くありました。

また、2階と3階の通路には20を超えるスポンサーの展示が行われており、ここでも参加者が展示スタッフと話をしていました(**写真6、7**)。筆者もほぼすべての展示を覗いてみました。興味ある展示では、いろいろと質問もしてみたのですが、参加証に印刷されているQRコードをスキャンされたため、その翌週には多くの企業からメールがあり、フォローアップで個別に(ネット越しに)デモなどをしてもらえることになりました。筆者の英語もとても流暢なものではありませんが、みなさんも海外のカンファレンスなどに行った際には、ぜひとも参加者同士、または出展者などと交流してみてください。最近ではセッションの資料や動画などは後からネットに公開されることも多くなっているので、現地で参加することの醍醐味は現地でしか経験できない交流となってきています。

◆ ◆ ◆

以上、世界的なプロダクトマネージャーのカンファレンスであるMind The Productをレポートしました。次回のMind The Productカンファレンスは9月にロンドンで開催です。もともとロンドンが発祥の地ということもあり、こちらのほうがより大規模とも聞いています。このレポートを読んで興味を持った方はぜひとも参加を検討してみてください。SD

▼写真6　スポンサーによる展示が行われており、休憩時間には多くの参加者が製品やサービスの紹介を受けていた



▼写真7　プロダクトマネージメントやUX、リーン開発などの本が販売されていた

Profile
ソフトウェアエンジニアとして社会人キャリアをスタートした後、MicrosoftやGoogleでプロダクトマネージャーやエンジニアリングマネージャーを経験。現在はいくつかのスタートアップのプロダクトマネージメントをサポートしている。

宮原徹の

オープンソース放浪記

第19回 OSC沖縄のついでに石垣島でリモートワークしてみた

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

宮原流リモートワークのご紹介

OSCで全国を飛び回っているなか、とくに好きなのが沖縄です。OSC沖縄を2005年に初開催して以来、毎年のように開催していますが、ここ最近はOSC終了後、離島にしばらく行くのを習慣にしています。

今回は6月15日(木)に沖縄に入ったあと、6月25日(日)まで10泊11日の長期滞在となりました。もちろん、その間遊んでいたわけではなく、石垣島に行ってリモートワークをしたりしていたので、今回は日記ふうに、その11日間を綴ってみたいと思います。

15 雨の沖縄

午後から沖縄に移動。沖縄はまだ梅雨が明けていないということで雨。到着後、沖縄に来たら必ず行く「ジャッキーステーキハウス」で食事。毎回供される謎の白いスープがあるのですが、今回なぜか味が美味しくなっていて、逆に物足りなさを感じます。その後、4月から私の会社に新卒入社した北村壮大氏、本コーナー常連の坂井恵氏と一緒に牧志公設市場の裏手にある立ち飲み屋でせんべろ[注1](**写真1**)。

16 美ら海水族館→会場設営

スタッフと一緒に美ら海水族館まで。名護市のオリオンビール工場で美味しいビールを試飲。いつもはドライバーですが、今回は坂井氏が運転してくれたので飲めました。感謝。

夕方からOSC会場の設営を1時間ほど行いました。終了後、海鮮居酒屋で沖縄の美味しい魚を堪能。2次会は泡盛専門バー「泡盛倉庫」で泡盛の古酒を満喫。AKB総選挙は中止に。

17 OSC当日→琉球居酒屋

OSC当日。雨はまだ降っていましたが、今回の会場はモノレール駅からほど近く、影響少なし。参加者は100名程度と例年並み。展示スペースをゆったりと取ったので快適でした。終了後、琉球居酒屋で懇親会。2次会は泡盛倉庫(**写真2**)。

やや飲み足りなかったので、工藤淳氏と居酒屋で刺身にビール。この時点で深夜1時。お疲れ様でした。

18 那覇市観光

スタッフと那覇市内観光。識名園、首里城、牧志公設市場、壺屋の焼物街をブラブラ。疲れを癒すために昼寝後、沖縄転勤中の友人と合流して「肉山」で肉を味わいました。

19 沖縄オープンラボ訪問

6月から加入した沖縄オープンラボ(うるま市)を訪問し、午前午後じっくりとディスカッション。終了後、那覇市内に戻り、ヘリオス酒造の直営店でビール三昧懇親会。2次会は泡盛倉庫へ行きました。

注1) 千円でべろべろに酔える、が由来の低価格で飲める居酒屋や立ち飲み店。

▼写真1 1,000円でつまみ1品、飲み物3杯が沖縄のせんべろスタンダードらしいです

▼写真2 泡盛倉庫にて。壁にもぎっしりレアな泡盛の瓶が並んでいます

▼写真3 日本UNIXユーザ会の法林浩之氏と。離島ターミナルは八重山の各島への高速船が発着しています

▼写真4　喜田紘介氏、由衣さんと婚姻届。石垣島で婚姻届を出すダイビング好きな人は多いらしいです

▼写真5　2日間お世話になったカフェスペース。リノベーションしたばかりのお洒落空間でした

20 石垣島へ移動

梅雨明け快晴。石垣島に移動。まずは竹富島に行って食事、海で泳ごうと離島ターミナルに行くと、西表島に行く法林浩之氏とばったり遭遇(**写真3**)。

竹富島では「そば処 竹の子」で八重山そばを堪能し、コンドイビーチで泳ぎました。石垣島に戻って日本最南端の樽生ギネスビールを堪能後、一足先に石垣島に来ていた日本PostgreSQLユーザ会(JPUG)の喜田紘介氏と合流して海鮮居酒屋で泡盛を堪能しました。

21 リモートワーク開始

喜田氏が石垣島市役所に婚姻届を出しに行くというので、カメラマン(野次馬)として同行(**写真4**)。1時間ほどで終わったので、そのあとは宿でリモートワーク。沖縄そばで遅めの昼食後、ホテルエメラルドアイランドのカフェでリモートワーク。無線LAN、電源も使えて、フリードリンクで500円は安いですね(**写真5**)。19時まで仕事。夜は喜田氏と海鮮居酒屋で泡盛を嗜みました。

22 リモートワーク(訪、辺銀食堂)

午前中は宿でリモートワーク。ランチは石垣島ラー油で有名な辺銀食堂でジャージャー麺。午後は昨日と同じカフェでリモートワーク。夜は喜田氏、JPUGの尾高 保氏と海鮮居酒屋で泡盛を満喫しました。

23 那覇へ戻る

午前中は宿でリモートワーク。昼からANAインターコンチネンタル石垣リゾートにある真栄里ビーチで泳ぐ。ホテルのシャワーや更衣室が無料で使え、空港までのバスも停まるので、石垣島でもう一泳ぎするときにお勧めです。那覇到着後、翌日のイベントの準備のお手伝い。夜はたまたま社員旅行で沖縄入りしていたミラクル・リナックスのみなさんと琉球居酒屋で懇親会。2次会は泡盛倉庫で泡盛の古酒を堪能しました(**写真6**)。

24 ハッカーズチャンプルー

沖縄のみなさんが企画しているイベント「ハッカーズチャンプルー」にスタッフ参加。100人以上集まり、なかなかの盛り上がり。LTでラズパイオーディオを紹介、会場の片隅でデモをしたところ、興味を持ってくれる人が多く有意義でした。懇親会では琉球大学の学生さんといろいろお話ができました。沖縄でもYAPC(Yet Another Perl Conference)を開催するそうです。

25 帰途

もうさすがにやることもなくなったので、ホテルをチェックアウトしたらそのまま那覇空港に。11日間、お疲れ様でした。

♣

さすがにここまで長期間不在にしていると、多少仕事に支障はありましたが、一方で集中的に仕事をこなすこともできたので、たまには1人で離島合宿も悪くないなあと思いました。来年は宮古島あたりで合宿ですかね。SD

▼写真6　2次会は泡盛倉庫で泡盛古酒三昧

ツボイのなんでもネットにつなげちまえ道場

第27回

Electric Impを使ってみる

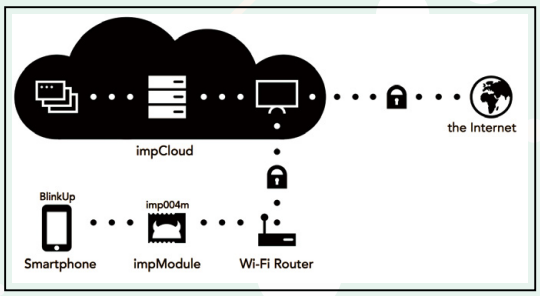
Author 坪井 義浩（つぼい よしひろ） Mail ytsuboi@gmail.com @ytsuboi

Electric Impとは

Electric Imp（エレクトリック・インプ）は、IoTデバイスを開発するためのプラットフォームの1つです（**図1**）。つまり、この連載でおもに使っているARM mbedと同じような存在です。Electric Impは、アメリカのElectric Imp, Inc.が提供しているプラットフォームです。特徴的なのは、impModuleと呼ばれる無線LANに接続できるマイコン内蔵のモジュールがハードウェアとして提供され、このモジュールでimpOSという専用のOSが動作します。

Electric Impの特徴の1つに、impCloud側でimpModuleごとに用意されるマイクロサーバ、エージェントの存在が挙げられます。エージェントは、オンラインサービスやほかのクラウドサービスなどインターネットに接続されたリソースと通信できます。また、エージェントはサーバで動きますので、電力を消費する複雑な計算などをimpModuleに代わって実行させることもできます。たとえば、impModuleが省電力モードに入ってインターネット接続をしていない状態でも、代わりに処理を行うことができます。

▼図1　Electric Impの全体像



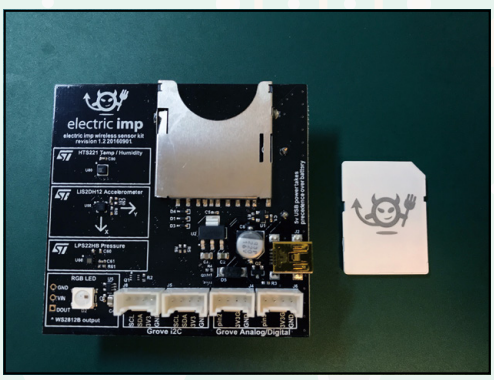
エージェントの利点はもう1つあって、エージェントによってインターネット側からimpModuleが直接的に見えない存在となることです。エージェントが通信を仲介することで、impModuleはセキュアな状態にいることができます。

初期のimpModuleであるimp001は、SDカードに似た形状の製品でした（**写真1**）。こういった、はんだづけをしないモジュールは電子工作には差し替えができて便利なのですが、IoTデバイスを量産するときには不便です（ソケットのコストがかかりますし、組み立て工程が増えます）。

imp001は、技適マークが付いていなかったこともあり、国内で見かけることはありませんでした。しかし最近、量産向けのimpModuleであるimp004mが登場し、これには技適マークが付いています。imp004mを搭載した開発キットは、ソフトバンクの+Style（プラススタイル）というWebサイト[注1]で販売されています（**写真2**）。

注1）　https://plusstyle.jp/shopping/item?id=219

▼写真1　imp001と評価用のimpExplorer Kit



impExplorerの準備

Electric Impのimpモジュールやエージェントの開発は、オンラインで提供されているElectric Imp IDE[注2](以降、IDE)で行います。IDEにアクセスし、アカウントを作りましょう。

IDEで開発したプログラムのimpモジュールへの書き込みは、impCloudからネットワーク経由で行われます。imp004mは無線LANでネットワークに接続しますので、まずimp004mを手元のWi-Fiネットワークに接続しなければなりません。impモジュールにWi-Fiへの接続情報を渡すには、AndroidやiPhoneといったスマートフォンを使い、BlinkUpと呼ばれる技術を使います。iTunes App StoreやGoogle Playで「electric imp」を検索すると、BlinkUpするためのアプリケーションをダウンロードできます。

アプリをスマートフォンで起動すると、ログインを求められます。ここで、先ほど作ったアカウントのIDとパスワードを入力してください。そこでimpモジュールを接続したいWi-Fiネットワークの SSIDを選択し、パスワードを入力すればスマートフォン側の準備は完了です。次にimpExplorerの電源を入れましょう。impExplorerにはUSBか、単三電池3本で給電できます。筆者はUSBケーブルを接続して給電してみました。

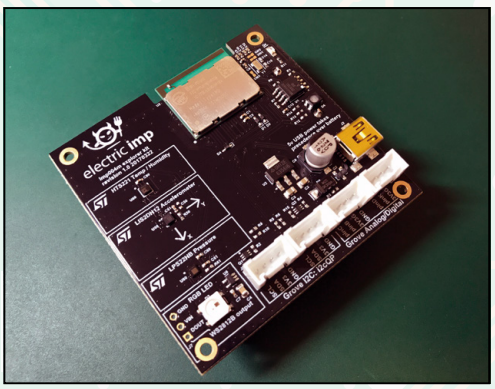
SSIDを選んだ後では、**図2**のような画面が表示されているはずです。ここで、「START BLINKUP」というボタンを押し、給電しているimpExplorerに**写真3**のようにスマートフォンの画面を向けて置くとBlinkUpできます。そうです、BlinkUpとは、スマートフォンのバックライトを点滅させた光[注3]のON/OFFでSSIDやパスワードを転送しています。

impExplorerの無線LANへの接続状況は、LEDで知ることができます。BlinkUpが成功すると、impExplorerのLEDが緑色に3秒間点灯

注2) https://ide.electricimp.com/

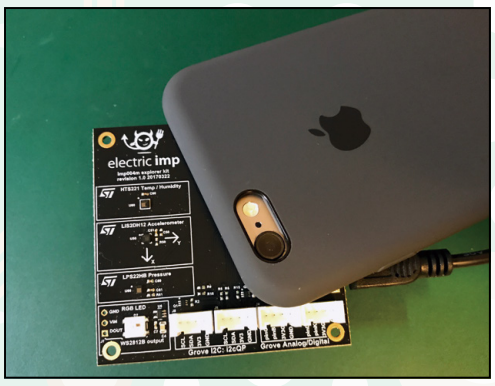
注3) アプリにも注意が出ますが、光が点滅しますので画面は注視しないほうがよいでしょう。

▼写真2 impExplorer Developer Kit



▼図2 スマートフォンの画面

▼写真3 BlinkUpしている様子



します。次に、impExplorerがネットワークに接続している間、LEDは赤とオレンジと交互に点滅します。接続が成功すると、LEDは緑色のゆっくりとした点滅をします。impExplorerがimpCloudと接続できると、IDEの「Devices」という画面にBlinkUpしたimpExplorerが追加されます(**図3**)。

Hello World

impExplorerがIDEのDevicesに追加されたのを確認したところで、さっそくHello Worldしてみましょう。追加されたデバイスの行にマウスポインタを移動すると、**図3**のようにSettingsのアイコンが表示されます。これをクリックすると、デバイスの名前を変更したり、モデルを選択・作成したりする画面に移動します(**図4**)。モデルというのは、Electric Impのアプリケーションのことで、impModuleで動かすアプリケーションと、エージェントで動かすアプリケーションの組み合わせをこう呼んでいます。

初めてIDEを使う段階では、モデルは何も登録されていませんので、New Modelを選択して、ここでは「Hello World」というモデルを作って割り当てることにしてみました。「Save Changes」ボタンをクリックすると、デバイスの名前が変更され、新しいモデルが作られます。

ふたたびデバイスの行にマウスポインタを移動させ、Settingsの左どなりにあるCodeボタンをクリックしてみましょう。すると、Hello Worldモデルのコードを編集する画面に遷移します(**図5**)。この画面にはコードを編集する欄が2つあり、左側がエージェント用、右側がデバイス用になっています。下のペインにはログが表示されます。

ここで、デバイスのコード編集ペインで次のようなコードを入力してみてください。

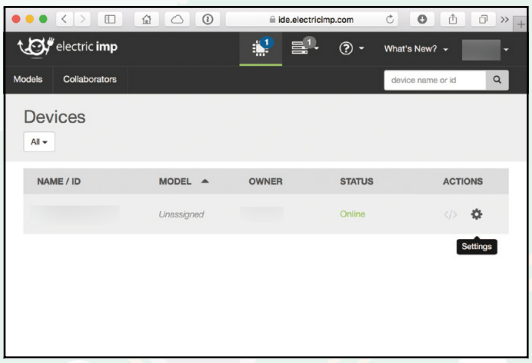
```
server.log("Hello from the impExplorer");
```

そして、「Build and Run」というボタンをクリックしてみましょう。

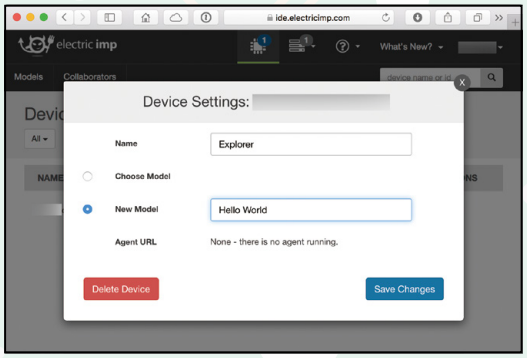
```
[Device]    Hello from the impExplorer
```

と表示されるはずです。ボタンをクリックすることでコードがサーバ側でコンパイルされ、ネットワーク経由でimpModuleに転送・実行されて、impModuleからimpCloudにメッセージが送られてIDEの画面に表示されました。すごく簡単ですね。

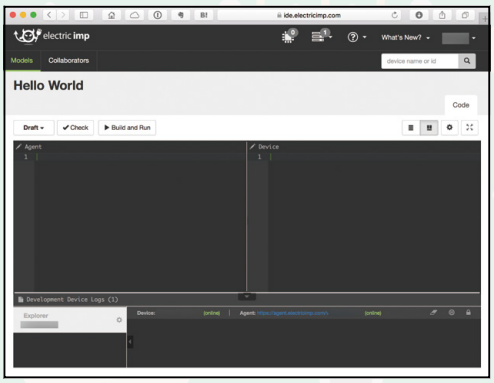
▼図3　impExplorerがIDEに追加された



▼図4　デバイスの設定



▼図5　エディタの画面



もっと複雑なことをしてみよう

次に、一気に先に進んで、エージェントとデバイス両方のコードを用意して、WebブラウザからimpExplorerに搭載されているLEDをON/OFFしてみましょう。

サンプルコードは、スイッチサイエンスのページ[注4]に掲載されています。デバイスコードとエージェントコードをそれぞれIDEのペインにコピー&ペーストして、「Build and Run」ボタンをクリックします(**図6**)。すると、ログのペインに、URLが2つ表示されます。これがLEDをONするURLと、OFFするURLです。

このURLをコピーして、手元のWebブラウザでアクセスしてみてください。impExplorerのLEDを赤く点けたり、消したりできます。印象的なのは、これをコメント含めエージェント側27行、デバイス側23行でコーディングでき、さらにサーバを用意する必要がないことです。

このコードは、Squirrel[注5](スクワール)という言語を使ってImp API[注6]をコールするように記述されています。

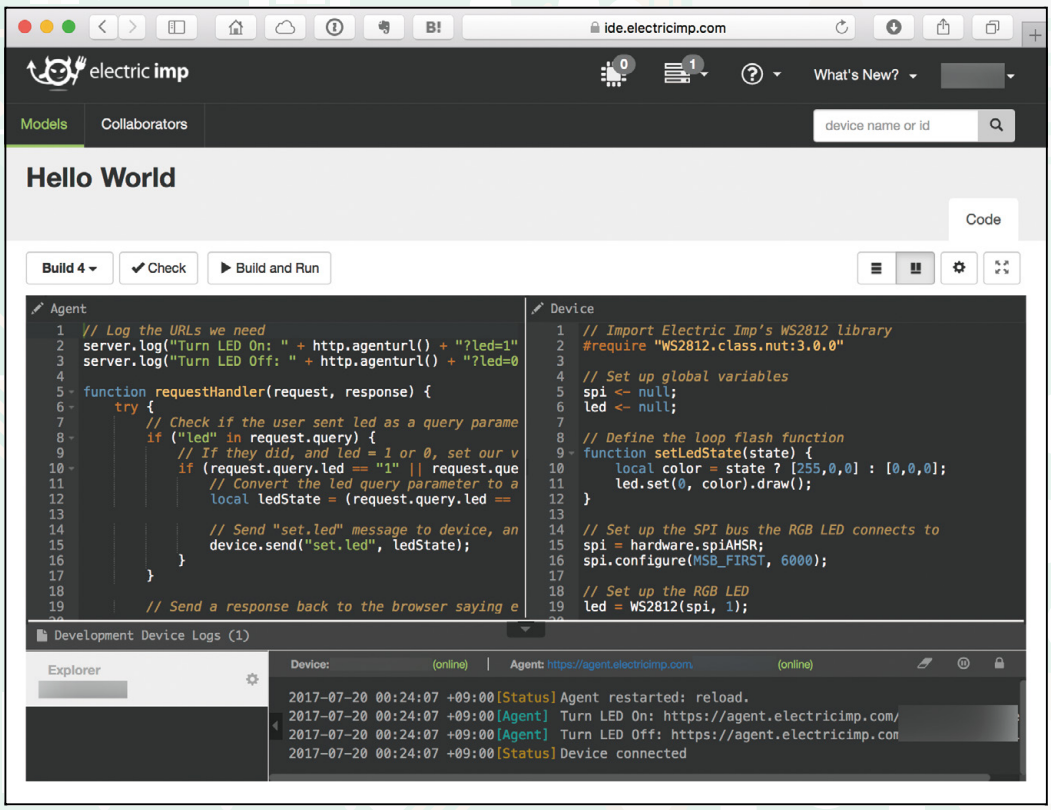
impExplorerの基板を見てみると、前回(2017年8月号)の本連載で紹介したGroveのコネクタが4つ搭載されているのが見て取れると思います。impExplorerには今回使ったLEDのほかに、温湿度センサ、加速度センサ、大気圧センサが搭載されています。Groveを使えば、はんだづけをすることなく、これら以外のデバイスも簡単にimpExplorerに接続できるでしょう。SD

注4) http://pages.switch-science.com/electricimp/agents.html

注5) https://electricimp.com/docs/squirrel/

注6) https://electricimp.com/docs/api/

▼図6 サンプルコードをビルドして実行してみた



PRESENT 読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト**http://sd.gihyo.jp/**の「読者アンケートと資料請求」にアクセスし、アンケートにご協力ください（アンケートに回答するにはgihyo.jpへのお名前と住所のアカウント登録が必要となります）。ご希望のプレゼント番号を記入いただいた方の中から抽選でプレゼントを差し上げます。締め切りは**2017年9月14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 HHKB Professional BT（無刻印）

2016年12月で20周年を迎えたコンパクトキーボード「Happy Hacking Keyboard」のBluetooth接続モデルに、新色「白」が追加されました。スペックそのほかは現行のカラー「墨」と変わらず、「静電容量無接点方式のキー入力」「Bluetooth 3.0対応」「単三電池2本駆動（給電用USB micro-Bコネクタも搭載）」といった特徴を持ちます。無刻印（英語配列）をプレゼント。

提供元 PFU http://www.pfu.fujitsu.com

1名

02 ドッキングステーション「PUD-PDC3L」

USB 3.0ハブを3ポートと、有線LANポート（1000BASE-Tに対応）を拡張できるドッキングステーションです。Windows/Mac環境に対応し、PC側のインターフェースはUSB3.1 Type-C。またPower Deliveryに対応しているので、使用しながら充電もできます。

提供元 プリンストン
http://www.princeton.co.jp

1名

03 ヌーラボ ノベルティTシャツ

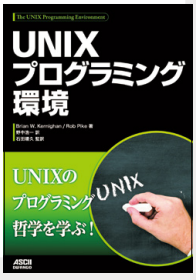
チームにおけるコーラボレーションのためのWebツール「typetalk」「cacoo」「backlog」を提供するヌーラボの、ノベルティTシャツです。MまたはXLサイズを、ステッカーとセットにしてプレゼント。

提供元 ヌーラボ
https://nulab-inc.com

3名

04 UNIXプログラミング環境

Brian Kernighan、Rob Pike 著



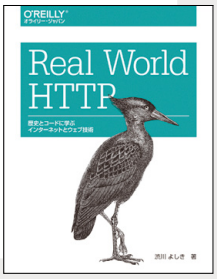
端末の基本的な使い方から、シェルの活用、sedやawkによるフィルタの作り方、C言語によるプログラミングまで、UNIXをより深く使いこなすための知識を、豊富なサンプルとともにわかりやすく解説。

提供元 アスキードワンゴ
http://asciidwango.jp

2名

05 Real World HTTP

渋川 よしき 著



HTTP/1.0/1.1/2と、HTTPの進化をたどりながら、ブラウザ内部の動作、サーバとのやりとりについて、Go言語やJava Scriptによるプロトコルの実例や実際の使用例などを交えながら紹介しています。

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

2名

06 現場で役立つシステム設計の原則

増田 亨 著

日本最大級の求人情報サイト「イーキャリアJobSearch」の主任設計者であり、システム設計のベテランである筆者が、コードの具体例を示しながら、良い設計のやり方と考え方を解説した1冊。

提供元 技術評論社
http://gihyo.jp

2名

07 IBM Bluemixクラウド開発入門

常田 秀明、水津 幸太、大島 騎頼 著

IBMのクラウドサービス「Bluemix」について、基本的な導入方法の解説から実際のアプリケーションを作る方法までを解説。Raspberry Piと拡張知能「Watson」を組み合わせた事例も紹介。

提供元 技術評論社
http://gihyo.jp

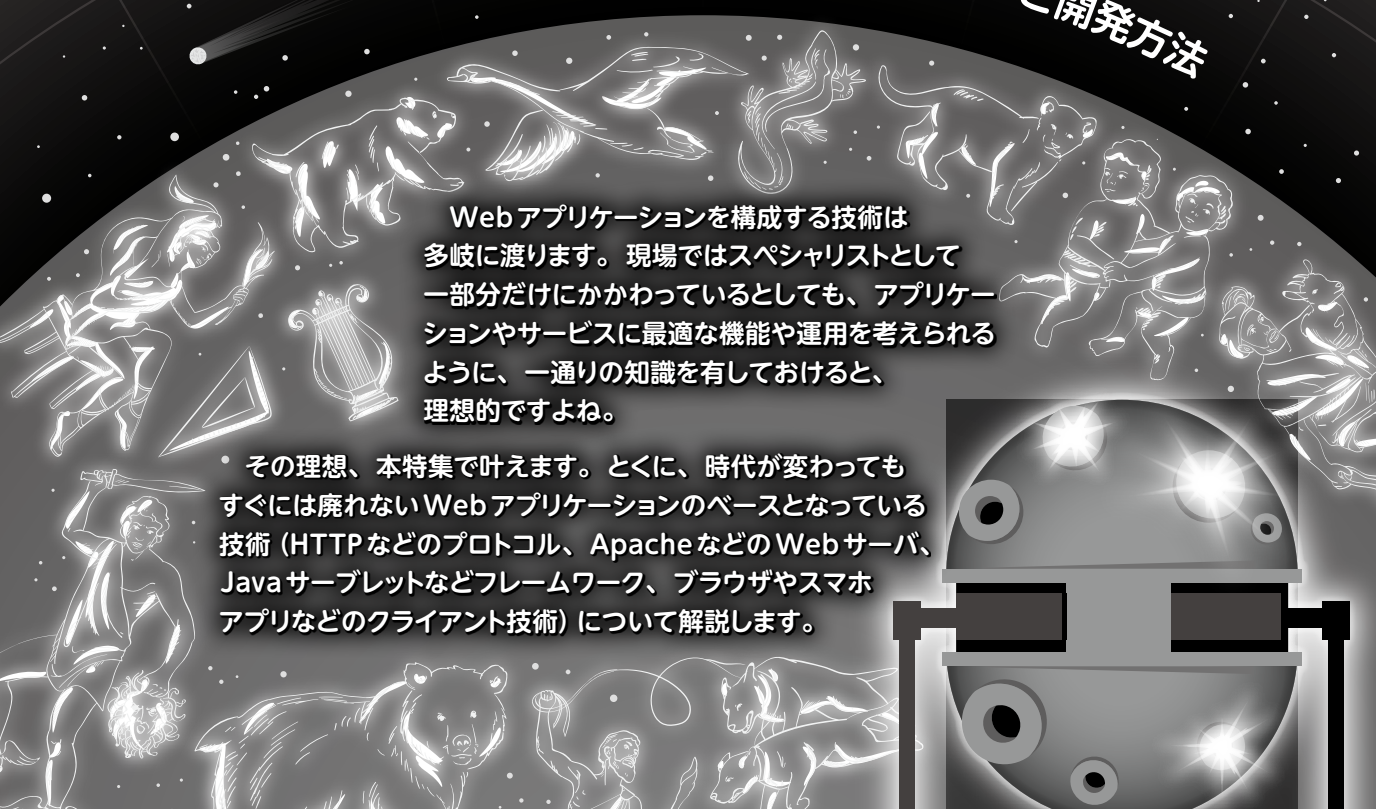
2名

第1特集

根底から理解していますか？

Web技術[超]入門

いま一度振り返るWebのしくみと開発方法



Webアプリケーションを構成する技術は多岐に渡ります。現場ではスペシャリストとして一部分だけにかかわっているとしても、アプリケーションやサービスに最適な機能や運用を考えられるように、一通りの知識を有しておけると、理想的ですよね。

その理想、本特集で叶えます。とくに、時代が変わってもすぐには廃れないWebアプリケーションのベースとなっている技術（HTTPなどのプロトコル、ApacheなどのWebサーバ、Javaサーブレットなどフレームワーク、ブラウザやスマホアプリなどのクライアント技術）について解説します。

第1章

通信すればしくみがわかる

Webサーバと Telnetしてみよう！

インターネットを勉強するときに、いろいろな通信プロトコルを学びます。そのうちの1つがTelnetです。ファイル転送のFtpなどもありますが、リモートからコンピュータに接続して操作するためのプロトコル（でありソフトウェア）です。本章ではTelnetでWebサーバとアクセスすることで、そのしくみを紐解き本質に近づきます。

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

Telnetで会話しよう

みなさんはGoogle、Twitter、Facebookと、毎日何らかのWebサイトを見ていると思います。しかし、その裏側で何が起こっているのかを気にすることはあまりないのではないでしょうか？

今回の特集では古くから標準的に使われてきたTelnet（telnetコマンド）を使ってHTTPリクエストを手書きしつつ、Webサーバとクライアントの動きを探ってみます。

Webサーバとの会話の方法

さっそくターミナルからtelnetコマンドでWebサーバと直接会話してみましょう。macOSを使用している方は、標準でTelnetがインストールされているのでそのままターミナルを起動してください。Windowsではデフォルトでは無効になっているので［コントロールパネル］→［プログラムと機能］からTelnetクライアント選択し、それを「有効」にする必要があります。

それでは、Telnetが使えるターミナルを起動して**図1**の❶から順番に入力します。入力したあとにEnterキーを2回押すとHTTP/1.1から始まるレスポンスが返ってくるはずです（**図1**）。

試しにWebブラウザからもexample.comにアクセスしてみましょう。example.comは、さまざまな設定確認のためにIANA（Internet Assigned Numbers Authority）が運営し公開しているサーバです。URLバーに“http://example.com”を入力すると、**図2**のように“Example Domain”などコマンドレスポンスと同じ文字列が表示されます。このことから、WebブラウザとTelnetは、Webサーバに対して同じリクエストを送っていたことがわかります。

同じリクエストを送っていたにもかかわらず、TelnetではWebブラウザに比べて多くの文字を入力する必要がありました。Telnetでは、Webブラウザが隠蔽（いんぺい）しているHTTP通信のリクエストを、すべて入力しなければならないためです。

それでは、これらのコマンドはどのような意味を持っていたのでしょうか？　次節からはこのリクエストの意味を解説していきます。

メールも送れるtelnetコマンド

telnetコマンドとは、RFC 854[注1]として公開されているTelnetプロトコルを実装したコマンドラインツールです。このプロトコルという

注1）URL https://www.ietf.org/rfc/rfc854.txt

▼図1　telnetコマンドでWebサーバと会話してみる

```
$ telnet example.com 80          ←❶ 接続するコンピュータを指定する
Trying 93.184.216.34...
Connected to example.com.
Escape character is '^]'.        ←❷ TCPコネクションを確立

 ❸ ここからリクエストの入力

GET / HTTP/1.1
Host: example.com      ←❹ Enterキーを2回押す（空行の送信）

HTTP/1.1 200 OK        ←❺ 以下Webサーバからのレスポンス
Cache-Control: max-age=604800
Content-Type: text/html

 ……途中省略……

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative ➡
examples in documents. You may use this
    domain in examples without prior coordination or asking for➡
permission.</p>
    <p><a href="http://www.iana.org/domains/example">More ➡
information...</a></p>
</div>
</body>
</html>
```

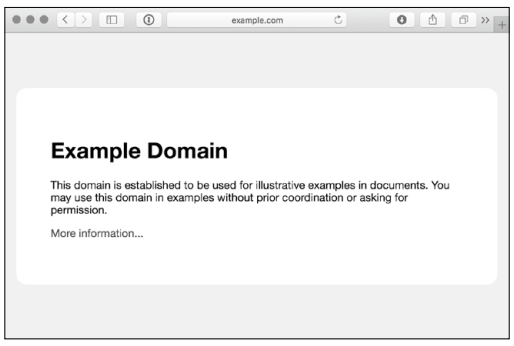
のは、ある処理を共通化し正確に実行できるようにまとめた手順書のようなもので、通信関連のプロトコルはIETF（Internet Engineering Task Force[注2]）という団体によって定められています。また、RFC（Request for Comments）というのはそれらのプロトコルを公開するためのフォーマットのことを言います。少しそれましたが、話を戻してTelnetの動きを見ていきましょう。

図1で実行したtelentコマンドでは、HTTPリクエストを送信しました。しかし、TelnetはHTTP専用ツールではなくネットワークによって接続されたコンピュータを遠隔操作するためのツールです。なぜHTTPリクエストが送信できたかというと、接続先のサーバのアドレスとポート番号を指定してTCP接続[注3]を行い、任意のリクエストを送信するという流れが、TelnetとHTTPプロトコルに共通しているためです。ほかにもSMTPプロトコル[注4]が同じ動作をするのでtelnetコマンドからメールを送信できます。

では、Telnetがわかったところで**図1**のコマンド実行の流れを振り返ってみましょう。まず、❶で接続するWebサーバのホスト名example.comとポート番号の80番を指定しています。Enterを押すとTCP接続が確立され、ホストコンピュータ（Webサーバ）へのリクエストを受け付けるようになります。❸から実際のHTTPリクエスト入力です。1行入力[注5]するごとに接続先に入力内容が送信され、HTTPリクエストを入力し終わると、❺のようにWebサーバからレスポンスが返ってきます。もし、リクエストの入力途中でHOSTをHOTなどのように入力ミスすると正しいリクエストと判断されずに“400 BadRequest”が返ってきます。

注2）URL https://www.ietf.org

▼図2　Webブラウザからアクセスしたexample.com



注3）TCPプロトコルに則り、3ウェイ・ハンドシェイクが 正常に完了した状態のこと。
注4）URL https://www.ietf.org/rfc/rfc5321.txt
注5）送信タイミングはTelnetのオプションで変更可能です。

では、どんなリクエストを送ると正常に解釈されるのでしょうか？　Webサーバが対応すべきHTTPリクエストフォーマットはHTTPプロトコルによって決められています。

そもそもHTTPプロトコルとは何だろうか?

HTTPプロトコルとはHyper Text Transfer Protocolの略で、HTML[注6]や画像などのリソースをやりとりするためのものです。この仕様はTelnetと同じくIETFによって決められています。HTTP関連のRFCは複数ありますが、そのうちのHTTP 1.1の本体はRFC 2616[注7]、HTTP 2.0はRFC 7540[注8]で公開されています。

図3はHTTPリクエストとレスポンスの構造を示しています。それでは、先ほどTelnetで送ったリクエストとこのフォーマットが一致しているか比較してみましょう。

1行目はリソースの取得を行うGETを指定しており、Webサーバ内のリソースを指定するURIは/、バージョンはHTTP 1.1を指定していることがわかります。

2行目からはヘッダ領域です。複数指定することもできますが、ここではリクエスト送信先のホストを示すHOSTだけが書かれています。これを指定することで"/"のようにURIのスキーマとホストが省略できます。ヘッダの次は空行のあとにボディ領域になりますが、ここで手書きしたリクエストにはボディがありません。この違いはリクエストメソッドによって生まれています。今回はGETメソッドのリクエストを送信していますが、GETリクエスト時にボディを送信することは推奨[注9]されていません。逆にボディを使うことが想定されているメソッドはPUTやPOSTリクエストになります。

また、ボディのほかにもメソッドによって使い方が分かれるのがURIパラメータです。URIはメソッドにかかわらず常に指定されますが、ボディとは逆にパラメータ部分はPOSTやPUTでは使用されず、GETやDELETEメソッドで使用されることがほとんどです。

続いて、HTTPレスポンスを見ていきましょう。レスポンスではリクエストと同じバージョンのHTTP 1.1、そしてステータスの200 OKが返ってきています。ステータスは100から500まで100桁ごとに大まかな意味が分かれていますが、よく使われるのは処理の正常終了を示す200番台とリクエスト内容が不正であることを示す400番台、そしてサーバ内でエラーが発生したことを示す500番台です。2行目以降のヘッダは、キャッシュ[注10]の振る舞いを制御するためのCache-Controlとボディのタイプを示すContent-Typeの2つが指定されています。

注6) Hyper Text Markup Languageの略でテキストを構造的に表現するための書式。

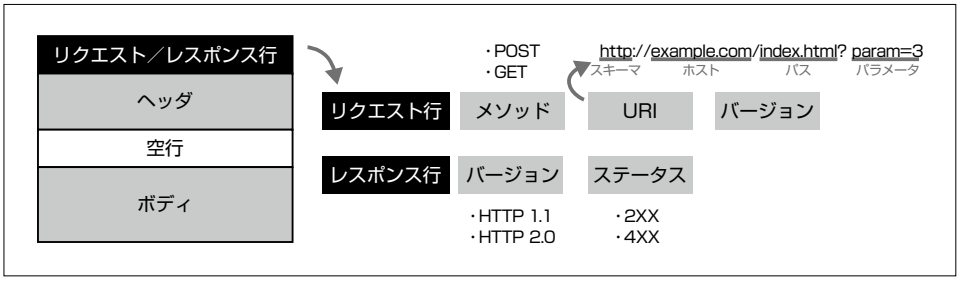
注7) URL https://www.ietf.org/rfc/rfc2616.txt

注8) URL https://www.ietf.org/rfc/rfc7540.txt

注9) 禁止はされておらず、GETリクエストにボディを付けて送ることもできますが、多くのWebサーバでは値が無視されます。

注10) 同じページを何度もリクエストするような場合、毎回Webサーバがリクエストを返していたのでは効率が悪くなってしまいます。これを避けるためにレスポンスを一定期間保存したものをキャッシュと言います。

▼図3　HTTPリクエストとレスポンスの構造と例



このように、レスポンスのヘッダも複数指定できます。

最後に、空行を挟んだボディにはContent-Typeで指定されていたHTMLが返ってきています。リクエストのように明確にボディの有無の規則は決まっていませんが、空の場合も存在します。

HTTPはステートレス?

あるショッピングサイト[注11]を仮定し、理解をより深めていきましょう。これから商品ページの商品をカートに入れていくのですが、先にカートページが**図4**のように空であることを確認します。それでは**図5**のような商品一覧ページを開きます。このページではHTMLのForm要素[注12]が使われており、キーボードの［カートに追加］ボタンを押すとPOSTリクエストでitem情報が送信され、**図6**のキーボードが追加されたカートページに遷移します。今度はWebブラウザでカートに追加するボタンを押したときのHTTPリクエストをTelnetで送信してみましょう。HTTPリクエストとそのレスポンスは**図7**のようになります（以降TelnetとWebサーバの接続処理は省略しています）。

もう一度Webブラウザに戻ってカートページにアクセスしてみましょう。結果は、前回と同じようにキーボードが入った状態が表示されます。では、Telnetではどうでしょうか。GETリクエストをカートページに送ると、「商品がありません」というレスポンスが返ってきました（**図8**）。

同じカートページ（/cart）のリソースを取得

注11）手元で動かす方法はコラムで説明しています（P.25）。

注12）HTMLのForm要素は属性のactionにPOSTを指定すると、"application/x-www-form-urlencoded"の形式でformデータを送信することが期待されています（URL https://www.w3.org/TR/html5/forms.html#the-form-element）。

▼図4　空のカートページ

▼図5　商品一覧ページ

▼図6　キーボードが追加されたカートページ

▼図7　Telnetでカートに商品を追加する

```
POST /cart HTTP/1.1
Host: localhost:7070
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

item=keyboard
HTTP/1.1 200 OK ここからレスポンス
Set-Cookie: cart=keyboard
Date: Sat, 08 Jul 2017 04:31:33 GMT
Content-Length: 741
Content-Type: text/html; charset=utf-8

<html>
<head>
  <title>cart</title>
……以降省略……
```

▼図8　Telnetだけでショッピングサイトにアクセスしカートを確認してみる

```
GET /cart HTTP/1.1
Host: localhost:7070

HTTP/1.1 200 OK
 ……以降省略……
    <tr><th>名前</th><th>価格(円)</th>
</tr>
    <tr><td>商品がありません</td><td>
</td></tr>
  </table>
  <form action="/catalog" method="GET">
    <button type="submit" name=
"purchase">商品一覧に戻る</button>
  </form>
  </div>
</body>
</html>
```

するというGETメソッドのリクエスト送っているのにもかかわらず、なぜレスポンスが変わったのでしょうか？　この違いは「カートにキーボードを追加する」というリクエストを送ったときにヘッダとして返ってきたSet-Cookieの取り扱いにあります。

このCookieというのは、RFC 6265[注13]によって定められた、HTTP通信の状態を管理をするためのものです。もともと、HTTPプロトコルでは以前にどんなリクエストを受けたかを区別せず、都度リクエストに応じたリソースを返却するだけのステートレスなプロトコルでした。しかし、そのままでは今回のショッピングカートのように以前に入れた商品を保持したりと状態を引き継ぎたいときに困ります。そこで考えられたのがCookieというユーザエージェント[注14]（以降UAとする）側に値を保持させる方法です。基本的な流れは**図9**のとおりです。まずWebサーバ側でUAに保持してもらいたい値をレスポンスのSet-Cookieヘッダを使って送信し、受信側はその値をローカルに保存します。UAが次にリクエストを送信するときは、Cookieの有効期限などの条件を満たしていればリクエストヘッダにCookieを付けてWebサーバへと送信します。

このようにCookieを送り合うことで、Webサーバは以前の状態をふまえたうえでレスポンスを返すことができます。Webブラウザでは Cookieをヘッダに添付するかの判断を行ってくれますが、Telnetではリクエストを手書きすることになるので、自分自身でCookieを送るべきか判断してリクエストを作成する必要があります。

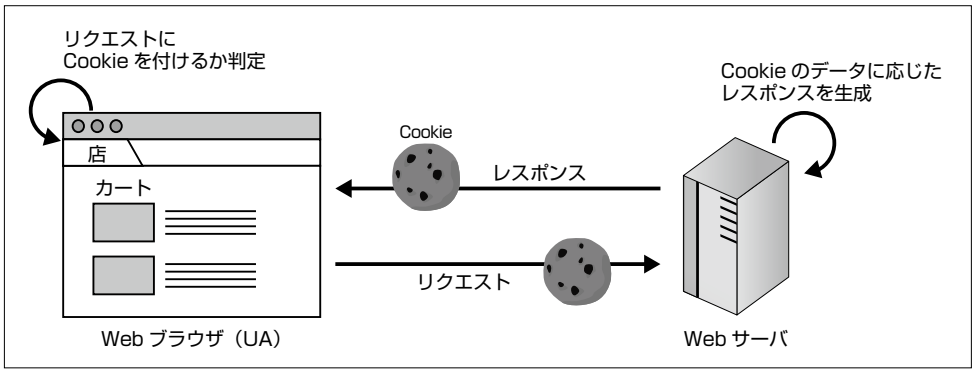
では、Cookieを付けたリクエストをTelnetから送ってみましょう。**図10**のようにヘッダにCookieを持たせると、Webブラウザのときと同じキーボードがカートに入ったページが返ってきます。

手軽に状態管理できるCookieですが、1つあ

注13）URL https://tools.ietf.org/html/rfc6265

注14）HTTPリクエストをサーバに送る側をユーザエージェントと呼びます。ここではWebブラウザがユーザエージェントにあたります。

▼図9　Cookieによる状態管理の流れ



▼図10　TelnetでCookie情報を付加してカートを確認してみる

```
GET /cart HTTP/1.1
Host: localhost:7070
Cookie: cart="keyboard"  Cookieを指定

HTTP/1.1 200 OK
Set-Cookie: cart=keyboard
 ……以降省略……
  <h1>カートの中身</h1>
  <table>
    <tr><th>名前</th><th>価格(円)</th>
</tr>
    <tr><td>キーボード</td><td>22000
</td></tr>
 ……以降省略……
```

たり最大4,096byteという制限やリクエストに常に添付されるという性質があるため、大きなデータを保持する必要がある場合やセキュリティの担保が必要な用途には向いていません。これらの用途にも対応する方法として、CookieやURIのパラメータを利用した「セッション」があります。このセッションという方法は、Webサーバ側の仕様なのでHTTPのRFC[注15]には含まれていません。そのため、利用する場合はWebサーバ[注16]に応じて仕様を確認する必要があります。セッションの細かい挙動は異なりますが、流れはどれも同じです。これをふまえてショッピングカートをセッションを使って実現する例を考えてみると、**図11**のようになります。

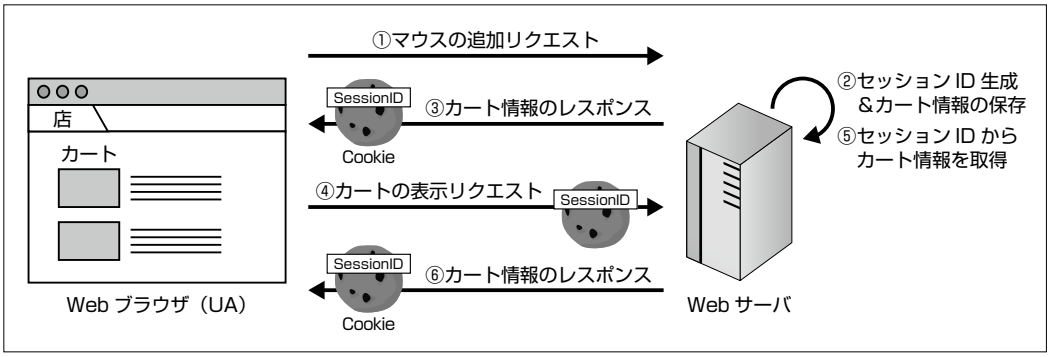
① 商品「マウス」の追加リクエスト
② 8Webサーバはセッション ID を生成し、ID に紐付けてカート情報を保存する
③ Webサーバはセッション ID をヘッダの Set-Cookie に乗せたカートページのレスポンスを送信する
④ UA はセッション ID を Cookie にのせてカートページをリクエストする
⑤ Webサーバはセッション ID から該当するカートの中身を取得する
⑥ Webサーバはカート情報に応じたレスポンスを生成して送信する

グラフィカルなWebページはどうやって表示されるのか?

これまではTelnetをWebクライアントとしてHTTP通信を試しました。そこではレスポンスが文字のまま表示されていましたが、Webブラウザではグラフィカルなページが表示されます。この差はいったい何によって生まれているのでしょうか？　この違いは、Webサーバからレスポンスを受け取ったあとの処理から探ることができます。Telnetの受信後処理は文字キャラクタ表示だけなので、どんなにきれいに装飾されているページであってもレスポンスがそのまま表示されるだけです。

一方で、Webブラウザは受信後にレスポン

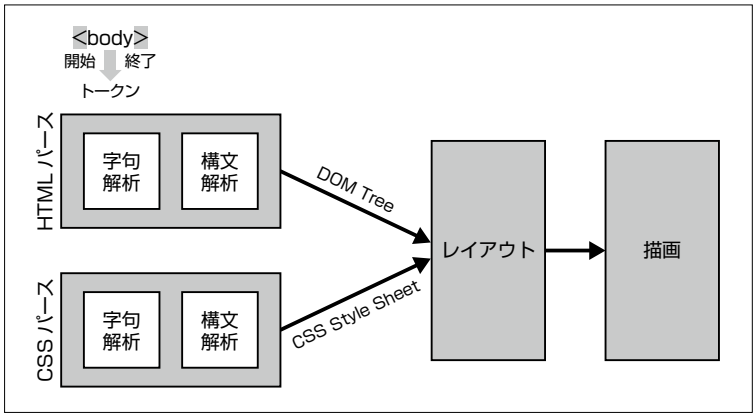
▼図11　セッションを使ったショッピングカートの商品管理



注15) セッションそのものの話ではなく、Cookieにのせるセッション識別子の話はRFC 6265に含まれています。

注16) PHPの例を挙げると標準ライブラリのドキュメントから確認する必要があります（URL http://php.net/manual/ja/refs.basic.session.php）。

▼図12　レンダリングの流れ



▼表1　ブラウザを構成するおもなパーツ

パーツ	説明
通信モジュール	HTTPプロトコルに則って通信を行う
ストレージ	Webストレージのデータ格納場所
JavaScriptエンジン	JavaScriptの解析と実行を行う
レンダリングエンジン	HTMLとCSSのパースとレンダリングを行う
UI(User Interface)	更新ボタンやブックマークバーなど

スの解析（parse）と描画処理を行います。たとえばWebページのHTMLファイルに<H1>というタグと文字列があれば、それを分析し該当する表示をブラウザは出力します。ほかにもさまざまなタグがありますが、それぞれに対応したグラフィカルなページを表示します。通信した内容にJavaScriptやJSファイルがあれば、それに対応した動作を行います（**図12**）。

ドキュメントの構造は、HTMLをDOMツリーという形で表現します。そこにスタイルシートが構造をさらに指定します（**図13**）。

では、もう少し詳しくブラウザのしくみを追ってみましょう。まず、ブラウザは**表1**のように大きく5つのパーツに分けることができます。ページの見た目を生成するのはレンダリングエンジンとJavaScriptエンジン[注17]です。

それらの仕様は、前者がW3C[注18]のHTMLとCSS[注19]、後者がECMAScript[注20]によって決められています。しかしみなさんご存じのように、ブラウザによってそれらの仕様に対応しきれていなかったり、逆に拡張されたりしていることもあります。そのため実際の動きは各エンジンの仕様を見る必要があります。

有名なものとしてレンダリングエンジンとJavaScriptエンジンの対応があります。ChromeではBlink[注21]とV8[注22]、SafariではWebKit[注23]とJavaScriptCore[注24]、FireFoxではGecko[注25]とSpiderMonkey[注26]です。

さまざまな技術とともに進化するWeb

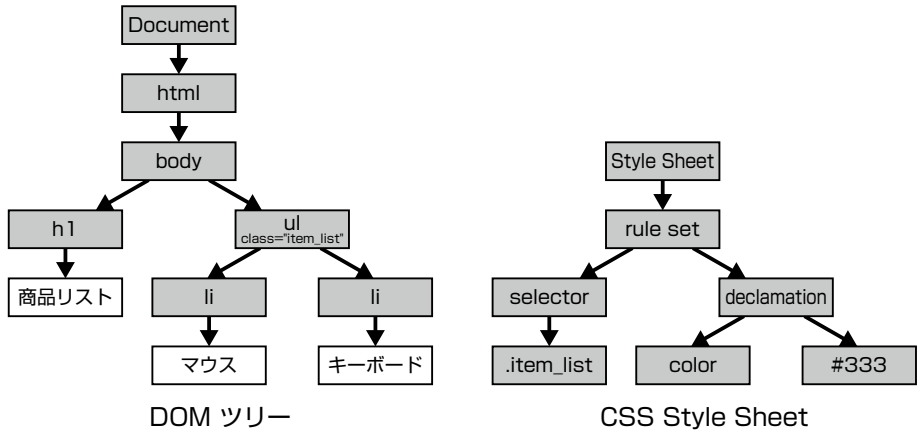
今回、Telnetやブラウザのしくみを通していろいろな技術が組み合わさってWebができていることがわかったと思います。

また、CookieはHTTPのRFC、セッションはサーバを実装する言語やライブラリのドキュメントと、それぞれの技術ごとに仕様が公開されている場所が違っていました。これらを暗記する必要はありませんが、それぞれが担当する範囲を覚えておくとより深く知りたくなったと

注17）正確にはMozillaによって実装されたスクリプト言語のことをJavaScriptと呼びますが、ここでは広義にECMAScriptの仕様に則ったスクリプト言語の意味で使っています。
注18）URL https://www.w3.org
注19）Cascading Style Sheetの略でHTMLの装飾を行うもの。URL https://www.w3.org/Style/CSS
注20）URL http://www.ecma-international.org/publications/standards/Ecma-262.htm
注21）URL http://www.chromium.org/blink
注22）URL https://github.com/v8/v8/wiki
注23）URL https://webkit.org
注24）URL https://trac.webkit.org/wiki/JavaScriptCore
注25）URL https://developer.mozilla.org/en-US/docs/Mozilla/Gecko
注26）URL https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey

きや、今後バージョンアップによって動きが変わったときに便利です。SD

▼図13　パース（parse）完了時のDOMツリーとCSS Style Sheet



・HTML ファイル（抜粋）

```
<html>
<body>
  <h1>商品リスト</h1>
  <ul class="item_list">
    <li>マウス</li>
    <li>キーボード</li>
  </ul>
</body>
</html>
```

・CSS ファイル（抜粋）

```
.item_list {
  color: #333;
}
```

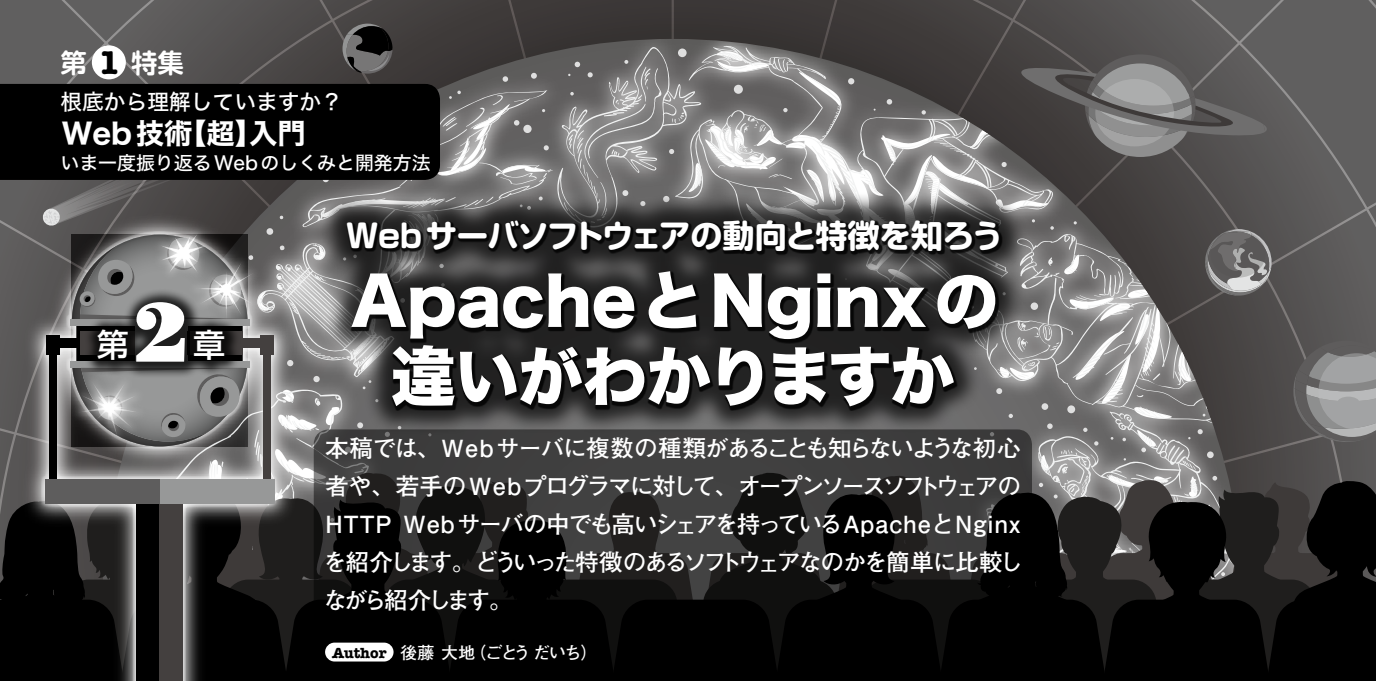
Column　仮想ショッピングサイトのインストールと実行

本原稿で作成した仮想ショッピングサイトをGitHub上に公開しました。ローカル環境でも動かしたい場合は、https://github.com/Ladicle/http-handson/releasesから該当するバイナリをダウンロードしてください。ソースコードを改造したい場合やローカル環境に該当するものが存在しない場合は、Goのソースコードからバイナリをビルドする必要があります（Goの実行環境の構築方法は“http://golang.jp/install”が参考になります）。環境構築が完了すればあとは次のコマンドを実行するだけです。“go get”を実行すると設定した$GOPATHの/src配下へソースコードがダウンロードされ、$GOPATH/bin配下にバイナリが生成されます。

```
$ go get -u github.com/Ladicle/http-handson
```

バイナリを入手できたらあとは実行するだけです。デフォルトではポート7070番で起動しますが、別のポートで起動したい場合は次のようにpオプションで変更できます。コンソールにAccess to［Webサーバのアドレス］が表示されれば起動完了です。8080番ポートで起動した場合、Webブラウザに127.0.0.1:8080を入力するとショッピングサイトのトップページが表示されます。

```
$ ./http-handson -p 8080
2017/07/10 06:46:58 Access to 127.0.0.1:8080
```



第2章

Webサーバソフトウェアの動向と特徴を知ろう

ApacheとNginxの違いがわかりますか

本稿では、Webサーバに複数の種類があることも知らないような初心者や、若手のWebプログラマに対して、オープンソースソフトウェアのHTTP Webサーバの中でも高いシェアを持っているApacheとNginxを紹介します。どういった特徴のあるソフトウェアなのかを簡単に比較しながら紹介します。

Author 後藤 大地（ごとう だいち）

Apache HTTP Server

Apache（アパッチ）HTTP Serverは、The Apache HTTP Server Project（**図1**）によって開発およびメンテナンスが実施されているオープンソースのHTTP Webサーバ（以下、Webサーバ）です。堅牢で自由に利用できるWebサーバを開発することを目的として世界中のエンジニアが協力しながら作り上げてきたソフトウェアで、開発が始まった1995年には世界中でもっとも使われているWebサーバになりました。電光石火で普及したソフトウェアです。

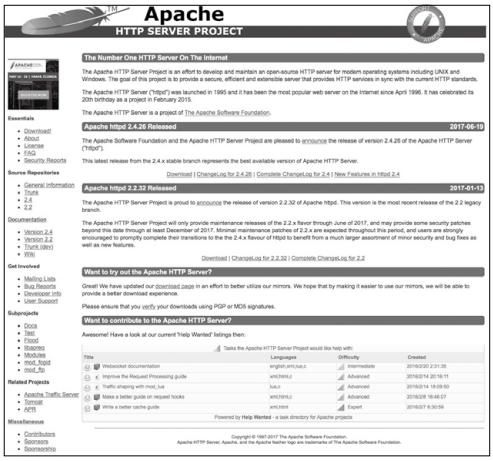
Apache HTTP Server（以降、Apache）は、2016年ごろまで世界中でもっとも使われているWebサーバの地位にありました。しかし以降、Apacheは急速にシェアを減らしていきます。Microsoft IISやNginxへシェアが移っており、今後も減少が続くのではないかとみられています。しかし現在でも多くのサイトで使われている重要度の高いソフトウェアです。

Nginx

Nginx（エンジンエックス）はApacheよりも後発のWebサーバで、2004年にIgor Sysoev氏によって開発されたソフトウェアです（**図2**）。Apacheには同時アクセス数が高くなってくると処理をさばけなくなる問題がありました。NginxはこうしたApacheの問題を解決するために開発されたソフトウェアで、同時アクセス数が多い場合でも処理をこなせるという特徴があります。

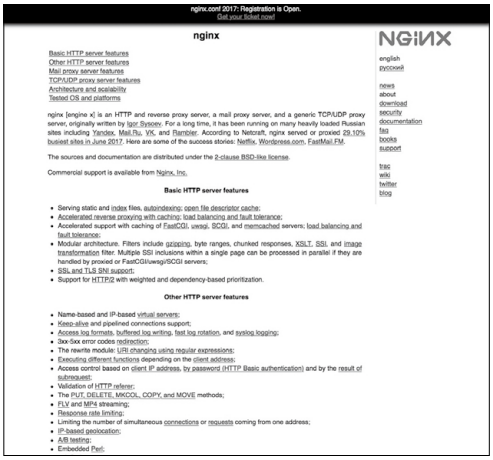
またApacheと異なり、HTTPに特化したサーバというよりも、リバースプロキシやメールプロキシサーバ、TCP/UDPプロキシなど、プロキシサーバやロードバランサとしても動作します。Nginxだけを使って処理をするというよりも、必要に応じてほかのソフトウェアと組み

▼図1　The Apache HTTP Server Project[注1]



注1）URL https://httpd.apache.org/

▼図2　Nginx[注2]



合わせて利用するタイプのソフトウェアです。高いスケーラビリティを持っていることから、規模の大きなビジーサイトでの採用が進んでいます。

もともとロシアで開発されたこともあってロシアの大規模サイトでの採用も進んでいるWebサーバです。Netflixのようなストリーミング配信を行っているベンダもNginxを採用するなどその性能には定評があります。

注2）URL https://nginx.org/en/

ApacheとNginxのシェア動向

これからWebサーバを学ぼうとしているのであれば、最近のWebサーバのシェア動向について知っておいたほうが良いでしょう。使う機会が多そうなものから優先的に学んでいったほうが効率が良いからです。

Webサーバシェアは複数のベンダや機関から定期的に報告されています。ここではNetcraft社が毎月公開しているWeb Server Surveyからデータを紹介します[注3]。**図3**のグラフは1995年から2017年6月までのWebサーバのシェアの推移を示したグラフです。

Apacheは2011年から2012年にかけてシェアを減少させはじめ、2015年から2016年にかけてはMicrosoft IISと入れ替わってシェア2位まで割合を減らしました。その後も減少傾向が続いており、このままいくとNginxともシェアが入れ替わりそうな感じです。

注3）"June 2017 Web Server Survey", Netcraft（URL https://news.netcraft.com/archives/2017/06/27/june-2017-web-server-survey.html）

▼図3　Webサーバシェア推移グラフ（June 2017 Web Server Survey, Netcraftより抜粋）

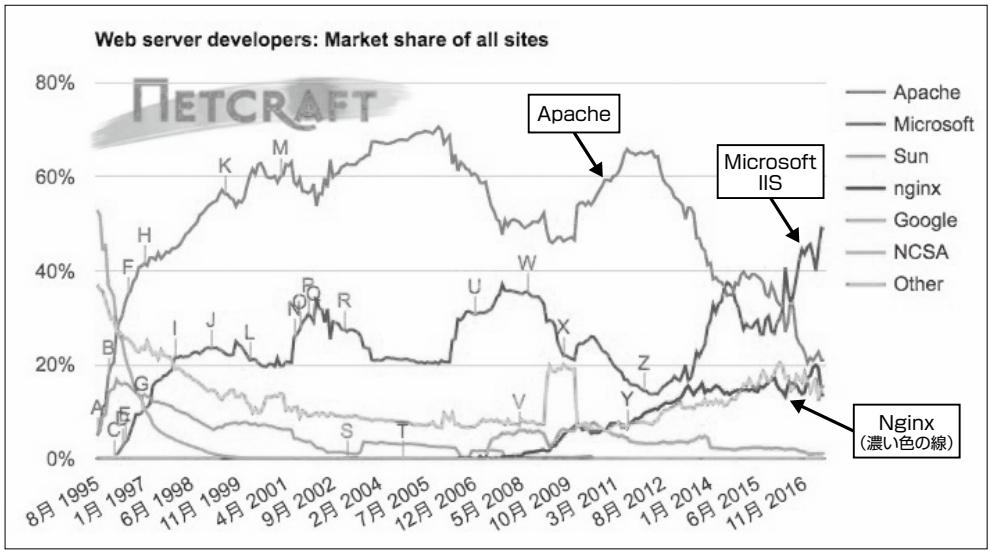


図4のグラフはビジーサイトトップ100万にしぼった場合のシェア推移グラフです。ビジーサイトではMicrosoft IISはそれほどシェアを確保できていません。この分野では依然としてApacheがもっとも高いシェアを確保しています。しかし、Apacheはトップビジーサイトでも2011年から2012年ごろ以降減少傾向を続けており、そう遠くない段階でNginxに抜かれそうな傾向を示しています。

現在の動向が変わるといった強い理由は今のところ見つかりませんので、今後も現在の動向が続くのではないかと思います。一般的にはMicrosoft IISが、ビジーサイトではNginxが採用されるといった形になりそうです。全体のホスト数の増加と割合の変化を見ていくと、これまでApacheを使ってきたサイトやサービスがMicrosoft IISやNginxへ切り替えているといった動きが見えてきます。

ApacheとNginxの機能比較

ApacheとNginxでは登場してきた背景が違いますので、それぞれ特徴が異なっています。

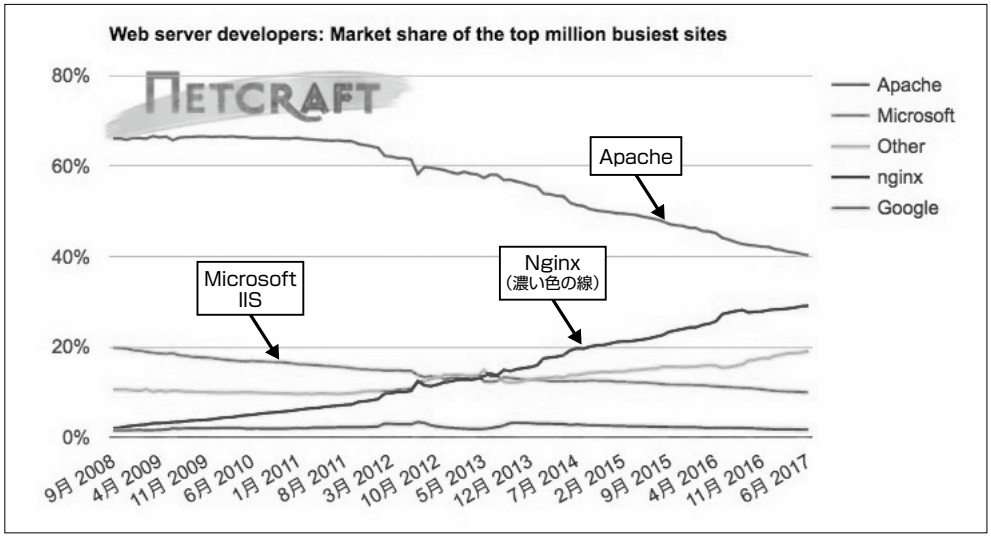
機能豊富なApache、シンプルで扱いやすいNginx

Apacheはモジュールを追加することでさまざまな動作を行えるWebサーバとして機能します。これと比べるとNginxは作りがシンプルで単体での機能はすっきりしたものです。必要があればほかのソフトウェアに処理を割り振る構造をとっています。

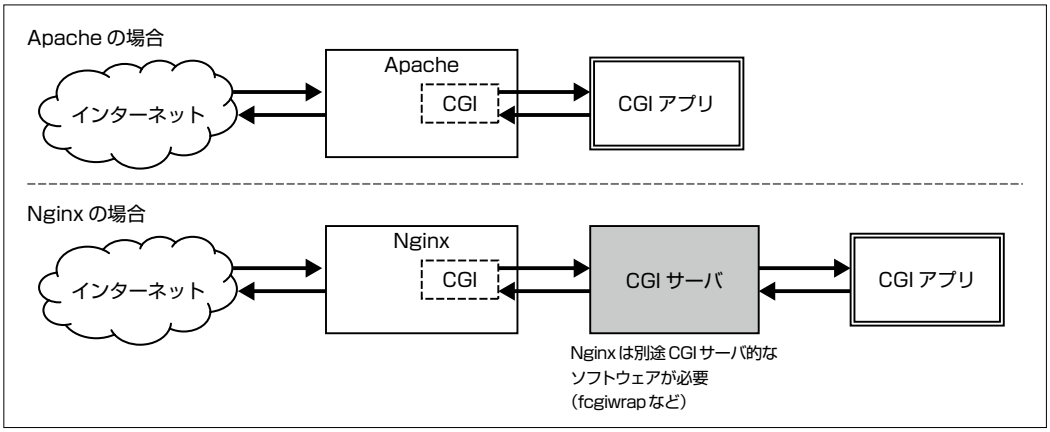
PHPやCGIを利用するケースを考えるとわかりやすいと思います。Apacheではこれらに対応した機能やモジュールが最初から用意されていますので、あとは設定を書けば利用できます。

一方、Nginxにはそうした機能がありません。NginxはFastCGIには対応しているので、この機能を経由してほかのソフトウェアへ処理を振ります（**図5**）。このため、NginxでCGIやPHPを使おうとした場合、単体で利用できるApacheと比べると設定や構築が難しい印象を受けると思います。しかし、Nginxは後発だけあって構造がシンプルで設定ファイルもわかりやすく、いったん動作させれば扱いやすさに気がつくと思います。スケーラビリティも高く、実用的なソフトウェアです。

▼図4　ビジーサイトトップ100万におけるWebサーバシェア推移グラフ（June 2017 Web Server Survey, Netcraftより抜粋）



▼図5　CGIを利用するケース



ApacheとNginxの使い分け

逆に、Apacheをリバースプロキシサーバ的に使うといったこともできます。しかし、そもそもWebサーバとして開発が始まったソフトウェアですので、Nginxのようにシンプルにはできません。やはりそれぞれに得意な部分があるので、無理に本来の用途以外の使い方をしてもあまりお得なことはありません。

なお、ApacheとNginxは完全に入れ替える関係のソフトウェアかと言えば、そういうわけでもありません。フロントエンドをNginxにしておいて、背後に必要な機能を有効にしたApacheを動かしておき、そちらに処理を振るという使い方もできます（**図6**）。

メモリ消費量の少ないNginx

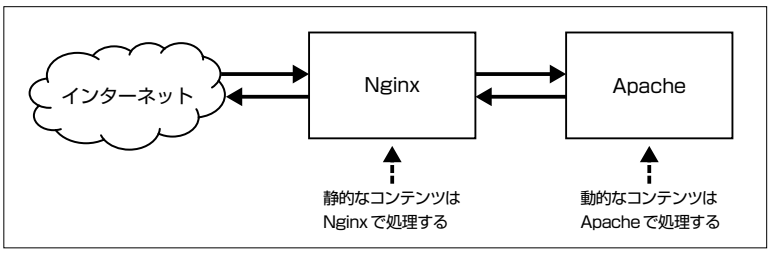
メモリの消費量という面で見ると、Nginxのほうがapacheよりもメモリを消費しない傾向がみられます。動作もNginxのほうが軽快ではあります。IoTデバイスのようにメモリ量の限られた環境や、かなり制限された仮想ホストなどで動作させる必要がある場合にはNginxが便利なシーンが増えてくるでしょう。

Apacheの長所と短所

Apacheは世界中でもっとも長い間ナンバーワンシェアを確保していたWebサーバであり、現在でもかなりの数のサイトで使われているソフトウェアです。Apacheに関するドキュメントの数はかなりの量にのぼりますし、運用のノウハウも溜まっています。何か問題が発生した場合に役に立つ情報が手に入りやすい状況にありますので、運用するにあたっては心強い状況です。

一方、Nginxが開発された理由がそうであるように、Apacheは同時アクセス数が多くなってくると処理をさばけなくなってくるという問題を抱えています。Apacheでもこうした問題を解決するた

▼図6　フロントエンドでNginxを動かし、背後でApacheを動かす



めの取り組みは行われていましたので、機能を有効にして適切に設定を行えば性能上限をさらに引き上げることが可能です。しかし、Nginxではそうしたチューニングを行わなくても性能がスケールしやすいと言われています。

Nginxの長所と短所

これはNginxに限ったことではありませんが、後発のソフトウェアは先行しているソフトウェアの利点や問題点を整理して開発が行われますので、何かと便利な状況になることが多いです。Nginxの場合はビジーサイトでも処理がさばけるように構造が変更されていますし、設定ファイルも読みやすくなっています。Nginxの設定ファイルは**リスト1**のような構造になっています。

Nginxの設定ファイルはかなり整理されていますので、Apacheのときのように設定ファイルをいろいろ変更していったら意味がわからなくなってしまった、といったような状況は起こりにくいと思います。

Nginxのわかりにくいところは、Nginx自体は処理の割り振りや静的コンテンツの配信用に開発されていますので、Nginx単体では用意が終わらないことがあるという点でしょう。PHPやCGIを利用しようとした場合にはそれ用に設定をしないといけませんし、Nginx以外のソフトウェアのセットアップも必要になります。慣れないうちはこのあたりの設定が煩雑に感じられるのではないかと思います。

ApacheとNginx、どっちを選ぶべき？

最近Web開発を始めたプログラマからみた場合、ApacheとNginxの違いは設定ファイルと制御方法（設定の再読み込み、サーバの再起動など）、開発したアプリケーションのデプロ

▼リスト1　Nginx設定ファイルの例（nginx.conf）

```
 1: user  web web;
 2: worker_processes  1;
 3:
 4: events {
 5:     worker_connections  1024;
 6: }
 7:
 8:
 9: http {
10:     include       mime.types;
11:     charset       UTF-8;
12:     charset_types text/plain;
13:     default_type  application/octet-stream;
14:
15:     sendfile        on;
16:
17:     keepalive_timeout  65;
18:
19:     gzip  on;
20:
21:     server {
22:         listen       80;
23:         server_name  192.1.1.50;
24:
25:         location / {
26:             root     /home/web/contents/;
27:             autoindex on;
28:         }
29:
30:         error_page   500 502 503 504  ↵
/50x.html;
31:         location = /50x.html {
32:             root   /usr/local/www/nginx-dist;
33:         }
34:
35:     }
36:
37:     # HTTPS server
38:     #
39:     #server {
40:     #    listen       443 ssl;
41:     #    server_name  localhost;
42:
43:     #    ssl_certificate      cert.pem;
44:     #    ssl_certificate_key  cert.key;
45:
46:     #    ssl_session_cache    shared:SSL:1m;
47:     #    ssl_session_timeout  5m;
48:
49:     #    ssl_ciphers  HIGH:!aNULL:!MD5;
50:     #    ssl_prefer_server_ciphers  on;
51:
52:     #    location / {
53:     #        root   html;
54:     #        index  index.html index.htm;
55:     #    }
56:     #}
57: }
```

イ方法といったところでしょう。いったんセットアップしてしまえばそもそもWebサーバに触ることはあまりありませんし（そしてないほうが何かと幸せな状態です）、違いが気になることはないと思います。

ところで現在もっとも高いシェアを確保しているMicrosoft IISですが、こちらはホスティングサービスを提供しているベンダなどが一気に大量のホストを提供するといった目的で使われていることが多いようで、実際にMicrosoft IISを操作することになるエンジニアの数はそれほど多くないのかもしれません。

こうした状況を鑑（かんが）みると、これから初めてWebサーバについて調べてみるとか、自分でセットアップしたことがなかったのでセットアップしてみるということであれば、Nginxを選択しておくのが無難ではないかと思います。Nginxは現在も活発に開発が続いていますし、最新機能の実装も積極的です。Apacheはかなり枯れてきており、Nginxほど積極的には開発は行われていません。Nginxを使えるようにしておけばこれから潰しがききやすいと思われます。

ただし、既存のサーバの管理などの業務が降ってきた場合、レガシーになりつつあるApacheの設定ファイルを書き換える必要に迫られることも、今後まだまだ発生するのではないかと考えられます。ですので、Nginxを学んでまだ余裕があるのであれば、次にApacheの設定ファイルの書き方などを勉強しておくと良いと思います。

Nginxのセットアップサンプル

ここでは簡単にですが、Nginxをセットアップする方法を紹介しておきます。セットアップの種類としては（1）配布物に同梱されているデフォルトページの表示、（2）自分で用意した静的コンテンツの表示、（3）ほかのサイトを使ったリバースプロキシとして動作させる方法、の3つを取り上げます。Nginxでどういったことができるのかを簡単にですが、ざっと見通せるようになるのではないかと思います。

インストール

まず、Nginxをインストールします。macOSであればHomebrew、Linuxディストリビューションであればyumかapt、FreeBSDであればpkgといったように、プロジェクトやベンダが提供しているパッケージ管理システムを使うのが良いでしょう。たいていのディストリビューションにはNginxのパッケージが用意されていますので、それを使ってください。

次に設定ファイルを編集します。パッケージ管理システムからインストールした場合、デフォルトの設定ファイルがどこかにインストールされます。macOS/Homebrewの場合には/usr/local/etc/nginx/の下に、Linuxディストリビューション系では/etc/nginx/の下のどこかに、FreeBSDの場合には/usr/local/etc/nginx/の下にインストールされます。

静的コンテンツを配信するWebサーバとして動作させる

デフォルトページの表示

設定ファイルの名前はパッケージ管理システムごとに異なっています。だいたいはnginx.confといった名前になっています。デフォルトの設定がdefault.confといった名前で用意されていたり、nginx.conf.defaultやnginx.conf-distといった名前で用意されていたりします。このあたりは使っているパッケージ管理システムに合わせて読み替えてください。

インストールされるデフォルトの設定ファイルを標準の設定ファイルにコピーしたあとで、不要な設定を削除して**リスト2**のような感じに書き換えます。

上記設定ファイルがNginxを、静的コンテンツを配信するWebサーバとして動作させるための基本的な設定となります。設定内容の詳しい説明はマニュアルを読んでほしいのですが、とくに次の部分を押さえておいてもらえればと

▼リスト2　Nginxの設定ファイル（nginx.conf）

```
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include       mime.types;

    server {
        listen       80;
        server_name  192.168.185.50;

        location / {
            root   /usr/local/www/nginx;
            index  index.html index.htm;
        }
    }
}
```

▼図7　Nginxに含まれているデフォルトのWebページ

▼リスト3　Unicodeのビールのアイコンを表示するHTML（index.html）

```
<!DOCTYPE html>
<html>
<head>
<title>Love Beer</title>
</head>                          Unicodeに登録されているビールアイコン
<body>                                    ↓
<p style='text-align:center;font-size:1500%'>🍺</p>
</body>
</html>
```

思います。

・server_name：サーバ名。IPアドレスか名前解決できるホスト名
・root：コンテンツがおいてあるトップディレクトリ
・index：URLにファイルが指定されていない場合に仮定するファイル名

ここまで設定したらNginxに設定を反映させるために、設定ファイルの再読み込みまたはNginxの再起動を実施します。Nginxの起動や停止、再起動、再読み込みはsystemdが導入されているLinuxディストリビューションではsystemctl(1)コマンドを使い、それ以前のLinuxディストリビューションやFreeBSDならservice(1)コマンド、macOS/Homebrewならnginxコマンドを直接実行します（**表1**）。

この状態で設定したホストにアクセスすると**図7**のようなページが表示されます。これがNginxがデフォルトで提供しているWebページです。

自分で用意した静的コンテンツの表示

Nginxでは静的コンテンツを配信するだけであれば設定はとてもシンプルです。たとえば、**リスト3**のようなHTMLファイルを用意したとします。

このファイルを/home/www/index.htmlとしてデプロイします。そうしたらNginxの設定ファイルを**リスト4**のような感じで用意します。

さっきとの違いはrootで指定するディレクトリが変わっている点（**リスト4-①**）と、indexで指定するファイル名がindex.htmlになっている点（**リスト4-②**）だけです。静的コンテンツの配信だけならNginxの設定ファイルはかなり

▼表1　Nginxの起動、停止、再起動、再読み込み

操作	systemctlを使う方法	serviceを使う方法	nginxを直接実行する方法
起動	systemctl start nginx	service nginx start	nginx
停止	systemctl stop nginx	service nginx stop	nginx -s stop
再起動	systemctl restart nginx	service nginx restart	nginx -s stop;nginx
設定ファイル再読み込み	systemctl reload nginx	service nginx reload	nginx -s reload

▼リスト4　リスト3のHTMLファイルを表示するためのNginx設定ファイル（nginx.conf）

```
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include       mime.types;

    server {
        listen       80;
        server_name  192.168.185.50;

        location / {
            root   /home/www/; ←❶
            index  index.html; ←❷
        }
    }
}
```

シンプルなものになります。こうした設定を追加していくだけです。この設定ファイルを反映させた状態で先ほどと同じようにホストにアクセスすると、**図8**のようにビールの絵文字が拡大された状態で表示されることを確認できます。

リバースプロキシとして動作させる

最後は、NginxをWebサーバとしてではなく、Webサーバのリバースプロキシとして動作させる例を紹介します。この場合の設定は、静的コンテンツを配信するWebサーバとしてセットアップするよりもシンプルです。**リスト5**のように設定を記述します。

▼図8　用意したコンテンツが閲覧できることを確認

▼リスト5　Webサーバのリバースプロキシとして機能させる場合のNginxの設定ファイル（nginx.conf）

```
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    server {
        server_name http://192.168.185.50;

        location / {
            proxy_pass http://gihyo.jp/;
        }
    }
}
```

この設定ファイルで、server_nameで指定したホストに対するアクセスがhttp://gihyo.jp/に対するアクセスとして扱われるようになります。この設定を反映させた状態でホストにアクセスすると、**図9**のようにgihyo.jpのページを得ることができます。

アドレスバーに表示されているIPアドレスが設定ファイルに書いてあるホストのIPアドレスになっていることに注目してください。アクセスはあくまでもローカルネットワークのホストに対して実施しているものの、コンテンツはgihyo.jpから引っ張ってきたものが表示され

▼図9　リバースプロキシとして機能しているNginx

ています。これはNginxがリバースプロキシとして動作することで実現しています。

Nginxはこのように通信に対して透過的に処理を行わせることができます。設定ファイルもよくできていて、慣れてくると設定ファイルを書きながら整理するといったこともできるようになると思います（ちなみに上記の書き方だと対象のサイトによってはリダイレクトしてしまうので、その場合には多少設定をそのサイトに合わせて変えてあげる必要があります）。

こんな感じでNginxでは簡単にリバースプロキシとして機能させられます。同様に、ロードバランサとして動作させるのも簡単です。いったんこうした設定方法に慣れてしまうと、Apacheの設定はかなりの煩雑感を覚えてきますが、どのみちどっちも慣れの問題です。どちらも使えるようになっておいて損はないでしょう。Nginxを使ったことがない場合には、こんな感じで一度試してもらえればと思います。SD

注4）URL https://www.litespeedtech.com/products/litespeed-web-server

Column Apacheを使い続けたい！

Apacheは10年以上に渡ってもっともシェアの高いWebサーバとして存在してきました。現在でもかなりの数のサイトがApacheで動作していますし、その設定ファイルの数は相当なものになるはずです。これを全部Microsoft IISやNginxに移行させるというのは、かなり骨の折れる作業です。

しかし、同時アクセス数が増えるなどして処理が重たくなってきた場合、やはりNginxなどのより高い性能が期待できるソフトウェアへ移行させる必要がでてくるでしょう。そうなると設定を一からやりなおす必要が出てくるわけですが、Webサーバの場合はこの設定そのものがプログラミングのように重要な部分になってくるわけで、いきなりNginxの設定ファイルとして書けと言われても無理なこともあるでしょう。

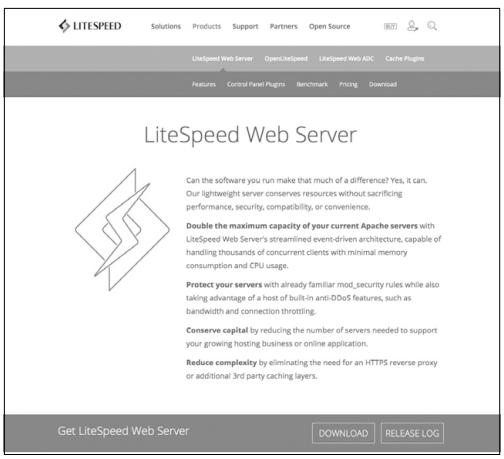
とくに、すでに辞めたり転職したりした担当者が作成したとみられる設定ファイルを読んで理解し、それをNginxの設定ファイルとして起こしなおすというのは面倒くさい以外のなにものでもありません。できればApacheのまま性能を上げることはできないかと考えるのは人間の性というものです。

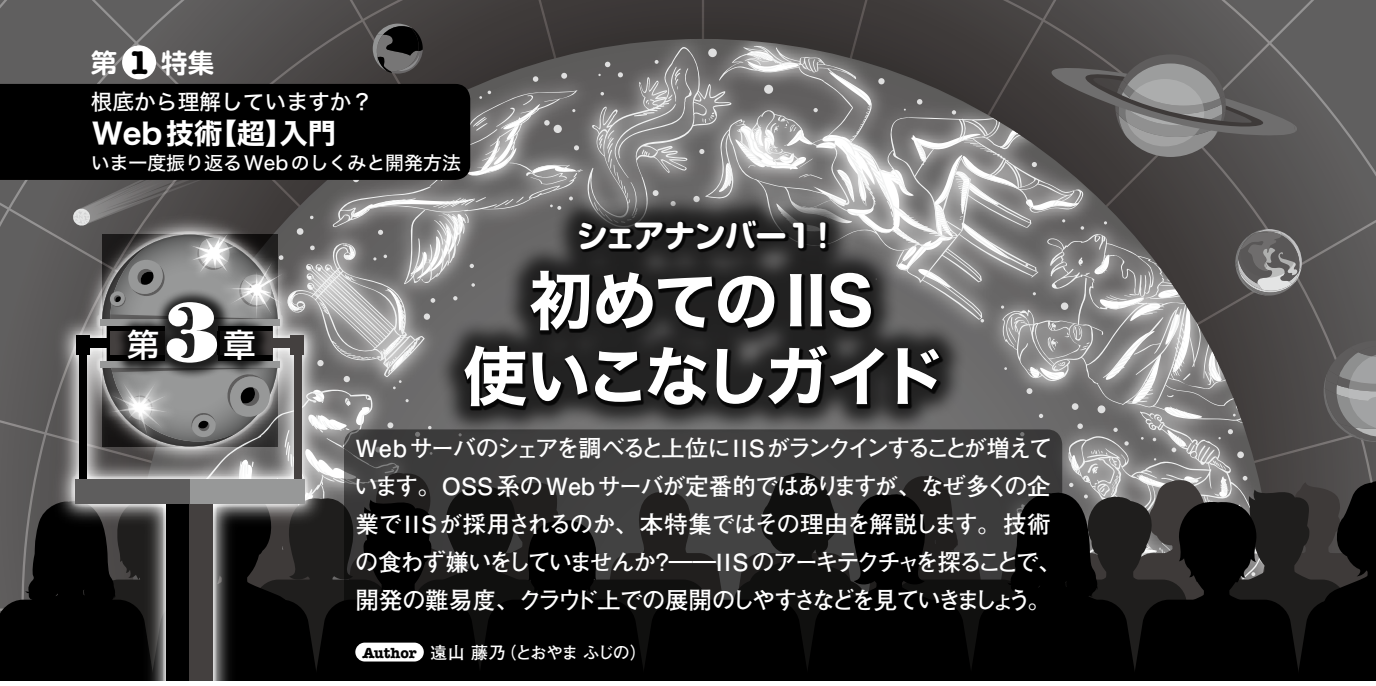
そんな場合に知っておきたいのがLiteSpeed Web Serverです（**図10**）。

LiteSpeed Web ServerはApacheの設定ファイルと互換性があります。ApacheからそのままLiteSpeed Web Serverに置き換えることができるというわけです。LiteSpeed Web ServerはApacheよりも同時アクセスのキャパシティが大きいため（2倍と謳われています）、入れ替えるだけで上限を引き上げられるとみられています。

LiteSpeed Web ServerはNginxと同じイベント駆動型の作りを採用したことで、Apacheでやってくるスケールの限界を引き上げているようです。執筆現在ではLinuxとFreeBSDのバイナリが提供されています。現在動作しているApacheを設定ファイルや構成などを変えることなく、サーバのアップグレードやハードウェアの入れ替えといった作業も発生させずに、なんとかサーバを入れ替えるだけで上限を引き上げ、そして作業を終わりにさせたい、そういった場合にはLiteSpeed Web Serverは魅力的な選択肢だと言えます。

▼図10　LiteSpeed Web Server注4





第3章

シェアナンバー1！

初めてのIIS使いこなしガイド

Webサーバのシェアを調べると上位にIISがランクインすることが増えています。OSS系のWebサーバが定番的ではありますが、なぜ多くの企業でIISが採用されるのか、本特集ではその理由を解説します。技術の食わず嫌いをしていませんか?——IISのアーキテクチャを探ることで、開発の難易度、クラウド上での展開のしやすさなどを見ていきましょう。

Author 遠山 藤乃（とおやま ふじの）

IISの特徴

Webサーバは機能が単純なため、製品間の差別化は難しいのが実情です。ではIIS（Microsoft Internet Information Services）が長年にわたって使い続けられている理由は何でしょう？　ここではまず、IISの特徴を次の3点について考えてみます。

・信頼性
・開発生産性
・運用の容易さ

「信頼性」の高さを支える技術

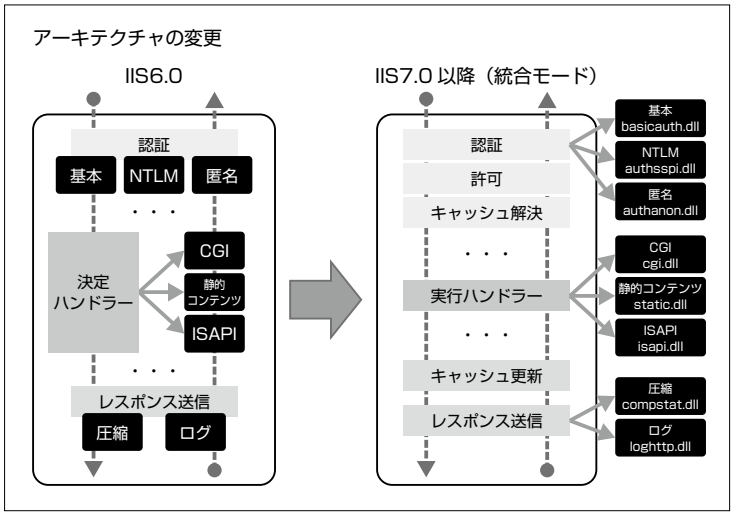
IISはWindowsだけで稼働するサービスで、LinuxやUNIXでは動きません。ApacheやNginxは稼働プラットフォームを選ばない反面、どのバージョンをどのプラットフォームで実行すれば、誰が、どのくらいの費用で、どのくらいの期間サポートしてくれるのかが明確ではありません。IISの信頼性はWindowsの信頼性と同義です。セキュリティ更新プログラム提供などのIISのサポートは、Windowsの一部としてMicrosoft社から、製品出荷後10年間提供されます。また、Windows Server 2008以降のサーバOSでは、さらに＋6年間サポート期間を延長するPremium Assuranceというオプションサービスを購入できます。つまり、最大16年間、同じバージョンのIISを安心して使い続けられます。このような長期サポートの選択肢は、ほかにはないIISの特徴です[注1]。

特徴とセキュリティについて

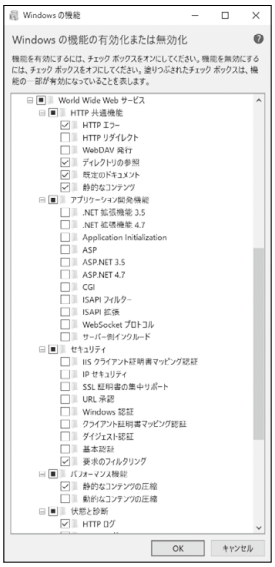
Windows Vista/Windows Server 2008以降のWindowsは、それより前のバージョンのWindowsと比較して、安全なOSとなっています。それ以前のWindowsは、利用者の利便性を最優先に設計されており、高度な機能が標準の設定で使える状態になっていました。つまり、悪意のある人物からの攻撃を受けやすく、結果、以前のWindowsはさまざまなセキュリティインシデントを起こすことになりました。このころの印象を持っているIT関係者は、Windowsを敬遠し、その上で動作するIISを知る機会さえないという状況が長く続きました。しかし、Windows Vista/Windows Server 2008以降のIISは、標準で脆弱性の可能性が小さい状態（標準機能を少なくする設定。"Secure by Default"と呼ばれる製品思想）で提供され、脆弱性は他

注1）Windows Server Premium Assurance - 6年間を製品サポートのライフサイクルに追加
URL https://www.microsoft.com/ja-jp/cloud-platform/premium-assurance

▼図1　IISのアーキテクチャ



▼図2　Windows 10でのIISの機能の有効化



製品と比較しても少なくなっています。

また**図1**で示すように、IISのアーキテクチャもIIS 7以降とIIS 6以前では大きく異なります。

IIS 6以前では、“認証”機能ブロックと、“決定ハンドラー”機能ブロックの両方に認証機能が実装されるなど、複雑で自由度の低いアーキテクチャでした。IIS 7以降では、シンプルで自由度の高い構成に変更されました。Windows 10のIISを有効化する画面（**図2**）で、各種機能を選択したことがある方もいると思います。これが**図1**の右側の機能ブロック（各種機能を提供するDLL）に該当します。

なお、IIS 7以降のアーキテクチャは**図1**のままで、大きな変更はありません。ネット上の各種情報がIIS 7以降であれば、大枠は現在の最新バージョンであるIIS 10でも同じと考えて差し支えありません。

「開発生産性」の高さを支える技術

IISは、Microsoft社の開発ツールであるVisual Studio（以下VSと略）と、開発フレームワークのASP.NETを組み合わせることで、高い生産性を提供します。またIISはASP.NETだけでなく、**図1**の“実行ハンドラー”に各種言語の処理系を追加することで、PHP、Pythonなど、さまざまな言語に対応します。**表1**で示すようにIISで提供される開発スタイルは、①Webページ、②ASP.NET Webフォーム、③ASP.NET MVCの3つに分類できます。

Visual Basic（以下VBと略）を使い、ボタン

▼表1　IISでの開発手法分類

分類	経験のある開発フレームワーク	開発スタイル	難易度
①Webページ	クラシックASP、PHP	HTMLとプログラムコードが1つのファイルに混在	初心者と中級
② ASP.NET Webフォーム	Winフォーム、WPF（Windows Presentation Foundation）、.NET Framework	高機能のライブラリとコントロールでHTMLを隠蔽し、迅速に開発	中級レベル、高度なRAD
③ ASP.NET MVC	Ruby on Rails、.NET Framework	HTMLによる完全な制御。コードとHTMLを分離。簡便なテスト。モバイルアプリや、シングルページアプリでの最良の選択肢	中級、上級

やドロップダウンリストといった部品（コントロール）を画面に貼り付け、イベントに対してロジックを記述していくスタイルに慣れている場合は、②ASP.NET Webフォームを選択するのがベストです。IIS + ASP.NET Webフォーム + VSの環境は、VBの開発スタイルが好きな方には最適です。IISを利用する最大のメリットは、ASP.NET Webフォームが使えることだと言ってよいかもしれません。ASP.NET WebフォームによるVBライクな開発がどのようなものかは、次のURL[注2]を参照ください。かなり古い内容ですが、参考になると思います。

開発時の注意点

開発生産性が高いASP.NET Webフォームですが、1点注意があります。ASP.NET Webフォームは、Post Backと呼ばれる方法で、頻繁にWebブラウザとWebサーバ間で通信が発生します。また、これらの動作が隠蔽（いんぺい）されるため、敬遠される場合があります。

VBの開発スタイルを知らず、これからIISでWeb開発をされる場合は、③のASP.NET MVCを勧めます。Microsoft社からの新機能の提供は、ASP.NET MVCに集中しているうえ、ASP.NET MVC であれば、オープンソースのASP.NET Coreを利用することで、Linuxなどのほかのプラットフォームでも利用できます[注3]。

開発生産性の議論は、個人の主観によるところが大きいため、①Webページや、③ASP.NET MVCでの開発生産性におけるIIS + VSの優位性を明示するのは難しいですが、IIS + VSの生産性の高さを理由に、IISを選択される場合も少なくありません。以前は、VSが比較的高価なため、個人にはハードルが高かったのですが、現在は商用利用でなければ、フルセットのVSを無償で利用できます[注4]。時間のある際に、ぜひIIS + VSの生産性の高さを試してください。

「運用の容易さ」を支える技術

使い慣れたWindowsと同じ方法で運用できることもIISの特徴です。各種設定作業はすべて“インターネットインフォメーションサービス（IIS）マネージャー”のGUIを通して直感的に行うことができます（**図3**）。

IISでは、1サーバで複数の異なるWebサイトをホストすることを想定して、“アプリケーションプール”と呼ばれる独立した実行環境を用意します。このアプリケーションプール単位に、CPUやメモリの使用量を制限できます。また、質の悪いアプリケーションが、一定回数のエラーを発生させた場合に、自動的にアプリケーションを停止させる設定もできます。これらの機能により、VMwareやHyper-Vなどのサーバ仮想化技術を使うことなく、サーバリソースを効率的に活用しながら、複数のWebアプリを安全に稼働させることができます。

ARRとURL Rewrite Module

さらに、Application Request Routing（ARR）[注5]

注2） 10行でズバリ!! ASP.NET WebフォームによるWebアプリケーション開発（VB） URL https://code.msdn.microsoft.com/10-ASPNET-Web-Web-VB-355a76d3

注3） Set up a hosting environment for ASP.NET Core on Linux with Nginx, and deploy to it URL https://docs.microsoft.com/ja-jp/aspnet/core/publishing/linuxproduction

注4） Visual Studio Community 2017のダウンロード先 URL https://www.visualstudio.com/downloads/）

注5） Application Request Routing URL https://www.iis.net/downloads/microsoft/application-request-routing

▼図3　IISマネージャー

とURL Rewrite Module[注6]を利用することで、複数のIISを束ね（スケールアウトと言います）て、1つの大きなサイト（Webファームと言います）としてサービスを提供できます。ARRはWebファームを構成するだけでなく、コンテンツのキャッシュとしても動作します。

小規模から大規模まで、一貫した機能が提供され、かつ、使い慣れたGUIで操作できることは、IISを選択する理由の1つです。

一方で、GUIでは複数のWebサーバをリモートで運用できないというデメリットもあります。そこで、GUIでできる作業は、すべてCLI（PowerShellベース）で行うこともできるようになっています[注7、注8]。

IISを稼働させるプラットフォームの選択

IISはWindowsでしか稼働しませんが、稼働させるにあたっては、複数の選択肢があります。ここでは、次の選択肢について記述します。みなさんの要件に合わせて稼働環境を選択してください。

- Windows クライアントOS＋IIS Express
- Windows Server＋IIS
- Container上のIIS環境
- Azure App Service

Windows クライアントOS＋IIS Expressの場合

IIS Expressは、開発に特化したIISのサブセットです。フルセットのIISは、OSのサービスとしてバックグラウンドで稼働しますが、IIS Expressは、通常のアプリケーション（ユーザ権限）で稼働します。管理ツールも“IISマネージャー”（**図3**参照）のようなGUIの管理ツールはなく、直接設定ファイルを編集することで管理します。IISの追加機能（**図2**参照）は、“Web Platform Installer”で個別にインストールします（**図9**、**図14**参照。インストール方法については後述）。

IIS Expressは、VS（Visual Studio）などから、テスト実行する際だけ起動し、必要がなくなれば、すぐに終了させることができます。フルセットのIISよりも少ないリソースで利用でき、開発には最適です。

なお、Windowsクライアント OSには、フルセットのIISをインストールすることもできます。もしフルセットのIISが必要な場合は、“コントロール パネル”の“Windowsの機能の有効化または無効化”（**図2**参照）から追加するか、PowerShellを管理者権限で起動し、

```
Enable-WindowsOptionalFeature - Online ↵
-FeatureName IIS-WebServerRole
```

を実行するだけでセットアップが完了します（**図4**）。

Windowsクライアント OS上のIISと、Windows Server上のIISの主な機能差は、同時アクセス数くらいです。同時アクセス数の上限は、Windowsクライアント OSのエディションで異なり、3～10の同時アクセスをサポートします。1人で開発をするのであれば、Windows Server は不要です。

注6）URL Rewrite URL https://www.iis.net/downloads/microsoft/url-rewrite

注7）IIS Administration Cmdlets：Windows Server 2008 R2 / Windows Server 2012 R2
URL https://technet.microsoft.com/en-us/library/ee790599.aspx

注8）Windows Server 2016 / Windows 10 URL https://technet.microsoft.com/ja-jp/library/mt270166.aspx

▼図4　PowerShellによるIIS機能の追加

```
管理者: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:¥Windows¥system32> Enable-WindowsOptionalFeature -Online -FeatureName IIS-WEbServerRole


Path          :
Online        : True
RestartNeeded : False
```

Windows Server＋IISの場合

Windows Server上のIISでは、Windowsクライアント OSのような同時アクセス数の制限はなく、最もリッチな環境が手に入ります。最初に紹介した最大16年間のサポートも、Windows Serverでのみ提供されます。よって、実運用環境のIISではWindows Serverを利用することになります。

Windows Serverのインストール方法には、GUIを含んだフルインストールと、GUIのないServer Coreインストールの2つがあります。フルインストールでは、多機能であるために、OSそのもののサイズが大きいうえ、脆弱性の可能性も高くなります。アドホックな操作が多い開発環境ではGUIは便利ですが、運用環境のIISは、サイズが小さく、脆弱性の可能性が低いServer Coreインストールを利用することが推奨されます。

Container上のIIS環境の場合

Linux上で広く普及しているDocker Containerの Windows 版“Windows Container”が、Windows Server 2016より提供されるようになりました。これにより、Docker HubからContainerイメージをダウンロードするだけで、すぐにIIS環境を利用できます。また、自分で構築したIIS上のWebアプリケーションを、簡単にほかの環境に移すことができます。すでにDocker Hubには、Windows Server 2016 ベースのIIS のContainerイメージが提供されています（**図5**）。

▼図5　Docker Hubに提供されているWindows Serverイメージ

流行りのDevOpsをするうえでContainer化は必須ですので、IISをContainer環境で利用できることは知っておいたほうがよいでしょう。

なお、Container環境に向けて、Server Coreよりも小さなNano ServerというOSがWindows Server 2016より提供されています。IISだけを利用したい場合、小さなNano Serverは重宝します。すでにDocker HubにNano Serverのイメージも提供されています。ただ、Nano Serverを利用する場合、フルセットのASP.NETが利用できず、オープンソース版のASP.NET Coreしか利用できない点には注意が必要です。ASP.NET Coreとフルセット版ASP.NETの大きな違いは、ASP.NET Webフォームが使えるかどうかという点です。もし、ASP.NET Webフォームを使いたい場合は、Server Core版のContainerイメージを利用してください。

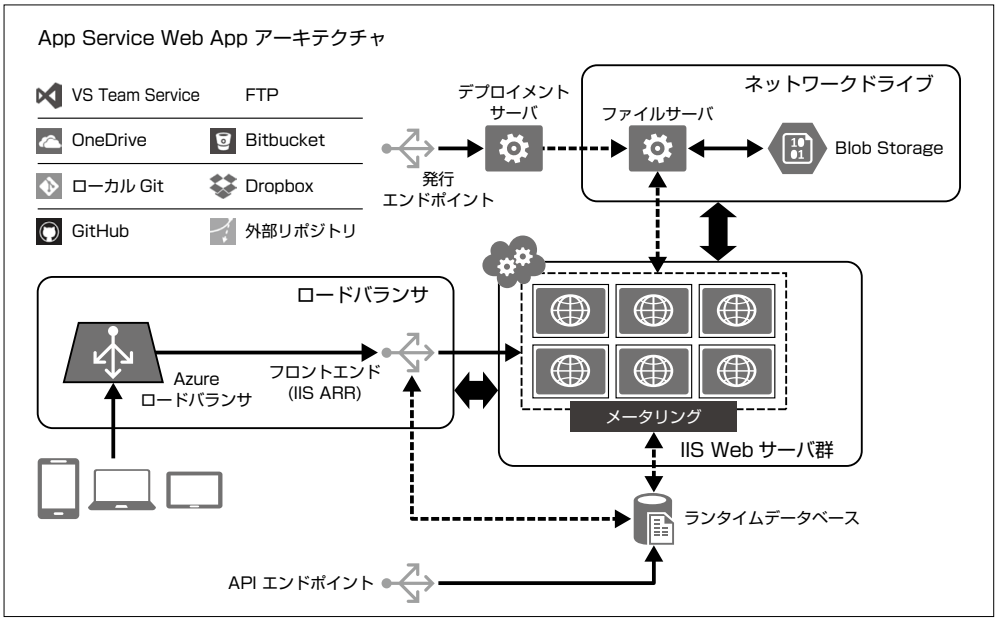
余談ですが、Docker Hubから、Linux + ASP.NET Core環境が提供されています。ASP.NET Coreはオープンソースのため、実行環境を選ばず、Linuxでも利用ができます。ASP.NET Coreは、Javaと同様、WindowsとLinuxで、共通のアプリケーションフレームワークとして利用できるようになっています。

Microsoft Azure App Serviceの場合

Azure App Serviceは、IIS（+ ARR）が稼働しているServerファームを予算に合わせて切り出して利用できるサービスです（**図6**）。すでに稼働しているIISを利用するため、わずか数分で自分専用のWebサーバが手に入ります。

App Serviceは、PaaS（Platform as a Service）として提供されるため、セキュリティ更新プログラムの適用や、ウイルス対策などの運用を行う必要もありません。アプリケーションフレームワークも、.NETだけでなく、PHP、Java、Pythonを複数バージョンサポートしています。これらのインストール作業やバージョンアップ作業も不要です（**図7**）。

▼図6　Azure App Serviceのアーキテクチャ



ほかにApp Serviceには、“デプロイメントスロット”という特徴的な機能があります。これは、本番用サイトのほかに、開発用、ステージング用、以前正常動作していた環境など、複数のサイトを持ち、必要に応じて、本番用サイトとテストサイトを瞬時にスワップする機能です（**図8**）。もしスワップ後に不具合が発生した場合には、もう一度スワップするだけで復旧できます。この機能により、Webアプリケーションのバージョンアップを安全に行うことが可能になります。

スケーラブルで、好きなアプリケーションフレームワークが利用でき、デプロイメントスロットのような便利な機能のあるIIS環境を、独自で構成、運用するのはたいへんな作業ですが、Azure App Serviceであれば簡単です。

WindowsクライアントOSとIIS Expressで開発したWebアプリケーションを、App Serviceへ展開するだけで、安全で、スケーラブルなWebアプリケーションを提供できます。

▼図7　Azure App Serviceのアプリケーション設定

IISを使ってみよう――簡単なWebサイトの公開まで

IISの特徴を最大限に発揮するには、ASP.NETとVSの組み合わせが最適です。ただし、単なる静的なHTMLの公開だけのために、VS（Visual Studio）をインストールするのは費用対効果がよくありません。VSは多機能なゆえに、アプリケーションのサイズが大きく、かつ、初心者には取り組みにくい印象があります。そこ

▼図8　Azure App Serviceのデプロイメントスロット

でここでは、Windows 10上のVS CodeとIIS Expressを使って、簡単なWebサイト公開までを紹介します。VS Codeは、Microsoftが無償で提供するエディタで、必要に応じてさまざまな拡張機能を追加できます。また、オープンソースとして開発が行われており、Windows以外のOS（mac OS / Linux）でも利用ができます。

Windows 10上のIIS ExpressとVS Codeで、静的HTML公開

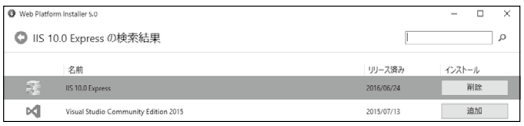
ここでは、Windows 10へIIS 10.0 Expressをインストールし、VS Codeで静的なHTMLを作成、ブラウザへ表示させる手順を確認します。まずは、IIS ExpressとVS Codeのインストール手順です。

①https://www.microsoft.com/web/downloads/platform.aspx にアクセスし、“Microsoft Web Platform Installer（Web PI）5.0”をインストール

②Web PIの検索から、“IIS 10.0 Express”を検索し、“IIS 10.0 Express”の［追加］をクリック（**図9**）そのあと、［インストール］ボタンでインストールを完了する

③https://code.visualstudio.com/ にアクセスし、VS Codeをダウンロードしてインストール（インストール手順は省略）

▼図9　Web Platform Installer 5.0



▼図10　IIS Express拡張機能を追加

④VS Codeを起動し、Ctrl＋Pで“コマンドパレット”を開き、“ext install iis”と入力（**図10**）

⑤“拡張機能”に表示された、“IIS Express <バージョン番号>”をインストール

以上で事前準備は完了です。次に、HTMLページをIIS Expressで表示してみましょう。

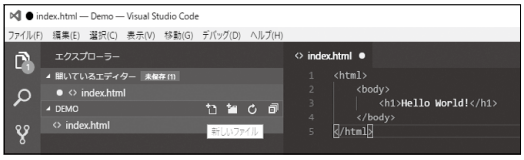
⑥VS Codeを起動し、［ファイル］→［フォルダーを開く］メニュー（もしくは、Ctrl＋K、次にCtrl＋O）で、表示したい静的HTMLファイルを保存するフォルダ（ここでは、“C:¥users¥<ユーザ名>¥Documents¥Demo”フォルダ）を選択

⑦⑥の“Demo”フォルダにマウスカーソルを合わせ、［新しいファイル（ドキュメントに＋のアイコン）］で、**図11**のように“index.html”ファイルを作成後、適当なHTMLを作成して保存（Ctrl＋S）

⑧Ctrl＋Pで“コマンドパレット”を開き、“>IIS Express: Start Website”と入力すると、IIS ExpressとWebブラウザが起動し、作成したHTMLが正しく表示されるかを確認できる。IIS Express初回起動時、IISの起動が失敗する場合があるので、その場合は“コマンドパレット”から、“>IIS Express: Restart Website”と入力する（**図12**）

⑨“タスクトレイ”をクリックすると、“IIS Express”のアイコンを確認でき、IIS Expressのポート番号の確認や、［サイトの停止］を行うこ

▼図11　index.htmlの追加



とができる（図13）

IIS ExpressとVS Codeで、PHPを使ってみる

IISは、アプリケーションフレームワークを選びません。初期のIISから提供されているIDC（Internet Data Connector）や、ASP（Active Server Page）、ASP.NET、ASP.NET Coreといったマイクロソフト純正フレームワークのほか、CGI、PHP、Perl、Pythonなどの利用ができます。ここでは、IIS ExpressへPHPを追加する次の手順を通じて、IISでのアプリケーションフレームワークの追加手順をイメージしてください。

①Web PIを起動し、“iis php”で検索。検索結果で、“PHP 7.1.1（x86）For IIS Express（英語）”の［追加］をクリック。同時に、“Windows Cache Extension 2.0（x86）for PHP 7.1 in IIS Express（英語）”も［追加］されます（図14）

②Web PIで、［インストール］をクリックし、インストールを完了

以上で、PHPのインストールは完了です。以降の手順で、IIS Expressから、PHPを利用できるように設定します。

▼図12　IIS Expressの再起動

▼図13　IIS Expressの稼働状況確認

③まず、“php-cgi.exe”のインストール先を確認します。デフォルトのインストールでは、“C:¥Program Files¥IIS Express¥PHP¥v7.1¥php-cgi.exe”となります。64bit版のWindowsの場合は、“C:¥Program Files(x86)¥...”となります（図15）

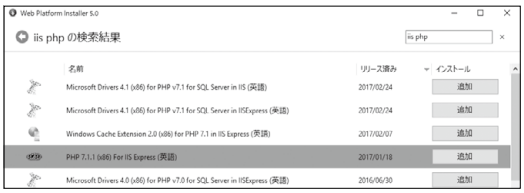
④VS Codeで、“C:¥Users¥<ユーザ名>¥Documents¥IISExpress¥config¥applicationhost.config”ファイルを開く

⑤“applicationhost.config”ファイルの、“<fastCgi />”を検索し（Ctrl＋F）、次の文字列に置換（図16）

```
<application fullPath="%ProgramFiles%\IIS Express\php\v7.1\php-cgi.exe" />
```

⑥同様に、図17のように“applicationhost.config”ファイルの、“<handlers accessPolicy”を検索し、次の文字列を追加したあと、上書き保存（Ctrl＋S）

▼図14　Web PIからPHPのインストール



▼図15　php-cgi.exeのパスを確認

▼図16　php-cgi.exeのパスを設定

▼図17　拡張子*.phpファイルの処理方法を定義

```
applicationhost.config
934   <handlers accessPolicy="Read, Script">
935         <add name="WebDAV" path="*" verb="PROPFIND,PROPPATCH,MKCOL,PUT,COPY,DELETE,MOVE
936     <add name="FastCGI-php_32bit" path="*.php" verb="*" modules="FastCgiModule" scriptProcessor="%ProgramFiles%\IIS Express\PHP\v7.1\php-cgi.
937     <add name="AXD-ISAPI-4.0_64bit" path="*.axd" verb="GET,HEAD,POST,DEBUG" modules="IsapiModule" scriptProcessor="%windir%\Microsoft.NET\Fra
```

```
<add name="FastCGI-php_32bit" path=
"*.php" verb="*" modules="FastCgi
Module" scriptProcessor="%ProgramFiles%\
IIS Express\PHP\v7.1\php-cgi.exe"
resourceType="Unspecified" />
```

⑦Visual Studio Codeを起動し、Ctrl＋Pで“コマンドパレット”を表示し、“ext install php”で、PHPに関する拡張機能を検索し、“PHP IntelliSense”をインストール。このステップはオプションです。ほかに使いたい拡張機能があれば、追加してください（図18）

⑧先に作成した“C:¥Demo”フォルダに、“test.php”ファイルを新規に作成し、次のコードを入力し、上書き保存（Ctrl＋S）。

```
<?php
    phpinfo();
?>
```

⑨“コマンドパレット”を開き（Ctrl＋P）、“>IIS Express: Restart Website”と入力してIIS Expressを再起動（図19）

⑩IIS Expressの再起動後にWebブラウザが開き、“http://localhost:<ポート番号>/index.html”が表示されるのを確認。続いて同じWebブラウザから、“http://localhost:<ポート番号>/test.php”を開いて、PHPの情報が表示されることを確認（図20）

以上、Windows 10での手順を記述しましたが、Windows 7でも、一部フォルダ名称などを読み替えれば、こちらの手順は適用できます。Windows 10以外のみなさんも、ぜひ試してください。

最後に

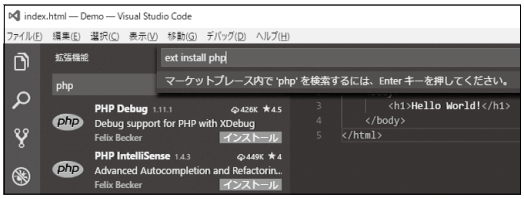
残念ながら誌面の都合で、IISの最大の特徴であるVSを使ったASP.NETによる開発を紹介できませんでしたが、ASP.NET Coreの初めの第一歩については、わかりやすい解説[注9]と動画[注10]があるので、紹介しておきます。

本稿で食わず嫌いにならず、IISに興味を持っていただき、簡単なところから試していただければ幸いです。SD

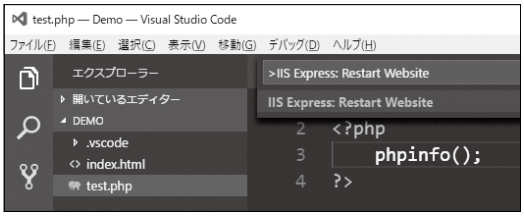
注9）Visual Studio Code を使用した ASP.NET Core Web アプリの作成　URL https://docs.microsoft.com/ja-jp/azure/app-service-web/web-sites-create-web-app-using-vscode

注10）Get started with VS Code using C# and .NET Core on Windows　URL https://channel9.msdn.com/Blogs/dotnet/Get-started-VSCode-Csharp-NET-Core-Windows

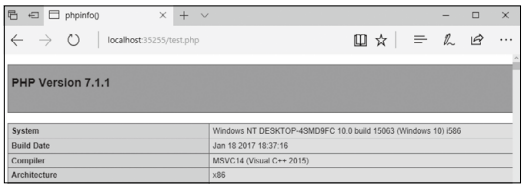
▼図18　PHP関連の拡張機能を追加



▼図19　test.phpファイルの追加と、IIS再起動



▼図20　phpinfo()の結果確認



第4章

HTTPの次はWebアプリを理解しよう

定番のJavaサーブレットで開発の基礎を学ぶ

第1～3章でHTTPやWebサーバについて詳細に見てきましたが、本章ではそれらの知識をふまえて、業務システムでどのようにHTTPが利用されているのかを見ていきます。Java Servlet（サーブレット）を中心に、最新技術まで解説します。

Author 長谷川 裕一（はせがわ ゆういち）　合同会社Starlight&Storm　日本Springユーザ会会長

業務システムの構成要素

Webアプリケーションとくに業務システムの開発を学ぶにあたって、まずHTTPをベースとした業務システムの基本的な構成要素（**図1**）を理解しましょう。

まず、Webクライアント上のWebブラウザから「何かの検索条件」や「商品の注文」などのリクエストが送信されます。Webサーバがそのリクエストを受信し、Webアプリケーションが DBと連携して処理を行って、処理結果を表示するためのWebドキュメント（Webブラウザ上でユーザに参照されるHTMLや画像など）を作成します。Webサーバはレスポンスとしてそ のWebドキュメントを送信して、Webブラウザにて表示されます。最新の技術を用いることで、異なる構成になることもありますが、まずはこの基本構成をベースに解説を進めていくことにします。

Java Servletの登場

初期のインターネット（HTTPなどによって相互接続されたコンピュータのネットワーク）は、もともと科学者間の情報交換を目的として、表示される内容が変わらない、いわゆる静的なWebドキュメントを表示していました。1993年に登場したCGIで動的なWebコンテンツを作り出せるようになったのですが、当時のインターネットは現在のように普及していなかったなどさまざまな問題で、一般的な業務システムとして広く利用されるには至りませんでした。

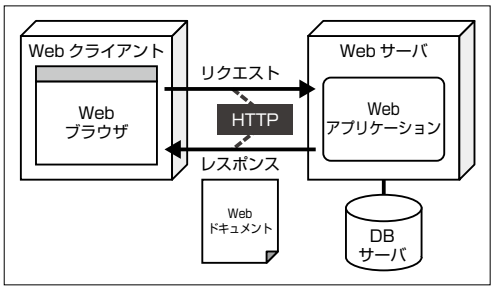
Java Servletは、CGI登場から約4年後の1997年に登場し、その後の業務システムの発展に大きく貢献します。

Servlet

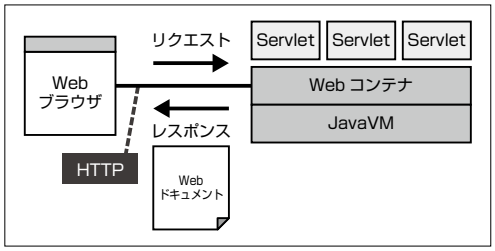
Servletは、動的なWebドキュメントを作り出せるプログラム（もしくは、その仕様）です。

ServletはWebコンテナ上で動作します。Webコンテナは複雑なHTTP通信を隠蔽（いんぺい）してくれます。そのため、Servletをプログラミングするときにはは HTTP通信の複雑な仕様や動

▼図1　業務システムの基本的な構成要素



▼図2　ServletとWebコンテナ



▼図3　URLの構成

きを気にすることなく、HTTPのリクエストを受け取り、レスポンスとしてWebドキュメントを返すWebアプリケーションを開発できるようになっています（**図2**）。

まずは、簡単なServlet（**リスト1**）を見て、その後、その動作を確認してみましょう。

ServletはHttpServletを継承して作ります（**リスト1-①**）。

先頭の@WebServlet（**リスト1-②**）はServlet 3.0から利用できるようになった機能で、Webブラウザから Servletを呼び出すときに使うURLの一部となります（**図3**）。@WebServletがなかったServlet 2.5までは、web.xmlでサーブレットマッピング（**リスト2**）を書かなければならなかったため、プログラマはServletのプログラムと、web.xmlを行ったり来たりしながら開発を行っていましたが、@WebServletのおかげでずいぶんと楽になりました。

doGetメソッド（**リスト1-③**）は、Webクライアントから HTTP通信でGETメソッドが送られてきたときに動作するメソッドです。

表1はHttpServletで定義されているHTTP

▼表1　HTTPとHttpServletのメソッド対応

HTTPのメソッド	HttpServletのメソッド
GET	doGet(HttpServletRequest req, HttpServletResponse resp)
POST	doPost(HttpServletRequest req, HttpServletResponse resp)
PUT	doPut(HttpServletRequest req, HttpServletResponse resp)
DELETE	doDelete(HttpServletRequest req, HttpServletResponse resp)

▼リスト1　最初のServlet「HelloWorldServlet」

```
@WebServlet("/hello")  ←②
public class HelloWorldServlet extends HttpServlet {  ←①
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)  ←③
        throws ServletException, IOException {
    response.setContentType("text/html; charset=UTF-8");

    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<meta charset=\"UTF-8\"/>");
    out.println("<title>最初のServlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Hello World!</h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

▼リスト2　web.xmlのサーブレットマッピング

```
<?xml version="1.0"?>
<web-app>
  <servlet>
    <servlet-name>helloworldServlet</servlet-name>
    <servlet-class>HelloWorldServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>helloworldServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

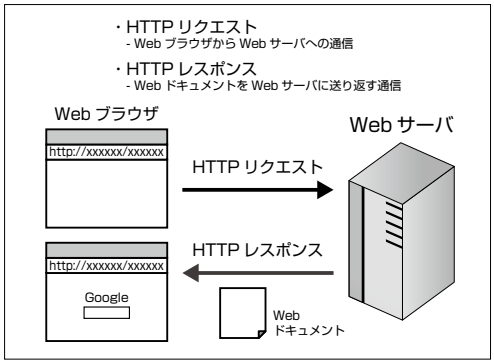
のメソッドに対応したおもなメソッドです。

doGetメソッドの引数にあるHttpServletRequestとHttpServletResoponseは、HTTP通信のリクエストとレスポンスを表現しています（**図4**）。

さて、**リスト1**のプログラムの半数を占める行に記述されているout.printlnメソッドを見ればわかるように、ServletはWebドキュメントであるHTMLを出力しています。

では、実際にWebサーバを起動してServletを動作させてみましょう。Webブラウザから URL に「http://localhost:8080/sdsample/hello」と入力することで、Webブラウザに「Hello World!」が表示されるのが確認できます（**図5**）。

▼図4　リクエストとレスポンス



ここで1つポイントがあります。先ほどの手順で動作確認をした場合、HTTPのGETメソッドがServletに送られてdoGetメソッドで処理されたということです。試しに、ServletのdoGetメソッドをdoPostメソッドに書き換えて動作確認すると、**図6**のようにHTTPステータス405が表示されることがわかります。

では次に、実際にGETとPOSTメソッドの使い分けができていることを、doGetとdoPostメソッドを持つGetPostServlet（**リスト3**）で

▼図5　HelloWorldServletの実行

▼図6　HelloWorldServletで405

Column　コンテナの話——EJB、Spring、ハリウッド法則

Servletを動かすWebコンテナだけではなく、EJBコンテナやSpring FrameworkのDIコンテナなど、いろいろなところでコンテナという言葉を聞く機会があると思います。

コンテナは基本的に、プログラマから複雑な技術を隠蔽して、プログラマに楽をさせてくれるモノと考えてください。今日、Javaが業務系システムのWebアプリケーションで多く利用されている理由の1つとして、こうしたさまざまなコンテナによって複雑な技術が隠蔽され、プログラマの負荷が低くなっていることが挙げられると思います。

コンテナの特徴を表現した言葉に、ハリウッド法則（Hollywood Principle）があります。それは「プログラムからコンテナを呼んではいけない。プログラムはコンテナから呼び出されるまで待て」ということです。たとえば、HelloWorldServletの動作例を見てもらったように、WebコンテナはWebクライアントからHTTP通信が来たときに適宜該当するServletのメソッドを呼び出しますが、その逆、ServletからWebコンテナを呼び出したりはしません。

こうした動きを、ハリウッドの女優（Servletに相当）とプロデューサ（Webコンテナに相当）の関係に例えて、「女優はプロデューサに自分から電話してはいけない。必要があればプロデューサから女優に電話する」というのがハリウッド法則です。

確認してみましょう。HTMLを出力するところは、両メソッドとも大きな違いはありません（**リスト3-①**）。

先ほどのHelloWorldServletでは、リクエストとして何の入力データも取得しなかったので、ここではWebブラウザから挨拶（たとえば、Helloとか）を入力データとして送り、GetPostServletでその入力データをリクエストから取得して（**リスト3-②**）、出力するようにしています（**リスト3-③**）。

GetPostServletを呼び出すためにget_post.html（**リスト4**）を用意しました。GET（**リスト4-①**）とPOST（**リスト4-②**）のどちらかのメソッドをServletにリクエストするようになっています。

▼リスト4　get_post.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>GetPost</title>
</head>
<body>
  <h3>挨拶を入力</h3>
  <form action="getpost" method="GET">
    <input type="text" name="greet" size=20>
    <input type="submit" value="GETメソッド">
  </form>                                          ❶
↑↓どちらかに入力してボタンを押下してください
  <form action="getpost" method="POST">
    <input type="text" name="greet" size=20>
    <input type="submit" value="POSTメソッド">
  </form>                                          ❷
</body>
</html>
```

▼リスト3　GetPostServlet

```
@WebServlet("/getpost")
public class GetPostServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        String greet = request.getParameter("greet");  ←❷
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<meta charset=\"UTF-8\"/>");
        out.println("<title>GETでRequestパラメータを受取る</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>GETで挨拶</h1>");
        out.println(greet);  ←❸
        out.println("</body>");
        out.println("</html>");
    }                                                                    ❶

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        String greet = request.getParameter("greet");  ←❷
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<meta charset=\"UTF-8\"/>");
        out.println("<title>POSTでRequestパラメータを受取る</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>POSTで挨拶</h1>");
        out.println(greet);  ←❸
        out.println("</body>");
        out.println("</html>");
    }                                                                    ❶
}
```

実際の動作結果は、図7のようになりました。

▼図7　GetPostServletの動作結果

JSPの登場

ここまでServletを解説してきましたが、業務システムでWebブラウザに表現するWebドキュメントはデザイン的にもっと複雑です。そのようなWebドキュメントをServletでデザインするには、あまりに多くのout.printlnメソッドが必要となり、デザインの修正などもたいへんになることは、ここまでのServletのプログラムを見れば容易に理解できます。

そうしたServletの欠点を補うため、1999年にHTMLライクなファイルの中にスクリプトを埋め込むページ・インライン・モデルとして、ServletをベースとしたJSP（JavaServer Pages）が登場しました。

まず、簡単なJSPのソースコード、hello.jsp（**リスト5**）を見てみましょう。HTMLライクで、WebドキュメントのデザインはServletよりJSPで行ったほうが良いことがわかります。

▼図8　JSPの実行結果

そして、hello.jspの実行結果は**図8**のようになりました。

しかし、JSPはHTMLではないことに注意してください。JSPは、Servletと同じようにout.printlnメソッドが大量に書かれたServletプログラムに変換されます（**リスト6**）。

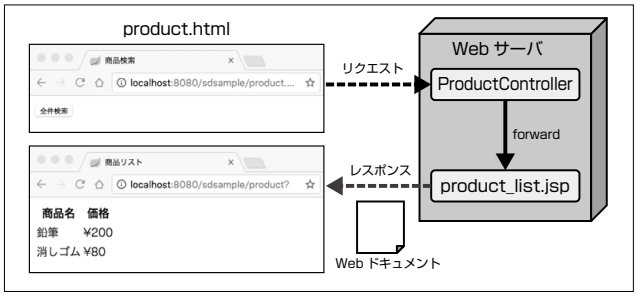
▼リスト6　hello.jspの変換後（一部）

```
public final class hello_jsp extends org.apache.jasper.runtime.HttpJspBase
  implements org.apache.jasper.runtime.JspSourceDependent,
             org.apache.jasper.runtime.JspSourceImports {
……途中省略……
    out.write("\n");
    out.write("<!DOCTYPE html>\n");
    out.write("<html>\n");
    out.write("<head>\n");
    out.write("<meta charset=\"UTF-8\">\n");
    out.write("<title>First JSP</title>\n");
    out.write("</head>\n");
    out.write("<body>\n");
    out.write("<h1>Hello World! jsp!</h1>\n");
    out.write("</body>\n");
    out.write("</html>");
  } catch (java.lang.Throwable t) {
……途中省略……
  }
}
```

▼リスト5　hello.jsp

```
<%@ page language="java" contentType="text/ﾌ
html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>First JSP</title>
</head>
<body>
  <h1>Hello World! jsp!</h1>
</body>
</html>
```

▼図9　JSP-Servletを組み合わせたWebアプリケーション



▼リスト7　product.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>商品検索</title>
</head>
<body>
  <form action="product" method="GET">
    <input type="submit" value="全件検索">
  </form>
</body>
</html>
```

▼リスト8　ProductController

```
@WebServlet("/product")
public class ProductController extends HttpServlet {
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
                    throws ServletException, IOException {
        List<Product> products = new ArrayList<Product>();
        products.add(new Product("鉛筆", "\200"));
        products.add(new Product("消しゴム", "\80"));
        request.setAttribute("products", products);  ←❷
        getServletContext()
          .getRequestDispatcher("/WEB-INF/jsp/product_list.jsp")
          .forward(request, response);
    }
}
```

❶ ❸

▼リスト9　Product

```
public class Product {
  private String name;
  private String price;
  public Product(String name,
                 String price) {
    this.name = name;
    this.price = price;
  }
  public String getName() {
    return name;
  }
  public String getPrice() {
    return price;
  }
}
```

JSP-Servletのプログラム

では、JSPとServletを組み合わせたWebアプリケーションの動作概要（**図9**）を見ながら、Webアプリケーションを構成するプログラム（**リスト7、8、9、10**）を解説します。

product.html（**リスト7**）は、Servletである ProductControllerのdoGetメソッドを呼び出すために作られたHTMLです。見てわかるように、何も特別なことはしていません。

リスト8はServletである ProductControllerです。画面に表示するデータとして、名前と価格を持つProduct（**リスト9**）をインスタンス化してListに加え（**リスト8-**①）、HttpServlet Requestに詰め込んでいます（**リスト8-**②）。

最後（**リスト8-**③）は、JSP（**リスト10**）へ、処理を移譲しているところです。forwardメソッドは“JSPに処理をお願いするメソッド呼び出し”だと考えてください。

▼リスト10　product_list.jsp

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>商品リスト</title>
</head>
<body>
  <table>
    <tr>
      <th>商品名</th>
      <th>価格</th>
    </tr>
    <c:forEach var="product" items="${products}">
      <tr>
        <td>${product.name}</td>
        <td>${product.price}</td>
      </tr>
    </c:forEach>
  </table>
</body>
</html>
```

HttpSession

HTTP通信は状態を持たない、いわゆるステートレスな通信です。たとえば入力画面、確認画面、完了画面のように複数画面で、利用者の固有な情報（利用者の名前などの情報）を保持する必要がある場合、利用者の固有な情報をHttpServletRequestに格納しても、クライアントにレスポンスを返してしまうと消えてしまいます。そこで、複数画面に渡って保持する必要がある、利用者の固有な情報はHttpSessionに格納します（**図10**）。

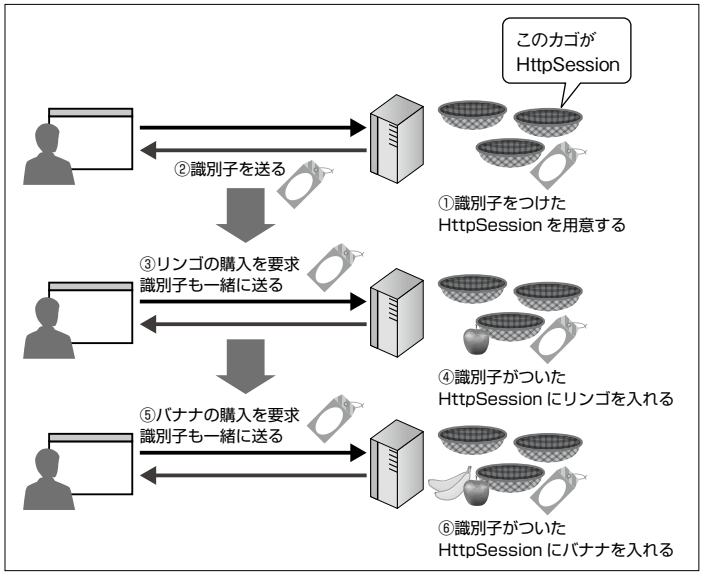
たとえば、Webアプリケーションのサンプルとしてよく登場するショッピングサイトのショッピングカートの場合、「バナナを3つ購入」のようなリクエストをしてきた利用者の固有、かつ、複数画面に渡って保持したい情報はHttpSessionに保存します。

HttpSessionがどうやって利用者に割り当てられるか、その動作を**図11**に示します。

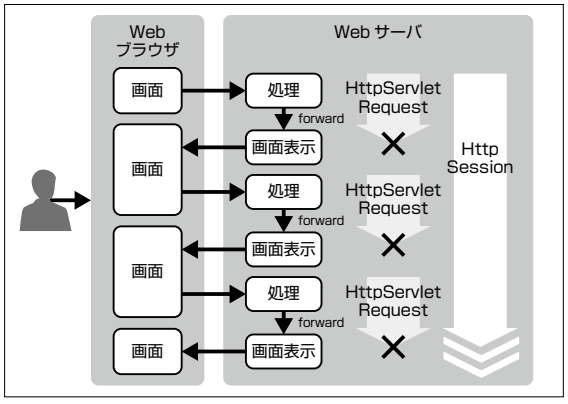
では、サンプルのWebアプリケーションの動作概要（**図12**）を見ながら、Webアプリケーションを構成するプログラム（**リスト11**、**12**、**13**）を解説します。なお、JSP-Servletのプログラミングで登場したProductクラス（**リスト9**）も利用しています。

product_session.html（**リスト11**）はServletであるProductControllerのdoPostメソッドを呼び出すために作られたHTMLです。Productのプロパティとなる名前と値段が入力されて、登録ボタンが押下されると、ProductControllerにHTTPのPOSTメソッドを投げるようになっ

▼図11　HttpSession



▼図10　HttpServletRequestとHttpSessionのスコープ



▼リスト11　product_session.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>商品登録</title>
</head>
<body>
  <form method="POST" action="productsession">
    <p>名前<input type="text" name="name"></p>
    <p>値段<input type="text" name="price"></p>
    <input type="submit" value="登録">
  </form>
</body>
</html>
```

ています。

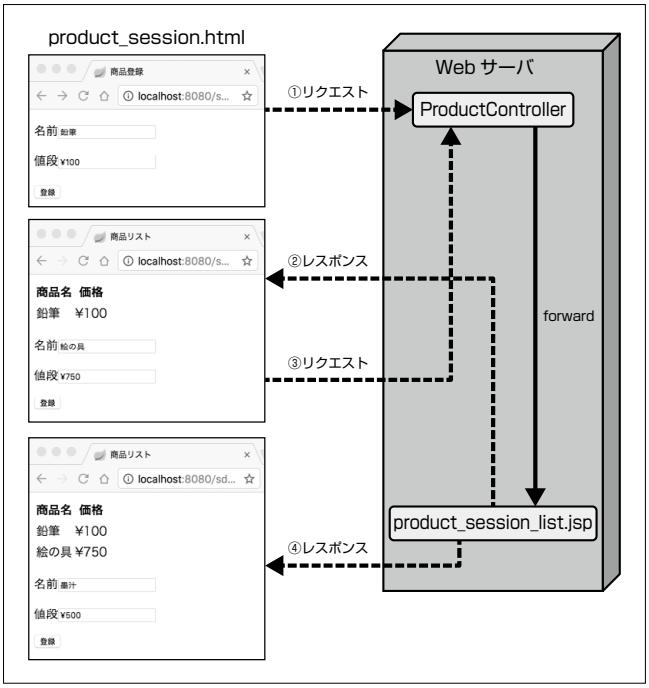
リスト12はServletであるProductControllerです。product_session.htmlで入力された名前と値段をHttpServletRequestから取得して、Product（**リスト9**）をインスタンス化しています（**リスト12-①**）。

続けて、HttpServletRequestからgetSessionメソッドでHttpSessionを取得しています（**リスト12-②**）。getSessionメソッドの引数にtrueが設定されている場合に、まだHttpSessionが作られていなければ新しく作り、falseが設定されている場合に、HttpSessionがまだ作られていなければnullが返ります。

HttpSessionの中に、Productインスタンスを保存するListができていなければ、Listを作成して、HttpSessionに加えます（**リスト12-③**）。

次に、画面に表示するデータとして、ProductインスタンスをListに加えます。本来はそのままJSPにforwardしても、JSPはHttpSessionにアクセスしてListからProductインスタンスを取得することも可能ですが、ここではあえてListをHttpServlet

▼図12　HttpSessionを使ったProduct管理



▼リスト12　ProductController

```
@WebServlet("/productsession")
public class ProductController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String name = request.getParameter("name");
        String price = request.getParameter("price");
        Product product = new Product(name, price);  ←❶

        HttpSession session = request.getSession(true);  ←❷
        List<Product> products = (List<Product>)session.getAttribute("products");
        if(products == null) {                                    ┐
            products = new ArrayList<Product>();                  ├ ❸
            session.setAttribute("products", products);           ┘
        }
        products.add(product);
        request.setAttribute("products", products);  ←❹
        getServletContext().getRequestDispatcher("/WEB-INF/jsp/product_session_list.jsp")  ┐
            .forward(request, response);                                                    ┘ ❺
    }
}
```

Requestに詰め直しています（**リスト12-④**）。

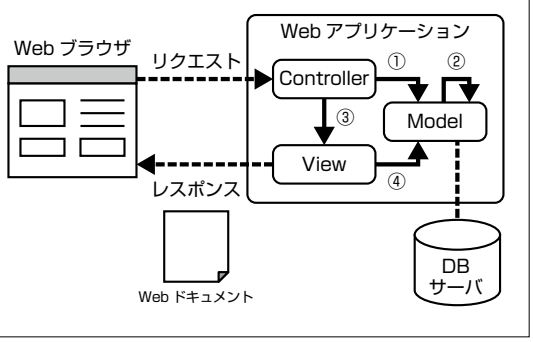
最後（**リスト12-⑤**）は、JSP（**リスト13**）への処理を移譲しているところです。JSPはProductのデータを表示して、再度入力を促すようになっています。

このように、Webブラウザから入力された情報をHttpSessionに保存することで、**図12**のように入力されたProductの情報が増えていくのがわかります。

▼リスト13　product_session_list.jsp

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>商品リスト</title>
</head>
<body>
  <table>
    <tr>
      <th>商品名</th>
      <th>価格</th>
    </tr>
    <c:forEach var="product" items="${products}">
      <tr>
        <td>${product.name}</td>
        <td>${product.price}</td>
      </tr>
    </c:forEach>
  </table>
  <form method="POST" action="productsession">
    <p>名前<input type="text" name="name"></p>
    <p>値段<input type="text" name="price"></p>
    <input type="submit" value="登録">
  </form>
</body>
</html>
```

▼図13　Webアプリケーションの MVC



Webアプリケーションの MVC

ここからは、Webアプリケーションの設計や、最新技術などを解説します。最初は、Webアプリケーションのアーキテクチャとして最も有名なMVC（Model-View-Controller）です。

業務システムのWebアプリケーションは、MVCの3つの役割に分けて設計、実装されることが一般的です（**図13**）。

MVCそれぞれの役割ですが、Controllerは、HTTPで送信されてきた情報を解釈し、適切なModelに情報を渡し処理を依頼します（**図13-①**）、Modelは、DBと連携して処理を行ってModelの状態を更新します（**図13-②**）。Modelの処理が終わると、ControllerはViewに対して処理を指示し（**図13-③**）、ViewはModelの状態を取得して（**図13-④**）Webドキュメントを作成し、WebブラウザにWebドキュメントを返却します。

MVCのそれぞれをJavaの技術に割り当てると、ViewはJSP、ControllerはServletです。ModelはRDBにアクセスしたり、業務ロジックをこなしたりするJavaのアプリケーションになります。

しかし、なぜMVCという役割分担した設計や実装が必要なのでしょうか。それは、電気製品のように役割ごとに何を行うかが明確なほうが、プログラムでも同様に何かと便利だからです。たとえば、デスクトップPCであれば、本体とディスプレイとスピーカーの役割は説明するまでもなく明確ですね。役割が明確になっていれば、コンポーネント化（部品化と読み替えてもかまいません）することも容易になります。そして一体になっているよりも、コンポーネント化されていたほうが、そのコンポーネントごとに開発を分けたり、分担して作業したりすることが可能ですし、変更に柔軟であったり、故障原因だって簡単に特定できたり

します。プログラムも電気製品のように、そこを目指したいと考えているのです。MVCはそうした役割を明確にするための1つのアーキテクチャであり、現在のWebアプリケーションの多くはMVCに則って開発されています。

フレームワークの登場

ServletやJSPを用いてMVCの解説を行いましたが、実は、ここまで解説してきたようなServletを直接プログラミングして、Controllerを開発するのはたいへんな労力と時間がかかります。

Servletは、Webアプリケーションの基本的な技術要素として新人研修で利用されることはあっても、実際の開発現場ではそのまま利用されることはほとんどないことは理解しておいてください。開発現場では、より簡単で効率的にWebアプリケーションを構築できる、Servletをベースに作られたControllerの役割を持ったフレームワークを利用します。

そのControllerの役割を持ったフレームワークのしくみは、WebアプリケーションとMVCの特徴に着目すると容易にわかります。

Webアプリケーションは、Webブラウザに表示するWebドキュメントの内容を動的に変えます。しかし、Webドキュメントの表示順番は紙芝居と同じように絶対に変わりません。つまり、A画面でOKボタンが押されたらA処理が動いて、それが正常終了ならB画面が表示される。こうした画面遷移と処理の関係は静的で絶対に変わらないということです（**図14**）。

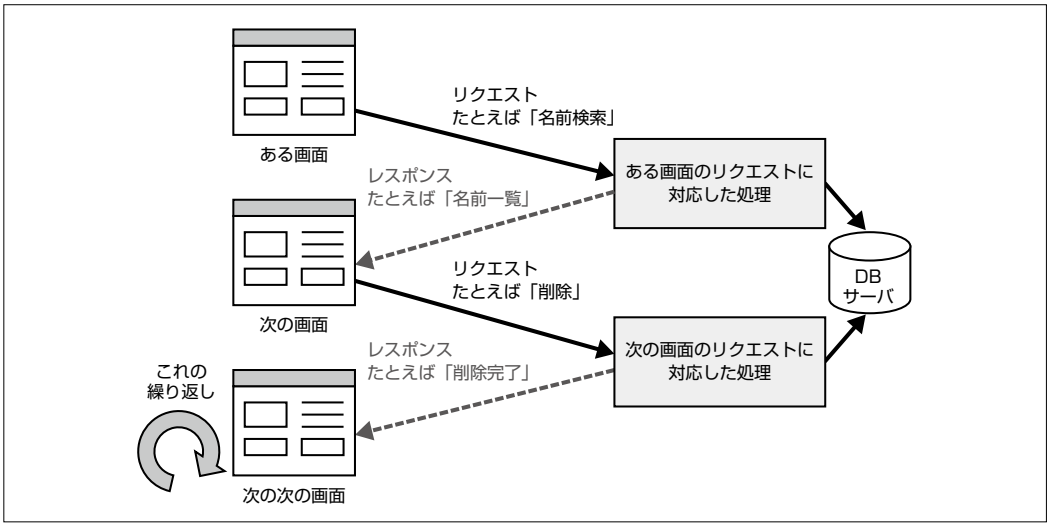
その部分に着目するとControllerは「ViewであるA画面でOKボタンが押されたら、ModelのA処理が動いて、それが正常終了ならViewであるB画面が表示される」ということが書かれたファイルなどを参照しながら動作すれば良いことになります。

こうしたMVCのController（と一部のView）部分に着目して実現されたオープンソースのフレームワークで成功を収めたのが、2001年に公開されたStrutsです（Strutsはすでに EoL[注1]を迎えているため、今後の利用は避けなければなりません）。

そしてその後、DBにアクセスするためのフレームワークとしてHibernateやMyBatis、それらすべてをカバーするSpringのようなフレームワークが登場して、現在にいたっています。

注1）End of Life：障害対応も含めた開発や、サポートの終了。

▼図14　Webアプリケーションの動き



また、JavaもStrutsに対抗するようにMVCのVCに着目した標準規格としてJavaServer Faces（JSF）を登場させています。

現在のJava Webアプリケーション

ここまで、Servletを中心に解説してきましたが、最後に現在のWebアプリケーションについて、さらに視野を広く解説します。

現在のJava Webアプリケーション開発では、Java EEかSpring Frameworkのどちらかを選択することになると思います（もしくは、それらをベースとした企業独自のフレームワークを使います）。そして、そのどちらを選択しても開発者が、ViewとしてJSPを利用することは少なくなっています。Java EEであればJSFのデフォルトView定義言語であるFacelets、Spring FrameworkであればHTMLのテンプレートエンジンであるThymeleafを使うことが一般的になりつつあるからです。先に解説したように、Servletを直に利用することはありません。フレームワークとして、Java EEであればJSF、Spring FrameworkであればSpring MVCを使うことになるでしょう。

また、Webアプリケーションが業務システムとして使われ始めた初期段階では、その利用を許されないことが多かったJavaScript（JS）も、現在ではユーザインターフェースを向上させるなどの目的のために普通に利用されており、JSフレームワークのBackbone.jsやAngularが注目されています。

FacebookやGoogle Mapsのように一般ユーザ向けに発展したSPA[注2]では、こうしたJSフレームワークが多用されています。このSPAを使った場合のアーキテクチャとして、WebクライアントからWebサーバのやりとりはRESTというHTTPのメソッドを正しく使った[注3]アーキテクチャが採用されることも多くなっています。

Servletは最先端のこうした流れを取り込むように、非同期など新機能が追加されており、開発者がServletを直接触ることはなくとも、Servletをベースにしたフレームワークなどから、その恩恵を受けています。

このように、現在でもHTTPは非常に重要な位置にあり、また、Webアプリケーションで動作するフレームワークなどはServletの技術をベースとしています。そのため、今後のスキルを上げていくためにもHTTPと、現在では直接プログラミングすることの少ないServletも、エンジニアの基礎知識として理解しておくことはたいへん重要なのです。SD

注2） Single Page Application：普通のWebアプリケーションのようにページ遷移によって画面を変更するのではなく、1つのページで画面遷移が行われる。

注3） 一般的なWebアプリケーションでは、GETやPOSTなどのHTTPのメソッドが、そのメソッドの意味のとおりに正しく利用されていなかった。

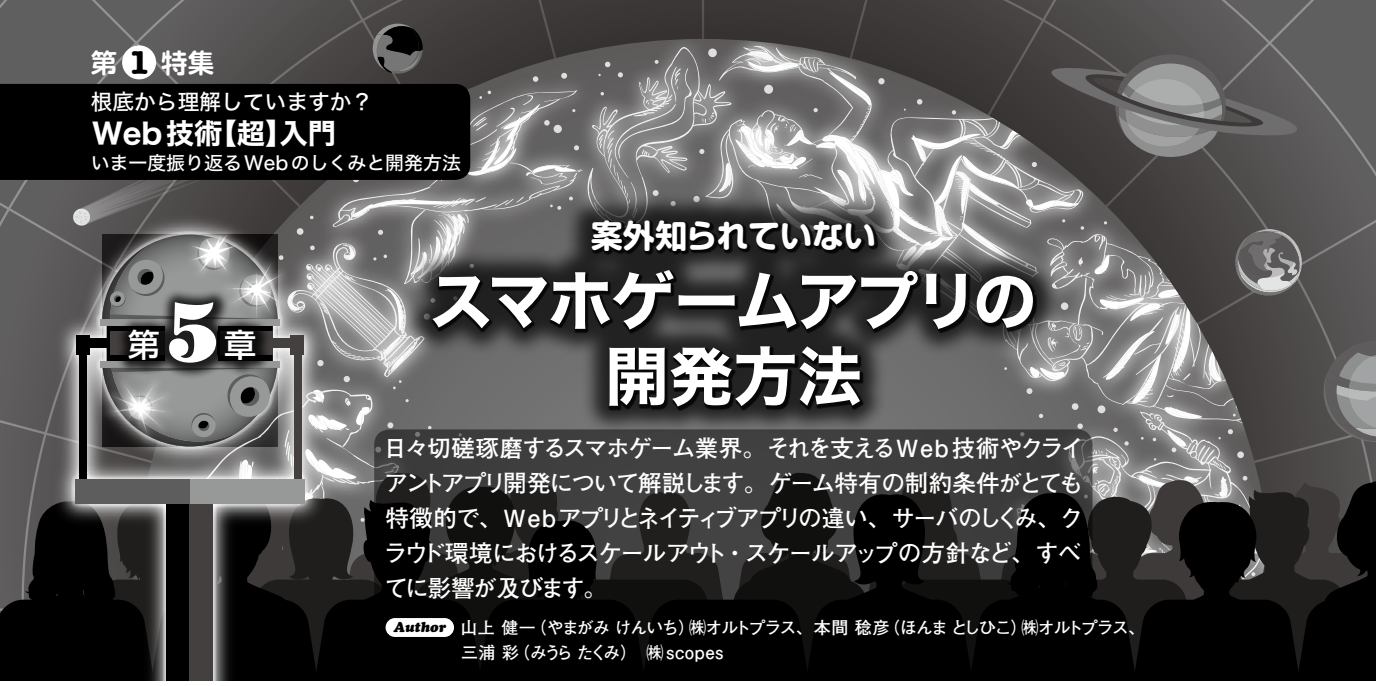
Column JSPはオワコン？

JSPの存在意義が危なくなっています。Java EEの標準MVCはJSFです。本来であれば2017年の夏頃に登場予定のJava EE 8に、Java EEの標準としてMVC（JSR 371）というフレームワークが登場し、JSPはそのViewとして存在意義が残るはずだったのですが、MVC（JSR 371）はJava EE 8にのらなくなってしまいました。

もともと、JSPはJSFが出てきた時点で、Java標準としての存在意義があいまいで、かろうじて、Strutsのようなアクション指向MVCフレームワークによって存在意義があったのですが、そのStrutsもEoLを迎えてしまいました。

そして、Struts同様のアクション指向MVCフレームワークであるSpring MVC（正確にはSpring Boot）でもJSPは推奨されなくなっています。

こうなるとJSPの存在意義は、新人研修以外にはなくなっているように感じるのは筆者だけでしょうか。



第5章

案外知られていない スマホゲームアプリの開発方法

日々切磋琢磨するスマホゲーム業界。それを支えるWeb技術やクライアントアプリ開発について解説します。ゲーム特有の制約条件がとても特徴的で、Webアプリとネイティブアプリの違い、サーバのしくみ、クラウド環境におけるスケールアウト・スケールアップの方針など、すべてに影響が及びます。

Author 山上 健一（やまがみ けんいち）㈱オルトプラス、本間 稔彦（ほんま としひこ）㈱オルトプラス、
三浦 彩（みうら たくみ） ㈱scopes

Webアプリケーションとクライアントアプリの違い

スマホアプリの2つの分類

みなさん通勤中の電車内やちょっとした空いた時間で、さまざまなスマホゲームにいそしんでいると思います。本稿はそんなスマホゲームの開発について、いわゆる既存のビジネスや通販サイトで使われているWebアプリとの違いや、どのようにゲームを作って運用しているのかを解説します。

みなさんご存じのようにスマホゲームの実装方法は、大きく2とおりに分類できます。1つはWebアプリ型、もう1つはいわゆるクライアントアプリ型です。それぞれの特徴は次のようになります。

Webアプリ型

Webブラウザ上で動作するアプリケーションです。MVCモデルでいうViewの実装にHTML、CSS、JavaScriptを使用します。ゲームロジックはWebフレームワークを利用して、プログラミング言語はRuby、PHP、Perlなどで実装する方法が一般的です。これは、すべての処理をサーバサイドで実装できるため比較的更新が容易ですが、Webブラウザ上で動作するという制約があります。そのため操作レスポンスが鈍くなりがちです。リッチなゲーム表現を実現することが難しい方式と言えます。また、GoogleやAppleの公式ストアから配信できないため、集客が難しいというデメリットがあります。しかしながら、最近ではWebテクノロジの進歩により操作性や表現力が向上しており、PC向けに移植する前提で、あえてWebアプリとして実装するケースも見受けられます。

クライアントアプリ型

スマホ端末上でWebブラウザを使わずに、独立して動作するアプリケーションです。スマホゲームの場合、ゲームデータの保存やユーザ間コミュニケーションを実現するために、サーバと協調して動作するクライアント／サーバモデルとして実装することが一般的です。

クライアントアプリの開発をUnityなどで行い、サーバ側のプログラムの開発はWebアプリケーションと同様にRuby、PHPなどのWebフレームワークを利用することが一般的です。クライアントはネイティブアプリケーションとして動作するので、リッチな表現を実現しやすく、操作レスポンスも良好な実装ができます。

公式ストアから配信することで、多くのユーザが使用し、遊んでもらえることが期待できます。そのためほとんどのスマホゲームがクライ

アントアプリケーションとして実装されています。その反面、クライアントに実装している処理を変更するためには、更新インストールが必要になります。それがユーザにとって不便さを感じる一因になっています。そんな問題を解決するために、クライアントアプリ内にWeb View（システムコンポーネント）を使用した簡易Webブラウザを実装し（**図1**）、更新頻度が高い個所はWebページとして実装する方式などの工夫をしている場合もあります。

スマホアプリがユーザに届くまで

Webアプリケーションと違い、クライアントアプリの多くは各プラットフォームのストアからインストールされます。App Store/Google Playストアに配置するためには、Apple、Googleへの申請が必要です（**図2**、**図3**）。

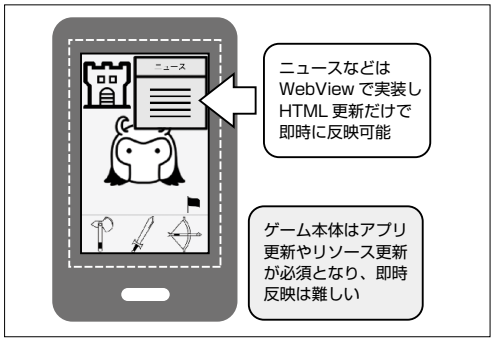
アプリの審査について

iOSとAndroidで大きく異なるのが審査に対する姿勢です。iOSは悪意のあるプログラムや公序良俗に反するコンテンツ、そのほかレギュレーションに抵触する機能が含まれていないか厳格なチェックが行われます。また、審査に要する日数は平均2日程度となっていますが、時期により前後する場合があり注意が必要です。基本的な審査の流れは次のようになります。

①アプリケーション開発
②ストア申請（Apple/Google）
③審査（iOSのみ）
④公開

一方、AndroidはiOSのような厳格な審査は設けられていませんが、こちらも同様に悪意のあるプログラムや公序良俗に反するコンテンツが含まれていた場合、削除されることがあります。なお、ストアのレギュレーション（規約）はApple、Googleによって規定されています。その更新が行われることもありますので、最新のレギュレーションを確認しましょう。

▼図1　クライアントアプリ内のWebView使用例



▼図2　Google Playへのアップロード画面

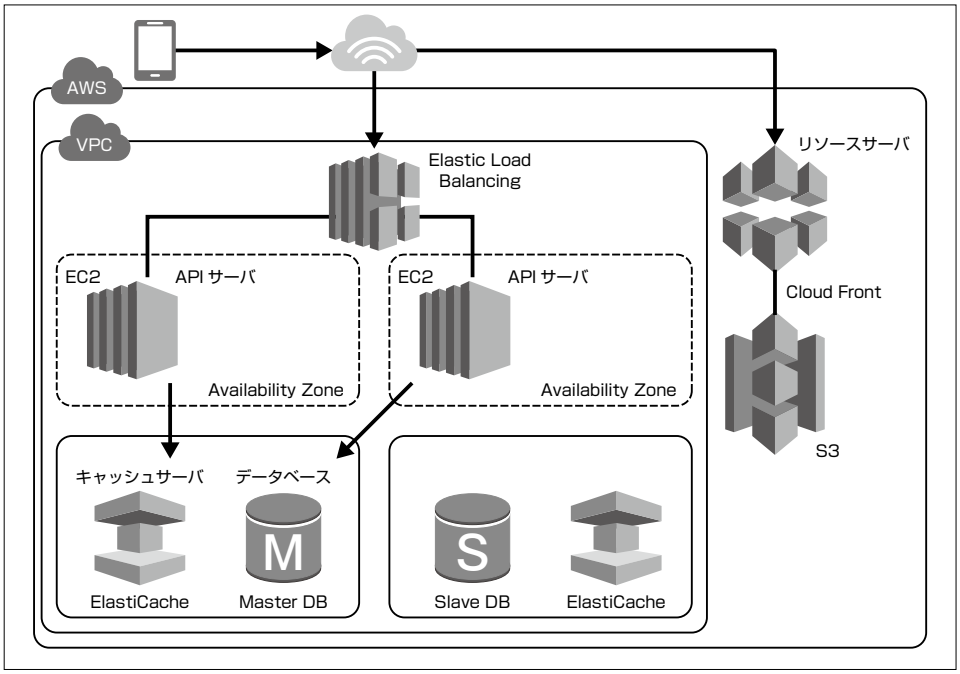
スマホゲームのサーバのしくみ

スマホゲームにおけるサーバの役割

スマホゲームにおいてサーバはクライアントから通信を受けて、処理を行い、クライアントにデータを返す役割を担っています。ユーザがストレスなくゲームをプレイできるようにパフォーマンスよくレスポンスを返すことがとても重要です。さらに、継続的にサービスを提供

▼図3　App Storeへのアップロード画面

▼図4　インフラ構成図（簡略版）



できるように「スケーラビリティ」や「冗長性」などを事前に考えておくことが重要です。ゲームの施策やユーザの急増加などに対応できるような設計にしておく必要があります。

サーバ側のインフラ構成

図4はAWSを利用した際のスマホゲームでよくあるインフラ構成を簡略化したもです。本例では、クラウド環境を前提にしています。サーバの用途は**表1**にまとめました。

APIサーバで用いられるプロトコルはHTTP/HTTPSが主流ですが、ほかのプロトコルを組み合わせて使用する場合があります。たとえば、チャット機能のように投稿内容をリアルタイムに表示更新する場合、HTTP/HTTPSではリクエストをポーリングする必要があり、通信量が増えてしまうデメリットがあります。リアルタイム更新が必要なAPIについては、WebSocketなどの双方向通信が可能なプロトコルを使用することで、サーバからクライアントに更新通知を行えるようになります。

こうしたゲームサーバ側に実装するプログラミング言語やフレームワークにとくに決まったものはありませんが、当社（オルトプラス）ではPHPやRubyなどを採用しています。直近のゲームタイトルではRuby on Railsを採用しました。Rubyはライブラリ群も多く、大きなスマホゲームのタイトルでも採用実績があります。

▼表1　サーバの種類

サーバ	用途	AWS（Amazon Web Servicse）
APIサーバ	ゲームに必要な処理を行うサーバ	EC2
データベース	ユーザのデータやゲームのデータを保持しているサーバで、運用上ボトルネックになりやすい	RDS、Aurora
リソースサーバ	画像やサウンドなどを配信するサーバ	CloudFront、S3
キャッシュサーバ	セッションや一時的なデータを保持するサーバ	ElastiCache

スケーラビリティの方針

スマホゲームはヒットすると、ユーザの数が急増加するためトラフィックや負荷が増大します。その変化に対応するためにスケーラビリティが非常に重要です。そのアプローチ方法として、サーバの性能を向上させて対応する「スケールアップ」、サーバの台数を増やして対応する「スケールアウト」があります。

たとえば、ユーザが急激に増えやすい傾向があるソーシャルゲームでは、スケールアップは一時しのぎにしかならないことがよくあります。ユーザ数が増えれば増えるほど負荷が集中してしまい、クラウドプロバイダが提供するサーバスペックの上限に簡単に達してしまいます。そうなるとそれ以上スケールアップできなくなってしまい、ユーザに快適にプレイをしてもらうことができなくなってしまいます。

ソーシャルゲームでは、スケールアウトをスケーラビリティの中心に考えます。スケールアップは奥の手として残しつつ、スケールアウトが可能な設計をします。負荷が高くなる例としては、メンテナンス終了直後が挙げられます。ゲームに熱中しているユーザ（熱量のあるユーザと言います）が一斉にアクセスするからです。

そこで、ユーザに快適にプレイしてもらえるように「APIサーバのCPU負荷が全体の45%を超えたらAPIサーバの台数を増やすスケールアウトをする」などの指標を立て、負荷を監視し、柔軟に対応できるようにしておくことが重要です。

とくにボトルネックになりやすいデータベースについては、スケールアウトをしやすくするため水平分散や垂直分散が可能なように作っておくことが重要です。また、データベースが分割されると、SQLでJOIN（表の結合）をすることが難しくなります。そのため実装時はJOINが不要なように設計しておく、JOINが必要なテーブルについては同じデータベース内にテーブルを持つなどの注意が必要になります（**表2**）。

ゲームアプリ間の通信の難読化

ユーザの情報を正しく保持することもサーバの重要な役割です。データ改ざんなどのセキュリティにも十分に注意する必要があります。その1つにAPI通信の難読化があります。

Web APIで多く使われているJSONの書式構造は人間に判読しやすいものになっています。このデータ構造の意味が判明してしまうと、悪意のあるユーザがデータの改ざんを簡単に行えてしまいます。API通信をHTTPSにすることで一定の効果が認められますが、ソーシャルゲームの場合は、それに加えて難読化の工夫を実施することが少なくありません。

たとえば、「クリアタイム」と「スコアのランキング」があるようなゲームがあるとします。そのゲーム内で**リスト1**のようなJSONデータをクライアントからサーバへ送信している場合、この通信を見ただけでscoreがスコアであると容易に想像がついてしまいます。scoreの値を有効な数値に変更して、サーバに送信をすれば

▼リスト1　リクエストのJSONサンプル

```
{
  "stage_id": 5,
  "clear_time": 82,
  "score": 31867
}
```

▼表2　データベースのスケールアウト手法

手法	用途
垂直分散	機能ごとにデータベースを分割し、読み書きを分散する。たとえばカードなどのデータと履歴などの異なる用途、機能のデータを分けるなど
水平分散	レコードごとにデータベースを分割し、読み書きを分散する。たとえばユーザIDが偶数と奇数でデータベースを分けるなど

改ざんができてしまうかもしれません。

この対策の1つの方法として、共通鍵暗号を利用してリクエストやレスポンスを暗号化・復号する方法があります。暗号化することで、その構造が簡単にわからないようにします。

リスト1のJSONデータを、サーバとクライアントで共有しているパスワード共通鍵とアルゴリズムを使って難読化します。これを使えば通信内容を見ただけで改ざんをすることが難しくなります。

たとえば、AES-256-CBCでエンコードし、さらにBase64エンコードしたテキストが次のようになります。

```
U2FsdGVkX19TPCmIrJ70eE1A3Oh0QEB2iwlgzki➥
R+Znk3gXW5vLGL8a4chDjwlvu 0BfA+NfY1SN2I➥
g6CJBq+Se0jrOEP0V+kuaypeNPiTCU=%
```

ところで、強い暗号を使用すればいいのかというと、そうではありません。暗号化・復号処理が大きくなってしまったり、通信サイズが大きくなったりすることでゲームの動作に影響が出ることがあり得ます。単に暗号強度が高ければよいというわけではなく、処理速度とのバランスを取る必要があります。また、暗号化をしたから安全というわけではありません。絶対に解けないものではないので、復号した入力値のチェックを行い、ユーザのデータを守る工夫も必要です。

スマホゲームのクライアント開発の実際

ゲームエンジンについて

昨今のスマホゲーム開発では、特別な事情がない限りiOS/Androidをはじめとするマルチプラットフォーム開発が前提となっています。単純に考えた場合、iOS向け開発ではSwift/Objective-Cを使用し、Android向け開発ではJavaを使用することになります。これはプラットフォームごとに開発環境を用意し、個別にプログラミングを行うことを意味します。つまり、単純計算でシングルプラットフォームに対して2倍の工数が必要となります。

ですが、実際にはプラットフォームごとに開発環境を複数用意することはあまりありません。多くの場合、「マルチプラットフォーム向けゲームエンジン」を導入します。これにより、いわゆる「ワンソースマルチユース」に近い体制で開発できるようになり、飛躍的に生産性が向上します。

もちろんデメリットもあります。本来ネイティブアプリが持っているパフォーマンスを完全に引き出すのは難しくなります。経験的な話になりますが、電力消費の増大や端末発熱に顕著な影響が出ることがあるように思います。また、後述しますがプラットフォーム間の差異を完全に吸収できるわけではありません。部分的には個別対応が必要となります（ちなみにコレが地味に辛かったりします）。ではゲームエンジンの例を見てみましょう。

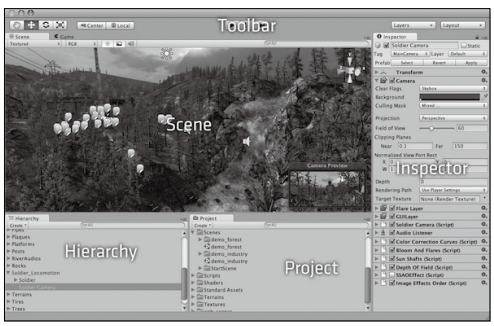
Unity

そのはじまりは個人開発者向けのツールとしてスタートしたものでした。現在では多くのスマホゲームで利用されています。ユーザコミュニティや技術ブログなど情報も豊富です（**図5**）。C#で開発します。

UnrealEngine

ハイエンドかつリッチな表現に強く、専用機

▼図5　Unity（http://tutorial.unity3d.jp/）



向けゲーム開発でも十分な性能を発揮します。C++およびビジュアルプログラミング環境「Blueprint」で開発します（**図6**）。

Cocos2d-x

オープンソースのゲームSDKです。Unity/UnrealEngineほどユーザ指向の開発環境ではありませんが、プログラマにとって自由度が高いのが特徴です。3D表現にも対応していますが2Dゲームに利用されることが多いです。C++で開発します。

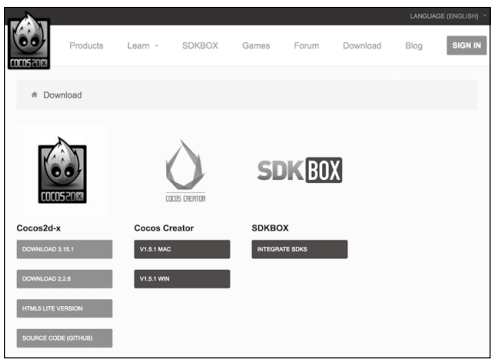
上記に挙げた以外にもさまざまなゲームエンジンがありますが、当社（オルトプラス）ではおもにUnityを利用しています。

チーム構成とワークフロー

次に実際の開発工程や手法を見ていきましょう。多くの場合、スマホゲームアプリ開発では、

▼図6　UnrealEngine
（https://www.unrealengine.com/ja/blog）

▼図7　Cocos2d-x
（http://www.cocos2d-x.org/download）



プログラマがコーディングするだけでは完結しません。UIデザイナやモーションデザイナなど多種多様なクリエイターと協力しながら開発を進めていきます。異なる職種間でのチームワークは高品質なゲームを作るうえで欠かせません。

よくあるパターンとして、ユニットの強化演出（アイテムなどを用いてユニットのパラメータ強化をすること）を例に挙げます。そのワークフローを次に示します。

①UIデザイナが強化用画像素材を作成
→強化結果が成功、大成功のようなランダム性を持つ場合、複数画像を用意する

②モーションデザイナが強化演出を作成、実装
→前述の成功、大成功に応じて画像を切り替えるが、スピードや緩急などのパラメータ調整、サウンド切り替えも行う

③プログラマが強化演出の表示制御を実装
→ロジックやユーザインタラクション（相互作用）に基づいて成功、大成功などを判定、表示や動作に結果を反映する

上記の①から③までの3フェーズが一方通行であることは少なく、フェーズ間で意見交換や企画のブラッシュアップなどを行い、フェーズを行きつ戻りつしながら完成へと近づいていきます。

なお、チーム規模やクリエイターのスキルセットによって、UIデザイナがインタラクション実装を兼任することや、プログラマがモーションデザインを兼任することもあり、上記のワークフローはあくまでも一例に過ぎません。

ツール構成の実際

当社では、Unityをプログラマだけでなく、モーションデザイナやUIデザイナも使用します。使用目的はそれぞれ異なりますが、最終的にはUnity上に制作物が統合されます。またそのほかにデザインツールや映像ツールなど多岐

▼表3　職種とツール

職種	ツール	使用目的
UIデザイナ	Photoshop/Illustratorなど	グラフィックデザイン／レイアウト作成
	Unity	素材インポート、レイアウト調整
モーションデザイナ	Live2D、AfterEffectsなど	モーション、演出作成
	Unity	モーション、演出作成および実装
プログラマ	Unity	コーディング、実装

にわたるツールを駆使し開発を進めています。主要なツールはおおむね**表3**のような構成になっています。

表3の構成からでは、プログラマはUnityだけ使用できればいいといった印象を受けますが、実際は少し異なります。プログラマが各種ツールに対して一定の知識を持っていると非常に頼りになります。そのほうが、デザインや演出のフェーズで実装に無理のない制作方法や設計を提案できるからです。これで手戻りや修正コストを下げられます。すべてのツールに精通する必要はありませんが、それぞれの特性や使い方を把握していればチーム全体の生産性向上に貢献できます。

クライアントアプリ実装の工夫

最後にゲーム開発の本筋からやや離れますが「対応しないとリリースできない」類の話をいくつか紹介したいと思います。

アプリサイズとの戦い

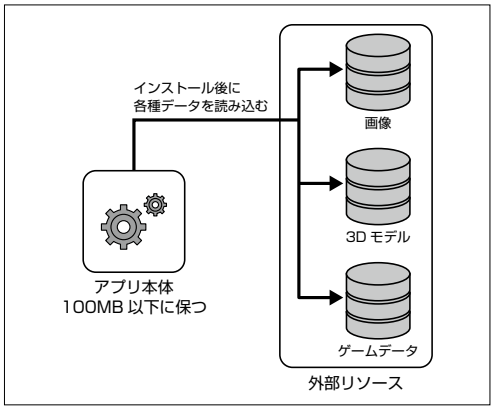
完成したアプリはバイナリファイルipaやapkなどとしてApp StoreやGoogle Playにアップロードし公開します。モバイル通信やプラットフォームの都合上、ファイルサイズは100MBに収めるのが一般的です。また、ダウンロード時間を短縮するためにも可能な限り小さく抑えたいところです。ですが昨今のアプリ事情では使用する画像や3Dモデル、各種データの合計が100MBに収まることはまずありません。そこでこれらを外部リソース化し、本体のファイルサイズを小さく保ちます。

UnityではAssetBundleという外部リソースのための機能が備わっており、外部リソースを作成すること自体は比較的簡単です（**図8**）。しかし、更新頻度や運用スケジュールなどを考慮したうえで適切に外部リソース構造を設計する必要があります。未対応の場合、公開後のアプリ更新時に思わぬ事態が発生したり、煩雑な手作業に工数を奪われたりすることになります。

各プラットフォームの避けられない差異

「プラットフォーム間の差異を完全に吸収できるわけではなく、部分的には個別対応が必要」と述べましたが、最たる例としてアプリ内課金を挙げます。基本的にはiOS/Android間で大きな違いはありませんが、微妙な処理順の違いや仕様の差異がかえってプログラマの頭を悩ませます。さらに動作確認では、実機検証が必須となるので検証時間も長くかかります。性質的にも最もセンシティブな部分なので万全を期して対応する必要があります。

▼図8　アプリと外部リソースの関係



Asset Storeは開発の味方

Unityには、サードパーティライブラリや素材などを集めたAsset Storeというエコシステムがあります。Asset Storeでは次のようなアセットがあります。

・3Dモデル
・エフェクト素材
・アニメーション補助ツール
・アプリ内課金補助ツール
・SNS連携
・動画キャプチャ

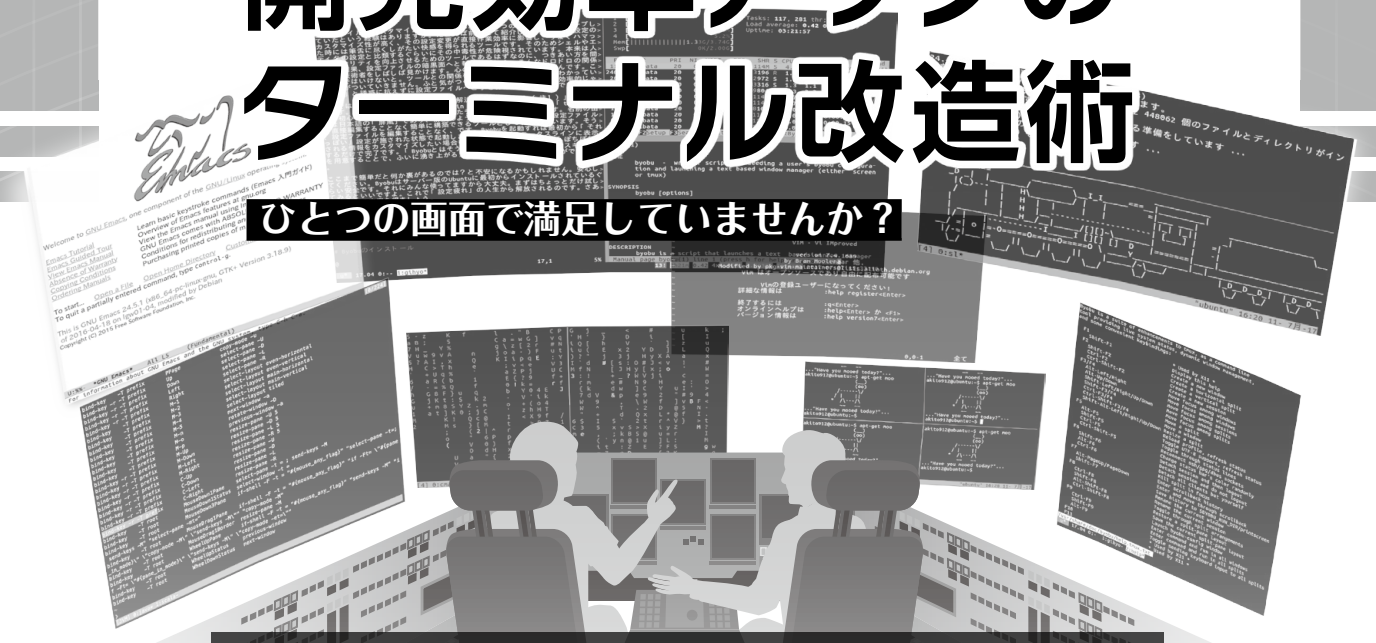
このように多岐にわたります。有料アセットの購入もAsset Store上で行うことができます。また、アプリ開発者自らがアセットを出品できます。ゲームアプリ開発では、ゲームの核となる部分以外に、アプリ内課金やSNS連携、プッシュ通知、実況動画投稿など周辺機能は無視できません。このような機能をAsset Storeで迅速に導入して、ゲームアプリの本質的な部分の開発に工数を注ぐことができます。SD

▼図9　Asset Store（検索条件などを詳細に設定できる。開発の強力な味方になる

tmux & Byobu

開発効率アップのターミナル改造術

ひとつの画面で満足していませんか？



「エディタを開きながらすぐ横でコマンドを動かしたい」「同じコマンドを4つ同時に実行したい」といった場面では、1つの端末画面だと力不足を感じるでしょう。そんな問題の最適解として、端末画面を分割・多重起動して開発・運用効率を上げる「ターミナルマルチプレクサ」があります。本特集ではその代表ソフトとも言えるtmuxを取り上げ、入門から使いこなしまで解説します。さらに、設定要らずでtmuxを“それっぽく”使えるラッパー「Byobu」も紹介。自分でとことんカスタマイズしたい方はtmuxを、カジュアルに使いたい方はByobuを使って、マルチターミナルな環境を実現してください。

第1章 tmuxをはじめよう

ウィンドウ・ペイン・セッションの概念と操作をマスター

Author 田中 智文（たなか ともふみ） 株式会社ビズリーチ
Twitter @tanacasino

本章では開発や運用でターミナルを使うエンジニアに人気のソフトウェアである「tmux」の導入手順と使い方を紹介します。ぜひ実際に手を動かして試してみてください。

tmuxの基本

どんなことができるのか?

tmux[注1]はターミナルマルチプレクサ（terminal multiplexer）と呼ばれるソフトウェアの1つで、1つのターミナルから複数のターミナルを簡単に操作できるようにしてくれます。同様のソフトウェアには古くから存在するGNU Screenがあります。

tmuxの本質は、「Webブラウザのタブ機能をターミナルの世界でも実現してくれるもの」ととらえるとイメージしやすいかと思います。tmuxを使えば、Webブラウザのタブを使うのと同じぐらい簡単にターミナルの作成、切り替えができるので、エディタでコードを編集しつつ、テストを実行するといった並行作業を効率的に行うことができます。

tmuxを使うメリット

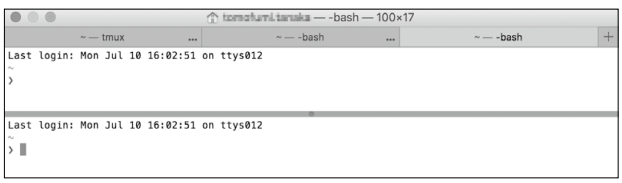
タブ機能であれば、そもそもターミナルを起動するアプリに標準で付いているのではないか？と思うかもしれません。実はそのとおりで、macOSに標準で入っている「ターミナル（Terminal.app）」（**図1**）や、Ubuntuのデスクトップ版に標準で入っている「端末（gnome-terminal）」といったソフトウェアには、タブやウィンドウの機能があります。

それでもなおtmuxを使うメリットとしては、次の点があります。

(1) セッション、ウィンドウ、ペインといった管理単位が存在しており、複数のターミナルを整理して管理できる
(2) リモートのサーバでtmuxを実行しておくことで、ネットワーク接続が切れてもすぐに作業に復帰できる（デタッチ・アタッチ）
(3) ショートカットキー（キーバインド）や表示といった、好みの分かれる部分をカスタマイズでき、ほかのコマンドやシェルスクリプトと連携させやすい

(2)はとくに大事なメリットです。運用や検証といった作業は、リモートのサーバにSSHでログインして行うことが多いと思います。サーバで重いコマンドやスクリプトを実行中に、ロー

▼図1 macOS標準のターミナル



注1） https://github.com/tmux/tmux

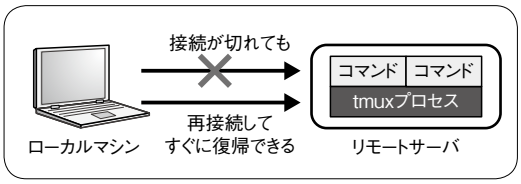
カルのラップトップがうっかりスリープしてしまったり、無線LANの接続が切れてしまったりするとどうなるでしょうか？　ほとんどのプログラムは途中で終了してしまうか、実行が終わっていたとしても、出力がないので終わったかどうかを外から判断できないといったことになります。もし大事なデータを扱う処理を実行していたら、不整合が発生してトラブルに発展するかもしれません。

このような事態も、リモートサーバでtmuxを使えば避けることができます。リモートサーバでの作業には、tmuxが必須といっても過言ではないでしょう(**図2**)。

また最近ではローカルでの開発でも、HTML/JavaScript/CSSといったフロントエンドのビルドとバックエンドサーバのビルドが分かれているソフトウェアや、マイクロサービスアーキテクチャのように複数のサービスが連携して動作するソフトウェアがあり、複数のプログラムをターミナルから同時に実行することが増えてきています。ローカルマシンでもtmuxを使うことで、それぞれの開発・運用に適したカスタマイズを行い、効率をアップすることができます。

このようにサーバでもローカルでもtmuxを使うメリットがあります。次節から、インストールして実際に触ってみましょう。

▼図2　サーバでtmuxを使う



インストールしよう

macOS

macOSでは、Homebrewを使用すると簡単にインストールできます。公式ページ[注2]にしたがってHomebrewをインストールし、次のコマンドでtmuxをインストールします。

```
$ brew install tmux
```

本特集第1、2章では、執筆時点(2017年7月)の最新版である「2.5」というバージョンを基準に解説します。バージョンが異なると、設定や振る舞いが違うことがあります。以降の説明ではなるべく補足するようにしますが、うまく動作しないときはバージョンを調べてみてください。次のコマンドでバージョンを調べられます。

```
$ tmux -V
tmux 2.5
```

Linux(Ubuntu 16.04 LTS)

各Linuxディストリビューションでは、バイナリが用意されています。それぞれのディストリビューションに入っているパッケージ管理ソフトを使用してインストールできます。ただし、tmuxのバージョンはディストリビューションごとに異なりますので注意してください。

ここでは、Ubuntu 16.04 LTSを対象にします。

```
$ sudo apt-get install tmux
```

Ubuntu 16.04では、tmux 2.1がインストールされます。基本的な動作は変わりませんが、設定などが異なる場合があります。本特集で紹介するオプションと同じものを使いたい場合は、**図3**の手順でソースコードからビルドしてインストールできます。この手順でインストールしたtmuxコマンドは/usr/local/bin/tmuxになります。次のコマンドで確認しておきましょう。

```
$ which tmux
/usr/local/bin/tmux
$ tmux -V
tmux 2.5
```

注2) https://brew.sh/index_ja.html

まずは使ってみよう

tmuxは、画像や説明文だけでは雰囲気をつかみづらいツールです。実際に手元でtmuxを実行してみてください。

```
$ tmux
```

起動すると、ターミナルの下部の表示が変わっています(**図4**)。これは「ステータスライン」と呼ばれる表示領域です。それ以外は普通のターミナルと同じように使用できます。

確認のため、なんでも良いのでコマンドを実行してみましょう。続いて、**Ctrl-b**(Ctrlキーを押しながらbキーを押す)としたあとに、cキーを押してみてください。先ほど実行したコマンドの出力がなくなり、きれいなコンソールが表示されたかと思います。実はこれで利用できるターミナルが増えており、ステータスラインの表示も変わっています。

```
[0] 0:bash*
          ↓
[0] 0:bash- 1:bash*
```

元のターミナルに戻るため、**Ctrl-b**としたあとにpキーを押してみてください。これで先ほどコマンドを実行したターミナルに戻ってくることができました。先ほど元のターミナルで実行したコマンドが表示されるはずです。

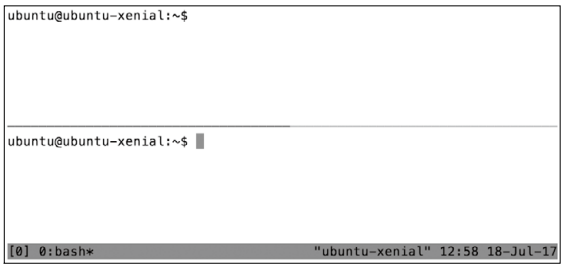
以降では、「**Ctrl-b**としたあとにpキーを押す」操作を**Ctrl-b p**と表記するとします。

この、新しくターミナルを増やしたり、移動したりできるターミナルを「ウィンドウ」と呼びます。ステータスバーに表示されている**0:bash**や**1:bash**はウィンドウを表しています。

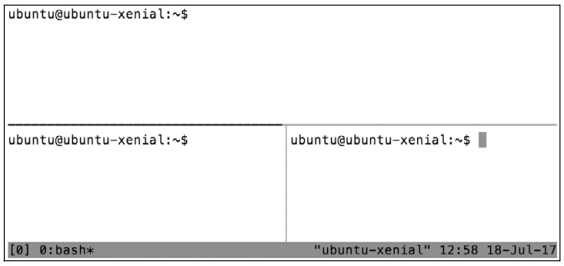
次は「ペイン」を使ってみましょう。**Ctrl-b "**(ダブルクォーテーション)を入力してみてください。なんと画面が上下半分に分割されました

▼図4　tmux起動直後

▼図5　画面分割(上下)



▼図6　画面分割(左右)



▼図3　tmux 2.5をインストール

```
$ sudo apt-get install build-essential libevent-dev libncurses5-dev
$ curl -LO https://github.com/tmux/tmux/releases/download/2.5/tmux-2.5.tar.gz
$ tar xzvf tmux-2.5.tar.gz
$ cd tmux-2.5
$ ./configure && make
$ sudo make install
```

(**図5**)。

この状態でさらに**Ctrl-b %**を入力してみてください。次は、下半分のターミナルが左右に分割されました(**図6**)。このようにウィンドウの中で画面を分割してターミナルを増やすことができます。これらをペインと呼びます。

次は接続の終了「デタッチ」と、再接続「アタッチ」をしてみましょう。あとでわかりやすいように、次のコマンドを実行しておきます。

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

このコマンドは**Ctrl-c**で止めるまで、5秒おきに日時を表示します。コマンド実行後に**Ctrl-b d**を押すと、次のような表示とともにステータスラインの表示が消えます。

```
$ tmux
[detached (from session 0)]
```

tmuxが終了してしまったと思われるかもしれませんが、tmuxはバックグラウンドで動作しているため、先ほど実行したコマンドは問題なく動作を継続しています。動作中のtmuxに再度接続するには、次のコマンドを実行します。

```
$ tmux attach-session
```

これで、先ほど実行した日時を出力するコマンドが動き続けていたことを確認できます。

このようにtmuxを使うことで、簡単に複数のターミナルを増やしたり、分割して並べて表示したり、接続を切って離れたあとに再度接続したりといったことができます。

用語を覚えておこう

一通りの雰囲気をつかんだところで、tmuxを使ううえで覚えておきたい用語を整理しておきましょう。tmuxにおける管理単位(**図7**)と「プレフィックスキー」について説明します。

ウィンドウ(window)

Webブラウザのタブのように、ターミナル(シェル)を作成、移動できます。ウィンドウの一覧は、デフォルトでステータスラインに表示されます。ウィンドウは1つ以上のペインを持つことができ、ペインの管理単位と考えることができます。

ペイン(pane)

ペインは、ウィンドウの中に複数持つことができる領域で、上下左右に分割して増やすことができます。ペインを使うと複数の出力を同時に表示できるので、複数台のサーバのログを同時に**tail**してみたり、サーバのリソース(CPU、メモリ、I/Oなど)の使用状況を比較するといったことができます。

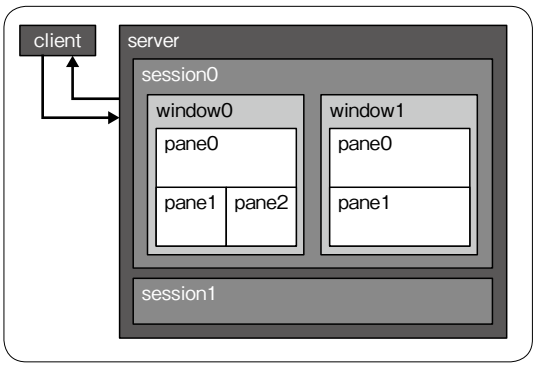
セッション(session)

セッションはウィンドウを管理する単位です。ウィンドウがペインを1つ以上持つように、セッションは1つ以上のウィンドウを持つことができます。ウィンドウの数が多くなってきたら、セッションを分けて管理すると良いでしょう。

サーバ(server)／クライアント(client)

サーバはすべてのセッションを管理するtmuxプロセスで、バックグラウンドで動作します。

▼図7　管理単位の整理



デタッチ／アタッチは、このtmuxプロセスと通信を行うtmuxクライアントのプロセスを開始／終了するということです。

tmuxを起動すると、自動的にサーバプロセスとセッション、ウィンドウが作成され、最後にtmuxクライアントのプロセスとしてそのサーバに接続します。

プレフィックスキー(prefix key)

ウィンドウの作成やペインの分割をした際にショートカットキーを使いました。このとき、毎回はじめに押していた**Ctrl-b**のことをプレフィックスキーと呼びます。tmuxではこのプレフィックスキーを押したあとに別のキーを押すことで、特定のアクションが実行できるショートカットをよく使います。

基本操作をマスターしよう

ウィンドウ

まずはウィンドウの操作をマスターしましょう(**表1**)。作成、削除、移動といった操作は想像がつくと思いますので、少しややこしい概念の「名前」について紹介します。

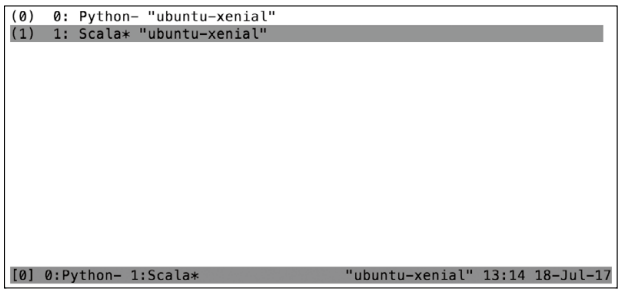
ウィンドウには自動的にセッション内で一意な番号が割り当てられますが、それとは別に名前を付けることができます。この番号と名前が、まさにステータスバーに表示されている内容です。たとえば**0:bash**は、番号「0」のウィンドウで、名前が「bash」のウィンドウを表しています。わかりやすい名前を付けることで、複数あるウィンドウの中から目当てのウィンドウをすぐに見分けることができます。

名前を変更するには、変更したいウィンドウ

▼表1 ウィンドウ操作のショートカットキー(プレフィックスキーのあとに押すキー)

キー	サブコマンド	意味
c	new-window	ウィンドウを作成
n	next-window	次のウィンドウに移動
p	previous-window	前のウィンドウに移動
,	rename-window	ウィンドウの名前を変更
0〜9	select-window	0から9の番号のウィンドウに移動
l	last-window	直前のウィンドウに移動
w	choose-window	ウィンドウの一覧を表示して、移動先のウィンドウを選択(図8)。一覧が表示されている状態では、カーソルキー←↑→↓で移動、エンターキーEnterで確定
&	kill-window	現在のウィンドウを削除

▼図8 ウィンドウの一覧を表示(choose-window)



COLUMN ショートカットキーの正体

tmuxのショートカットキーは、それぞれtmuxのサブコマンドのショートカットになっています。

たとえば、ウィンドウの作成はプレフィックスキーのあとにcキーでしたが、これは**tmux new-window**というコマンドが割り当てられているためです。試しに**tmux new-window**コマンドを実行してウィンドウが作成されることを確認しておきましょう。

ただしこの**tmux new-window**によるウィンドウ作成の操作は、tmuxのプロセスが起動しているマシンと同じマシンで実行した場合のみの動作となります。tmuxのターミナルからsshコマンドなどでリモートサーバにログインしている状態のシェルで実行した場合は、そのリモートサーバのtmuxに命令が飛ぶので、意図したとおりに動かないことに注意してください。その場合は手元のマシン上でショートカットを使いましょう。

に移動して、Ctrl-b ,(カンマ)を押します。すると、ステータスバーが次のように変化し、任意の文字を入力可能になります。

```
(rename-window) bash
```

たとえばテキストエディタを起動するウィンドウとして「editor」という名前を付けておけば、別の作業をしたあとにすぐにエディタに戻りやすくなります。

```
[0] 0:editor* 1:bash  2:bash-
```

またステータスバーでは、現在いるウィンドウの名前の後ろに「*」が、その直前に使っていたウィンドウの名前の後ろには「-」が付くようになっています。見分けるために覚えておくと良いでしょう。

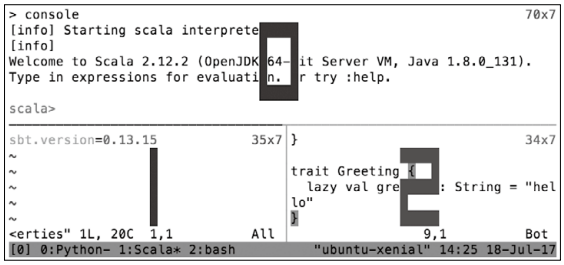
ペイン

続いてペインの使い方をマスターしましょう(**表2**)。ペインもウィンドウと同様に、作成、削除といった操作が可能ですが、名前を付けることはできません。代わりに、ウィンドウごとに自動的に一意な値が割り当てられるペインの番号を使うことができます。加えて、ペインでは入れ替えや、サイズの変更といった細かい操作を行うことができます。

▼表2　ペイン操作のショートカットキー(プレフィックスキーのあとに押すキー)

キー	サブコマンド	意味
"	split-window	画面を上下に分割
%	split-window -h	画面を左右に分割
x	kill-pane	現在のペインを削除
q	display-panes	ペインの番号を表示(図9)。表示中にペインの番号を押すことで移動できる
o	select-pane -t :.+	次のペインに移動
;	last-pane	直前にいたペインに移動
↑	select-pane -U	上下左右のいずれかのペインに移動
↓	select-pane -D	
←	select-pane -L	
→	select-pane -R	
{	swap-pane -U	ペインを入れ替え
}	swap-pane -D	
Ctrl-↑	resize-pane -U	ペインのサイズを変更
Ctrl-↓	resize-pane -D	
Ctrl-←	resize-pane -L	
Ctrl-→	resize-pane -R	

▼図9　ペインの番号を表示(display-panes)



macOSでは、ペインのサイズ変更のショートカットキーがOSのショートカットキーと被っています。OS側のショートカットキーを変更するか、第2章で紹介するカスタマイズ方法でtmux側のキーを変更するようにしてください。

セッション

セッションはウィンドウと同様に作成、移動、名前の設定ができます(**表3**)。特徴的なのは、先ほど紹介したデタッチ・アタッチの機能です。

デタッチは、正確には**detach-client**というコマンドで、tmuxクライアントを終了するコマンドです。これに対してアタッチは**attach-session**となっており、tmuxクライアントを作

▼表3　セッションのショートカットキー(プレフィックスキーのあとに押すキー)

キー	サブコマンド	意味
$	rename-session	セッションの名前を変更
s	choose-tree	セッションの一覧をウィンドウの一覧とともにツリー表示。↑↓でカーソルの移動、←→でツリーの開閉、Enterで選択の確定
(	switch-client -p	前のセッションに移動
)	switch-client -n	次のセッションに移動
L	switch-client -l	直前にいたセッションに移動
d	detach-client	デタッチ

ると同時に指定されたセッションに接続するコマンドです。クライアントとセッションは違うということを覚えておきましょう。

セッションの作成はデフォルトのショートカットでは用意されていないため、コマンドを実行する必要があります。tmuxクライアントを終了した状態では、**tmux**を実行するだけで新規セッションが作成されますが、tmuxクライアントを使用中に新規にセッションを作る場合は、次のコマンドを実行します。

```
$ tmux new-session -d
```

また、セッションの一覧から選択したセッションに移動するコマンドとして、**choose-session**と**choose-tree**の2つがあります。前者は**choose-window**と同様に、単純にセッションの一覧から選択できる機能です。後者はセッションの一覧に加えて、そのセッションのウィンドウの一覧まで見ることができるコマンドです。カーソルキーの左右でウィンドウのツリーを開閉できます(**図10**)。デフォルトのショートカット(**Ctrl-b s**)には、この**choose-tree**が割り当てられています。

セッションを終了するには、次のコマンドを実行します。

```
$ tmux kill-session
```

▼図10　セッションの選択(choose-tree)

```
(0) - FP: 3 windows
(1)    ├-> 0: Haskell "ubuntu-xenial"
(2)    ├-> 1: Idris* "ubuntu-xenial"
(3)    └-> 2: Ocaml- "ubuntu-xenial"
(4) - JVM: 3 windows (attached)
(5)    ├-> 0: Scala* "ubuntu-xenial"
(6)    ├-> 1: Java "ubuntu-xenial"
(7)    └-> 2: Kotlin- "ubuntu-xenial"
(8) - LL: 3 windows
(9)    ├-> 0: Python* "ubuntu-xenial"
(a)    ├-> 1: Ruby- "ubuntu-xenial"
(b)    └-> 2: Perl "ubuntu-xenial"


[JVM] 0:Scala* 1:Java  2:Kotlin-       "ubuntu-xenial" 14:28 18-Jul-17
```

▼図11　コピーモード

```
bind-key    -T prefix M-3      select-layout main-horizontal      [0/89]
bind-key    -T prefix M-4      select-layout main-vertical
bind-key    -T prefix M-5      select-layout tiled
bind-key    -T prefix M-n      next-window -a
bind-key    -T prefix M-o      rotate-window -D
bind-key    -T prefix M-p      previous-window -a
bind-key -r -T prefix M-Up     resize-pane -U 5
bind-key -r -T prefix M-Down   resize-pane -D 5
bind-key -r -T prefix M-Left   resize-pane -L 5
bind-key -r -T prefix M-Right  resize-pane -R 5
bind-key -r -T prefix C-Up     resize-pane -U
bind-key -r -T prefix C-Down   resize-pane -D
bind-key -r -T prefix C-Left   resize-pane -L
bind-key -r -T prefix C-Right  resize-pane -R
ubuntu@ubuntu-xenial:~$
[0] 0:Python  1:Scala- 2:bash*          "ubuntu-xenial" 14:15 18-Jul-17
```

▼図12　コピーモードでの選択中の画像

```
bind-key    -T prefix M-3      select-layout main-horizontal      [0/89]
bind-key    -T prefix M-4      select-layout main-vertical
bind-key    -T prefix M-5      select-layout tiled
bind-key    -T prefix M-n      next-window -a
bind-key    -T prefix M-o      rotate-window -D
bind-key    -T prefix M-p      previous-window -a
bind-key -r -T prefix M-Up     resize-pane -U 5
bind-key -r -T prefix M-Down   resize-pane -D 5
bind-key -r -T prefix M-Left   resize-pane -L 5
bind-key -r -T prefix M-Right  resize-pane -R 5
bind-key -r -T prefix C-Up     resize-pane -U
bind-key -r -T prefix C-Down   resize-pane -D
bind-key -r -T prefix C-Left   resize-pane -L
bind-key -r -T prefix C-Right  resize-pane -R
ubuntu@ubuntu-xenial:~$
[0] 0:Python  1:Scala- 2:bash*          "ubuntu-xenial" 14:16 18-Jul-17
```

コピーモード

最後にコピーモードをマスターしましょう。コピーモードを使用すると、各ターミナルの表示の履歴をスクロールして閲覧し、任意の文字列をpaste bufferにコピー、別のターミナルに貼り付けることもできます。

コピーモードには**Ctrl-b [**で入ることができます。キーを押すと、ペインの右上にバッファの行数が表示されるようになります(**図11**)。この表示がある場合はコピーモードにいる状態です。

コピーモードに入ると、これまでとは異なるショートカットキーが有効になります。キーには「emacs」と「vi」の2種類のモードが用意されています。デフォルトでは「emacs」というモードのキーが有効になっています。ただし、**EDITOR**か**VISUAL**の環境変数にviが設定されている場合は、viモードになりますので注意してください。現在のキーは次のコマンドで確認できます。

```
$ tmux show-window-options -g mode-keys
mode-keys emacs
```

emacsモード

ここでは、デフォルトの「emacs」モードの操作を簡単に紹介します(**表4**)。

▼表4　コピーモード(emacs)のショートカットキー

キー	意味
Ctrl-n/Ctrl-p/Ctrl-f/Ctrl-b	カーソルを移動
↑/↓/←/→	カーソルを移動
Ctrl-Space	選択を開始
M-w	選択範囲をコピーしてコピーモードを終了
Esc	コピーモードを終了

まずはカーソルの移動ですが、カーソルキーが使用できるほか、Ctrl-n、Ctrl-p、Ctrl-f、Ctrl-bといった、Emacsでお馴染みの移動キーも使用できます。コピーは、Ctrl-Spaceで選択を開始し、カーソルを移動させて必要な範囲を選択後(図12)、M-w[注3]でコピーするという流れになります。コピーした内容の貼り付けは、コピーモードに入る場合に入力したキーCtrl-b [の逆側のCtrl-b]を押します。

viモード

続いて「vi」モードも紹介します(表5)。viモードに変更する場合は次のコマンドを実行します。

```
$ tmux set-window-option -g mode-keys vi
```

カーソルの移動はお馴染みのh、j、k、lキーです。選択の開始はSpaceキーとなっており、Enterで選択範囲をコピーします。貼り付けの操作はコピーモードのキーには依存しないため、emacsの場合と同じくCtrl-b]を入力します。

困ったときは

「ショートカットを忘れて困った！」や、「使い方忘れた！」というときのために、ウィンドウやペインといった用語に加え、次の3つのコマンドを覚えておくことをお勧めします。

注3) MはMetaキーと呼ばれるキーで、Altキーや Optionキーが該当します。

▼表5　コピーモード(vi)のショートカットキー

キー	意味
h/j/k/l	カーソルを移動
↑/↓/←/→	カーソルを移動
Space	選択を開始
Enter	選択範囲をコピーしてコピーモードを終了
q	コピーモードを終了

コマンドの一覧を表示

次のコマンドを実行するとtmuxのコマンドの一覧が表示できます。

```
$ tmux list-commands
```

コマンドが思い出せないときのヒントになるかもしれません。量が多いので「window」や「pane」といったキーワードでgrepすると良いでしょう。

ショートカットの一覧を表示

ショートカットキーを忘れた場合は、次のコマンドで現在設定されているショートカットの一覧を表示できます。

```
$ tmux list-keys
```

デフォルトでもそれなりの数のショートカットが設定されているので、コマンドやキーワードで絞ると良いでしょう。

manを読む

次のコマンドでtmuxの使い方の詳細を表示することができます。

```
$ man tmux
```

用語を押さえておけば、manの内容はだいたいわかると思いますので、manだけでも覚えておきましょう。SD

第2章

tmuxを使いこなそう

キーカスタマイズとプラグインで自分好みの環境に

Author 田中 智文（たなかともふみ） 株式会社ビズリーチ
Twitter @tanacasino

tmuxにはさまざまなオプションがあり、多様な項目を自分好みにカスタマイズできます。本章ではtmuxをより使いこなして効率アップするためのカスタマイズの基本と、サードパーティのツールを紹介します。カスタマイズの基本を理解し、Webに転がっている設定をただコピー&ペーストするのではなく、活用できるようになりましょう。

ショートカットキーをカスタマイズしよう!

第1章でtmuxの基本的な使い方を学び、主要なショートカットキーを紹介しました。ここでは、ショートカットキーを変更したり、新たに追加したりする方法を紹介します。自分好みにカスタマイズして効率アップを図りましょう。

プレフィックスキーを変更しよう

まずは、プレフィックスキーを変更しましょう。デフォルトでは、**Ctrl-b**に割り当てられています。しかし、このキーはターミナル上でカーソルを左に移動するキーと被っているため、このキーに慣れている人には非常に不便です。また、GNU Screenから移行した人には、tmuxでもデフォルトの**Ctrl-a**にしたいという方も多くいます[注1]。

プレフィックスキーは、次のコマンドを実行することで変更できます。この例では**Ctrl-a**に変更します。

```
$ tmux set -g prefix C-a
$ tmux unbind C-b
$ tmux bind C-a send-prefix
```

setコマンドはオプションを設定するコマンドです。このコマンドはエイリアス（別名）となっており、元は**set-option**というコマンドです。

1行目は**prefix**というオプションの値を**C-a**、つまり**Ctrl-a**に設定するという意味になります。オプションについての詳細は後述します。

2行目の**unbind**コマンドは**unbind-key**コマンドのエイリアスで、キーバインド（ショートカットキー）を解除します。ここではデフォルトの**Ctrl-b**に設定されたキーバインドを解除しています。逆に3行目の**bind**は**bind-key**コマンドのエイリアスで、ショートカットキーを設定します。ここでは、**Ctrl-a**に**send-prefix**というコマンドを割り当てています。これは、**Ctrl-a**を2回連続で実行した場合は、ターミナルに**Ctrl-a**キーを送るという意味です。プレフィックスキーに使ったキーをtmux内のターミナル側に送りたい場合に使用します。**Ctrl-a**を送ると、たいていの環境ではカーソルキーが先頭に移動します。

設定ファイルを書いておこう

このようにオプションの変更やキーバインドの変更もコマンドで実行できるようになっていますが、このままではtmuxを終了すると設定も消えてしまいます。そこで、設定ファイルを作っておくことができます。設定ファイルはホームディレクトリの直下に「.tmux.conf」というファイルを作成して記述します。

注1）余談ですが、**Ctrl-b**になったのは、tmuxの初期開発時に開発者がGNU Screenを使っていたので、被らないようにしたからとの噂です。

```
# 好きなエディタで編集。ここではvimを使用
$ vim ~/.tmux.conf
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

設定ファイルはサブコマンド以降をそのまま記述するだけです。よくわからないオプションを試す場合、まずはtmux内でコマンドを実行して設定をチェックし、意図した動作になれば設定ファイルに記述するのが良いでしょう。

また設定ファイルの内容を変更した場合は、次のコマンドで読み込み直すこともできます。

```
$ tmux source-file ~/.tmux.conf
```

ただし、後述するオプションの種類によっては、設定の記述を削除した場合に、以前の変更の適用が残るなどのこともあってうまく反映できないこともあります。最終的には次の手順でtmuxサーバのプロセスを再起動して確認しましょう。

```
# tmuxサーバのプロセスを停止
$ tmux kill-server
[server exited]
# 再起動する
$ tmux
```

ショートカットキーの設定方法

プレフィックスキーの変更で使用した**bind-key**と**unbind-key**を使用すれば、自由にショートカットキーを設定できます。次のステップで、よく使用しそうなコマンドを探してショートカットキーを設定しましょう。

①ショートカットキーにしたいコマンドを探す
②そのコマンドが現在ショートカットキーになっているかチェック
③古いショートカットキーを解除
④新しいショートカットキーを設定
⑤設定ファイルに記述

last-windowコマンドのショートカットを**l**から**a**へと変更する具体例で、それぞれのステップを説明します。

■①コマンドを探す

コマンド一覧を表示するか**man**コマンドで詳細を見ましょう。

```
$ tmux list-commands
$ man tmux
```

■②現在のショートカットキーを探す

last-windowコマンドにどのようなショートカットキーが設定されているか探します(コマンドの詳細は後述)。**l**だとわかりました。

```
$ tmux list-keys -T prefix | grep last-window
bind-key    -T prefix l       last-window
```

■③キーを解除する

lのショートカットキーを解除します。

```
$ tmux unbind l
```

■④新しいキーにする

新しいショートカットキー**a**に設定します。

```
$ tmux bind a last-window
```

■⑤設定ファイルに記述

必要に応じて、設定ファイルに追記します。

◆ ◆ ◆

②でショートカットキーを探す際、**list-keys**コマンドを**-T prefix**というオプションを指定して実行しています。これは、「key-table(キーテーブル)オプションが**prefix**」のキーを表示するという意味です。これまでのショートカットはプレフィックスキーのあとにキーを押すものでしたが、これが**prefix**と呼ばれるkey-tableです。ショートカットキーには**表1**の4種類があり、それぞれ有効になるタイミングが異なります。

実は**bind-key**コマンドも、**-T**で**key-table**

を指定することで、prefix以外のキーテーブルを指定できます。たとえばrootを指定すると、プレフィックスキーなしでショートカットを実行できますが、多用するのはあまりお勧めしません。これは、ターミナル内で使用するプログラムのキーと被って操作できないなどの問題が起きやすいためです。ここぞというときだけにしましょう。

コピーモードのキーは少し特殊ですので、ここで簡単に解説します。list-keysコマンドを実行するとわかりますが、コピーモードではsend-keys -Xというコマンドを使っています。これはコピーモードにキーを送るという意味です。send-keys -Xで指定できるキーもコピーモード用に用意されていますが(表2)、ここでは一部を紹介します(図1)。

bind、unbindで指定するオプションがkey-tableのみになったのは、tmuxのバージョン2.4以降です。それ以前のバージョンの場合は、設定が異なるので注意してください。過去のバージョンの設定のほうが難しいため、できるだけ新しいバージョンのtmuxに移行することをお勧めします。

▼表1　ショートカットキーの種類

key-table名	意味
prefix	プレフィックスキーを押したあとに押すことで実行できるショートカットキーの設定
root	プレフィックスキーなしですぐに実行できるショートカットキーの設定
copy-mode	コピーモード時に実行できるショートカットキーの設定。mode-keysオプションがemacsの場合の設定
copy-mode-vi	copy-modeと同じ。こちらはmode-keysオプションがviの場合の設定

▼表2　コピーモード用のキー

キー	意味
cancel	コピーモードを終了
begin-selection	選択の開始
copy-selection	選択範囲をコピー
copy-selection-and-cancel	選択範囲をコピーし、コピーモードを終了
cursor-up	カーソルキーを移動
cursor-down	
cursor-right	
cursor-left	
history-top	履歴の最初の行に移動
history-bottom	履歴の最後の行に移動
page-up	ページ戻る
page-down	ページ進む

▼図1　copy-mode-viのキーに、vを押すと選択が開始されるように設定

```
$ tmux bind -T copy-mode-vi v send-keys -X begin-selection
```

ショートカットで使える便利コマンド

ショートカットキーの実行時にメッセージを表示できると便利なことがあります。たとえば、設定ファイルをリロードするショートカットを考えてみましょう。

```
# プレフィックスキーのあとにRで設定ファイルの読み込み
bind R source-file ~/.tmux.conf
```

このショートカットを設定し、実行しても、何も変化がありません。このコマンドは設定を読み直すだけですので、変更した内容によっては画面に変化がありますが、たとえばキーバインドを設定しただけでは画面に変化がなく、本当に実行されたかどうか判断できません。

そこで使えるのが、display-messageコマンドです。このコマンドを使うと、ステータスラインにメッセージを表示できます(図2)。

```
$ tmux display-message 'Hello World!'
```

bindコマンドで複数のコマンドを実行させるには;で区切ります。エスケープするために\;という指定になります。たとえば設定ファイルの読み込み後にメッセージを表示するには、設定ファイルに次のように記述します。

```
bind R source-file ~/.tmux.conf \; ➥
display-message 'Reloaded config!'
```

▼図2　display-messageの表示

```
ubuntu@ubuntu-xenial:~$ tmux display-message 'Hello World!'


Hello World!
```

▼図3　confirm-beforeが使われているコマンドを確認

```
$ tmux list-keys -T prefix | grep confirm-before
bind-key    -T prefix &       confirm-before -p "kill-window #W? (y/n)" kill-window
bind-key    -T prefix x       confirm-before -p "kill-pane #P? (y/n)" kill-pane
```

このほかにも、**kill-window**や**kill-pane**のように、間違って実行したら困るコマンドを、確認を出してから実行する**confirm-before**コマンドがあります。デフォルトでも、**図3**のように使われています。**confirm-before**では、**-p**オプションで表示するメッセージを指定できます。

この際使用する**#W**や**#P**はtmuxのコマンドで使える変数(のエイリアス)で、メッセージなどの書式を受け取るコマンドに渡します。**display-message**コマンドで試してみると良いでしょう。次はウィンドウの名前を表示する変数の指定方法です(1行目と2行目は同じ動作をします)。

```
$ tmux display-message "Hello Window ➥
Name: #W"
$ tmux display-message "Hello Window ➥
Name: #{window_name}"
```

変数は基本的には、**#{variable_name}**の形式で指定しますが、よく使われる変数にはエイリアスが用意されており、**#W**のように**#{}**を省略できるものがあります。tmuxで使用できる主要な変数には**表3**があります。**confirm-before**では、簡単な**y/n**しか受け取れませんが、よりインタラクティブなことがしたい場合は、**command-prompt**コマンドを使います。このコマンドを使うと、ステータスラインからユーザの入力を受け取ることができるようになります。試しに次のコマンドを実行してみましょう。

▼表3　tmuxのコマンドで使える変数

変数名	エイリアス	内容
window_name	#W	ウィンドウの名前
window_index	#I	ウィンドウの番号
pane_index	#P	ペインの番号
session_name	#S	セッションの名前
host	#H	ホスト名

```
$ tmux command-prompt
```

すると、ステータスラインが次のように変わったはずです。

```
:
```

ここでtmuxのサブコマンドを呼び出すことができます。続けて次のように入力して[Enter]を押してみます。

```
:display-message 'Hello'
```

これでコマンドが実行できました。

また任意の値を受け取って、あらかじめ決めておいたコマンドに入力を渡して実行することもできます。たとえばウィンドウ名の変更をショートカットキーにする場合は、次のように組み合わせます。

```
$ tmux command-prompt -I "#W" "rename-➥
window '%%'"
```

-Iには、入力部分に初期値として設定したい値を記載します。**#W**は先ほど出てきた変数名です。この例では、初期値にウィンドウ名を入れてコマンドプロンプトを表示、ユーザの入力を

受け取ったら、それをrename-windowコマンドに渡して実行します。%%の部分がユーザの入力と置換されるしくみです(図4)。

オプションの種類を覚えておこう!

本章序盤のプレフィックスキーの変更の際、set-optionコマンドを使ってオプションの値を変更しました。tmuxではさまざまなオプションが用意されており、細かなカスタマイズができます。tmuxのオプションは以下の3種類あって少し複雑ですので、ここで一度整理します。

①サーバオプション
②セッションオプション
③ウィンドウオプション

サーバオプションはtmuxサーバに紐づくオプションで、変更はグローバルに適用されます。セッションオプションはセッションに紐づくオプションで、特定のセッションにおいてだけオプションを変えることができます。同様に、ウィンドウオプションはウィンドウに紐づくオプションです。

セッションオプションとウィンドウオプションには、すべてのセッションやウィンドウを変更する方法として、-gオプション(グローバルオプション)が用意されています。設定ファイルに書く際には、特定のウィンドウやセッションでだけ設定を変えるということはないので、基本的に-gオプションを指定する必要があると考えましょう。ただし-gで指定したオプションは、あくまでウィンドウやセッションのデフォルト値という扱いになるので、個別のウィンドウやセッションに明示的にオプションを設定した場合はそちらが優先されます。設定ファイルではなく、コマンドでオプションの動作を確認する際は気を付けてください。

オプションの指定方法

たとえば、序盤のプレフィックスキーの変更ではprefixというオプションをC-aに設定しましたが、その際次のように、-gオプションを指定していました。

```
set -g prefix C-a
```

prefixはセッションオプションです。セッションオプションということは、個別のセッションで変更可能なオプションということで、-gを指定しなかった場合は、現在のセッションにおいてだけプレ

▼図4 command-promptの使用例

```
ubuntu@ubuntu-xenial:~$ tmux command-prompt -I "#W" "rename-window '%%'"


(rename-window) bash
```

COLUMN run-shellで外部のコマンドと連携

外部のコマンドを呼び出すことができるrun-shellコマンドを紹介します。任意のスクリプトが実行できるので、分岐処理を伴うなど、より高度な処理をショートカットから呼び出すことができます。のちほど紹介するプラグインなどは、このコマンドを使って便利な機能を追加していることが多く、覚えておいて損はありません。コマンドプロンプトやショートカットから実行したrun-shellコマンドの出力はコピーモードで表示されます。ショートカットから呼び出した場合も同じです。tmux command-promptを実行したあと、ステータスラインにて、

```
:run-shell 'echo HELLO'
```

と入力すると、コピーモードで「HELLO」と表示されるはずです。command-promptとrun-shellを組み合わせることでより幅広い処理で活躍できます。

フィックスキーを変えることができます。次の設定では別のセッションに移動した場合、別のプレフィックスキーになります。

```
# 現在のセッションだけプレフィックスキーをCtrl-tにする
$ tmux set prefix C-t
```

各オプションを指定するコマンドは、**表4**のようになっています。

ウィンドウオプションに2つずつコマンドがある点に注意しましょう。またオプションの設定を外したいときは**-u**オプション(**unset**)を使います。現在のオプションの設定状況を見たい場合は、**show-options**コマンド(エイリアス：**show**)や**show-window-options**コマンド(エイリアス：**showw**)を使います。使い方は**set-option**と同じで、一覧と個別取得のオプション指定ができます。

```
# セッションオプションのグローバル値(デフォルト値)の一覧
$ tmux show -g
# セッションオプションのグローバル値(デフォルト値)を
個別に取得
$ tmux show -g prefix
# 現在のセッションに設定しているセッションオプションを取得
$ tmux show
もしくは、
$ tmux show prefix
```

主要な各オプションを**表5**～**7**に紹介します。そのほか詳細を含めてどのようなオプションがあるかは、**man tmux**で確認しましょう。

ステータスラインのカスタマイズ

一通りのオプションの指定方法を理解したところで、ステータスラインの表示項目や見た目のカスタマイズをしてみましょう。ステータスラインの主要なオプションは**表8**

▼表4　各オプションの指定方法

コマンド	説明
set-option -s	サーバオプションを設定
set-option	セッションオプションを設定
set-option -g	グローバルセッションオプションを設定
set-option -w	ウィンドウオプションを設定
set-window-option	
set-option -wg	グローバルウィンドウオプションを設定
set-window-option -g	

▼表7　ウィンドウオプション

オプション	説明
mode-keys	コピーモードのキーとしてemacsかviのどちらを使うか設定
wrap-search	コピーモードの検索時にループするかどうかを設定

▼表5　サーバオプション

オプション	説明
default-terminal	使用するターミナルを指定。指定がない場合は環境変数**TERM**の値が使用される。最近の環境では「screen-256color」に設定しておくことが多く、とくに問題がなさそうであればこちらを設定しておくと良し
escape-time	Esc を押したあとの待ち時間。デフォルトでは500(ミリ秒)。メタキーの入力の一部かどうかを判断するために待ち時間が設定されている。VimでEscを押したい場合に待ち時間が発生してしまうので「0」にする人が多い

▼表6　セッションオプション

オプション	説明
prefix	プレフィックスキーを指定
history-limit	コピーモードで履歴を見る際の上限値。大きくしておけば、さかのぼれる範囲が増える
default-shell	新しくウィンドウやペインを作った際に使用するシェルを指定
status	ステータスラインを表示するか否か。「off」を指定すると表示が消える
status-keys	**command-prompt**を使った場合など、ステータスラインで文字入力する際のキー入力のモードを指定。コピーモード同様、「emacs」と「vi」が用意されている
display-panes-time	**display-panes**コマンドで表示するメッセージの秒数。デフォルトは1,000ミリ秒
mouse	マウスを使用してペインの移動やリサイズを可能にする機能を有効にする。「on」を指定することで有効になる

のとおりです。これらはセッションオプションになります。

status-leftと**status-right**には、固定の文字列だけではなく、動的な値を表示することもできます。たとえば、先ほど紹介した変数を使うこともできます。

```
$ tmux set -g status-left "[ #W ] "
```

また、外部のコマンドを実行した結果も表示できます。指定するには、**#(command)**という書式を使います。これらのコマンドは**status-interval**で指定された間隔で実行されます。

```
# 現在のユーザ名を表示
$ tmux set -g status-left "[#(whoami)] "
```

そのほかにも**#[]**という書式を使うことで、色や属性を変更できます。次のような設定ができます。

```
$ tmux set -g status-left "#[fg=yellow,bg=black]#(whoami)#[default] : "
```

#[fg=yellow,bg=black]は前景の色を黄色にして、背景の色を黒にするという指定です。この指定があると、その後ろの出力にこの書式が適用されます。今回の例では**#(whoami)**の文字が黄色になり、文字の背景色が黒になります。そのあとの**#[default]**で元の書式に戻すことで、後ろの **:** にはもともとの色が使用されます。

色や属性のみを指定する場合は、**style**を使います。ステータスラインでは、**status-style**、**status-left-style**、**status-right-style**を使って、たとえば、

```
$ tmux set -g status-left-style fg=yellow,bg=black
```

のように指定できますが、表示する内容ごとにより細かな設定をしたい場合は、先ほどの例のように**status-left**や**status-right**で直接指定する必要があります。

色は**black**、**red**、**green**、**yellow**、**blue**、**magenta**、**cyan**、**white**が指定できるほか、ターミナルによっては、**colour0**から**colour255**までの256色を指定できます。属性は**none**を指定すると「指定なし」となり、そのほかはカンマ区切りで太字を表す**bright**(もしくは**bold**)、**dim**、**underscore**、**blink**、**reverse**、**hidden**、**italics**、**strikethrough**を指定できます。それぞれ実際に試してみると良いでしょう(**図5**)。

```
# fgにblinkとboldを追加して文字を太字にし、点滅させる
$ tmux set -g status-left "#[fg=yellow,blink,bold,bg=black]#(whoami)#[default] : "
```

▼図5　ステータスラインの表示とスタイル

```
ubuntu@ubuntu-xenial:~$ tmux set -g status-left "#[fg=yellow,bg=black]#(whoami)#[default] : "
ubuntu@ubuntu-xenial:~$

ubuntu : 0:bash*                                    "ubuntu-xenial" 13:19 17-Jul-17
```

▼表8　ステータスラインの主要なオプション

オプション	説明
status-interval	ステータスラインを更新するインターバル(秒)。デフォルトは15秒。間隔を短くすればするほど頻度が増え、負荷が上がるので注意
status-position	ステータスラインを上部(top)と下部(bottom)のどちらの位置に表示するかを設定。デフォルトは「bottom」
status-left	ステータスラインの左側に表示する内容を設定
status-right	ステータスラインの右側に表示する内容を設定
status-left-length	status-left/status-rightのそれぞれの最大長を指定。これより長くなった場合はこの値でカットされる
status-right-lenght	

そのほかにも、ステータスラインのウィンドウの表示などをカスタマイズするwindow-status-formatやwindow-status-current-formatなどがありますが、書式などは基本的に同じですので、基本を押さえておけばだいたいのカスタマイズができます。

tpmを使ってプラグインを入れてみよう

これまでの内容で、tmuxはオプションやショートカットキーの設定により、いろいろなカスタマイズができることを確認しました。ここではtmuxをより便利にするためのライブラリをプラグインとして管理できるようにしてくれる、tpm(Tmux Plugin Manager)を使ったプラグインのインストール方法を紹介します。tpmを使うことで、プラグインとして便利な機能や設定を追加でき、それらのインストール・アップデートも簡単に実行できるようになります。

tpmをインストールする

まずはgitを使ってtpmをインストールしましょう。次のコマンドでtpmのソースを取得します。

```
$ git clone https://github.com/➥
tmux-plugins/tpm ~/.tmux/plugins/tpm
```

続いて、**リスト1**の設定内容を~/.tmux.confの末尾に追記し、tmuxを再起動して設定ファイルを読み直します。

これでtpmが使えるようになりました。プレフィックスキーのあとにI(大文字)を押すことでプラグインがインストールされ、次のメッセージが表示されます。

```
TMUX environment reloaded.
Done, press ENTER to continue.
```

使いたいプラグインを設定に追記した場合は、毎回インストールが必要ですのでプレフィックスキー＋Iは覚えておいてください。

また、プレフィックスキーのあとにUを押すことで、インストールしているプラグインのアップデートができます。アップデートは、アップデートしたいプラグインの名前を打ち込んでEnterを押すか、「all」と打ち込んでEnterを押すことですべてのプラグインをアップデートできます(**図6**)。

さて設定に戻りますが、

```
set -g @plugin 'tmux-plugins/➥
tmux-sensible'
```

ではtmux-plugins/tmux-sensibleというプラグインを指定しています。プラグイン名のスラッシュの前半はGitHubのユーザ名もしくは組織名を指し、後半がリポジトリ名を指しています。GitHub上のリポジトリを自由に指定できるので、自分で独自に作成したプラグインをtpmで管理することもできます。tpmを作っている組織名はtmux-pluginsですので、tpmで使えるプラグインをhttps://github.com/tmux-pluginsで探すことができます。

▼リスト1　tpmの設定

```
# プラグインのリストを記述する
# tpm自身も対象にする
set -g @plugin 'tmux-plugins/tpm'
# tmux-sensibleというプラグインをインストールする例
set -g @plugin 'tmux-plugins/tmux-sensible'

# tpmを初期化する(この設定はtmux.confの最下行に記述)
run '~/.tmux/plugins/tpm/tpm'
```

▼図6　tpmでプラグインをアップデート

```
Installed plugins:                                    [0/0]
  tpm
  tmux-sensible
  tmux-yank
Type plugin name to update it.
- "all" - updates all plugins
- ENTER - cancels


plugin update: all
```

tmux-sensible

先ほどの例でインストールしたプラグインの「tmux-sensible」は、公式ページ[注2]で、「A set of tmux options that should be acceptable to everyone.」と説明されています。ざっくり言うと、tmuxユーザにお勧めの設定を適用してくれるプラグインです。どんなオプションにどの値を入れるといいのかわからない初心者は、とりあえずインストールして使うのも良いですし、ソースを見てオプションについて学ぶのも良いと思います。

tmux-yank

コピーモードを使用してコピーしたテキストは、tmux内だけで使えるpaste-bufferにコピーされるしくみです。これをOSのクリップボードと連携させる設定を自動化してくれるのが「tmux-yank」プラグインです。OSによって異なる設定が必要になることが多いので、OSの差分を自動的に吸収してくれるプラグインとして使うことができます。

macOSでは、バージョンによっては、「reattach-to-user-namespace」というプログラムのインストールが必要です。Homebrewを使ってインストールできます。

```
$ brew install reattach-to-user-namespace
```

現在のところ「macOS 10.12: Sierra」では必要となっています。詳細は公式ページ[注3]で確認しましょう。

プラグインをインストールすると、コピーモードのキーバインドにyが追加され、OSごとに適した方法でクリップボードにコピーする機能が実行されるようになります。

次の設定を追記後、プレフィックスキー＋Iでインストールできます。

```
set -g @plugin 'tmux-plugins/tmux-yank'
```

tmux-resurrect

「tmux-resurrect」を使うことで、セッション・ウィンドウ・ペインの情報を保存しておき、tmuxサーバを終了しても、再起動した際に復元できます。とくにローカル開発などで、よく使うウィンドウやペインが決まっていて、OSを再起動したあとにすぐに元の状態に戻したいといった場合に非常に便利です。各ペインのワーキングディレクトリの位置も戻してくれるので、復帰後すぐにビルドやテストができます。

どんな機能かピンと来ない方は、公式ページ[注4]に動画がありますので、これを見るのが早いと思います。ぜひ一度チェックしてみてください。

次の設定を追記後、プレフィックスキー＋Iでインストールできます。

```
set -g @plugin 'tmux-plugins/➥
tmux-resurrect'
```

使い方も非常に簡単で、プレフィックスキー＋Ctrl-sで現在の状態を保存し、プレフィックスキー+Ctrl-rで保存した状態を復元してくれます。ローカル開発環境にはぜひ導入してみてください。

複数のターミナルを同時に操ろう!

tmuxを使用することで、複数のターミナルでの並行作業が楽にできるようになります。ここではさらに一歩進んで、同じような並行作業を同時に実行する方法や、並行作業をするためのターミナルを一気に作る、といった方法を紹介します。

ペインへの入力を同期する

Webサーバやアプリケーションサーバのよう

注2） URL https://github.com/tmux-plugins/tmux-sensible
注3） URL https://github.com/tmux-plugins/tmux-yank
注4） URL https://github.com/tmux-plugins/tmux-resurrect

に、ロードバランサの下に同じ構成の同じアプリケーションが複数動いているサーバがあり、それぞれにログインして同じパスにあるログを確認するという場面があるかと思います。その際に便利なのが、ペインの入力を同期するウィンドウオプションのsynchronize-panesです。これを「on」にすると、入力内容が、そのウィンドウ内のすべてのペインに送られます。この機能を使って、ウィンドウ内に同じ構成のサーバにそれぞれログインしたペインを作っておけば、ログをtailするコマンドや、プロセスの再起動といったコマンドを同時に行うことができます(図7)。

その場でさっと同期させる場合は、プレフィックスキーのあとに:を押して、command-promptを起動し、そこでset synchronize-panes onコマンドを実行します。同期を解除するには、synchronize-panesオプションに「off」を設定します。よく同じ構成のリモートサーバにログインして作業するのであれば、これらの設定にショートカットキーを割り当てておけば、効率良くペイン同期のon/offを切り替えられて作業効率がアップすること間違いなしです。

xpanesを使って複数の作業を効率化する

ペインの同期では、ペインを自分で作ってから実行する必要がありましたが、そもそも必要なペインの数とそこで実行するコマンドをまとめて指定したいものです。この願いを叶えてくれるのが「xpanes」[注5]というソフトウェアです。xpanesコマンドを使うことで、たとえばペインを2つ作り、それぞれでtopとvmstatを実行するといったことや、複数のホストにSSHログインする、複数のホストに同時にpingを送ってホストごとにペインを分割するといったこともできます。

macOSでは次の手順でインストールできます。

```
$ brew tap greymd/tools
$ brew install tmux-xpanes
```

xpanesの基本的な使い方を紹介します。まずは次のコマンドを実行してみてください。

```
$ xpanes -c 'echo {}' A B C D
```

実行すると新しいウィンドウが作成され、ペインが4つに分割され、それぞれにechoコマンドが実行されているはずです(図8)。「pane 0」ではecho Aが、「pane 1」ではecho Bが……といった感じです。このように、実行したいコマンドを-cオプションで渡すと、{}が後ろの引数にそれぞれ置き換えられます。

このときの注意事項として、先ほど説明したsynchronized-panesがデフォルトで「on」になることがあります。同期されると困る場合は-dオプションを一緒に指定すれば同期しません。

次は、複数のホストに同時にpingを送るコマンドの例です。

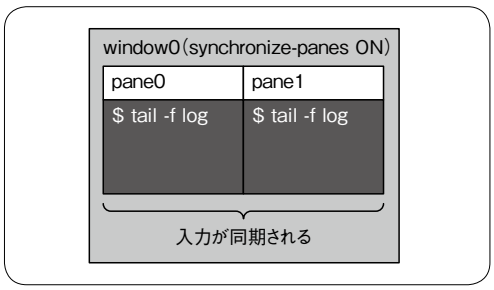
```
$ xpanes -c 'ping {}' 192.168.10.{1..4}
```

192.168.10.{1..4}部分はシェルの展開で、次のように展開されます。上記コマンドではこれを組み合わせることで、複数のホストの指定も簡単にできています。

```
$ echo 192.168.10.{1..4}
192.168.10.1 192.168.10.2 192.168.10.3 192.168.10.4
```

同時に複数のホストにSSHログインするには、pingコマンドをsshコマンドに置き換えれ

▼図7　ウィンドウ内のペインの入力を同期する



注5) URL https://github.com/greymd/tmux-xpanes

ばOKです。

```
$ xpanes -c 'ssh {}' 192.168.10.{1..4}
```

次のように**-e**オプションを使うと、引数に指定したコマンドがそのまま実行されます。ローカル開発などで、複数のコマンドを並べたい場合に便利です。

```
# top、nettop、watch df -hをそれぞれ別のペインで実行
$ xpanes -e "top" "nettop" "watch df -h"
```

このようにさまざまな使い方ができる非常に便利なツールです。自分の作業環境に合わせて使い方を工夫して作業効率をアップさせましょう。より詳細な使い方は公式ページのREADMEにまとまっているので、熟読をお勧めします。

トラブルシューティング

最後に、tmuxを使い始めた人が遭遇しやすい問題について解決方法を紹介します。

tmuxを二重に起動してしまった

tmux内のターミナルから、さらに別のサーバにログインしてtmuxを起動してしまうことがあります。プレフィックスキーが同じキーになっていると、どうやっても奥のtmuxに届かず、閉じることもできない、といったケースです。そんなときは慌てずに、奥のtmuxのターミナルでコマンドを打ち込んでデタッチしましょう。

```
$ tmux detach-client
```

ショートカットキーはあくまでコマンドですから、使えないときは通常のコマンドを使って切り抜けるということを覚えておいてください。また、間違って二重に起動したわけではなく、むしろ二重で使いたいといった場合は、プレフィックスキーを一時的に変更することで回避できます。この場合は、手前か奥か、どちらかのtmuxのプレフィックスキーを、オプション設定コマンド(**set**)で変更しましょう。

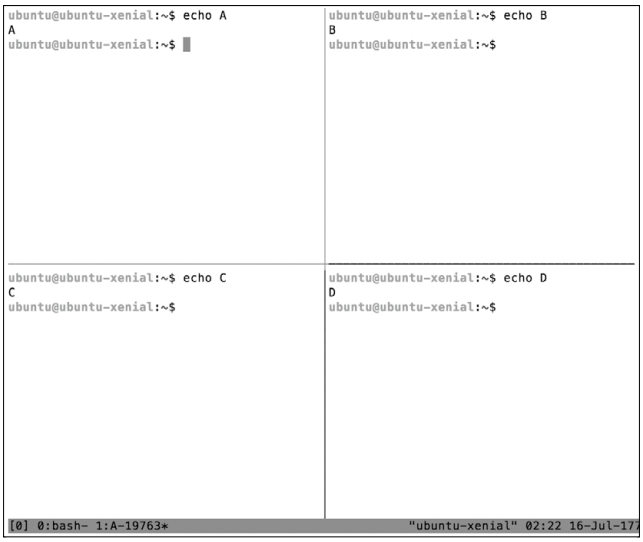
ショートカットキーが効かない

bind-keyコマンドで設定したキーが有効にならない、という話もよく聞きます。これはそのショートカットキーが、OSやターミナルのアプリで使われていて、そちらに取られてしまうというケースです。ショートカットキーが別の動きをする場合はその可能性が高いので、OSやアプリの設定を見直して被らないように設定しましょう。

設定が有効にならない

source-fileでリロードしたのに、前の設定が残っていて……というパターンもあります。設定ファイルを触っているときは、tmuxを再起動してみることも試してください。また、**-g**オプションを付けずに、ウィンドウオプションやセッションオプションをコマンドで指定した場合、そちらが優先されてしまい、思った動作をしなかったという場合もあります。リロードを過信しないように気を付けましょう。SD

▼図8　xpanesコマンドを実行して分割されたペイン



第3章

屏風（びょうぶ）のような美麗なスクリーンで快適な操作を！

tmuxを気軽に使える「Byobu」

Author 柴田 充也（しばた みつや） ㈱創夢

本章では、GNU screenやtmux用のラッパースクリプトである「Byobu（屏風）」について紹介します。tmuxを扱う際、個々のカスタマイズを行わなくても、Byobuを使うだけで快適な操作環境を構築できます。Byobuを起動するだけでファンクションキーを利用したウィンドウ・ペイン管理ができます。ステータスラインには、よく参照される情報が最初から表示されます。

はじめに

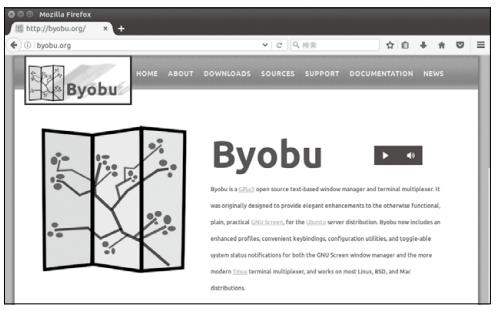
設定ファイルのカスタマイズに没頭していると、いつの間にかタイムリープしていたという経験はありませんか？　たとえば第1、2章にて紹介した「tmux」は設定のカスタマイズ性が高く、その設定変更が直接作業効率に影響し、うまくハマったときには筆舌に尽くしがたい快感を得られるツールです。そのためシェルやエディタの設定と比類するぐらいにその中毒性が危険視されています。

本来は人生のクオリティを向上させるためのツールであるはずなのに、つきあい方を間違えた結果、設定ファイルの暗黒面へと転落していく。そんなドロドロの関係を築く技術者をしばしば見かけます。心のどこかではわかっているんです。「このままではいけない」と。ツールとの関係を見直すべきだと。心ではわかっていても頭がついていきません。ふと気がつくと「こうすれば効率的じゃないか」と誘惑に抗えずに設定ファイルへと手を伸ばしてしまうのです。

本章ではそんな悩みをお持ちの方への解決策として「Byobu[注1]」を紹介します（**図1**）。「Byobu」はCanonicalのDustin Kirklandが開発している「tmuxやscreenを初心者でも簡単に使えるようにしたラッパースクリプト」です。名前の由来は日本語の「屏風（びょうぶ）」であり、屏風のような美麗なスクリーンを、設定ファイルを直接編集することなく簡単に構築できるツールとして作られています。そう、設定ファイルを編集することなく、です。Byobuを起動すれば最初から「それっぽい」設定が施された状態で起動します。たとえばステータスラインに表示される情報をカスタマイズしたい場合も、ファンクションキーとカーソルキーを「ぽちぽち」するだけで完了です。「Byobuとはそういうものだから」というエクスキューズを用意することで、ふいに湧き上がるカスタマイズ欲を抑えることができます。

ここまで簡単だと「何か裏があるのでは？」と不安になるかもしれません。安心してください。Byobuはサーバ版のUbuntuに最初からインストールされているぐらい安全です。それにみんな使ってますから大丈夫。まずはちょっとだけ

▼図1　Byobuの Web サイト



注1）http://byobu.org/

試してみるといいですよ。これで「設定疲れ」の人生から解放されるのです。さあ一緒に心の安寧を得ましょう。

Byobuのインストール

Byobuは GNU screenやtmux用のラッパースクリプトです[注2]。Byobuを起動すると、screenもしくはtmuxの設定ファイルをセットアップしscreenもしくはtmuxを起動します。標準の設定ファイルで基本的なステータスラインの設定やショートカットの割り当てを行うため、「インストールしてすぐに使える」状態を実現できます。ですからscreenやtmuxが動く環境であれば、どこでもByobuをインストールできます。ArchやFedora、Gentooといった主だったLinuxディストリビューションであれば、ほかのソフトウェアと同じ方法でインストールできますし、macOSもHomebrewを使ってインストールできます[注3]。

Ubuntuの場合、サーバ版やクラウドイメージであれば最初からインストールされています。デスクトップ版であればリポジトリからbyobuパッケージをインストールしてください。もし最新版を使いたい場合は、PPA経由でインストールできます(**図2**)。

byobuコマンドを実行すると新しいセッションでByobuを立ち上げます。デフォルトのバックエンドはtmuxですので、実際には新しいセッションでtmuxが起動することになります。基本的な操作はtmuxと同じです。ただエスケープキー(プレフィックスキー)がF12になっています。最初に覚えておくべきことは次の3つです。

- F9でByobu設定メニューの表示
- Shift-F1でヘルプの表示(qで閉じる)
- 標準のエスケープキーはF12

ヘルプについてはF9からでもアクセスできます。またエスケープシーケンスもF9のメニューから変更できます。ですから実質的に最低限F9キーだけ覚えておけばなんとかなるでしょう。

Byobuをデスクトップ環境で利用する場合、原則としてどの端末ソフトウェアを使ってもかまいません。Ubuntuの場合はDashから「Byobu」を検索するとD-Bus経由でGNOME端末が起動し、その中でByobuが立ち上がります。

Byobuの自動起動

byobu-enableコマンドを実行すると、ログインしたときに自動的にByobuを立ち上げます。またデタッチすると自動的にログアウトします。自動立ち上げを解除したい場合は**byobu-disable**を実行してください。これらのコマンドは単に~/.profileの末尾にByobuを立ち上げるスクリプトを追加もしくは削除するだけです。F9からも設定を変更できます。「Byobuは現在ログイン時に起動しません(有効にする)」を選択しスペースキーを押すと自動立ち上げになり、「Byobuは現在ログイン時に起動します(無効にする)」を選択すると無効化されます。

一時的に自動起動を無効化したい場合は、「~/.byobu/disable-autolaunch」ファイルを作ってください。

エスケープキーの変更

Byobuにおいてエスケープキーは F12に設定されています。つまり新規ウィンドウを生成したい場合はF12 cと入力します。あとで紹介するByobuのショートカットがファンクションキーで統一されているためです。

エスケープキーをF12から変更したい場合は、

注2) もともとはscreen-profilesというscreen用の設定集でしたが、名前をByobuに改名したうえでtmuxもサポートするようになりました。
注3) Creators Update以降であればBash on Windows上でも動作するはずです。

▼図2 byobuパッケージのインストール

```
$ sudo add-apt-repository ppa:byobu/ppa
$ sudo apt update
$ sudo apt install byobu
```

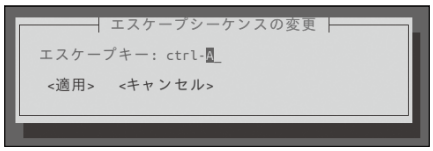
F9で設定メニューを表示し、「エスケープシーケンスの変更」を選択しましょう（**図3**）。ただし変更できるのは「ctrl-(KEY)」の「(KEY)」の部分だけです。「ctrl」はこのインターフェースからは変更できません。大文字／小文字は自動的に両対応となります。

もしF12に戻したい場合は**byobu-ctrl-a**コマンドを実行します（**図4**）。**図4**の(1)のScreenモードがエスケープキーをCtrl-aにするモードで、(2)のEmacsモードがエスケープキーをF12にし、Ctrl-aを行頭へのジャンプとするモードです。引数に「**screen b**」と渡すと、Ctrl-bがエスケープキーになります。

バックエンドの切り替え

Byobuはtmuxとscreenの両方に対応しています。インストール直後はtmuxを使うようになっていますので、screenのほうがよい場合は**byobu-select-backend**で切り替えましょう。一時的にtmuxもしくはscreenを起動したければ**byobu-tmux**や**byobu-screen**コマンドを使用します。

▼図3　エスケープシーケンスの変更



▼図4　エスケープキーをF12に戻す

```
$ byobu-ctrl-a
Configure Byobu's ctrl-a behavior...

When you press ctrl-a in Byobu, do you want it to operate in:
    (1) Screen mode (GNU Screen's default escape sequence)
    (2) Emacs mode  (go to beginning of line)

Note that:
  - F12 also operates as an escape in Byobu
  - You can press F9 and choose your escape character
  - You can run 'byobu-ctrl-a' at any time to change your selection

Select [1 or 2]:
```

豊富なステータスたち

Byobuの便利な点の1つに、画面下部に常時表示されるステータスライン（**図5**）に表示できる豊富なステータス表示機能があり、それをワンタッチでオン／オフできます。F9で表示される設定メニューの「ステータス通知の切り替え」（**図6**）でコントロールできます（**表1**）。

これらの情報は**byobu-status**コマンドが一括して収集・出力します。有効になっているステータスのみを取得し、ステータスラインの右端もしくは左端に出力するわけです。**status-interval**で変更できるステータスラインの更新間隔について、Byobuの場合は1がセットされます。つまり1秒間隔で**byobu-status**コマンドが呼び出されることになります。ただしすべてのステータスの取得を常に1秒間隔で行う

▼図6　F9キーからステータス通知の切り替えを選択したところ

▼図5　Byobuのカラフルなステータスライン

▼表1　Byobuでサポートするステータスコマンド

ステータス名	概要
apport	Apport用のクラッシュデータが存在する場合に通知
arch	ホストのアーキテクチャ(uname -mの結果)
battery	バッテリの残容量(百分率)と充電状態
cpu_count	getconf _NPROCESSORS_ONLNか/proc/cpuinfoから取得したCPUの数
cpu_freq	sysfsもしくは/proc/cpuinfoから取得したCPU周波数
cpu_temp	hwmonなどから取得したシステム温度
custom	~/.byobu/bin以下の「数字_名前」なスクリプトを実行した結果
date	~/.byobu/datetime.tmuxのフォーマットに合わせた日付
disk	dfコマンドをもとにしたディスクの使用量(環境変数MONITORED_DISKで対象指定可)
disk_io	/sys/block/*/statから算出したディスクI/O
distro	/etc/os-releaseなどから取得したディストリビューション名
ec2_cost	稼働時間や通信量から見積もったAmazon EC2の料金
entroypy	/proc/sys/kernel/random/entropy_availの値
fan_speed	hwmonなどから取得したファンの回転速度
hostname	ホスト名(hostname -sの結果もしくはAWSのpublic-hostname)
ip_address	ipコマンドを使ったIPv4もしくはIPv6アドレス
load_average	/proc/loadavgなどから取得した直近1分のロードアベレージ
logo	ディストリビューションごとのロゴマーク
mail	/var/spool/mail/以下に未読メールが貯まっている場合に通知
memory	/proc/meminfoなどから取得したメモリの使用量
network	ネットワーク通信が発生したときの送信サイズ・受信サイズ
processes	起動しているプロセスの数(/proc/[0-9]*の数)
raid	/proc/mdstatから取得したRAIDの状態
rcs_cost	起動時間や通信量から見積もったRackspace Cloud Serverの料金
reboot_required	自動アップデート中や要再起動などのフラグの通知
release	/etc/os-releaseなどから取得したリリースバージョン
services	Eucalyptus用に起動中のサービスを表示
session	tmux/screenのセッション名
swap	/proc/meminfoから取得したスワップ使用量
time	~/.byobu/datetime.tmuxのフォーマットに合わせた時刻
time_utc	協定世界時の時刻(date -u +%H:%Mの結果)
updates_available	OSのパッケージ管理システムから取得した更新が必要なパッケージ数
uptime	/proc/uptimeなどから取得した連続稼働時間
users	sshでリモートログインしているユーザ数(pgrep sshdの結果から算出)
whoami	現在のユーザ名(whoamiの結果)
wifi_quality	iwconfigで取得したWi-Fiの信号強度

のは非効率です。そこでByobuは、/dev/shm以下に取得したステータスのキャッシュを保存しておき、必要に応じてキャッシュを利用するしくみを採用しています。またステータスの更新頻度は/usr/lib/byobu/include/shutilの`status_freq()`にて、ステータスごとに管理しています。archのように変更する可能性がないステータスについては、初回に取得した情報をそのまま使い続けます。

`custom`ステータスを使うと~/.byobu/bin以下に作成した「数字_名前」という名前のスクリプトを5秒間隔で実行します。スクリプトファイル名の数字の部分は、そのスクリプトの実行間隔を秒単位で指定します。これにより異なるスクリプトを異なる間隔で実行し、ステータスラインに表示できます。

Byobuのキーバインディング

Byobuのもう1つの特徴が「ファンクションキーベースのショートカットキーバインディング」です。tmuxにはさまざまな機能が存在しますが、これらがすべてショートカットキーとして割り当てられているわけではありません。必要に応じてユーザが設定ファイルを編集することで、特定のキーコンビネーションを特定の機能に割り当てることになります。むしろいかに自分にとってベストなキーバインディング設定を編み出すかが、tmuxの醍醐味と言えるでしょう。より使う機能はより打ちやすい場所のキーを使い、複数の機能を組み合わせる場合はタイピングのリズム感も考慮し、ほかのツールの操作との整合性をもたせ、さらに覚えやすく体に馴染みやすい設定を。世の中のハッカーたちは、日夜そのような研鑽（けんさん）という名の現実逃避（はげ）に励んでいるわけです。

Byobuはそんな「玄人の嗜（たしな）み」を地平線の彼方に追いやり、ただただ覚えやすさを重視して、機能ごとに分類しファンクションキーに割り当てたバインディングを最初から設定しています。ファンクションキーなんてホームポジションから遠いですし、キーボードによってはほかのキーと組み合わせないと入力できないかもしれないキーです。おそらくゲーマーと一部のソフトウェアの愛好家、それに日本語入力ユーザ以外には「存在感のない子」扱いされているようなキーです。Byobuはそんな不遇のキーたちに、ウィンドウ・ペイン管理を始めとしたtmuxの重要な機能を割り当てています。

ちなみにショートカットキーのリストはShift-F1で確認できます。ここからはショートカットについて個別に見ていきましょう。ヘルプの「Up、Down、Left、Right」はそれぞれカーソルキーの↑、↓、←、→に読み替えてください。また、キー操作の後の括弧中にtmuxの該当コマンドを記述しておきますので、独自ショートカットを作成する際の参考にしてください。

また、ターミナルソフトウェアがここで挙げたショートカットキーを奪ってしまうこともあります。たとえばGNOME端末の場合、編集の「キーボードショートカット」において、「メニューのアクセスキーを有効にする」を外したり、ショートカットでかぶっているものを無効化しておかないと、Byobuに一部のキーイベントが渡りません。

F1：ヘルプの表示

F1は一般的にアプリケーションのヘルプに割り当てられているため、Byobuでは使用していません。その代わりShift-F1で新しいウィンドウにショートカットキーリストを表示します（**図7**）。

▼図7　ショートカットキーリスト

```
Byobu is a suite of enhancements to tmux, as a command line
tool providing live system status, dynamic window management,
and some convenient keybindings:

  F1                                * Used by X11 *
    Shift-F1                        Display this help
  F2                                Create a new window
    Shift-F2                        Create a horizontal split
    Ctrl-F2                         Create a vertical split
    Ctrl-Shift-F2                   Create a new session
  F3/F4                             Move focus among windows
    Alt-Left/Right                  Move focus among windows
    Alt-Up/Down                     Move focus among sessions
    Shift-Left/Right/Up/Down        Move focus among splits
    Shift-F3/F4                     Move focus among splits
    Ctrl-F3/F4                      Move a split
    Ctrl-Shift-F3/F4                Move a window
    Shift-Alt-Left/Right/Up/Down Resize a split
  F5                                Reload profile, refresh status
    Alt-F5                          Toggle UTF-8 support, refresh status
    Shift-F5                        Toggle through status lines
    Ctrl-F5                         Reconnect ssh/gpg/dbus sockets
    Ctrl-Shift-F5                   Change status bar's color randomly
  F6                                Detach session and then logout
    Shift-F6                        Detach session and do not logout
    Alt-F6                          Detach all clients but yourself
    Ctrl-F6                         Kill split in focus
  F7                                Enter scrollback history
    Alt-PageUp/PageDown             Enter and move through scrollback
    Shift-F7                        Save history to $BYOBU_RUN_DIR/printscreen
  F8                                Rename the current window
    Ctrl-F8                         Rename the current session
    Shift-F8                        Toggle through split arrangements
    Alt-Shift-F8                    Restore a split-pane layout
    Ctrl-Shift-F8                   Save the current split-pane layout
  F9                                Launch byobu-config window
    Ctrl-F9                         Enter command and run in all windows
    Shift-F9                        Enter command and run in all splits
    Alt-F9                          Toggle sending keyboard input to all splits
  F10                               * Used by X11 *
  F11                               * Used by X11 *
/usr/share/doc/byobu/help.tmux.txt
 u*  17.04 0:-  1:gihyo- 2:help*
```

F2：ウィンドウ・ペインの作成

F2キー系列は、ウィンドウやペイン、セッションの作成に関するショートカットです。

F2は新規ウィンドウを作成し、フォーカスを作成したウィンドウに移動します(**new-window**)。

Shift-F2は現在のペインを水平方向に(上下に)分割(**split-window -h**)します。Ctrl-F2だと垂直方向に(左右に)分割(**split-window -v**)です(**図8**)。

Ctrl-Shift-F2の場合は、新しいセッションを作成します(**new-session**)。

F3/F4：ウィンドウ・ペイン間の移動／サイズ変更

おそらく最も利用するショートカットが、このウィンドウ・ペイン間の操作に関連するキーでしょう。よく使うため、カーソルキーでも操作できるようになっています。

F3で逆方向にフォーカスウィンドウを移動(**previous-window**)します。ステータスの下に表示されるウィンドウタイトルリストを左方向に移動するキーです。F4は逆に、右方向にフォーカスウィンドウを移動します(**next-window**)。F3/F4はそれぞれAlt-←/Alt-→でも代用できます。

▼図8　水平方向と垂直方向に分割した例

▼図9　図8からCtrl-F3で移動したペインの状態

Shift-F3、Shift-F4は、1つのウィンドウに表示されているペイン間のフォーカス移動に使います(**select-pane**)。Shift-←、Shift-→、Shift-↑、Shift-↓を使うとより柔軟に移動できます。また、ペイン間の移動時は現在の場所を理解しやすいようにペイン番号を表示するようになっています(**display-panes**)。Alt-↑、Alt-↓ではセッションを順番に変更します(**switch-client**)。

Ctrl-F3、Ctrl-F4はペインそのものを移動します(**swap-pane**)。つまり同じウィンドウに表示されているペインの順番を入れ替えるのです(**図9**)。Ctrl-Shift-F3、Ctrl-Shift-F4はウィンドウの番号を入れ替えます(**swap-window**)。現在のウィンドウを特定の番号に変更したいときに便利でしょう。

Shift-Alt-←、Shift-Alt-→、Shift-Alt-↑、Shift-Alt-↓を使うと、フォーカスが当たっているペインのサイズを変更できます(**resize-pane**)。

F5：設定などの再読み込み

F5はプロファイルを再読み込みし、ステータスラインを更新します。ステータスラインが何かおかしくなったときに試してみるとよいでしょう。また設定メニューで変更した内容によっては、F5を押すように通知されることがあります。

Alt-F5はByobuのUTF-8モードのオン／オフを切り替えます。UTF-8モードがオンのときはプロンプト文字列の末尾が二重山括弧「≫(U+27EB)」になり、ステータスラインの左端にあるUbuntuロゴがUbuntuフォントの私用領域(PUA)に格納されているUbuntuのロゴマーク(U+E0FF)となります。Ubuntu以外の環境の場合、Debianなら「@」、CentOSなら「※」、Archなら「A」、Gentooなら「>」、Macなら Appleのロゴ(U+F8FF)などが表示されます。ステータスラインがずれたり、プロンプトの表示がおかしくなる場合はUTF-8モードを切ってみてください。

Shift-F5ではステータスラインの設定を切り替えていきます。切り替え可能な設定は、「~/.byobu/status」の「**tmux_right**」です。つまりコメントアウトされていない行をコメントアウトしたうえで、次のコメントアウトされている行を有効化しているだけです。普段は表示しないが一時的に表示するようにしたいステータスなどを設定しておくと便利です。

Ctrl-F5では**リスト1**の環境変数のリロードを行います。ssh-agentやgpg-agent、X転送などが有効な複数のホストからデタッチ／アタッチして接続するときに便利です。なおこの処理自体は**byobu-reconnect-sockets**を実行しているだけですので、手動で実行してもかまいません。

Ctrl-Shift-F5では「**byobu-select-profile -r**」を実行することで、ステータスバーのカラースキームをランダムに変更します。複数の異なるホストに接続しているとき、カラースキームを変えておけば、区別しやすくなるでしょう。

F6：セッションのデタッチ

F6系列は基本的にセッションから抜けるタイプのショートカットです。そこまで頻繁には使わないかもしれません。

まずF6キーを押すとセッションをデタッチしてログアウトします(**detach**)。Shift-F6キーだとデタッチのみを行いログアウトはしません。

▼リスト1　環境変数のリスト

```
DISPLAY DBUS_SESSION_BUS_ADDRESS SESSION_MANAGER GPG_AGENT_INFO
XDG_SESSION_COOKIE XDG_SESSION_PATH GNOME_KEYRING_CONTROL
GNOME_KEYRING_PID GPG_AGENT_INFO SSH_ASKPASS SSH_AUTH_SOCK
SSH_AGENT_PID WINDOWID UPSTART_JOB UPSTART_EVENTS
UPSTART_SESSION UPSTART_INSTANCE
```

Alt-F6キーでは自分以外のクライアントをデタッチします(**detach-client**)。複数のセッションを動かしているときに役立つでしょう。Ctrl-F6では現在のペインを破棄します(**kill-pane**)。

F7:コピーモードの操作

F7キーでコピーモードに入ります(**copy-mode**)。Alt-PageUpやAlt-PageDownはコピーモードに入りつつ、スクロールアップやスクロールダウンします。コピーモード中の操作については、第1章を参照してください。

Shift-F7キーは現在のペインを最大32,768行分ペーストバッファに取得し(**capture-pane -S -32768**)、ペーストバッファを「$BYOBU_RUN_DIR/printscreen」ファイルに保存し(**save-buffer**)、その内容を新しいウィンドウに環境変数EDITORで指定したコマンドで表示します。なおEDITORが空の場合は、ファイルの保存だけで終了します。

F8:ウィンドウの状態変更

F8キーはウィンドウの名前を変更します(**rename-window**)。名前を付けなかったときは実行中のプロセス名がウィンドウ名として使われます。Ctrl-F8にするとセッションの名前変更になります(**rename-セッション**)。

Shift-F8は現在表示しているペインのレイアウト変更です(**next-layout**)。ショートカットを入力するたびに次の順番で切り替わっていきます。

- even-horizontal:すべてのペインを水平方向に等間隔に並べる
- even-vertical:すべてのペインを垂直方向に等間隔に並べる
- main-horizontal:画面上半分を0番のペインとし、残りを画面下半分に水平方向に並べる
- main-vertical:画面左半分を0番のペインとし、残りを画面右半分に垂直方向に並べる
- tiled:すべてのペインをできるだけ等間隔にタイル上に並べる

Alt-Shift-F8とCtrl-Shift-F8は現在のウィンドウのレイアウトをリストア/保存します。内部的には**list-windows**の内容を保存する**byobu-layout**コマンドを実行しているだけです。保存先は「~/.byobu/layout/」となります。

F9:Byobuの設定やコマンドの実行

F9キーはByobuの設定メニューを開きます。これは**byobu-config**コマンドで表示されるものと同じです。

Ctrl-F9はすべてのウィンドウに対して同じ文字列を送り、Enterキーを押します。内部的には**command-prompt**で表示されたプロンプトの入力を、**list-windows**で表示されたウィンドウに対して**send-keys**で送り、最後にEnterを**send-keys**しています。よって「同時に実行する」というよりは「順番に実行する」感じです。Shift-F9はCtrl-F9のペイン版で、**list-windows**の代わりに**list-panes**を使っていること以外は同じです。たとえば複数のサーバに接続し、同時に「**sudo rm -rf --no-preserve-root**」を実行しなくてはならないときに便利でしょう。

Alt-F9はそのウィンドウに表示されているすべてのペインに同時に入力を行えるようにします。「**set-window-option synchronizepanes**」を実行して設定をトグルしているだけです。こちらは1文字ずつ送るのでほぼ同時となります。

F11:ウィンドウとペインのサイズ変更

F11キーはGNOMEアプリケーションのフルスクリーンに使われるため割り当てられていません。

Alt-F11は現在のペインを新規ウィンドウに移動します(**break-pane**)。それに対してShift-F11は現在のペインを「一時的に」現在のウィンドウで最大化します(**resize-pane -Z**)、もう

一度Shift-F11を押すと元に戻ります。

Ctrl-F11は現在のペインを「1つ前のウィンドウ」に統合します(**join-pane**)。「1つ前のウィンドウ」とはウィンドウ番号が小さい隣のウィンドウです。Ctrl-F11を何度も押せば、ペインがウィンドウの間を移動していく様子がわかるでしょう。ちなみに統合時は垂直方向に(左右に)分割して統合します。

F12：その他の設定

F12キーは初期設定におけるプレフィックスキーとして使われます。Shift-F12はByobuのキーバインディングの有効／無効切り替えです。ファンクションキーを別の用途に使いたい場合は、無効化してください。ただし、ここで紹介したファンクションキー以外のショートカットも無効化されます。

Alt-F12はtmuxのマウスサポートのオン／オフを切り替えます。初期状態ではオフになっています。マウスサポートをオンにすると、ウィンドウ名のクリックでウィンドウの切り替えや、ペインの領域クリックによるカレントペインの切り替え、ペインの枠のドラッグでサイズ変更ができるようになります。ssh越しでByobuを実行していても、リモートのByobuにマウスイベントが送られるようです。

Ctrl-Shift-F12は「**Mondrian**」が呼び出されます。言葉で説明するのは無粋ですので、具体的に自分の目でたしかめてください[注4]。

注4) 図形の意味がわからない場合は、Wikipediaで「ピエトモンドリアン」を参照。

Byobuの起動と設定ファイル

Byobuに独自のtmux設定を追加したい場合は、「~/.byobu/.tmux.conf」を編集してください。内容は「~/.tmux.conf」と同じで、Byobu起動時もしくはF5キー実行時に再読み込みされます。

Byobu起動時にあらかじめ特定のウィンドウを表示させたい場合は「~/.byobu/windows.tmux」にtmuxのコマンドを記述します(**リスト2**)。

また、「~/.byobu/windows.tmux.名前」のように異なる名前のファイルをいくつか用意しておいて、起動時に次のように選択することもできます。

```
$ BYOBU_WINDOWS=名前 byobu
```

ちなみに**byobu**コマンド実行時の引数はそのままtmuxに渡されます。このとき上記のようなファイルによるウィンドウ設定は反映されないので注意してください。

Byobuの設定ファイルやスクリプトは「/usr/share/byobu」や「/usr/lib/byobu」にまとまっています。ユーザ定義の設定は「~/.byobu/」です。これらのファイルの中身を見ておけば、おおよそByobuの動作がつかめるでしょう。

◆ ◆ ◆

このようにByobuは最初からそれなりにセットアップされた状態でtmuxを起動します。設定ファイルを編集することなくターミナルマルチプレクサをそれなりに便利に使いたい人にお勧めです。SD

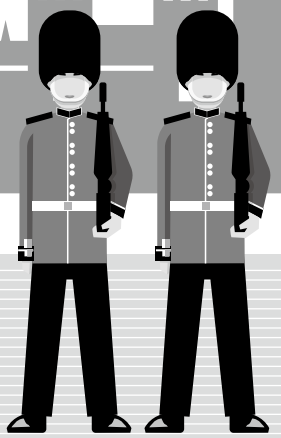
▼リスト2　~/.byobu/windows.tmuxの内容

```
new-session -n office -s eaglejump ;
new-window -n aoba zoi --jsonFile gan.json --config baru.js ;
new-window -n hifumi irssi ;
new-window -n yun at teatime ;
new-window -n hajime vim motion.xml ;
new-window -n rin ssh ko.local ;
split-window ;
split-window ;
select-layout tiled ;
select-pane -t 0 ;
pipe-pane -o 'cat >>~/ko.log' ;
```

不正アクセス・なりすましを防ぐ

認証を支える技術

Author
鈴木 賢剛(すずき ただよし)
F5ネットワークスジャパン㈱



BASIC認証からシングルサインオンまで

自社システムやWebアプリへのログイン時、多くの場合IDとパスワードが要求されることでしょう。ユーザにとっては煩わしくもあるこの「認証」ですが、不正アクセスやなりすましといった攻撃を防ぐために、なくてはならないしくみです。本記事では、基本的な認証技術とそれに対する脅威、一度の認証で複数個所へのアクセスを可能にする「シングルサインオン」、それらを利用した現場のソリューションを紹介します。

Webアプリケーション(以下Webアプリ)の利用が当たり前になった現在、電子商取引や金融取引だけでなく、SNSやオンラインゲーム、各種店舗のポイントアプリなどさまざまなシーンで「認証」――本人からのアクセスであることを正しく確認すること――が使われています。読者のみなさんも、少なくとも数個以上のWebサイトに対して複数の資格情報(ID、パスワードの組など)を持ち、1日に(実際の入力回数はより少ないとしても)数十回以上は、「認証」をしてからメールの確認や電子商取引、金融取引などを行っているでしょう。

本人であることを確認する方法としてこれまでもっともよく使われてきたユーザ名とパスワードの組だけ、つまり一要素認証は、そのユーザ名とパスワードを他人が知り、本人になりすまして利用できるようになることから、昨今のセキュリティ的には不十分になってきています。

本記事では、まずWebアプリにおける現存の認証技術やそれに対する脅威についておさらいをします。その後、新しい認証方式を追加してWebアプリをよりセキュアに利用できるようにする方法と、複数のWebアプリに対して1回のログインで利用できるようにするシングルサインオン技術の現在、そしてこれからの認証のあるべき姿について紹介していきます。

Webアプリにおける認証・認可

まずは各認証方式について、基本的なものから1つずつ紹介していきたいと思います。

BASIC認証

アクセスの際、ユーザ名とパスワードの組をBase64エンコードして使用する認証方式です。基本的な設定例として、Apacheの.htaccessあるいはhttpd.confに設定を追加する方法があります。いずれも、「Apache　BASIC認証　設定」などのキーワードで検索することで設定方法の書かれたWebサイトはたくさん見つけられますので、具体的手法については割愛します。

問題点

この方式では特定のディレクトリに対して認証の設定ができるため、手軽に認証サイトを構築できる反面、セキュリティ面ではいくつかの問題が出てきます。まずBASIC認証はしくみ上、Authorization:ヘッダに「ユーザ名:パスワード」の文字列をエンコードしたものを付加してアクセスするため、Webサイトの通信方式がHTTPSではなくHTTPの場合、通信経路上で暗号化されていない資格情報が、

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

のように流れ、第三者に「ユーザ名:パスワード」の組を容易に知られてしまう問題があります。HTTPSを使用することで暗号化されますが、最近ではSSLのさまざまな脆弱性の問題やハッシュの衝突などの問題から、使用する暗号スイートも制限するTLS1.2以降の使用が強く推奨されています。HTTPSが使えない場合、MD5を使ってユーザとパスワードの組をハッシュ化する「Digest認証」を使用するのも1つの方法です。

Webサーバに第三者がアクセス可能な場合、新たな課題が発生します。多くの場合、.htaccessファイルにはユーザ名とパスワードの組を設定したファイル(よく使われるものとして.htpasswdなど)のパスが平文で書かれており、この.htpasswdファイルにユーザ名とパスワードのハッシュ情報が書かれています。直接パスワードが知られる心配はあまりありませんが、このファイルにアクセス可能な第三者がいたり、悪意あるCGIプログラムなどがアクセス権を獲得した場合は危険です。パスワードを奪うのではなく、新たに認証可能なユーザを追加したり、またはなりすましたいユーザの資格情報の行を削除し、パスワードを変更して作りなおしてしまえば、悪意ある第三者により設定された新しいパスワードで利用できてしまうのです。

ID管理の面から考えると、この方式では資格情報がApacheに利用される形式のものに限定されるため、少ないユーザ名とパスワードの組を多くの人に知らせる場合などには便利ですが、数十、数百ユーザを登録して管理するには、あまりにもめんどくさいという課題も出てきます。

そのほかの実装

BASIC認証に類する実装を紹介します。Apache以外のWebサーバソフト、たとえばWindows Serverで動くIISであれば、Active Directoryユーザに対してBASIC認証と同じ「基本認証」のほか「NTLMv1」「NTLMv2」、そしてKerberos認証[注1]と同じ「統合Windows認証」を選択できます。もちろんApacheでも、LDAPサーバに問い合わせて認証する設定ができます。macOSのServer.appでも、内部的にはOpenLDAPやApacheが使用されていますが、GUIで直感的に、ディレクトリユーザによる認証を行うWebサイトの設定ができます。

HTTPフォーム認証

ユーザ名とパスワード(のハッシュ値)を独自の設定ファイルやデータベース、ディレクトリサーバ上に持ち、通常のHTML FORMタグで指定されたURLに対してHTTP POSTされた資格情報を、Webアプリ側で受け取って認証する方法です。この方式の場合、暗号化されないHTTPでPOSTしてしまうと、やはり資格情報が通信経路上で容易に盗聴されるため、HTTPSを使用したサイトで使われます。この方式の場合、通信経路はSSLで暗号化されていても、Webアプリ側では生のユーザ名とパスワードを受け取り、それを独自のデータベースや設定ファイルを参照して正しいユーザかどうかを確認するため、多要素認証・多段階認証などへの拡張も自由に行えるメリットがあります。

最近ではWeb2.0の流れで、リッチな演出を伴った形で認証画面を出すものもありますが、HTTPフォーム自体を動的に表示しているもののほかに、JavaScriptを使った独自の実装で認証するなど多様化してきています。

HTTPヘッダーベース認証

Webアプリにとっては認証方式やユーザのパスワード、指紋や顔などは重要度が低く、「誰がアクセスしてきたか」の正確な情報がわかれば良いでしょう。そのため、リバースプロキシや

注1) RFC 4120などで規定されるネットワークでの認証を行うプロトコルで、一度認証を通ればその後はチケットのやりとりを行うことで認可し、パスワードを使用せずに安全にWebアプリを利用できるシングルサインオンのしくみを提供できる。

SSLアクセラレータなどで正しく認証したあとに、HTTPヘッダの拡張ヘッダ（Xヘッダ）にユーザ名を、

```
X-UserName: taro
```

のように付与してWebアプリに渡すと決めておくことで、Webアプリは誰がアクセスしてきたのかを環境変数として取得できます。Webアプリは認証をせず、ユーザ名や組織名など、認可に関連する付加情報があらかじめHTTPヘッダに入ってくる前提でユーザを判別する方法です。

クライアント証明書認証

HTTPS通信のSSLネゴシエーション中にクライアント証明書の提出を求め、あらかじめ設定されたクライアントルート証明書により発行された証明書でない場合に、SSL接続確立を拒否する方式です。クライアント証明書はpkcs12形式のファイルとして存在することもあれば、USBドングル（またはトークン）やICカードなどの物理デバイスになっている場合もあります。

クライアント証明書認証を利用する場合、ユーザの初回利用開始時にクライアント証明書を発行し、利用できるようにする処理が必要になります。ユーザがクライアント証明書の入ったデバイスを紛失したときや、異動や退職などによってクライアント証明書を無効化する必要があるときに、該当クライアント証明書を失効させる処理も必要となります。そして、紛失時に再度利用可能にするための手続きとしてもクライアント証明書を再発行する必要があるほか、クライアント証明書自身にも有効期限が設定されているため、定期的な再発行手続きが必要です。

とくにクライアント証明書をファイルでコピー・エクスポート可能な形で配布してしまうと、そのファイルを無制限にデバイスに設定可能となり、その時点でセキュリティはなくなってしまうので、厳格な運用も併せて必要になります。クライアント証明書自体は、openssl コマンドさえ使えれば誰でも無料で発行できるものですが、実際にはクライアント証明書のインストールは社内のIT部門で直接実施するようにする、などの適正な運用も必要になります。こうした手間を軽減させる目的で、クライアント証明書の適正な運用ができる、㈱JCCHのGléas[注2]などの商用製品やサービスを利用するのも1つの方法です。

クライアント証明書認証は、SSL接続確立時に認証するしくみから、BASIC認証やHTTPフォーム認証との併用が容易にできるため、「クライアント証明書＋ID／パスワード」の二要素認証も広く使われています。クライアント証明書認証は、適切な運用がされていれば不正アクセスをほぼ完全に防ぐことができますが、SSL可視化の名のもとにSSLを終端するタイプのProxyサーバ経由が必須となる最近の環境では、利用できなくなるという課題も出てきています。

ワンタイムパスワード（OTP）

認証のたびに変わるパスワードを用いる方式です。6桁程度のランダムな数字が表示される物理デバイスやスマートフォン用のアプリを使用して、ID／パスワードとは別のランダムな数字を認証に使用するものや、あらかじめ配布してある表の行と列を指定してその位置の文字を入力させるもの、SMSで送られてきた6桁程度の数字を入力するものなどがあります。

ログインの際にほかのデバイスをいちいち取り出して見なくても済む優れた方法として、ブラウザのログイン画面に表示される表の中のランダムな数字の位置の順序をあらかじめ覚えておく、パスロジ㈱[注3]のパスロジック方式などもあります（**図1**）。

乱数を表示する物理デバイスを配布する方式の場合、クライアント証明書よりもさらに、再発行の手続きが煩わしくなります。液晶に数値が表示されるデバイスはいつかは電池が切れるほか、壊してしまうこともあり、安くはないデ

注2） https://www.gleas.jp
注3） https://www.passlogy.com

バイスの再配布手続きが嫌われがちです。そのため、ワンタイムパスワードの世界ではパスロジック方式が広く使われてきています。

利用環境情報・デバイス固有情報

専用アプリケーションやブラウザの拡張機能（プラグインやWebExtension）などを使って、ユーザの利用環境（IPアドレスやUser-Agentなどから複合的に判断）やデバイス固有情報を用いて利用デバイスを限定しておき、普段の利用環境と明らかに異なる場合は利用できなくしたり、追加の認証を利用したりするなどの二段階認証や、あらかじめ利用を許可するように設定したデバイス以外からの利用はできないようにするデバイス制限などの方法があります。

macOSやWindowsでは、前述のクライアント証明書のほかに、マシン固有で持たせられるマシン証明書を利用する方法もあります。

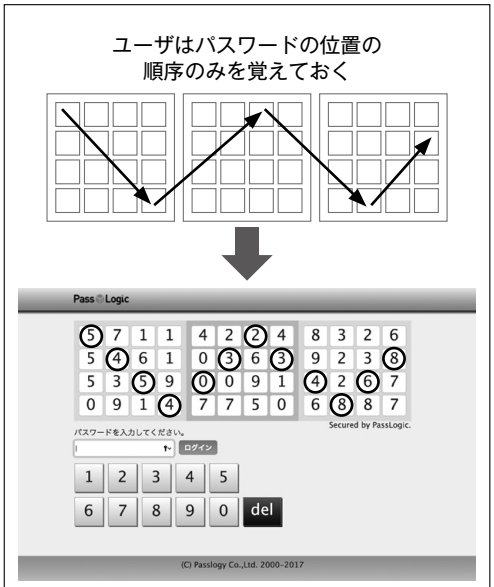
生体認証

Webアプリでの利用シーンは多くはありませんが、指紋、網膜、声紋、顔、掌形（手のひらの形）、静脈など人の生物学的特徴に基づいて認証するしくみです。通常のパソコンにはそれらの認証デバイスが標準で搭載されているわけではなく、現在はあまり使われていませんが、iPhoneやMacBook ProのTouch IDなど、指紋認証機能、Webカメラを搭載した機器も増えてきているため、将来的には指紋認証や顔による認証は広く使われていくと考えられます。

ソーシャルログイン

Webアプリ自体には認証のしくみを持たせず、外部のWebアプリに認証または認可をゆだねる方式として近年使われてきています。アカウント登録をしようとすると、FacebookやGoogleなどのアカウントでログインするよう促されたことがあるかと思います。このように、外部のSNSなどのアカウントを利用してログインまたはアカウント登録をすることで、個人情報入力の手間が省ける場合もあります。この方式としてOAuth 2.0やOpenID Connectというしくみがよく使われています。

▼図1　パスロジック方式の認証（スクリーンショットは実際の「PassLogic」の製品画面）



毎回ランダムな数列が表示されるので、
その位置の数字をパスワードとして入力する

シングルサインオン

ここまでで紹介した認証方式を使用していくうえで、たくさんの認証を要するWebアプリが乱立してくると今度は、「WebアプリA」でログインしたら同一の資格情報を利用する「WebアプリB」もログインせずに使いたい、というシングルサインオン（以下SSO）への要求が出てきます。SSOを実現する方式として、代表的なものとして大きく次の4つの方式が挙げられます。

SSL-VPN型

トンネルネットワークを張って、VPNと同じように、外部の端末が社内にいるのと同じように社内のリソースを利用できるようにする方式です。社外からの接続時にSSLによる暗号化と認証を必要とすることで、SSL-VPN接続確立後は、ドメインに参加している端末はシームレ

スに利用できるようになります。

SSL-VPN製品の代表的なものとしてBIG-IP APM、Cisco ASAなどが挙げられます。

HTML書き換え型

HTTPSによるリバースプロキシで認証してHTMLコンテンツを書き換え、イントラネットのWebアプリを外部から利用できるようにする方式です。独自の認証画面を持ち、認証されたら配下のWebアプリ群を利用できるようにします。SSL-VPN製品にはこの機能も持つものも多いため、この機能もSSL-VPNと言われることがあります。

代表的な製品としてHP IceWall SSO、BIG-IP APM、Juniper SA(現Pulse Secure)が挙げられます。

Webサーバエージェント型

ApacheやIISなどが動作するサーバ上でエージェントとして動作し、Webアプリへ情報を渡すときに、資格情報をWebサーバ側で渡すことでSSOを実現する方式です。代表的なものとして、CA Federation(旧SiteMinder)などが挙げられます。

SAML 2.0

ここまで紹介した方式は、それぞれの製品ごとの独自実装で実現されているため、互換性が保てない問題がありました。そこで互換性の問題を解決し、複数Webアプリへのセキュアなシングルサインオンを実現するしくみとして、SAML(Security Assertion Markup Language)が登場しました。執筆現在の最新の仕様は「2.0」で、現在ではSAMLというとほぼSAML2.0を指します。

SAMLにはSAML IdP、SAML SPの2つの構成要素があります。IdPとはIdentity Providerの略で、認証を受け持つWebアプリを指します。そしてSPはService Providerの略で、実際に利用するWebアプリを指します。SPとIdPはお互いにPKI(公開鍵基盤)のしくみで信頼関係を結び、SPに認証情報のない状態でアクセスすると自動的にIdPにリダイレクトして認証を通し、ふたたびSPにリダイレクトしてWebアプリを利用できるしくみを実現します。通常は、「IdP」1つにひも付く「SP」が複数あり、どれか1つのSPにアクセスしてIdPで認証されれば、ほかのSPもそのまま利用できるようになるのが特長です。

企業向け市場では、たとえばGoogle AppsやOffice365の認証など、クラウド環境のアプリケーションの認証に対して、自社の持つLDAPやActive Directory(以下AD)などの認証基盤をSAMLで利用できるように拡張し、IdPとして認証させる方式も増えてきています。

SAML自体はRFCでも標準化された一般的なものであるため、BIG-IP APM、HP IceWall SSOのオプションとして利用できるIWFA(HP IceWall Federation Agent)連携モジュールやADFS(Active Directoryフェデレーションサービス)など、さまざまな製品やクラウドサービスで利用できます。しかしSAML 2.0対応を謳(うた)う製品でも、一度のログアウトですべてのアプリから同時にログアウトして次回利用時には認証を必要とさせるシングルログアウト(SLO)に対応していない製品が意外と多いので注意が必要です。また、SLOはSAML連携したときにIdP、SP双方が対応していないと実現できません。

Webアプリにおける認可

ここまで認証について見てきましたが、さまざまな方式で認証を通過したあとで、利用できるWebアプリ、機能をそのユーザの属性情報に基づいてアクセス許可・拒否する「認可」のしくみも追加で必要となるでしょう。たとえば、/admin/は管理者専用の管理ページですので、認証を通ったユーザでも、管理者でなければ利用させたくないという場合です。また企業では、人事情報に関するWebアプリは人事部以外の人

にはアクセスできなくする必要があったりします。逆にプログラムコードや営業機密にアクセスする場合、「特定プロジェクトに参加しているメンバーだけや営業メンバーだけ」のように認証したあと、そのユーザの属性情報も見たうえで厳密にアクセス可否を判別する必要があります。

こうした属性情報は、LDAPやADなどのディレクトリサーバやRADIUSサーバでは、ユーザ情報とともにサーバ上に格納されているため、認証時または認証後にクエリを投げることで取得できます。組織的なグループ・複数組織に属している場合、組織単位ではなく役職単位やプロジェクト単位で制限するなど、さまざまな要素に基づいて認可の処理を行うのが一般的です。

また認可に付随する話題として、ユーザにとってわかりやすいよう、ログインユーザ名情報だけでなく利用者氏名や会社名、所属名などを表示する目的で、ログインユーザ名からデータベースまたはディレクトリを検索してブラウザに表示させるケースがあります。そのほか、HTTPの拡張ヘッダ(Xヘッダ)に付随情報を持たせて受け取り、それに基づいて認可判断を行うといったケースもあります。

認証を要するサイトにおける脅威

認証の行われるWebサイトのうち、とくに金銭と関わるようなものは、それを狙う攻撃に日常的にさらされます。オンラインバンキングサービスや電子商取引だけではありません。ゲーム内のアイテムをオークションなどを使用して現金で取引するリアルマネートレードを目的に、オンラインゲームなど一見お金に結び付くようには見えないものも狙われます。ここでは、それらの脅威について見てみましょう。

ユーザ自身

脅威としてまず一番に挙げられるのは攻撃者やシステムではなく、その利用者自身です。

せっかく本人であることを確認するしくみがあってパスワードを設定できても、利用者がパスワードとして、ログインユーザ名と同じ文字列や、「123456」「qwerty」「abcdef」といった第三者からも容易に想像がつく文字列を使用した場合、簡単に第三者によるなりすましに遭ってしまうでしょう。

これは技術の問題というよりは、認証の重要性を理解せず、認証をただの「利用するための手続き」程度にしか考えていないことに起因するリテラシーの問題です。すべての人は完璧なリテラシー教育を完了したあとに初めてインターネットが利用できるようにする、というのはもちろん不可能で、使い方は使いながら覚えることになっている現状からすると、これは永遠に解決しない問題と言えます。

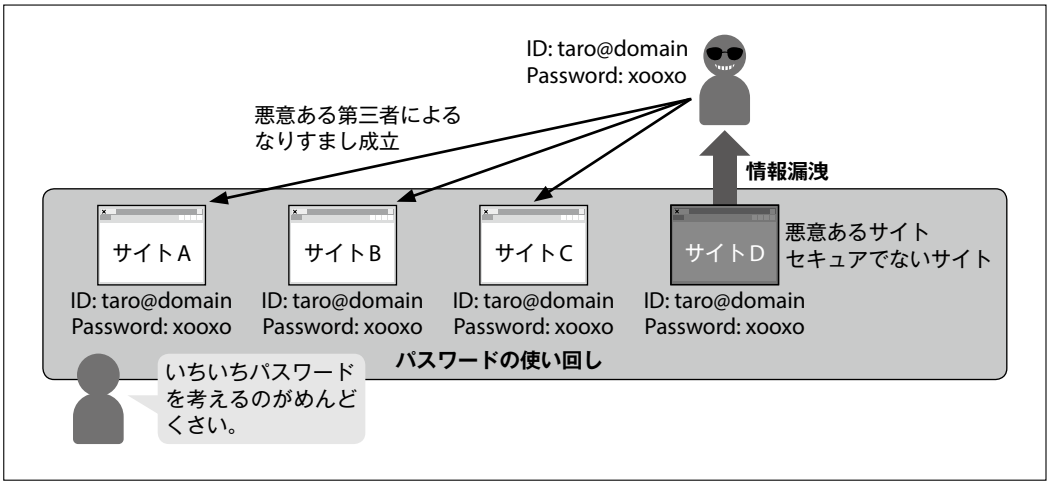
ただ、容易に類推されにくい複雑なパスワードを使用していても、すべてのサイトで同じパスワードを使用していた場合は重大な脅威となります。どこかのサイトで情報漏洩したID、パスワードの組をほかのサイトでも利用できるのであれば、攻撃者にとってはこの資格情報は非常に有用なものとなります(**図2**)。

辞書攻撃・総当たり攻撃

これはユーザ名のパスワードを獲得するために行われるもので、ユーザ名とよく使われるパスワードの辞書や、あらゆる文字列の組み合わせを総当たりで試します。総当たり攻撃はブルートフォース攻撃とも呼ばれます。

対策としては、認証失敗時に意図的にN秒以上待たせて時間を消費させることで攻撃を緩和し、攻撃成功率を低くする方法や、特定IPアドレスからX回認証に失敗したら、そのIPアドレスからは24時間接続できなくするなどの方法があります。しかし、基本的にログインという行為自体は悪いことではないため、たとえば特定IPアドレスをブロックしてしまうことにより、たまたま同じIPアドレスを利用する正規のユーザが利用できなくなる可能性があるなど、メリットとデメリットの問題は尽きません。

▼図2　パスワード使い回しの脅威



BASIC認証やHTTPフォーム認証では、HTTP接続リクエストのときに資格情報を含めるため、機械的に効率良く辞書攻撃を行うのに向いています。高度なものになると、認証成功時には0.3秒以内にレスポンスがあるとわかっていれば、0.3秒以上は待たずに次の資格情報を試すなど効率化したうえに、IPアドレスも数秒ごとに変更していくといったプログラムもあります。これらは今も世界中の悪意あるクライアントで動作し、資格情報を狙っています。

リスト型攻撃

漏洩した、意図的に入手したID・パスワードを使った攻撃です。Webアプリにおける認証は、先ほども述べたとおり、ユーザ本人であることを証明する重要な行為であるのにもかかわらず、必要な手続き程度の認識で利用されています。複数のサイトで同一のユーザ名とパスワードが使用される「パスワードの使い回し」や、「123456」「qwerty」「abcdef」など攻撃者が最初に試すような脆弱なパスワードが利用されていることで、リスト型攻撃の餌食（えじき）になっています。

たとえば、Yahoo!、楽天、Amazon、Google、Facebookなどでは、メールアドレスをユーザ名としています。これらすべてのサイトで同一のパスワードが使用されていると、悪意をもって作られた偽のログイン画面で正しいパスワードを入力してしまったり、どこか別のWebサイトのアクシデントでたまたまユーザ名とパスワードの組が漏洩してしまったりすれば、悪意ある第三者によってその資格情報を使用され、すべてのサービスを利用されてしまうという危険な状態になります。

近年のリスト型攻撃は、ブルートフォース攻撃として認識されるほどにはアクセス頻度は高くないのに、ヒット率（資格情報の正解率）が異様に高いという特徴を持ち、まさにパスワードの使い回しが多くなされていることが証明されていると言えるでしょう。

マルウェア感染

おもにオンラインバンキングサービスを狙う攻撃で使われる方式です。マルウェアは何らかの脆弱性を悪用して端末上に感染し、ユーザが正しい手順でオンラインバンキングのサイトにログインするのを待ち続けます。ログインしたあとで、その正規のログインセッションを利用して勝手に第三者（犯罪者）への振り込みを行うなど、ユーザに気づかれずに金銭を奪い取ります。マウスの移動距離やキーボードのイベントをJavaScriptで判別し、本当に人がキーボードを叩いて資格情報を入力して、マウスを移動さ

せてボタンを押したかをチェックするなど、複雑な工夫がなされています。

この分野では、マルウェア作成者側とオンラインバンキングサービス側でいたちごっこの対策が続いており、1ヵ月に何度もWebアプリが変更されることもあります。

Webブラウザ・Webサーバ

何か秘密のコンテンツを渡すときに、秘密のURLを当事者だけで共有する方法を取る場合があります。この場合、ApacheなどのWebサーバ側のログにはアクセスがあった時点で記録されるので、Webサーバの管理者にもそのURLの存在が知られてしまいます。

ここまではWebアクセスのしくみを理解していれば当たり前のことです。しかし実は、ユーザがアクセスしたURLがユーザに無断でブラウザの開発元に通知されていることがあります。筆者の経験では、知人にしか知らせていなかった類推しにくい長い文字列を付加したURLが、知人のアクセスから数時間後に、Webブラウザ開発元のIPアドレスからのアクセスにも含まれていたことがありました。開発元に確認したところ、機能としてそうした(スパイウェア的な)行為を行っていると悪びれずに認めました。しかもこの機能はOSに統合されているため、アンチウイルス製品などのセキュリティ製品でも駆除できません。

このような例からわかるとおり、秘密のURLの共有はセキュリティ上まったく意味をなさないので、きちんと認証を行う必要があります。

脅威に対する有効な対策

これら脅威に対する、ユーザ側、Webアプリ開発でできるいくつかの対策方法を述べたいと思います。

ユーザ側でできる対策

ユーザのリテラシーの問題については、パスワードポリシーとして「大文字、小文字、数字、記号を含み、同一文字を3回以上使用せず、合計9文字以上の文字列を使用すること」というような、システム側では容易に類推されにくい複雑なパスワードの利用を強制する策があります。こういったルールではどうしても、入力ミスが増え、タイプミスなのに「正しいパスワードを入力したのにログインできない」という言いがかりや水掛け論的なトラブルに見舞われ、その都度意味もなくパスワードの初期化をする手間がかかる場合もあります。それでも、なりすましされるよりはましです。

▼図3　Safariの、サイトごとにパスワードを設定支援する機能の例

リスト型攻撃による乗っ取りは、Webアプリごとに異なるパスワードを使用し、重要度に応じて複雑なパスワードを使用するようにすることで防げます。たとえばmacOSのSafariでは、Webアプリごとに複雑なパスワードが提案され、それをキーチェーンアクセス(鍵束)のようにセキュアに保護された領域で持っておき、利用できる機能があります(**図3**)。これにより、Webアプリごとに異なるパスワードを使用していても覚えられなくなるという心配は減らせます。Firefoxにも同様の機能があり、パスワードの提案はありませんが、複雑なパスワードを記憶させてマスタパスワードで保護できます。

また信頼できないOS、ブラウザ、プラグイン、Webサイトの利用をきちんと意識して避けることも重要な選択になってきます。たとえばGoogleのスパイウェア行為をいっさい排除したことを謳(うた)うUngoogled Chromiumというブラウ

ザもオープンソースで出ていますので、こうしたブラウザを利用するのも良いでしょう。

Webアプリ開発側でできる対策

辞書攻撃も総当たり攻撃もリスト型攻撃も、狙うのは一要素認証を行うサイトです。

HTTPフォーム認証への攻撃には、「特定のURL」の「特定のFORMタグ」の「特定のParameter」として「特定のURL」にHTTP POSTされるしくみが利用されます。ですので、アクセスされるごとに認証に必要なパラメータやURLが動的に変化するような複雑な方式や、パスロジック方式などワンタイムパスワードによる認証を行うのも一定の効果が期待できますが、一要素認証である時点で効果は限定的です。

そのためHTTPフォーム認証といった従来の認証に加えて、クライアント証明書がないと認証画面にすらたどり着けないクライアント証明書認証の利用や、デバイス固有情報と人をひも付けて「人が」「その人のデバイスを持っている」ことで認証するのが効果的な対策です。これはSSOでも同様で、たとえばIdPで認証を行うときに一要素認証のみだと、資格情報が盗まれた時点で悪意ある他人にすべてのアプリケーションを利用されてしまいます。これらの点から考えて、二要素認証は重要です。

これからの認証

ここからは、最近の認証の現場でよく聞く課題と、それを解決するアプライアンス製品やクラウドサービスを組み合わせたソリューションを紹介します。

認証・認可のファイアウォール

従来、認証は個々のWebアプリで完結し、一要素認証で公開されるといった形が一般的でした。それからさまざまな方式のSSO製品が登場し、それぞれ独自の方式でSSOを実現していましたが、非常に高額だったり、近年のWeb 2.0と呼ばれるHTML5やCSS3、Ajaxといった技術を駆使したWebアプリでは利用できなかったりと、相性の問題も出てきました。

後継製品の不在

たとえば、社内のSharePointアプリをThreat Management Gateway（TMG）やForefront Unified Access Gateway（UAG）を使用して公開していた企業からは、現在TMGとUAGは販売が終了し、そして後継製品がないことから、今後の移行に困っているケースをよく聞きます。とくに社内Webアプリをインターネットに公開する場合、認証・認可されていないトラフィックを社内またはDMZ[注4]へは通したくないというニーズがあります。トラフィックの制御を行う製品としてはファイアウォールが一般的ですが、通常のネットワークファイアウォールはIPアドレスとポートの開け閉めしかできず、HTTPSトラフィックのうち認証されたものだけを通す、ということはできません。

そこで、リバースプロキシとして代理認証を行い、認証・認可されたユーザにのみ必要なリソースへのアクセスを認めるアクセス管理製品「認証・認可のファイアウォール」が必要となってきました。

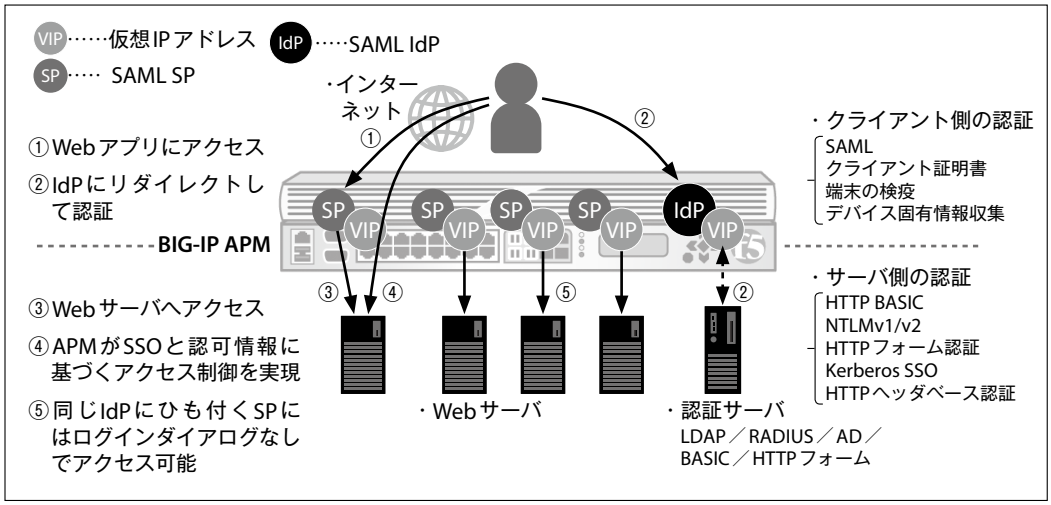
柔軟・多機能なBIG-IP APM

F5ネットワークス社のBIG-IP APM（Access Policy Manager）はSSL-VPN製品として利用されることが多いですが、認証・認可のファイアウォールとしても機能する製品です。

クライアント側に対してはiOSやAndroidなど多くのデバイスに対応し、ドメインに参加している端末か、端末にアンチウイルス製品が動作しているか、ファイアウォール製品が動作しているか、P2Pソフトが動作しているかなどさまざまなセキュリティチェック（検疫）をかけることができます。会社で配布したPCと自宅PC

注4）DeMilitarized Zone。外部のネットワークと内部のネットワークの中間に配置するネットワーク領域。

▼図4　BIG-IP APMによる認証・認可のファイアウォール



を区別して、必要なセキュリティ要件を端末ごとに変える設定も可能です。とくにバージョン「v13」以降では、FirefoxやChrome、Microsoft Edgeなど従来のプラグインが利用できなくなったWebブラウザでも、URIスキームに似た方式でヘルパーアプリケーションを利用することで端末の検疫チェックが可能です。

サーバ側についても、従来のWebアプリがHTTP BASIC認証やNTLMv1/v2認証をしている場合でも、ユーザがAPMに入力した資格情報を使用して、自動的にWebサーバに認証情報を渡すクレデンシャルキャッシングのしくみを持っているため、ユーザは都度ログインダイアログに資格情報を入力する必要がなくなります(**図4**)。またAPMはWebコンテンツを書き換えず(書き換える設定もありますが)、宛先のIP・ポートのみを書き換えてWebアプリに渡しますので、Webアプリのコンテンツとの相性問題もほとんど発生しなくなり、Web2.0アプリケーションも快適に利用できます。

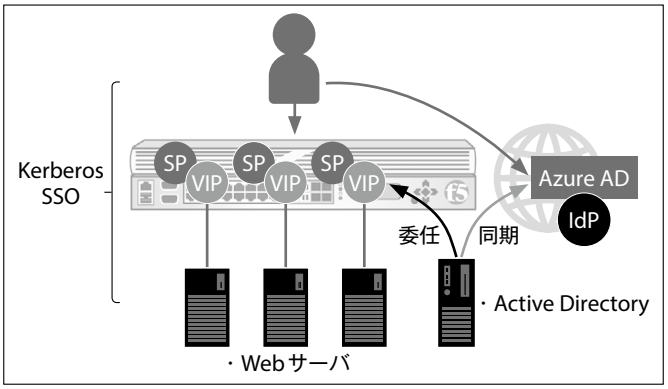
クラウドサービスとオンプレミスでの認証

最近では、ID管理にAzure ADなどの外部のクラウドサービスを利用する例も増えてきました。これによって、アクセスしてくるユーザに応じて、認証サーバを自社ADにするかAzure ADにするかを変えながら透過的に利用できるようにもなりました。ただこの場合でよく問題になるのが、「既存のWebアプリは変えたくない」と「SAMLで認証してWebアプリにSSOしたい」の両立です。

SAMLの属性情報を使ってパスワードをSP側に渡すようにすることで、NTLMv1/v2やHTTP BASIC認証へのSSOは可能ですが、Azure ADをIdPとした場合、Attributeによるパスワードの受け渡しは困難です。できたとしても、自社ADのユーザ名とSAMLのnameID(SAMLで認証したときのユーザ名)とのマッピングも必要になります。

認証方式のみ変更した場合でも、Webアプリから見た環境変数のユーザ名は変わりません。ですので、既存のWebアプリの認証方式の設定をKerberos(統合Windows認証)に変更し、AD上に委任アカウントを作成してAPMに登録、Azure ADでUPN形式のユーザID、たとえば「taro@domain.net」で認証を通れば、自社ADの「taro」としてアクセスできるKerberos SSOの設定をすることで、クラウド上のWebアプリとオンプレミス環境のWebアプリを、どこからで

▼図5　SAMLのようにパスワードがなくてもSSO可能



も、どのデバイスからでもシームレスに利用できるような環境を実現できます(**図5**)。

BIG-IP APMはハードウェアアプライアンスのほかESXiで動くVE(Virtual Edition)、Amazon Web Services、Azure ADなどのパブリッククラウドでも利用できるため、オンプレミス環境、自社プライベートクラウド、パブリッククラウドなど、適材適所で配置して認証・認可のコントロールを厳密にかけることができます。

デバイス固有情報とワンタイムパスワード

二要素認証として、デバイス固有情報を利用しようとした場合、対象となるユーザの全デバイスの固有情報を収集して登録する手間がどうしても問題になります。

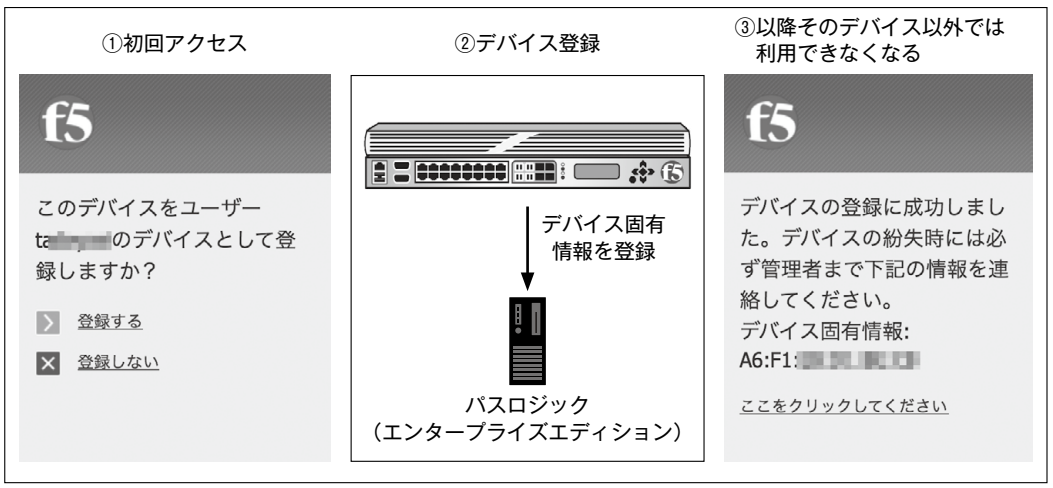
パスロジック(エンタープライズエディション)とAPMを併用することで、初回アクセス時にユーザの利用デバイスを自動的に登録でき、管理者があらかじめすべてのデバイスの情報を収集・登録する必要がなくなります。この方法では、仮にSSL可視化の名のもとにSSL通信がProxyで終端されてクライアント証明書認証ができなかったり、通信経路でSSLがほどかれてパスワードが漏れたりしても、毎回パスワードが変わるため非常に強力です。加えて、登録したデバイス以外からは利用できなくする、人(パスロジック方式)と物(デバイス固有情報)の二要素認証を簡単に実現できます(**図6**)。

認証と運用負荷

認証を行うシステムの構築・運用において課題となってくるのはその運用負荷です。

認証を必要とするシステムを運用開始すると、「サイトにアクセスできない」「ログインできない」「ログイン後ページが表示されない」ことすらもほとんど区別されず、一緒くたに「あれにつ

▼図6　デバイス固有情報の自動登録(参考：https://devcentral.f5.com/articles/big-i)



ながらない」というような形で、利用者からの苦情が上がってきます。こうしたときに速やかに解決するためにも、認証のしくみと同じくらい重要になってくるのが「アクセスログ」です。

▼リスト1　資格情報なしでの初回アクセス時のログ

```
106.133.41.129 n-bahn.club:443 - [13/Jul/2017:14:12:05 +0900] "GET / HTTP/1.1" 401 6129 "-" ...(略)...
```

▼リスト2　資格情報を付与したアクセス時のログ

```
106.133.41.129 n-bahn.club:443 taro [13/Jul/2017:14:12:18 +0900] "GET / HTTP/1.1" 200 10942 "-" ...(略)...
```

▼リスト3　認証に失敗したときのログ

```
106.133.41.129 n-bahn.club:443 taro [14/Jul/2017:09:59:14 +0900] "GET / HTTP/1.1" 401 935 "-" ...(略)...
```

たとえばApacheでBASIC認証をかけたときに、Webアプリ上でユーザ名を知る必要があるのはもちろん、トラブル発生時に「そのユーザからのアクセスがあったのどうか」「認証に成功したのか失敗したのか」を調べる必要が出てきます。これらをきちんとログに記録し、正しい見方を押さえておくことで、利用者が実際にアクセスしてきているのか、認証に成功しているのかなどを追うことができます。

Apacheのアクセスログの場合では、あらかじめhttpd.confなどで、アクセスログにユーザ名も記録するような設定になっている必要があります。最近のディストリビューションではデフォルトのままでも記録されます。HTTP BASIC認証の例では、

①ブラウザが資格情報なしでアクセス
②Webサーバが認証を要求し、HTTPレスポンスコード401(認証が必要・認証に失敗)を返す
③ブラウザがユーザに資格情報の入力を促す
④資格情報をAuthorization:ヘッダに付与してアクセス
⑤資格情報が正しければアクセス許可しHTTPレスポンスコード200(正常)を返し、正しくなければ401を返す

というような流れでやりとりされます。そのため、Apacheのアクセスログではまず、**リスト1**のように資格情報なしで401が返されます。次にブラウザから資格情報を付与してアクセスすると、**リスト2**のようにユーザ名(ここではtaro)とHTTPレスポンスコード200がログに記録されます。パスワードが正しくないなどでアクセスが拒否されると、**リスト3**のようにユーザ名と401の両方が記録され、taroさんが認証に失敗したと判断できます。

とくに金融取引などに関係する重要なシステムでは、監査証跡目的でアクセスログの保管が7年間義務づけられています。ログを定期的に別のシステムにコピーして保管し、同時に改ざんされていないことを確認する目的で、保管時にハッシュ値も取ってさらに別のシステムに保存するなどのしくみを設計すると良いでしょう。

運用の理想は「安心して放置しておけること」です。より複雑な認証を行うシステムでも、ログにきちんとアクセスでき、どこまで成功して何に失敗したのかが明確に記録される製品を選択することが、最も重要となります。

◆ ◆ ◆

認証は、いちユーザから見ているだけなら、IDとパスワードを入力するとWebアプリが利用できる、というだけの地味なものですが、一方で働き方の多様化、スマートフォンやタブレットなどのモバイルデバイスの普及、BYODなどを考えると、認証・認可に基づくアクセス管理は以前よりも複雑、かつ重要になってきています。利便性とセキュリティをできる限り両立させながら、スムーズに業務を遂行できるしくみを、「認証・認可のファイアウォール」を活用していくことでスマートに実現していけるものと考えています。SD

光学ドライブ好き違う方法で“再生”

Ejectコマンドで遊んでみませんか？

―― スマートコンセントで扇風機を回そうの巻

Author あっきぃ　Twitter@Akkiesoft

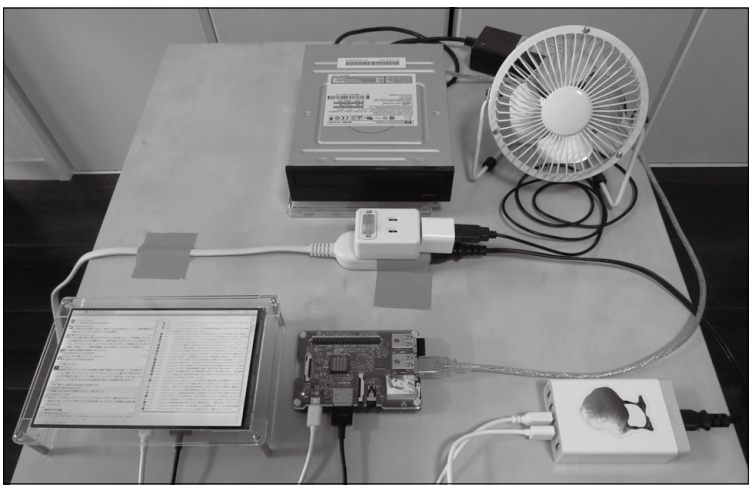
こんにちは、Ejectコマンドユーザー会のあっきぃ(@Akkiesoft)です。Software DesignにまたEjectがやってきました。今回はUSB扇風機をTwitterから簡単にオン・オフできる夏にピッタリなEjectコマンド工作として「Ejectスマートコンセント（**写真1**）」を作成していきます。

Ejectコマンド工作とは

Ejectコマンド工作について簡単におさらいします。Ejectコマンド工作とは、5インチベイタイプの光学ドライブをトレイ開閉させて、その動作を利用したピタゴラスイッチのような工作物を作り、リモートから実行可能にしたものです。おもちゃとして遊びに使ったり、生活に役立てたりできます。

前回(本誌2017年1月号)は年末の風物詩「除夜の鐘」をEjectコマンド工作で作って遊ぶ方法について紹介しました。

▼写真1　Ejectスマートコンセント



スマートコンセントを作ろう

スマートコンセントは、リモートから電化製品の通電・遮断を制御できる装置・製品です。昨今話題のIoTを組み合わせて、IoTコンセントとも呼ばれているようです。筆者はスマートコンセントの製品を購入、使用したことはないのですが、スマートコンセントはIoT製品として家庭用に売られているほか、業務用としても販売されています。

家庭向け製品はスマートコンセントをWi-Fiなどで接続して、スマートフォンのアプリなどから遠隔で家電製品の通電・遮断ができます。製品によっては通電中の電力使用量に異常を感知したら通知するような機能もあります。価格は2,000円ほどから購入できるようです。

業務用のスマートコンセント製品は、コン

ピュータやルータなどのITインフラの管理に特化した機能が搭載されているものがあります。たとえば、ping監視結果などの結果から電源のオン・オフを実行したり、コンピュータの電源を切る前にシャットダウンスクリプトを実行できたり、スケジュール機能が搭載されていたりします。こちらは8万円前後で販売されています。

注意事項

スマートコンセントは活用方法しだいでは非常に便利なものですが、火災や事故などのリスクには注意が必要と言えます。リモートから稼働させたときに周囲の安全が確認できないと危険な器具との組み合わせはするべきではありません。

たとえば、冬になると暖房器具にスマートコンセントを組み合わせたくなったりするかもしれません。しかし、リモートから操作したときに不幸にも暖房器具の上に可燃物が乗っていて引火……といったこともあるかもしれません[注1]。そのようなことを考える人はいないのでは考え過ぎでは？と思われるかもしれませんが、過去にEjectコマンド工作でやろうとしていた方を見かけて止めたことがあります。

光学ドライブでスマートコンセントを作るには

さて、光学ドライブで作る“Ejectスマートコンセント”では、トレイの動きを使って電源をオン・オフさせるために、100円ショップなどで売られているスイッチ付き電源タップを使用します。節電タップなどとも呼ばれています(**写真2**)。スイッチと光学ドライブの組み合わせについては前回も少し触れましたが、スイッチを光学ドライブの前に設置することで、トレイの開閉時にスイッチに引っかかり、オン・オフの動作をさせることができます。

▼写真2　スイッチ付き電源タップ

Twitterからリモート操作できるようにするためには、BOTプログラムを用意する必要があります。今回はRaspberry Pi上にTwitterクライアント「mikutter」をインストールして、Ejectを実行するためのプラグインを作成します。

工作の材料

工作の材料は次のとおりです(**写真3**)。

- 光学ドライブ(IDEもしくはSATA) … 1台
- IDE(SATA)-USB変換ケーブル・電源アダプタセット …… 1つ
- Raspberry Pi …… 1つ
- スイッチ付き電源タップ …… 1つ
- 適当なテーブルタップ …… 1つ
- USB扇風機用AC-USB変換アダプタ …… 1つ
- USB扇風機 …… 1台
- 光学ドライブの高さ調整をするもの … 適当
- 養生テープ …… 適宜

100円ショップのスイッチ付き電源タップはショップによって形状が異なりますが、スイッチの操作が試せそうなら、できるだけ軽い力で操作できるものを選択してください。操作するときに力が必要なものでは、光学ドライブのトレイを押す力・引く力が足りず、操作に失敗することがあります。いくつか試してみたところ、モリトクのFLD-202Jというスイッチが工作に向いていました。

注1) そもそも暖房器具は、スマートコンセントを使っても使わなくても取扱いに注意が必要と言えますが。

▼写真3　今回の材料

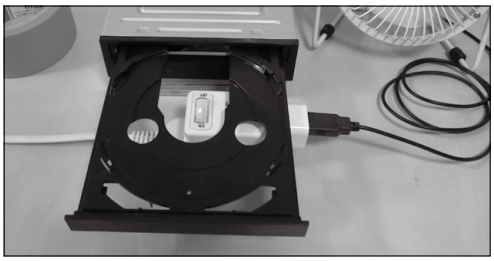
工作しよう

スイッチ付き電源タップのスイッチ部が上になるように設置するため、テーブルタップを用意して床に固定し、そこに差し込みます。スイッチの向きは、オフが光学ドライブに向くように設置してください。スイッチ付き電源タップにAC-USB変換アダプタとUSB扇風機をつなげておきます。

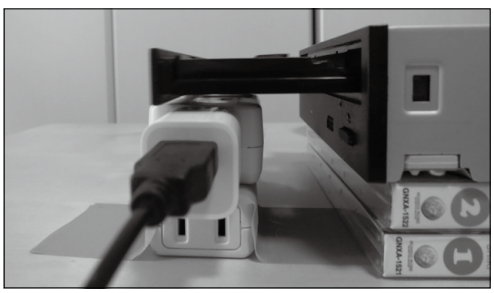
設置したスイッチ付き電源タップのスイッチの高さに合わせて、光学ドライブの設置する高さを調整します。光学ドライブや電源タップの機種によって必要な高さが異なるため、いろいろなものを置いて試してみてください。今回はアニメのブルーレイディスクケース2枚がちょうどよい高さになりました。物理的な工作はこれで完成です(**写真4**)。設置できたら手でEjectさせてみて、トレイがスイッチをオン・オフすることを確認します(**写真5**、**写真6**)。

▼写真4　Ejectスマートコンセントの工作部分

▼写真5　電源がオンになった様子



▼写真6　トレイがスイッチに引っかかる様子



光学ドライブの調整が済んだら、電源ケーブルを配線し、USB変換ケーブルで光学ドライブとRaspberry Piとをつなぎ合わせます。

mikutter環境の用意

mikutterは、Ruby言語で書かれたオープンソースのTwitterクライアントです。プラグインで自由に拡張できるという特徴を持っており、さまざまなプラグインがユーザ有志によって作成されています。たとえば、定形ツイート機能や、キーワードに反応して返信するBOTのような機能のほか、Slack[注2]やMastodon[注3]など別のSNSクライアントプラグインを開発中の方もいます。今回はBOTプラグインをEject用に作成していきます。

Raspberry PiにRaspbianの環境を作成します。デスクトップが含まれたRaspbian Jessie With PIXELを用意してインストールします。

RaspbianへのmikutterのインストールはI、

注2) mikutter_slack(@ahiru3netさん)https://github.com/Na0ki/mikutter_slack/

注3) mikutter-don(@mogunoさん)https://github.com/moguno/mikutter-don

コラム「Eject職人の朝は早い。目覚めをサポートしてくれる光学ドライブ」

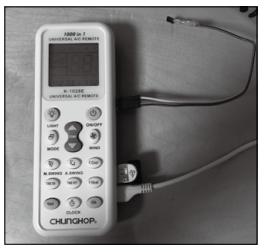
光学ドライブによるEjectコマンド工作を作りはじめて早8年ほどになります。生活の些細な困りごとや、改善できそうなものごとを解決するためにいろいろな作品ができましたが、常用に至ったEjectコマンド工作は、エアコンのリモート操作と、カーテンを開く装置の2つです(**写真A**、**写真B**)。

エアコンのリモート操作は、電源ボタンにプラ板で作ったバネを取り付けてトレイに押させるといった非常にシンプルなものでした。現在はRaspberry Piにその役割が移りましたが、「リモコンの電源ボタンを押してもらう」という制御のコンセプトは変わらず、フォトカプラーを使用して汎用リモコンの電源ボタンに相当する回路をショートさせています。本来ならば、赤外線学習を実装したりするものですが、汎用リモコンはメーカーコードの設定だけで作業が済むという利便性があります。もしエアコンのリモート操作の方法に興味があれば『Raspberry Pi[実用]入門』(技術評論社刊)で紹介していますのでよければ見てみてください。現在のエアコンリモート操作は、小型のRaspberry Pi Zeroを汎用リモコンに埋め込むことで小型化がさらに進みました(常用しているのはこれではなく普通のRaspberry Piとリモコンですが……)。

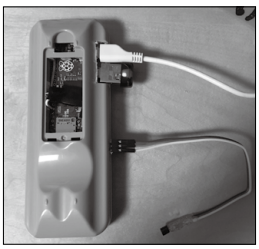
もう1つのカーテンを開く装置は、現在も毎朝使用している装置です。まず、カーテンレールに滑車と紐を取り付けて、紐を引くとカーテンが開くしくみを作ります。紐の先にはおもりを取り付けて、光学ドライブのトレイの前に置きます。光学ドライブがトレイを開くと、おもりが突き落とされて、ヒモが引かれ、カーテンが開くというしくみです(**写真C**)。なお、カーテンを閉じることはできませんので、そこは手動になります。IoT製品では「めざましカーテン mornin'」が近いです。こちらはカーテンの開閉両方に対応しています。

カーテンを開く装置ももちろん制御にRaspberry Piを使用しています。cronジョブで毎朝6時ごろにEjectをするだけでほかには何もしておらず、起動時の時刻同期以外はネットワークにも接続していないので少しもったいない気もしますが、安価ですのでヨシとしています。また、Ejectコマンド工作は装置一式がかさばるという問題を少しでも緩和するために、光学ドライブから不要なコンポーネントを取り外してRaspberry Piを光学ドライブに内蔵しました。Raspberry Piの電源を光学ドライブから供給できるように手を加えてあるので、光学ドライブに電源をつなぐだけで一式が稼働を始めます。このあたりも機会があれば紹介できればと思います。

▼写真A　エアコンリモート操作装置(その1)



▼写真B　エアコンリモート操作装置(その2)



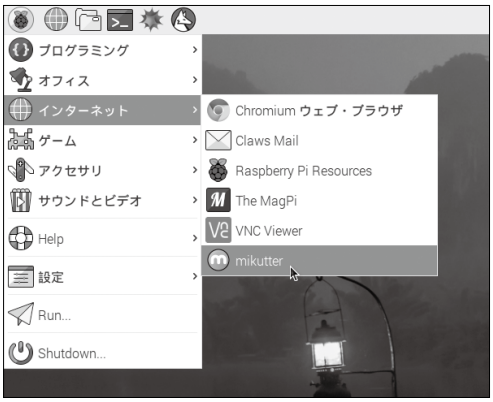
▼写真C　カーテンを開く装置

aptコマンドが利用できます。なお、ejectコマンドのパッケージも必要になるため、ここで一緒にインストールします。

```
pi@raspberrypi:~ $ sudo apt-get update
pi@raspberrypi:~ $ sudo apt-get install -y
mikutter eject
```

スタートメニューのインターネットにmikutterという項目が追加されるので、ここからmikutterを起動します(**図1**)。初回起動時はアカウントの設定が必要になります。ゲームのチュートリアルのように案内してくれるので、とくに迷うことなく設定できるでしょう(**図2**)。

▼図1　mikutterの起動方法



▼図2　アカウント設定後のmikutterの画面

mikutterプラグインの作成

いよいよEjectスマートコンセント用BOTプラグインの作成です。BOT系プラグインとしては、特定のキーワードに反応してリプライを送る「あひる焼くな」プラグイン[注4]というものがすでに公開されているので、これを参考にEject用プラグインを**リスト1**のように作成しました。

このプラグインは、ツイートがタイムラインに表示されるときに発生するon_appearイベントを利用して、自分がツイートしたときだけejectコマンドを実行します。キーワードに反応して動作させる方法もありますが、Twitterでは同じツイートを連続投稿できず、トレイ開閉を繰り返すには不便なため、ツイートするだけで動作するようにしました。また、Ejectを実行したときは、開閉したことをmikutterのアクティビティタブで表示するようにしています。

ejectコマンドはsystem()で実行させます。パラメータを付けない場合はトレイを開き、-tオプションを付けた場合はトレイを閉じることができます。

プラグインファイルは/home/pi/.mikutter/plugin/remote_eject.rbに配置します。配置したあとはmikutterを再起動してください。

完成！

mikutterで何か適当にツイートをしてみましょう。mikutterがejectコマンドを実行すると、光学ドライブのトレイが開きEjectスマートコンセントがオンに変化します。もう一度ツイートすると、今度は光学ドライブのトレ

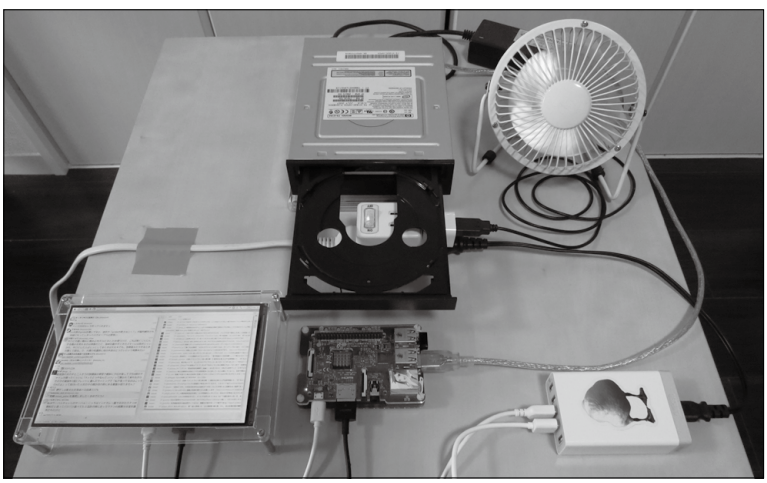
注4) ahiru_yakuna(@ahiru3netさん)https://github.com/Na0ki/ahiru_yakuna

イが閉じてEjectスマートコンセントがオフに変化します(**写真7**)。

▼リスト1　Ejectプラグイン(remote_eject.rb)

```
# remote_eject.rb
# mikutter remote_eject plugin
# reference: https://github.com/Na0ki/ahiru_yakuna

Plugin.create :remote_eject do
  defactivity "remoteeject", "リモートEject"

  # プラグインの起動時間を記憶
  @boottime = Time.new.freeze
  # トレイの開閉フラグ
  @tray = 0

  # ejectコマンドの実行内容
  @eject = "/usr/bin/eject /dev/sr0".freeze

  # メッセージがタイムラインに差し込まれた時のイベント
  on_appear do |messages|
    messages.each do |message|
      # 起動前の過去のツイートは無視
      next if message[:created] < @boottime
      # 自分以外はEjectできない
      next if Service.primary.user_obj.id != message.user.id
      # リツイートは対象外
      next if message.retweet?

      if @tray == 1
        activity :remoteeject, "トレイを閉じます"
        system(@eject + " -t")
        @tray = 0
      else
        activity :remoteeject, "トレイを開きます"
        system(@eject)
        @tray = 1
      end
    end
  end

end
```

▼写真7　稼働中のEjectスマートコンセント



mikutterからツイートする必要はありませんので、スマートフォンなどから同じTwitterアカウントでツイートしても、もちろんEjectスマートコンセントが動作します。

まとめ

光学ドライブとスイッチ付き電源タップとRaspberry Pi上のmikutterを組み合わせた、Ejectスマートコンセントの作り方について解説しました。

Ejectコマンド工作全般に言えることではありますが、恒久的に使用するにはEjectスマートコンセントは場所を取り過ぎてしまうため、この点では市販製品のほうが有利です。安定性も市販製品には劣るでしょう。しかし、たまにしか発生しないちょっとした用事なら、わざわざ買わずともさっと光学ドライブを取り出して作れる・使えるという点はEjectコマンド工作の魅力です。

冒頭で挙げた既製品の機能を実装する場合、電力使用量の異常検知など高度な機能を作ることは難しそうですが、業務用製品のような監視機能やシャットダウンスクリプト機能を実装することは、シェルスクリプトなどを作成すれば実現できそうです。こちらもぜひチャレンジしてみてはいかがでしょうか。SD

人工知能時代のLispのススメ

～ラムダ式からLispの作り方まで

最終回

Java 500行でLispを作る！これがLispを理解する一番の早道

Author 五味 弘（ごみ ひろし）　沖電気工業㈱

プログラミング言語を深く理解したいなら、その言語を作ってみるのが一番の早道です。少し無茶振りかもしれませんがLisperなら当然です。そこで今回はLispを深く学ぶために、Java 500行でLispを作ってみることにします。これであなたもきっとLispを愛することになるでしょう。

Lispのデータを作る

Lispを500行で作るなんて無理だと思うかもしれません。でも階乗計算やフィボナッチ関数[注1]、たらい回し関数（tarai）[注2]ぐらいを動作させるだけなら、実は400行もいりません。案外、簡単に作れますので安心してください。

まずLispを作るときにはデータ構造から作ります。シンボルとリスト、整数を作れば、とにかくLispぽく見えます。本当です。

1 どんなものでも入るデータを作る

Lispのシンボルにはどんな型のデータでも格納でき、型宣言も必要としません。最初にこのLispデータを作ります。つまりどんな型でも格納できる汎用的なデータ構造を作りますが、そのデータの型情報（これをタグと呼びます）をどこかに持たせる必要があります。

このとき、プロセッサの基本サイズ（intのサイズ）内にタグを格納する（即値型コース）のか、それとも外出し（構造型コース）にするのか、タグをクラス情報として管理する（タグお任せコース）のか、オブジェクトを配置するアドレスをタグ代わりにする（アドレスコース）のかなど、いろいろなタグの実装方法があります。またガベージコレクション（GC）を手作りする場合であれば、GCのための情報をタグと同様にどこかに置くのか、それとも実装言語処理系のGCにお任せするのかでデータ構造も変わってきます。**図1**にいろいろなLispのデータ構造の例を示します。

今回の実装では一番お手軽な**図1**中の（3）の「タグ・GCお任せコース」を選びます。実装方針も「簡単、小さい」にしてJava 500行を目指します。これを選んだ場合は型と副型[注3]の関係を実装言語のクラス継承にマッピングすると型システムが簡単に実装できますが、Common Lispの型システムは複雑ですので、お手軽に作る場合は大胆に簡略化した方がよいでしょう。今回、採用する型システムを**図2**に示します。

図2の「T」型はJavaやSmalltalkのObjectに相当するルートクラスで、S式すべてが入る型になります。またConsはnilでないリストです。またnilはリストでもあり、シンボルでもある（多重継承している）ことに注意してください。つまりnilは二重人格なのです（そしてnilは真理値の偽も意味しますから三重人格です）。

注1）再帰的処理の例によく使われる関数。前の値と前の前の値を加えた値を今の値にする数列。自然界によく出現する。

注2）竹内郁雄氏によって作られたベンチマーク用関数。処理をたらい回しして大量の計算を実行する。

注3）副型とは親の型を継承する型。たとえば、数型の副型には複素数型と実数型があり、実数型の副型には有理数型、浮動小数点数型、分数型があり、有理数型には整数型があり、整数型には固定整数型や無限整数型などがあります。

❷ リストこそLispの源

まずはリストを作ってみます。リストはConsクラスで実装し、コンスセル（carとcdrがセットになったセル）を作るだけです。**図2**のようにConsクラスはListクラスを継承し、Listは「T」を継承しています。リストの要素は何でも入れることができますので、carとcdrのクラスは「T」にしています。実装は**リスト1**のように簡単で、リスト構造の例を**図3**に示します。

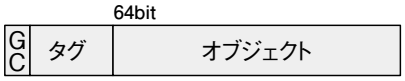
❸ 整数型も作る

何はともあれ、整数型を実装してみます（**リスト2**）。実装はこれだけです。

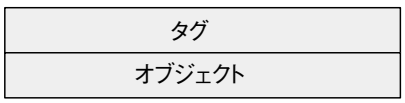
▼図1　Lispのデータ構造の実装

(1) 即値型・GC手作りコース

可変長先頭タグ・現物タグ方式・GCビット方式
（他方式もいろいろ。高速化・省メモリのときにおススメ）



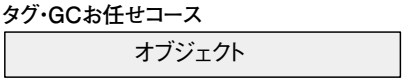
(2) 構造型・GCお任せコース




```
class Integer {
 int tag;   // タグ(型情報)をオブジェクトから外出し
 int value; // オブジェクト
} // GCは実装言語処理系まかせ
```

（Cの構造体で作るときのおススメ。ただしCではGCは手作りになる）

(3) タグ・GCお任せコース

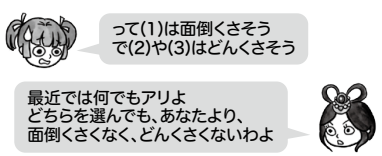



```
class Integer extends Number {
 int value;
} // タグ(型情報)はクラスから得る
```

（Javaなどのクラスで作るときのおススメ）

(4) その他アラカルトコース

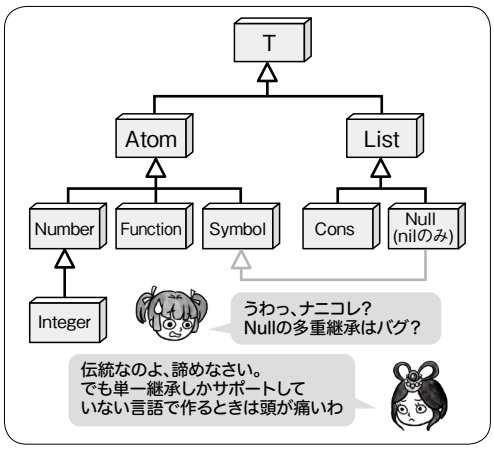
オブジェクトマップ方式、アドレス識別方式、……



❹ シンボルこそLispの肝

同じ名前のシンボルは唯一になるように実装します。このためにハッシュテーブルを用いて、既存のシンボルを調べて、なければ新規のシンボルを生成しハッシュテーブルに登録（これをインターンと呼びます）、あれば既存のシンボルを

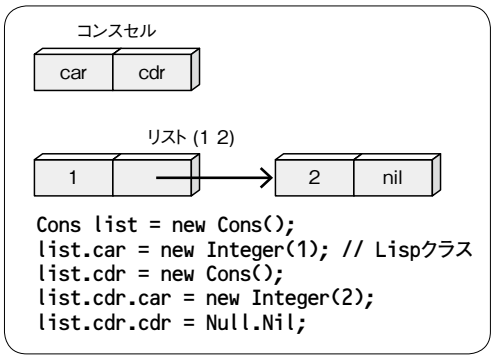
▼図2　Lispの型システム（Common Lispの型システムを大胆に簡略化している）



▼リスト1　リストの実装

```
public class Cons extends List{
  public T car; // コンスセルのcar部
  public T cdr; // コンスセルのcdr部
} // セッタとゲッタは行数削減のため省略
```

▼図3　リストのデータ構造




```
Cons list = new Cons();
list.car = new Integer(1); // Lispクラス
list.cdr = new Cons();
list.cdr.car = new Integer(2);
list.cdr.cdr = Null.Nil;
```

▼リスト2　整数の実装

```
public class Integer extends Number {
  public int value;
  public Integer(int value){
    this.value = value;
  }
}
```

返します。その実装を**リスト3**に示します。

なお、Lispのシンボル「T」はシステムで定義されていて、その値は自分自身になっていますので、**リスト4**のように実装します。**図4**にこれらの関係を示します。

Lispのリーダとプリンタを作る

次にS式を表示するプリンタと、キーボードやファイルなどから入力された文字列をS式にするリーダを作っていきます。

1 プリンタで見える

S式を表示するプリンタは、S式を文字列に変換するシリアライザ[注4]を作ることで代替します。メモリは無駄になるかもしれませんが、変換した文字列を画面表示すればプリンタになります。プリンタ(シリアライザ)は各データ構造のクラスでメソッド関数「toString」を作ります。シンボルや整数のプリンタは簡単ですので、ここではリストのプリンタ(シリアライザ)を**リスト5**に紹介します。

2 リーダでキーボードから入力できる

リーダは、キーボード入力された文字列からS式を生成します。最初の文字が開き括弧であればリスト、数字であれば整数型、英字であればシンボルを生成します。また**'abc**がくれば**(QUOTE ABC)**、**'(1 2 3)**であれば**(QUOTE (1 2 3))**のように読み替えなければなりません。本当に面倒です。大文字インターン(小文字でキーボードから入力されても、シンボルテーブルには大文字でインターンする)もリーダの仕事で、わざわざ小文字を大文字に変更します。**リスト6**にリーダの一部を示します。

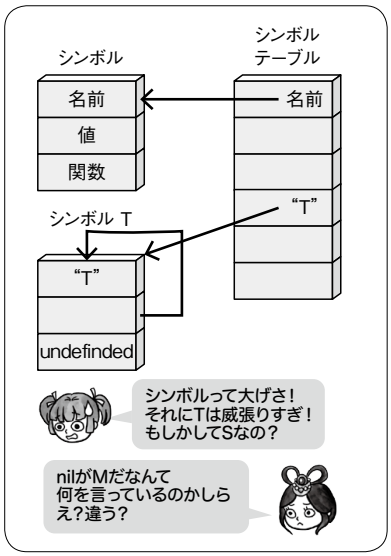
ここまでで、キーボードから**(add 10 20)**のような整数やシンボルのリストを読み込み、それを**(ADD 10 20)**のS式に変換して、それが表示できるようになりました(まだ実行はできません)。

注4) シリアライズ(直列化)とはばらばらに配置されているデータオブジェクトを一方向の列の並びにすること。たとえば、データの内容を文字列やバイト列に変換して直列化すること。シリアライザはシリアライズするプログラム。

▼リスト3 シンボルの実装

```
public class Symbol extends Atom {
  public String name;       // シンボルの名前
  public T value;           // シンボルの値
  public T function;        // シンボルの関数
  // シンボルテーブル
  private static HashMap<String, Symbol> symbolTable
    = new HashMap<String, Symbol>();
  private Symbol(String name){ this.name = name; }
  /** シンボルの生成とインターン */
  public static Symbol symbol(String name) {
    if (symbolTable.get(name) == null){
        Symbol symbol = new Symbol(name);
        symbolTable.put(name, symbol);
    }
    return symbolTable.get(name);
  }
}
```

▼図4 シンボルの実装



▼リスト4 シンボルTの実装

```
public static Symbol symbolT = Symbol.symbol("T"); // シンボルTの登録
static { symbolT.value = symbolT; }                // Tの値は自分自身
```

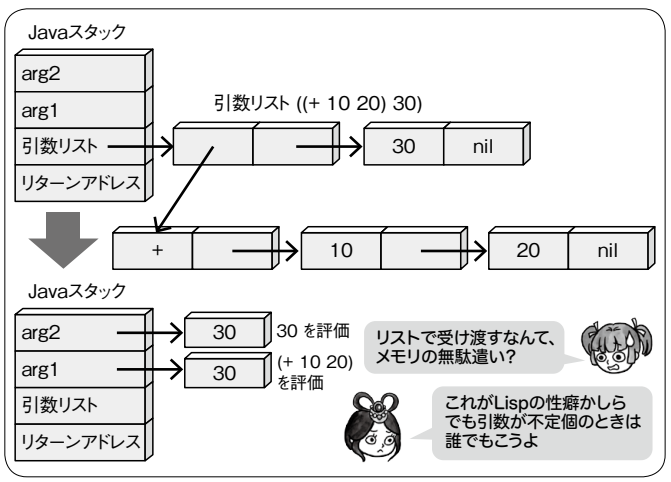
▼リスト5　リストのプリンタ（シリアライザ）

```
public String toString() {
  StringBuilder str = new StringBuilder();
  Cons list = this;
  str.append("("); // Open "("
  while (true) {
    str.append(list.car.toString());      // Car部
    if (list.cdr == Null.Nil) {
        str.append(")");                  // Close ")"
        break;
    } else if (list.cdr instanceof Atom) {
        str.append(" . ");                // ドット対
        str.append(list.cdr.toString());  // Cdr部
        str.append(")");                  // Close ")"
        break;
    } else {
        str.append(" ");                  // 空白
        list = (Cons)(list.cdr);          // 次のCdr部へ
    }
  }
  return str.toString();
}
```

▼リスト6　リーダ（の一部）

```
private static T getSexp(){
  while (true) {
    skipSpace();  // 空白の読み飛ばし
    switch (ch) {
      case '(' : return makeList();
      case '\'': return makeQuote();
      case '-' : return makeMinusNumber();
      default :
        if (Character.isDigit(ch)) return makeNumber();
        return makeSymbol();
    }
  }
}
```

▼図5　システム関数のコーリングコンベンション



Lispの関数を作る

ここではシステム関数（Javaで作る関数）とS式関数（Lispで作る関数）を作っていきます。

1 コーリングコンベンション（呼び出し規約）を決める

コーリングコンベンションとは関数を呼び出すときに、スタックにどのように積んでいくかの規約です（**図5**）。どの順序で、いつ誰が引数を評価するのかなどを決めるものです。今回はシステム関数の場合は引数を評価せずに、Javaスタック[注5]にリスト形式で積むようにします。つまり、呼び出し側ではほとんど何もせずに、呼び出され側に全部やってもらう方式です。たとえば、引数2個の場合は**リスト7**のようになります。

呼び出し側でたとえば**(+ (+ 10 20) 30)**のようなシステム関数呼び出しであれば、**((+ 10 20) 30)**のような引数リストが呼び出され側の「+」に来ます。関数「+」では引数リストのcarを取り、**(+ 10 20)**が第1引数になります。これを**リスト7**のEval.evalで評価し、30が返り、arg1に代入されます。続いて引数リストのcdrを取った**(30)**のcarを取り、30になります。30を評価すると30になりますので、arg2には30が代入されます。最初の30と後の30を用いて「+」が実行され、60が返ります。

次にS式関数の場合も同様ですが、引数を評価し、それを変数束

注5）Java VMが使用しているスタック。

縛を行った結果をLispスタック[注6]に積みます。

❷ システム関数はおまかせ

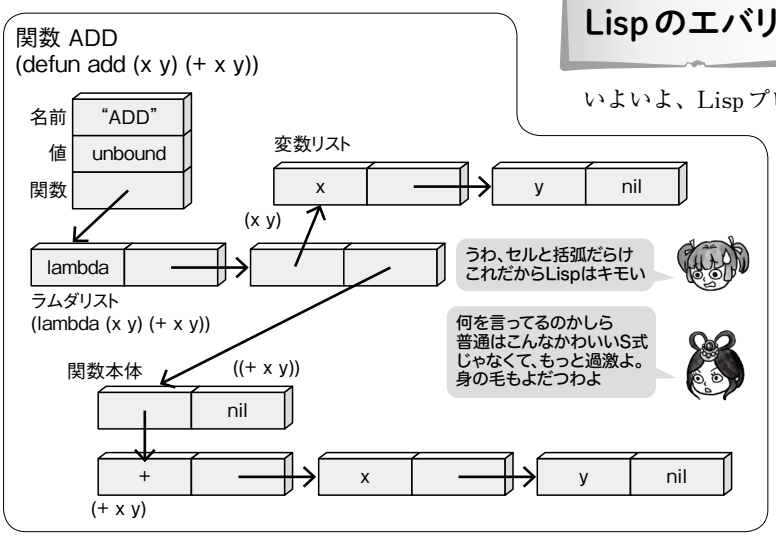
今回はcarやcdr、cons、eq、ifの基本5関数と四則演算、比較演算などと特殊形式のquoteやsetq、defunをJavaで作ります。これだけであれば、関数のルックアップ(検索)もシーケンシャルで検索しても問題ありませんが、将来関数の個数が増える場合を考えて、O(1)[注7]でルックアップできるようにします。実装はFunctionクラスの内部クラスとして関数を実装し、その関数クラスのインスタンスをシンボルの関数部に入れます。そしてそのシンボルをシンボルテーブルに格納します。たとえば、**リスト8**のようにシンボルCARをシンボルテーブルに格納し、シンボルCARの関数部に関数CARのインスタンスを入れます。また内部クラスCarの定義も**リスト8**に紹介しています。

❸ S式関数はリストで

S式関数は、関数をS式(リスト)で作ります。たとえば、**(defun add (x y) (+ x y))**の関数であれば、シンボル「ADD」の関数として、**(lambda (x y) (+ x y))**のようなラムダリストを格納します(**図6**)。

注6) Lisp処理系がJava VMのヒープ上に作るスタック。
注7) O(1)は、オーダー1と読み、実行時間が個数に依存することなく一定であることを示しています。

▼図6　S式関数の実装



Lispのエバリュエータを作る

いよいよ、Lispプログラムを実行する心臓部であるエバリュエータ(評価機構)evalを作ります。Javaのスタティックメソッド Eval.evalで作ります。ここでは評価機構本体と変数束縛を見ていきます。

▼リスト7　関数呼び出しの例(2引数評価)

```
public T funcall(List arguments) throws Exception { // 引数はリストで受ける
  T arg1 = Eval.eval(((Cons)arguments).car);             // 第1引数を評価
  T arg2 = Eval.eval(((Cons)((Cons)arguments).cdr).car); // 第2引数を評価
```

▼リスト8　クラスCarの定義

```
Symbol sym = Symbol.symbol("CAR"); // シンボルCARをシンボルテーブルに格納
sym.function = new Car();          // シンボルCARの関数部にCarクラスのインスタンスを代入
/** CAR */
class Car extends Function {
  public T funcall(List arguments) throws Exception {
    T arg1 = Eval.eval(((Cons)arguments).car);
    return arg1 == Null.Nil ? Null.Nil : ((Cons)arg1).car;
  }
}
```

1 evalのお仕事

関数evalは与えられたS式を評価します(**図7**)。ここではJavaのEvalクラスのスタティックメソッドevalでこれを実装することにします。まずevalは数値や文字などの即値型がくればそのまま返します。次にシンボルを評価すると、その値を取り出します。これは一般のプログラミング言語で変数の値を取り出すのと同じです。

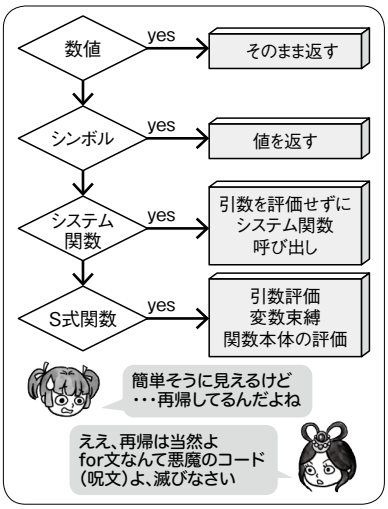
S式がリストの場合はその第1引数を関数とみなして評価します。たとえば**(foo (bar x))**のリストを評価するときは、fooを関数とみなして、**(bar x)**をその関数の引数として評価します。そして引数の評価のときに**(bar x)**のbarを関数として、xをその引数として、再帰的に評価します(**図8**)。

第1引数がシステム関数のときは前節で紹介したように引数の評価や変数束縛は各システム関数が行いますので、関数evalは**リスト9**のようにそれを呼び出すだけになります。

S式関数のときは、引数評価を左側優先・内側優先で行い[注8]、引数の評価と同時に変数束縛を行います。そして関数本体のS式を評価します(**リスト10**)。たとえば、**(defun add (x y) (+ x y))**で定義されているとして、**(add 10 20)**の評価を考えます。まず引数評価では10と20はそのままの値が返り、変数束縛として、xが10に、yが20に束縛されます。そして関数本体**(+ x y)**は変数束縛から**(+ 10 20)**になり、システム関数の+を呼び出し、その結果として30が返ります。

注8) 左側優先や内側優先などの評価戦略はシステム関数ではいろいろな場合があります。つまりシステム関数のときの評価は各システム関数に任しています。たとえばifのときは第1引数を評価して、その結果が真か偽かによって、第2引数か第3引数を評価する戦略になります。

▼図7　evalのお仕事



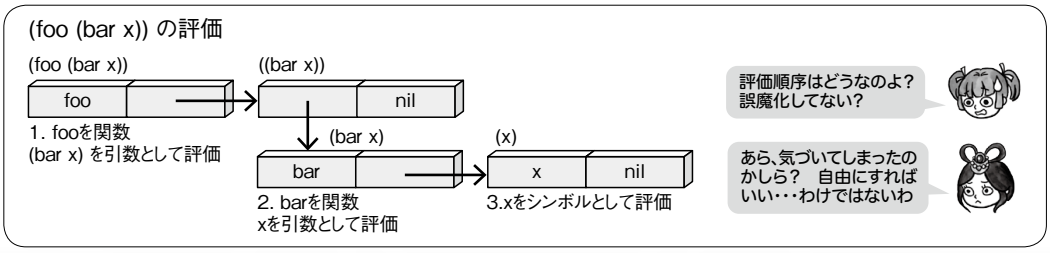
▼リスト9　システム関数の評価

```
// システム関数の評価
if (fun instanceof Function) {
   T argumentList = ((Cons)form).cdr;
   return ((Function)fun).funcall((List)argumentList);
}
```

▼リスト10　S式関数の評価

```
// S式関数の評価
if (fun instanceof Cons) {
   Cons cdr = (Cons)((Cons)fun).cdr;
   T lambdaList = cdr.car;
   Cons body = (Cons)cdr.cdr;
   // 引数がないときは、束縛せずに本体bodyをそのまま評価
   if (lambdaList == Null.Nil) return evalBody(body);
   // 引数があるときは、引数評価と束縛、本体bodyを評価
   return bindEvalBody((Cons)lambdaList, body, (Cons)((Cons)form).cdr);
}
```

▼図8　リスト(関数)の評価



2 変数を束縛する

変数束縛は再帰呼び出しにも対応しなければなりません。つまり前の変数束縛をどこかに保存し、再帰呼び出しから戻ったら、変数束縛を戻す必要があります。この変数束縛の実装には浅い束縛と深い束縛があります。浅い束縛はシンボルの値に最新の変数の値を格納して、以前の変数束縛をスタックに格納します。シンボル経由ですので値の検索はハッシュテーブルアクセスになり、O(1)の高速な変数検索を行います。ただし再帰が多くなるとシンボルとスタックとのやりとりが煩雑になります。一方、深い束縛はスタックに変数束縛を順に格納し、スタックを深く使います。値検索はシーケンシャルな検索になり遅くなります。しかし再帰のやりとりが簡単で実装は楽です(**図9**参照)。今回はあえて面倒な浅い束縛で実装します(**リスト11**)。

Lispのトップレベルを作る

ここまででLispの基本的なデータ構造から、リーダ、プリンタ、そしてエバリュエータが完成しました。ここではこれらを組み合わせたread-eval-printループを作り、これがLisp処理系のトップレベルループ(コマンドプロンプトみたいなもの)になります。

1 read-eval-printでGo!

Lispのトップレベルループは、readの前にプロンプト(今回は>)を表示し、キーボードから打鍵された文字列を読み込み(read)、S式を生成します。そのS式を評価(eval)し、その結果を表示(print)します。これを繰り返します(**リスト12**)。

これでLisp処理系が完成しました。最初の約束どおり、500行程度の大きさになりました。このLisp処理系の使用例を**図10**に紹介します。

2 ダウンロード

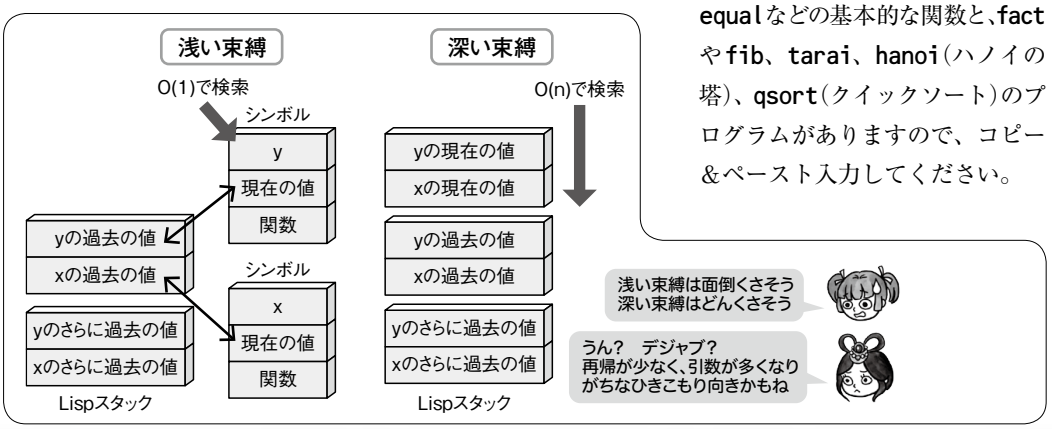
今回、紹介してきたLisp処理系(名付けてSDLisp)のソースプログラムを次のURLで公開しています。

http://train.gomi.info/lisp/sdlisp.html

SDLispについて

このSDLispのシステム関数には、`car`や`cdr`、`cons`、`eq`、`if`の基本5関数、四則演算(`+`、`-`、`*`、`/`)、比較演算(`>`、`<`、`>=`、`<=`、`=`)、型を返す`type-of`、関数を取り出す`symbol-function`と、そして特殊形式の`quote`や`setq`、`defun`があります。また`'`(クォート)を`QUOTE`に展開するリードマクロも実装しています。start.lspには SDLispで動作する`1-`や`1+`、`null`、`atom`、`symbolp`、`numberp`、`listp`、`append`、`reverse`、`last`、`member`、`assoc`、`eql`、`equal`などの基本的な関数と、`fact`や`fib`、`tarai`、`hanoi`(ハノイの塔)、`qsort`(クイックソート)のプログラムがありますので、コピー&ペースト入力してください。

▼図9 浅い束縛と深い束縛



さらに機能を拡張したSD2Lispと、機能をぎりぎりに抑えて360行で作ったSD0Lispもあります。Lispの実装をどのようにしていけばいいのかを楽しんでください。きっとLispと友達になれるでしょう。SD

▼リスト11　S式関数の引数評価と変数束縛

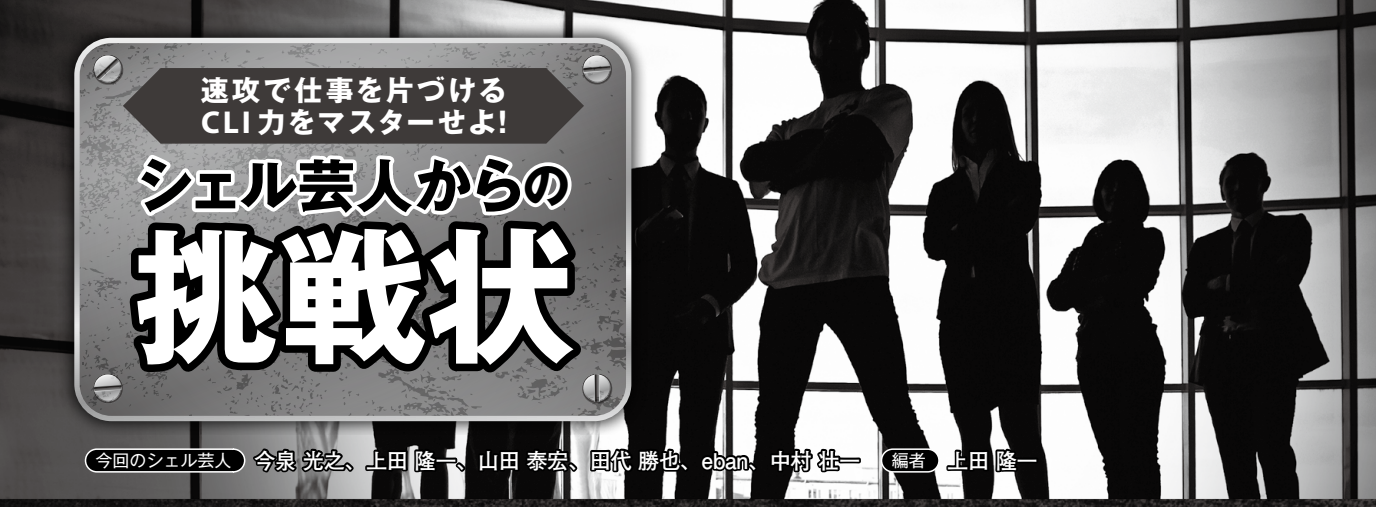
```
private static T bindEvalBody(Cons lambda, Cons body, Cons form) throws Exception {
  // (1) 束縛前の環境で引数評価 (評価した値の格納場所に一時的にスタックを使用)
  int OldStackP = stackP;
  while (true) {
    T ret = eval(form.car);
    stack[stackP++] = ret;
    if (form.cdr == Null.Nil) break;
    form = (Cons)form.cdr;
  }
  // (2) 束縛 (シンボルの過去の値をスタックに退避し(1)で評価した値をシンボルに入れる)
  Cons argList = lambda;
  int sp = OldStackP;
  while (true) {
    Symbol sym = (Symbol)argList.car;
    T swap = sym.value;
    sym.value = stack[sp];
    stack[sp++] = swap;
    if (argList.cdr == Null.Nil) break;
    argList = (Cons)argList.cdr;
  }
  // body の評価
  T ret = evalBody(body);
  // 以降は、束縛を元に戻し、ret を返すプログラム
```

▼リスト12　Lispのトップレベルループ

```
while (true){
  try {
    System.out.print("> ");  // プロンプト表示
    T sexp = Reader.read();  // リーダ
    if (sexp == Symbol.symbolQuit) break; // quitと入力されればLisp終了
    T ret = Eval.eval(sexp); // 評価
    System.out.println(ret); // toString (プリンタ) が呼ばれる
  } catch (Exception e){
    System.out.println(e.getMessage());
}
```

▼図10　トップレベルの使用例

```
Welcome to SDLisp! (2017-4-4)
> Copyright (C) GOMI Hiroshi 2017.
> Type quit and hit Enter for leaving SDLisp.
> (+ 10 20)
30
> (cons 1 '(2 3))
(1 2 3)
> (defun fact (n) (if (< n 1) 1 (* n (fact (- n 1)))))
FACT
> (fact 10)
3628800
> (defun tarai (x y z) (if (<= x y) y (tarai (tarai (- x 1) y z) (tarai (- y 1)
z x) (tarai (- z 1) x y)) ))
TARAI
> (tarai 12 6 0)
12
>
```



速攻で仕事を片づける
CLI力をマスターせよ!

シェル芸人からの挑戦状

今回のシェル芸人 今泉 光之、上田 隆一、山田 泰宏、田代 勝也、eban、中村 壮一　編者 上田 隆一

第1回 ファイルとファイルシステムの操作

連載開始にあたって

この連載は、本誌2017年1月号で実施した「シェル30本ノック」を毎月4〜5本ノックにして、みなさんと一緒に苦しんで、いや、楽しんで勉強していこうというものです。「シェル」と言っても、単にシェルのことだけをやるわけではありません。普段、我々はシステムの設定からデータの管理、プログラミング、原稿書きなど、シェルの上でコマンドを駆使してさまざまな仕事をしています。本連載では、その中からテーマを決めて、簡単(主観です)なものから難しいものまで順番に問題を出し、解答例を示していきます。出題、解答陣はこの記事の最後のコラムのメンバーです。

初回は、基本中の基本として、ファイルの整理やファイルシステムから5問選びました。解答例はUbuntu 16.04 LTSで動作確認しています。シェルはbash 4.3です。Linuxに限定するとよくツッコミが入るのですが、環境にこだわるレベルの人は解答例をアレンジできるはずですので、ここはシェルを覚えたての人向けに、ベタな環境を選びました。コマンドもLinux標準のGNU版を前提にします。標準でインストールされていないコマンドも紹介がてらガンガン使用していきます。また、aptで簡単に入るコマンドはインストールの指示がない場合があるのでご注意ください。

問題1 先週のファイル

初級 ★

出題、解答、解説：今泉

システム管理などをしていると、特定の日時に更新されたファイルを抽出したい場合があると思います。そこで、カレントディレクトリ以下で先週更新されたファイルを抽出してください。ただし、1週間の開始を日曜からとします。

準備

問題を解く際の検証ファイルを作成する方法を案内します。作業用の一時ディレクトリを作成し、**図1-1**のワンライナーを実行してください。今日の0時から400時間前までの1時間ごと

▼図1-1　検証ファイルを作成する

```
$ seq -f "$(date +%F) %g hour" 0 -1 -400 | date -f - '+touch -t %Y%m%d%H%M %F_%T' | sh
```

のファイルが作成されます。

解答

図1-2のコマンドで抽出できます。findは指定されたディレクトリ以下のファイルを列挙するコマンドで、`find .`でカレントディレクトリのファイルを列挙します。findには条件を与えることができますが、ここで指定した-mtimeオプションは「ファイルが最後に修正された時間」を意味します。引数の`$(())`、`$()`は括弧内でそれぞれ計算、コマンドの実行をして結果を引数に置き換えるためのものです。括弧内にある`date '+%w'`は、現在(本日)の曜日を数字で出力します。日曜が0、土曜が6になります。ということで、仮に本日が木曜ならこの部分は`-mtime -12 -mtime +4`となります。-12は通常「12×24時間前」を意味しますが、-daystartというオプションがつくと1日の開始、終了が起点となり、`-mtime +0`で前日以前のファイル、`-mtime -2`で前日以降のファイルが抽出されます[注1、注2]。これで木曜から見ると11日前の日曜から5日前の土曜以前のファイルが抽出され、問題の要件を満たすファイルのリストが得られます。

別解

ほかに執筆陣から挙がった解答例を**図1-3**に掲載します。別解の解説を細かくやると マニアックになり過ぎるので説明は割愛しますが、腕に覚えのある人や興味のある人は動かしたりオプションを調べたりすると良いでしょう。

注1) 法則性がよくわからないので必死こいてfindのソースを読んだところ、+<日数>だとその日の始まりが起点になり、-<日数>だとその日の終わりが起点になるようでしたが、manとも世の中の解説とも食い違いがあるのでまだ調査中です(上田談)。

注2) BSD系のfindでは、-daystartは不要です。

▼図1-2 解答

```
$ find . -daystart -mtime -$((8 + $(date '+%w'))) -mtime +$(date '+%w') -type f
```

▼図1-3 別解

```
別解1(上田) $ find . -print0 -type f | xargs -0 ls --full-time | awk -v s=$(date -d "$(( 7 + $(date +%w) )) days ago" "+%Y-%m-%d") -v e=$(date -d "$(( 1 + $(date +%w) )) days ago" "+%Y-%m-%d") '$6>=s && $6<=e'
別解2(山田) $ ls -alR --time-style='+@%Y%U@' | grep -E "@$(date -d 'last week' +%Y%U)@" | sed -r 's/.*@[0-9]{6}@ //'
```

問題2

ファイル名の長さの限界は?

出題、解答、解説:田代

初級 ★

Linuxのファイル名(パス情報なし)の最大長は255バイトのようです。ワンライナーで検証してください。

【注意】 この問題を解く際にたくさんのファイルが作成される可能性があります。作業用の一時ディレクトリを作成し、その中で作業しましょう。

解答

ファイル名の最大長を検証するやり方として、1バイトずつ長いファイル名のファイルを繰り返し作成する方法が考えられます。長さ1バイトから愚直に試し、ファイル作成に失敗するまで繰り返し試みる解答を**図2-1**に示します。

まずは**図2-2**のように、1バイトずつ長くした

▼図2-1 解答

```
$ yes | awk '{printf NR" ";for(i=1;i<=NR;i++) printf "@";print ""}' |
while read n s; do touch $s 2> /dev/null && echo $n && rm $s || break; done | tail -n 1
```

文字列を連続して作成します。awkで行数を表す特殊変数NRを使い、行数分の長さだけ@を連結した文字列を作成してファイル名として利用します。

▼図2-2　文字列を作る

```
$ yes | awk '{printf NR" "; ↵
for(i=1;i<=NR;i++) printf "@";print ""}'
1 @
2 @@
3 @@@
4 @@@@
（以下略。Ctrl+Cで止める）
```

▼図2-3　ファイルを作成し、成功したら文字列長を表示

```
$ （図2-2のコマンド） | ↵
while read n s; do touch $s 2> /dev/null ↵
&& echo $n && rm $s || break; done
1
2
（..略..）
254
255
```

この出力をwhileループで読み込んで繰り返し処理を行います(**図2-3**)。readコマンドでシェル変数nに文字列長、シェル変数sにファイル名の文字列を代入します。touchコマンドでファイルを作成し、ファイルの作成に成功したら文字列長を表示してファイルを削除します。ファイルの作成に失敗したらbreak文でwhileループを終了するので、最終行に表示された文字列の長さがファイル名の最大長になります。

doneのあとにパイプで`tail -n 1`(最後の行だけ出力)をつなげると、**図2-1**の解答例になります。

別解

図2-4に執筆陣が挙げた別解を掲載します。この問題はトリッキーなものからPythonを使う反則(？)までいろいろ挙がりました。自信のある方は打ち込んでいただければと。

▼図2-4　別解

```
別解(eban) $ for i in {300..1}; do f=$(printf "%0*d" $i); (: > $f) 2> /dev/null ↵
&& { rm $f; echo $i; break; } done
別解(田代) $ yes | awk '{for(i=1;i<=NR;i++) printf "@";print ""}' | ↵
sed 's/.*/touch &; echo -n @*@ |wc -c; rm &;/' | bash -e 2> /dev/null | tail -n 1
別解(上田) $ python -c 'print "a"*255;print "b"*256' | xargs touch
別解(山田、田代) $ sed -nr ':a s/^/@/;p;/^.{300}/!t a' <<<'' | ↵
sed -n 's/.*/touch &; rm &/e' 2>&1 | grep touch | head -n 1 | tr -dc @ | wc -c | xargs -I{} expr {} - 1
別解(山田、田代) $ sed -nr ':a s/^/@/;p;/^.{256}/!t a' <<<'' | sed 's/^/w/' | sed -f - 2>&1 | ↵
grep -o @ | grep -c . | xargs -I{} expr {} - 1
```

問題3

中身が同じファイルの検索

出題、解答：上田
解説：中村

初級 ★

/etc下で、中身が同じファイルを探してください。シンボリックリンクやハードリンクに関しては抽出してもしなくても良いことにします。

解答

複数のファイルの内容が同じかどうかを確認するために、私たちはよくmd5sumコマンドを利用します。md5sumコマンドは、ファイルの中身を入力にして何桁かの16進数を作ります。この16進数はMD5値やMD5ハッシュ値と呼ばれ、ファイルの中身が同じ場合は同じ値になります。したがって、ファイルの中身が同じ可能性があるかどうかを検出できます。

まず、/etc下のファイルに対してMD5値を求めてみましょう。**図3-1**のように1問目で使ったfindに、ファイルを列挙するための`-type f`オプションを付けてファイルを列挙し、xargsコマンドでパイプから渡ってくるファイル名に対してmd5sumを適用します。

次に、MD5値が同じファイルパスの組み合わせを抽出します。**図3-2**のように、awkの連想配列を活用すると、同じMD5値を持つファイルを1行に並べられます。awkの{a[$1]=a[$1] " "$2}で、aという連想配列へ1列目のMD5値をキーにして、2列目のファイル名を空白区切りで連結しています。END{for(k in a){print k, a[k]}}は全行の処理が終わってから実行される処理で、連想配列aのキーを取り出してキーと値を出力しています。

あとは3列以上ある行だけ選べばMD5値が重複しているファイルのリストができます。列数による抽出にもawkをよく使います。awkには列数を表す組み込み変数NFがあり、次のように書くとデータが2列より多い行を抽出できます。

```
awk 'NF>2'
```

以上を組み合わせると、**図3-3**のような解答例が得られます。このワンライナーは写真のファイルがたくさん入ったディレクトリで重複を探す場合などに応用が利きます。ただし、重複だと思っていたら片方がシンボリックリンクであったり、ごく稀(まれ)に違うデータから同じMD5値が得られたりする可能性があるので、片方を消去して整理したい場合には、さらに精査が必要です。

別解

図3-4に1つ挙げておきます。bashのglobstarという機能を使い、uniqコマンドで重複を調べるというものです。globstarについては本誌2017年7月号の第1特集「理論&応用でシェル力の幅を広げる」のP.47にも紹介されています。同記事にも説明がありますが、対象のファイル数が多くなると実行時間が長くなるため注意が必要です。

▼図3-1　ファイルひとつひとつにmd5sumを適用

```
$ sudo find /etc -type f | sudo xargs md5sum
a749ca975db772edde8499272dc78d12  /etc/init.d/postfix
9b73de9dd725586325690dd705d0a801  /etc/init.d/.depend.boot
c599894b3222405f5c419c7804db3102  /etc/init.d/.depend.start
(..略..)
```

▼図3-2　awkの連想配列を使って同じMD5値のファイルをまとめる

```
$ (図3-1のコマンド) | awk '{a[$1]=a[$1] " "$2}END{for(k in a){print k, a[k]}}'
(出力結果を一部抜粋)
a326c972f4f3d20e5f9e1b06eef4d620  /etc/pam.d/common-auth
ac1446cd28de7387e63388ad0ce833f2  /etc/rc5.d/README /etc/rc4.d/README /etc/rc3.d/README
/etc/rc2.d/README
2a3bc26e39035de74291c3a900a9797f  /etc/postfix/postfix-files
d6b276695157bde06a56ba1b2bc53670  /etc/python2.7/sitecustomize.py /etc/python3.5/sitecustomize.py
```

▼図3-3　解答

```
$ sudo find /etc -type f | sudo xargs md5sum |
awk '{a[$1]=a[$1] " "$2}END{for(k in a){print k, a[k]}}' | awk 'NF>2'
ac1446cd28de7387e63388ad0ce833f2  /etc/rc5.d/README /etc/rc4.d/README /etc/rc3.d/README
/etc/rc2.d/README
272913026300e7ae9b5e2d51f138e674  /etc/magic /etc/magic.mime
d7b46d3ee8cfc9bddb71d411a240e351  /etc/subgid- /etc/subuid-
(..略..)
```

▼図3-4　別解(中村)

```
$ shopt -s globstar  ←globstarを有効に
$ md5sum /etc/**/* 2> /dev/null | sort | awk '{ print $2, $1 }' | uniq -f 1 -D --all-repeated=separate
/etc/init.d/procps 021482ebab1024f5ed76e650e5191e8f
/etc/rcS.d/S02procps 021482ebab1024f5ed76e650e5191e8f

/etc/init.d/hwclock.sh 1ca5c0743fa797ffa364db95bb8d8d8e
/etc/rc0.d/K01hwclock.sh 1ca5c0743fa797ffa364db95bb8d8d8e
(..略..)
```

問題4

USBメモリのアンマウント

出題、解答、解説：山田

中級 ★★

フォーマットされたUSBメモリをPCにいくつか挿して、自動でマウントされなければマウントするワンライナーを考えてください。また、それらを全部アンマウントするワンライナーを考えてください。なお、USBメモリはFAT32でフォーマットされているとします。

準備

この問題に挑戦する方は、事前にデスクトップ環境の自動マウント機能を無効にしてください。Ubuntu 16.04をお使いの方は**図4-1**のコマンドでGNOMEの自動マウントを無効にできます。設定を戻す場合はfalseをtrueにして再度実行してください。

ヒント

GUIのデスクトップ環境が動作するLinuxマシンにフォーマット済みのUSBメモリを挿すと、自動的にマウントされて中身のファイルが閲覧できることがあります。これは自動マウント機能が有効になっているためで、たとえばUbuntu 16.04では特別な操作をしなくても、Nautilus（ノーチラス）という標準ファイルマネージャを開いてUSBメモリ内のファイルを閲覧できます。

対して、このようなデスクトップ環境のないLinuxでは基本的に、USBメモリを挿しても中身のファイルに自動的にアクセスできる状況にはなりません。そのため、CLI端末でいくつか操作をする必要があります。

USBメモリを挿すと、/dev/sd［アルファベット1文字］［数字］のようなファイルが作成されます。このファイルはデバイスファイルと呼ばれ、このファイルを利用することでデバイスドライバを経由して、デバイスのあらゆる操作を行えます。［数字］の部分がパーティションを表し、単一のデバイスに複数のパーティションが存在すると、数字が増えていきます。少なくともUbuntuでは、ハードディスクやUSBメモリなどの外部ストレージをマシンに接続するとこのようなファイルが出現します[注3]。

たとえば、USBメモリを挿して/dev/sdb1というデバイスが作成されたとします。このデバイスはmountコマンドで次のようにマウントできます。

```
$ sudo mount /dev/sdb1 /mnt/myusb
```

この操作のために、事前にマウントのためのディレクトリ（上の例は/mnt/myusbですが何でも良いです）を用意する必要があります。コマンド実行後にそのディレクトリにアクセスすると、USBメモリのファイルの中身が確認できます。

```
$ ls /mnt/myusb
aaa.txt  bbb.txt
```

解答：全部マウントする

まずはlsblkコマンドというデバイスの情報を表示するコマンドで、OSに認識されているデバイスの状態を確認してみましょう。また、-oオプションを付けると、出力する項目を選択できます。**図4-2**の例では、デバイスファイルの名前（KNAME）とファイルシステム名（FSTYPE）、そしてマウントポイント（MOUNTPOINT）

▼図4-1　自動マウント機能を無効にする

```
$ gsettings set org.gnome.desktop.media-handling automount false
```

注3）本来「/dev/sd……」はSCSIで接続されたディスクのためのデバイスファイルだったようですが、最近のLinuxディストリビューションではHDDに加え、USBメモリをはじめとしたデバイスも、汎用的にディスクドライブに割り当てられることがよくあります。

を表示しています。

ちなみに、**図4-2**ではUSBメモリを2本挿しており、sdb1とsdc1のパーティションがそれぞれ対応しています。FSTYPEにvfatという表記がありますが、これはFAT32で長いファイル名を使えるようにしたものです。ここではFAT32と事実上同じものと考えてください。すでにマウントされているデバイス(MOUNTPOINTの項目に値のあるデバイス)や、マウントができない性質のもの(sdaのようにパーティションを指さないデバイスファイルや、スワップ領域など)は無視してしまいましょう。スワップ領域についてはマウントポイントが[SWAP]となり、きちんとMOUNTPOINTの項目に値が入っています[注4]。そのため、MOUNTPOINTの項目が空(フィールド数が2つのもの)かつ、FAT32のファイルシステムのものを**図4-3**のように抽出すれば、「まだマウントされていないFAT32のデバイス」を抽出できそうですね。

次にこの結果からawkで、mountを使ったコマンドを組み立てます(**図4-4**)。mkdirでマウントポイントを/mnt/disk[数字]以下に作成し、mountの-tオプションでファイルシステム(vfat)を指定しつつ、マウントするコマンドを文字列として組み立てます。

作成した文字列をshコマンドに標準入力として与えると、そのコマンドを実行してくれます。その際、sudo shという具合に、sudoでコマンドを管理者権限で実行すれば、マウントがされていないUSBデバイスをマウントできます。

注4) ちなみにFSTYPEの個所はLVMの場合はLVM_memberとなったり、ZFSの場合はzfs_memberとなったりします。

▼図4-2 lsblkでデバイスの状態を確認

```
$ lsblk -o KNAME,FSTYPE,MOUNTPOINT
KNAME FSTYPE MOUNTPOINT
sda
sda1  ext4   /
sda2  vfat   /share
sda5  swap   [SWAP]
sdb
sdb1  vfat
sdc
sdc1  vfat
```

▼図4-3 未マウントのFAT32デバイスを抽出

```
$ (図4-2のコマンド) ⏎
| awk 'NF==2' | grep fat
sdb1  vfat
sdc1  vfat
```

▼図4-6 USBメモリをすべてアンマウント

```
$ df -l | grep '/mnt/disk' ⏎
| awk '{print "umount "$1}' | sudo sh
下記のようなコマンドが実行される
umount /mnt/disk1
umount /mnt/disk2
```

解答:全部アンマウントする

マウントされているデバイスに対応したデバイスファイルをumountコマンドで指定すると、そのデバイスをアンマウントできます。たとえばこのような具合です。

```
$ sudo umount /dev/sdb1
```

図4-4の解答例では/mnt/disk[数字]以下にマウントするようコマンドを組み立てました。そこでまずdf -lコマンドでローカルにマウントされているデバイスのデバイスファイル名を列挙します(**図4-5**)。

▼図4-4 解答

```
$ lsblk -o KNAME,FSTYPE,MOUNTPOINT ⏎
| awk 'NF==2' | grep fat | awk '{P="/mnt/disk"NR; print ⏎
"mkdir -p "P" ;mount -t vfat /dev/"$1,P}' ⏎
| sudo sh
最終的に下記の2行のコマンドが実行される
mkdir -p /mnt/disk1 ;mount -t vfat /dev/sdb1 /mnt/disk1
mkdir -p /mnt/disk2 ;mount -t vfat /dev/sdc1 /mnt/disk2

$ ls /mnt/disk1
aaa.txt  bbb.txt
```

▼図4-5 マウントされているデバイスを列挙する

```
$ df -l
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             1949848        0   1949848   0% /dev
tmpfs             393868    11152    382716   3% /run
/dev/sda1      241980824 6180900 223484944   3% /
(..略..)
/dev/sdb1       15727616 3619856  12107760  24% /mnt/disk1
/dev/sdc1       31247904 1455616  29792288   5% /mnt/disk2
```

そこから、USBメモリのマウントポイントとなっているディレクトリ名を含んだ行をgrepで抜き出し、さらにデバイスファイル名を抜き出してumountでアンマウントします(**図4-6**)。

問題5

謎のデータの調査

出題、解答、解説:上田

上級 ★★★

データ用リポジトリ(https://github.com/ryuichiueda/ShellGeiData)のファイルsd201709/enigma内のデータ(**図5-1**)からメッセージを読み取ってください。なお、このデータはバイナリがbase64という方式でエンコードされた文字列です。暗号解読に挑戦してもらうためではなく、バイナリを可読にするためにエンコードしました。

【注意】素性の不明なデータをむやみに解析しようとするとPC内のデータが壊れたり何かに感染したりするので注意しましょう。

▼図5-1 解読するコード(ファイル「enigma」をGitHub上で参照)

```
7 lines (6 sloc)  422 Bytes
1  H4sICDM8MlkAA2EA7dQxaxNxFADwf0KlkNLiJDj1T+lQlwMzOxjQDgWTUNM6FIQrveiRmAu5WwId
2  /Ah+DnERuhWK0DnfolsWx062hyDiLJjB348H7/He8uDBWz779GE0LJNhWoXmWiM0Q/NFuG2EGJ6H
3  Xz6GJ4cP+l+7vdjtvHoZa/udwdN2nbe2r07Ov+x8qzaOL7Yu18Pi4dvl9/bN4tHi8fLH4H1exjom
4  RRXTeFoUVXo6zuJZXo6SGPvjLC2zmE/KbPbHfDguptN5TCdnm63pLCvLupzHUTaPVRGrWT15l+aT
5  mCRJ3GwF/sbR59u7u1UvwQq5///tdW9/8KZz+POr/+7u7V4frGwlAAAAAAAAAAAAAAAAAAAAAAA
6  AAAAAAAAAAAAAAAAIB/6h5OzIvAAIgAAA==
```

解答

まず、データをデコードしましょう。**図5-2**のように、base64コマンドでデコードしてfileコマンドで何のデータか調べ、その結果gzipで圧縮されたデータだとわかるので展開、という流れになります。

展開したら再びfileで確認することになります。**図5-3**のように、問題4で出てきたFATのイメージ(HDDなどのストレージや、ストレージの1パーティションのデータを丸ごとファイルにしたもの)とわかります。

イメージの中身をどうやって読むかというところですが、Linuxにはループバックデバイスというものがあって、これを使うと読めるようになります。ループバックデバイスは/dev/の下に、**図5-4**のように、loop0、1、2……とあるデバイスファイルのことです。

これの1つにファイルを結び付けてマウントすると、そのファイルをストレージのように扱えます。このお題での手順は**図5-5**のようになります。

dfの出力で./tmpが/dev/loop0というデバイスファイルと結び付けられているのがわかります。OSは/dev/loop0を通じてHDDやSSD、USBメモリと同じ方法でイメージにアクセスすることになります。

▼図5-2 データの復元

```
↓何が出てくるかわからないのでファイルにリダイレクトする
$ cat enigma | base64 -d > hoge
$ file hoge
hoge: gzip compressed data, was "a", last modified: Sat Jun  3 04:33:55 2017, from Unix
$ mv hoge a.gz && gunzip a.gz
```

▼図5-3 デコードして得たファイルの調査

```
$ file a
a: DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "mkfs.fat", sectors/cluster 4, root entries 512, sectors 68 (volumes <=32 MB) , Media descriptor 0xf8, sectors/FAT 1, sectors/track 32, heads 64, serial number 0xb1500552, unlabeled, FAT (12 bit)
↓FATとあるのでfdiskで調べてみましょう
$ fdisk -l a
Disk a: 34 KiB, 34816 bytes, 68 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

▼図5-4 ループバックデバイス

```
$ ls -l /dev/loop?
brw-rw---- 1 root disk 7, 0  6月 30 18:44 /dev/loop0
brw-rw---- 1 root disk 7, 1  5月 21 11:16 /dev/loop1
(..略..)
brw-rw---- 1 root disk 7, 7  5月 21 11:16 /dev/loop7
```

余談ですが、この方法を使うと、OSのイメージファイルもマウントできるので、イメージファイルの中から何かを取り出したり、書き換えたりする場合に便利です。ただし、OSのイメージの場合、データが始まる位置を指定するなど、mountに追加のオプションが必要となります(ネット上で詳細な記事が探せます[注5])。

本題に戻ります。マウントしたら誰でもとりあえずlsコマンドを実行すると思いますが、softwareという空ファイルがあります。このファイル名が出題者からのメッセージとなります。

```
$ ls ./tmp/
software
$ sudo umount ./tmp   ←確認が終わったらアンマウント
```

注5) 2008-07-24"HDDイメージファイルをマウントして使う方法"、adsaria mood(http://d.hatena.ne.jp/adsaria/20080724/1216865687)など。

本当はSoftwareDesignという空ファイルを作ったのですが、実に20年ぶりにファイル名の8文字制限(そういう不自由な世界がかつてあった)に引っかかってしまい、softwareになってしまったのは内緒です。

最後に、手順だけですがこのenigmaというファイルの作り方を図5-6に示します。

終わりに

今回はファイルやファイルシステム関係の問題を5問こなしました。ファイル操作についてはもっとたくさん(ほぼ無限に)問題があるので、連載が長続きすれば、あらためて扱ってみようと考えています。次回は日付の計算の問題を扱う予定です。SD

▼図5-5 イメージをマウント

```
↓マウント先のディレクトリを作る
$ mkdir tmp
↓ファイル「a」をmsdos(FAT)で./tmp/にマウント
$ sudo mount -o loop -t msdos a ./tmp/
↓確認
$ df ./tmp/
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/loop0            16     0        16   0% (略) /tmp
```

▼図5-6 ファイル「enigma」の作り方

```
$ dd if=/dev/zero of=a bs=512 count=68
$ mkfs.fat a
$ mkdir hoge
$ sudo mount -o loop -t msdos a ./hoge/
$ sudo touch ./hoge/SoftwareDesign
$ sudo umount ./hoge
$ gzip a
$ base64 a.gz > enigma
```

COLUMN

本連載の出題・解答を担当するシェル芸人たち

・今泉 光之(いまいずみ みつゆき) Twitter @bsdhack
USP友の会の幹事をさせていただいています。古き良きUNIXライフが好きなおじさん。

・上田 隆一(うえだ りゅういち) Twitter @ryuichiueda
「シェル芸」が連載のタイトルになってしまって大丈夫なのかと心配する気の小ささ。千葉工業大学付近に出没。

・山田 泰宏(やまだ やすひろ) Twitter @grethlen
端末で遊ぶのが大好きなキュアエンジニア(プリキュアが好きなエンジニア)。「Cureutils」や「tmux-xpanes」という便利なCLIツールを公開しています。

・田代 勝也(たしろ かつや) Twitter @papiron
シェル芸勉強会福岡サテライト担当。プログラミングは対話的CLIをゆるふわに雰囲気で叩く感覚派。ターミナル内で生きる似非(えせ)Macユーザ。

・eban Twitter @eban
使えるものは何でも使う。最近jqで解けない問題はそれほどないと気づいた。

・中村 壮一(なかむら そういち) Twitter @kunst1080
シェル芸勉強会の大阪サテライトをやっています。南ことり推しのラブライバー。最近のマイブームはDockerで直接startxすること。

RDBアンチパターン

PostgreSQLとMySQLの失敗と対策

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

第5回 フラグの闇

本連載では、開発の現場で発生しやすいリレーショナルデータベース（RDB）全般の問題をRDBアンチパターンとして紹介していきます。今回のアンチパターンの主人公は、ブログサービスのDB設計を任されたIさんと、そのDB設計に苦しむアプリ開発者のAさん。

前回は「効かないINDEX」と題して、RDBのチューニングのコアとも言えるINDEXについて説明しました。今回は「フラグの闇」として、読者のみなさんも現場で見たことがあるでしょう事例を交えながら、RDBの設計について説明します。

フラグの闇

第5回目では、DB設計時についやってしまいがちなフラグのアンチパターンについてお話します。具体的な事例を交えながら、フラグの取り扱いがなぜ難しいかを説明します。

事の始まり

Iさんは、新規ブログサービスのDB設計を任されました。

Iさん：この会員削除機能、物理削除だと、あとから戻してくれって言われたときに対応できないな。同僚のNさんに相談してみよう。

Nさん：会員削除？　削除系は事故が怖いから、いつも全部のテーブルに削除フラグを付けてるよ。

Iさん：なるほど！　確かに便利そうだ。今回はそれを採用しよう。

時は開発段階に入った数ヵ月後へ。Iさんの DB設計を確認中のアプリ開発者Aさん。

Aさん：あぁ、これ全部のテーブルに削除フラグがついてるじゃん（**リスト1**）。これだと、blogテーブルに削除されたデータが混在していて、どれが正しいデータのなのかわからない……。確かに過去の編集履歴も持ってるけど、有効なブログを取り出すのにこんなSQL（**リスト2**）を書くのか……。

さらに悩むAさん。

▼リスト1　削除フラグ満載のDB設計

```
-- 削除の手順
UPDATE blog
SET blog.delete_flag = 1
WHERE blog.id = ?

-- 既存のブログを編集した場合
-- (1)
UPDATE blog
SET blog.delete_flag = 1
WHERE blog.id = ?

-- (2)
INSERT INTO blog
 ...(略)...
```

▼リスト2　有効なblog（ブログ）を取り出すためのクエリ

```
SELECT
  *
FROM
  blog
  INNER JOIN users
    ON users.id = blog.user_id
    AND users.delete_flag = 0
  INNER JOIN customers ←usersをJOINして確認
    ON customers.id = users.customer_id
    AND customers.delete_flag = 0
  INNER JOIN category ←usersをJOINして確認
    ON blog.category_id = category.id
    AND category.delete_flag = 0
WHERE
  blog.delete_flag = 0 JOIN
```

Aさん：これ、削除とか更新のトランザクションをしっかりしないと、データが重複しそう。しかも仕様変更でテーブルが追加されるたびにクエリにJOINが増えていく……。UNIQUE制約もdelete_flagのせいで付けられないし、有効なデータを取り出すクエリは複雑だし、INDEXも効かない……。

今回のアンチパターン

今回のアンチパターンは、テーブルに削除という「状態」を持たせてしまったことです。これは削除フラグ以外にも、課金状態（与信の有無）やユーザ状態（退会・休会など）のケースでも起こり得ます。では、なぜこのようにテーブルに状態を持たせてはいけないのかを、削除フラグを題材に解説していきます。

とりあえず削除フラグ

このアンチパターンは、「フラグの闇」の中でも「とりあえず削除フラグ」と呼ばれ、現場でもよく見られるアンチパターンです。削除フラグは次のようなRDBの問題を多く含んでいます。

クエリの複雑化

値を取り出すときのクエリが次のように肥大化していきます。

```
-- blogを取ってきたい場合
SELECT
  *
FROM
  blog
WHERE
  blog.delete_flag = 0
  ↑blogの削除フラグを見る
```

さらに更新者アカウントが削除された場合もカバーするには、次のようなクエリとなります。

```
-- 有効なアカウントのblogを取ってきたい場合
SELECT
  *
FROM
  blog
  INNER JOIN users ←usersをJOINして確認
    ON users.id = blog.user_id
    AND users.delete_flag = 0
WHERE
  blog.delete_flag = 0
```

加えて、そのユーザの所属やblogのカテゴリが削除された場合もカバーするには、最初の**リスト2**のようなクエリになります。

このように関連するテーブルすべてに影響を与える形になり、アプリケーション開発時のコストが増大します。

これは、「あのデータ取るには○○と□□と△△をJOINしてWHERIしてSELECT句にASで別名を付けて……」のようなクエリを書かなければならない事態が頻発し、とくにアプリケーションの仕様変更時などにおいては影響範囲が広く、バグの温床となります。

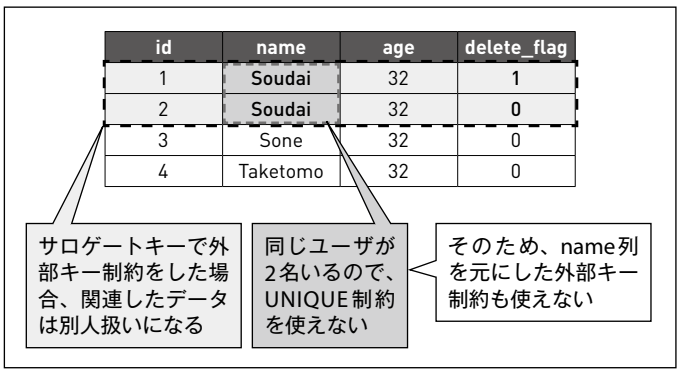
UNIQUE制約が使えない

削除フラグを使うと**図1**のように、name列に対してUNIQUE制約を使うことができません。

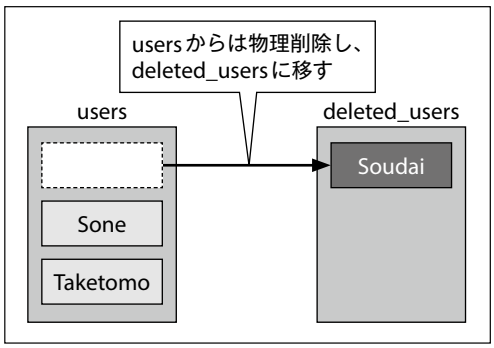
UNIQUE制約が使えないデメリットとしては、次の3点があります。

・データの重複を防げない
・該当列に対して外部キー制約を利用できない
・外部キー制約を利用できないことでデータの関連性を担保できない

▼図1　UNIQUE制約が使えないDB設計



▼図2　「削除済み」のためのテーブル



これらによってデータの整合性が担保できず、アプリケーション側の参照や更新の動作で、思わぬバグを埋め込むことがあります。

✤ カーディナリティが低くなる問題

前回(本誌2017年8月号)の「効かないINDEX」でも説明しました「カーディナリティ」についても、削除フラグの問題が影響します。カーディナリティとは「列に格納されるデータの値にどのくらいの種類があるのか？」を表す指標です。重複が少ないデータはカーディナリティが高いことになり、INDEXをうまく利用できます。

削除フラグの多くは、未削除(delete_flag = 0)の状態であり、削除フラグの列ではデータが重複し、カーディナリティが低くなります。そのうえ検索時には必ずdelete_flagを含めなければならず、ボトルネックの理由になります。

このように、削除フラグのようなカーディナリティが低く、必ず参照で利用される列に対してINDEXを利用する場合は、複合INDEXにdelete_flagを指定するといったことが必要になり、データベースの運用がより複雑になります。

このアンチパターンのポイント

今回のアンチパターンは「テーブルに状態を持たせた」ことで発生しました。テーブルに状態を持たせるのは非常に危険です。

このアンチパターンを回避するには、「事実のみを保存する」ことが大切です。今回の例ですと、削除済み会員テーブルを作るといった方法があります(**図2**)。

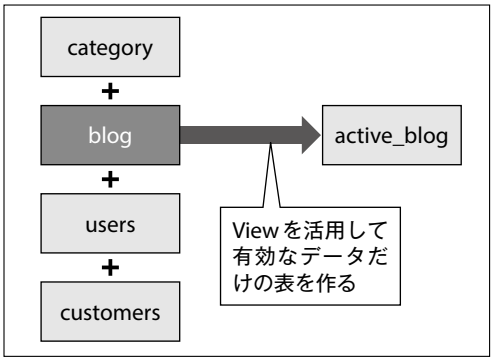
この場合、削除済みのデータを移す操作はアプリケーション側で実装するのも良いですが、RDBMSの機能である「トリガー」を使うと、アプリケーション側ではトランザクションを意識することなく、削除済みデータを作成できます。

しかし、トリガーはカラム追加などの仕様変更には弱いのでケース・バイ・ケースで使用してください。

すでに削除フラグがある場合は、Viewを活用する方法もあります(**図3、リスト3**)。Viewを活用することで、アプリケーションからの参照はシンプルになります。また、仕様変更の際もViewの定義を変更するだけで済みます。

しかしViewにアクセスする場合は、Viewで定義されたクエリが再実行されますので、高速化にはつながりません。高速化を意識するなら、

▼図3　Viewの活用



PostgreSQLならばマテリアライズド・ビュー、MySQLなら集計済みのテーブルを生成するのも良いでしょう。

ただしViewでは、更新が頻繁な場合は差分更新にはならないため、更新がボトルネックになります。

「状態」を持たせるのは絶対にだめ？

このように、テーブルに状態を持たせてしまった場合、その対処方法に銀の弾丸はありませんのでご注意ください。もし状態を持たせるなら、次のような点に十分に注意したうえで判断してください。

・対象のテーブルが小さく、INDEXが不要
・そのテーブルが関連するテーブルの親になることがなく、データを取得する際に頻繁にJOINの対象になることがない
・UNIQUE制約が不要で、外部キーでデータの整合性を担保する必要がない

これら条件を満たす場合、テーブルに状態を持たせることは許容されます。しかしテーブルに状態を持たせると、何らかの理由でそれをリファクタリングする必要が出た場合に困難を極めます。そのため、最初から正しく事実だけを持たせることが無難で大切です。

削除フラグを利用したくなるケース

削除フラグは使わなくて済むのであれば使わないほうが良いのですが、削除フラグを利用したくなる場面として、次のようなケースがあります。

▼リスト3　有効なblogを取り出すViewを作るクエリ

```
CREATE VIEW active_blog AS
SELECT
  *
FROM
  blog
  INNER JOIN users
    ON users.id = blog.user_id
    AND users.delete_flag = 0
  INNER JOIN customers ←usersをJOINして確認
    ON customers.id = users.customer_id
    AND customers.delete_flag = 0
  INNER JOIN category ←usersをJOINして確認
    ON blog.category_id = category.id
    AND category.delete_flag = 0
WHERE
  blog.delete_flag = 0 JOIN
```

・エンドユーザから見えなくしたいが、データは消したくない
・削除したデータを検索したい
・データを消さずにログに残したい
・操作を誤ってもなかったことにしたい
・削除してもすぐに元に戻したい

これらは確かに削除フラグで達成できますが、それ以外の設計でも対応できます。安易にテーブルに状態を持たせることなく設計することを心がけましょう。

このほかにも削除フラグについては多くの議論がされており、次の資料が非常に参考になりますのでぜひご確認ください。

・「とりあえず削除フラグ」[注1]
・「MySQLで論理削除と正しく付き合う方法」[注2]
・「PostgreSQLアンチパターン」[注3]

注1）URL https://www.slideshare.net/t_wada/ronsakucasual
注2）URL https://www.slideshare.net/yoku0825/mysql-52276506
注3）URL https://www.slideshare.net/SoudaiSone/postgre-sql-54919575

次回のRDBアンチパターン

今回のRDBアンチパターンはいかがでしたでしょうか？　フラグの闇はついやってしまいがちなアンチパターンです。しかし、ジワジワと時間とともに苦しみを生み出すアンチパターンです。もしみなさんの周りで起きている場合は、ぜひリファクタリングを検討してみてください。また、今から作る新規案件ではアンチパターンを作り込まないように気を付けてください。

次回は同じくRDBの苦手とするアンチパターンです。次回「ソートの依存」もお楽しみに！SD

RDB TIPS　フラグの闇はあとあと効いてくる

削除フラグ以外にも、フラグの闇としてよく見るのがstatusカラムです。たとえば、会員の状態を次のように持つケースをよく見ます。

- 入会手続き中（メール確認前）
- 入会済み
- 有料会員
- 休会中
- 退会

これらの状態を、たとえば1～5の数値型で持たせるケースがよく見受けられます。このケースは削除フラグ同様、取り出す際にWHERE句を利用したり、View側で表示のバグを防ぐためにif文でcheckを入れる必要が発生します。

次に多く実装で見られるのが、次のようなメールマガジンの送信ステータスです。

- プレビュー
- 配信予約
- 配信中
- 配信済み

このケースでは、2つの問題があります。

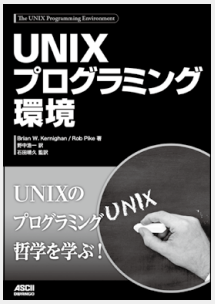
1つめはパフォーマンスの問題です。頻繁にメルマガを送る場合、送信済みのメールも対象のテーブルに残るためテーブルが肥大化していき、将来的にパフォーマンスのボトルネックになります。

2つめはトランザクションの問題です。設計によっては、メール送信中は配信が重複しないように、メールの送信リストに対して排他的な行ロックを取って管理しなければなりません（**リストA**）。テーブルに状態を持たせている場合に大量のメルマガを送信すると、長時間ロックを取ることになります。

フラグの問題は、サービスやデータが小さい場合や並列処理が少ない場合には、問題が顕在化しにくいです。しかし問題が顕在化したときには、手を付けられないほど大きな問題になっていることが多い、厄介なアンチパターンです。だからこそ、初期の段階で早めに対策することが非常に大切です。

▼リストA　mail_listにロックをかける

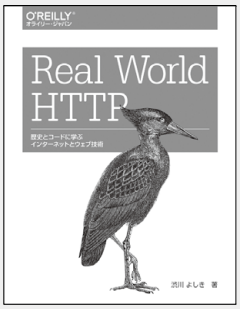
```
SELECT
  *
FROM
  mail_list
INNER JOIN
  users
ON mail_list.user_id = users.id
WHERE ....
FOR UPDATE
```



UNIX プログラミング環境

Brian Kernighan、Rob Pike 著、
石田 晴久 監訳、野中 浩一 訳
A5判／480ページ
3,200円＋税
アスキードワンゴ
ISBN＝978-4-04-893057-4

UNIXの使い方を初歩から応用までとことん解説した1冊だ。ログイン・基本コマンドから、ファイルシステム、sedやawkを使ったフィルタ、シェル（sh）、標準入出力を使ったC言語のプログラミング、システムコール、make・lexといったプログラム開発ツール、そして変わり種では、troffコマンドを使った文書作成まで、個々の小さなツールを組み合わせて大きな効果を発揮させる「UNIX哲学」を下敷きに解説していく。これらの知識はもちろん、コマンド名などを適宜読み替えることで、Linuxでの操作にも十分応用できる。ちなみに「UNIX」という言葉の由来だが、本書まえがきによると、開発者のケネス・レイン・トンプソンとデニス・リッチーがそれ以前に開発に関わっていたOS「Multics」から造語されたとのことだ。

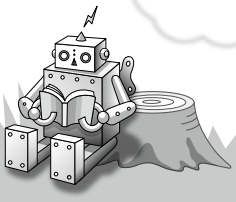


Real World HTTP

渋川 よしき 著
B5変型判／360ページ
3,400円＋税
オライリー・ジャパン
ISBN＝978-4-87311-804-8

HTTPはテキストベースの通信で仕様もシンプルなため、簡単な通信クライアントを使って学べるのが利点だ。本書もその利点を活かして、curlコマンドでデータを送受信したり、Goで簡単なHTTPクライアントを実装したりしながら、HTTPについて解説する。HTTP/0.9からHTTP/2まで順を追って解説されており、各仕様の目的や意味を確認しながら読み進められる。シンプルとはいえHTTPの仕様はかなりの量がある。それを一気に学ぶのはたいへんだし、現在の仕様を読むだけでは、なぜこうなっているのかの経緯はわからない。本書はその経緯をRFCや時代背景の中から丁寧に拾い、解説してくれていると感じた。副次的にcurlやGoのWeb関連のAPIの使い方も学べるので、Web開発の実務で役立つ情報が多く得られそうだ。

SD BOOK REVIEW



現場で役立つ システム設計の原則

増田 亨 著
A5判／320ページ
2,940円＋税
技術評論社
ISBN＝978-4-7741-9087-7

本誌ではオブジェクト指向の記事でお馴染みの増田亨さんによる、可読性が高く、変更しやすいシステムの作り方を解説した本である。「変数にはわかりやすい名前を付ける」といった部品レベルの原則から、「メソッドをロジックの置き場所にする」などの部品を組み合わせたアプリ全体の設計に関する規則まで、扱う項目の幅は広い。業務の関心事とコードを直接対応させる「ドメインモデル」についてもページを多く取って解説しているほか、これらのルールに則って書いたアプリとうまく連携するデータベース、GUIの設計についても言及している。使用言語はJava、使用フレームワークはSpringとなっているが、コード例はそれほど多くはなく、考え方や方向性の解説がメインなので、Java以外の言語のユーザでも応用が利くだろう。

IBM Bluemix クラウド開発入門

常田 秀明、水津 幸太、大島 騎頼 著、Bluemix User Group 監修
B5変型判／288ページ
2,800円＋税
技術評論社
ISBN＝978-4-7741-9084-6

さまざまなクラウドサービスが脚光を浴びている中、IBMのクラウド「Bluemix」は人工拡張知能Watsonとの連携を武器にPaaS系サービスのトップになるべく普及を図っているのは、皆さんご存じだろう。

本書は、Bluemixの成り立ちから基本的なサービス利用方法を解説しつつ、一般的なWebアプリの開発方法を紹介する。さらにRaspberry Pi＋IoTの組み合わせにも触れ、Watson APIとの連携方法まで言及しているのが特徴だ。BluemixはCloudFoundryを基盤にサービスを提供しているが、このメリットはIaaSのように低いレイヤーからすべて管理するのではなく、ビジネスを推進するためのアプリ開発にすべてを注力できることだ。そこに今後選ぶべきクラウドの形があるようだ。

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18回 Android 8.0の開発環境Android Studio 3.0

Androidは世界で出荷される約9割のスマートフォンに搭載される標準OSです※。そのため、多くのAndroidアプリが開発され続けており、そして多くのエンジニアが活躍しています。Androidで広がる新しい技術に魅了されたエンジニアが集うコミュニティもあり、そこでは自分が愉しむための技術を見つけては発信しています。その技術の一幕をここで紹介します。

※Gartner Worldwide Smartphone Sales to End Users by Operating System in 3Q16

三宅 理（みやけ おさむ）
日本Androidの会 運営委員
日本Androidの会 埼玉支部
合同会社ソニックスタジオ

はじめに

Google I/O 2017でAndroid Studioの最新版3.0の発表がありました。これまで2.4と呼ばれていたバージョンだったものが一気に3.0になりました。これにより多数の機能が追加・改善されています。またAndroidの開発言語にKotlinが公式にサポートされるようになりました。今回はAndroid Studioのアップデート内容に焦点を当てて解説します。

Android Studioの最新ベータ版は執筆時点でAndroid Studio 3.0 Preview Canary 6（以下AS3）です。これを元に記事を書いているため、正式リリース時には変更されている個所があるかもしれません。その点はご了承ください。

Android Studio 3.0

Android Studioは、もともとJetbrains社のIntelliJ IDEAをベースにした統合開発環境です。AS3ではIntelliJ IDEAのバージョンアップに合わせて、パラメータヒント（**図1**）やセマンティックハイライト（メソッドのパラメータとローカル変数に独自の色を割り当てる）といった機能が追加されています。

今回Android 8.0の新機能に対応するための機能もいくつか追加されています。次からAS3で増えた機能を見てみましょう。

Kotlinの正式サポート

Google I/O 2017の発表で話題になったKotlin[注1]ですが、AS3で正式にサポートされるようになりました。以前はASにPluginを導入することでKotlinを使って開発できましたが、あくまで非公式でのサポートでした。

KotlinはJavaと同じようにJava仮想マシン上で動作します。そのためJavaが動く環境であれば動作する点が互換性とともに評価されています。お約束のHello WorldをKotlinで書いたときのソースコードは**リスト1**のようになります。

これを見るとKotlinの特徴がよくでています。パッケージ宣言はオプションだったり、文末の

注1）http://kotlinlang.org/

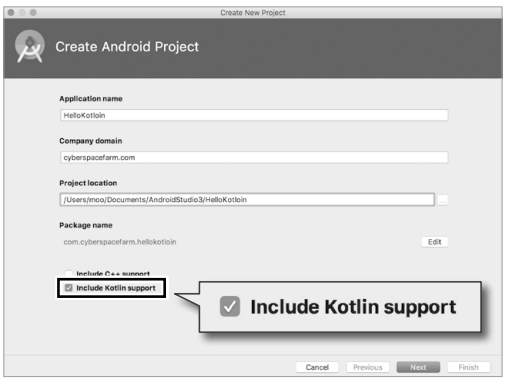
▼図1 パラメータヒント

```
fab.setOnClickListener(view -> {
        Snackbar.make(view,  text: "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction( text: "Action",  listener: null).show();
});
```

▼リスト1　Kotlinで書いたHello World

```
package hello

fun main(args: Array<String>) {
  println("Hello World!")
}
```

▼図2　新しいAndroid Project作成（Kotlin）



セミコロンは基本不要です。Javaのいいところを残しつつ、最近の言語の特徴を取り入れています。

Kotlinに対応したプロジェクトを作成してみましょう。

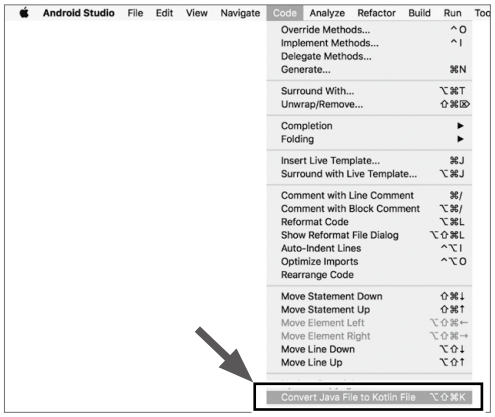
ポイントは、**図2**の画面で「Include Kotlin support」を選択することです。このチェックボックスをオンにすると、プロジェクトがKotlinをサポートするようにライブラリが追加されます。AS3ではKotlin 1.1.3がサポートされています。

KotlinはJavaと相互利用が可能なため、既存のプロジェクトに適用することもできます。作成済みのプロジェクトをKotlinのコードに変換するには、メニューの[Code]－[Convert Java File to Kotlin File]で変換ツール（**図3**）を使うことができます。

Java 8のサポート

AS3ではJava 8のライブラリと、いくつかの言語機能をサポートします。以前はJackツールチェーンを使ってJava 8の機能を利用していましたが不要になりました。また、Java 8を利

▼図3　JavaからKotlinへ変換



用しつつInstant Runを使うことが可能になったため、開発スピードを上げることができます。設定はモジュールのbuild.gradleを編集するか、モジュール設定画面で設定できます（**リスト2**）。

Java 8でサポートされる言語機能

Java 8でサポートされる言語機能は**表1**のとおりです。

Java 8でサポートされるAPIs

Java 8でサポートされるAPIsは次のとおりです（サポートされるのはAPI Level24以上）。

- java.lang.annotation.Repeatable
- AnnotatedElement.getAnnotationsByType（Class）
- java.util.stream
- java.lang.FunctionalInterface
- java.lang.reflect.Method.isDefault()
- java.util.function

Android Profiler

リアルタイムで、Activityの情報や画面のタッ

▼リスト2　モジュールのbuild.gradle編集

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
```

▼表1　Java 8でサポートされる言語機能

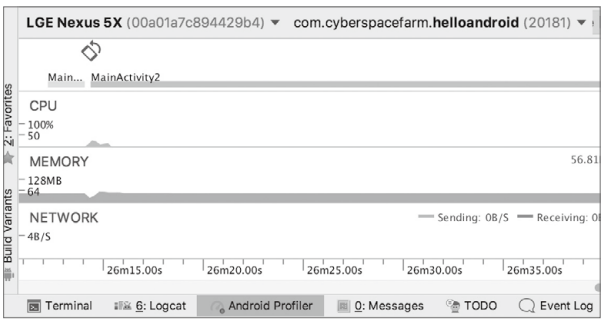
機　能	説　明
ラムダ式	1つのインターフェースに実装が必要なメソッドを1つだけ持つインターフェースである、関数型インターフェースを実装する際に利用
メソッド参照	関数型インターフェースの変数にメソッドそのものを代入する
タイプ・アノテーション(※)	デフォルトおよび静的インターフェースのメソッドインターフェースにメソッドを実装することが可能
反復アノテーション	同一名のアノテーションを複数指定することが可能

※ただし、ElementType TYPEはAPI Level24以下で対応。ElementType.TYPE_USE とElementType.TYPE_PARAMETERはAPI Level25以上で対応

プ、CPU、メモリ、ネットワークの利用状況を把握することができます。以前はAndroid Monitorと呼ばれていた機能を置き換えたものになります。

メニューの[View]－[Tool Windows]－[Android Profiler]で表示できます(**図4**)。Android Monitorで表示されていたlogcatは別ウィンドウとして定義されるようになりました。

▼図4　Android Profiler



Instant Appsのサポート

AS3は新しいリファクタリングモジュール化のアクションとApp Links Assistantが含まれているため、Instant Apps[注2]の実装ができます。Instant AppsはGoogle Playからアプリケーションをインストールしなくても自動的に実行され、使い終わったときに自動的に消去されるアプリです。HTMLのサイトからアプリに直接遷移できるメリットもありますし、ユーザにインストールしてもらう必要がないことが最大のメリットです。そのInstant Appsを作成できるようになりました。

APK Debugger

デバッグビルドしたAPKがあれば、APK単体でデバッグすることが可能になりました。メニューの[File]－[Profile or Debug APK]を選択するか、ようこそ画面で[Profile or debug APK]を選択することで利用可能です。APKを選択すると、APKの分析、プロファイルやデバッグが可能になります。リリースビルドしたAPKでは利用できません。

Device File Explorer

以前はAndroid Devcie MonitorのDDMS内にあったFile Explorerですが、AS本体で表示できるようになりました(**図5**)。

Android 8.0のサポート

AS3はAndroid O Developer Previewをサポートしています。Android OはAndroid 8.0としてリリース予定ですので、そのまま利用することができます。Android 8.0での新しい機能として、「Adaptive Icons」と「Downloadable fonts」があります。

注2) https://developer.android.com/topic/instant-apps/index.html

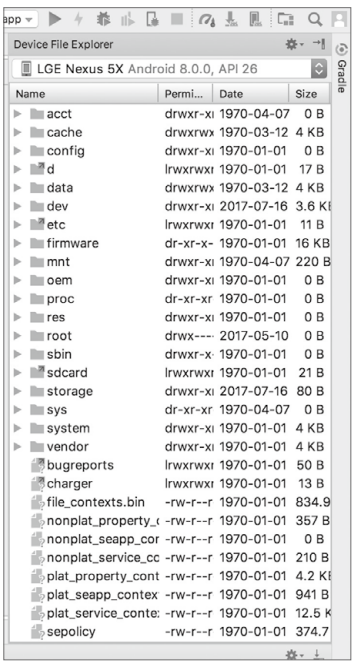
Adaptive Iconsはランチャアイコンの表示をHOMEアプリごとに適切な画像として表示します。AS3ではこの機能が含まれた状態でプロジェクトが作成でき、用意した画像にあわせて適切なアイコンを生成します。

Downloadable fontsはFont Providerを経由してフォントを利用する機能です。APK内にフォントを持つ必要がなくなるためアプリの容量を小さくできます。Google Fonts providerを利用する場合は、Google Play開発者サービスのバージョンが11以上である必要があります。

新しいAndroidエミュレータ機能

APIレベル24(Android 7.0)とAPIレベル26(Android 8.0)のシステムイメージにGoogle Playストアが含まれるようになりました。これでエミュレータでも、Google Playストアを含んだテストが可能になります。Google PlayストアはAndroid Virtual Deviceの設定画面でアップデートすることが可能なため、常に最新の状態でテストすることができます(**図6**)。

▼図5 Device File Explorer



▼図6 エミュレータ上のGoogle Playストア

そのほかの機能

ここまで紹介した機能以外にも、次のような強化が行われています。紙幅の都合で説明は割愛しますが、興味のある方はプレビュー版を試して確認してみてください。

- Android Things用の新しいテンプレート
- Layout Editorの改善
- Google Mavenリポジトリの追加
- APKアナライザの改善
- 新しいAndroid Plugin for Gradle

終わりに

AS3やAndroid 8.0は今年の秋ごろ正式版がリリースされます。筆者はAndroid 8.0に対応した開発機[注3]が日本で購入できるようにならないかと待っています。手元のNexus 5Xではそろそろ限界です。

Androidもまだまだ盛り上がっていきますのでみなさん楽しみましょう。SD

注3) 過去にはNexusシリーズやPixelがありました。

COLUMN

コミュニティ告知

日本Androidの会では2017年10月14日(土)に、「Android Bazaar and Conference 2017 in KAWASAKI」を開催します。https://japan-android-group.connpass.com/event/60022/を見てみてください。

また、本誌2017年7月号で記事にしたTangoのワーキンググループができました。JAG Tango Working Group(https://goo.gl/C4wJwj)まで。

一歩進んだ使い方のためのイロハ

Vimの細道

第21回

mattn
twitter:@mattn_jp

レジスタの使い方

今回はVimの「レジスタ」を深掘りします。普段のPCでの作業では、コピー&ペーストに使用するクリップボードしか意識していない方が多いかと思いますが、Vimではなんと9種類ものレジスタがあります。ひとつひとつ特徴がありますので、場面によって使い分けられるようにしましょう。

Vimのレジスタとは

Vimの特徴的な機能の1つにレジスタがあります。Vimをそれほど使いこなしていない方は、レジスタがいったいどんなものかご存じないかもしれません。

Vimを使いこなせるようになるために、レジスタは必要な通過点でもあります。ただしレジスタを知っているという人の中にも、漠然としか理解していなかったという方もいるかもしれません。Vimのレジスタとはいったい何なのでしょうか？　今回はVimのレジスタについて解説します。

Vimのレジスタは、情報を出し入れするための引き出しです。レジスタを知らなかったという人であっても、Vimを普通に使っているのであれば、実はレジスタを使っています。テキストを削除したり、コピーしたりしたその瞬間に、実はVimはレジスタに内容を格納しています。

レジスタの種類

レジスタは用途により9種類に分別されます。なお、Vimでレジスタを表記する際には"と「レジスタ名」を引っ付けて表します。

無名レジスタ(""")

無名レジスタには、dやc、sやxといったテキストの削除、またはyを実行した際のテキストの内容が格納されます。このレジスタは、ヤンクや削除などが実行されるたびに書き換えられます。Vimでyyをタイプした場合、行の内容が"レジスタ(表記："")に格納されます。ペースト時には、このレジスタ"の内容が参照されて貼り付けられます。

ちなみに、この先頭の"は、引き出しの場所を示しています。Vimではレジスタを指定してヤンクしたりペーストしたりできます。たとえば次のテキストでaというレジスタ名(名前付きレジスタについては後述)を指定して単語「world」をヤンクするには、worldの先頭に移動して"aywをタイプします。

```
hello world
```

レジスタに何が登録されているのかを確認するには、次のコマンドを実行します。

```
:registers
```

レジスタaの中身だけを確認するのであれば:registers aを実行します。

```
:registers a
```

```
--- Registers ---
"a   world
```

レジスタaを指定してペーストするには"apをタイプします。その際、行のすべてがヤンクされた場合には次の行にペーストされ、行の一部がヤンクされた場合にはカーソルの次の位置にペーストされます。

番号付きレジスタ("0〜"9)

0〜9の番号が付いたレジスタです。番号付きレジスタのうち、0にはヤンクされた内容が格納されます。1〜9は「行削除」された内容がローテーションされて保持されます。1回削除すると1に格納されます。2回削除すると1の内容が2にシフトされ、1に今回削除された内容が格納されます。テキストを編集していて、さっき削除した内容を取り出したいけれど、undoで戻るには遠過ぎると思ったときは、:registersを確認し、目当てのテキストが残っているのであれば、"5pなどと実行することで、5回前に削除したテキストを取り出すことができます。レジスタを多用する人の中にはこの番号付きレジスタのことを「ヤンクレジスタ」と呼ぶ人もいます。

小削除レジスタ("-)

1から9番までの番号付きレジスタは行の削除でしたが、1行内の削除は小削除レジスタ"-に格納されます。1行内で行われたd、x、c、sがこの操作にあたります。

名前付きレジスタ("a〜"z、"A〜"Z)

これまで説明してきたレジスタはすべて自動で格納されるものでしたが、名前付きレジスタは明示的に指定した場合にのみ格納され、使用するときは明示的に指定して参照します。

aからzのアルファベット文字でテキストが格納されます。人によって好みがあり、aからb、cと順に使う人もいれば、なぜかkを使い続けるといった人もいます。小文字のレジスタ名を使うと内容が上書きされ、大文字のレジスタ名を使うと、小文字のレジスタ内容への追加となります。たとえば、次のテキストの「private」の位置で"aywとタイプすると、

```
private final static int foo()
```

レジスタaに「private␣」(末尾にスペース1つ)が格納されます。続けてwを3回タイプして「int」の位置に移動し、"Ay$をタイプすると、レジスタaには「private int foo()」が格納されます。

単語を縦横無尽に移動するには

単語をヤンクしたり、選択して削除したりする場合に、Vimでは単語の取り得る範囲を把握しておく必要があります。

```
The pen is mightier than the sword.
```

この一文の「mightier」の「m」の位置にカーソルがあり、ywとタイプした場合、レジスタには次の単語までの空白スペースを含んだ「mightier␣」が格納されます。これは、英文やソースコードのように空白を区切りとする文章をコピー&ペーストする際に非常に役立ちます。なお日本語の場合、Vimは文字クラスを判別しており、平仮名と漢字の境界を単語の区切りと認識します。

たとえばこの「mightier」の「r」までをヤンクしたい場合には、yeとタイプします。eは単語の終わりを意味します。また、「mightier」の「g」の位置にカーソルがあった場合に「mightier␣」をヤンクしたい場合はyawをタイプします。さらに、前述のように空白を含まないようにヤンクしたい場合は、yiwをタイプします。

Vimでは、こういったしくみを「テキストオブジェクト」と呼びます。テキストオブジェクトに慣れていない方は、この方法を使うメリットがわからないかもしれません。「mightier」の「g」の位置にカーソルがあり、その単語を削除したいと思った場合、bdwとタイプするかと思います。実はこのbdwでは、b(単語の先頭に戻る)という操作とdw(単語を削除)という別々のオペレーションが実行されているのです。

▼表1　読み取り専用レジスタ

レジスタ	説明
"%	ファイル名レジスタ
"#	代替ファイル名レジスタ
".	ドットレジスタ(最後に挿入されたテキスト)
":	最終コマンドレジスタ

「mightier」の次の単語の「than」の「a」の位置で、.(最後のオペレーションを実行)をタイプした場合を考えてください。本来ならば「than」が消えてほしいと期待しますが、実際には「an」のみが削除されます。これはdwが、awやiwなどと同様に「モーション」であり、.の再生対象になるということなのです。

実際に上記のテキスト上で、「mightier」でdaw、さらに「than」のどこかで.をタイプしてみると、テキストオブジェクトがどのようなものか納得できると思います。テキストオブジェクトはとても便利ですので、また別の機会にさらに詳しく紹介したいと思います。

読み取り専用レジスタ("%、"#、".、":)

表1の4つの読み取り専用レジスタが用意されています。これらはおもに、インサートモードやコマンドモードで使用されます。

たとえば、現在開いているC言語のソースと同じ名称の「.hファイル」を開きたいのであれば、コマンドモードで:e␣(最後にスペース)までタイプし、Ctrl-r %をタイプします。すると、現在編集しているファイル名(例:main.c)が続けて挿入されるので、Backspaceを押して拡張子を変更し、main.hを開けば良いです。

ノーマルモードでの貼り付けの話題が出ましたが、インサートモードでの同様の操作は、たとえばクリップボードの中身を貼り付けるには、Ctrl-r *とタイプします。

式レジスタ("=)

このレジスタはかなり特徴的なレジスタで、通常のユーザが使うことはないかもしれません。このレジスタは、参照したタイミングで内容が評価されます。

式レジスタの内容は、ノーマルモードから"=をタイプすることで更新できます。たとえば"=strftime("%c", localtime())を入力すると、式レジスタ"=に現在の日付を取得する処理が格納されます。これ以降は、コマンドモードで:put =を実行するたびに現在の日付が挿入されます。

これとは別に、Vimには式挿入という機能があります。インサートモードでCtrl-r =をタイプしたあとに式を入力すると、その結果が挿入されます。ちょっとした計算結果(例:=350.0 / 12)を評価したい場合にも使えます。この式入力のプロンプトにて空が入力された際、この式レジスタが使われています。

選択レジスタ("*)とクリップボードレジスタ("+、"~)

Vimの選択レジスタ"*は少し特殊です。GUI版のVimで、設定ファイルのguioptionsにaが含まれている場合、マウスでテキストを選択すると選択レジスタに選択内容が格納されます。guioptionsにaが含まれていない場合は、クリップボードレジスタ"+と同様に動作します。

クリップボードレジスタ"+は、OSのクリップボードとして振る舞います。X11(X Window System)でのみ、テキストをドロップすることができ、最後にドロップしたテキストが"~に格納されます。

よく相談される問題で、「ヤンクしておいたテキストを使って、ビジュアル選択した個所を繰り返しペーストして置き換えていきたいと思っていたけれど、ペーストするとビジュアル選択した際のテキストでレジスタが置き換わってしまう」ということがあります。実はこれはVimの仕様で、詳しくは:help v_pを参照していただきたいのですが、簡単に言うと、ペーストされる際に選択されたテキストが削除されるため、その内容でレジスタが更新されてしまうために

起きる現象です。これを回避するには、ビジュアル選択した状態で単にpをタイプするのではなく、"0pをタイプします。

このしかけがなかなかおもしろいのです。ヤンクは"レジスタと番号付きレジスタの0レジスタを両方更新しますが、削除は"レジスタのみを更新します。ですので、0レジスタを使うことでペースト時に削除されたテキストで0レジスタが汚されることにはなりません。結果、0レジスタをペーストすれば繰り返しビジュアル選択した個所を置き換えられるのです。

ブラックホールレジスタ("_)

読み取り専用レジスタとは逆に、こちらは書き込み専用レジスタになります。

Vimのレジスタの扱いに慣れてくると、レジスタを箱のように取り扱えるようになります。テキストを削除し、別の個所に貼り付けるために格納しておくといったイメージで、さまざまなテクニックに応用できます。

しかしながら、Vimでは削除を行うとレジスタが更新されます。せっかくレジスタに格納しておいたのに、別の個所を削除したくなったとしても、レジスタが更新されてしまうので削除できないといった場面に遭遇します。そこで登場するのがブラックホールレジスタです。

"_dwのようにブラックホールレジスタを指定して削除することで、レジスタを何も汚さずに削除できます。このレジスタは、Vimプラグインの中で削除を行う場合に、ユーザのレジスタを汚さないための常套手段でもあります。

検索パターンレジスタ("/)

検索パターンレジスタには、最後に検索したパターンが格納されます。このレジスタは設定もでき、次のように実行することで、検索パターンをユーザ入力なしに設定することもできます。

```
:let @/ = '検索パターン'
```

検索パターンレジスタの応用例ですが、ユーザが入力した検索パターンを改変し、検索することができます。プラグインvim-fixedsearch[注1]とvim-prompter[注2]をインストールし、,/をタイプして検索すると、本来バックスラッシュでエスケープしないと検索できない\や()が、そのまま入力、検索できるようになります。しくみはいたって簡単で、ユーザが入力した正規表現パターン\[]を、

```
[\x5c][\x5b][\x5d]
```

に変換して検索パターンレジスタに設定しています。SD

注1）URL https://github.com/mattn/vim-fixedsearch
注2）URL https://github.com/mattn/vim-prompter

Vim豆知識

Neovimの:terminalコマンド

Vimの改良版であるNeovimには、:terminalというコマンドでVim内に端末を表示できる機能があります。これまで多くのVimユーザがこのNeovimの:terminalがほしいとvim-devで訴えてきました。そんな中、去る7月16日に、実験的なコードではありますがVimに端末機能が実装され始めました。まだほとんどの機能が動作していませんが、一度入れてしまったこれらコードが削除されることは恐らくありません。本記事が掲載されるころにも、まだ完成はしていないかもしれません。ですが今後、間違いなくVimに端末機能が実装されていくことになります。

また新しく、Vimの可能性が広がっていきますね。

書いて覚える Swift 入門

第29回 順序どおり問題を制する

Author 小飼弾（こがい だん）　twitter @dankogai

Sequenceプロトコル

前回はWWDC特集だけあって、Swift 4を含め現行のSwift 3ではまだ実行できないコードも多く登場したのですが、今回は順序どおりすべてSwift 3で実行できるコードを書いていきます。

そう、順序。Swiftで順序といえばSequenceプロトコル。ArrayもDictionaryもRangeもすべてこのSequenceプロトコルに準拠しています。

コンピュータ言語における順序の歴史

コンピュータプログラミングとはものごとを順序よく片付けるために存在するといっても過言ではなく、たいていのプログラミング言語にはそのための構文糖衣が用意されています。いにしえの行番号付きBASICですら、こんな感じに。

```
10 FOR N=1 TO 30
20 LET F$ = ""
30 IF N MOD 3 = 0 THEN LET F$ = "Fizz"
40 IF N MOD 5 = 0 THEN LET F$ = F$ + "Buzz"
50 IF F$ = "" THEN PRINT N ELSE PRINT F$
60 NEXT N
70 QUIT
```

なぜ構文糖衣かといえば、「順序よくやる」は単に「終わりが来るまで次をやる」のと同じことだからです。C言語はこのあたりはっきりしていて、

```
int i;
for (i = 0; i < 42; i++) {
  /* ... */
}
```

は、

```
int i = 0;
while (i < 42) {
  /* ... */
  i++;
}
```

と等価であることがはっきりとわかります。

しかしより現代的な言語では「順序よくやれ」という場合に「いつ終わるのか」「次とは何か」を省略して書くのが流儀となっています。たとえばPerlなら同様の繰り返しは、

```
for $i (0..41) {
  # ...
}
```

と、「0から41まで」という言い方で、「0から1つずつ増やしていって42だったら終了」という言い方を避けています。

Swiftも、かつてはC言語風のforループが存在しました。

```
for var i = 0; i < 42; i++ {
  // …
}
```

しかしこれはSwift 2で廃止され、

```
for i in 0..<42 {
  // …
}
```

という「0から42の手前まで」(あるいは0...41と書いて「0から41まで」)という書き方に統一されました。

なぜそうなったかといえば、そのほうが直感的で間違いが減るから。終了条件で<と<=を違えてしまうというのは、今でもしばしば脆弱性の原因となったりもします。

それでもSwift 1にはC言語風のforが残っていたのははなぜでしょう？　Sequenceプロトコルが未熟だったからです。このプロトコル、一時期はSequenceTypeに改名されていたりと言語設計者も迷った形跡が見られますが、Swift 3にいたってそのあたりの「ぶれ」もなくなり、Swift 4のSequenceもSwift 3と互換です。

Sequenceの正体

それではSwiftにおけるSequenceとはなんなのか？　「書いて覚える」という視点では、次の2つが等価であることさえ覚えておけば良いでしょう。

```
for i in s {
    doSomething(with:i)
}
```

```
var t = s.makeIterator()
while let i = t.next() {
    doSomething(with:i)
}
```

つまり、.makeIterator()という「イテレーター」を返すメソッドを持ち、そのイテレーターに対して.next()を呼ぶことで、順序よく値を取り出し、.next()がnilを返したらおしまい、というわけです。

それでは実際にIntをSequenceにしてしまいましょう。

```
public struct IntIterator : IteratorProtocol {
    var count:Int
    init(_ count: Int) {
        self.count = count
    }
    public mutating func next() -> Int? {
        if count == 0 {
            return nil
        } else {
            defer { count += count < 0 ? 1 : -1 }
            return count
        }
    }
}
extension Int:Sequence {
    public func makeIterator() -> IntIterator {
        return IntIterator(self)
    }
}
```

こんなので実際に動くのでしょうか？

```
for i in 4 {
    print(i)  // 4, 3, 2, 1
}
```

動きました！

しかし、そのためにIntIteratorというStructを作るのもなんだかめんどうです。この場合は次のようにしてまとめられます。

```
extension Int:Sequence,IteratorProtocol {
    public mutating func next() -> Int? {
        if self == 0 {
            return nil
        } else {
            defer { self += self < 0 ? 1 : -1 }
            return self
        }
    }
}
```

それでもC言語のforに比べてめんどくさそうに思えます。しかし、わざわざSequenceに準拠するだけの価値は確かにあるのです。こうしておくことで、.map()や.filter()や.reduce()が無料で手に入るのです。

```
42.map{$0}          // [42,41,…,1]
42.map{$0 % 2 == 0} // [42,40,…,2]
42.reduce(0,+)      // 903
```

Swiftにおいて型をSequenceに準拠させるということは、Rubyにおいてそのクラスを Enumerableにすることに相当します。Rubyにおける Enumerableの発展を見れば、Swiftで Sequenceに準拠させることがどれほどの利益をもたらすかをあらためて感じ取れるのではないでしょうか。

とはいえ、さすがにIntをSequeceにするのはやり過ぎでしょう。RubyでもIntクラスはEnumerableではないのですから。しかしRubyのIntには.timesメソッドが存在します。SwiftのIntに.timesは標準装備されていませんが、次のとおり簡単に追加できます。

```
extension Int {
    public var times:CountableRange<Int> {
        return 0..<self
    }
}
```

より実践的なSequence

Sequenceが真の威力を発揮するのは、単一の値ではなく複数の値をまとめて扱うときにあるのは、.map()、.filter()、.reduce()の例のとおりです。Swiftにはすでに標準でArrayを持っていますが、たとえば5,000兆個のデータをArrayに読み込んでというのはメモリが足りなさ過ぎるでしょう。そういう場合にもSequenceに準拠した型としてそれを実装すれば、少しずつストレージから読んで処理するのも「いつものやり方」で実現できます。

ここではLispやHaskellでよく用いられている片方向リストを実装してみましょう。こんな感じですか。

```
enum List<T> {
    case Nil
    indirect case Pair(head:T, tail:List<T>)
}
```

最低限のアクセサーと……、

```
extension List {
    var car:T? {
        get {
            switch self {
            case .Nil: return nil
            case let .Pair(v, _): return v
            }
        }
        set {
            switch self {
            case .Nil: return
            case let .Pair(_, l):
                self = List.Pair(head:newValue!, tail:l)
            }
        }
    }
    var cdr:List<T>? {
        get {
            switch self {
            case .Nil: return nil
            case let .Pair(_, l): return l
            }
        }
        set {
            switch self {
            case .Nil: return
            case let .Pair(v, _):
                self = List.Pair(head:v, tail:newValue!)
            }
        }
    }
}
```

……イニシャライザーを用意しておきます。

```
extension List {
    init(fromArray: [T]) {
        self = fromArray
          .reversed()
          .reduce(.Nil) {
              List.Pair(head: $1, tail: $0)
          }
    }
    init(_ values:T...) {
        self.init(fromArray:values)
    }
}
```

この状態で、

```
var l = List(0,1,2,3)
```

とすれば確かにlは0->1->2->3->Nilという具合に初期化されて、print(l)してみると、Pair(head: 0, tail: __lldb_expr_80.List<Swift.Int>.Pair(head: 1, tail: __lldb_expr_80.List<Swift.Int>.Pair(head: 2, tail: __lldb_expr_80.List<Swift.Int>.Pair(head: 3, tail: __lldb_expr_80.List<Swift.Int>.Nil)))) なんて感じになっているのですが、これをSequenceにするにはどうしたらよいのでしょう？ **先ほどの例のように**ListIteratorを追加しても良いのですが、実はSwiftには次のような簡単な方法も用意されています。

```
extension List : Sequence {
    public func makeIterator() -> AnyIterator<T> {
        var list = self
        return AnyIterator {
            let v = list.car
            if let l = list.cdr {
                list = l
            }
            return v
        }
    }
}
```

こうしてからfor v in l { print(v) }してみると、確かに値を頭から取れていることがわかります。

あとはこれを利用して、Array化したり……、

```
extension List {
    var asArray:[T] {
        return self.map{$0}
    }
}
```

もっと見やすいよう文字列化したりするのは楽勝です。

```
extension List : CustomStringConvertible {
    var description:String {
        return "List("
          + self.map{"\($0)"}
          .joined(separator:", ")
          + ")"
    }
}
print(List(0,1,2,3)) // "List(0, 1, 2, 3)"
```

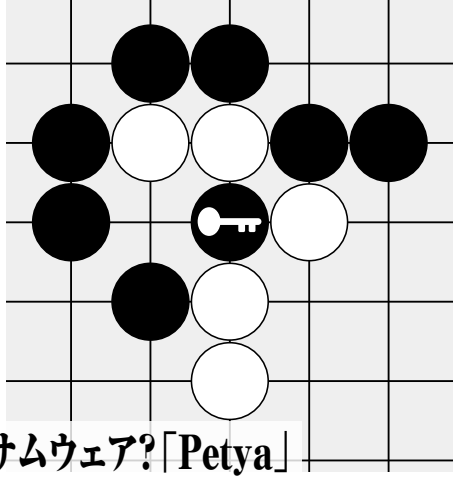
次回予告

今回はSwiftがSequenceプロトコルを通してどのように「順序」というものを抽象化しているかを見ていきました。しかし世の中は順序どおりにいくとは限らないもの。その場合には、どうしていくのか。

次回はCollectionプロトコルを通してそれを見ていきます。SD

セキュリティ実践の基本定石

すずきひろのぶ
suzuki.hironobu@gmail.com



みんなでもう一度見つめなおそう

【第四七回】マルウェアのトレンドを取り入れたランサムウェア?「Petya」

ランサムウェア「WannaCry」の騒動がまだ沈静化しない中、欧州ではランサムウェア「Petya」の被害が広がったという報道がありました。調べていくとPetyaは極めて興味深い特徴を持っています。ランサムウェアの話題が連続してしまいますが、あえてPetyaを取り上げたいと思います。

Petya

Petya(「ペトヤ」または「ペタヤ」と呼ぶ)は2017年6月27日に欧州を中心として一気に拡散したランサムウェアです[注1、注2]。最初はウクライナで感染が広がり、6月27日時点で感染台数は12,500台を越え、そこから欧州各国だけではなくブラジル、米国など64の国や地域で感染が確認されました。

Petyaはウクライナ製の会計ソフトMeDocのアップデータに混入されており、そこが第一の侵入経路になりました。ユーザは通常のアップデートと認識していますので、そのままインストールされてしまいます。いったんネットワーク内部のPCに入り込むと、そこから外部のサーバと通信を行うことが確認されています。次に内部ネットワークをスキャンし、WannaCryでも使われていたMS17-010の脆弱性を使うEternalBlueと、WannaCryでは使われていなかったEternal Romanceを使って内部感染を広げます。

EternalBlueとEternalRomanceはどちらも米国諜報機関NSAが作ったとされるMS17-010をターゲットにしたエクスプロイト手法(脆弱性を突いた侵入手法)です。自らをThe Shadow Brokersと呼ぶグループがネット上に公開しました[注3]。

Petyaは、PCをブートするために使うハードディスク上のMBR(Master Boot Record)を書き換え、MBRを書き換えた10分後にリブートをします。書き換えられたMBRによりWindowsではなくファイルチェックの偽装画面が現れます(**図1**)。このときにファイルの暗号化が行われます。次に身代金を要求する脅迫文を表示します(**図2**)。

しかし、この身代金要求の画面にあるメールアドレスはすでに封鎖されており、このビットコインアドレスに支払ったとしても、このメールアドレス宛に連絡することは不可能になっています。

さらにこの身代金要求画面にユーザを区別するためのキーID(personal Installation key)が表示されていますが、Kaspersky社の解析によれば、このIDは関連性のないランダム値であり、このIDから感

◆図1　偽CHKDSK画面、実際には暗号化が行われている(出典:Microsoft社のサイト[注1])

```
Repairing file system on C:

The type of the file system is NTFS.
One of your disks contains errors and needs to be repaired. This process
may take several hours to complete. It is strongly recommended to let it
complete.

WARNING: DO NOT TURN OFF YOUR PC! IF YOU ABORT THIS PROCESS, YOU COULD
DESTROY ALL OF YOUR DATA! PLEASE ENSURE THAT YOUR POWER CABLE IS PLUGGED
IN!

CHKDSK is repairing sector 32576 of 191968 (16%)
```

注1) "New ransomware, old techniques: Petya adds worm capabilities"
https://blogs.technet.microsoft.com/mmpc/2017/06/27/new-ransomware-old-techniques-petya-adds-worm-capabilities/
注2) "大規模な暗号化型ランサムウェア攻撃が欧州で進行中、被害甚大" http://blog.trendmicro.co.jp/archives/15339
注3) The Shadow Brokersに関しては本連載第45回(2017年7月号)でも取り上げています。

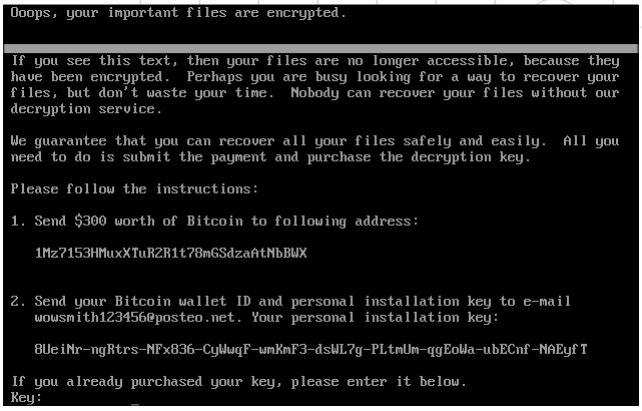
染先のデータを復号するための鍵を探すことはできないと結論づけられています[注4]。

しかし、Microsoft社の分析では、ファイルはAES-128を使って暗号化されており、AES-128で使ったセッション鍵はRSA-2048で暗号化されているとのことです。そしてその情報はドライブごとにREADME.TXT(**図3**)として保存されていると報告しています。つまり、このREADME.TXTの中のキーIDを送れば、この鍵を復号したものが戻ってきて、それを使えば元に戻せるしくみのようです。

混乱しそうなので暗号化から身代金脅迫の部分を整理すると次のようになります。

- **MBRが書き換えられているので本来のWindowsをブートできない**
- **MBRはPetyaを立ち上げる**
- **最初はCHKDSKでファイルをチェックしているような素振(そぶ)りをみせてファイルを暗号化している**
- **暗号化が終了すると身代金脅迫画面が表示される**
- **しかし、その身代金脅迫画面にある情報はフェイクで役に立たない**
- **本来のキーIDはドライブのREADME.TXTに入っているが、ユーザには提示しない**
- **ビットコインを支払っても指定先メールアドレスは閉鎖されているので連絡が取れない**

◆図2　300ドル分の身代金をビットコインで要求している画面(出典：Microsoft社のサイト[注1])



2016年のPetya

MBRを書き換えてブートシーケンスを乗っ取り、そのブート時に暗号化してしまうタイプのランサムウェアは2016年3月に発見されています[注5]。Petyaという名前は、このときにつけられました。なお、2017年版PetyaはオリジナルのPetyaとは暗号化する部分などが異なるため、Petyaファミリではなく別なものとして考えるケースもあります(後述)。

2016年版のPetyaは、MBRを書き換え、次のブート時にPetyaが立ち上がりCHKDSKに偽装して暗号化を行います。ただし、暗号化するのはファイルではなく、ドライブのMFT(Master File Table)です。MFTはファイルシステム上のファイルを管理するための情報部分で、ここが破壊されるとファイルがどこにあるのか見つけられなくなります。ファイルをひとつひとつ暗号化するよりも少ない時間でファイルシステムを使用不能にできます。そして、

◆図3　README.TXTの内容、AES-128の鍵をRSA-2048で暗号化しているようである

```
Ooops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because
they have been encrypted. Perhaps you are busy looking for a way to recover
your files, but don't waste your time. Nobody can recover your files without
our decryption service.

We guarantee that you can recover all your files safely and easily.
All you need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1.      Send $300 worth of Bitcoin to following address:

1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX

2.      Send your Bitcoin wallet ID and personal installation key to e-mail wowsmith123456@posteo.net.
        Your personal installation key:

AQIAAA5mAAAApAAA/yM8tPsoNwRpGRsJ9Ohu85ORQvnEk+nNoTIeEZzwe9TNkjfY
fQndHkeHXIKLEuIHrwjsYty536o88VfKArHR5jsvVf2yNXLBPMwtwripITpteWR7
bFrcdlkZ9L6xrlOzR7xLw/r5Wwfr/SZ6VZUPbbnDKSitTbjcX84UPow8c1dS7+xs
+XZVhUP703bGnJOFeBa8Sr+yR2O2Ae5lmp4d7hCoObrDT1JdoLkWXd2EqmlQOnRQ
VldJvMeTmBviZwe7LBpnyysd4wjY1OuHvwxUbMje4djclUXATQ8piGD7N9md63jF
uMa6S6j+pKUCWvK566i5XvuVw/iCVmLazkRMHw==
```

注4)　"ExPetr/Petya/NotPetya：ランサムウェアではなくワイパー"
https://blog.kaspersky.co.jp/expetrpetyanotpetya-is-a-wiper-not-ransomware/16707/
注5)　"ハードドライブを暗号化するランサムウェア「Petya」が発見される--研究者報告"　https://japan.zdnet.com/article/35080407/

被害者は元に戻したいなら、Torネットワークに接続して世間ではダークウェブとかディープウェブリンクとか呼ばれている一般のインターネットからは見られないネットワーク上にある支払いサイトで、ビットコインで0.9BTC（当時370ドル）を支払います。

MBRを書き換えて独自のプログラムを走らせたり、ファイルではなくOS側のファイルを管理する情報であるMFTを暗号化したりと、ランサムウェアとしてはこれまでになかった手法を使っています。技術的には特筆すべきランサムウェアでした。

ウクライナ製の会計ソフトMeDoc

ウクライナ製の会計ソフトMeDocとはどんなソフトウェアなのか調べてみたところ、たいへん興味深いことがわかりました。英語圏での情報はほぼなく、ウクライナ語のWikipedia[注6]を頼りにGoogle翻訳で調べました。

正式の表記はM.E.Docのようです。ウクライナのベンダが開発した会計と税務処理に使うソフトウェアで、MeDocのファイルはウクライナの税務当局に対して電子申請できる唯一のフォーマットのようです。ウクライナでは業者間の取引で使われる請求書や納品書といったものをMeDocのファイルフォーマットで交換しているようです。MeDocを使えば販売から税務まで一貫して利用できるようです。なかなか便利そうなソフトウェアで、ウクライナでは普及しているとの説明があるのもうなずけます。企業、とくに大手企業では効率化のために広く利用されているであろうことが容易に想像できます。

さらにWikipediaの内容を読んでいくと、ビックリする内容が書かれています。MeDocは、今回のPetya騒ぎ以前に、すでに更新システムが侵害されバックドアを埋め込まれていたという記述があります。2017年5月8日にMeDocのアップデータにランサムウェアXData[注7]が埋め込まれ、MeDocのアップデータを介して感染が広がったようです。

なぜアップデータが汚染されたのかを調べていくと、Cisco社のセキュリティチーム「Talos」が書いたブログ記事[注8]に参考になる情報がありました。それは、2017年6月27日にMeDoc社からデジタルフォレンジックの依頼があり、そこでTalosから現地に派遣されたエキスパートがサーバやネットワークを分析した結果をまとめたものです。

Talosは今回のPetya（の亜種）は、コードを流用してはいるが亜種とは呼べない別なものという見解を示しており、Petyaではなく「Nyetya」という独立した名前で呼んでいます。この記事にはバックドアがしかけられたアップデータが配布された背景も書かれていて、その部分を次に抜き出します。

- **MeDocのサーバ管理者が何らかの方法でパスワードを盗まれ、サーバのroot権限を取られていた**
- **ダウンロードサーバはNginxを使っているが、その設定をプロキシに変更されていた**
- **プロキシの先はバックドアが入っているアップローダを用意している別のサーバであった**

ほかにもwebshellが埋め込まれたphpファイルのタイムスタンプから、少なくとも2017年5月31日以降は、Webサーバに侵入できており、phpファイルなどを書き換えられたことがわかっています。

なお、XDataの問題は2017年5月8日に発生しているのですが、そちらに対してのサーバ側のフォレンジックに関わる情報は見つけられませんでした。

いずれにしても2017年5月8日時点で、すでにアップデート配布用サーバが侵入されてなんらかの方法でアップデータにバックドアをしかけられ、さらに2017年6月27日にまたアップデータにバックドアがしかけられ、その2つの間の期間である2017年5月31日にもサーバに侵入されてwebshellがしかけられていたということになります。MeDoc社のサーバ運用は極めてレベルが低く、最初の問題のときに適切な事後対応もできず、その後もアップデート用サーバはいいように外部から侵入されてお

注6）“M.E.Doc” **https://uk.wikipedia.org/wiki/M.E.Doc**
注7）XDataに関しては次の資料が詳しい“XData Ransomware on a Rampage in Ukraine”
https://www.bleepingcomputer.com/news/security/xdata-ransomware-on-a-rampage-in-ukraine/
注8）“MeDocコネクション” **https://gblogs.cisco.com/jp/2017/07/the-medoc-connection/**

り、最後Petyaで大騒ぎになって初めて本格的な対応をしたということです。まったく呆れるとしか言いようがありません。

Petyaは何だったのか

ここまでをまとめると、Petyaはあちこちのランサムウェアのモジュールをつなぎ合わせただけに見えます。MeDocのアップデータにバックドアをしかけたのも、すでに1回攻撃が成功しているのが知られている脆弱なサイトを狙っただけで独自性があるわけでも難しいわけでもありません。もしかするとウクライナのアンダーグラウンドでは侵入方法は知られていた、そう考えても不思議ではありません。

内部ネットワークをスキャンし、WannaCryでも使われたNSA製のEternalBlueとEternalRomanceを使って内部ネットワーク内にあるMS17-010のパッチが当たっていないPCに片っ端から感染していきます。そのため、会社や組織単位で麻痺してしまう結果になりますが、これもすでにWannaCryが効果を実証したもので独自性はありません。

そして、この犯人は、本来ランサムウェアで通知すべきキーIDをユーザがアクセスできないファイルREADME.TXTに書き込み、一方でPetyaのように最初に脅迫画面を表示し、そこには意味のないキーIDと、さらに簡単に停止させられてしまうメールアドレスを表示しているという理解しがたいマルウェアを作成しています。

Petyaのようにダークウェブのインフラを用意できないので苦し紛れにメールアドレスを使ったと考えるのが自然だと思いますが、ビットコインの振込とキーIDはマッチできないし、そもそも表示しているキーIDもどきは関係のない値です。もう一段考えてみれば、フリーフォーマットの書き方をしたメールが大量に届いたとして、どうやってそれを処理するのかという素朴な疑問に突き当たります。

Petyaの関連記事を読んでいくと、「戻せないということは、これはウクライナの企業や組織に打撃を与えるためにPCの中身を破壊するのが本来の目的なのでは」という記述も散見されました。しかし、それならば、わざわざひとつひとつファイルを暗号化するのは矛盾しています。オリジナルのPetyaのようにドライブのMFTを壊してしまえば簡単に致命的な損傷を与えられます。

オリジナルのPetyaは、MFTを損傷させてなおメッセージを立ち上げられるしくみなのですから(そのため、MBRのベクターを書き換えている)、ハードディスクの内容をきれいさっぱり消したうえで、ハードディスクに情報を書き込み、脅迫メッセージだけ立ち上がるようにすることだって可能でしょう。破壊してからランサムウェアに見せかけて時間稼ぎをするなら、こちらのほうが簡単です。

Petyaを一言で表せば「矛盾」です。何も新しいことはしていません。どの局面を見ても中途半端です。流用した個々のマルウェアのモジュールはバグがなくとも、全部合わせて動かすとバグっているとしか筆者には見えません。唯一、確実なのはPetyaが1回動作すると、もうPCを戻せないということだけです。結果は破滅的ですが、こんな洗練も何もないランサムウェアをサイバー攻撃というレベルの高いものと位置づけるのはどうかと思います。

ただし、サイバー攻撃よりも今回のPetyaのほうがもっと怖い状況にあります。なぜなら、最後までプログラムを作れないレベルの人間がアングラサイトから拾ってくる情報やコードをもとにマルウェアを作り、それが大規模で深刻な感染を引き起こし、企業や組織を麻痺させることができたからです。しかも、そのモジュールには、NSAのEternalBlueやEternalRomanceといった強力な兵器級とも思えるエクスプロイトコードが含まれているのです。

また、WannaCryが散々騒がれたにもかかわらず、いまだにMS17-010のパッチが当たっていないコンピュータが多数あるのです。行政とのやりとりに使われ、国レベルで利用が推奨されるようなソフトウェアのアップデートサーバが簡単に侵入され、長期間侵入者のコントロール下にあるというずさんな管理も懸念される事項です。

Petyaは何だったのかと問われるならば、考え得る最悪な状況がすべて含まれている事例だったと言えるのではないでしょうか。SD

SOURCES

レッドハット系ソフトウェア最新解説

第12回

RHEL System Rolesを利用した構成管理

RHEL7.4からAnsibleを提供するようになり、RHEL System Rolesを利用した構成管理手順を実施できるようになりました。今回は簡単な利用方法を紹介します。

Author 小島 啓史(こじまひろふみ)
mail：hkojima@redhat.com
レッドハット㈱ テクニカルセールス本部 ソリューションアーキテクト

RHEL System Rolesの概要

RHEL System Roles[注1]は、RHELのサブシステム(ネットワーク／SELinuxなど)の構成管理をAnsibleで行うためのRoles[注2]が集まってできたものです。AnsibleとRHEL System RolesはRHEL7.4から提供するようになったため、RHELユーザでAnsibleを使いたい場合、これまでのようにEPELリポジトリ[注3]を有効にする必要がなくなりました。ただし、現状でのRHELに含まれるAnsibleは、RHEL System Roles(執筆時点ではTechnology Preview)を利用してRHELの構成管理を行う場合に限りサポートされる予定ですので注意ください。なお、執筆時点ではRHEL7.4はPublic Beta[注4]にだけ提供している状態ですが、Public Betaの提供開始時期は2017年5月24日であり、通常のスケジュールではそこから2〜3ヵ月ほどテストや修正を行ってから製品版を提供しますので、本誌がみなさんのお手元に届くころには製品版を利用できるようになっているはずです。RHEL7.4提供開始時点で用意されているRHEL System Rolesは次の5項目です。

- kdump
- postfix
- network
- selinux
- timesync

これら5項目のサブシステムの構成管理をする場合は、専用のライブラリなどを利用できるため、AnsibleのPlaybookをよりシンプルに記載できるようになります。ちなみに、RHEL System RolesのアップストリームプロジェクトはLinux System Roles[注5]です。Linux System Rolesのロードマップには、上記5項目のほかにファイアウォール／システムロギング／ケルベロス認証などが含まれており、今後RHEL System Rolesにもバックポートされていく予定です。

RHEL System Rolesの利用方法

RHEL System Rolesを利用するには、執筆時点ではRHEL7.4のPublic Betaを入手する必要があります。製品版が利用できる場合は、以降

注1) URL https://access.redhat.com/articles/3050101
注2) URL http://docs.ansible.com/ansible/playbooks_roles.html#roles
注3) URL https://fedoraproject.org/wiki/EPEL/ja
注4) Red Hatのいずれかの製品を持っている場合だけ無償でダウンロードできる提供形態です。
注5) URL https://linux-system-roles.github.io/

▼図1　RHELの登録と製品リポジトリの有効化

```
# subscription-manager register --auto-attach
# subscription-manager repos --disable="*"
# subscription-manager repos --enable=rhel-7-server-beta-rpms --enable=rhel-7-server-extras-beta-rpms
```

▼図2　Ansibleのインストール／設定とSSH鍵の配布

```
# yum -y install ansible rhel-system-roles
# ssh-keygen -f /root/.ssh/id_rsa -N ''
# ssh-copy-id root@RHEL_CLIENT_HOST
# echo RHEL_CLIENT_HOST >> /etc/ansible/hosts
```

の記載でBetaやPublic Betaなどとなっているところを、製品版に置き換えて参照ください。まずはAnsibleとRHEL System Rolesをインストールする管理サーバ1台と、クライアント(RHEL6.9より新しいリリースのもの)を1台以上用意します。サーバの台数がない場合は、管理サーバとクライアントを同じものにしてもかまいません。管理サーバにRHEL7.4 Public Betaをインストールして、AnsibleとRHEL System Rolesをインストールするための製品リポジトリを有効にします(**図1**)。

続いてansibleパッケージとrhel-system-rolesパッケージをインストールします。インストール後は管理サーバ側で作成したSSH公開鍵をクライアントに配布して、クライアントのホスト名またはIPアドレスをAnsibleのインベントリファイル(ここでは/etc/ansible/hosts)に追加します(**図2**)。

これでRHEL System Rolesを利用する準備が整いました。ここからは具体例を挙げながら、RHEL System Rolesの使い方を紹介します。

たとえばセカンダリのネットワーク・インターフェース(eth1)に動的なIPアドレスを割り当てるためのPlaybookは**リスト1**のようになります。

AnsibleでRHELのネットワーク・インターフェースにIPアドレスを割り当てるには、nmcliモジュール[注6]を使うこともできますが、RHEL System Rolesを利用することでnmcliなどの実装技術が隠蔽され、「network_connections」変数としてインターフェース名やIPアドレスの割り当て方式などを設定するだけでよくなったシンプルなPlaybookになっています。このPlaybookをansible-playbookコマンドで実行すると、ネットワーク・インターフェース設定ファイルとしてifcfg-eth1がクライアント側に作成されて、動的なIPアドレスの割り当てが実行されます。ちなみに、このPlaybookではDHCPを指定していますが、静的なIPアドレスを設定したい場合は、変数「ip」の個所を次のように変更することで対応できます。

▼リスト1　「eth1」インターフェースに動的IPアドレスを割り当てるPlaybook

```
- hosts: all
  vars:
    network_connections:
      - name: eth1
        type: ethernet
        ip:
                dhcp4: yes
        interface_name: eth1
  roles:
    - role: rhel-system-roles.network
```

```
ip:
  gateway4: 192.168.199.254
  address:
    - 192.168.199.11/24
```

ネットワークについてはほかにもVLAN/Bonding/Bridgeの設定[注7]ができます。

また、kdumpの設定を行うPlaybookは**リスト**

注6) URL http://docs.ansible.com/ansible/nmcli_module.html

注7) URL https://github.com/linux-system-roles/network

▼リスト2　kdumpの設定を行うPlaybook

```
- hosts: all
  vars:
    dump_target:
      kind: raw
      location: /dev/sda1
  roles:
    - role: rhel-system-roles.kdump
```

2のようになります。カーネルクラッシュ時のデバッグに役立つkdump用のモジュールは執筆時点では存在しませんので、RHEL System Rolesを利用することで、kdumpにより出力されるvmcoreの置き場所を変数として指定するだけのシンプルなPlaybookの開発につなげることができるようになります。このPlaybookを実行すると、kdumpの設定ファイル(/etc/kdump.conf)が変更され、kdumpサービスが再起動されて変更後の設定が反映されます。vmcoreの置き場所はローカルホストのほかにも、SSHサーバやNFSサーバをPlaybook[注8]で指定できます。

今回、紹介したネットワークやkdumpのほかにも、SELinux[注9]/Postfix[注10]/timesync(NTPやPTPによる時刻同期[注11])の設定をRHEL System Rolesを利用して行うことができます。こうしたAnsibleによる構成管理を、kickstart機能によるRHELの自動インストールやRed Hatが提供するAnsible Tower(2016年8月号・9月号の同記事で紹介)と組み合わせることで、RHELシステム管理の自動化基盤を整備することにつながります。興味がありましたら、ぜひとも試してみてください。SD

注8) URL https://github.com/linux-system-roles/kdump
注9) URL https://github.com/linux-system-roles/selinux
注10) URL https://github.com/linux-system-roles/postfix
注11) URL https://github.com/linux-system-roles/timesync

ひみつのLinux通信

作)くつなりょうすけ
@ryosuke927

第43回 キーワードは“宿題”

「オマエは今まで食べたパンの枚数を覚えているか」と問う編集に、「何度目のお盆進行か」と切り返すくつな先生のマンガが読めるのは本誌だけ！

to be Continued

子供たちはまだ夏休みですよね。暑い中仕事に出てる皆様、ご苦労様です。そんな子供たちの夏休みの終わりがすぐソコまで迫って来てます。一部のお父さんとお母さんは宿題の手伝いをする心の準備をしてください。夏休みの宿題の片付け方を思い出してみると、仕事の片付け方とそんなに変わってないのに気づきますよね。そう考えると、学校に送り込まれたあのときから「宿題」から「仕事」に名前は変わったけどナニかを片付ける日常を過ごす、無限ループに突入したとは言えませんか！　無理矢理あのアニメの話と関連付けようと思いましたがカスリもしないし、Linuxの話題も出ないので今宵はここまでに致しとうござりまする。仕事爆発しろっ！

LibreOffice 5.4の新機能

Ubuntu Japanese Team　あわしろいくや

今回は、7月28日にリリースされたLibreOffice 5.4の新機能や変更点について解説します。

そして6.0へ

LibreOffice 5.xシリーズはこの5.4でおしまいとなり、次のリリースは6.0になることがすでに決定しています。メジャーバージョンアップに伴って大きな変更も予定されており、すでに(本連載とは関係ありませんが)Windows XPサポートの削除が決定しています。

そんな中、5.4は全体的にあまり大きな変更点はなく、細かな変更が目立ちます。LibreOfficeの場合はサポート期間的に大きな変更点がないからといって前のバージョンを使い続けることができないのが辛いところです。ちなみに5.2のサポートはすでに終了し、5.3も11月末でサポートが切れてしまいます。もちろんUbuntu 17.04にあらかじめインストールされている5.3は17.04のサポート終了(2018年1月末)までは継続します。

今回は検証に5.4.0.2を使用しました。5.4.0.3が5.4.0としてリリースされる見込みです。未訳の部分は随時翻訳するとともに、本文中では翻訳と原文を併記してあります。

では、詳しくみていきましょう。

全般

デジタル署名関連

LinuxプラットフォームではOpenPGPでのデジタル署名ができるようになりました。OpenPGPは規格の名前で、広く使われている実装はGnuPGです。もちろんGnuPGで作成した鍵でも署名はできました。

また、無効な署名がされている場合はわかりやすく警告が表示されるようにもなりました。

PDF関連の変更

LibreOfficeはPDFファイルを画像として挿入することもできますが、従来は一度ODFに変換してから画像にしていました。これだと劣化が避けられないということで、Google Chrome/Chromium で採用されているpdfiumというライブラリを使用し、PDFを直接レンダリングするようになり、大幅に表示品質が向上しました。

同時に動画のPDFエクスポートにも対応しました。動画へのリンクだけでなく、PDFの中に動画を埋め込むこともできます。

Writer

透かし

ドキュメントに透かしを入れられるようになりました。[書式]-[透かし]でダイアログを表示し、テキスト、フォント、角度、透明度、色を選択すると各ページに透かしが入ります(**図1**)。これまではPDFエクスポートの際にしか入れられなかったので、便利になりました。もちろんここで入れた透かしは

PDFエクスポートでも、Microsoft Wordファイル形式の保存でも保持されます。

オートコレクトオプションの追加

入力した文字や書式を自動的に置き換えるオートコレクトは評判の悪い機能の1つではありますが、LibreOfficeでは絵文字の入力にも使えるようになっているなど、活躍の機会を増やしています。

そんな中、5.4からは新たに「***太字***、**/斜体/**、**-取り消し線-**、**_下線_**を自動で入れる」というオプションが追加されました。この機能は、たとえば次のように、

太字にしたい文字

と入力すると、

太字にしたい文字

のように太字になるものです。いちいち文字の入力後にツールバーやサイドバーから太字にする必要がないので便利でしょう。アスタリスク(*)の代わりにスラッシュ(/)、ハイフン(-)、アンダーバー(_)を入力するとそれぞれ斜体、取り消し線、下線に変換されるという動きになります。

ユーザインターフェースの変更

[表示]-[ツールバー]に[書式設定(スタイル)]が追加されました。これまではスタイルから設定できなかった見出し(H1～3)などがツールバーから直接設定できるようになりました。

右クリック(コンテキストメニュー)に[スタイル]というサブメニューが追加されました。そのサブメニュー内には、これまであった[スタイルの編集]のほかにもいくつかあり、基本的な文字スタイルであればこれだけで完結できます。

ページ内に独立した段落を挿入する機能である[セクション]内で右クリックした場合、直接セクションの編集ができるようになりました。また、脚注ないし文末脚注でも同じく右クリックすると直接編集できるようになりました。

図1　透かしを入れるダイアログ

Calc

ROUNDSIG関数の追加

新しく追加されたのはROUNDSIG関数で、Excelには同様のものはないようです。ROUNDは四捨五入、SIGはSIGnificant digit、すなわち有効数字で、定義した桁数の有効数字で四捨五入ができる関数です(**図2**)。言うまでもなくROUND関数は小数点以下の桁数で四捨五入しますので、有効数字を求めたいときには便利でしょう。

条件付き書式の優先度変更

条件付き書式の条件を順序を入れ替えることにより、優先度を変更できるようになりました。

CSVエクスポートの設定

CSVエクスポートの設定が自動的に保存されるよ

図2　追加されたROUNDSIG関数

うになりました。

紀元前の入力

紀元前[注1]の入力が簡単にできるようになりました。たとえば紀元前7年7月15日の場合、"-7/7/15"と入力します。もちろんただ入力するだけではなく、"=WEEKDAY(-7/7/15,1)"を入力すると"7"が返ってくるので、土曜日であることがわかるなど[注2]、計算もできます。

ピボットグラフ

ピボットグラフが作成できるようになりました。ピボットテーブルで作成した表でグラフを作成すると、ピボットグラフになります。

セルのコメント

セルのコメントが扱いやすくなりました。[表示]-[コメント(Comments)]でコメントの一括表示と非表示の切り替えができるようになりました。

また、[シート]-[セルのコメント(Cell Comments)]に[すべてのコメントを削除(Delete All Comments)]が追加されました。

セルの保護のトグル

[編集]-[セルの保護]でセルの保護をトグルできるようになりました。なお、シートの保護を設定していない場合はセルの保護を行っても効果はありません。

シートの保護方法の追加

シートの保護で、行／列の追加／削除を許可するかどうかを選択できるようになりました(**図3**)。なお、Excel 2013ではより多くの制御が設定でき、まだまだ発展する余地がある項目です。

ツールバーの数式の書式ボタン

ツールバーにある[数字の書式]ボタンがトグルできるようになりました。たとえばセルに"1"を入力し、[数式の書式]ボタンを押すと"1.00"になります。もう一度押すと"1"に戻ります。なお、その横の[パーセントの書式]ボタンは以前からこのように動作していました。

セルスタイルの追加

[書式]-[スタイル]にたくさんのセルスタイルが追加されました。もちろん[スタイルと書式設定]でカスタマイズや追加ができますが、あらかじめ登録されている分でも十分に実用的です[注3]。

Draw/Impress

複製ダイアログの仕様変更

Shift+F3キーで表示する複製ダイアログの仕様が変更になりました。[回転角度]は小数点第2位まで設定できるようになったほか、以前の設定を記憶するようになりました。

その他

サポートOS

Ubuntuとは直接関係ありませんが、前述のとおり

注1) リリースノートでは、BCE=Before Common Eraと書かれていました。B.C.(Before Christ)ではないのですね。

注2) 現在の暦で紀元前の日付の曜日を計算することに意味はありません。

注3) Excelのスタイルにも同じような機能があるので、そこからインスパイアされたものと思われます。ちなみに翻訳もインスパイアしてあります。

図3　シートの保護で設定できる項目が増えた

Windows XPはこのバージョンまでのサポートです。Windows Vistaも同様にサポートが切れているうえにユーザがほぼいないので問題はなく、次の6.0からはWindows 7以降のサポートになると思われます。同様にmacOSは10.8までのサポートがこのバージョンまでで、次の6.0からは10.9以降のサポートになる予定です。

ノートブックバー

［ツール］-［オプション］-［LibreOffice］-［詳細］の［実験的な機能を有効にする］にチェックを入れ、LibreOfficeを再起動すると［表示］-［ツールバーのレイアウト(Toolbar Layout)］に［ノートブックバー］が増えます。さらに［表示］-［ノートブックバー］で2～5の表示モードに切り替えることができます。デフォルトは［タブ(Tabbed)］で、これがMicrosoft Officeのリボンメニューのような感じです。5.3で初登場以来5.4でもやはり実験的な機能のままですが、開発は確実に進んでいて、タブのところにショートカットアイコンが表示できるようになったり、テーマをサポートしたりといった変更が見られます。

GSoCの開発状況

LibreOffice 5.4の開発と並行して、今年もGSoC(Google Summer of Code)の支援による開発も進行しています。GSoCはGoogleがスポンサーとなり、LibreOfficeを含むたくさんのFLOSSプロジェクトに世界中の学生が集まって、夏の期間中に自ら立てた目標に向かって開発するというものです。LibreOfficeはプロジェクト開始後から毎年継続してGSoCに採択され、その後プロの開発者になった人も多いです。成果のうちの大部分は6.0またはそれ以降に取り込まれますが、中には忘れ去られるものもあります。その中から興味深いものをいくつか紹介します。

とくに興味深いのは、LibreOffice for Android関連で2つの開発が行われていることで[注4]、どちらもブラッシュアップが主眼であり、いずれ実用的に使用できるようになるのかもしれません。個人的に一番期待しているのは、ODFファイルの読み込みの高速化です[注5]。ODFファイルはZIPで圧縮されたXMLファイルですが、このXMLパーサーを新しくするという意欲的なプロジェクトです。

ほかには、一世を風靡(ふうび)したDTPソフトであるQuarkXPressインポートフィルタの実装というのもあります[注6]。QuarkXPress自体は現行の製品ですが[注7]、過去のファイルフォーマットとは互換性を失っているようで、バージョン4のファイルを読み込みできるように開発が進められています。

ユーザインターフェース関連では、［挿入］-［記号と特殊文字］と［ツール］-［カスタマイズ］をそれぞれ改良しています[注8]。

もちろんほかにも興味深いものがありますが、LibreOfficeではなくThe Linux Foundationのプロジェクトで CUPS関連の開発を行っていますが、そのうちの1つとしてLibreOfficeの印刷機能の改善も含まれています[注9]。

LibreOffice 5.4とUbuntu 17.10

LibreOffice 5.4は、10月19日リリース予定のUbuntu 17.10に収録される見込みです。随時PPA[注10]でも利用できるようになると思われますが、公式発表はないもののCanonicalのLibreOffice担当社員がどうもこの4月に退職したらしく、動きが鈍くなっているのが気になります。最新版を使いたい場合は、オフィシャルビルドを使用するのがいいかもしれません。SD

注4) https://summerofcode.withgoogle.com/projects/#6637875643809792
https://summerofcode.withgoogle.com/projects/#5462728836644864

注5) https://summerofcode.withgoogle.com/projects/#4624279755816960

注6) https://summerofcode.withgoogle.com/projects/#5806898726043648

注7) http://www.quark.com/jp/Products/QuarkXPress/

注8) https://summerofcode.withgoogle.com/projects/#4910404967858176
https://summerofcode.withgoogle.com/projects/#6712596873871360

注9) https://summerofcode.withgoogle.com/projects/#6401547484266496

注10) https://launchpad.net/~libreoffice

Unixコマンドライン探検隊
Shellを知ればOSを理解できる

Author 中島 雅弘(なかじま まさひろ) ㈱アーヴァイン・システムズ

テキスト処理の決定版、ストリームエディタのsedと、プログラミング言語としても、機敏なツールとしても使えるawkの基本について、ワンライナーを中心に探検します。

第17回 テキスト処理(その3)

Unixを使う理由の1つが、このテキスト処理能力です。あまりに便利なUnixのテキスト処理ツールは、ほかの環境にも移植されています。でも、特定のツールを移植するだけでは力を発揮しきれません。シェルとツールを組み合わせて使うのがUnixの考え方です。

多くのテキスト処理は、IDEや強力な対話型エディタで時間をかけてやればできることかもしれませんが、プログラムと同様に自動化すれば繰り返し使うことができます。そして小さなツールの組み合わせなら、一般的な言語でのプログラミングよりずっと簡便に解決を得られます。

本格的なテキスト処理

今回フォーカスするのは、テキスト処理の決定版とも言える2つのコマンド、sedとawkです。題材には、本連載の定番となったシェークスピアのハムレット英文を使いましょう[注1]。このテキストのファイル名はhmlt.txtとします。なお、ファイルの作り方によって、手元での出力結果と本稿の出力結果とが異なる場合があることをご承知おきください。

注1) 以前の連載を読んでいない方は、ネット上にある英語のテキスト(たとえば、http://shakespeare.mit.edu/hamlet/full.html)をhtmlではなく、プレーンテキストで保存して使ってください。

sed——Stream EDitor

sedは、正規表現を使える強力なテキスト処理ツールです。対話型のラインエディタedに対して、フィルタ処理に特化しています。edは、exとして発展し、viのexモードにも継承されています。exモードを自在に扱える人なら、sed[注2]は違和感なく使えることでしょう。

sedの基本的な操作

sedを使う基本として、正規表現に、置換コマンドs、削除コマンドd、範囲指定の方法、マッチの意味の反転!を修得しましょう。

はじめは、grepのような使い方、検索の例です。dは対象行を削除するコマンドです。dの前に範囲、もしくは対象になる行、/で挟んだパターン(正規表現)を指定できます。!で、前に指定した範囲を逆転させます。次の例は、"or not to be"を含む行以外の行を削除しろという指示です。

"or not to be"を探す。パターンにマッチした行以外を削除する方法
```
$ sed '/or not to be/!d' hmlt.txt ↵
To be, or not to be: that is the question:
```

次に、=コマンドを使って、処理対象の行が何行めかを表示させてみましょう。複数のコマンドは、;で区切って並べます。

注2) sedは、exではなくedコマンドに準じています。

```
"or not to be"を探す。パターンにマッチした行以外を削除し、マッチした行が何行目かと、マッチした行のテキストを表示
$ sed '/or not to be/!d;=' hmlt.txt
3791
To be, or not to be: that is the question:
```

次は、範囲指定の例です。

```
範囲の指定方法の例
1行目を削除する
$ sed '1d' hmlt.txt
1～20行目を削除する
$ sed '1,20d' hmlt.txt
FRANCISCOを見つけた行から、BERNARDOが見つかるまでの行を削除する。その後またFRANCISCOが見つかったらBERNARDOの行まで同様に削除、これを繰り返す
$ sed '/FRANCISCO/,/BERNARDO/d' hmlt.txt
```

続いてsコマンドを使って、文字列を置き換えます。

s/パターン/置き換える文字列/

と表記します。このままだと、行中の最初のパターンが1つだけ置き換えられます。この後ろに、gオプションを指定すると、行中で見つかったすべてのパターンに対して置き換えを実施します。

```
すべての"Hamlet"を"團十郎"にする
$ sed 's/Hamlet/團十郎/g' hmlt.txt
...略...
PRINCE FORTINBRAS

Let four captains
Bear 團十郎, like a soldier, to the stage;
For he was likely, had he been put on,
To have proved most royally: and, for his passage,
The soldiers' music and the rites of war
Speak loudly for him.
Take up the bodies: such a sight as this
Becomes the field, but here shows much amiss.
Go, bid the soldiers shoot.

A dead march. Exeunt, bearing off the dead bodies; after which a peal of ordnance is shot off
```

locale, I18N, L10N, M17N

先の例では、なにげなく日本語文字列を使いましたが、ASCII文字以外を扱う場合には注意が必要です。現在のソフトウェアの多くは、複数の国や地域で使われている言語(文字、記号など)に対応しています。

ソフトウェアの多言語対応は、Internatinalization(国際化)、Localization(特定の言語・地域に合わせる)、Multilingalization(多言語対応)の段階があります。それぞれ長い単語ですので、最初と最後の文字と、その中に何文字あるかという表記で省略して、I18N、L10N、M17Nと表記します。

各OSやコマンドがどの程度の対応ができているかはまちまちです。(同じコマンドであってもバージョンによって)多言語対応ができていないソフトウェア、多言語を扱えるとしても不完全なものもあります。こうした環境下では、(私達の場合とくに)日本語文字列を扱おうとすると、期待どおりの動作結果を得られないことがあります。

ロケール(LOCALE)は、言語・地域による文字、単位、記号、日付などの表記方法です。macOSやLinuxにおいては、環境変数(LANG、LC_*など)でロケールを指定してソフトウェアの動作を決定しています。目的の言語を適切に扱うには、そのロケールで、ソフトウェア(先の例では、sedはもちろん端末やシェル、OSのライブラリなど)がそれらを扱える状態であること、ロケールに合ったデータ(フォントやドキュメントなど)が導入されている必要があります。

ロケールC[注3]は、POSIXロケールと呼ばれる、すべてのPOSIX準拠システムで使える、もっともシンプルなロケールです。設定しているロケール(C以外)に対応したフォントやドキュメント(manpageなど)が導入されていないなどのときに文字化けしてしまう状況などで、このロケールを設定すれば、少なくとも英語での表示になります。ですので、探検隊で扱う多くの例も、環境が異なっていても混乱が少ないようにCロケールを前提にしたものが多いのです。

localeコマンドで、ロケールの設定状態を確認できます。

```
Ubuntu 16.04での例(部位ごとに異なる言語を指定できることが確認できる)
$ locale
LANG=en_US.UTF-8
LANGUAGE=en
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC=ja_JP.UTF-8
LC_TIME=ja_JP.UTF-8
LC_COLLATE="en_US.UTF-8"
LC_MONETARY=ja_JP.UTF-8
LC_MESSAGES="en_US.UTF-8"
LC_PAPER=ja_JP.UTF-8
LC_NAME=ja_JP.UTF-8
LC_ADDRESS=ja_JP.UTF-8
LC_TELEPHONE=ja_JP.UTF-8
LC_MEASUREMENT=ja_JP.UTF-8
LC_IDENTIFICATION=ja_JP.UTF-8
LC_ALL=
```

注3) LANG=C, LC_ALL=Cとすることで設定できます。

sコマンドでも、コマンドの前に範囲を指定できます。

```
1行目から20行目に出現する"FRANCISCO"を海老蔵に置換する
$ sed '1,20s/FRANCISCO/海老蔵/' hmlt.txt
Hamlet: Entire Play
...略...
SCENE I. Elsinore. A platform before the castle.

海老蔵 at his post. Enter to him BERNARDO

BERNARDO
...略...
```

-iオプションを指定すると、標準出力には出力されず、入力ファイルが直接編集されます。

```
-iオプションでファイルに書き込み
$ sed -i 's/Hamlet/團十郎/g' hmlt.txt
```

デリミタを"/"から変更

正規表現は、空白と改行以外ならなんでも区切り文字(デリミタ)として使えます。パス名などをマッチさせたいとき、/がデリミタ文字だとパスの区切りのたびに\でエスケープしなければならなくて入力も面倒ですね。このしくみは、vimのexモードでも使える便利技なのですが、あまり知られていません。;-)

```
前の例は、このようにも書ける
$ sed 's:Hamlet:團十郎:g' hmlt.txt
```

ご覧のようにsedのコマンドは、1文字のものが基本です。1文字のコマンド、範囲指定、正規表現になれることで自在に使いこなせるようになります。

awk

テキスト処理は、awk抜きでは語れません。とにかく強力で手軽です[注4]。

awkは、CやJavaなどの手続き的な言語の側面と、Prolog[注5]やmake[注6]のような宣言的な言語の両方の性質を備えたスクリプト言語です。具体的には、「パターン」にマッチしたときに「アクション」を実施する組を書き並べていきます。

また、組み込み変数や組み込み関数、ユーザ定義関数なども充実していますので、ワンライナーによる簡便な処理だけでなく、大規模なプログラムまで記述することも容易です。今回はワンライナーの事例を見ながら、awkの力を実感してみてください。

awkの基本

最初は、文字列の長さを調べるlength関数を使った例です。lengthに引数を渡さなければ、対象行の長さを返します。

```
長さが半角で30字を越える行を抽出
$ awk 'length > 30' hmlt.txt
The Tragedy of Hamlet, Prince of Denmark
SCENE I. Elsinore. A platform before the castle.
FRANCISCO at his post. Enter to him BERNARDO
...略...
```

組み込み変数NFは、対象行のフィールド数が収納されます。フィールドは、フィールドセパレータ(デフォルトではスペースとタブ)で区切られた文字の並びです。ハムレットのケースでは、単語とほぼ同義と考えて差し支えないでしょう。

```
フィールド数が5以上10以下の行を抽出
$ awk 'NF >= 5 && NF <= 10' hmlt.txt
The Tragedy of Hamlet, Prince of Denmark
SCENE I. Elsinore. A platform before the castle.
FRANCISCO at his post. Enter to him BERNARDO
Nay, answer me: stand, and unfold yourself.
...略...
```

headコマンドと同じように最初の5行だけ表示させてみましょう。これには、読み込んでいる行が何行目であるかが入っている組み込み変数NRを使います。

```
最初の5行だけ表示
$ awk 'NR <= 5' hmlt.txt
Hamlet: Entire Play



The Tragedy of Hamlet, Prince of Denmark
```

最初の単語の長さが2文字以上6文字未満の

注4) 一歩先に進みたい方は、拙著『AWK実践入門』(技術評論社)を参考にしてください。
注5) 一階述語論理を背景にしたプログラミング言語。
注6) 本誌2016年10月号の連載第6回で登場したビルドなどに使えるコマンド。

行だけ表示させてみましょう。

```
第1フィールドの長さが6文字未満の行を表示
$ awk 'length($1) >1 && length($1) < 6' hmlt.txt
The Tragedy of Hamlet, Prince of Denmark
ACT I
SCENE I. Elsinore. A platform before the castle.
Who's there?
...略...
```

awkは、ほぼERE[注7]が使えます。次の例は、Hamletまたはkingを含む行を表示させます。また、HamletかPrinceを大文字小文字を区別せずに含む行を表示してみましょう。

```
$ awk '/Hamlet|king/' hmlt.txt
Hamlet: Entire Play
The Tragedy of Hamlet, Prince of Denmark
     Hamlet
Long live the king!
In the same figure, like the king that's dead.
...略...

$ awk 'BEGIN{IGNORECASE=1} /Hamlet|Prince/' hmlt.txt
...結果省略...
```

次は、少し複雑な例です。6番めの単語を抜き出し、単語ごとにどの行に出現しているかを一覧にしてみましょう。

```
6番めの単語が出現している行を一覧化
$ awk 'NF > 5 {a[$6] = a[$6] " " NR} END {for (reg in a) print reg " " a[reg]}' hmlt.txt
dog  7999
littlest  4460
are  500 755 1379 ...略...
and  71 150 177 250 ...略...
pelican,  6617
...略...
```

今度は、実用性の高い例、改行だけの行およびタブと空白からなる行を削除します。

```
改行だけの行およびタブと空白からなる行を削除
$ awk '!/^[ ¥t]*$/' hmlt.txt
Hamlet: Entire Play
The Tragedy of Hamlet, Prince of Denmark
     Shakespeare homepage
     Hamlet
     Entire play
ACT I
...略...
```

最後の行だけ表示してみます。今回対象のファイルでは、空の行ですね[注8]。

```
最後の行だけ表示
$ awk '{line = $0} END {print line}' hmlt.txt
```

最大フィールド数を数えてみましょう。

```
最大フィールド数を表示
$ awk '{if (f<NF) f=NF} END {print f}' hmlt.txt
122
```

最長の1行あたりの文字数を数えてみましょう。

```
最長文字数を表示
$ awk '{if (l<length) l=length} END {print l}' hmlt.txt
671
```

nlコマンドのように行番号を付けてテキストを表示してみましょう。

```
行番号を加えて内容を表示
$ awk '{printf("%5d: %s¥n", NR, $0)}' hmlt.txt
    1: Hamlet: Entire Play
...略...
    6:      Shakespeare homepage
    7:      Hamlet
...略...
```

フィールドを1行に1つ、行番号を付けて表示します。

```
フィールドを1行に1つ、行番号を付けて表示
$ awk '{for(i=1;i<=NF;i++) printf("%6d: %s¥n",++s,$i)}' hmlt.txt
     1: Hamlet:
     2: Entire
     3: Play
     4: The
     5: Tragedy
     6: of
...略...
```

HamletとOpheliaのセリフの回数を数えてみましょう。戯曲ですので、名前がすべて大文字になっているだけの行を数えれば、おおむねの数はわかるはずです。

注7) Extended Regular Expression：拡張正規表現。詳しくは本誌2017年3月号の連載第11回を参照。

注8) hmlt.txtの作り方によっては違うかもしれません。

HamletとOpheliaのセリフの数
```
$ awk '/^HAMLET$/ {h++} /^OPHELIA$/ {o++}
END {print "Hamlet=" h "¥n" "Ophelia=" o}'
hmlt.txt
Hamlet=359
Ophelia=58
```

HAMLETと行頭からはじまっている行で、そのあとに何らかの文字が続いている行はあるのでしょうか。調べてみましょう。

HAMLETの考察
```
$ awk '/^HAMLET/ {h++} END {print "Hamlet="
h}' hmlt.txt
Hamlet=360
```

前の結果よりも1行多いですね。どのような行なのかgrepで絞り込んでみます。次の例では、2回grepを使っています。はじめに行頭にHAMLETがある行を抽出、次にHAMLETで終了していない行を抽出しています。

どの行が、HAMLETだけでない行なのか
```
$ egrep '^HAMLET' hmlt.txt | egrep -v
'HAMLET$'
HAMLET moves him to put on his hat
```

awkに戻りましょう。空白の削除は、置換関数gsubを使います。

空白を削除
```
$ awk '{ gsub(/ /, ""); print }' hmlt.txt
Hamlet:EntirePlay
...略...
TheTragedyofHamlet,PrinceofDenmark
Shakespearehomepage
...略...
```

gsubは文字や単語の置き換えで使える便利な関数です。

大文字を*に変換
```
$ awk '{ gsub(/[A-Z]/, "*"); print }'
hmlt.txt
*amlet: *ntire *lay
...略...
*he *ragedy of *amlet, *rince of *enmark
     *hakespeare homepage
...略...
```

改行を取り除いて、すべてを1行にしてしまいましょう。

1行に詰め込む
```
$ awk '{ printf $0}' hmlt.txt
Hamlet: Entire Play  The Tragedy of Hamlet,
Prince of Denmark      Shakespeare
homepage       Hamlet       Entire playACT
ISCENE I. Elsinore. A platform before the
castle.FRANCISCO at his post. E
...略...
```

2行ずつ1行にまとめてみます。

2行ずつ1行にまとめる
```
$ awk 'NR%2==1{printf$0} NR%2==0{ print }'
hmlt.txt
Hamlet: Entire Play

The Tragedy of Hamlet, Prince of
Denmark      Shakespeare homepage
     Hamlet       Entire play
...略...
```

テキスト処理ツールの選び方、使いどころ

いかがでしょうか、ほんの少しの例ですが、awkがいかに強力かおわかりいただけたかと思います。awkを使えばこれまでのコマンドのほとんどの機能をカバーできます。

しかし、awkだけ使いこなせればいいのかというと、そうではありません。機能面の使い勝手はもちろん、各コマンドが使うリソースも異なります。小さな仕事は小さなコマンド、複雑な機能もシンプルなコマンドを組み合わせて解決する習慣をつけましょう。

図1のファイルサイズだけでは、実際にプログラムがどのくらいのメモリ領域を使うかはわかりません。確認するには、**size**コマンドを使います(**図2**)。

sizeの出力で、textはプログラム領域、dataは初期化されたデータ領域、bssは初期化されていない変数の領域です。decは10進数での合計、hexは16進数での合計サイズです(サイズを確認したいコマンドが、シンボリックリンクになっていたり、シェルスクリプトから別のコマンドを呼び出したりしていることがありますので、ls -lの結果やfileコマンドで実行可能バイナリ形式であることを確認しておきま

す)。sizeコマンドで得られる情報は静的な情報ですので、実際にプログラムが動作しているときに使用している領域の確認は、topなどを用いるべきです。

特定のソリューションに依存せず、複数の選択肢を持っておくことで不測の事態にも備えのあるエンジニアを目指しましょう。

▼図1 Ubuntu 16.04でのls -lによるファイルサイズの確認

```
$ ls -l /usr/bin/tr /bin/grep /bin/sed /usr/bin/gawk ↵
-rwxr-xr-x 1 root root 211224  4月 29  2016 /bin/grep
-rwxr-xr-x 1 root root  73424  2月 12  2016 /bin/sed
-rwxr-xr-x 1 root root 658144 12月 24  2015 /usr/bin/gawk
-rwxr-xr-x 1 root root  47856  3月  3 03:07 /usr/bin/tr
```

▼図2 プログラムの使うメモリ領域の確認(Ubuntu 16.04)

```
$ size /usr/bin/tr /bin/grep /bin/sed /usr/bin/gawk ↵
   text    data     bss     dec     hex filename
  40401    1100    9408   50909    c6dd /usr/bin/tr
 203548    4312    7840  215700   34a94 /bin/grep
  66552    2240   33232  102024   18e88 /bin/sed
 637401   17504   25240  680145   a60d1 /usr/bin/gawk
```

シェルに解釈されてしまう文字が悩ましい――コマンドラインでテキストツールを巧みに使うためのヒント

シェルが解釈する特殊文字と(sedやawkなどの)呼び出すコマンドで使う特殊文字とで重複しているものがあり、テキスト処理コマンドに期待していない形で文字が加工されてしまうことがあります。呼び出し元のシェルの種類(sh、bash、cshほか)によっても解釈される文字(列)が異なり、上級者でも試行錯誤が必要なやっかいな問題です。

でもこれを乗り越えないと、シェルとテキストツールの連携ができません。克服のためのヒントを記しておきます。

①おかしな動作をしているコマンドラインで、どのようにシェルが解釈しているか、対象のコマンドラインを、コマンドの代わりにechoで引数を表示してみましょう。置き換えが発生している個所を確認できるかもしれません。

②シェルが解釈する文字を意識しましょう。*、$、%、!、"、'などに代表される特殊文字と続く文字列が、どのように解釈・展開されるのかを知っていると、問題を整理できます。bashでは、複数の段階に分けてこれらの文字(列)を処理しますので、文字列展開の順序も重要です。

bashにおける、文字列展開の順序は、1)brace expansion({ }の展開)、2)tilde expansion(チルダ展開:~、~ユーザ名をホームディレクトリパスに置き換えるなど)、3) 次のものは同じフェーズで行の左に現れたものから順に展開:parameter and variable expansion(パラメータと変数展開)、arithmetic expansion(算術展開$((数式))を展開)、コマンド置換($(コマンド)もしくは` コマンド `)、可能な環境ならプロセス展開(<(コマンド)、>(コマンド))、4)単語分割、5)pathname expansion(パス名展開:グロビング[注9]など)です。

brace expansionやグロビングなど、動作を決めるオプションなどで、展開を抑止するなどができます。詳しくはman pageを参照してください。

③解決するには適切な文字(列)のエスケープです。シェルに解釈させたくない文字はエスケープしましょう。'でくくれば、くくられた文字列はシェルによる展開は生じません。特定の文字だけをエスケープしたいなら、\を使いましょう。"でくくった文字列は、スペースなどが含まれていても1つの文字列として解釈されますが、文字列内の特殊文字はシェルによって解釈されます。本稿であつかった例も参考に、連載でもこれまで機会があるたびに取り上げていますので、過去記事[注10]を読み返してみてください。

④とにかくたくさんのスクリプトを書く。経験こそ1番の特効薬です。多くのスクリプトを書いて、sedの1文字コマンド、awkの文法・変数・関数、そして正規表現などに精通しましょう。

注9) グロブ文字を使った文字列式です。連載第4回(2016年8月号)で解説しました。

次回について

次回は、半年に一度のゲーム&ジョーク特集です。SD

注10) 第4、10、11回。それぞれ2016年8月号、2017年2、3月号。

今回の確認コマンド

【manで調べるもの
(括弧内はセクション番号)】
sed(1), ed(1), ex(1), locale(1), awk(1), grep(1), size(1), bash(1)
【infoコマンドを使って確認】
ed, sed
【書籍で確認】
『AWK実践入門』技術評論社

第65回

Linux 4.4の変更点
メモリ領域のロック遅延とSYNパケット処理の効率化

Text：青田 直大　AOTA Naohiro

7月3日にLinux 4.12がリリースされ、15日にはもうLinux 4.13-rc1がリリースされています。oldconfigの限りでは、気になる新規設定項目はなかったのですが……こういうときには意外と内部が変わっていたりしてかえっておもしろいこともあります。どんな機能が入っているのか楽しみです。

今月はまだ紹介していないLinux 4.4の機能から、メモリ領域のロックを遅延する機能と、TCPにおけるSYNパケットの処理を効率化した変更について解説します。

mlock2 VM_LOCKONFAULT

mlock()は、指定したメモリ領域を物理メモリ内に「ロック」するシステムコールです。そのメモリ領域は、アンロック、プロセスの終了、ほかの実行ファイルのexecのいずれかを行うまでの間、物理メモリに常駐し、スワップアウトされなくなります。

mlockには2つの代表的な用途があります。1つはセキュリティの用途です。暗号化に使う鍵や作業バッファをmlock()しておくことで、swapに鍵や暗号の作業経過が洩れてしまうことを防ぎます。もう1つは、パフォーマンス用の用途です。大規模な計算で作業バッファを常にメモリに載せておけば、ディスクから読みに行くことがなくなり、パフォーマンスの改善が期待されます。もちろん、レイテンシもより予測しやすくなるので、リアルタイムの用途にも使いやすいでしょう。

これまでmlockは呼び出した時点で、指定したメモリ領域をすべてメモリに載せていました。しかし、場合によってはこの動作が「もったいない」ときがあります。たとえば、ある暗号化ソフトが最大では10GBの作業バッファを使うとしましょう。このソフトがいつでも10GBのバッファを使うのであれば、今のmlockでかまいません。これがもし、たいていの場合は1MBも使わないのであれば、10GB近くのメモリ領域が無駄に物理メモリに固定され、まったく割り振られずに解放されることになります。

ハイパフォーマンス処理の用途でも同様なことが起こり得ます。たとえば、巨大なグラフィックモデルを処理するプログラムがあるとしましょう。グラフの局所性から一度アクセスした部分はメモリに載せておくとパフォーマンスを向上できます。その一方で、モデル全体を使うわけではないのでモデル全体をmlockするのは無駄になるというケースです。こちらのケースでは、モデル全

体をmlockすると0埋めでもよい作業バッファとは違ってそのデータを読み込む初期コストもかかります。

原理的には、これを少しずつmlock()していくこともできますが、どこをmlock()したのかを管理するのもたいへんですし、システムコールのコストもかかります。そこで、mlock()を拡張して、「次にメモリに載ったときにロックする」機能が追加されました。

この機能を実現するmlock(MLOCK_ONFAULT)の効果を見てみましょう。**リスト1**のようなコードを使い、mlock()の方法が異なる3つのプログラムを作ります。どのプログラムも5GBのファイルをmmap()して、そのバッファ内をランダムに読み出します。その際、アクセスする前にバッファをmlock()しています。mlock()の方法はそれぞれ異なっており、1つは最初にバッファ全体をmlock()する方法、1つはアクセスの直前に該当領域をmlock()する方法、最後の1つはmlock(MLOCK_ONFAULT)を用います。それぞれのプログラムにおいて、mmap()とmlock()にかかるsetupの時間と、バッファへのランダムアクセスにかかるrunの時間を計測します。何度ランダムアクセスを行うかは、プログラム

▼リスト1　mlock()の方法が異なる3つのプログラム

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/syscall.h>

#define SIZE (5UL << 30)

void timespec_diff(struct timespec *start, struct timespec *stop,
                   struct timespec *result)
{
    if ((stop->tv_nsec - start->tv_nsec) < 0) {
        result->tv_sec = stop->tv_sec - start->tv_sec - 1;
        result->tv_nsec = stop->tv_nsec - start->tv_nsec + 1000000000;
    } else {
        result->tv_sec = stop->tv_sec - start->tv_sec;
        result->tv_nsec = stop->tv_nsec - start->tv_nsec;
    }

    return;
}

int main(int argc, char *argv[])
{
  if(argc < 2)
    return 1;

  unsigned long CNT = strtoul(argv[1], NULL, 10);
  srand(time(NULL));
  printf("access %lu\n", CNT);

  struct timespec start;
  clock_gettime(CLOCK_REALTIME, &start);

  int fd = open("file", O_RDONLY);
  char *buf = mmap(NULL, SIZE, PROT_READ, MAP_SHARED, fd, 0);
```

➡ 次ページに続く

前ページの続き ➡

```
#ifndef EACH
  int flag = 0;
#ifdef ONFAULT
  flag = 1;
#endif
  if(syscall(SYS_mlock2, buf, SIZE, flag)) {
    perror(NULL);
    return 1;
  }
  printf("locked: %p\n", buf);
#endif

  struct timespec setup;
  clock_gettime(CLOCK_REALTIME, &setup);

  for (unsigned long i = 0; i < CNT; i++) {
    volatile char x;
    unsigned long pos = ((unsigned long)rand() * 3) % SIZE;
#ifdef EACH
    if(mlock(buf + (pos & 0x1000), 4096)) {
      perror(NULL);
      return 1;
    }
#endif
    x = buf[pos];
  }

  struct timespec end;
  clock_gettime(CLOCK_REALTIME, &end);

  struct timespec setupdiff, rundiff;
  timespec_diff(&start, &setup, &setupdiff);
  timespec_diff(&setup, &end, &rundiff);

  printf("setup: %ld.%09ld\n", setupdiff.tv_sec, setupdiff.tv_nsec);
  printf("run: %ld.%09ld\n", rundiff.tv_sec, rundiff.tv_nsec);

  close(fd);
  return 0;
}
```

の引数で制御します。

まず、setupの時間から見ていきましょう(**表1**)。「**最初にmlock()**」のプログラムとそのほかの2つでは、setupの時間が大きく異なることがわかります。「**最初にmlock()**」では、ファイルの内容を読み込んでメモリに載せている分、時間がかかってしまうというわけです。細かく見れば、**mlock(ONFAULT)**のほうが「**都度mlock()**」のプログラムよりも、setupに時間がかかっています。これも「**都度mlock()**」では何もしていないのに対して、**mlock(ONFAULT)**では一度はシステムコールを呼び出して、ページフォルト時にロックする設定をカーネル内で行っていることを考えれば自然な結果です。それでも**mlock(ONFAULT)**は、「**最初にmlock()**」に比べて大幅にsetupの時間を削減できていると言えます。また、どのプログラムでもアクセス回数ではsetup時間が変動していません。setupに関わる**mmap()**、**mlock()**がアクセス回数に関係ないことを考えればこれも至極当然の結果です。

次にrunの時間を見てみましょう(**表2**)。setupのときとは逆の傾向で、「**最初にmlock()**」がほかの2つよりも顕著(けんちょ)に、短くrunが終わっているのがわかります。これが最初にsetupに時間をかけて、ファイルをメモリに載せたことでアクセスが速くなったことの効果です。残り

の2つのプログラムについて見てみましょう。どのアクセス回数においても、mlock(ONFAULT)のほうが「都度mlock()」よりも速い結果となっています。また、アクセス回数が1,000回のときには5%ほどであった差が、アクセス回数が1,000,000,000のときには90%近くにまでなるというように、アクセス回数が増えるにつれて差が広がっていくという傾向が見てとれます。これは「都度mlock()」が毎回システムコールのコストを支払って領域のロックを試みているのに対して、mlock(ONFAULT)ではフォルトが起きたときにカーネルが自動的にロックをしてくれることによる差が出ていると言えます。アクセス回数が増えるにしたがって、ページフォルトが起きない確率が高まり、2つの差が開いていく要因になる、というわけです。今回は完全にランダムにアクセスしていましたが、局所性のあるデータの場合にはより少ない回数でも、こうした傾向が現れてくることでしょう。

総合的にみれば、今回のプログラムではアクセス回数が1,000回のとき以外は、最初にmlock()のコストを支払ってすべてメモリに載せてしまうほうが良い結果が出ています。とはいえ、mlock(ONFAULT)が「都度mlock()」よりも高効率でかつ「最初にmlock()」のように最初の固定コストがかからないことを考えると、アクセス回数がわからないケース、より局所性のあるケースでは使いやすいのではないでしょうか。

さて、今回のプログラムでは「最初にmlock()」、「都度mlock()」そしてmlock(ONFAULT)を比較しましたが、パッチの作者はさらにシグナルハンドラを使った実装とも比較を行っています。これはバッファ全体を、はじめはアクセス権限なし(PROT_NONE)に設定しておき、アクセスが起きるとセグメンテーション違反が発生することを利用して、シグナルハンドラでmlock()とアクセス権限の変更を行うという実装です。いわば、mlock(ONFAULT)をカーネルではなく、ユーザ空間で実装したということになります。

パッチ作者の実験によれば、シグナルハンドラで1ページずつアクセス権限を許可し、mlock()する実装では「都度mlock()」よりも2倍ほど遅くなっています。シグナルハンドラ自体が一度カーネルからユーザランドに戻り、シグナルハンドラを実行してからまたユーザランドに戻ると「重い」処理であることを考えると自然な結果です。作者はさらに、シグナルハンドラ内で複数ページをまとめてアクセスの許可・mlock()する実装も試しています。それによれば、16ページまとめて処理することで「都度mlock()」よりも良い結果にはなるようです。ただ、それでもmlock(ONFAULT)には勝ててはいません(まあ、これも局所性のあるパターンであればまた違ってくるとは思うのですが……)。なんにせよ、シグナルハンドラの面倒さも考えるとmlock(ONFAULT)に軍配が上がる結果だと思います。

lockless TCP listener

ご存じのとおり、TCPの接続は3ウェイハン

▼表1　setupの時間

アクセス回数	最初にmlock()	都度mlock()	mlock(ONFAULT)
1,000	36.790229298	0.000012158	0.036673941
1,000,000	37.609418490	0.000012296	0.038661837
1,000,000,000	37.644513284	0.000012191	0.037976124

▼表2　runの時間

アクセス回数	最初にmlock()	都度mlock()	mlock(ONFAULT)
1,000	0.000130222	8.751868252	8.298185494
1,000,000	0.081488180	500.326835691	486.101204963
1,000,000,000	56.311546150	1037.415524951	549.642965671

ドシェイクによって確立されます。サーバ側がlistenしているポートに対して、クライアントはSYNを送ります。SYNを受け取ったサーバはSYN/ACKをクライアントに送り返し、さらにクライアントがACKを返すことで接続が確立されるというわけです。

このとき、SYN/ACKを返してからACKを受け取るまでの間に、サーバはSYNを送ってきたクライアントの情報を管理しています。この管理情報の保持方法を変えることで、Linux 4.4では秒間350万回ものSYNパケットを処理できるようになりました。

まず、Linux 4.4以前でのSYN受け取り後のソケット(SYN_RECV状態のソケット)の管理方法について解説します。SYNパケットが到着すると、カーネルはまずそのSYNパケットの宛先を、なんらかのサーバがlistenしているかどうかを確認しなければなりません。システム上で現在listenしているソケットの情報は**listening_hash**というtableの中に保持されています。カーネルは、このtableから宛先アドレス・ポートをlistenしている**socket(listener socket)**を取得します。

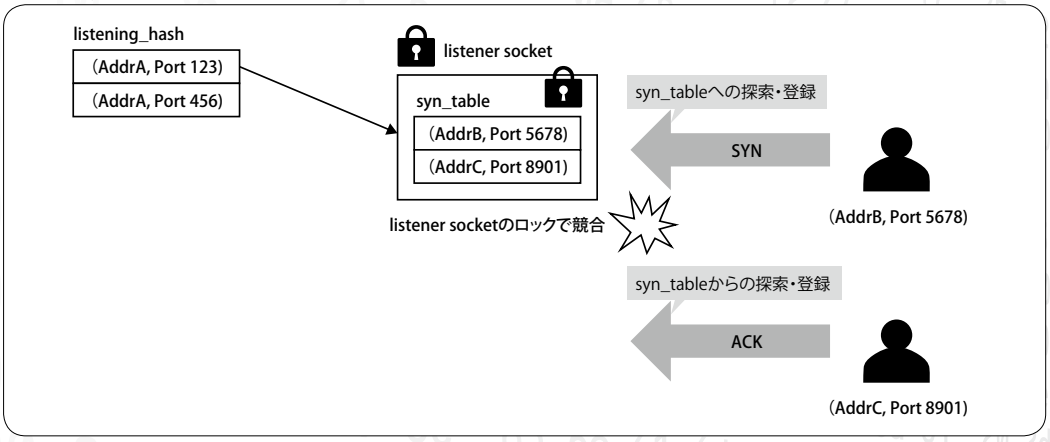
次に、SYNを受け取ったという情報を記録します。この情報は**structrequest_sock**という通常の**struct socket**よりも小さいサイズのsocket構造体の形をとり、listener socket内の**syn_table**というhashtableに保持されます(**図1**)。**request_sock**の保持が終われば、SYN/ACKを返す準備は完了です。サーバはクライアントからのACKを待つことになります。

ACKが返ってくると、ふたたび**listening_hash**からlistener socketを探索し、さらに、このsocketの**syn_table**の中から、返ってきたACKに対応する**request_sock**を探索します。この時点で、**request_sock**を通常のsocketに変換し、**syn_table**からエントリを削除します。このsocketはaccept queueに追加され、サーバアプリケーションの**accept()**を待つことになります。

このデータ構造においては、2個所でロックをとられています。1つはlistener socketに対するロックで、もう1つは**syn_table**に対するロックです。**syn_table**はRCUを使っていないため、どちらのロックも**syn_table**を参照・変更する際に獲得されます。ほとんどのケースでは、socket側のロックだけで十分ですが、/procからの情報取得時にsocketのロックをとらずに、**syn_table**のロックのみで作業できるようにするために**syn_table**のロックはあります。結局のところ、同じポートに対して届くSYNパケット・ACKパケット同士でlistener socketに対するロックを取り合うことになります。

このロックの競合により、Linux 4.4以前では

▼図1　syn_tableの動作



SYNパケットの処理に限界が生まれていました。Linux 4.4では**struct request_sock**の保持方法を刷新することで、listener socketに対するロックをなくしました。

具体的にどのように構造が変わったのかを見ていきましょう。Linux 4.4以降では、**struct request_sock**がehashというhashtableの中に保持されるようになりました。ehashは、もともとは接続が確立した**(ESTABLISHED)socket**を保持するhashtableで、送受信アドレス・ポートの4つの組みをキーとします(**図2**)。

struct request_sockをehashに登録することで、どのように処理が変わるのか、Linux 4.4以前の例と合わせて見ていきましょう。まず最初にSYNパケットが到着します。このときの処理は先ほどと一緒です。カーネルは、**listening_hash**からlistener socketを探索してきます。先ほどは、ここでlistener socket内の**syn_table**に**request_sock**を登録していましたが、代わりにehashに保存し、SYN/ACKを返します。

続けてACKが戻ってきます。Linux 4.4以前では、ここでもlistener socketを探索していましたが、実はlistener socketを探索する前にアドレス・ポートが一致するehashのエントリがあるかどうかが先に確認されています。すなわち、**request_sock**がehashに登録されたことで、ACKではlistener socketを見ることなく、**request_sock**を取得できるようになったわけです。

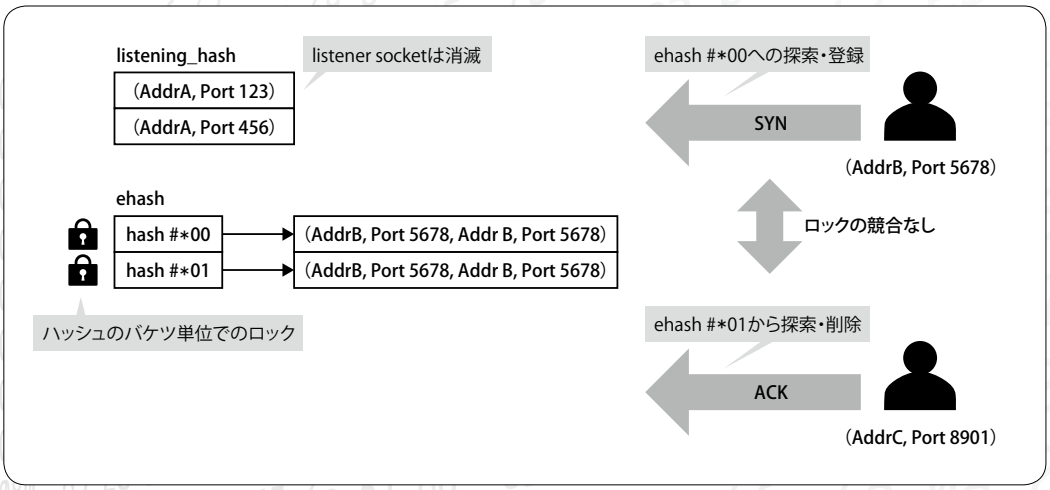
ehashの場合は、ロックはどうなるのでしょうか。ehashは、送受信のアドレス・ポートの4つの組みから計算するハッシュ値の下位bitを使って、socketを適当な数のバケツに割り当てます。各バケツの中のsocketは、リスト構造でつながります。このとき、ロックはバケツの単位でとられます。すなわち、ある2つのSYNパケット・ACKパケットがロックの競合を起こす確率は1/(バケツ数)となります。これは接続先が一緒であれば、必ずロックが競合していた以前の実装とは対照的です。

このようにして、Linux 4.4ではehashに**request_sock**を保持することで、listener socketのロックをなくし、大量のSYNパケットをさばく際のロック競合を大きく減らすことに成功しました。

まとめ

今回はLinux 4.4の変更点から、**mlock()**をページフォルトまで遅延する**mlock(ONFAULT)**と、TCPのlisten socketをlocklessにした変更について解説しました。SD

▼図2　listener socketを介さないehash



September 2017

NO.71

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
榎 真治　ENOKI Shinji　enoki-s@imail.plala.or.jp
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp

「ITコミュニティ運営を考える」セッションの新たな試み

今回は、5月に名古屋、6月に沖縄で行った研究会の模様をお伝えします。

jus研究会 名古屋大会

■ITコミュニティの運営を考える
【講師】法林 浩之（日本UNIXユーザ会）、
榎 真治（日本UNIXユーザ会／LibreOffice日本語チーム）
【日時】2017年5月27日（土）14:00～14:45
【会場】名古屋中小企業振興会館

今年の名古屋大会は昨年同様に「ITコミュニティの運営を考える」をテーマとして行いました。このテーマを思い立ったのが2年前の名古屋大会で、スタートしたのはその2ヵ月後からですので、ほぼ2周したことになります。参加者は6名でした。

まず法林さんから、これまでに行ってきた「ITコミュニティの運営を考える」セッションのまとめについて報告がありました（次の沖縄大会の節で報告内容の詳細を後述）。

その後、参加者全員で輪になって座ってディスカッションを行いました。OSSコミュニティのリーダーや、大学や高校の先生、エンジニアでコミュニティに興味がある方、企業でOSSコミュニティをサポートされていた方など多彩なメンバーで議論しました。

■レベルの低いプルリクエストにどう対応するか

「レベルの低いプルリクエストにどう対応するか」というお題が出されました。そのまま取り込むわけにもいかず、いきなりクローズするのも感じが悪いとのことです。

お題を提示された方によると、そのOSSには1年間で100近いプルリクが来るそうでメイン開発者2名で対応するにはたいへんそうです。

筆者（榎）の所属するLibreOfficeは、開発に参加するハードルが高いこともあって、貢献者を育てる意識が強いコミュニティです。そのため、コードレビューで丁寧にコメントして育てる開発者が多いことを紹介しました。プルリクを送ってくれる人の中で、できそうな人に権限を渡してレビューを手伝ってもらうことがよくある手であり、良いかもしれない、という議論になりました。

■勉強会のレベルの高低がわからない

参加する立場では、レベル感がわからずに参加して良いのか躊躇（ちゅうちょ）するというお題も出されました。初心者向けなどレベルが明確にわかるようになっていると良さそうです。ただ、コミュニティが続いていくと徐々にマニアックになる傾向があり、初心者向けを維持するのが難しいという話もあります。Slackなどで初心者向けチャンネルを作り、質問用であることやマサカリ（技術的な事柄について厳しく指摘／批判すること）禁止であることを書いておくと良いかも、というアイデアが出ました。

開発者とは違う立場で熱心にサポートしてくれるメンバーがいると助かるという話になり、質問へのサポートが得意な貢献者を探すことは重要だという話に展開しました。

■コミュニティに入るときのハードルを下げるには

「コミュニティに入るときのハードルを下げるには」というお題もありました。最初に参加するきっかけとして、すでに参加している誰かに誘ってもらうケースが多そうという議論になりました。参加者を増やすには現在のメンバーがまわりの人を誘うこと、参加したいなら参加している人を探すのが良いかもしれません。

また、懇親会に参加すると交流しやすい、誘うときには懇親会込みで誘う、Facebookでつながって時々いいねを押しておくと次に会ったときにも話しやすいという意見がありました。

◆ ◆ ◆

今回は、これまでのパネルデスカッションをやめて参加者全員で議論してみました。参加者数は少なかったものの、「開発話がもりあがった」「参加者視点の話が掘り下げられた」「これまでにないお題が出された」「活発な議論が行われた」などの点が良かったと感じました。今後も参加者全員によるグループディスカッションを行っていこうと考えています。

jus研究会 沖縄大会

■ITコミュニティの運営を考える
【講師】法林 浩之（日本UNIXユーザ会）
【日時】2017年6月17日（土）15:15～16:00
【会場】沖縄県市町村自治会館

■過去のセッションのまとめ

沖縄大会も、昨年に引き続き「ITコミュニティの運営を考える」セッションを実施しました。参加者は11人でした。

はじめに筆者（法林）から、これまでに実施してきた「ITコミュニティの運営を考える」セッションのまとめを報告しました。2015年7月からの2年間で計13回のセッションを実施しており、パネリストとして登壇したコミュニティは合計31団体にのぼります。セッションでパネリストから出た議題はたくさんありますが、分類すると、

- 存在を知ってもらう方法
- 仲間を増やすには
- コミュニティを長く続けるには
- 内部の体制作り
- 外部との関係
- イベント／勉強会運営のノウハウ
- 運営者のあり方

といったものに分けられます。このうち「存在を知ってもらう方法」については、具体的な議題とコメントも紹介しました。

■グループディスカッション

その後、参加者のみなさんを2グループに分けて、コミュニティ運営について自由にディスカッションを行いました。そしてセッションの最後に、各グループの代表者に議論の内容を報告してもらいました。片方のグループは1～2名がコミュニティ運営経験者で、残りはおもに参加する側の人でした。こちらのグループでは「なぜこのセッションに参加したのか」「コミュニティやイベントをずっとやっていても新しい人が入ってこないと平均年齢が上がるので、新規加入者やリピーターを増やすにはどうすれば良いか」などについて話し合いました。

もう一方のグループは1人だけが参加者側で、残りは全員コミュニティ運営者でした。こちらでは「コミュニティ運営者はなぜ若い人を呼びたがるのか」という若手からの疑問提示があり、それに関してディスカッションしました。代表として報告したのはその若手の人で、今まではコミュニティの集まりにただ参加しているだけだったが、裏ではいろいろなことを考えながら運営がなされていることを初めて知ったそうです。

◆ ◆ ◆

グループディスカッションは名古屋大会から導入したもので、やり方はまだ試行錯誤の余地がありますが、参加者のみなさんにコミュニティ運営が他人事でないと感じてもらうには良い方法だと思っています。今後も継続して開催予定です。SD

あなたのスキルは社会に役立つ

2011年3月11日の東日本大震災発生の直後にHack For Japanは発足しました。今後発生しうる災害に対して過去の経験を活かすためにも、エンジニアがつながり続けるためのコミュニティとして継続しています。防災や減災、被災地の活性化や人材育成など、「エンジニアができる社会貢献」をテーマにした記事をお届けします。

第69回
高齢者ならではの視点が開発の幅を広げるシニアプログラミングネットワーク企画

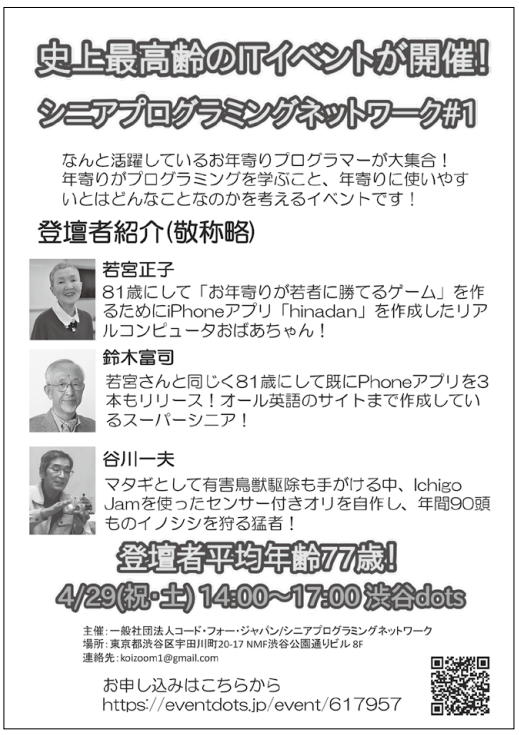
●Hack For Japanスタッフ　小泉 勝志郎（こいずみ かつしろう）　Twitter @koi_zoom1

はじめに

みなさんAppleの開発者の祭典「WWDC 2017」のキーノートはご覧になったでしょうか？　iOSやMacの新機能が多く紹介されましたが、WWDC 2017の冒頭で、ある日本人が大きく紹介されたのです。その方の名は若宮正子さん。2017年2月当時81歳でiOSアプリ「hinadan」をリリースした最高齢開発者としての紹介でした。実は筆者は若宮さんのアプリに、企画開始からリリースに至るまで関わっているのです。

そして、このWWDCの直前に筆者が企画して開催したイベントが、2017年4月29日に開催した「シニアプログラミングネットワーク #1」。今回の記事ではその模様をお知らせします。

▼写真1　シニアプログラミングネットワークのチラシ



史上最高齢のITイベント開催！

Twitterで筆者が配ったあるチラシが2万を超えるリツイート数となっていました。「いちばんやばいチラシ」としてTweetされたそれはどんなものなのかというと「登壇者平均年齢77歳！　史上最高齢のITイベント」というものでした（**写真1**）。そう、今回の記事で扱うシニアプログラミングネットワークのチラシだったのです！

イベントスタート！

シニアプログラミングネットワークは渋谷にあるdots.（現在は名称変更してTECH PLAY）で開催しました。ネットで話題になったおかげかすぐに満員御礼となり、増席をしたほど。

集まったのは100人ほど（スタッフ、登壇者、メディア含む。一般来場受付81人）。アンケート結果を見ると年齢層は10代から80代まで幅広くいます。一番多いのは40代ですが、あまり山がはっきり出ず満遍なくさまざまな世代が来ているという、ある意味イベントには珍しい客構成です。40代が多いのも

単に40代は人口が多いからかもしれません。今回話題となるのがシニアのため、シニアが多く来るのは想像できましたが、Twitterで2万リツイート越えをするほど話題になったため、若い人も多く来たと思われます。登壇者の親類の方で80代の方も！

若宮正子さん

```
while (Japan.recovering)
    we.hack();
}
```

若宮さんと、しかけ人でもある筆者のペアで登壇しました(**写真2**)。この項目だけ自分のこともあるので主観も入ることをご容赦ください。

大きく話題となった「hinadan」、お年寄りが若者に勝てるゲームというコンセプトがすごくよかったと思います。6年ほど前に筆者が若宮さんと初めて話をしたときに、若宮さんが筆者に言ったのは「年寄りが若者に勝てるゲームを作ってください。若い人との交流でゲームをすることがあるけど、必ず負けるのがおもしろくない！」というものだったのです。言われてみれば確かにそうだよなとすごく印象に残っていたため、それ以降も会うたびに「年寄りが若者に勝てるゲーム」の話をしていて、2016年2月14日にほかのお年寄りの方も交えて「どういうゲームならお年寄りが若者に勝てるのか？」というブレインストーミングを仙台で行いました。この中で生まれたアイデアが「雛人形を雛壇に飾るゲーム」だったのです。

この段階ではまだ筆者に作ってくれという話だったのですが、「私が作ったところで話題にならないけど、若宮さんが作ったら超話題になりますよ」と伝えて若宮さんが作ることになりました。このようにアプリの開発以前での過程、とくにアイデア出しから関わっているのもあって、宮城県塩竈市在住の筆者が神奈川県藤沢市在住の若宮さんを教えたというわけです。

「ひな祭りには絶対にリリースしたい」という若宮さんからの強い要望もあり、なおかつSkypeなどを利用した遠隔の教育でどうやって間に合わせるのか？　そして、筆者が手伝って作っちゃったら話題性を低くしてしまうので、完全に本人が理解して実装できるようにすること。それもまったくのプログラム初心者が半年で！

▼写真2　若宮正子さん

ここで筆者が取った方法は次のものです。

- 余計なことをいっさい教えない

  カメラや地図、そしてTableViewも教えていません。それどころか配列すら教えるのをやめています。必要ないものは当然のこと、「若宮さん自身が理解して期日までに実装できる」という範囲に収めるためです。ソースをきれいにもしていません。本誌は技術者が多いので異論は多そうですが(笑)。

- ひな祭りに間に合わせるために、機能をどんどんあきらめる

  これも期日に間に合わせるため。勝敗を決めるためのタイマー機能、何度も遊べるようにするためのシャッフル機能もあきらめています。

今回のやりとりはすべて録音してあるので、実際に若宮さんが作ったという証拠もきちんと残っています！　リリース間際からは遠方から来てもらったり、筆者が行ったりの対面指導を頻繁に行い、なんとかリリースできました。

そして、完成した「hinadan」で若宮さんは一躍時の人に！　テレビや雑誌にも出るし、韓国で講演したりと大活躍。その後の活躍は本人から語っていただきました！

鈴木富司さん

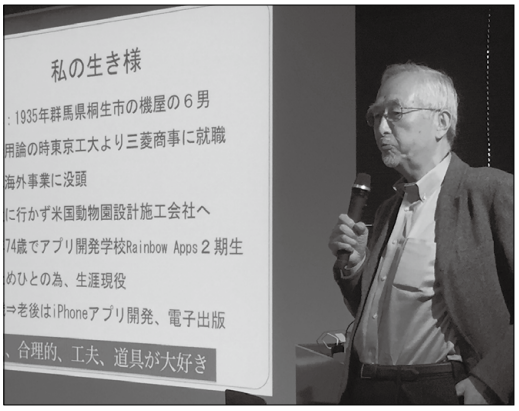
```
while (Japan.recovering)
    we.hack();
}
```

WWDC 2017で若宮さんは「最高齢開発者」として紹介されました。鈴木富司さんは若宮さんの10

日後の誕生日で、なおかつすでに5本もアプリをリリースされています（**写真3**）。若宮さんがリリースするまではおそらく長きに渡り最高齢のiPhoneアプリ開発者だったのではないでしょうか？

その鈴木富司さんが作ったアプリは「スマホの勉強」シリーズという「お年寄りがiPhoneの使い方を勉強するためのアプリ」。設定のところではどういったことができるのかなどを動画を交えてアプリから見ることができます。動画のナレーションは鈴木さんご本人が吹き込んでいて、ダンディな音声も楽しめます（笑）。おもしろいのが「応援者の頁」です。お年寄りだけでは難しいところを周りの人が支援してあげるために、「ここに来たら周りの人に聞け！」というのをまとめてあり、お年寄りの周囲の人もどこで詰まったのかがわかりやすくなっているのです。

▼写真3　鈴木富司さん



▼写真4　谷川一男さん

鈴木さんいわく「そんなに工数かけるな！」と言われないのがシニアプログラミングのメリットとのこと。今後も「スマホの勉強」シリーズを展開していくそうです。

谷川一男さん

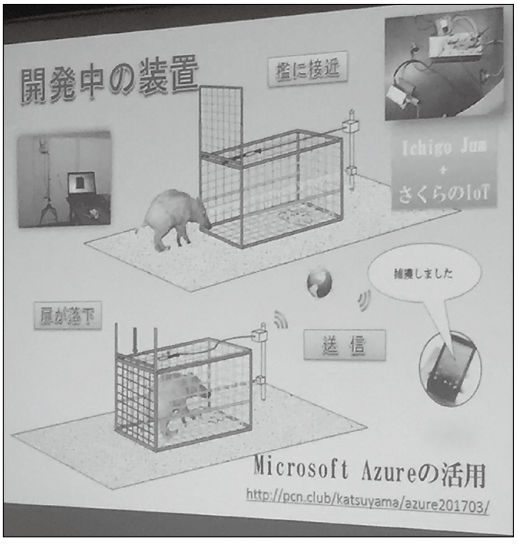
```
while (Japan.recovering)
    we.hack();
}
```

メディアでは若宮さんがWWDCに出たのもあってこちらの報道が多いですが、実はTwitter上で一番人気だったのはこの谷川一男さん（**写真4**）。なんとIchigoJamを利用したセンサー式の檻を利用することで、以前もの何倍もイノシシを狩猟したというその武勇伝が「ITマタギ、マジか？」と大きく話題になっていたのでした。

猪は警戒心が強く、従来の檻罠では成猪の警戒を招くので捕獲が困難。センサーで作動する罠なら猪に警戒されにくい。しかし超音波センサーは聞こえる可能性があるので赤外線を採用したとのこと。そして得た成果。平泉寺地区の捕獲状況を見ると平成27年度は17頭、そして平成28年度にはそれがなんと93頭へ！　そしてIchigoJam全ソースコードも公開されました。なんとたったの9行！

次バージョンの構想もあり、さくらのIoTとMicrosoft Azureを使い、罠が作動したら携帯に通知が来るということを進めているそうです（**写真5**）。

▼写真5　現在開発中の檻



懇親会

なんとイベント当日は鈴木富司さんの82歳の誕生日！　その10日前に誕生日を迎えた若宮さんと一緒に懇親会ではケーキで誕生日を祝いました！（**写真6**）

▼写真6　鈴木さんと若宮さんの誕生日ケーキ

シニアプログラミングネットワークをはじめた理由

なぜシニアプログラミングネットワークを立ち上げたのか？というと、「プログラミングをするシニアが増えることで、シニアの知見をより社会に反映していきたい」という思いからです。

たとえば、筆者が若宮さんのhinadan開発に関わる中で「お年寄りは指が震えるからドラッグ＆ドロップが苦手」などの知見を得ることができています。シニアがプログラミングをしていくことで、こういった知見がより社会に出て行く。お年寄りが使いやすいものはお年寄りが知ってるんですよね。また、絵や音楽に比べると才能の割合が小さいので、いろいろな方が新しく出てくるのではないかと思っています。そのためには、シニアがプログラミングをよりやりやすくする環境を作っていきたいという思いです。また、プログラミングを楽しんでいるシニアが互助できるようにすることで、より長くプログラミングを楽しんでもらいたいという意図もあります。

そして、現在若宮正子さんのことが各メディアに報道され、筆者もオマケで出てたりはしますが、これは一過性の現象だと思っています。これを一過性にせず「シニアプログラミング」というジャンルを作り出したく「シニアになってからプログラムを始めたけどすごい成果を出している人たち」を集めてのイベントにしました。

今後の展開

シニアハッカソンというイベントを行おうと思っています。「お年寄りが自分が作りたいアプリのアイデアを生み出し、その実現を技術者がフォローする」ということを実現するためのイベントです。

今回若宮さんが短期間でhinadanを完成させることができたのは「ゴールから逆算して余計なことをいっさい教えない」という方針がうまくいったのも大きな要因だと思っています。若宮さんはプログラミングのスキルは完全に初心者レベルですが、hibadanを作るのには必要最低限のことは学んだため、ちゃんと自分で作ったものが世に出たわけです。この「作りたいというゴールから逆算してサポートする」というしくみを人為的にイベントの中で作り出したいと考えています。

最後に

若宮さんの件が多く報道されたため、プログラミングに興味を持つ人は増えて来ている印象です。この興味を持った方達の後押しや支援を、シニアプログラミングネットワークで行えればと思います。シニアプログラミングネットワークでは高齢者向けプログラミングのイベントの実施などを行うスタッフを募集しています。希望者の方はシニアプログラミングネットワークのFacebookページに「スタッフ希望」と書いてご連絡ください！　TECH PLAYの部活動にもなっているので定期的にイベント開催が可能です。「もくもく会」のようなイベントも行いたいのですが、筆者は東北在住のため、ぜひ首都圏の方にご協力いただけるとありがたいです！　お待ちしています！ SD

温故知新
IT むかしばなし

第69回

拡張メモリ ～限界の壁をどのように乗り越えてきたのか

速水 祐（はやみ ゆう） http://zob.club/ twitter @yyhayami

はじめに

1980年代のパソコンでは、いかにメモリを増やして効率的に使うかに知恵を絞っていました。今回は、Windowsが普及する以前の拡張メモリについての話をしましょう。

メモリはどのように増設されてきたか

1980年代当初の8bit CPUを搭載したパソコン（以下、8bitパソコン）は、16bitのアドレスバスによる64KBが扱えるメモリの限界でした。その64KBのメモリエリア中の半分程度をBASIC ROMなどが占め、プログラムなどで利用できるのは残りの32KB以下の狭いメモリエリアでした。その中に収まるように、ギリギリまでサイズを切り詰めてユーザプログラムを作成していたのです。

その後に登場した16bit CPUを搭載したパソコン（16bitパソコン）は、この限界を突破できるアドレスバスを備えており、IBM-PCやPC-9801に搭載されたIntel i8088/i8086では、アドレスバスが20bitで、論理的には8bit CPUの16倍の1MBのメモリが扱えるようになっていました。

1982年に登場したPC-9801では、128KBのメモリを標準搭載しており、そのRAMエリアは登場時には広く感じたこともありました。i8086の1MBのメモリエリアの中で、PC-9801/IBM-PCは後ろの340KBをBIOS ROMやビデオRAMで使用し、前の640KBをRAM（基本メモリ領域）と使用できる構成になっていました（**図1左**）。1984年11月に登場するPC-9801Mでは640KB中の256KBのRAMが標準搭載されました。

PC-9801Mでは、残りの384KB（640KB－256KB＝384KB）を埋めるために、1MBの拡張メモリボードを入れると640KB（1,024KB－384KB＝640KB）が残ることになります。その分のメモリは、RAMエリアの裏側のバンクメモリ[注1]として利用[注2]することになります。

バンクメモリは、I/Oポートにバンク番号を書き込むことで切り替えが行われますが、当時はメモリボードのメーカーが異なると、このI/Oポートや書き込む値も違っていました。そこでメモリボードを発売していた主要メーカーの1つであるアイ・オー・データ機器がポート番号などを統一規格化して、I・Oバンクメモリとして、普及していくことになります。

I・Oバンクメモリ

I・Oバンクメモリは、基本メモリ領域内にバンクエリアがあることが問題でした。実行プログラムがバンクエリア内にあった状態で、バンク切り替えを行うと不具合が生じるため、プログラムがバンクエリアに重ならないように使用エリアを制限するなどの対策が必要でした。

1985年に登場したPC-9801VMでは、標準搭載メモリが384KBとなりました。翌年に発売されたワープロソフト「一太郎Ver.2」が640KBのメモリを必要としたため、それを動作させるためにメモリボードが大きく販売を伸ばしました。メモリボード上の256KBを基本メモリ領域

注1）特定のメモリ領域に複数のメモリ（バンク）を割り振り、切り替えて利用する方法。

注2）バンクメモリは、RAM上位の8000H～A0000Hエリアの128KBを使うのが一般的でしたが、256KBや512KBをバンクメモリの単位とするメモリボードも存在していました。

に割り当て、残りはI・Oバンクメモリ方式のRAMディスクとして使われることになります。

PC-9801VMでは2台のフロッピードライブ構成で、1台目に一太郎の実行ファイルが入ったFD(フロッピーディスク)を置き、2台目に辞書ファイルが入ったFDを使っていました。この使い方ですと日本語変換のたびにFDにアクセスすることになり、変換スピードが遅く、頻繁にアクセス音が生じます。そこで、動作音がなく高速なRAMディスクが使われることになります。パソコンの起動時に2台目のドライブにあるFDから辞書ファイルをRAMディスクに転送し、快適な作業を実現していたのです。

EMSメモリ

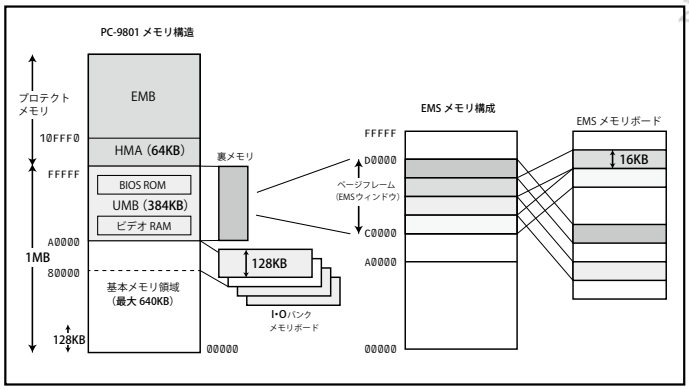
IBM-PCでは、EMS(Expanded Memory Specification)メモリが普及していました。EMSメモリは、16 KB単位(ページ)でバンク切り替えを行います。4ページ(64KB)を連続して割り当てた領域をページフレームと呼びます。各種操作はEMM(Expanded Memory Manager)により隠蔽され、ユーザプログラムはEMMの機能を呼び出すことでEMSを使うことができます。ページフレームは、基本メモリ領域ではなくBIOS ROMなどが存在する上位アドレスのUMA(Upper Memory Area)領域(**図1右**)に設定されるため、アプリケーションソフトウェアからの使用において自由度が高く、ワークエリアとしての利用やプログラムコードを置くこともできました。

ただFM音源ボードROMとのエリアの衝突がゲームユーザにとって大きな問題になることもありました。

PC-9801ではEMSメモリの普及はかなり遅くなりましたが、1989年の「一太郎バージョン4」の登場により大きく利用が広がっていきます。

一太郎バージョン4は、640KBのメモリでは足りず、ワークエリアとして利用できるEMSメモリが必要になってきたのです。また、1986年に登場していた日本語版ロータス1-2-3のユーザが増える時期とも重なり、EMSメモリは一気に利用が広がっていきました。

▼図1　PC-9801のメモリマップ



プロテクトメモリ

PC-9801シリーズのCPUは、i8086からV30を経てi80286が載ることになります。i80286はアドレスバスが24bitとなり16MBのメモリエリアが使えるようになりました。しかし、プロテクトメモリ[注3]へのアクセスは、プロテクトモード[注4]への切り替えが必要となるため、x86リアルモード[注5]で動作するMS-DOS用アプリケーションからは、ほぼ使うことができない状態でした。

x86リアルモードにおいても、かろうじて0FFFFFH〜10FFFHの64KBサイズのHMエリア(**図1**)だけは使うことができたので、MS-DOSの一部システムをHMAに置いていたりしました。

i80286の次に登場する32bit x86CPUのi80386には、プロテクトモードの機能の一部として仮想86モードがあり、プロテクトメモリを効果的に利用できる非常に優れた互換システムが用意されています。仮想86モードを使ってプロテクトメモリをソフトウェアEMSメモリとして利用するEMM386.SYS/EMM386.EXEなどがMS-DOSへ搭載され、i80386マシンの拡張メモリはすべてプロテクトメモリへ移っていきます。SD

注3) 1MB以上のエリアのメモリ(図1参照)。
注4) メモリやI/Oの保護を行い、アドレス空間の拡張を行う動作モード。
注5) x86CPUの16bit動作モード。

NETGEAR ReadyNAS徹底運用

～大切なデータの保護に耐えうるか実力を試す！～

第2回 ReadyNASのバックアップ

NASを運用する最大の目的は情報共有です。これには誰もがうなずいてくれるでしょう。そして、次に来るのがデータの安定的な保存です。わかりやすくいえば、バックアップ先としての役割です。バックアップにはさまざまな考え方があり、その運用方法もまた千差万別です。ここでは、永遠のテーマともいえるバックアップについて、ReadyNASを選択するメリットを考えていきます。

Author 中山 一弘（なかやま かずひろ）

データのバックアップとは?

データは新規作成されることもあれば、更新されることもあります。仕事で使う共有データであれば、自分が使っていなくても、ネットワークにつながる誰かが更新していることも多いでしょう。仕事で作られるデータや共有すべきデータは重要なものばかりです。万が一これらが失われると、ビジネスとしての損失は計りしれません。

筆者のように執筆業を生業（なりわい）にしていると、少なくともテキストは秒単位で新しく発生します。たとえば、上書き保存したデータがあったとして、その数十秒後には何文字か、何百文字かは更新または上書きされているのです。この状況で、何らかのタイミングでデータが壊れれば、最悪の場合にはすべてのテキストが失われ、良くても前回上書き保存したところまでさかのぼらないといけません。さすがに最近はやらなくなりましたが、駆け出しのころは、上書きしたと思い込んでデータを消去、丸一日分の作業をやりなおすなどということもありました(※言い訳になりますが、寝不足が最大の原因でした)。みなさんも、書きかけのプログラムや作りかけのPowerPoint資料、Wordで書いている論文などがそのような憂き目にあえば、かなり悲惨な状況になることはすぐに想像できると思います。

このような失敗を防ぐ意味でも、データのバックアップは大切です。ローカルPCにデータを保存するにしても、マメに上書き保存するのは基本です。筆者の事務所で働く新人には「5分に1回、Ctrl + S」を義務付けています。それでも、一度マシントラブルが起これば、ドライブのデータが無事だという保証はありません。そこで、少なくとも1日1回はNAS上に与えられた自分のフォルダへ重要なデータを保存することを推奨しています。

もちろんこれは一例ですが、データの安定的な保存というのは実に厄介な問題であることに違いはありませんし、ビジネスでは(あるいは趣味でも)とくに気をつけないといけない部分です。紙に比べてデジタルデータは効率が良い反面、データの消失という不測の事態に備える必要があるのです。バックアップの方法は今や星の数ほどありますが、個人事業主、SOHOなどの小規模ビジネスや中小企業では、NASを活用するのが一番汎用性が高く、お手軽かつ安心できる方法でしょう。

ReadyNASのバックアップの特長

単純なバックアップ先としても、ReadyNASにはユニークな特長がいくつかあります。

高速かつ省スペースなブロック単位のバックアップ

1つは、バックアップをファイル単位ではなくブロック単位で行うという点です。一部では使われ始めているものの、中小企業や個人用途のNASとしては画期的です。ReadyNASのReady DR機能を使うことでこれが可能になります。

通常のバックアップではファイル単位でデータがNASに保存されますが、ブロック単位でのバックアップでは、一度保存されたデータであれば、変更があったブロックのみを上書きする

COLUMN

ReadyDRとは？

ReadyDR(**図A**)は、スナップショットベースのブロックレベルによるディザスタリカバリソリューションの略称です。このサイズのNASで本格的なDRソリューションが扱える機種は少ないのでたいへん興味深い機能ですが、ファイルシステムにBtrfsを使用しているため、宛先にUSB HDDやeSATAドライブを指定できない、リアルタイムバックアップができないといった制約があります。

しかし、小規模ビジネスの場合、ディザスタリカバリへの対応はコストやデータの規模などの面から大きな課題がいくつも存在します。いくつか制約はあるものの、ReadyNASさえあれば災害対策が可能になるため、DRソリューションが必要なケースでは大きな武器となるはずです。

▼**図A**　ReadyNAS OSの管理ページの「バックアップ」タブへ移動し、「ReadyDR」機能を「ON」にする。「ReadyDRジョブの追加」で各種設定を行う

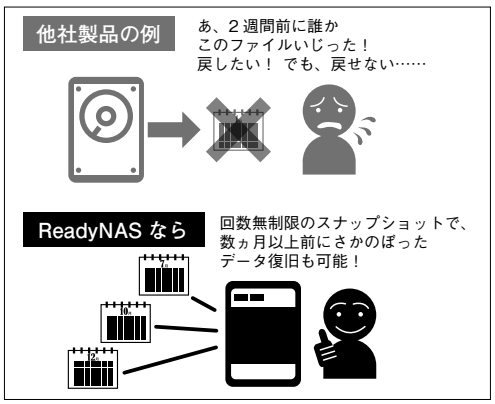
ので、データの転送やディスク上での処理速度が大きく向上します。また、ディスクの容量も最小限で済むため、たとえばフルバックアップを頻繁に行うようなケースでとくにメリットがあります。つまり、バックアップが速く、最小のスペースでできるため、非常に効率が良いのが特長です。

回数無制限のスナップショット

2つめの特長はスナップショットです。「なんだ、それならどんなNASにでも……」と思われるかもしれませんが、ReadyNASではスナップショットの回数が無制限になっています。厳密には、残りのディスク容量に応じて回数に限界はありますが、容量内であれば何回でも可能という意味での「無制限」です。

毎日スナップショットを数ヵ月間取り続ければ、かなり古いデータへさかのぼることもできます。毎時でスナップショットを取れば、ピンポイントで過去のバージョンを呼び戻すこともできます。スナップショットの頻度は業務内容にもよるでしょうが、ディスク容量に応じて回数が無制限になるという安心感は非常に頼もしいはずです(**図1**)。

▼**図1**　回数無制限のスナップショット



保存中のアクシデントにも安心なCopy-on-Write

3つめは、データを保存している最中のトラブルに威力を発揮する「Copy-on-Write(コピーオンライト)」機能です。これまでのNASの多くはデータを上書きしていくタイプのファイルシステムを使っています。この場合、ファイルを保存している最中に停電などがあると、上書き中のファイルそのものが壊れてしまうことがありました。

ReadyNASのCopy-on-Write機能は、元のファイルを残したままファイルを更新します。これによって、ファイルを上書きしているときに何かしらの事故が起こり、データに影響があったとしても、変更する前のファイルへ復旧できます(**図2**)。

この機能のありがたいところは、たくさんのデータや大きなファイルを数多く扱うようなケー

▼図2　Copy-on-Write機能

スでしょう。NASへの保存にかかる時間がそれなりに長いと、こうしたトラブルにあう確率も高くなります。業務でこのような状況が多く発生する場合でも、Copy-on-Write機能があれば保存前の状態にすぐに戻せるというわけです。

ほかにも数々の特長を持つReadyNASシリーズですが、バックアップに関する機能としてこの3つはとくに注目しておいてください。これらを使いこなせば、業務面だけでなく運用面でもさまざまな恩恵があるはずです。

スナップショットの基本操作

さて、バックアップに関する機能だけでもさまざまな特長のあることがおわかりいただけたかと思いますが、誌面に限りがあるので、まずは最も汎用性が高いと思われる回数無制限のスナップショットをもう少し詳しく見ていきましょう。

ReadyNASのスナップショットは、フォルダごとに細かく設定できます。ちなみに、デフォルトでも各フォルダに対してスナップショットを採取するようになっているので、まずはそれを確認してみましょう。ReadyNAS OSの管理画面を開き、「共有」タブから任意のフォルダを指定します。次に右上にある「スナップショット」ボタンを押してください(**図3**)。デフォルトでは1日1回午前0時にスナップショットを取るようになっているため、ご覧のようなリストになっています。

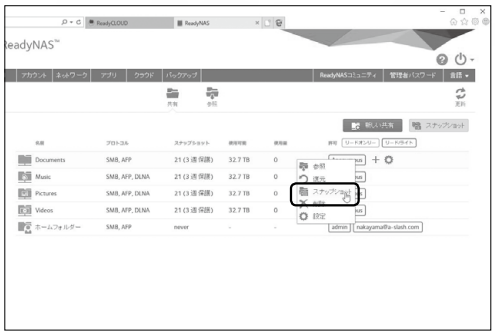
次に手動でスナップショットを取ってみましょう。同じ画面で任意のフォルダを指定し、右クリックしてメニューにある「スナップショット」をクリックします。すると、名前を入力するウィンドウが表示されるので、自分でわかるような文字列を入れてください(**図4**)。そして先ほどのリストを表示すると、リストの最上段に手動で取ったスナップショットが表示されます(**図5**)。

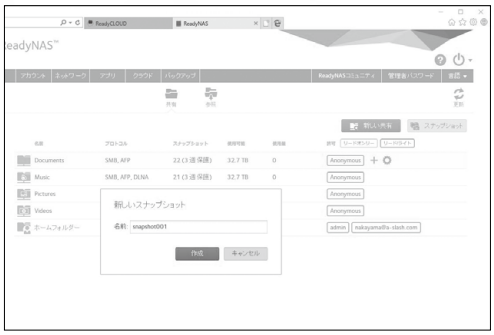
今度はスナップショットを採取するタイミングを変えてみましょう。**図3左**の画面で任意のフォルダを指定して、右にある歯車型の「設定」ボタンをクリックします。出てきた画面の「スナップショット」タブを開いてください(**図6**)。

ここではオートマチックに設定できるスナップショット管理「スマート」と「カスタム」が選べ

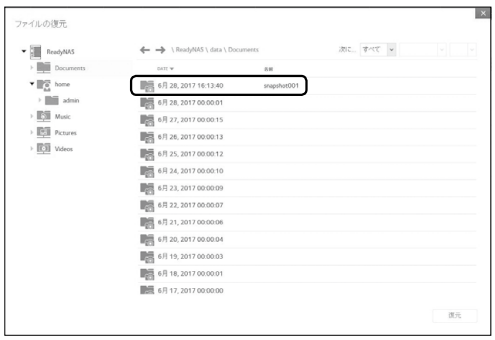
▼図3　任意のフォルダを指定し「スナップショット」ボタンをクリック(左)。該当フォルダのスナップショットのリストが確認できる(右)

▼図4　任意のフォルダを指定して右クリックし、出てきたメニューから「スナップショット」を押して（左）、自分でわかる名前を入力する（右）





▼図5　リストを見ると、今取ったスナップショットが最上段に確認できる



▼図6　任意のフォルダを指定したあと、右にある歯車マークの「設定」ボタンを押して「スナップショット」タブを開く

ます。スマートでは、スナップショットのスケジュールをあらかじめ用意されている毎週、毎日、毎時間の中から選べばOKです。ちなみに、Neverを選ぶとスナップショットが無効になります。このモードを選ぶ際、画面に設定内容の説明文が表示されるので、これを参考にしながら設定するとよいでしょう。実際の運用を考えれば、ここにある設定だけでも十分な選択肢があるはずです。

さらに詳細な設定をする場合は、スナップショット管理を「カスタム」にします。ここでは曜日、時間帯、スナップショットの保持数や期間などを細かく指定できます。

まとめ

基本的なスナップショット管理は、フォルダごとに、手動、スマート、カスタムのいずれかの手法を使うことで、ほとんどのシチュエーションに対応できるはずです。バックアップ用アプライアンスの中心となるNASですから、データを安全に保存するだけでなく、万が一の場合にもすぐに元に戻せることはとても重要です。スナップショットに加えて、Copy-on-Write機能やReadyDRなどを組み合わせれば、物理的なトラブルに備えるケースから災害対策まで、本格的なファイルサーバを運用するのと同等の機能を実現できるでしょう。これと同じことを外部へ委託した場合のコストを考えれば、やってみる価値があるかどうかはすぐにわかってもらえるのではないでしょうか。ビジネスはもちろん、個人利用でもヘビーユースの方はぜひチャレンジしてください。次回も引き続きReadyNASの可能性を探っていきます！SD

トレジャーデータ、エンタープライズ向けデータ転送ミドルウェア「Fluentd Enterprise」を提供開始

トレジャーデータ㈱は7月11日、Fluentdのエンタープライズ版「Fluentd Enterprise」の提供を開始した。

Fluentdは、米トレジャーデータ社の共同創業者、古橋貞之氏が開発したデータをリアルタイムに収集するためのオープンソースソフトウェア（OSS版）で、約5,000社の企業、100万台以上のサーバで導入されている。

Fluentd Enterpriseは、ミッションクリティカルな場面でFluentdを使いたいとの要望を受けて開発されたもので、OSS版よりもセキュリティ面が強化されている。コアになるモジュールはOSS版と共有しているが、別のバイナリとして提供される。転送されるデータはEnd to Endで暗号化され、製品もセキュリティテストをクリアしたものとなっている。データを転送するために用いるプラグインも、同社で検証したり、新たに開発したりしたエンタープライズ向け専用のものが提供される。

さらにFluentdに精通した同社の専門家が、ユーザ企業への導入コンサルティングやトレーニングを提供するほか、24時間365日体制で、SLA（サービス品質保証契約）に基づくサポートを提供する。

日本国内では、各種事業者のほか、SIerやデータセンター事業者などを対象に販売していく。

CONTACT
トレジャーデータ㈱ URL http://www.treasure-data.com

ネオジャパン、カスタムメイド型社内システム構築ツール「AppSuite」を2017年秋に提供開始

㈱ネオジャパンは、業務にかかわるすべての情報処理のシステム化をカスタムメイドで実現する社内システム構築ツール「AppSuite（アップスイート）」を、2017年秋に提供開始することを発表した。

AppSuiteは、ネオジャパンが提供するグループウェア「desknet's NEO」上で、さまざまな業種や業態に対応したアプリケーションを、ユーザ自ら作ることができるツール。「アプリライブラリ」に多数用意されたアプリの中から目的に合ったものを選び、自社の業務や運用に合わせて足りない項目や機能をドラッグ&ドロップなどの操作で追加しながら、アプリを構築していく。

ベースとなるアプリライブラリのアプリは、営業、総務、IT管理、製造業、自治体、医療・福祉、サービス業などの約100種類の中から選べる。

同製品はHTML5などのWeb技術で開発されており、ブラウザ上で利用する。desknet's NEOをプラットフォームとしているため、作成したアプリをdesknet's NEOの一機能として利用可能。アプリで蓄積したデータを集計・グラフ化し、グループウェアで公開もできる。

クラウドとオンプレミスの2通りの提供方法が用意され、価格は販売開始時に発表される予定。

CONTACT
㈱ネオジャパン URL http://www.neo.co.jp

パラレルス、「Parallels Remote Application Server 16」発売

パラレルス㈱は8月1日、「Parallels Remote Application Server（RAS）」の最新バージョン「16」を提供開始した。価格は、販売パートナーを通しての個別見積もりとなっている。

Parallels Remote Application Serverは、仮想アプリケーション・仮想デスクトップをクライアント端末に配信する企業向け製品。サーバ環境としてはオンプレミス、Amazon Web ServicesおよびMicrosoft Azureのクラウドサービスに対応し、クライアント環境としてはWindows、Linux、macOS搭載のPCのほか、iPhoneやiPad、Androidタブレットなどに対応している。「16」では、次のような新機能が追加された。

- PowerShell APIの拡張
- MSPおよび独立系ソフトウェアベンダ向けのホワイトラベル化
- iOSクライアント向けのTouch ID対応
- Quick Keypad機能およびSwiftpoint GT Bluetooth iOSマウスへの対応
- VDI向けのリンククローンの追加
- 複数バージョンのアプリを簡単に管理できるTurbo.netの統合

CONTACT
パラレルス㈱ URL http://www.parallels.com/jp

Event

ITインフラの今を知る、「July Tech Festa 2017」、開催

ITインフラエンジニアのためのテクノロジイベント「July Tech Festa 2017」が、8月27日（日）に開催される。開催時間は10:00から18:30。会場は東京都品川区にある産業技術大学院大学。

今回で5回目となる本イベントのスローガンは「ITエンジニアリングの本質を極める」。技術者は、激変し続けるITトレンドの中から真に自身の業務を変革するものを探しており、本イベントは、そのための情報交換、人脈構築の場になることを目的としている。

セミナーでは、ITインフラの構成管理や運用の自動化と、そのテスト方法の模索、Webサービスの品質や信頼性を守るための取り組み、ネットワーク監視やセキュリティ対策などが、現場で活躍する技術者たちから語られる。

イベント参加はチケット制で、販売中の前売りチケットの販売は8月27日の午前2時まで（売り切れしだい終了）。前売りのチケット代はセミナーのみが1,000円、セミナー＋懇親会が4,000円。当日チケットはセミナー受講のみの販売で2,500円。詳細は以下のチケット申し込みサイトを参照のこと。

・http://jtf2017.peatix.com

CONTACT
July Tech Festa 2017 URL http://2017.techfesta.jp

Report

「JAIPAクラウドカンファレンス2017」、開催

7月19日、コクヨホール（東京都港区）において、日本インターネットプロバイダー協会（JAIPA）クラウド部会主催のクラウド関連事業者向けイベント「JAIPA Cloud Conference 2017」が開催された。本イベントは中立の立場でインターネット利用環境の向上に寄与するJAIPAの存在を広くアピールするもの。参加者は400名を越えるものになった。講演内容の一部は、記事末のJAIPAのWebページからPDFとしてダウンロードできる。

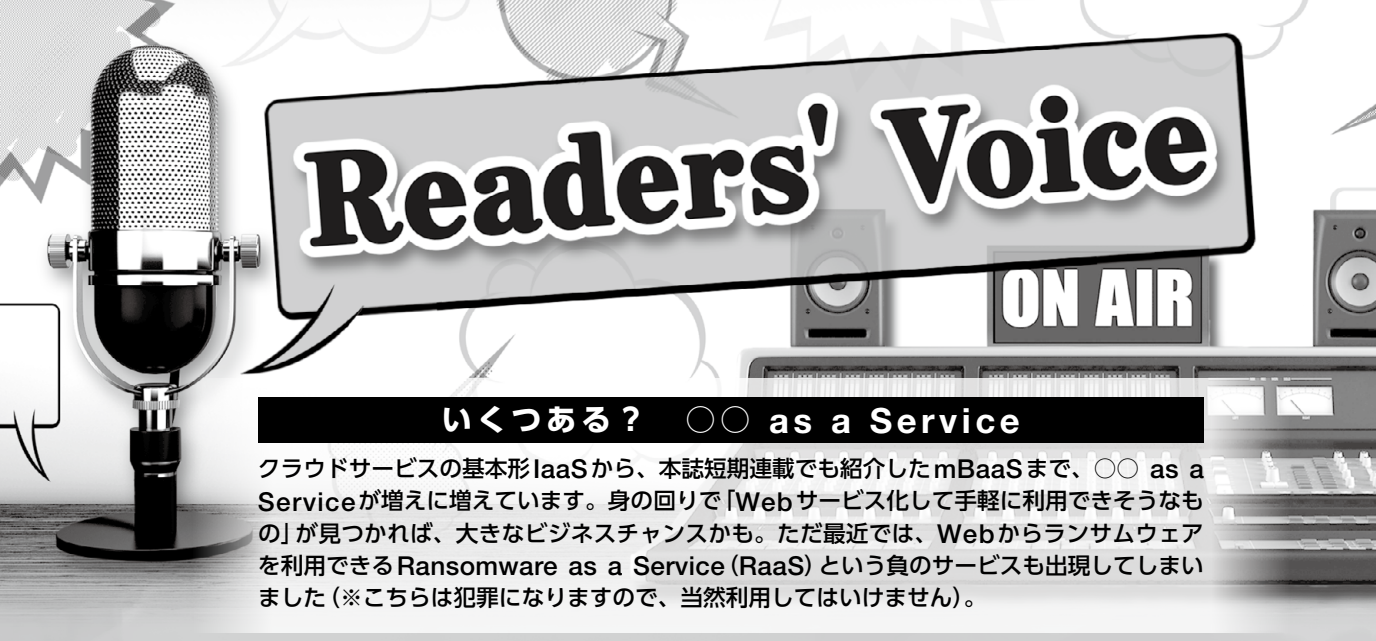
本カンファンレンスは技術者向けの専門性の高い講演がすべてではなく、むしろサイボウズ㈱代表取締役社長 青野慶久氏の発表のように「IT企業は社会にどう貢献すべきか、従業員の働き方はどうあるべきか」といったテーマの講演が多かった。後半に行われたパネルディスカッションでは、㈱三井住友トラスト基礎研究所主席研究員 伊藤洋一氏をモデレータに、青野慶久氏、㈱DGホールディングス代表取締役社長 松栄立也氏、GMOクラウド㈱代表取締役社長 青山満氏、さくらインターネット㈱代表取締役社長 田中邦裕氏、㈱IDCフロンティア代表取締役社長 志立正嗣氏という参加者で、クラウド技術から少し離れて、経営者／技術者の目線で今のビジネスシーンを考えるものとなった。

外資系のクラウド事業者との競合については、サイボウズでは分散オフィスの実現・グループウェアのブランド化、DGホールディングスでは動画ダウンロード直接販売化など、ともに外資がしないことを想定した「逆張り」戦略を採ることで一定の成果を得たという。また、経済産業省が推し進める「働き方改革」を実現するにあたり、社員の副業を許すことで事業を大きく発展させているサイボウズ社の事例をふまえ、各企業がどのような試みを行っているのか話し合われた。田中邦裕氏は自身が今夏1ヵ月にわたる長期休暇をとったことを例として挙げ、まず経営者自らが自分の成功体験を捨て、意識改革しなければ解決しないと提言した。

▲パネルディスカッションの様子


▲閉会の挨拶を行うさくらインターネット㈱田中邦裕社長

CONTACT
JAIPA Cloud Conference 2017
URL https://cloudconference.jaipa.or.jp



いくつある？　○○ as a Service

クラウドサービスの基本形IaaSから、本誌短期連載でも紹介したmBaaSまで、○○ as a Serviceが増えに増えています。身の回りで「Webサービス化して手軽に利用できそうなもの」が見つかれば、大きなビジネスチャンスかも。ただ最近では、Webからランサムウェアを利用できるRansomware as a Service（RaaS）という負のサービスも出現してしまいました（※こちらは犯罪になりますので、当然利用してはいけません）。

2017年7月号について、たくさんの声が届きました。

第1特集　理論＆応用でシェル力の幅を広げる

設計思想やしくみをふまえながらシェルの使い方や機能を解説する「理論編」と、熟練者がさらにシェルの活用範囲を広げられる知見を紹介する「応用編」で、bashをより効率良く使いこなすための術を解説しました。

後半のほうに登場する方々とシェル芸に取り組んでいますが、まだまだですね……。
大西さん／神奈川県

初心者にもわかりやすく、新しい一歩を踏み出せました。
田中さん／和歌山県

新機能が出ても迂闊（うかつ）に飛びつけないというのは残念ですね。POSIXのほうがレベルアップすると良いのですが。
tack41さん／愛知県

雑誌でbashの最新機能を解説する記事というのは初めて見るような気がします。他誌でもなかったですよね、たぶん？
福田さん／神奈川県

BonUonWinが意外に使えるのは、ほんとだと思いました。
体重も増えたさん／奈良県

普段から何げにbashスクリプトを書いているが、込み入ったところまではあまり知らなかったので、非常に有用だった。
tokichieさん／東京都

bashの新機能の紹介は勉強になりました。いつもエラーを記録する1>&2を忘れるのですが、|&なら覚えられそうです。
村橋さん／北海道

今まで手癖で使っていたシェルがよくわかりました。ありがとうございます。
藤田さん／静岡県

初心者・中級者、どちらの読者の方からも勉強になったという声が寄せられました。そのほか、POSIXとの兼ね合いが難しい、bashの最新機能がためになった、という声も多かったです。

第2特集　MySQL［SELECT文］集中講座

SQLの中で最も高い頻度で使われるであろう「SELECT文」にスポットを当て、基本構文からORDER BY、GROUP BYを使ったデータの抽出、各種JOINとサブクエリを使ったデータの加工までと、基本から応用までを一気に解説しました。

ビギナーの人はよく読んでほしい。
西村さん／東京都

「RDBアンチパターン」「RDB性能トラブルバスターズ」の連載もあり、JOIN周りの理解がとても進みました。
のりぃさん／埼玉県

SELECTって基本だろと思いがちですが、大半のSQL文はSELECTなので、とても参考になります。
藤原さん／東京都

桐とかのRDBMSにも触れてほしかった。
しゅさん／熊本県

基本ながらも重要なSELECTに的を絞ったことで、入門に最適な内容となりました。同号では本特集のほかにもSQLを扱った記事が多く、DBの良い勉強になったという声が寄せられました。

一般記事　ハッシュ関数を使いこなしていますか？【後編】

ソフトウェア開発のさまざまな場面で利用されているハッシュ関数を深掘りする前後編です。後編ではハッシュを実際に利用するときの注意点と、Drupal 8における実装例を紹介しました。

7月号のプレゼント当選者は、次の皆さまです

①Wi-Fiホームルータ「Aterm WG1200HP2」
dede様(神奈川県)

②USBメモリ「600-3TC16GN」
西村浩晃様(広島県)、ねっとまん様(埼玉県)

③すごーい！ オライリーオリジナル サーバルTシャツ
坂尾智様(東京都)、オトト様(神奈川県)、
ジューC言語様(大阪府)

④『プログラミング道への招待』
黒田圭介様(宮崎県)、延原正規様(兵庫県)

⑤『純粋関数型データ構造』
夘月翔平様(埼玉県)、renren様(東京都)

⑥『Java本格入門』
島本泰輔様(石川県)、ryoka様(福岡県)

⑦『Intel Edisonマスターブック』
小林采豊様(神奈川県)、鈴木浩様(熊本県)

※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。

推測されにくいURLの生成など、知識がない分野なので興味をそそります。 masaki_hashimotoさん／沖縄県

表に出ないごく一部分のみで活用していたので、良い刺激となった。 Rohtoさん／神奈川県

ここまで幅広いことができるとは知らなかった。 林さん／愛知県

ハッシュ関数はさまざまなところで活用されていますよね。Gitとか。 そうでげすさん／神奈川県

前編と同様、普段から何気なく使ってはいるが、どのようなしくみかは知らなかったという声が多かったです。セキュリティにも直結する技術の1つなので、覚えていて損はありません。

一般記事　Windows Server 2016で構築する最新ファイルサーバ【後編】

Windows Serverの最新バージョン「2016」の新機能を紹介する前後編です。後編では、Microsoft Azure上に検証用サーバを構築する方法を解説しました。

なかなかWindows Serverは触る機会がないのですが、機会があれば検討してみたいと思える機能でした。 安部さん／東京都

現場では2003や2008が現役なので、2016の機能を紹介する記事はありがたいです。 ra1hennさん／東京都

Windows Server 2016の重複除去機能、すばらしいです。先日導入したのですが、サーバ上のディスク使用量が半分くらいになっています。 KKさん／愛知県

Windows Serverは使っていないのですが、クラウドなどで手軽に試すことができるのは良いなと思いました。 吉良さん／東京都

Linux好きの読者が多い弊誌ですが、寄せられた声からすると、仕事ではやはりWindows Serverを使っている方が多いようでした。OSSと商用製品、場面によってうまく使い分けられれば良いですね。

一般記事　Jamesのセキュリティレッスン【10】

セキュリティ対策・トラブルシューティングに活用できるパケットキャプチャのツール「Wireshark」の使い方を紹介する短期連載です。第10回では、これまでの連載の振り返りとして、10問の実践問題を出題しました。

問題を読むのに疲れてしまいましたが、リアルな想定で考えることができて良かったです。 コメットさん／兵庫県

Wiresharkすごいんですね。というかキャプチャされてしまうのが、セキュリティとしては案外怖いですね。 Tayuさん／千葉県

Wiresharkは機能が豊富過ぎるので、問題形式は自分の理解度を試すことができて良い。 psiさん／東京都

手元でできるセキュリティの実践記事ということで、人気の高い本連載。最終問題は、パケットから侵入者の行動を探るというアプローチで、実用的で良かったという声が寄せられました。

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
October 2017

2017年10月号
定価（本体1,220円+税）
184ページ

9月16日発売

［第1特集］個人でも会社でも使える!
これだけは知っておきたいGitのキホン

［第2特集］あなたのサーバは本当に安全ですか?
脆弱性スキャナVuls入門
システムのセキュリティチェックをもっと楽に

［特別企画］
AWS Lambdaで実現した画像管理環境／ネットワークエンジニアのなり方／なんでもTelnetメールサーバと通信してみる

※特集・記事内容は、予告なく変更される場合があります。あらかじめご容赦ください。

休載のお知らせ

「Debian Hot Topics」（第49回）は都合によりお休みさせていただきます。

お詫びと訂正

以下の記事に誤りがございました。読者のみなさま、および関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

■2017年8月号
●P.26　第1特集 第2章「ディープラーニング入門」
8月号Webサポートページに掲載していたサンプルコードのうち、「image_classify.py」に誤りがございました。7月25日より修正したものを掲載しなおしましたので、それ以前にダウンロードいただいた方は「sd201708_sp1_ch02_sample_rev1.zip」をダウンロードしなおしてください（image_classify.pyのみ差し替えてあります）。

SD Staff Room

●およそ3ヵ月前に行ったピロリ菌除去。1週間にわたり朝晩抗生物質を飲み続け、検査で晴れて陰性になった。医者が「胃がんのリスク低下おめでとうございます。でも胃酸が元の強さに戻るので、逆流性食道炎とか胃炎に注意してください」という。実のところ胃のムカつきが多くなった。副作用だ!（本）

●個人レベルの情報発信が簡単にできるようになり、普通の人でも（加工していない）RAW情報にアクセスできるようになった。それに伴い、報道にバイアスがかかっていたことが明るみになってきた。今こそ個人のリテラシが重要と思う。人の言ったことを鵜呑みにせず、自分の目で確かめることは重要だ。（幕）

●暑さのせいか身の回りの機器の不調が重なる。最初はディスプレイ切り替え機の故障。次に約4年間愛用していたAndroidタブレットが故障。通勤カバンのショルダーも金具が壊れた。スマホもポケットに入れておくと熱くなって強制リセットが。ま、タブレットが新品になってちょっと(*´ω`*)（キ）

●夏の猛暑が好きになる方法は、家庭の洗濯・布団干し担当になることでしょうか。ここぞとばかりに、大量の洗濯物や布団を干したくなります。逆に、雨が好きになるには家庭菜園がお勧めです。水やりの手間が省けるのはもちろんですが、心から「めぐみの雨がきた~」と思えるようになります。（よし）

●SNSで少し話題になっていた「カルピス×ビール」のカクテルを作ってみました。カルピスの原液1に対してビール5で作ります。甘酸っぱさのあとに苦味が残る、暑い日にぴったりな一杯です。ほかにもヨーグルトや抹茶で割るビアカクテルもあるそうで、二日酔いに気を付けながら試していきたいです。（な）

ご案内

編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［E-mail］
sd@gihyo.co.jp

［FAX］
03-3513-6179

※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

Software Design
2017年9月号

発行日
2017年8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
松井竜馬
大橋　涼
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
法人営業課（広告）
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。