Special Feature 01
新人に勧めたい本
Special Feature 02
正規表現トレーニング
2013 May
05
[ソフトウェア デザイン]
OSとネットワーク、
IT環境を支える
エンジニアの総合誌
Software Design
2013年5月18日発行
毎月1回18日発行
通巻337号
（発刊271号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
1,280円

Part 1

IT業界ビギナーのための お勧め書籍 55冊+α

Special Feature 01

新しい季節に君へ

Part 2

IT業界副読本

The Perfect Book Guide *for* Absolute Beginners

Special Feature 02

覚えておきたい、ちゃんと使いたい!
正規表現をマスターしていますか?

Extra Feature

OpenFlowを実装してみた!
OpenFlow/SDNフレームワーク
バーチャルネットワーク
コントローラ2.0開発の実際

新連載!

創造力を刺激する魔法
enchant

コレクターが独断で選ぶ
偏愛キーボード図鑑

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

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

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

第53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。

レスポンシブ Web デザイン

レスポンシブ Web デザインとは

Web サイトのデザインを行ううえで問題になる要素の 1 つに、ユーザが使用する端末の画面サイズがあります。今やインターネットを利用することができる端末は PC だけではなく、タブレットやスマートフォンなどの小さな画面の端末もあれば、何十インチという大画面テレビで Web ブラウジングをすることも可能です。それらのまったく条件の異なる画面に対して、1 種類のデザインだけでコンテンツを見せるというのは現実的ではありません。

そこで注目されているのが「レスポンシブ Web デザイン」です。レスポンシブ Web デザインとは、ユーザが使用している端末やブラウザの画面サイズや解像度に合わせて、Web サイトのデザインを自動で最適化して表示する手法です。レスポンシブ Web デザインの最大の特徴は、このようなデザインの切り分けを単一のソースコードで実現するという点です。端末や Web ブラウザごとに異なる Web サイトに誘導するのではなく、あくまでも 1 つの HTML/CSS ファイルだけでデザインの調整を行うわけです。

レスポンシブ Web デザインの実現には、CSS3 の「Media Queries」という技術を利用します。これは画面サイズやデバイスの解像度に応じて、適用する CSS を自動で切り替えることができる機能です。たとえば画面サイズのしきい値を決めておき、その値よりも小さな画面でアクセスした場合にはスマートフォン用の CSS を、大きな画面でアクセスした場合にはデスクトップ PC 用の CSS を適用するといった具合です。

レスポンシブ Web デザインのメリット

従来のスマートフォン向けサイトでは、ユーザエージェント情報を元にしてアクセスしている端末の種類を特定し、それぞれの端末向けに用意された専用のページにリダイレクトするという手法が一般的でした。ところが、端末の種類が増えるにしたがってこの方法には次のような問題が指摘されるようになりました。

- さまざまな端末に対して専用のページを用意するため開発コストが増加する
- 同じ内容の更新を複数のページで同時に行わなければならず、メンテナンスの手間が増える
- 新しい端末が登場するたびに解像度や画面サイズ、ユーザエージェントなどを調査して対応しなければならない
- 端末ごとに使用する URL が異なってしまう

これに対してレスポンシブ Web デザインでは、基本的な部分では同じ素材を共有しながら、CSS の記述を追加・上書きするだけでデバイスごとの最適化を実現することができるため、サイトのメンテナンスが容易になるというメリットがあります。デザインの切り分けは画面サイズと解像度をベースに行うため、新しい端末にも即座に対応することが可能です。また、ベースとなる CSS の設定は共有できるため、サイト全体のデザインを統一しやすいという利点も挙げることができます。

デメリットとデザイン上の注意点

レスポンシブ Web デザインでは作成した Web サイトのメンテナンスが容易になる一方で、最初のデザインを組み上げるまでの製作時間が増えやすいという性質もあります。それぞれの端末専用のデザインを作るわけではなく、あくまでも画面サイズと解像度を基準にして複数の端末でデザインを共有するため、最適なデザインを探し出すには多くの試行錯誤が必要になるからです。

もう 1 つの弱点は通信のデータ量です。異なる画面サイズのデザインでコンテンツを共有するということは、どの端末も同じ容量のファイルを読み込む必要があることを意味しています。たとえばデスクトップ用に高画質の画像が用意されている場合、回線速度の遅いモバイル環境からのアクセスでも同じ画像を読み込むことになってしまうため、アクセシビリティを損ねる可能性があります。

そのほかに留意すべきポイントとしては、単に画面サイズだけに気を配るのではなく、端末ごとの利用シーンの違いなどにも注目してレイアウトを決めることが挙げられます。自宅の PC からアクセスしている場合と、外出先でスマートフォンからアクセスしている場合とでは、ひとつのサイトでも探し出したい情報の内容が同じとは限らないからです。したがって、コンテンツの表示順など、利用シーンごとの特性を踏まえたデザインを構築することが大切だと言えます。SD

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。

Contents

Software Design May 2013

新しい季節に君へ

IT業界ビギナーのための お勧め書籍 55冊+α

017

Part 1 教科書どおりじゃない

先達直伝「効く」ブックガイド

Part 2 読まずに入れるか!? IT業界

IT業界副読本

1

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。

Contents

Software Design May 2013

2

第2特集 Special Feature 02

覚えておきたい、ちゃんと使いたい!

正規表現をマスターしていますか?

一般記事 Article

巻頭Editorial PR Editorial PR

アラカルト A La Carte

広告索引 AD INDEX

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。

ソフトウェア・デザイン［目次］

Contents

Software Design May 2013

3

Column

Development

OS/Network

Inside View

Logo Design ロゴデザイン ＞ デザイン集合ゼブラ+坂井 哲也

Cover Design 表紙デザイン ＞ Re:D

Cover Photo 表紙写真 ＞ Dirk Freder/Getty Images

Illustration イラスト ＞ フクモトミホ、高野 涼香

Page Design 本文デザイン 岩井 栄子、近藤 しのぶ、SeaGrape、安達 恵美子

［トップスタジオデザイン室］轟木 亜紀子、阿保 裕美、佐藤 みどり

［BUCH+］伊勢 歩、横山 慎昌

森井 一三、Re:D、［マップス］石田 昌治

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

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

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

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

Pocket Reference

小さくても、中身充実！

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

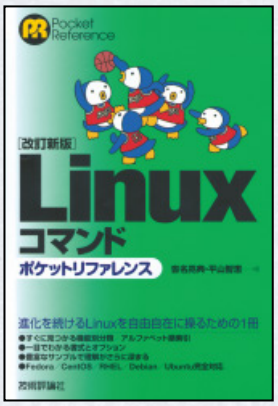
田口貴久ほか　著、日本仮想化技術株式会社　監修
四六判／352 ページ
定価 3,129 円（2,980 円＋税）
ISBN 978-4-7741-5600-2

牟田口大介　著
四六判／592 ページ
定価 2,919 円（2,780 円＋税）
ISBN 978-4-7741-5542-5

岡本隆史、武田健太郎、相良幸範　著
四六判／272 ページ
定価 2,604 円（2,480 円＋税）
ISBN978-4-7741-5184-7

山森丈範　著
四六判／304 ページ
定価 2,499 円（2,380 円＋税）
ISBN 978-4-7741-4396-5

石田つばさ　著
四六判／448 ページ
定価 2,289 円（2,180 円＋税）
ISBN 978-4-7741-4836-6



沓名亮典，平山智恵　著
四六判／576 ページ
定価 2,394 円（2,280 円＋税）
ISBN 978-4-7741-3816-9

細島一司　著
四六判／464 ページ
定価 2,919 円（2,780 円＋税）
ISBN 978-4-7741-5135-9

土井毅、髙江賢、飯島聡、髙尾哲朗 著　山田祥寛 監修
四六判／488 ページ
定価 2,709 円（2,580 円＋税）
ISBN 978-4-7741-4948-6

朝井淳　著
四六判／640 ページ
定価 2,079 円（1,980 円＋税）
ISBN 978-4-7741-3835-0

片渕彼富　著、山田祥寛　監修
四六判／448 ページ
定価 2,709 円（2,580 円＋税）
ISBN 978-4-7741-5067-3

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

たった５万円の予算で１００万円売り上げる方法、教えます。

ネットで儲ける王様のカラクリ

～物語でわかるこれからのWebマーケティング

竹内謙礼　著／四六判／256 ページ
定価 1,659 円（1,580 円 + 税）
ISBN 978-4-7741-5606-4

・誰も教えてくれなかった
　“ホームページの値段のカラクリ”とは？
・SEO対策をしても結果が出ない理由とは？
・ネット広告がお金のムダにならないか、どう見極めればいいか？

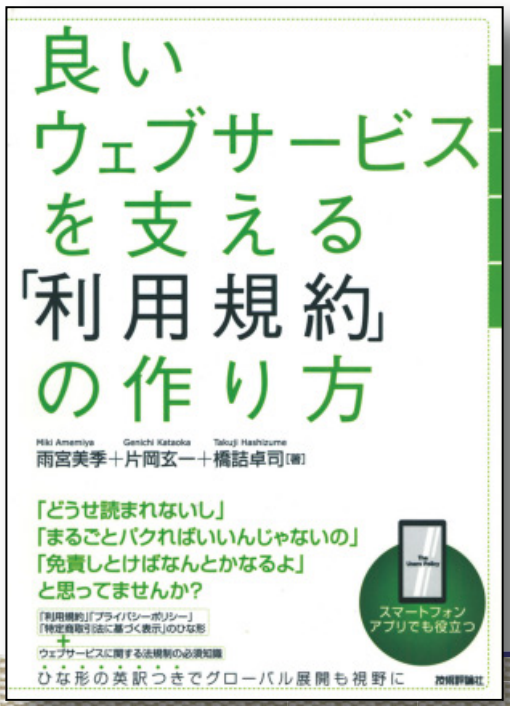
ネットショップの店長・コンサルタントとして数多くの実績を持つ著者だから書けた、IT業界のリアルと売上アップのノウハウがみるみるわかる新感覚ビジネスノベル！

日本初！大切なウェブサービスを守る「利用規約」の解説書

良いウェブサービスを支える「利用規約」の作り方

雨宮美季、片岡玄一、橋詰卓司　著／ A5 判／ 240 ページ
定価 2,394 円 (2,280 円 + 税)
ISBN 978-4-7741-5594-4

「利用規約なんてどうせ読まれないし」
「まるごとパクればいいんじゃないの」
「免責しとけばなんとかなるよ」...なんて思ってませんか?
本書は、多くのベンチャー企業の支援を通じてウェブサービスに関するリーガルサポートを数多く手がけている弁護士と、スタートアップ企業から上場企業までさまざまな規模・業態の企業でウェブサービスの運営をサポートしてきた法務担当者2人が、その経験をもとに、「ウェブサービスを安全に・円滑に運営するために本当に気を配る必要のある事項」をわかりやすく解説。



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

紙面版
A4判・16頁
オールカラー

電脳会議

DENNOUKAIGI

一切無料

新規購読会員受付中!

『電脳会議』は情報の宝庫、世の中の動きに遅れるな!

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

今が旬の情報満載!

新規送付のお申し込みは…

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

電脳会議事務局　検索

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

毎号、厳選ブックガイドも付いてくる!!

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

一切無料!

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

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

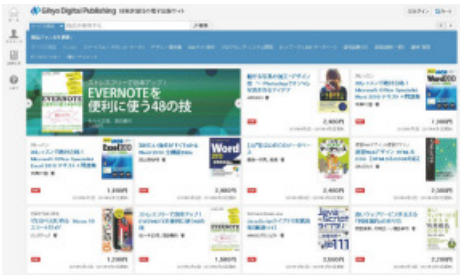
技術評論社の本が電子版で読める！

電子版の最新リストは

Gihyo Digital Publishingの

サイトにて確認できます。

http://gihyo.jp/dp



GIHYO Digital Publishing 未来の"普通"を、今。

Google play

楽天 Rakuten KC　kobo

※販売書店は今後も増える予定です。

法人などまとめてのご購入については
別途お問い合わせください。

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

新連載

ネットワークエンジニア虎の穴

「自宅ラックのススメ」

文／ tomocha（http://tomocha.net/diary/）

第1回

基本の19インチラック

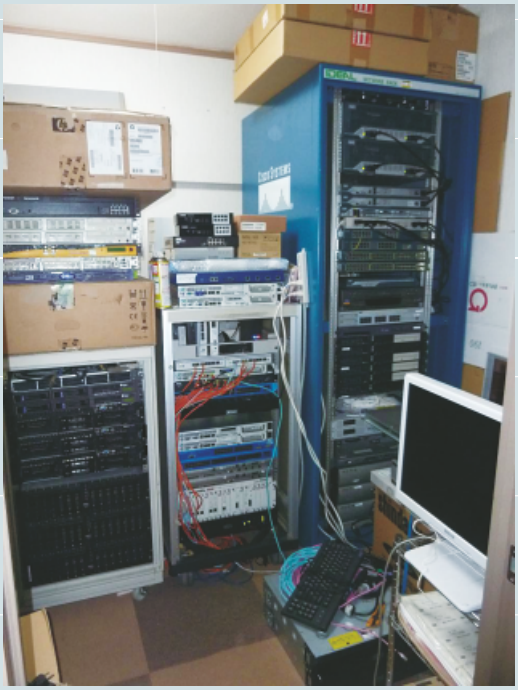
イラスト：高野涼香

自宅ラックの良さを紹介してほしいということで、連載をいただくことになりました。tomochaです。

気がついたらラックができていた……

まず、なぜ自宅にラックがあるの？　ということから説明しましょう。気づいたらパソコン、サーバ、ネットワーク機器などたくさん増えていることが多々あると思います。あの機器を触ってみたい、欲しいなと思い、中古（Yahoo!オークションとかeBayとか）を見ていたら、ついつい買っていたり、「処分するから、いらない？」というようなことで、ついつい譲り受けて気づいたらものすごく増えていたということは読者の皆さんも経験しているでしょう。

▼写真1　どこの電脳要塞（データセンター）なのか……自宅です！



機器が増えるのは良いのですが、どのように収納するのかというのが課題になりはじめ、最初は機器を縦に積み上げたり、スチールラックに乗せてみたり、机の上に置いてみたりと試行錯誤することが多いと思います。しかし、地震など来たらアウトですし、設置するにあたっても、上に積み上げるにしても限度があります。また、収納効率も悪く、専用のサーバラックを導入してみたいと思うようになります。

今回お話させていただくのは、19インチのサーバラックのお話です。サーバラックは、比較的中古で安価に出回っているということ、手頃なサイズで、ハーフラックからフルラックまで、探せばそれなりの種類が出ています。こういった経緯から、自宅にサーバラックがたくさん設置されるにいたったわけですね。

ラックを選ぶならばEIA規格が吉

実際に、サーバといっても大きく分けて、タワー型、ラックマウント型の2種類あります。

タワー型だと、数が増えてくると、メーカーやモデルによって、形やサイズが違うので、どのように積み上げるか課題になってきます。台数が増えれば増えるほど、扱いに困るでしょう。重量棚などをおいて、横に並べるのも良いですが、ものすごく効率が悪くなります。ここで、EIA（米国電子工業会）により、規格化されたラックに搭載できる機器をあらた

▼写真2　自宅ラックの詳細。ケーブルの整理はまだです（汗。

めて検討するようになります。機器の高さはU（ユニット）で規定され、1Uあたり、高さが1.75インチ（44.45mm）、横幅が19インチ（482.6mm）で、奥行きは規定されていません。奥行きが規定されていないため、ラックを選ぶときは、サーバ本体の長さや、レールの長さを考慮して、適合するラックを選びましょう。また、IT機器用のほかに、シンセサイザー等楽器用の19インチラックの場合は、奥行きが短かったり、ケーブルをまとめるスペースがないなど、困ることになりますので、注意が必要です。ちなみに、EIA規格のラック以外にもJIS規格のラックも存在しますが、ラックマウントタイプのサーバやネットワーク機器の場合、EIA規格となりますので、間違えないようにしましょう。こうやって、横幅、高さが規定された機器を美しく収容できるということですね。「あぁ。これはすばらしい」と思うわけです。自宅で実現すると夢のように浮かれること間違いなし！

目指すは「自宅データセンター化!!」

筆者の環境では、外部公開のWebサーバ、メールサーバ、DNSサーバ。そして、自宅内用途のファイルサーバ、DNSサーバ、データのバックアップサーバ、VPNサーバなどが立ち上がっています。以前は20台ぐらい動いていた時期もありますね。そのほかには、ルータやスイッチ、その他検証用の機材が入っていたりします。基本的にはすべて、ラックマウント可能な機器を使用しています。

以前は、仮想化がこなれておらず、サーバの性能がそれほどよくなかったころは、用途ごとにサーバなどを用意しており、たくさんの台数があった時期もありますが、最近はずいぶん安価で高性能になった分、集約することにより、稼働台数が減っています。で、成し遂げた構成として、VPNや踏み台を経由し、検証環境や自宅リソースへアクセスができるようになります。それ以外では、インターネットへ接続するためのルータ（しかもBGPしゃべってるよ！）、ネットワーク機器等も設置されています。そのほか、ネットワーク機器へリモートでアクセスすることを想定し、シリアルコンソールサーバなど置いています。ここまでやると、データセンターに設置しているような構成の環境ができあがるわけです。機器が増えてくると、耐荷重は大丈夫か、電源容量は足りるのか、接続するLANケーブルや電源ケーブルをどのようにきれいにまとめるかなど、学ぶことが増えてきます。このあたりは次回以降にお話させていただきます。

とりあえず、イメージをつかんでいただくために、自宅のラックの写真を2枚ほど紹介しておきますね（写真1、2）。とはいえ、ケーブルをキレイにまとめたりせず、積み上げているだけの様子なのであまりよろしくない写真ですが……。単なる物置ですが収容効率は良さそうでしょ（笑）。SD

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

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

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

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

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

DIGITAL GADGET

安藤 幸央 —— Yukio Ando ——
EXA CORPORATION
[Twitter] >> @yukio_andoh
[Web Site] >> http://www.andoh.org/

Volume 173 >>>>>

新LEGO MindStorms EV3にみる、コンピュータとブロックの融合

LEGOブロックの歴史

皆さんは幼少のころ、ブロックで遊んだことはあるでしょうか？　LEGOブロックはデンマーク語で「よく遊べ」を意味する“leg godt”に由来すると言われる80年以上の歴史を持つ玩具です。LEGOブロックが現在の形になったのは1958年ごろ。それ以来基本形状は変わらず、昔のブロックと今のブロック同士がつながるという驚異的な互換性を保っています。

LEGOは単なる玩具、ブロックとして何かを作り上げて楽しむだけではありません。その表現力や多数の部品の自由度を活かして、建築物を再現したり、映画のシーンを再現したり、実際に使える家具を作ってしまったりと、世界中で思いもよらない応用がなされています。

最近は手軽に使える3Dプリンタも普及しはじめていますが、形状のプロトタイピングの素材としてブロックほど色と形状に関して、素早く試行錯誤ができるものがあるでしょうか？　スマートフォン「INFOBAR」の初代機の形状や色使いのデザイン検討の際にはLEGOが活用されたとも言われています。画一的に部品化された個々のブロックとカラフルな色合いは、人間の想像力、形状的・空間的な感覚を研ぎすますことに優れています。ほんの数個のブロックの組み合わせでも、想像力によって補強されたその物体は、太平洋を越える飛行機にも、荒波を乗り越える海賊船にもなるのです

驚くべき数値

LEGOは現在、1年間におよそ約190億個生産され、累計4,000億個。400億個のレゴブロックを積み重ねると、月に届くとのことからその数の多さに圧倒されます。また、全世界の1人1人に現存するLEGOを分け合ったら、1人当たり62個になるそう。

基本ブロック（2×4ポッチ）2個では24通りの組み合わせがあり、ブロック6個の組み合わせでは、なんと9億1,510万3,765通りの組み合わせがあるという計算になるそうです。現在のさまざまな商品で使われる個々のブロックには約2400種類の形と計53色の色使いがあり、各ブロックは0.002mm以下の誤差しか許さない品質管理がされています（参考 http://en.wikipedia.org/wiki/LEGO）。

長く使えること ——IT技術の指針にも

LEGOブロック的な考え方はITテク

※本記事は海外での発表に基づいた内容です。国内でのMindStorms EV3については、一般店舗販売となるリテール版の予定は未定で、教育機関や企業向けに販売される教育版は2013年9月に発売予定です。

ノロジの中でも参考になる要素が数多く含まれています。

- 基本設計の堅牢さ
- 過去と将来にわたる互換性
- バリエーションや拡張性の余地
- 変化や改善を受け入れられる土壌があり、常に新しい要素を取り入れる余地があること
- デファクトスタンダードな基本機能を持ち、基本機能の組み合わせで実現できることが多いこと
- 相互接続性:さまざまな部品同士がさまざまな方法でつながりあう
- 多様性:ある形状を作るための部品や方法が1種類ではなく、いくつかの方法で作れること

Webの技術やテクノロジで何かを作ろうと考えたとき、現在の最適解を追い求めがちです。しかし、今後長く使い続けられるものを考えた場合、近い将来のことも見据えなければいけません。それは現在のハードウェア環境、ネットワーク環境での最適解だけでなく、数年後の未来の環境を考慮したうえで、スケール(拡大)するか、アーキテクチャに余裕があるかがテクノロジの長生きのコツになってくるのです。

最近ではレゴエデュケーションと呼ばれるLEGOブロックや、続いて紹介するMindStormsを活用した教育分野にも広く展開してきています。都内でもLEGOを使った子ども向けの教室などが開催されています。子どもだけでなく、大人の皆さんも、最新の携帯ゲーム機とはまったく違った世界のブロックで想像力を刺激し、さまざまな新しい発想を広げてみてはいかがでしょうか?

初代MindStorms登場の衝撃

最初のLEGO Robotics Invention System(RIS)は1998年、もともとはMITメディアラボとLEGO社とのパートナーシップを通して教材としてリリースされたものでした。その後2006年に、LEGOロボティックス用キット「MindStorms NXT」が発売されました。そして2009年には「MindStorms NXT 2.0」が発売されます。価格が手頃であったことと、仕様やファイルフォーマットなどがオープンであったため、本来の目的であるロボティックな玩具として楽しまれました。この登場によって、さまざまなプログラミング言語による利用と、目を見張るような応用例に恵まれたのです。

モーターやセンサー、ギアといったさまざまなLEGO部品、それにMindStorms用電子機械部品を組み合わせることで、物理的な形状だけでなく、それらの動きや動作などの機械的な動きまでもがプログラミングできる環境です。センサー群にはタッチセンサー、光センサー(紙に黒ペンで描いた個所を認識できる)、色センサー、ジャイロセンサー、回転センサー、温度センサー、音量センサー、超音波センサーなどが用意されています。そのほかにも赤外線リモコン、カメラ、フィードバック機能のあるサーボモーターなど、至れり尽くせりでした。

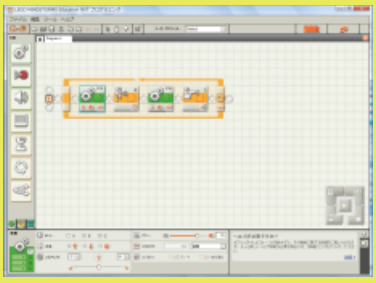
MindStormsの名前の由来は、プログラミング言語「LOGO」の作者であり数学教育者でもあるシーモア・パパートの著作『マインドストーム　子供、コンピューター、そして強力なアイデア』(未来社;1982)からきているそうです。シーモアは博士の持論としてプログラミング教育を通して「バグの体験こそコンピュータリテラシーには必要」とのこと、プログラミング体験を通して試行錯誤したり、不具合を完全にしたりする行為そのものを具現化したのがMindStormsなのです。

MindStorms NXT 2.0
http://www.legoeducation.jp/mindstorms/

LEGO MindStorms EV3

MindStorms NXT 2.0以後、登場からしばらく時間が経っていたり、旧バージョンの流通が減ってきてパーツが手に入れにくくなりつつある状況から、新バージョンの登場が望まれてきました。そんな中、2013年始めに開催された国際家電見本市CES 2013にて、MindStormsの新バージョン「MindStorms EV3」が発表されました。2013年後半(秋頃)に339.95ドル(基本セット)、433.95ドル(ソフトウェア付きフルセット)で発売予定とのことです。

最近の流行もかんがみて、iPhoneやAndroidとの連携も強化され、Bluetoothなどの実際にプログラミングす



④NXTのプログラミングツール画面

⑤NXTで作られたロボット

⑥教育版EV3の基本セット(左)、⑦拡張セット(右)

るとなると面倒な要素も、平易に利用できるのが良いところです。また、旧来のMindStormsからMindStorms EV3への大きなパラダイムシフトの1つは、個々の部品に対して動きをプログラミングしなければいけなかった環境から、部品1つ1つが命令や決められた動きを内包した機構を持ったことです。これにより部品同士を組み合わせるだけでどのような部品がつながったかを認識し、新しい働きを生み出せるようになります。もちろん旧来の全体をプログラミングする方式も可能ですが、EV3ではブロック1つ1つにプログラミングした振る舞いを与え、そのブロック同士の協調で全体が動作するという考え方になります。

MindStorms EV3はiPhoneなどに搭載されているARM系のCPU ARM9と16MBのフラッシュメモリ、64MBのRAM、SDカード、USB 2.0、Wi-Fiの利用が可能。ベースとなっているOSはLinuxです。基本構成で4つのセンサー用インプット、4つのモーター用アウトプットを持ちます（もちろん拡張可能）。

スマートフォンの専用アプリから声でEV3をコントロールしたり、内蔵スピーカーでしゃべらせたりすることができるそうです。あらかじめ17種類のロボット制作法が書かれたマニュアルが同梱されるそうですが、それ以上にLEGOコミュニティで共有される創意工夫や制作技法に期待が寄せられるところです。SD

⑧教育版EV3のプログラミングツール画面

gadget 1

ATOMS

http://www.kickstarter.com/projects/atoms/atoms-express-toys

iPhoneアプリ連動のおもちゃがLEGOで作れる

ATOMSはLEGOブロックと組み合わせてiOSデバイスからコントロールできるようにする、おもに子ども向けの電子ブロックです。ATOMS専用のコントローラとiOSデバイスとは、Bluetoothで通信し、そこからはケーブル接続の有線でモーターやセンサーをコントロールします。クラウドファンディング Kickstarterで資金調達に成功し、現在生産に向けてがんばっているようです。

gadget 2

Life of George

http://george.lego.com/en-us/products/life-of-george/

レゴとAR（拡張現実）

Life of GeorgeはAR（拡張現実）認識用のボードとLEGO部品のセットで、このセットで作った素材をスマートフォンでARマーカーとして認識するしくみです。スマートフォンアプリに表示されるお題に従って、ボードの上にLEGOモデルを構築してそのモデルをAR認識して楽しむことや、自分オリジナルのモデルを作って登録することもできます。だんだんと、どちらがARでどちらが現実なのか曖昧な遊びになっているところがポイントかもしれません。

gadget 3

LEGO CUUSOO

http://lego.cuusoo.com/

空想したLEGOを商品として実現

LEGOにはスターウォーズシリーズをはじめ、さまざまなセットが存在しますが、実現していないモノは数多くあります。LEGO CUUSOO（空想）サイトは、ユーザが試作したりリクエストしたLEGOモデルを商品として販売してもらえるよう試作モデルや投票を集めるサイトです。深海6500のモデルや、小惑星探査機はやぶさのモデルなど、商品展開が実現してるものもあります。

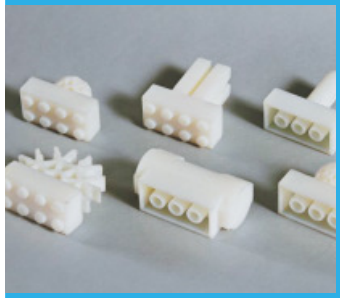
gadget 4

Free Universal Construction Kit

http://www.sy-lab.net/Free-Universal-Construction-Kit

他の玩具にLEGOを接続

Free Universal Construction Kitは LEGO以外のブロックや玩具に接続可能なアダプタ部品群です。部品そのものを販売するのではなく、制作に必要となるデータを配布する形態で、各自が3Dプリンタや、オンラインの3Dプリントサービスで作ってくださいというスタンスです。LEGOそのものにも数多くの部品がありますが、それらを他の玩具と連係させる可能性へとさらに拡張する、新しいアイデアの流通方法です。



新連載

コードは一行も書かずに!!

プログラム知識ゼロからはじめる
iPhoneブックアプリ開発［出張所］

わずか20ステップで写真集アプリが作れる!

GimmiQ（ギミック；いたのくまんぼう&リオ・リーバス）
URL http://ninebonz.net/
URL http://www.studioloupe.com/

はじめられない理由なんていらない

「iPhoneアプリを開発してみたい」

iPhoneアプリ開発には興味があるけど、難しそう、勉強する時間がない、そもそもどこから手をつけて良いのかもわからない……、なんて方も多いのではないでしょうか？　iPhoneアプリ開発はプログラミング未経験者だってやる気さえあれば必ずできます！　本連載は、アプリ作りの楽しさを知ってもらうためにプログラムを一切書かないところからはじめ、1年を通して徐々に基礎を固めていく記事として構想を練りました。

本連載ではおもに「ブックアプリ（まんが・絵本・写真集など）」を題材に、アプリ開発の基礎を身につけることを主旨とします。題材としてブックアプリを選んだ理由は、制作の過程にアプリ開発に必要な基礎がたくさん詰まっているからです。オリジナルの作品を作りやすいという点もあります。ブックアプリ自体にはあまり興味がない、という方もここはひとつお付き合いくだされば幸いです。

これからの全12回の連載に詰め込む知識を応用すれば、きっと自分が作りたいと思ったアプリをどのようなアプローチで作れば良いかがしっかりイメージできるようになるはずです！

今回は連載第1回目にして早速、1本のシンプルなブックアプリを完成させるまでのチュートリアルを作成しました。しかも、プログラムをまったく書かずにです！　ぜひ110ページからの本編記事をチェックしてください！

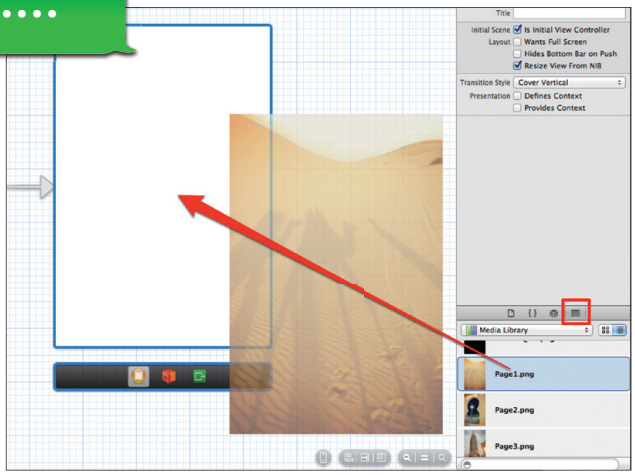
Column

国連WSAのアワードを受賞

今回のサンプル写真集アプリで使った写真は、著者がアブダビ（アラブ首長国連邦）に行ったときに撮影したものです。なぜアブダビ……それは国連が主催するWorld Summit Awards Mobile 2013で、GimmiQの『MagicReader』がm-Inclusion & Empowerment部門で受賞し、アブダビでの式典に招待されたからなのです！　詳しくはブログ（http://ninebonz.net/magicreader-wsa）にまとめてありますのでよろしければご覧ください。

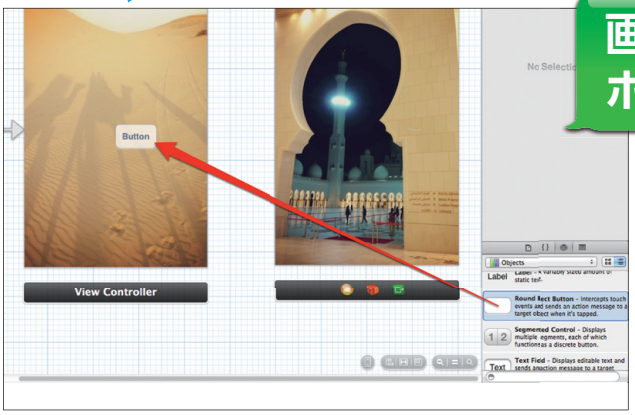
画像を貼り付けて……

OSX 10.7.4以降のMacさえあれば本当にだれでも作れます。自分の持っている写真や画像を使って記事の手順どおりに進めれば、オリジナルアプリのできあがり！



画面切り替えのボタンをつけて……

アプリの基本動作となる画面遷移。プログラムのことなど知らなくても、ドラッグ＆ドロップ操作だけで簡単に実現できることがわかります。



ごく単純な紙芝居風アプリですが、ここがスタート地点。連載を重ねるごとに機能を加えながらアプリ開発に馴染みましょう。いまこそiOSアプリ開発をはじめるチャンスです！

ハイ、完成!

開発の手順はP.110の本編へ!

新連載

enchant ～創造力を刺激する魔法～

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

第1回 長い旅のはじまり

企業がオープンソースソフトを作るということ

Software Designの読者の皆さま、初めまして。株式会社ユビキタスエンターテインメントの代表をしている清水と申します。

僕たちの会社は、いま、2つのプロダクトを作っています。1つは誰でも簡単にゲームプログラミングができる「enchant.js(エンチャント・ジェイエス)[注1]」というオープンソースのライブラリ。もう1つは、「enchantMOON(エンチャント・ムーン)[注2]」という名前のハードウェア製品です。

オープンソースのライブラリを営利企業が作り、供給し続けるということはどういうことなのか。これまでにもたくさんの方々からご質問をいただきました。そしてさらに、ソフトウェアの会社が、独自のハードウェアを創りだそうというのはどういうことなのか。それに関しても多方面から質問を頂戴しています。

enchant.jsが公開されてから、2013年4月17日でちょうど2年になります。enchant.jsの発表にあわせてサービスを開始したゲーム投稿サイト「9leap[注3]」は、公開してわずか10日で100本ものゲームが投稿され、現在までに3,000以上の作品が寄せられています。2013年度からは、新たに高校生向けのコンテストも開始される予定で、全国のあちこちの大学や高校の授業でenchant.jsが使われはじめています。また、海外でもenchant.jsを教材に使う大学も現れてきました。これまでに4冊の本が発売され、学ぶための環境も整いつつあります。年内には英語版の解説書も英国・米国の出版社から発売される予定です。

僕たちがなぜ、どのようにしてenchant.jsというソフトウェア、そしてenchantMOONというハードウェアを作ろうと思うようになったのか。この連載では、これまでの道程を振り返りつつ、僕たちがこれから本当にやりたいと思っていることは何なのか、願わくばその想いをエンジニアの皆さんと共有したいと考えています。

オースチンの景色が呼び起こした80年代の記憶

enchant.jsをめぐる旅。

これは長い長い旅です。いまはその旅のほんの入り口に過ぎません。

旅の始まりは2011年の初春。

僕たちはテキサス州オースチンにいました。テキサスに来た目的は、全米最大の音楽と映画

注1) http://enchantjs.com/ja/
注2) http://enchantmoon.com/
注3) http://9leap.net/

とインターネットサービスの祭典、SXSWことサウス・バイ・サウスウエストに参加するためです。ちょっとした展示も行うことになっていました。

前日のうちに展示の準備はすべて整い、あとは開催を待つのみ、ということで時間を持て余した僕たちはレンタルしたセグウェイでオースチンの小さな街を疾走していました。

セグウェイは乗ってみると結構速く、疾走感があります。身体を前に傾けるとグイーンとスピードが乗って、僕は高速に流れて行くアメリカの田舎町の景色をぼんやりと眺めました。オースチンは古い街です。レンガ作りの建物はどれも色褪せていて、とりわけ時間が夕刻だったせいもあるのですが、景色はまるでモノクロの映画のように見えました。そのとき頭の奥でパチンとなにかが弾けて、僕は思わず上体を浮かせました。急ブレーキがかかります。

「いいことを思いついたぞ」

僕は思わずニヤリと笑いました。

それはほんの小さな思いつきでしたが、ずっと考えていたことに答えが出た気がしたのです。

何を思いついたのか。

うまい説明をするのは難しいのですが、そのとき僕の脳裏に浮かんだのは、1980年代のコンピュータ雑誌でした。

このオースチンの古くさく、無彩色な景色が、まるでHyperCard[注4]のスタックのように思えたのです。アンティークなレンガ作りの建物。街を行く人々。カウボーイがそのまんまの格好で闊歩するバー。まるで古い映画の世界。僕はモノクロのHyperCardのスタックの中に入り込んだ登場人物のような錯覚に陥り、それがあの懐かしい、マイクロコンピュータ雑誌黄金時代を思い出させました。

80年代はマイコン雑誌の全盛期でした。月刊『アスキー』、月刊『I/O』、そして月刊『マイコンBASICマガジン』。もちろん月刊『The BASIC』。

その頃のコンピュータ雑誌の記事は、投稿が主体でした。原稿料も、いまでは考えられないくらい高かったと思います。記事を書ける人が少なかったし、投稿そのものは多かったのです。まさに古き良き時代、いま振り返ると単なるノスタルジィとも思えますが、しかし確実に、いまは喪われたカルチャーがそこにはありました。投稿者が投稿し、読者がいつ投稿者になってもおかしくない、独特の緊張感。誰もがみな、次にこの原稿を書くのは自分だと夢想しながらページをめくっていたと思います。僕自身もそうでした。

ニコニコ動画とストリートミュージシャン

ニコニコ動画が登場するまで、ミュージシャンのストリートパフォーマンスは下火になっていたと思います。歩行者天国で演奏するとか、そういうカルチャーが消え去り、古いもののように思われていました。バンドブームというのもとっくになくなって、いまどきビジュアル系でもないロックバンドを組んでるのはいい年をした大人ばかり。

ところがニコニコ動画の登場によって、ミュージシャンには別のジャンルが産まれました。ボカロPと呼ばれる作曲家(プロデューサー)や、歌い手、踊り手、それまでバラバラだった人々が、バラバラなままニコニコ動画という場を得て連合したのです。ニコ動は新しい形のクリエイティヴ・プラットフォームになりました。いわばネットの中に現れた、21世紀の歩行者天国です。

同じことが、プログラマに関しても言えるのではないかと思います。いまや趣味でゲームプログラミングをする人間というのはほとんど世の中から消滅してしまいました。いま、趣味でプログラムを書こうとしたら、それはWebサー

注4) 1987年にアップルコンピュータ(当時)のビル・アトキンソンが開発したソフトウェア。ハイパーテキストのノードとなるカードとスクリプト言語のHyperTalkによって簡単にアプリケーションが作れるマルチメディアオーサリングツール。

ビスをなにか立ち上げるとか、iPhoneアプリを作るとかを意味します。

しかしiPhoneアプリにせよWebサービスにせよ、ホンモノのプロが一線で活躍する領域に素人が徒手空拳で殴り込みをかけるような場所では、ふつうの人は気後れしてしまって入っていけません。とくに若い人はそうだと思います。学生にとって、年間1万円のiPhone SDK使用料金は決して安いものではありません。さらに開発にMacが必要となれば、なおのことです。

アマチュアプログラマが成長し、活躍する場としては、21世紀はむしろ進化ではなく衰退しています。それはちょうど、ストリートミュージシャンがいなくなった歩行者天国のようなものです。プロと消費者にキッパリと別れて、消費者は与えられたコンテンツをただただ消費するだけ。

ならばむしろ、80年代のマイコン業界を、21世紀に蘇(よみがえ)らせてはどうでしょうか。

もちろんモダンで、インターネット時代ならではの要素を取り入れつつも、その精神の根底は80年代と同じものとします。それはすなわち、「明日はキミもプログラマになれる！」という力強いメッセージです。

プログラマこそが人類の進歩を支える職業

実はオースチンに来る直前、僕はサンフランシスコのゲーム開発者会議(GDC)に参加していました。

そのとき、たまたま会社の社員同士で「8時間でゲームを8本作る」という開発レースを遊び感覚でやってみたのです。時間制限があると1つ1つの作りは荒くなりますが、その制約のなか苦し紛れにでてきたものが、実は意外と面白かったのです。そしてその経験は、僕自身、プログラミングの喜びや楽しさを思い出すいいきっかけになりました。アイデアを実装し、バグと戦い、なんとか完成にもっていく。手に汗握る緊張と興奮。まるで少年の頃に戻ったようでした。

その感覚は、まさにマイコン雑誌に投稿していた頃の僕自身の感覚に重なりました。

「この楽しさを、もっと多くの人と共有できたら……とてつもなく面白いことが起きるんじゃないか」そう思いました。

いつの時代にも、プログラミングに興味を持つ人は必ず一定数以上はいるはずです。それに昔と違っていまはコンピュータはすべての家庭とすべての学校にあるわけです。けれども高校のパソコンにVisual Studioをインストールできるケースは少ないだろうし、生徒が勝手にOSをUbuntuあたりに入れ替えるわけにもいきません。

そういうどこにでもある環境で、プログラミングを愉(たの)しむことができたら、昔よりコンピュータを使う人そのものはずっと増えているわけだから、もっと面白いことが起きるのではないかと考えました。

ひょっとすると、僕が高校生の頃にはプログラミングになんて見向きもしなかったような女の子たちも、いまの女子高生なら少しは興味を持ってくれるかもしれません。そうするとプログラミングをする人の数というのはぐんと増えることになります。

プログラマが増えれば、それだけ優秀なプログラマが出現する可能性も高まります。

僕はプログラマこそが人類の進歩を支える重要な職業だと信じていますので、プログラマの母数が増えるということは、すなわち人類の進歩そのものを加速することになります。それはなんとやりがいのある仕事なのでしょうか。

「80年代のマイコン文化を、21世紀にリバイバルする！」

これはノスタルジィではなく、ルネッサンスなのだと自分に言い聞かせました。単なる先祖帰りでは意味がありません。しかし一度喪われてしまった、あの素晴らしい興奮の日々を再び現代に蘇らせるというのは生涯を賭して挑むに相応(ふさわ)しいテーマだと強く思わずにいられませんでした。

道具と場 ── That's one small step……

となれば、あとは行動あるのみです。

マイコン文化。作り手と受け手が同じ目線に立つ世界。誰もがプログラミングをする世界。それを実現するためには、まずいろいろな道具立てが必要です。

ひとつはBASICにあたるプログラミング学習環境を開発し、いまならそれをオープンソースで世界に無償で提供すること。そして昔のマイコン雑誌にあたるゲームコンテストを開催し、プログラミングを志す人に腕試しの場と作品発表の機会を与える必要があります。いわばニコニコ動画のゲーム版です。ゲーム投稿サイトと、ゲームコンテスト。なにしろゲームはプログラミングのとてもいい題材の1つだし、ゲームなら、作る人だけでなく、遊びにくるだけの人も呼び寄せることができます。ちょうどMSXがそうだったようにです。

そして、その地ならしをしたら、次はプログラムを書けない人、書こうとも思わない人たちをどうやってクリエイティブな世界に惹き付けるか。その大きなヒントは、HyperCardです。

HyperCardなら、幼稚園の先生や主婦までもがコンテンツ作りに参加できます。子どもに読ませる絵本を作ったり、家計簿を作ったり、80年代から90年代前半にかけて、HyperCardでは無数のスタックが作られ、共有されました。まさに作り手と受け手がフラットになった世界だったのです。

これらの道具立ての考えが、enchant.jsと9leap、そしてenchantMOONへとつながります。

そういう環境までふくめて、まるごと再構築するにはいまはとてもいい時期です。ソフトウェアプラットフォーム戦争に終止符が打たれつつあり、どんなプラットフォームでもとりあえずHTML5は動作するようなことが期待されています。

HTML5に関する技術の多くは、オープンソースとして誰もが利用できるよう公開されています。もちろんその下のレイヤであるLinuxや、Android、WebKitまでもがすべてオープンソースです。こうした技術基盤の上で、流通性の高いアプリケーションを提供することは今や誰にでもできます。あと一歩なのです。HTML5アプリの作り方そのものは、(いまもですが)2011年3月当時はまだ、いま以上に確立されていませんでした。

HTML5アプリを誰もが簡単に作れるようになると、その先に広がるのはまったく新しいコンピュータの世界です。

誰もが作り手になり、誰もが作られたものを交換する時代。まるで年賀状をやりとりするように、誰もが自分を表現し、誰もが誰かの表現を受け取るような世界です。発信者と受信者という二元論から、双方向に発信し合い受信し合う世界。それこそが道具としてのコンピュータの次なる姿なのかもしれません。

いちどコンピュータの原点に立ち戻り、そういうパラダイムを目指そう。

この長い旅は、そんな思いつきからはじまりました。SD

コレクターが独断で選ぶ!

偏愛キーボード図鑑

濱野 聖人 HAMANO Kiyoto
khiker.mail@gmail.com
Twitter：@khiker

新連載

Dvorak/Colemak配列に切り替えられる TypeMatrix

写真1　TypeMatrix 2020

はじめに

本誌の2012年7月号のEmacs/Vim特集の際にキーボード紹介の記事を書かせていただいた御縁で今月号から毎月、筆者所有のキーボードをいくつか紹介させていただくこととなりました。どうぞよろしくお願いします。

できる限り現在でも購入が可能なものを中心に紹介していく予定です。第1回は、特殊なキーボード配列であるDvorak配列、Colemak配列[注1]を利用できる「TypeMatrix」を紹介します。

TypeMatrix

TypeMatrixはアメリカのTypeMatrix社が販売しているエルゴノミクスキーボードです。英語キーボードですので、OSのキーボード設定を英語に変更する必要があります。主な種類は次の3つがあります。

- TypeMatrix 2020（PS/2接続、**写真1**）
- TypeMatrix 2030（旧バージョン、PS/2接続、**写真2**）
- TypeMatrix 2030（現行バージョン、USB接続、**写真3**）

2020はQwertyのみ印字したバージョンとQwertyとDvorak両方を印字したバージョンの2種類が存在します。2030は印字なし／Qwerty印字／Dvorak印字／BÉPO印字[注2]の4種類が存在します。

また、TypeMatrix2030専用のキーボードカバーもあります（**写真4**）。色と印字されている配列が異なるタイプが複数種類あります。印字されている配列はDvorakだけでなく、ColemakやWorkmanのようなマイナーな配列もあります。

入手方法

TypeMatrix社の本家Webページ[注3]より購入ができます。価格は2020が$60、2030が$110、2030用キーボードカバーが$20です。なお、2020は本家Webページでは販売しておらず、現在のところ、eBayを通しての販売となっているようです。

特徴

- 薄型で軽量
- 格子状のキーボード配列

写真2　TypeMatrix 2030（旧バージョン）

写真3　TypeMatrix 2030（現行バージョン）

写真4　TypeMatrix 2030（Dvorakカバー装着）

注1）キーボード配列の種類です。一般的なキーボードのキー配列はQwerty配列と呼びます。
注2）フランス語用Dvorak配列です。
注3）`http://typematrix.com/`

- パンタグラフ機構
- 中央にある大きなエンターキーとバックスペースキー
- Dvorak/Colemakにワンタッチで切り替え可能

`	1	2	3	4	5	6	7	8	9	0	[	]	BS
TAB	'	,	.	p	y	f	g	c	r	l	/	=	\
Caps	a	o	e	u	i	d	h	t	n	s	-	Enter	
Shift	;	q	j	k	x	b	m	w	v	z	Shift		

図1　Dvorak配列

`	1	2	3	4	5	6	7	8	9	0	-	=	BS
TAB	q	w	f	p	g	j	l	u	y	;	[	]	\
BS	a	r	s	t	d	h	n	e	i	o	'	Enter	
Shift	z	x	c	v	b	k	m	,	.	/	Shift		

図2　Colemak配列

写真5　Dvorak配列時のLED点灯

薄型・軽量で、持ち運びも楽です。横幅は普通のテンキーレスキーボードより短いので、マウスの位置も近くなります。

キーボード配列は格子状配列を採用しています。格子状配列は慣れると指を上下にスライドさせるだけでよく、スムーズにキーを打てます。

キースイッチはパンタグラフ機構を採用しており、ノートPCと似た打鍵感です。打鍵音も静かで、2030の場合、専用のキーボードカバーを付けると打鍵音はほとんど気にならなくなります。

また、キーボードの機能として、キーボード配列を102キー配列、106キー配列、Dvorak配列、Colemak配列に変更する機能があります[注4]。たとえOSのキーボード設定が日本語キーボードのままでも、TypeMatrixを106キー配列にすれば、OSのキーボード設定を英語キーボードに変更することなく、すべての文字／記号を入力できます。

次に、Dvorak配列とColemak配列についてもう少し説明します。

Dvorak配列

Dvorak配列は**図1**のようなキーボード配列です。英文の入力に最適化された配列で、英文をよく入力する場合、良い選択肢となります。

2020、2030の旧／現行バージョンすべてで切り替え可能です。2020は[Fn]+[F7]で、2030の旧バージョンは[Fn]+[F3]で、現行バージョンは[Fn]+[F1]で切り替えられます。Dvorak配列に変更するとLEDが点灯します(**写真5**)。

Colemak配列

Colemak配列は**図2**のようなキーボード配列です。2030の現行バージョンでのみ切り替えが可能で、[Fn]+[F5]で切り替えられます。

Colemak配列は、2000年代になって登場した比較的新しいキーボード配列で次の特徴があります。

- **エルゴノミックで快適に打鍵可能**
- **Qwerty配列から移行しやすい**
- **入力のほとんどを強い指で行え、同じ指の打鍵率が低い**

英文を入力する際、Qwerty配列と比べて指の移動が少なくなるように設計されています。また、Qwerty配列からの移行コストも低いです。たとえば、キーの位置は[e]、[p]、[y]を除いて最大で2キー分しか移動しておらず、また、一番下の段は、[n]が[k]に変わっているだけです[注5]。

さらに、英文で比較的よく入力する文字は中指や人差し指といった比較的強い指で入力するように設計されているため、指にやさしいです。たとえば、[l]や[s]は、Qwerty配列では薬指、Dvorak配列では小指でしたが、Colemak配列では中指と人差し指で入力します[注6]。

もし新しいキーボード配列に挑戦しようと考えているならば、Dvorak配列だけでなく、Colemak配列も選択肢に入れると良いかもしれません。

まとめ

TypeMatrixはキーボードの設定でDvorak/Colemak配列に変更できる数少ないキーボードの1つです。とくにColemak配列に対応しているキーボードはこのTypeMatrixが唯一と言っても良いかもしれません。特殊な配列の練習用に1つ購入してみては如何でしょうか。SD

注4）102キー配列、106キー配列、Colemak配列への変更は、2030の現行バージョンのみです。
注5）そのため、Windowsでよく使われる[Ctrl]+[z]、[Ctrl]+[x]、[Ctrl]+[c]、[Ctrl]+[v]はそのまま使えます。
注6）UNIXユーザでよくあるDvorak配列にしたらlsが打ちづらくて困ったということはColemak配列だと回避できます。

はんだづけカフェなう

Raspberry PiでI/Oしてみよう(後編)

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

はじめに

焦電型赤外線センサモジュール選定のトラブルで1回mbedの話題を挟んでしまいましたが、その後良さそうなセンサモジュールで試作させることができました。

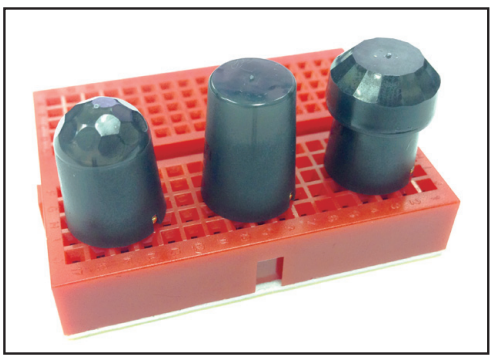
焦電型赤外線センサというのは、周囲と温度差のある人や動物などが動く際に起こる赤外線の変化を検出するセンサです。センサから出力される電気信号はとても弱いので、アンプ回路での増幅が必要になり、そういった周辺回路を組み込んだモジュールが、多くの会社によって生産されています。こういったモジュールは人が通ると自動的に点灯する照明などに使用されています。

今回筆者が入手して使ったモジュールは、Panasonic社のMPモーションセンサ NaPiOn(ナピオン)です。本体色が白と黒の2種類が用意されているのに加えて、NaPiOnにはデジタル出力型と、感度の調整が可能なアナログ出力型があります。また、NaPiOnには標準、微動、スポット、10mといった検出エリアが異なるタイプが存在します(**写真1**)。

筆者はとりあえず、白色の標準検出タイプを使ってみました。

NaPiOnを入手するにあたって調べてみたところ、我々の身近なところでは、千石電商[注1]か、共立電子[注2]で取り扱いがあるようです。今回は千石電商のネット通販を利用しました。

▼写真1　NaPiOn各種



※左から、標準検出タイプ、スポット検出タイプ、微動検出タイプ

センサをとりあえずつなげてみる

データシートを参照すると、NaPiOnの動作電圧はDC3～6Vとなっていますので、5V動作のArduinoでも3.3V動作のRaspberry Piでも使うことができそうです。また、検出時の出力電圧もVdd-0.5Vということで、どちらのマイコンボードでもすんなり使うことができそうです。ただし、「非検出時はオープン状態となります。」と書かれており、検出されていないときには回路がつながっていない状態になるということがわかります。オープン状態ですと外来ノイズの影響を受け、マイコン側で入力ピンの状態を読み取っても内容が不定となってしまいますので、プルダウン抵抗を付ける必要があることもわかりました。ここでは、モジュールの出力ピンとGND(0V)の間に10kΩの抵抗をプルダウン抵抗として付けることにします。

NaPiOnには3つのピンがありますが、ピン配置は**図1**のようになっています。Vddが電源、

注1)　http://www.sengoku.co.jp/mod/sgk_cart/search.php?multi=napion&cond8=and
注2)　http://eleshop.jp/shop/c/c110716/

OUTから出力となっています。

Raspberry Piにつなげてみる

とりあえず初めて使うセンサーですので、試しにブレッドボードで配線をしてみました（**写真2**、**図2**）。

このブレッドボードにRaspberry Piのピンを引き出している部品は、本連載前々回で紹介したAdafruitの製品で、最近スイッチサイエンスでも取り扱いが始まった[注3]ものです。

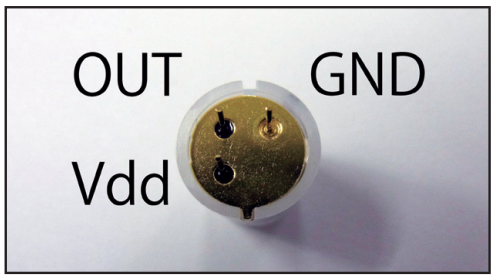
配線は先述のとおりNaPiOnのOUTに10kΩのプルダウン抵抗を接続しています。このOUTをRaspberry PiのGPIO 4と接続します。Raspberry Piの3.3VとGNDからNaPiOnに繋いで電源を供給するため、そちらの配線も行いました。

組んだら、簡単なスクリプト（**リスト1**）で実験をしてみましょう。このスクリプトは、1秒に1回GPIO 4の状態を見に行き、HIGH(検出状態)だったら"Detect"と表示をするだけのものです。GPIOを使用するのでroot権限が必要ですのでsudoして実行するのを忘れないようにしましょう。

センサモジュールに電源を投入してから最大45秒間は、回路安定時間ということでセンサ出力のON/OFF状態が定まらないということです。スクリプトはセンサモジュールを繋いでから1分以上経過してから実行するなどして、起動したての入力は無視してください。

スクリプトを実行してセンサに手のひらをかざすと、“Detect”という文字列が出力されました。またこれは焦電型赤外線センサの特性なのですが、センサは“生体が動く際に起こる変化”を検出します。このため、手のひらをかざして動かさずにいるとセンサはOFFになります。手のひらを少しでも動かしていると、センサはONのままでいます。こうして焦電型赤外線セ

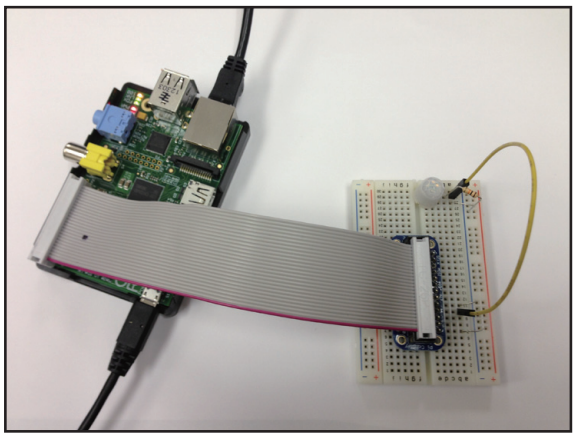
▼図1　NaPiOnのピン配置



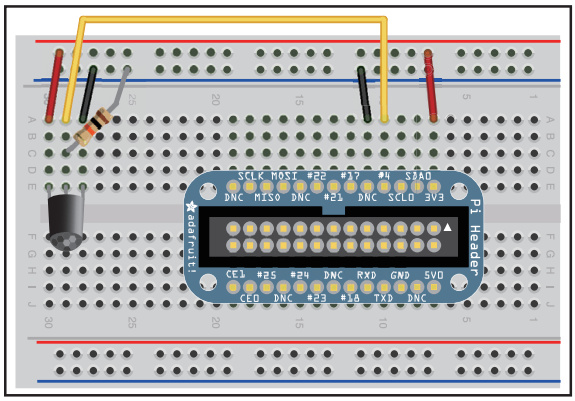
▼リスト1　実験用のスクリプト

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)

while True:
    input= GPIO.input(4)
    if(input):
        print("Detect")
    time.sleep(1)
```

▼写真2　組んでみた写真



▼図2　Fritzingの配線図



注3）http://www.switch-science.com/products/detail.php?product_id=1258

▼リスト2　/etc/network/interfaces

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet dhcp
#wpa-roam /etc/wpa_supplicant/ ➡
wpa_supplicant.conf
wpa-conf /etc/wpa_supplicant/ ➡
wpa_supplicant.conf
iface default inet dhcp
```

▼リスト3　/etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="SSID"
    scan_ssid=0
    key_mgmt=WPA-PSK
    proto=WPA2
    pairwise=CCMP
    group=CCMP
    psk=wpa_passphraseコマンドで得たキー
}
```

ンサの特性を把握しました。

Raspberry PiのWi-Fi接続

焦電型赤外線センサとWebカメラを接続したRaspberry Piを設置するにあたって、設置する場所にコンセントだけでなく、Ethernetケーブルを一緒に用意するのが面倒になったので、筆者はRaspberry PiにUSBのWi-Fiインターフェースを接続することにしました。筆者の手元にはBUFFALOのWLI-UC-GNM2[注4]という無線LANアダプタがありましたので、これをRaspberry Piに取り付け、自宅のWPA2-PSKを使用しているWi-Fiに、**リスト2、3**のような設定を追記するだけで接続できました。

なお、この設定ファイルに記述するキーは、「**$ wpa_passphrase SSID パスフレーズ**」を実行することで得ることができます。

コンフィグを書き終えたら、次のように無線LANインターフェースを起こし、Wi-Fiにつながることを確認します。

```
$ sudo ifdown wlan0
$ sudo ifup wlan0
```

正常に接続できることが確認できたら、次回起動時からWi-Fiに自動的に接続され、Ethernetが不要になります。なお、Raspberry Pi のUSBポートは一般的なUSBポートよりも給電能力が低いです。このため、安定動作させるにはセルフパワーのUSB HUBを接続することをお勧めします。

センサをトリガに写真を撮ってみる

リスト4のようなPythonスクリプトで写真を撮って呟くことができました。第29回ではTwythonを使っていましたが、なぜか本稿執筆時に実験した際にはうまく動かず、Tweepyの改良版を使用しています。

この改良版はこちら[注5]で公開されており、次のような手順でインストールしました。

```
$ git clone git://github.com/laiso/tweepy.git
$ cd tweepy/
$ sudo python setup.py install
```

トークンについては、第29回で紹介したとおり、TwitterのDevelopersサイト[注6]でアプリケーションの登録をして取得をします。これで無事に焦電型赤外線センサモジュールを使って動く生体を検出し、写真を撮り、呟くことができました。

なお、Twitterは先日よりAPI 1.0を一時的に停止させるブラックアウトテストが行われるなど、状況が逐次変わってきています。このライブラリもいつまで快適に利用できるかどうかは分かりませんので注意が必要です。

注4）http://buffalo.jp/product/wireless-lan/client/wli-uc-gnm2/

注5）https://github.com/laiso/tweepy

注6）https://dev.twitter.com/apps

▼リスト4　Pythonスクリプト

```
#!/usr/bin/env python
# -*- coding: utf-8; -*-

import time
import RPi.GPIO as GPIO
from SimpleCV import Camera
import tweepy

GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)

CONSUMER = 'TwitterのCONSUMER_KEY'
CONSUMER_SECRET = 'TwitterのCONSUMER_SECRET'
ACCESS_TOKEN = 'TwitterのACCESS_TOKEN'
ACCESS_TOKEN_SECRET = 'TwitterのACCESS_TOKEN_SECRET'

auth = tweepy.OAuthHandler(CONSUMER, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)

#myCamera = Camera(prop_set={'width':640, 'height': 480})
myCamera = Camera(prop_set={'width':320, 'height': 240})

while True:
    input= GPIO.input(4)
    if(input):
        print("Detect")
        frame = myCamera.getImage()
        timestamp = time.strftime("%d%m%Y-%H%M%S")
        frame.save("/home/pi/temp.jpg")
        api.status_update_with_media("/home/pi/temp.jpg", status="@ytsuboi " + timestamp)
        print timestamp, "tweeted"
        time.sleep(10)
    time.sleep(1)
```

最後に

この原稿を書くにあたって、久しぶりにUSBカメラをRaspberry Piに接続したのですが、普段使いにしていたLogicoolのカメラを接続して実験してみたところ、うまく撮影をできずにハマってしまいました。筆者はSimpleCVを使って撮影をしていますが、pygameでも同様のことができるようですので、SimpleCVでうまく動かない場合はpygameを試してみるのも良いでしょう。

また、今回はMicrosoftのLifeCam HD-3000[注7]を使用しました。このカメラでは、640×480ピクセルの写真を撮るとJPEGが少し壊れてしまうので、320×240ピクセルで撮影するようにしました（写真3）。お手持ちのWebカメラで写真を撮る際にうまくいかないなと感じたら、ここを変更してみるのも手かもしれません。SD

▼写真3　撮影された写真

注7）http://www.microsoft.com/japan/hardware/lifecam/hd_3000.mspx

PRESENT 読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト http://sd.gihyo.jp/ の「読者アンケートと資料請求」からアクセスし、アンケートにご協力ください。ご希望のプレゼント番号をご入力いただいた方には抽選でプレゼントを差し上げます。締め切りは2013年5月17日です。プレゼントの発送まで日数がかかる場合がありますが、ご了承ください。

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

01 スマートフォン用車載ホルダー形FMトランスミッター

1名

車載用ホルダーとFMトランスミッターが一体化しました。各種スマートフォンをステレオミニプラグでFMトランスミッターに接続し、手軽に音楽を再生できます。シガーソケットからのスマートフォン充電も可能です。

提供元 上海問屋
URL http://www.donya.jp

02 ツイスパソーダスターターキット

2名

水、ジュース、アルコール飲料などを簡単に炭酸飲料にすることができる炭酸水製造マシン。オリジナルの炭酸ドリンクを作ったり、気の抜けた炭酸を復活させたりと、幅広く利用できます。

提供元 グリーンハウス
URL http://www.green-house.co.jp

03 ピコットフセン

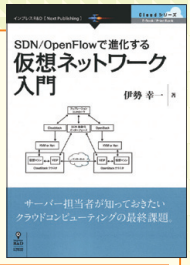
3名

スマートフォンのアルバム内の写真をQRコードに登録し、手書きの記録とリンク付けできる付箋。リンク済みの付箋を手帳などに貼れば、手書きの記録とともに目的の写真をすぐに閲覧できます。

提供元 カンミ堂
URL http://www.kanmido.co.jp

04 SDN/OpenFlowで進化する仮想ネットワーク入門

2名



伊勢 幸一 著／
A5判、198ページ／
ISBN＝978-4-8443-9575-1

SDN/OpenFlowを検討するエンジニアやビジネスマン向けに、仮想ネットワークの基礎知識を網羅した1冊。専門用語の意味や背景から、個別に参照すべき規格名や標準化資料名までを幅広く解説。

提供元 インプレスR&D URL http://www.impressrd.jp

05 パーフェクトPython

2名

Python サポーターズ 著／
B5変型判、464ページ／
ISBN＝978-4-7741-5539-5

1冊で言語仕様から最新の技術までを取り扱う。各技術に関しては基本からしっかり解説し、必要な個所では、内部処理が裏で何をしているのかを掘り下げて解説する。最新のPython3.3に対応。

提供元 技術評論社 URL http://gihyo.jp

06 Windows 8［業務アプリ］開発読本

2名

技術評論社編集部 編／
B5判、176ページ／
ISBN＝978-4-7741-5605-7

ストアアプリとデスクトップアプリの特徴と使い分け、Visual Studio Expressによる開発のツボ、タッチデバイス対応のポイントなど、Windows 8ベースの業務システム構築に重要な技術を解説。

提供元 技術評論社 URL http://gihyo.jp

07 Software Design総集編【2001～2012】

2名

Software Design 編集部 編／
B5判、100ページ／
ISBN＝978-4-7741-5593-7

『Software Design』の2001年1月号～2012年12月号の特集、連載、一般記事、特別企画などの記事をPDFにしてDVD-ROMに収録しました。12年分のIT技術／ノウハウが詰まった1冊です。

提供元 技術評論社 URL http://gihyo.jp

第1特集

Date 2013.5.18

新しい季節へ君と

IT業界ビギナーのためのお勧め書籍55冊+α

新人を育てることは、会社の将来における発展／成長につながります。

IT業界で実績を上げている筆者自ら探し当て、技術の源としてきた本当にお勧めの本と、ビジネスや自分のこころのあり方の柱となる本を＋αとして紹介します。重複して紹介された本もありますが、多面的な評価として読んでみるのも一興です。また、あえて絶版本も紹介しました。古書店や図書館などで探すのも勉強のうちです。求めよさらば与えられん！　難しい本こそ血肉になる率高し！

本を紹介してくれた筆者の皆さんは、システムエンジニア、プログラマ、ネットワークエンジニア、インフラエンジニア、サイエンティスト、ITジャーナリストなど、幅広く19名の方々です。

Writers

池田秀一、伊勢幸一、大石良、大久保修一、柏野雄太、首藤一幸、谷口有近、谷本真由美、中島聡、並河祐貴、西澤無我、長谷川猛、藤田稜、藤本真樹、法林浩之、星暁雄、元井正明、山下秀登、湯本堅隆（敬称略、50音順）

IT業界で、コンピュータ用語以外に押さえておくネタがあります。ぶっちゃけそれは日本のアニメーションやコミック、ライトノベルなどです（**ドンッ！**）。「ファーストガンダム」はとくに研究すべき課題といえましょう（キリリ）。「まど★マギ」も必修科目です。教科書どおりじゃないのがこのIT業界ですが、無闇に勉強するよりも、優秀なガイドさんが必要です。本特集では、とくに突き抜けてエキスパートの皆さんに紹介をしてもらいました。

Writers

砂金信一郎、小飼弾、前佛雅人、田中邦裕（敬称略、50音順）

Part1、Part2を通して読みたい本を探せば、
きっと視界が広くなって柔軟な考え方ができるようになるかもしれません！
迷ったら、本特集を片手に本を探しにいきましょう！

1 天邪鬼な事業会社のSEへ

IT，Systems Engineer

いきなり本特集の趣旨を否定するような暴言ですが、勧められた書籍って身につかない気がしません？筆者が携わってきた仕事は、よく言えばカッティングエッジ、悪く言えば情報がなんもなく基本手探り。正解がないって言えば聞こえは良いですが、仕事で覚えたってのも多いです。なので、有名どころの技術書は、あとからその存在を知って話題のネタとして買ってはみたんだけど積読ってのが多々あります。

理解を深め身につけるには、その情報を必要とする強いモチベーションってのが重要なんですよ。なので、書籍の内容の紹介っていうよりは、そこに到達するまでに至った経緯を軸に、筆者のこれまでの主たる経験である「事業会社でのビジネス寄りエンジニア」という立場で効果がありそうなネタをご紹介。

未知を知るのに月刊誌の連載はほど良い

新人SEのための ネットワーク入門

戸根 勤、日経オープンシステム(著)、日経BP社、2002年
※完売／入手困難

まず、他社のムックを紹介するあたりなかなかのKYです。学生時代、モラトリアムをトレースしつつも、「面接大嫌いの社会不適合なオレが企業に就職せずとも飯を食っていくにはビジネスというものを勉強せねばならん！」と前向きなのか後ろ向きなのか不明な決心をして、なけなしの生活費から新聞代と日経ビジネスの購読代を捻出して読んでたんです。で、もともと好きだったパソコンの世界にデカい進歩があり、どうやら今後はインターネットという技術が世界を変えていきそうですぜ、ってのを知るんですな。同世代には詳しいヤツがほとんどどいなかったので、ヨシ、オレでもMSDNがあればいけるんじゃね、ってなノリで、BASICの世界から飛躍しWeb系に手を出して、在学中に派遣会社を通じて仕事を請け負うようになりました。

本の内容を説明してないけど忘れてないよ。もうちょい待たれい。

で、学生派遣の会社で仕事し初めて気がつくわけです。「秀才ばっかりじゃんこれ！」やばいわけです。甘過ぎたわけです。あかん勉強せな死ぬ、と。幸いオフィスには月刊や隔週のIT系雑誌やビジネス雑誌が落っこちていて(タダで読めることを良いことに)片っ端から借りて中身を浚っていったんですね。その際に読んでた日経オープンシステムという雑誌に連載されていたのが『新人SEのためのネットワーク入門』です。

このムックの良いところは、インターネットの基本となるTCP/IPやSMTP/HTTP/DNSといったしくみを、実際のコマンドラインなどの操作で演習し学べるところ。連載記事なので図示が多く可視化(笑)されているので悩まない。SMTPコマンドをtelnetで叩いて、メールの発信元を偽装できたときは、ちょっと感動したのを覚えてます。

筆者は物事を一度に理解しようとしても無理な人でして、覚える量が多いと飽きるんですな。なので、知りたい領域があれば、まず専門誌を1年間は読むと。業界のブームを把握しつつ基本をカバーする連載記事で土台を肉付けしてくってのが、とっても有効でした。ここで得た知識は、迅速なデバッギングや新技術を用いた基盤系の設計などで、大いに役立ってます。

PROFILE

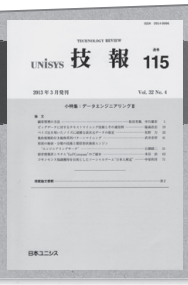
谷口有近（たにぐちありちか）　Twitter：@arichika

Ajaxがない時代に開発した独自の非同期通信技術を売りに、IT業界向け学生派遣の受託部門やベンチャー企業にて開発を担当。一転、ネット専業オンライン証券会社へ。ネットワークやIA基盤の構築、開発と運用の橋渡しやビジネス開発などを担当し、11年には社長付IT戦略担当に就任、現在は独立。事例や講演も多数。

無料なのに個性的

UNISYS技報

日本ユニシス株式会社
※Webページより申し込み
http://www.unisys.so.jp/tech_inf/



仕事仲間や取引先が理解してる標準的なネタを知らないと、話題に乗り遅れて白い目で見られるってどころか、まったく仕事にならんこともありますが、とはいえ、さも「これが先端だ！」と言わんばかりの情強ネタを披露したところで仕事はいっこうに楽になりませんし、むしろムダな反論を上司に叩きつけてややこしくなることのほうが多いわけです。

一方、その業界の常識だけに詳しくなっても、どうもこのご時世、それでは長続きしないって感じです。天文学と物理学の量子視点での融合は世界を前進させてますし、ビッグデータってな卑近な例に漏れず、数学の幅の利かせっぷりは、たいへんなものがあります。

我が身を振り返りつつ乱暴に言い切れば、何かを解決しなきゃならない際、その世界に閉じた情報(≒つまり現在～過去)でのみ思考してると、なっかなか解決しないんだぜってことを、もっと早く実感してりゃー、もうちょい給料上がってたかもな、と。領域をクロスオーバーし止揚(しよう)し続けていかないと、早々に自らの感覚の限界にぶつかってしまうわけで、自然科学的なアプローチを武器として理解し実践することの意味を、10～20代で実感しておきたかった。

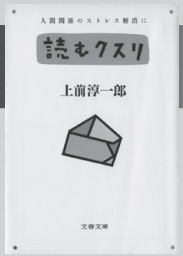
自らの領域外をどう知るか。これは20代後半から意識的に実践してたテーマ。雑誌やWebは既知と重複しがち。飲みニケーションは非効率で無理。そうこうで発見した1つが、企業が顧客を啓蒙しつつ技術を売りつけるために作る技術情報誌や、社内の士気を高めるために開示する技術論文集でして、この類(たぐ)いの文献が、なかなかに面白いです。

NTTドコモさんの『テクニカル・ジャーナル』や日本ユニシスさんの『UNISYS技報』は無料で得ることができる技術論文集です。掲載内容が業界横断的で、それでも技術の観点で記事は理解可能なので、「なるほど！」ってな発見に出会えることもあり、自らの専門領域にもつながるであろうヒントが得られます。SD

+α 温故知新

読むクスリ――人間関係のストレス解消に

上前淳一郎(著)、文藝春秋、1999年



中学時代に入院する機会がありましてね。担任が見舞いに持ってきてくれた1冊が、文庫版の第1巻。ビジネスの現場で起きたさまざまなエピソードを読みやすくまとめた“企業版ちょっといい話”で、文藝春秋に連載されていました。ストレス社会に立ち向かうビジネスマンにとって、会話やアイデアのネタにもってこいの内容で、“ちょっといい”感じが、一服の清涼剤ともなるような小話集。

ただの田舎者で気持ちだけはませた中学生には、なかなか衝撃的な内容でして。「あぁ、会社ってこれほどにもエキサイティングでチャレンジングで、切り開いていくことが評価されるところなのか！」と派手な勘違いをするわけです。まぁ間違いではないんですが、書かれてない事情ってのを大人になって知ることになります。その衝撃たるやもう。バブル時代とその後10年を追体験できる良書です。

Date. 2013 . 5 . 18

2 自分の価値の高め方

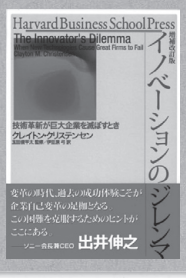
IT，Systems Engineer

さまざまな業界でソフトウェアが重要になって来た今、「どんなソフトウェアを、どんなしくみで、どのくらいのコストをかけて、どのタイミングで提供するのか」などのさまざまな判断をエンジニア自身ができるようにならなければならないと筆者は考えています。その意味では、単にプログラミングのスキルを学ぶだけではなく、業界全体の流れを読む力だとか、市場のニーズを読み取る力などを身に付けることが重要で、それが自分の人材としての価値を高めることにつながるのです。

IT業界の疑問に応える良書

『増補改訂版 イノベーションのジレンマ――技術革新が巨大企業を滅ぼすとき（Harvard business school press）』

クレイトン・クリステンセン（著）、玉田俊平太（監修）、伊豆原弓（翻訳）、翔泳社、2001年、2,100円



この業界のリーダーがなぜ数年置きに入れ替わるのか、なぜ資本力も人材も豊富に持つ大企業よりも、ベンチャー企業のほうが画期的なイノベーション起こすことができるのかをわかりやすく説明した良書で、「エンジニアにとってのバイブル」と呼んでも良い必読書です。

筆者自身にとっても、13年間働いたMicrosoftを辞めて自分で会社を起業するというキッカケを与えてくれたという意味でとても重要な書物です。

Microsoftは、90年代には、Windows 95/Windows NT、Office 95、Internet Explorer 3.0/4.0などのソフトウェアをリリースし、パソコン業界で圧倒的な地位を占めていました。しかし、ネットやモバイルの重要性が増すにつれ、リーダーシップの座は、Google、Amazon、Appleへと移ってしまいました。

筆者は、1999年末という、まさに時代の転換期に本書を読むことにより、イノベーションを起こしたいのであればMicrosoftを飛び出す必要がある、と気づくことができたのです。

大企業の中で閉塞感を感じているエンジニアや、起業を考えているエンジニアにはとくにお勧めの1冊です。

知識労働者のための本

ドラッカー名著集1 経営者の条件

P.F.ドラッカー（著）、ダイヤモンド社、2006年、1,890円

日本語のタイトルが「経営者の条件」となっているため、会社の経営者のための書物だと勘違いする人がいますが、本書はソフトウェア・エンジニアに代表される「21世紀の知識労働者」のために書かれた本です。

本書は、知識労働者と企業との関係が、肉体労働者のそれとは大きく異なることを明確にした上で、人材市場での自分の価値を高めるためには何をすれば良いのか、そして、会社とどんな関係を築くべきかに関して、さまざまな面か

PROFILE

中島聡（なかじまさとし）　Blog：http://satoshi.blogs.com/

Microsoft本社でソフトウェア・アーキテクトとしてWindows 95、Internet Explorer 3.0/4.0をリリース。2000年に独立し、UIEvolution（本社米国シアトル）を設立。現在はneu.Pen LLCでiPhoneアプリケーションの開発をしつつ、メルマガ「週刊 Life is Beautiful」を発行。工学修士／MBA。

ら考察を加えています。

日本では、「プログラマの3K（キツい、給料安い、帰れない）」「ブラック企業」「IT土方」などの言葉が一人歩きをしています。そんな状況が原因なのか、大学では情報系の学科の人気が落ちているそうです。

一方では、米国のシリコン・バレーは毎年のように数千人単位で億万長者を生み出しており、ソフトウェア・エンジニアはまさに知的労働者の花形です。優秀なソフトウェア・エンジニアとは企業にとっては「価値を生み出す宝」なのです。

この違いを生み出している原因はどこにあるのか、そして、そんな状況を打破するには何をすれば良いのか、というヒントを本書は与えてくれると思います。SD

▲Life is beautiful「永遠のパソコン少年のうんちく」

◀Web+DB Press誌での連載も注目。エンジニアにとってどう生きるべきかという指針を示している

＋α　自分を理解するための文章トレーニング

理科系の作文技術

木下是雄（著）、中央公論新社、1981年、735円

理科系まっしぐらの人生を送って来た筆者は、この本を読むまでは「自分は作文が下手だ」と思い込んできましたが、本書のおかげで文章を書くことが好きになり、今では毎週のように雑誌のコラムやメルマガにプロの執筆者として文章を書くようになりました。

文章の一番の目的は、学校の国語の授業で教わるような「人に感動を与えること」や「美しい文章を書く」ことにあるのではなく、「情報をわかりやすく的確に伝えること」にあることを本書は教えてくれます。そこさえ理解できれば、文章を書くことはものすごく楽になるし、楽しくもなります。逆に、そこをしっかりとやらなければ、人を感動させることも美しい文章を書くことも不可能なのです。

「読み手のことを考えて、可能な限りわかりやすく記述する」というコンセプトは、文章だけでなく、プレゼン資料、ユーザ・インターフェースの設計、そして、プログラムそのものにも通じる普遍的なコンセプトです。

コミュニケーションの重要さを再認識させてくれる、という意味でも、ぜひともソフトウェア・エンジニア全員に読んでいただきたい1冊です。

3 成長できるエンジニア、成長できないエンジニア

IT，Systems Engineer，SIer

もう10年前になりますが、筆者自身も新卒でIT業界に飛び込みました。当時はまったくと言っていいほどプログラミングができず入社してからキャッチアップしました。振り返って、活躍している方々に共通しているのは「なぜ？」に対して非常に誠実であること、そして俯瞰的視点を持っていることに気づきました。エンジニアにとってプログラミングはすべての基本であると同時に、あくまで手段でしかありません。重要なのは、プログラミングスキルという最高の武器を使ってどうやって素材を調理してすばらしい料理へと進化させ、新しい世界を築いていくのかにあります。その素地を付けるために重要な2冊を紹介しましょう。

プログラミング言語を問わず、普遍的なスキルを身に付ける

楽々ERDレッスン

㈱スターロジック羽生章洋（著）、翔泳社、2006年、2,310円

筆者が自社の業務システムを作るにあたって一番参考になったのがこの本です。業務システムの開発では、現実社会の商慣習を噛み砕いて、そのうえでどのように設計してシステムに落とし込むのかという工程が非常に大切になります。その工程の根本になるのが、データベースの設計です。どのようなテーブルを作り、どんなカラムを用意して、テーブル間の整合性をどう取っていくのか。データベースの構造設計はシステムの大黒柱です。ここで大きく道を踏み外してしまうと、非常に高い技術を持つスーパープログラマが頑張ってもどうにもなりません。データベースの構造が複雑怪奇になると、プログラムのロジックも複雑になりメンテナンスが困難なシステムになります。

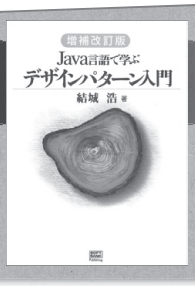
また、たとえばユースケースや業務シナリオなどを記述してシステム化したい物事を整理できたとしても、それをプログラムに落とし込むにはデータベースの設計をしなければなりません。これができなければ、単なる絵空事になります。

新卒で入社されるエンジニアの皆さんには、まずこの本を読むことでスマートなデータベース設計の考え方を知ってもらい、現実でいつか渡されるであろうデータベース仕様書を見ながら比較できるようになってほしい。「なぜ、この設計なのか？　こうすればもっと柔軟ではないか」――そういった議論ができる素地を鍛えてほしい。この種のトレーニングが技術力を鍛えるのに一番有効です。言語が変わっても活用できる普遍的なスキルを身に付けることにつながります。

成長し続けるエンジニアの秘密とは

増補改訂版 Java言語で学ぶデザインパターン入門

結城浩（著）、ソフトバンククリエイティブ、2004年、3,990円



プログラミング初心者のころは、与えられた課題をどのようにプログラミングするかで手一

PROFILE

湯本堅隆（ゆもとみちたか）　Twitter：@gothedistance

1979年生まれ。2003年アイ・ティ・フロンティアに入社。プログラマ・リーダー・PM・コンサルを経験し、2009年4月にインテリア用品の製造・卸企業であるエフ・ケーコーポレーションへ転職。ひとり情報システム部として、自社業務システムの開発・運用を行いつつ、新規事業開発に従事しています。

杯になります。どうプログラムを紡いで実行すれば良いのかも四苦八苦するからです。1年もすれば慣れてきます。しかし、ある程度慣れてきたころが一番危険で、ここから成長が加速するエンジニアと成長がピタっと止まるエンジニアに別れます。具体的には、より優れた解決策を追い求めて歩みを止めない人と、動いていることで満足して学習しなくなる人に分かれてしまいます。会社に頼らずとも食っていけるエンジニアになるためには、後者の人になってはいけません。なので、自分の書いたコードがイケてないことを知るために、プログラミングの設計技法のベストプラクティスを学びましょう。「上級者への道のりは、己が下手さを知りて一歩目」と申します。安西先生のお言葉です。

本書はオブジェクト指向型言語を利用して、現実の出来事をシステムに落とし込むときに問題となる複雑さを設計によって解決する手段がたくさん用意されています。設計の肝（きも）は、複雑なことに構造を与えて単純なものにすることです。複雑なものをそのまま複雑なままに実現しようとすると破綻します。できるエンジニアは常に自分の頭に地図を書いて単純なものにしようと設計しています。メンテナンスが楽になることを肝に銘じて知っているからです。

本書を読むにあたっては、現場のコードを見て「これはひどい」と思ったときがベストのタイミングです。自分で書いたコードで自分の首を絞めて苦労したときもまた最良のときです。ひどさがわからないと、すばらしさも理解できません。生兵法は大怪我のもとです。デザインパターンも万能ではありませんので、注意してくださいね。

ですが、このデザインパターンを知って利用させてもらったおかげで、筆者1人でWindows／ガラケーやスマホ／iPadのマルチデバイス環境で利用している、結構複雑な業務システムを運用できています。運用するとは、進化をうながすためにコードに手を入れることを意味します。屋台骨が揺らぐことがないので、どんどん手を入れて作りを変えていくことが可能になります。運用とは現状維持ではありません。維持しかできないなら、それは放置と同じことです。システムは作ってからが始まりです。いかに継続的に改善ができるかによって、資産価値も大きく変わります。

自分の書いたコードがダメな理由を知るトレーニングを積んでいる人とそうでない人では、30歳を境にスキルセットに非常に差がつきます。SD

＋α　心のカンフル剤として

非属の才能

山田玲司（著）、光文社、2007年、735円

新卒で会社に入ると一番怖いのが、今まで就業経験がないので、その会社の常識が社会全体の常識であるかのような錯覚に陥ることです。で、会社でちゃんと働いていれば必ずと言っていいほど「そうは言うけどさ……」と飲み込むのに納得がいかなかったり、我慢ができないときがやってきます。そのときにこの本を読んで元気を出してほしいと思います。常識を疑っているから自分を確立できます。優秀なエンジニアは、みなさんどこか本書でいう非属の才能を持っています。違和感というのはとても大切なシグナルです。僕も違和感だらけの人生を送っています。だからこそ、何かを創って変えてやろうという気が起きるのですけれどね。

Date. 2013 . 5 . 18

4 人生は長く、技術と市場の変化はとにかく速い

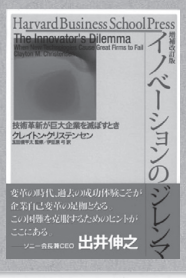
IT，Systems Engineer

「変化を乗り切り、正確なコミュニケーションを学び、情報のフロンティアを想像する」――IT業界に長い間身を置いている人は、変化に負けない、ある種の「必殺技」を身につけています。今回は「変化に負けない」という観点で本を選んでみました。1冊目は職業人生の中で必ず体験するであろう変化の実態を知る本、2冊目は書き言葉によるコミュニケーションの原則と技術を学ぶ本です。さらにもう1冊、変化に耐えて残った理論の本を選んでみました。

自分のキャリアの途中で起こるであろう変化の正体を知る

『増補改訂版 イノベーションのジレンマ――技術革新が巨大企業を滅ぼすとき（Harvard business school press）』

クレイトン・クリステンセン（著）、玉田俊平太（監修）、伊豆原弓（訳）、翔泳社、2001年、2,100円



IT分野で起きる変化の正体を理解するうえで欠かせない考え方が、本書が明らかにした「破壊的イノベーション」の理論です。

有力な企業が優れた経営を実施して、製品の完成度も向上させ、顧客からも高い支持を得ているとします――それにもかかわらず、新しく登場したいっけんチープな製品に破れてしまう現象が、歴史上、何回も繰り返されました。1990年代初頭に起こった大型汎用コンピュータの巨人IBMの不調と、同時に起きていた小型コンピュータの台頭がその一例です。今は時代が一回りして、パソコン関連企業の不調とスマートフォン／タブレットの台頭が同時に起こっています。このような現象に名前を付け、事例研究をした成果が、この『イノベーションのジレンマ』です。

「よく売れている優秀な製品・サービスを、優秀なスタッフが改善し続けること」だけでは、新興勢力との地位の交替を防ぐことができない――まさにジレンマです。

本書はこの「高度すぎる技術が市場で負ける」という恐ろしい現象を冷徹に記述しています。

非常に有名な本なので、読者の皆さんも内容の概要をどこかで目にしたことがあるかもしれません。ですが、ITに関わる人であれば、ダイジェストや伝聞で知るだけでなく、この本そのものを通読したほうが良いと思います。IT業界に身を置く限り、自分の長い職業人生のどこかで必ず破壊的イノベーションに遭遇するはずだからです。

事実、論理を誤解なく伝えるコミュニケーションの方法を学ぶ

理科系の作文技術

木下是雄（著）、中央公論新社、1981年、735円

本書の趣旨は「理科系の人向け」に文章の書き方を指南する本ですが、あまりにも内容が優れていたので、当初の狙いよりはるかに広い対象の人々に、しかも30年以上の長い年月にわたっ

PROFILE

星暁雄（ほしあきお）　Blog：http://hoshi.air-nifty.com/

ITジャーナリスト。1986年から『日経エレクトロニクス』記者、1997年からオンラインマガジン『日経Javaレビュー』編集長などの経験を積む。2006年に独立、現在はフリーランスとして活動。最近はスマートデバイスの急成長や、現実世界とソフトウェアを結ぶ技術にとくに注目している。

て読み継がれてきました。時代の変化に耐えたコミュニケーション方法の名著といえます。

日本の国語教育を受けてきた人なら「思ったこと、感じたこと」を表現する訓練には慣れ親しんでいると思います。その一方で、「事実と論理」を意見や感想ときちんと切り離して誤解なく伝えている日本語の文章は希少です。それでも著者は「私は、わが愛する日本語は、事実や論理を冷徹に、明確に伝える能力を内蔵した言語だと信じている」と書いています。

IT業界では書き言葉によるコミュニケーションが決定的に重要です。「事実と論理を、誤解なくわかりやすく伝える」基本的な訓練ができているか否かで、コミュニケーションの精度、効率は大きく変わり、それは仕事の質そのものに影響を及ぼします。こうした現場のコミュニケーションの向上のために即効性がある本といっていいと思います。

即効性だけではありません。本書は、基本的な考え方から個別の文章技法上のノウハウに至るまでが、とても高い基準で書かれています。ただの実用書というより、著者の思想が結晶したような1冊となっているのです。文章の技術を高めるには、「良い文章を読むこと」が欠かせないわけですが、この本を読む行為自体が読む人の文章表現の水準を高める方向に作用するはずです。

ところで、この本は個人的にも思い出深い1冊です。学生時代には卒論と修論の参考に読み、雑誌出版社に入ってからは新人研修の教材として渡され、同じ本を2冊持っていたのです。今でも時々読み返します。この本は私を育ててくれました。必ずあなたの助けにもなってくれるはずです。SD

＋α　変化に耐える理論と思想

認識とパタン

渡辺慧（著）、岩波書店、1978年、入手困難／絶版

初版が1978年ですから、もう35年も前に出版された本です。IT業界の人全員が読むべき、とまではいいませんが、「人と違うことを考える」立場の人にとっては貴重なヒントを得られる1冊だと思います。そして、本書が述べている機械学習とパターン認識のアプローチは、豊富な計算資源や公開データ群を駆使できる今の時代にはさらに応用範囲が広がっているのではないかと思います。

今この分野を勉強中の人は、「パンチカードで計算機を扱っていた時代からこんなことを考えていたのか！」と驚く人がいるかもしれません。機械学習の分野は、その歴史の途中で不遇の時期があり、その後見直しが進んだという経緯があります。何十年も前の人が考えた知見が、今の技術との組み合わせにより、大きな成果を生む場合があるかもしれません。それだけでなく、長い時間の変化に耐えた「考え方」を知ることは、今後起きる変化を乗り切るうえで勇気を与えてくれるとも思います。また著者の姿勢は「理科と哲学の交流」を目指すもので、こうした高い目線と実証的な論の進め方に触れることも、良い読書経験になると思います。なお、この分野の系統だった勉強をしたい場合には教科書として『パターン認識と機械学習』（C・M・ビショップ（著）、丸善出版、2012年）があります。

5 多くのエンジニアに足りない本とは何か

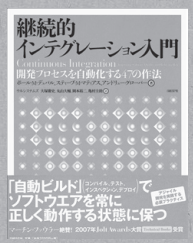
IT，Systems Engineer

昔話になってしまいますが、僕がプログラミングを始めた中学生のころはまだネットワークを通じた情報の入手は一般的ではなく、主な情報源は雑誌と書籍でした。毎週末神保町の大きな書店に通い、自分の知らないことばかり書かれている本を見ては不安になったり、新しい技術にワクワクしたりしながら、これぞ、と選んで少しずつ買ってきた多くの技術書は今のぼくのエンジニアとしての原点となっています。今はもう、書籍以外の多くの手段で多くの情報が得られますし、書籍を選ぶのも多くのレビューや評判を基に買うことが当然のことになっているので、書籍の位置づけ、選び方はまったく異なるものになっていますが、書店でコンピュータ書籍の多さを見てこんなに多くのことを知らなければいけないのかと絶望したり、1冊1冊手に取って自分に合う本を探したり、その過程で自分が何を勉強しなければいけないのかを真剣に考えたり、というのはそのプロセス自体に意味があると思いますので、新卒の皆さんなりに、今現在ならではの書籍の活かし方を見つけていただければと思う今日このころです。そして、その良いきっかけにでもなれば、ということで僕からも数冊オススメさせていただきます。

やる気にさせるCIの本

継続的インテグレーション入門
――開発プロセスを自動化する47の作法

ポール・M・デュバル、スティーブ・M・マティアス、アンドリュー・グローバー(著)、大塚庸史、丸山大輔、岡本裕二、亀村圭助(訳)、日経BP、2009年、3,360円



僕は、弊社に興味をもってくださった方との面接もしていたりするのですが、いわゆる業務経験がない方とお話しをしていると、セキュリティとテストの2点で認識にギャップを感じることが多いかな、と思っています。

今回は新人の方々向けの書籍紹介、ということなので、この2つの分野から1冊ずつ挙げてみます。もちろんそれ以外にも基本となる書籍、あるいは議論をしていくときの共通言語となる書籍は多くあると思うのですが(個人的には、G.M.ワインバーグの書籍もお勧めしたいです)、きっと他のみなさんが紹介してくださるに違いないと勝手に信じてそこはパスしてみます。

ということで、これはテストというかCI(Continuous Integration)に関する書籍ですが、CIやテストについてはこれ以外にも多くあり、どうしてもこの本でなければ、ということではありません。が、最初の1冊としてはわかりやすく、そして一番大事なのが「なんとなくやる気にさせてくれる」というところかと思っていますので、もしTDD(Test-Driven Develop-ment)やCIに関する本を読んだことがないのでしたら、まずはこちらを読んでみてはいかがでしょうか。個人的にはテストやCIは、最初のうちに癖を付けておく、というか「そういうものだ」と思うのが大事かなと思っていますので、ぜひ後回しにせず始めてみることをお勧めします。僕がプログラミングを始めたころはTDDやCIという概念もなかったですし、初期のころはツールやノウハウも充実していなくてむしろ苦労が多いだけじゃないかと思ったり、TDDやAgile開発でよく名前のあがるKent

PROFILE

藤本真樹（ふじもとまさき）　Twitter：@masaki_fujimoto

1979年2月17日生、中学生時代にPC-9801で初めてのプログラミング、そのあとなぜか英文学科を卒業して、オープンソースソフトウェア関連の活動をしながら、人が少なめの会社でWebサービス開発中心の業務に携わったあと、さらに人の少ないグリー株式会社に転職して、数年経ったら多くの仲間が集まる会社になっていました。2013年現在、グリー株式会社取締役CTOを務めています。

Beckの本は(あくまで個人的には)読んでても楽しくなかったりで、概念になじむまでに結構時間がかかりました。年齢のせいじゃないと信じたいです。逆にTDDやCIを当たり前のものとして身に付けることができれば、今後のソフトウェアエンジニア人生がきっと楽しいものになることは多分間違いないと思いますので、ぜひお試しください。

転ばぬ先の杖として

体系的に学ぶ安全なWebアプリケーションの作り方――脆弱性が生まれる原理と対策の実践

徳丸浩(著)、ソフトバンククリエイティブ、2011年、3,360円

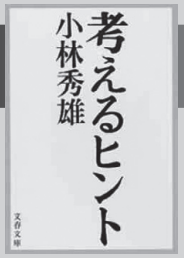
2冊目、セキュリティの本はこちらです。最近ではスマートデバイスの普及により、ネイティブアプリケーションやJavaScript、いわゆるクライアントサイドでの開発が増えてきたので、セキュリティ上で気を付けなければいけないポイントは増える一方ですが、サーバサイドアプリケーションに関するセキュリティに関しての最初の1冊としては間違いなくお勧めです。

そして言うまでもなくセキュリティは大事です。が、どうしてもソフトウェアを開発していると、つい楽しくて我を忘れたり、スケジュールに追われたり、とりあえず早く動かしたかったり、などでセキュリティに関する考慮が二の次になったり、あるいはそもそも注意を怠ってしまうことがあるかと思います、少なくとも僕はあります。もちろん、人は必ずミスをするので、機械でそれを防ぐ、というのも欠かせない概念ですが、セキュリティに関してもテストやCIと同じく、最初からこういったことを気にする癖を付けておく、当然のこととして扱うようにしておく、というのが非常に大事かと思います。逆に、最初からこのあたりがおさえられていると、きっと後で楽をできる場面がたくさんでてくるかと思いますし、インターネット利用時間、人口が増え続ける昨今、セキュリティの持つ重みは増える一方なので、詳しく学んでおいて損はないと思います。SD

＋α　あなたにとっての大切な1冊を見つけていますか

考えるヒント

小林秀雄(著)、文藝春秋、2004年、590円



以上、いわゆる技術書を紹介いたしましたが、ソフトウェアエンジニアとして生きていくにしてもそうでないにしても、技術書以外の書籍が役に立つことも多いと思いますので、まったく別ジャンルから批評家小林秀雄の本、その中でもたぶん一番有名で読みやすいものを挙げてみました。

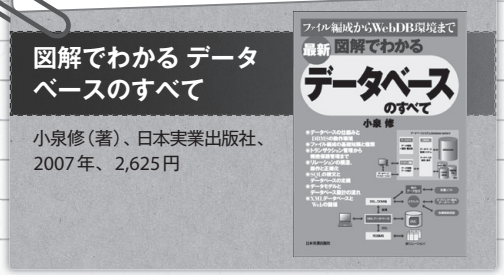
これは僕の言葉ではありませんが、良い本は人生を生きていくうえでの錨になるのだそうです。これから仕事で、私生活で、大きな変化が起こっていくこともあると思うのですが、そうしたときに自分がどうすべきか、しっかりと考える礎となる力と知識を与えてくれる本を探して、人生の支えとしていただければと思います――というほど僕も年齢を重ねているわけではないのですが！

6 10年闘えるエンジニアになるために

Open Source Software，Linux，Data Base

「ジオンはあと10年は闘える」と豪語したのは、マ・クベ大佐の名言ですが(わからない人は『機動戦士ガンダム('79)』を勉強しましょう♪)、新人として社会人になった人が10年後に闘える人財となるには、どうするべきか？　筆者は、30年近くIT業界にいますが、伸び悩んでいたり、劣悪な環境にいたりする技術者をよく見ます。一方で、楽しそうに好きなことをやって良い環境にいる技術者もいます。この差はなぜ起きるのでしょうか。「できる人」は、周囲とも自分自身とも、必要に応じ闘うからです。そのためには、十分な知識と情報が必要です。もちろん書籍から得る情報収集でも大きく差が出ます。

データベース管理システムを深く知るために読む本



筆者は、Linux版Oralcleの立ち上げを立案し、その後ミラクル・リナックス社の設立企画やNPO法人LPI-Japanを運営し、とくにマーケティングを10年以上やってきました。OSS-DB/Linuxの企業向け浸透も仕事の1つです。ということで、まずはデータベース管理システム(DBMS)の本を勧めます。技術者になるのなら、即戦力として商用やOSSデータベースを使いこなすことが必須です。しかし、本当の基本を知らずに、特定の製品知識だけに特化していると、10年後には高確率で残念な技術者なります。また、そのとき30代前半になっていますが、職務経歴とともに取得資格も評価の対象になるでしょう。

日本初のOracle書籍出版をはじめとして筆者は何冊もの書籍に携わりました。90年代初頭と比べて膨大に増え、資格取得対策本もたくさんあります。情報処理技術者試験の「データベーススペシャリスト」もありますが、本当に「実践」を求めるならば資格試験としては、商用やOSS固有の製品に特化したほうが良いと筆者は考えます。ORACLE MASTERやOSS-DB認定試験の勉強を通して、製品固有の基本を身につけることが重要という認識です。しかしながら、それと並行して基礎となるDBMS全体を俯瞰した学習が必要です。RDBMSというよりも、DBMS全体観も重要です。その目的を満たすのは、ファイル編成の基礎からの話題も記載している書籍が良いでしょう。これまでに筆者が読んだオススメの入門書が前掲のものです。DBMSが単なるファイルからなぜ必要なのかをきちんと解説しています。さらに、DBMSの基盤も実はファイル編成であることを説明しています。『OLTPシステム――オンライントランザクション処理(J・グレイ著)』なども古典的ではありますが、オススメの1つです。

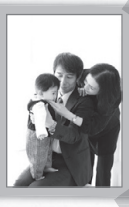
どんな分野でも当てはまりますが、特定分野の情報を取得するために、入門書を読むのであれば、書店や図書館で3冊ぐらいを手に入れ、さらっと速読して、その中で現状に一番合っている1冊をさらに読むのがオススメです。DBMS全体観で3冊、自分のかかわるRDBMS製品で3冊、SQL関連で3冊ぐらいを読むと、一挙にDBMS技術者として良い人財に育っていくと期待できますね♪

教科書どおりじゃない！　先達直伝「効く」ブックガイド Part 1

PROFILE

池田秀一（いけだひでかず）　Blog：http://blog.marketing.itmedia.co.jp/redcommet/

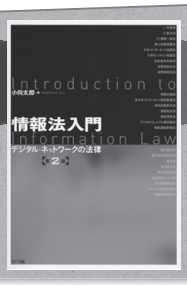
1965年生、汎用機上でのCOBOL＋NDBを手始めに、UNIXでのオープンシステムでC＋RDBMSを経験。その後、日本オラクル社にてサポート技術職から、プロダクト・マーケティングに職種を変え、競合対策などを通して「闘う！　マーケッター」と呼ばれる。



IT関連で犯罪者にならないために

情報法入門【第2版】
――デジタル・ネットワークの法律

小向太郎（著）、エヌティティ出版、2008年、2,940円



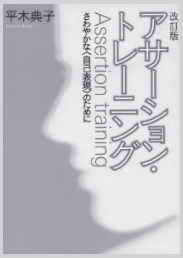
IT業界の技術者として、ソフトウェアやサービスを創る立場にいるのであれば、「著作権」や「ライセンス（使用権許諾）」などについての最低限の知識が必要です。OSSもライセンスを知らずに使うと、無意識に不当行為となっている可能性もあります。また、受託開発にかかわる立場であれば、さらに「契約」の知識も必要です。たとえば、法律における「強行規定」と「任意規定」を知らないと契約書が有効なのかも判断できません。

IT業界に関する法律に関しての網羅的な書籍として、この本がオススメです。現状の全体観を把握して、そこから著作権などの個々の分野に対しての知識を深くしていくのが早道です。著作権については平成24年に、著作権法が改正されていますので、それに対応した入門書である『著作権のことならこの1冊（飯野たから著）』を入手するといいでしょう。ちなみに、Kindleを持っているのであれば『著作権法』が、現時点では無料で提供されています。条文を読む趣味がある人には良いかもしれません。読んでみたのですが、即座に気力がなくなりました（苦笑）。上記とは別に、IT業界にも蔓延（はびこ）るブラック企業にかかわらないため、もしくは起業した場合にブラック経営者にならないためにも、労働三法の基本は知っておいたほうが良いでしょう。劣悪な労働条件で働かないためにも、社会人としてのオススメです。自分の身を守るためにも、労働法関連の入門書を1冊は必ず読んでおきましょう。SD

＋α　アサーションをご存じですか？

改訂版　アサーション・トレーニング
―さわやかな〈自己表現〉のために

平木典子（著）、日本・精神技術研究所／金子書房、2009年、1,575円



コミュニケーション能力の課題として「どう他者と接するか？」ということは、技術よりも重要です。とくに現代の日本にある問題として、議論と感情を分けられない人が多いとことがあります。たとえ正論でも、言い方を気をつけないと、恨まれたり、正しく評価されなかったりする可能性が少なくありません。それを避けるために「アサーション」という技術があります。

第一人者の著書です。職場だけでなく、家庭などの私生活においても、周囲の方と適切にコミュニケーションを取るのに、アサーションは非常に有用な技術ですので、機会があれば学習されることを強く勧めます。とくにプロジェクトマネージャなどの立場で、外部との交渉、部下への指示を行う立場には強くお勧めします。可能ならば、セミナーを受講するのも良いでしょう。IT業界でも共同作業は多いですから、人間の心理、心理学関連の知識も得ておくことは、知識の幅が広がるので良いと思います。同じ技術力であれば、より積極的で、よりコミュニケーションが上手な人が、評価されていくのは自明でしょう。

ITの技術を身につけるのと同時並行でコミュニケーション能力も高めて、外部に積極的にかかわることが、より高いレベルの技術者に近づく近道です。OSSコミュニティや外部の勉強会などで、多くの他の技術者と、積極的にかかわっていくことも、生き方としてオススメします。

Date. 2013 . 5 . 18

7 OSSの技術と精神を学ぶには

Open Source Software，Linux

レッドハットのエンジニアは中途採用が多く、他の企業のように新人教育を毎年行っているわけではありません。しかしパートナー企業やエンドユーザの「OSSビジネスを始めたい」、あるいは「OSSを活用してコストを削減したい」といった要望は年々増加しており、それに伴い「OSSの新人」も増えていると言えます。こういった「新人」の方は、技術面とOSSの精神面、大げさに言うと哲学の2つを理解しなければなりません。それぞれに関する良著を紹介します。

Linuxの広辞苑をご存じですか

詳解 Linuxカーネル 第3版

Daniel P.Bovet、Marco Cesati（著）、高橋浩和（監修）、杉田由美子、清水正明、高杉昌督、平松雅巳、安井隆宏（訳）、オライリージャパン、2007年、6,930円

OSSに関する書籍を技術面から見ると、プログラミング言語、ソフトウェアの設計や構造の解説、設定に関するもの、ツールの使い方といったように、ものすごい数が存在します。筆者の職場にもそれらのうち代表的な書籍の相当数を蔵書としていますが、1人のエンジニアがすべてを読破し理解することは不可能ですし、OSSにおける開発速度を考慮すると、ある時点の情報を網羅的に記憶したとしてもあっという間に陳腐化することは明らかです。しかし一方で、分散して細切れになってしまっているネット上の情報に対して、体系化され、かつ著者に加えて査読者、編集者のチェックを経て出版される書籍の情報に一定以上の価値があることは間違いありません。

そこでOSS、とくにその代名詞ともなっているLinuxカーネルの書籍として本書を紹介しないわけにはいきません。この本を手に取った方はその重さに圧倒され「これを読むのか……」とげんなりしてしまうかもしれませんが、この本はLinuxにおける「広辞苑」です。もちろん広い世の中には広辞苑を書籍として読む方がいるので、この本を最初から読んでもいっこうにかまいません。ですが、自身が携わる問題、たとえばファイルシステムやメモリの管理、ネットワークの性能といったあるカテゴリに相当する章から読み始めても、あるいは全体をサラッと通読しておいて、それらの問題が起きたときに「あの辺に書いてあったな」というきっかけで読むのでもかまわないのです。ちなみに私がオススメする順番は、第3章の「プロセス」を最初に読む、つまり「ユーザランド」から掘り下げる方法です。1つ前の第2章の「メモリアドレッシング」も重要なのですが、最初に読むには難易度が高い内容だと私は考えているからです。

Linuxにかかわるエンジニアにとって「お守り」とも呼べる大著であることは保証しますので、ぜひ、1冊を手許に置きましょう。

さて、次にOSSの哲学を理解するという点からお勧めする書籍としては、エリック・レイモンドのOSS4部作に含まれる『伽藍とバザール』や、Linuxカーネルの開発者であるリーナス・トーバルズの『それがぼくには楽しかったから』ということになるのが一般的かもしれません。ですが、少し異なる視点から書かれた『「みんなの意見」は案外正しい』を筆者はお勧めします。

PROFILE

藤田稜（ふじたりょう）　Twitter：@rioriost

塾講師を3年、SIerを通算6年経験したあと、レッドハットに8年在籍。おもにパートナーを担当するSolution Architect、いわゆるプリセールスエンジニアとして、Red Hat Enterprise Linuxを中心としたオープンソースソフトウェア（OSS）の普及に努める毎日を送る。

OSSの神髄とは

「みんなの意見」は案外正しい

ジェームズ・スロウィッキー（著）、小高尚子（訳）、角川書店（角川グループパブリッシング）、2009年、780円

筆者自身、OSSビジネスを長年続けていると疑問を感じることがないわけではありません。OSSはソフトウェアの開発手法として本当に正しいのだろうか、クローズソースのソフトウェアと比較してOSSの品質は果たして本当に高いのだろうか、といった質問に対して常に自信を持って回答するためには、ソースコードを読む以外に「ソフトウェア業界の外」の視点から書かれた本書が非常に役立ちます。本書で紹介されている事例だけでなく世の中には同様の「みんなの意見が案外正しい」ことが多いことに気づかされ、インターネットの普及に伴って同様の事例がますます増えていることに思い当たり、あらためて自信を持ってOSSの普及に携わることができます。書中に挙げられているエピソードの中には客先などで「ネタ」として用いることができるものもあります。たとえば瓶に入ったボールの数を当てるゲームについては「集合知」の何たるかを説明するのに格好のネタと言えます。一方で「集合知」を正しく伝えるのは難しいことでもあるのですが……。

これからOSSに関わるエンジニアだけでなく、筆者と同様の疑問を感じることのあるエンジニアにもお勧めできる良書です。SD

＋α　1バイトの大きさの違いを知る人だけがわかる

神の火

高村薫（著）、新潮社、1995年、620円

筆者自身はいわゆる活字中毒というやつです。書籍は何でも、明治期から現代作家まで、ラノベから政治・哲学まで読むタチで、とくに緻密に構成された物語には非常に強く惹かれるものがあります。ソフトウェアやITに携わるエンジニアは1バイトの間違いが恐ろしい結果を導くことを経験的に知っていると思いますが、現代作家というくくりの中で僭越ながら仕事の完成度の高さという評価をさせてもらうと、高村薫は群を抜いていると思います。高村作品のいずれも、連載、ハードカバー、文庫で加筆・修正がされていることが多く、エンディングが異なっていたり、人物描写がより深みのあるものに変更されていたり、ある意味で偏執的とも言える作品に対する執念には頭が下がります。

もちろん、小説家というものは長期にわたる取材を経て物語に現実世界を投射する技術に長けているものだと思いますし、そうあるべきでしょう。この観点でも高村作品は「プロとしての矜持」を強く感じさせてくれます。

『神の火』は技術的な描写が含まれることからお勧めとしましたが、『レディ・ジョーカー』や『照柿』も自信を持ってお勧めできますので、ぜひ一読してみてください。

Date. 2013 . 5 . 18

8 技術の根底にある考え方を学んでみませんか？

UNIX，Open Source Software，Linux

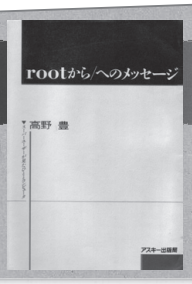
この春、新しい生活をスタートさせた皆さん、おめでとうございます。そして本誌の読者には、IT業界を選んで入ってこられた方も多いでしょう。ようこそITの世界へ。

筆者は企業に所属していないので、春になると新人がやってくるという感覚があまりないのですが、昔は大企業に務めていたので、毎年入ってくる新人たちにいろいろと教えていたものです。そんなころを思い出しながら、これからIT業界でやっていく皆さんにお勧めの書籍、とくに筆者のバックボーンであるUNIX（ここでいうUNIXにはLinuxやMacのOS Xなども含まれます）を深く理解するために読んでほしい書籍を紹介します。

UNIXシステム管理の考え方を知る

rootから/へのメッセージ

高野豊（著）、アスキー、1991年、1,631円



これから筆者が推薦する書籍すべてに共通することですが、直接的なテクニックよりも、その根底にある考え方を知ってほしいという想いがあり、それに触れることができる本を紹介しています。

さてこの本ですが、「ルートからルートへのメッセージ」と読みます。著者の高野さんは筆者にとっては日本UNIXユーザ会の大先輩であり、筆者が皆さんのようなIT業界の新人だったころに、多くのことを教えてくれた人でもあります。本書の構成は基本的に、各章の前半は著者の体験や趣味に関する知見が示され、後半はそこから連想されるUNIXのシステム管理に関する考えが提示されるのですが、その前半と後半の関連性が絶妙で、それが本書を読み物としてとても興味深いものにしています。

本書が刊行されたのは1991年、さらにその元となる記事が『UNIX MAGAZINE』という雑誌に連載されていたのが1986年から1988年と、時期的にはかなり古いものです。当然ながら技術的な記述には当時の名残があり、現在の状況にはそぐわない部分もあると思います。にもかかわらずこの本が今でも価値を持っているのは、UNIXのスーパーユーザであるroot、それはUNIXのシステム管理を務める人のことをも指しますが、rootが何をしなければならないか、その心構えや考えるべきことが記されているからだと思います。この本は発売から20年を経過した現在でもまだ販売されていて、ちゃんと書店の本棚にも並んでいます。もし見かけたら、ぜひ手にとってみてください。

プログラムの書き方を知る

リーダブルコード—より良いコードを書くためのシンプルで実践的なテクニック

ダスティン・ボズウェル、トレバー・フォシェ（著）、角征典（訳）、オライリージャパン、2012年、2,520円

PROFILE

法林浩之（ほうりんひろゆき）

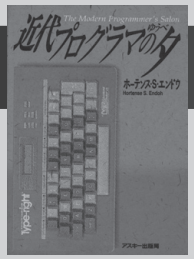
日本UNIXユーザ会幹事（元会長）。大阪大学からソニー、インターネット総合研究所を経てフリーランスエンジニアに。並行してLightweight Languageイベント、TechLIONなど多彩なITイベントの企画・運営を手がける。2012年には日本OSS貢献者賞を受賞。

UNIXと対峙するとき、1つはコマンドを打つユーザとして、もしくは前述したようにシステム管理者として相手をするケースが多いと思いますが、もう1つの使い方として、UNIXをプログラミング環境として利用し、そこでプログラムを書くという一面が挙げられます。それはWebサービスなどの巨大なプログラムかもしれませんし、システム管理の一環として書く小さなプログラムかもしれませんが、いずれにせよある程度プログラムを書くようになったら、各言語のコマンドや関数を知るだけでなく、他人が読んでも理解しやすいプログラムの書き方というものを勉強してほしいと思うのです。

筆者がコンピュータ専攻の学生だったころに教わったのは『プログラム書法 第2版』（ブライアン・カーニハン、P.J.プローガー（著）、木村泉（訳）、共立出版、1982年、3,360円）という本で、これも今でも売っているので見かけたら読んでほしいのですが、いかんせん30年前ぐらいの本で、書籍の中で使われている言語もFORTRANやPL/1が中心なので、もう少し現代的な本がいいだろうと思い、この本を紹介します。本書もプログラムを書くときに気を付けるべきことが多々記されていますが、刊行されたのが2012年ということでサンプルコードもPythonやC++、JavaScriptなどが中心になっており、読者の皆さんにも身近に感じてもらえるでしょう。また、文章も軽快なタッチで読みやすく、随所に描かれた挿絵のおかげもあって、楽しく読み進められると思います。SD

＋α　ハッカーのスタイルとは

近代プログラマの夕（ゆうべ）



ホーテンス・S・エンドウ（著）、アスキー、1991年、入手困難／絶版

もう一冊はとても悩みました。UNIXに限らずITの世界でやっていく人はぜひIT系のコミュニティ活動に参加してほしいので、そういう意味では『アート・オブ・コミュニティ』などもお勧めしたいのですが、もっと広くコンピュータやハッカー文化的なものに親しんでもらいたいという想いから、この『近代プログラマの夕』を紹介します。

著者の「ホーテンス・S・エンドウ」は、現在は角川アスキー総合研究所の所長であり、かつては月刊アスキーの編集長を務めていた遠藤諭さんのペンネームで、本書も月刊アスキーに連載されていたコラムをまとめたものです。内容としては、ハッカー（ここでは熱心なプログラマの意）の生態、さまざまなプログラムに関する蘊蓄（うんちく）、そしてプログラマたちが好むあれこれ（たとえばゲームや辛い食べ物など）についての考察が、文筆のプロならではの楽しく読みやすい文章で描かれています。

読者の皆さんがIT業界とどのような形でかかわっていくのかは筆者には知る由もありませんが、ハッカーと呼ばれる人種がどのような嗜好を持ち、何を考えているのかを知るという意味で、本書はとても参考になると思います。ちなみに本書には『近代プログラマの夕2』という続編もあります。残念ながらどちらも現在は絶版になっていますが、古書店やAmazonなどで見つけたら、ぜひ入手してみてください。

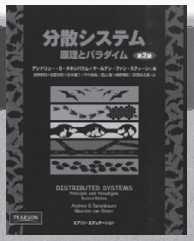
9 歴史を学ぼう

Information Technology, Science

ソフトウェアの仕様や詳細な挙動を理解するうえで、筆者が参考にしているものは、そのソフトウェアの歴史です。ここで歴史とは、過去から現在へのソフトウェアの設計の設計思想や実装の変化のことです。とくに分散システムを構築するソフトウェアは、複数の要素から構成されており、それぞれの要素技術には長い歴史があります。これらの歴史を知ることにより、分散システムを理解しやすくなると考えています。また新しい分散システムが登場しても、既存のものとの差分を調査することで、そのシステムの概要を理解できるようになり、学習コストも下がります。今回筆者が紹介する2冊は、新人の方々が分散システムの歴史を知るための導入として良い書籍だと筆者は考えています。

分散システムの歴史を知りたいときに読む本

分散システム 第二版



アンドリュー・S・タネンバウム、マールテン・ファン・スティーン(著)、水野忠則、佐藤文明、鈴木健二、竹中友哉、西山智、峰野博史、宮西洋太郎(訳)、ピアソン・エデュケーション、2009年、7,980円

分散システムを学ぶうえでは必読とも言えるタネンバウム先生の1冊です。大学のときにこの本の洋書と筆者は出会いました。この本の特徴は分散システムの要素技術を広く網羅していることです。新人の方々が各要素技術の歴史を知る良いきっかけになるでしょう。分散システムを理解するうえで重要となる“同期”、“一貫性”、“複製”、“耐故障性”などのスタンダードなトピックが各章に並び、ざっと読んだだけでもある程度それらの体系立った知識を得ることができます。自分がとくに興味をもっているトピックから選んで読んでもかまいません。

さらに詳しく知りたい方々は、各章のところどころに散りばめられている参考文献をひとつひとつていねいに調べてみてください。簡単な作業とは言えないかもしれませんが、その分野における歴史を知ることができるでしょう。また、今まで、分散システムの研究および開発に取り組んできた方々の試行錯誤がわかるかもしれません。さらに、分散システムのこれらのトピックは、現在でも着々と築かれ、成長しています。そのような最先端の要素技術や最新の分散システムの動向を理解するのにも歴史は役立ちます。

筆者は現在分散システムの開発や運用に携わっていますが、今でもこの本をしばしば読み返します。分散システムのアーキテクチャや設計に携わったときには、自分が求める分散システムを実現するために、要素技術をどのように組み合わせるかの決定をする参考にしました。個々の要素技術のいくつかは、長い間に揉まれ、成熟しています。それらの要素を組み合わせることでも、分散システムを実装できるのです。また、既存の分散システムの改良や運用に携わったときには、そのシステムの振る舞い(たとえば、どのようなネットワーク通信をしているのか)を理解したり、不具合を修正するための参考にしました。

本の角で殴られたら、分厚いので大けがをしそうな1冊(かつ少し高価)ではありますが、分散システムを知るためのエッセンスが凝縮されています。読めば、けっして高い買い物をしたとは思わないことでしょう。

PROFILE

西澤無我（にしざわむが）　Twitter：@muga_nishizawa

2008年東京工業大学大学院情報理工学研究科数理・計算科学専攻博士課程修了。博士（理学）。楽天（株）楽天技術研究所にて分散システムの研究開発を経て、2012年1月よりTreasure Dataに勤務。同社のサービスプラットフォームの開発を行いつつ、MessagePackやFluentdなどのOSSコミュニティへコミッターとして参加している。言語処理系や分散コンピューティングなどの基盤ソフトウェアの研究および開発に興味を持っている。

ネットワークの歴史を知りたいときに読む本

コンピュータネットワーク　第4版

アンドリュー・S・タネンバウム（著）、水野忠則、相田仁、東野輝夫、太田賢、西垣正勝（訳）、日経BP、2003年、8,190円

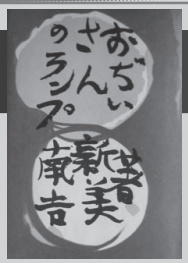
実は、この本もタネンバウム先生の1冊です。同じ著者の書物を同時にお勧めしたくはなかったのですが、歴史を知るうえでどうしても必要だと思い並べました。「分散システム」と同様、大学のときにこの本の洋書に会いました。

分散システムの挙動を理解するためには、ネットワークの知識が欠かせません。この本はネットワークの概要を深く解説しています。“物理層”、“データリンク層”、“ネットワーク層”とTCP/IPモデルの各層を下位層から順々に章立てし解説しており、それらの説明は具体例も多く非常にわかりやすいです。とくにネットワークプログラミングの経験をもっている方々には、より厚みのある知識を身に付けることができるでしょう。先に推薦した「分散システム」と同様に参考文献も充実しているため、その歴史を知る手助けにもなります。

余談ですが、この本を最初に読んだころ、筆者はまったくと言っていいほど物理層の知識を持っていませんでした。本書籍の“物理層”の章にはデータ伝送（信号の振る舞い）のモデル化とその解析について取り上げられていたり、データの伝送媒体である同軸ケーブル・光ファイバーケーブルの特徴が取り上げられ、それらが比較されていたりと、読み進めれば進めただけ、目から鱗が落ちてきたのを覚えています。『分散システム』と同様にお勧めの1冊です（同じように分厚く少し高価です）。SD

＋α　最近、童話を読んだのはいつですか？

おぢいさんのランプ



新美南吉（著）、ほるぷ出版、1971年、入手困難／絶版

“ごん狐”などで有名な新美南吉先生の童話の1つです。もしかしたら、多くの方々が読んだことのあるものかもしれません。筆者がこの童話を初めて読んだのは、小学校の授業のときでした。当時の小学校の先生が授業中に音読していたのですが、感動のあまり、その先生が途中で泣き出してしまったということがありました。

そのようなエピソードもあり、この童話の内容を筆者は鮮明に覚えていたわけですが、その内容はとても考えさせるものです。端的に内容を説明しますと、“時代が変わり自分の仕事がなくなろうとしていることに気づいた主人公は、自分の仕事を必要のないものにした人や物を恨まず（最初は恨んでしまっていますが最後に改心します）、その仕事から身を引き、別の新しい仕事をする決意をする”という話です。それがとてもリアルに描写されており、主人公の強い生き方に胸を打たれ、共感します。短い童話ですので、ぜひ手にとって眺めてみてはいかがでしょうか。

Date. 2013 . 5 . 18

10 サバイバルするための武器＝書籍

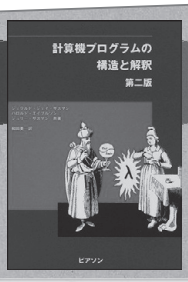
Information Technology, Science

「新卒お勧め本」というテーマですが、あえて新卒に限定せずに、新興国の攻勢や「機械との競争」にさらされる職業人として、不確実な競争環境において生き残る技術者になるための本をオススメする、という観点から選書してみました。

功徳を積んで技術を身に付ける本

計算機プログラムの構造と解釈

ジェラルド・ジェイ サスマン、ジュリー・サスマン、ハロルド・エイブルソン（著）、和田英一（訳）、ピアソン・エデュケーション、2000年、4,830円



プロのエンジニアとしては、やはり計算機のことがよくわかってなければいけません。そのためにはプログラム言語にきちんと取り組む必要があります。

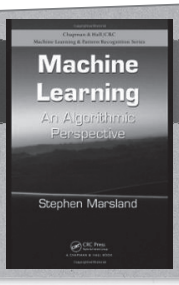
本書はソフトウェアエンジニアにとってのまさに「古典」と呼ぶべき本です。SICPと短く省略して呼ばれています。はっきり言って形式は古いですし、本家MITでもあまりに「古」典なので講義が廃止されてしまったほどですし、面白いと感じるためにはかなり大きな山を乗り越えないといけません。この本の真骨頂は本文より問題にあります。何しろ問題の答えを次に使っているので、どうにかして解かないと次が読めません。つまり問題をひたすら修行僧のように解いて功徳を積んでいくと、そのうち悟りが開けるという狂った構成になってます。独習本というより、MITの講義と演習をそのまま本にしたものを今オススメする理由の一番は、現在においてはいろいろなリソースが充実し比較的アクセスしやすい、ということです。サスマン師の実際の講義ビデオを見て「直接」教わることができますし、勉強会も頻繁に開かれています。

もしもSICPに抵抗ある、地方に住んでいて独りぼっちだし、という方には別のオススメ本があります。『素数夜曲』吉田武（著）です。これも狂った本の1つです。前半330ページの整数論初歩の本文より、後半520ページのSchemaで数論計算環境を作る付録のほうが断然面白いです。この付録を頭から「写経」して功徳を積めば、そのうち悟りが開けるという構成になっています。何より、この本で自作したScheme上の整数論計算環境を用いて素数定理を調べたり四元数で遊んだりできる面白さがあります。著者独特の言葉遣いも力強く、技術書のビルディングスロマンといいますか、教養小説のような雰囲気を放っています。

コーディング以外の武器を装備する本

Machine Learning - An Algorithmic Perspective

Stephen Marsland、Chapman and Hall/CRC、2009年、7,403円（為替レートにより変更あり）



ソフトウェアエンジニアという職業について、今は曲がり角にあるように筆者は感じています。ソフトウェアエンジニアといえば、誰かのアイデアを計算機で実現できるように、ソフトウェアの部品をつなぐ便利屋というか「配管工」のイメージですね。

PROFILE

柏野雄太（かしのゆうた）　Twitter：@yutakashino

バクフー株式会社代表取締役。Zopeの開発に関わっていました。最近は非同期Webアプリケーションシステムの開発、データ解析・統計解析の請負業務、リアルタイムデータ解析などをしています。代表的な書籍・翻訳書に『Pythonポケットリファレンス』、『Zope3 入門篇』、『Zope3 発展篇』

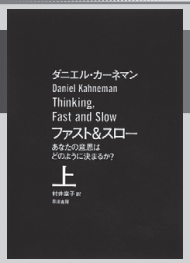
しかし、このようなイメージの職能だけでやっていくには、そろそろ難しくなってきたのではないかと思っています。それは、OSSが成熟してきて「配管」作業なら誰でも簡単にできるようになったからです。将来に渡ってエンジニアとして食っていくには、単に「配管」技術だけでない、コーディング以外のスキルが必要になってくると思います。そういったスキルの大きなトレンドが、大きなデータから何か意味のある情報を引き出す技術だと思います。ここではコーディング以外の武器を装備するという文脈から、手を動かしながら学ぶことができる機械学習の本について紹介します。

機械学習の初歩の原理的な側面を手を動かしながら学ぶには、本書がかなりよく書けていて入門的教科書です。アルゴリズムの解説も短くわかりやすいですし、一通りこの分野をさらうことができます。ただ、実際に機械学習を自分で実装するときは、車輪の再発明を避けて、すでに定評あるパッケージを利用することが多いでしょう。よく利用されるのが定評があるscikit-learnです。それを利用したチュートリアル[注1]が機械学習の入門「書」としてよくまとまっています。これにはビデオチュートリアル[注2]もあります。これらの本やサイトを利用すれば、短時間で機械学習の初歩を学ぶことがことができます。この分野は広大で専門分岐が進んでいるので、エキスパートとしてかかわるのはたいへんですが、「コーディング以外の武器」としてエンジニアが身につける知識として強くお勧めできます。今は学習リソースで溢れていて、学習障壁がとても下がっていますので、とてもコストパフォーマンスの高いスキルになると思います。SD

注1）http://scikit-learn.github.com/scikit-learn-tutorial/
注2）http://www.youtube.com/watch?v=cHZONQ2-x7I

+α 最近の私のとっておき本

ファスト&スロー



ダニエル・カーネマン（著）、村井章子（訳）、早川書房、2012年、2,205円

プロの職業人として生きるためには自分の失敗から学ばなければいけません。人間には誰にでも偏見やバイアスがありますし、物を筋立てて考えるのはたいへんですから端折って考える傾向があります。そのバイアスとヒューリスティクス研究の権威が、意思決定のしくみとエラーを避けるノウハウをバランスよく著述しています。意思決定がどうやって行われ、どうすればよりよく意思決定をできるかを学ぶことができます。

脳の働きを、直感的に即座に動作するシステム1と筋道を立てて努力しながら考えるシステム2に分け、それらの2つのシステムが複雑に相互作用しながら意思決定が行われることを、著者自身の研究成果や自身が犯した失敗を具体的に例示しながら解説してくれます。とくにシステム1が引き起こす間違いや誤認識についての記述に力が注がれていて、この本をキチンと読めば、（システム1の本源的な性質があるために）自分の犯すエラーを減らすことはできないかもしれませんが、他人のエラーを把握できるようになると思います。

この本において、より良い意思決定をするためのノウハウとして個人的に衝撃を受けたのは、ゲーリー・クラインの"premortem meeting"です。これは、プロジェクトの開始前にそのプロジェクトが大失敗したと仮定して「死亡原因」を列挙し、そのことによりプロジェクトを進行する上でのシステム1が引き起こす楽観主義バイアスを牽制するしくみです。このように、この本は職業人だけでなく、不確実な社会を生きる市民の基礎教養としても必要な本だと思います。

11 インフラエンジニアの技術を学ぶには

Networking，Operating System，Infrastructure

ITインフラ周りの業務を担当される方への本を紹介します。ITインフラと一言にいっても、具体的に何を示すのかは会社や人によってさまざまというのが実状だと思います。しかも範囲が広いので、最初に何から勉強して良いかわからないという方も多いのではないでしょうか。そうした方への勉強の第一歩となれば幸いです。

ネットワークとサーバから1冊ずつ紹介しますが、最初は両方読んである程度の知識を得ておくといいでしょう。また、理解を深めるために、実践してみると良いことも紹介しますので、参考にしてください。

まずは定番のネットワークの教科書

マスタリングTCP/IP 入門編 第5版

竹下隆史、村山公保、荒井透、苅田幸雄(著)、オーム社、2012年、2,310円

初版が1994年という定番本です。皆さんのオフィスの本棚に置いてあるかもしれません。2012年に第5版が出ているので内容も古くなく、むしろ改訂が繰り返されて、説明がよりわかりやすくなっています。

この本の良い点は、プロトコルの説明から始まり、OSI参照モデルの各レイヤで必要な知識がひとつひとつ順序立てて説明されていることです。複数の本を読むよりも、まずはこの本に書いてあることをしっかり理解すると良いでしょう。あとでわからないことが出てきたときに、参照する際にも使えます。各項目がたくさんの図を入れてていねいに説明されているので、初心者の方にも理解がしやすいでしょう。まずはこの本を読んで、OSI参照モデルから、Ethernet、ARP、IP、ルーティング、TCP/UDPなどを押さえます。レイヤの下方から理解を積み上げることが大事です。

本を読むと同時に、できればルータやL3スイッチなどを数台用意して、サーバやクライアントPCを両端に接続し、静的ルーティングやOSPFなどの動的ルーティングでIP到達するところまで、自分で設定してみると良いでしょう。読むだけでは理解があいまいだったところも、実践することでずっと理解が深まります。さらに、Wiresharkなどのパケットキャプチャを使って実際にパケットの中身を見ると良いでしょう。各レイヤのパケットを追って、MACアドレスはどうやって解決されているのか、DNS名前問い合わせやhttp getが発生したときはどういうパケットが送受されているのかなど、自分で確認するとよくわかるようになります。

サーバを理解するための定番本

はじめてのCentOS6 Linuxサーバ構築編

テージーネット(著)、秀和システム、2011年、3,150円

サーバを触ったことがなく何から勉強すれば良いのかわからない——そういった方に本書を勧めます。最初はどれかUNIXでいえば特定のディストリビューションOSについて書かれている

PROFILE

山下秀登（やましたひでと）　http://www.idcf.jp/

㈱IDCフロンティア ビジネス推進本部 新基盤開発部。CISSP。1997年に国際電話会社へ就職後、国際伝送路から始まり、ネットワーク、サーバ、セキュリティと、さまざまなインフラシステムの設計・構築・運用に携わる。役割も技術担当からSE、プロジェクトリーダーなどを経験し、今はクラウドのインフラ管理に従事。

本が良いと思います。本稿ではCentOSの本を紹介しますが、職場でよく使うOSのディストリビューションに合わせるのが良いでしょう。その際はOSインストールからWebサーバなどのアプリケーションを立ち上げるところまで、図を多くいれて説明している本が良いでしょう。

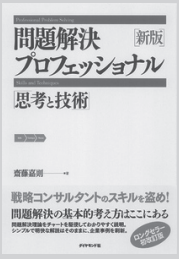
さて、本書の良い点は、そもそもサーバを初めて触る人を対象に書かれていることです。サーバ構築するために必要なものを、予備知識から重要な項目までていねいに説明されています。図や画面キャプチャ、コマンド出力例が多く取り入れられ、それらにていねいに注釈がされて直感的にわかりやすくできています。タイトルどおり、UNIXやLinux全般の一般論が書かれているわけではなく、CentOSを対象にサーバ構築を目的とした本なので、より具体的に学べます。

本を読むと同時に、実際に自分でOSのインストール作業からやってみると良いと思います。本書にはCentOS6のインストールCDが同封されているので、手軽にためすことができます。とりかかりにくいコマンドライン入力も、この本に沿って1つずつ順番に触ていけば必ず理解できるようになります。OSのインストール手順から、ファイルシステム、パッケージ管理などに一通り触れて勉強するのが良いと思います。そのあと、自分の業務に関連性の強いプログラム言語やアプリケーション、クラウド技術などに特化した本を選んで読んでいくと良いでしょう。先にOSを知っておかないと、のちのちになって、ちょっとしたことでつまづいて余分な時間を消費してしまいます。最初に勉強しておけば、こういう話があったなと、解決策に結び付くことは少なくありません。遠回りのようで、結果的には近道なのです。下調べ的に勉強しておくことはきっと役に立ちます。SD

＋α 自分の変化に気づく本

新版 問題解決プロフェッショナル ——思考と技術

齋藤嘉則（著）、ダイヤモンド社、2010年、2,447円



ビジネス書のロングセラーです。ITとは離れていますが、ロジックツリーやMECEといった考え方のフレームワークを知ることができます。と言ってもすぐに実践できるようになる類のものではないのですが、仕事の中で少しずつ使ってみると良いでしょう。長期的に仕事の効率向上や、適切な判断や行動へつながっていきます。そして時々、たとえば数年に一度程度、この本を読み返すと良いでしょう。自分の理解や経験の変化につれて新しい発見があります。

最後に少し気楽に読める本として『勝ち続ける意思力』（小学館101新書）という本を紹介します。著者はテレビゲームのプロ、世界チャンピオンの梅原大吾さんというユニークな方なのですが、皆さんの中にはご存じの方もいらっしゃるかもしれません。皆さんが従事するIT技術は言うまでもなく日々進歩します。今後継続して勉強し続けることが必要になるでしょう。ゲームという異分野ですが、勝つことではなく、勝ち続けることを主眼に置いて努力した著者のスタンスは、とても刺激になります。

Date. 2013 . 5 . 18

12 ネットワーク技術の大きな変革の中で

Networking，Operating System，Infrastructure

最近の若い方はインフラ技術への興味が薄れてきていると聞きます。しかし、現在のインターネットには解決しなければならない課題が山積していますし、クラウドを中心としたシステム構築の流れに相まって、OpenFlowやSDNといったこれまでとはまったく異なるアーキテクチャのネットワークも提唱されています。

まさに今、ネットワークエンジニアには大きな変革に対応できるスキルが求められています。変化の激しいネットワーク業界で生き残るためには基礎をしっかりとマスターする必要があります。今回紹介する書籍は、いずれも基礎を固めるために最適なものを選んでみました。

10年たっても変わらないネットワークの本質を理解する

コンピュータネットワーク　第4版

アンドリュー・S・タネンバウム（著）、水野忠則、相田仁、東野輝夫、太田賢、西垣正勝（訳）、日経BP、2003年、8,190円

実はこの本、筆者が高専生時代に受けたネットワークの授業で教科書として使用されたものです。当時、学生の身分としてはかなり高い（税込8,190円）書籍だったのですが、担当の先生に「10年使えるから」と諭されたのを懐かしく思い出されます。

あれから10年以上経ちましたが、ふと振り返ってみると、当時この本で学んだ基礎知識が今でも役に立っていることに気づきます。2013年現在の最新技術を学べるものではありませんが、筆者自身、これから先もずっと使える本であると確信しています。

本書の内容は、物理層に始まり、データリンク層、ネットワーク層、アプリケーション層までOSI参照モデルの順に説明が進みます。また、最後の章はネットワークセキュリティがテーマとなり、暗号化や認証方式、さらには攻撃者の手口まで解説されています。

普段、OSI参照モデルという言葉は聞くものの漠然としか理解できていない、という方でも、体系的にネットワークのレイヤを学ぶことができ、それぞれのレイヤが何を担っているのか、深く理解できることと思います。

一方、実際に動いているネットワークは、OSI参照モデルのようにきれいにレイヤが分かれているわけではありません。事実、インターネットで使用されているTCP/IPは、完全にOSI参照モデルに準拠しているわけではありません。また、バックボーンで使われるMPLSやIPv6/IPv4トンネルといった技術、あるいはGoogleが提唱しているSPDYのように、いわゆるレイヤをバイオレーションしているプロトコルは、現在世の中に多数存在します。

しかし、本来の理想的なレイヤの概念をマスターしておけば、今後このようなレイヤを跨ぐ新しいプロトコルが出現したとしても、迷子にならずに理解できるはずです。

価格もさることながら、ボリュームもかなりありますので（約850ページあります！）、時間のあるうちにじっくり読まれることをお勧めします。

PROFILE

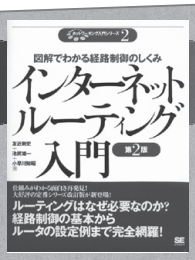
大久保修一（おおくぼしゅういち）　Twitter：@jq6xze_1

2003年4月、大学卒で現さくらインターネットに入社。おもにバックボーンネットワークの運用を担当する。2009年7月、研究所の発足と同時に異動となり、IPv4アドレス枯渇対策などをテーマに研究活動を行う。2011年3月以降、「さくらのクラウド」の立ち上げ時より、サービス開発にも携わっている。

実践的なルーティングの知識を身に付ける

インターネットルーティング入門

友近剛史、池尻雄一、小早川知昭（著）、翔泳社、2006年、入手困難／絶版



本書は、複数のルータをダイナミックルーティングで接続し、キャンパス、あるいはそれ以上の規模のネットワークを構築する機会のある方に、ぜひ読んでいただきたいと思います。前書と同様、こちらも筆者が学生時代に読んだ書籍の1つです。当時アルバイトをしていた地域ISPにて、BGPを使ったマルチホーム環境に移行する機会があったのですが、その際、本格的にルーティングの知識が必要になったことが、本書と出会うきっかけになりました。

内容のほうですが、はじめにIPアドレス、サブネット、ダイナミックルーティングの概要から入り、その後メジャーな3つのルーティングプロトコルであるOSPF、RIP、BGPについて解説が進みます。詳細なメッセージフォーマットや、Cisco社ルータの設定例まで説明されており、現場でのリファレンスとしても活用できます。

また、なんといっても著者の皆さんは超大規模ISPを支えていらっしゃる方々で、その経験に基づいた深い説明も得がたい内容となっています。たとえば、BGPはインターネット全体で用いられるルーティングプロトコルですが、プロトコルの知識だけでは不十分で、インターネットのしくみを理解していなければ使いこなすことはできません。本書では、ASやAS間の接続形態の概念についても述べられており、はじめてインターネットルーティングを担当することになった方に対しても不足のない内容となっています。

なお、本書を読み進める際、実機を用意して（もし用意できない場合は、QuaggaやVyattaなどのフリーのルーティングデーモンでも良いです）、設定を入れながら動作を確認するとより理解が深まることでしょう。SD

＋α　個人的にお勧め本

2分以内で仕事は決断しなさい

吉越浩一郎（著）、かんき出版、2005年、1,470円

過去、数多く読んだビジネス書の中でも最も印象に残っている1冊です。2005年発刊当時、著者の吉越さんはトリンプ・インターナショナル・ジャパンの社長を務められており、2006年に退任されるまで、19年連続増収増益という偉業を成し遂げられました。退任されたあとも、多くの講演活動を精力的にこなされ書籍の執筆も継続されています。

本書には「まず川に飛び込め」「すべての仕事にデッドラインを設ける」「仕事が宿題にならないうちに動き出せ」といったスピード重視の行動指針が数多く紹介されています。普段仕事をしていると、いつでもできることをついつい先延ばしにしたり、面倒なことを後回しにしてしまいがちですが、本書ではそのようなことを一番危険であると指摘しています。

実は、本稿のために8年ぶりに読み返してみたのですが、今でも読んでいてワクワクする内容です。今では若干入手しにくい本かもしれませんが、もし古書店などで見かけた際にはぜひ手にとってみてください。

Date. 2013 . 5 . 18

13 ネットワークエンジニア 座右の書とは

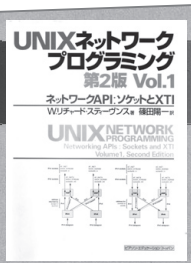
Networking，Operating System，Infrastructure

筆者はもともと情報科学や情報工学を専攻していたわけではなく、26才のときに突然機械業界からこのIT業界に飛び込みました。数年の社会人経験だけはありましたが、ITについては新卒同然の状態です。当時インターネットも今ほど発達しておらずコンテンツも充実していなかったので、技術情報を習得する手段は書籍と雑誌だけでした。そしてそれらの本を参考に自分で実験と検証を繰り返し技術を身に着けてきました。そのころ、座右の書として利用していた本の一部を紹介しましょう。

1冊でUNIXとネットワークをほぼ網羅できる本

UNIXネットワークプログラミング 第2版 Vol.1

W.リチャード.スティーヴンス（著）、篠田陽一（訳）、ピアソン・エデュケーション、1999年、8,400円



オープンシステムのSEになった筆者がまず理解しなければならないのはTCP/IPよりもUNIXというOSについてです。何冊も「UNIX」という単語を含む書籍を買い漁って読みました。しかし、それらの本はUNIXのアーキテクチャを図を用いて概念的に説明したり、シェルコマンドの使い方やシステムの設定ファイルの記述方法を解説するものが多く、UNIXカーネルの中身やOSの全体構造、システムコールの実装について詳細を示している本は少なかったのです。次第に筆者の仕事はUNIXシステム管理からTCP/IPネットワークの構築運用へとシフトしていくわけですが、その頃、片言のC言語でツールやスクリプトを書き始めた筆者が出会ったのが本書でした。

最初はソケットやRPCなどのプログラミング教本かと思って手にしたのですが、中身を読みはじめて愕然とします。そこにはネットワークプロトコルだけではなく、UNIXのアーキテクチャ、ファイルシステム操作、プロセス間通信など、重要なUNIXの特徴的技術についてそれぞれそのしくみと使い方を短いC言語サンプルコードを用いて解説されていました。それまで概念図と文章だけでUNIXやネットワーク通信のしくみを解説している本が多く、なかなか実体をイメージできませんでした。しかし、C言語コードを読むことによって必要なパラメータやシステムコール処理の流れから、カーネルの中身やネットワーク通信シーケンスなどがはっきりとイメージできるようになります。応用例としてpingやtftpのソースコードと解説もあり、ネットワークを理解するうえでたいへん参考となりました。

それまで雲の上の誰かが作った抽象的なUNIXというOSやネットワーク機能が、本書によってはっきりとイメージできるようになり、また「自分でも書けるんだ」という意識が生まれ、そこから急速にさまざまな技術やプロトコルを身につけることができました。UNIXとTCP/IPを理解し、またC言語入門書としても最も適した1冊でしょう。

さて次の本を紹介しましょう。ネットワークエンジニアはサーバやルータのCLI（Command Line Interface）で機器のコンフィギュレーションを設定するだけではなく、送受信パケット量や、ルーティングテーブルエントリ、ピアリングセッション状態などCLIによって得られるさまざま

PROFILE

伊勢幸一（いせこういち）　Twitter：@ibucho

1962年生、機械工学を専攻していたが、26才の転職を機にUNIXとTCP/IPの世界へダイブ。SE、CG映画製作とオンラインゲーム開発を経てデータセンター事業者にたどり着き現在に至る。著書に『4Gbpsを超えるWebサーバ構築術』、『SDN/OpenFlowで進化する仮想ネットワーク入門』など。

ネットワーク運用管理エンジニア専用プログラミング言語

awkでプログラミング

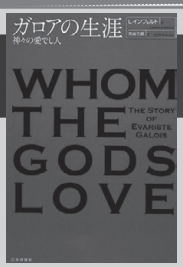
植村富士夫、富永浩之（著）、オーム社、1993年、入手困難／絶版

なテキストデータを分析してシステムの状態を把握します。しかしCLIだけでは思ったとおりの出力が得られないこともあり、一度フル出力をテキストに落として、必要な行を抜き出したり、値を演算したりするわけですが、シェルスクリプトでは機能的に不十分であり、C言語やPerlによってプログラムを作成しデバッグするほどの余裕はありません。その場合awkが使えると非常に便利でした。たとえば、マウントしているファイルシステムの全使用量を合算したい場合や、BGPルーティングテーブルからASパスが4つ以上の経路だけ抜き出すとかいう処理がたった数行で書けます。慣れてくるとshellのコマンドライン上でawkスクリプトを直接書き込むだけで、ある程度の処理ができるようになります。

awkの特徴としてテキストの1行1行を処理対象としていることからオペレータやエンジニアが実行するCLI出力を整形する作業にとても適しているのです。本書は2部構成となっており、基礎編では一般的な書籍と同じくawkの使い方をていねいに解説しているだけですが、応用編ではさまざまなプログラミングアルゴリズムをawkで実現する事例を多く掲載し、単にテキスト処理だけではなく、汎用のプログラミング言語として十分使えることを示しています。また、最後の方には簡易LISPインタプリタなどの実装例もあり、ネットワークエンジニアだけではなく、プログラマ志望者にとってもプログラミング学習の入り口として最適でしょう。また、シェルプロンプトに直接長いawkスクリプト行を叩き込んで、psの出力をソートしたりifconfigの出力からIPアドレスとMACアドレスのテーブルを作成して見せたりすると女の子にモテること間違いなしです。SD

＋α　エンジニアの心のエネルギー

ガロアの生涯――神々の愛でし人



レオポルト・インフェルト（著）、市井三郎（訳）、日本評論社、2008年、1,890円

エヴァリスト・ガロアとは日本でいうと源義経のような存在で数学少年達の憧れであり、群論の基礎を生み出した若者です。筆者は大学時代に漫画『栄光なき天才たち』でガロアの存在を知り、群論に興味を持ち本書や群論関連の書籍を読み始めました。群論は代数学の範疇に入りますが、それは量子力学や相対性理論にも応用されています。あまりにも若すぎ、しかもガロアの論文が当時のフーリエ、コーシー、ガウスなどの大数学者達にも理解できなかったという不運な人生と、女性をめぐって決闘をした結果命を失ってしまうという儚さの対比が実話とは思えません。彼は間違いなく天才であり、彼がもっと長生きしていたら世界の数学や科学、物理学はまた違った発展をしていたでしょう。本書を読むたび、自分がこの時代に生きている奇跡に感謝し、凡庸であっても最大限の努力を続けようという気になります。

14 ITエンジニアのための英語

IT, English,

日本を始め、アメリカ、イギリス、イタリアなどでベンチャーの買収から通信サービスの運用までさまざまな仕事にかかわってきました。文化も宗教も言葉も異なる人と仕事して来て気がついたのは、障害やプロジェクトの遅延などの「ありがち」な問題の原因の多くは「言いたいことが伝わっていないこと」ではないか、ということです。そこで今日は世界を舞台に活躍したいエンジニアの皆さんが「伝える技術」を磨くのに役立つ良書を紹介します。

コンサルティングファームの基本書

考える技術・書く技術
—問題解決力を伸ばすピラミッド原則

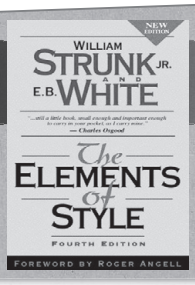
バーバラ・ミント(著)、山崎康司(訳)、ダイヤモンド社、1999年、2,940円

この書籍、英語圏の大企業や経営コンサルティングファームでは「基本書」と呼ばれている古典であります。筆者も某コンサルティングファームとベンチャー勤務時代には、マッキンゼー出身のマネージャに繰り返し読むように指導を受けました。「ピラミッド原則」と呼ばれる同氏の手法は、「伝わりやすい文章の構造とは何か」を教える物です。まず、結論を最初に書いてしまい、その下に結論を支える説明をつける、というやり方です。それを積み上げて行くと、一塊の「論理的でわかりやすい文章」ができあがります。大事なことが一発で伝わる文章ができあがるわけです。この原則を使うと、相手がインド人だろうがボツワナ人だろうが、短時間で言いたいことを「効率的に」伝えることが可能になります。結論が最初に書いてあるので、いちいち「これって何の意味のだろう？」と読み返す必要がないわけです。読者にとって親切な文章なのです。なお、成功している多国籍企業の社内通達や手順書、国際機関の政策報告書など「多国籍環境で伝えるプロ」の手の入った文書を入手して解析すると、どれも見事に「ピラミッド原則」に沿った構造になっています。世界を相手にしたいエンジニアの皆さんは、同書を読んで「ピラミッド原則」を身につけて、仕事で活用してみてください。

英語の文章表現を磨く

The Elements of Style

William Strunk Jr.　E. B. White
英語文章ルールブック



この本はアメリカやイギリスで大学の新入生の「バイブル」と呼ばれている「文章の書き方」の古典本です。新入生達は、この「バイブル」片手に毎日膨大な量の参考文献を読みつつ、論文を作成する方法を学びます。英語圏の大学では論文を書かせることで、「誰にでもわかる文章」「相手を説得する文章」の書き方を徹底的に仕込むわけです。その基本ルールを教えるのが同書なのです。この本、初版の出版はなんと1930年

PROFILE

谷本真由美（たにもとまゆみ）　Twitter：@May_Roma

公認システム監査人（CISA）。ロンドンの投資銀行にてITガバナンス、IT投資評価、プロセス改善担当。ネットベンチャー、コンサルティングファームに勤務後、イタリアの国連専門機関で情報通信官として世界86ヵ国向けの情報通信サービス運用に関わる。Syracuse University情報管理および国際関係論修士。

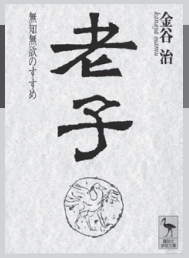
ですが、現在でも読み継がれている超有名本です。英語圏ではプロのライターも同書を手元に置いて参照しながら書くことがあります。同書が他の「文章書き方読本」と異なる点は「英語で何をどう書くべきか」「文章というのはどのように組み立てるべきか」「どこを大文字にするべきか」「どこにカンマを打つべきか」という「戦術」を「実務家の視点」で教えてくれる点です。他の「文章書き方読本」の場合は、実務にかかわっていない小説家や新聞記者が「どういう表現方法するか」「文章とは何か」というわかりにくい哲学や、実務では何の役にも立たない持論を展開する場合が少なくありませんので、まったく役に立ちません。コンパクトな上、手頃な価格なのもすばらしいです。将来海外プロジェクトにかかわりたい方は手元に1冊おいておくことをお勧めいたします。

イギリスは現在開発やテストをやるエンジニアの多くがインド人です。イギリス生まれのインド人移民の二世、三世も多いのですが、なにせ数が足りないので、インドから生インド人を「直輸入」しております。いろいろな会社さんがインドまで直接行きまして、宿敵アメリカやおフランスの会社さんに盗まれる前に、現地で賢そうな若いエンジニアを拾って飛行機に積めてつれてくるのでございます。三角貿易のころとやっていることが変わりません。なんでわざわざインドまで行くのかというと、イギリスの若いものはコーディングや数学や物理学など、「面倒くさいもの」はお勉強したがらないので働くお方がいらっしゃらないのです。そんなギークで辛気くさいものをネチネチ学ぶより、クラブに踊りに行ったり、週末は飲んだくれたりしている方がクールじゃん、というわけです。それから、イギリス人は働くのが大嫌いですが、人に仕事をさせるのが大好きなので、「おら！インド人このコーディングやれや！」とせっせとムチをふるうのです。ここは日本と違いましてもともと「物を作る職人」よりも「口で適当に言い負かして巻き上げる」という海賊型の商売人の方が「素晴らしい」という評価される土地柄がありますので、そういう物なのです。そういうわけで、足りない人をインドから輸入するわけですが、いったんインド人の方が採用されますと、はやり、言葉とか文化が通じやすいので続々とインド人が増えまして、気がつくと、オフィスの一角に「ムガール帝国」ができているわけです。そして、「ちょっとあのムガール帝国（テスト部屋）にこの申請書おいてきてよ」などという隠語がかわされるようになります。お昼になると、職場の電子レンジがカレーで占拠されます。その光景を見ていると、ワタクシの脳裏には、あの筋肉少女帯の「日本をインドにしってしまえ！」という『日本印度化計画』のフレーズがぐるんぐるんと回るのです。SD

+α　ときにはPCの電源を落として……

老子

金谷治（著）、講談社、1997年、1,008円



中国の古典中の古典であり、外国でもさまざまな解釈本が出ています。老子は、人間とは所詮大自然の一部でしかなく、大変無力であり、大自然の一部なのであるから自然に逆らってはならないことを説きます。人間の本質とは、自然の流れや本能に逆らわないことなのです。仕事に迷ったとき、人生に悩んだとき、逃げたいとき、家族が死んだとき、何が本当に大事なのかわからないときに、モニタの電源を消して読んでほしい本です。

Date. 2013 . 5 . 18

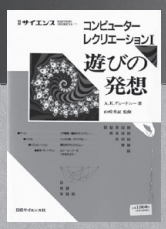
15 編集者の裏をかいてのお勧め本

Information Technology, Science

この特集は新人向けということになっています。しかし、本誌の読者層を考えると、読む人の大多数はどう考えても中堅以上の現役エンジニアです。彼らの興味はきっと「あの○○さんはいったいどんな本を出してくるのか？」という点にあります。書く側も、実のところ、後進のためというポーズをとりつつ「自分はこんな高尚な本を読むカッコいい人間なのだ！」というアピールを狙います。本棚に何を並べるかは実はファッションであり、この特集はいわばファッションショーです。動物園と言ってもいいかもしれません。なので、新人向けとかあまり考えずに書かせていただきます。

コンピュータの裏にある科学

コンピューターレクリエーションI～IV



発行：日経サイエンス社、
発売：日本経済新聞社、
1987年～、入手困難／絶版

高校生のころ、この本を読んで、コンピュータの裏に科学があることを知りました。日経サイエンスの連載をまとめた本です。Iが1987年、IIが1989年に出版され、現在は絶版となっています。コンピュータから離れた高校時代でしたが、あいかわらず興味は強く、知的刺激に飢えた僕の目に偶然とまったのがこの本でした。数学好きの中学生なら十分に読める内容です。

連載の各回は、ゲームを解くプログラム、核戦争での戦略、きれいな図を描くソフトウェアなど読者の興味をひくネタを取り上げています。その実、コンピュータ科学のさまざまな分野――人工知能、再帰、トポロジ、遺伝的アルゴリズム、フラクタル、アナログコンピュータ、オートマトン、チューリングマシン、計算複雑さ、ゲーム理論、物理シミュレーション、探索、複雑系――へと読者をいざなっています。

僕にとってのコンピュータは、最初はやはりゲーム機でした。ただ、消費者で終わるのはしゃくで、ゲームデータの改変や、プログラミング、現実世界の模倣(3Dワイヤフレーム描画、レイトレーシング、スピログラフ描画)など、産む側としての活用を試みていました。いわば、表現する道具としてのコンピュータ、です。そこに、この本が見せてくれた、コンピュータを支える科学、コンピュータあっての科学は、とても新鮮でした。

モンテカルロシミュレーション、ライフゲーム、マンデルブロ集合、重力多体シミュレーション、ロジスティック写像のカオス的振る舞いなどは、たまらず、自分でもプログラムを書いて試したものです。

一部、翻訳を竹内郁雄先生(IPA未踏 統括PM)がなさってます。子供のころに自分が読んだ本の訳者と、今、一緒に仕事をさせていただいているというのは、なんとも感慨深いものです。

教科書どおりじゃない！　先達直伝「効く」ブックガイド　Part 1

PROFILE

首藤一幸（しゅどうかずゆき）　Twitter：@shudo

東京工業大学准教授。IPA未踏プロジェクトマネージャを兼任。早稲田大学、産業技術総合研究所（エンジニアリングと研究）、ウタゴエ㈱（経営・執行）を経て、2008年12月より現職（研究と教育）。エンジニアとしての作品に、Javaスレッド移送システムMOBA、Java JITコンパイラshuJIT、peer-to-peerの基盤ソフトOverlay Weaver、『Binary Hacks』（共著、オライリー・ジャパン）など。

やるべきことに注力するために

ライト、ついてますか ――問題発見の人間学

ドナルド・C・ゴース、ジェラルド・M・ワインバーグ（著）、木村泉（訳）、共立出版、1987年、2,100円

僕を少し悪い人にした本です。大学に入ったころに出会いました。読むにあたって、コンピュータの知識はいっさいいりません。「問題」とは何であって、どう付き合うのがよいかを考えさせてくれます。「問題とは望まれた事柄と認識された事柄の間のズレである」に始まり、解くべきか？　解きたいか？　を問いかける第6部まで、愉快なエピソードが満載です。

第4部「それは誰の問題か？」にかなり影響を受けました。読んで以来、問題らしきものに遭遇するたびに、これは果たして自分の問題なのか？　そうでないなら……と考えるようになりました。たとえば任期付き職員にとって、組織をよくすることは自身の問題ではありません。限りある資金で突っ走っているスタートアップの一員にとって、学術を支える無償の論文査読は、まったく自身の問題ではないわけです。

ところが読み返したところ、自分の問題なのかどうか疑え、なんてことは全然書かれていません。むしろ「人の問題として任せておく」とか、「自分の問題として認識することで解決する」といった、いい人のままでできる方策が書かれてます。あれ？　どこで記憶が曲がったのでしょうか？

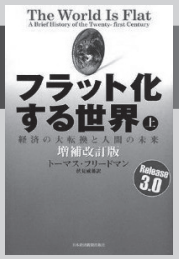
ともあれ、この本の影響で、本当は何が問題なのか？　と常に考えるようになりました。人生の貴重な時間を何にどう費やすか（または費やさないか）を考えさせてくれる本です。

翻訳者は木村泉先生（東工大 名誉教授）です。この本から影響を受けた自分も今は東工大 情報科学科の教員で、訳者の後輩にあたります。面白い御縁です。SD

+α　これからの社会とそこでの人の生き方

フラット化する世界

トーマス・フリードマン（著）、伏見威蕃（訳）、日本経済新聞出版社、2006年、2,100円



これもまた、本屋で偶然手に取った本です。ネットの広がりやサプライチェーンの高度化によって世界全体が平らなフィールドになってきていて、じゃ、そこで僕らはどうすればいいの？　ということを論じた本です。最初の章はインドIT企業インフォシスを採り上げています。彼らがいかに先進国の仕事、それも、低めの人件費を武器にするだけでなく、とても高度な仕事までを請け負っているかを紹介しています。僕自身、1999年にインドIT企業TCSの方々と仕事をしました。彼らの高い能力に触れ、また、当時のお給料（新卒US$200）を聞き、日本の情報サービス業やばい！　日本語の壁（と時差と距離）がなかったら即死！　と感じたことを強く思い出しました。あれから13年、なぜか意外とそうはなっていません。

この本でとくに印象に残っているのは、IQ（知能指数）よりもCQ（好奇心）とPQ（情熱）が重要、という言葉です。自分が感じ、考えてきたこととぴったり一致していました。人は、興味あることはいくらでも、強制されずとも、やります。しかも、今やネット上の資料／講義でいくらでも学習できますし、その道のすごい人と直接対話することもできます。たとえば、大学の研究室に入らずとも、論文を手に入れたり、第一人者に接触したりできるわけです。これを、梅田望夫氏は『ウェブ進化論』の中で「学習の高速道路」と言い、Bill Gates氏は2010年、「今後5年のうちに世界最高の講義がWeb上で無料で見つかる」と言いました。

16 運用スキルを無駄なく身に付ける

Networking, Operating System, Infrastructure

「優れたエンジニア」とは、いったい何をもって、そう呼べるのでしょうか。どんな困難な課題でも技術で解決できる人であったり、多くの新しい技術的挑戦を成功させる人であったり、技術面から確実にユーザ・事業価値へ貢献できる人など、今挙げたことはほんの一例ですが、場合によってさまざまなとらえ方ができるでしょう。

とはいえ、いずれの場合においてもエンジニアたるもの「スキル」は絶対に必要です。経験の浅いうちは、難しいことは考えず、とにかく技術力を磨くという選択も有効だと思います。筆者は普段、大規模Webサービスの現場で仕事をしていますが、そこで新卒エンジニアによくオススメする書籍を、インフラ(サーバ)構築／運用のスキルを伸ばすという観点で紹介したいと思います。

rootを任されたときに読む本

Linuxの教科書 改訂版

高町健一郎、大津真、佐藤竜一、小林峰子、安田幸弘(著)、IDGジャパン(2007年)、

毎日コミュニケーションズ(現現マイナビ)より再販(2011年)、入手困難／絶版

筆者も新人のころに受講したのですが、会社の新卒エンジニア向けの研修で、Linuxの講座が1～2日含まれているケースも多いのではないでしょうか。1～2日でLinuxの概要であったり簡単なコマンド操作や、基本的なソフトウェアのインストール方法くらいまでを勉強する座学です。ですが、開発の現場でLinuxサーバの構築や運用を行うにあたっては、いわゆるLinux OSのお作法(たとえば、ログファイルはどこに配置すべきかなど)であったり、プロセスやファイルシステムの操作、バックアップ・監視、セキュリティ対策、性能や障害時の調査・復旧手法など、運用するうえで学んでおくべきことがたくさんあります。

Linuxサーバを運用していく現場で必ず役に立つ内容を含めた実践よりの内容となっています。どこに何のファイルがあるのだろう？　とか、こういう場合どこを確認すればいいのだろう？　など、しくみとか概念よりも、まず手を動かすために必要なことやお作法・テクニックなどが網羅されています。内容も全240ページとそれほど厚くなく、深い内容には言及しませんが、必要な知識を浅く広くカバーしています。Linuxサーバの運用に慣れていくフェーズで読むケースにフィットしているため、実際に構築／運用するうえできっと出てくる「？」な内容に答えてくれる1冊になるはずです。

筆者のこれまでの経験からすると、本書の内容はいずれも、Linuxサーバを扱ううえで、root(OSのシステム管理者アカウントのこと)になって諸作業をする方には、一とおり読んで理解していただきたい内容です。本書を読んだ上で、物足りない部分や、もっと深い知識が必要となった段階で、その分野の専門書を購入すればよいと思います。

PROFILE

並河祐貴（なみかわゆうき）株式会社サイバーエージェント

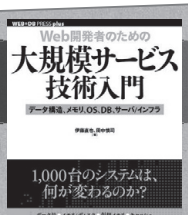
Twitter：@namikawa　Blog（http://d.hatena.ne.jp/rx7/）

これまで、数々のオープンソースソフトウェアの検証／導入／運用や、クラウドサービスを利用したサービス構築／運用を実践。近年は、サイバーエージェントにてAmebaなどの大規模Webサービスのバックエンドを支える業務に携わっている。主な著書は『クラウドAmazon EC2/S3のすべて～実践者から学ぶ設計／構築／運用ノウハウ～』、『Redmine――もっと手軽にプロジェクト管理！』。

事例からアーキテクチャを学べる本

[Web開発者のための]大規模サービス技術入門

伊藤直也、田中慎司（著）、技術評論社、2010年、2,709円



世の中にはたくさんのWebサービスが存在します。誰もが知っているWebサービスは、利用ユーザ数が多いことを意味しますので、世間一般には“大規模”と呼ばれる部類でしょう。

たとえば、Facebookは2010年の時点で、バックエンドのサーバ台数が6万台以上あったと言われています。そんな大規模なWebサービスの舞台裏はどのように動いているのでしょうか。それを解く鍵は、大規模なWebサービスを運営している数社が、この3～4年の間で自社のサービスのアーキテクチャを解説した書籍を出版しています。その中の1冊が本書です。この書籍は数十億PV／月（当時）の訪問がある「はてな」のWebサービスを運営する技術者によって書かれたもので、一般的なハードウェアとオープンソースソフトウェアで構成された大規模なWebサービスが、どういったアーキテクチャ・思想で作られてきたかが記載されています。書籍自体はインフラに限った話題ばかりではなく、アプリケーションの実装面の話もまんべんなく散りばめられていて、実際に「はてな」で起こってきた事例をベースに、浅く広くではありますが、Webサービスのアーキテクチャ全体からあらゆる視点で書かれていますので、大規模なWebサービスのシステムがどう構成されているかを俯瞰するのに最適な書籍だと筆者は考えています。

また、本書は㈱はてなで実際に行われているインターンシップの講義内容がベースとなっているので、本書を読み進めることで体系的に学べる点も魅力の1つです。大規模サービスを支えるためのインフラとアプリケーション実装についての理解が深まれば、エンジニアとしての幅が大きく広がるはずです。SD

+α　借金はエンジニアの敵なり

ナニワ金融道

青木雄二（著）、講談社、777円（文庫）

Linuxサーバ管理、大規模サービスのアーキテクチャの次は、エンジニアとして今のトレンドを考えるとクラウドコンピューティングを意識しておくべきですが、ここはあえて脱線して、コミックを紹介します（笑）。

ご存じの方も多いと思いますが、消費者金融を舞台として、さまざまなケースの“借金”にまつわる人間模様を描いた作品です。今回のテーマが社会人になってまだ日が浅い新人向けとうったテーマから、間違った形でお金のトラブルに巻き込まれた場合、人の行く末はどうなっていくのかを知るたとえを学ぶには良いコミックだと思います。

また、金融まわりのお金の流れが理解できたり、ローン関連の法律や用語の勉強ができます。これらが役に立つ機会はそうそうないかもしれませんが、予期せぬトラブルに巻き込まれないためにも普段から知識と意識を持っておくのも大事でしょう。

Date. 2013 . 5 . 18

17 試行錯誤をする前に読む!

Networking, Operating System, Infrastructure

新卒でシステム管理を任されてからインフラにどっぷり浸かってますが、インフラをやりたくてやったわけでもなかったことを思い出しました。実際触ってみたら面白くてどんどんはまっていったのですが、何が自分をそうさせたのか?——それは会社のメールやDNS、ホームページなどを構築・運用することでこのインターネットの一部を自分が作っていると実感したからです。当時から10年以上経ちましたがインターネットは進化し続けており、より良い本がどんどん出ています。これらはインフラを目指す新人へ。

インフラエンジニアの試金石

ウェブオペレーション —サイト運用管理の実践テクニック

John Allspaw , Jesse Robbins（編）, 角征典（訳）、オライリージャパン、2011年、2,730円

Web業界のインフラに携わるのであれば、言いたいことはすべて1章に詰まっているのでぜひ1章だけでも読んでもらいたいです。そして1章を読んでこんな仕事やキャリアがあるのかとワクワクしてどうしようもない人は、ぜひわが社へ。というのは冗談ですが、ウェブオペレーションというインフラエンジニアの新しいキャリアが描かれています。

Web業界はまだまだ未熟な世界なので今後どのように発展していくか正直わかりません。自分も最初は手探りで『UNIXシステム管理』というオライリー・ジャパンの分厚い本を読みながらUNIXシステムの基礎を覚え、『マスタリングTCP/IP』でネットワークの基礎を覚え、プロトコルはRFCを読み、といったことを続けながら、システム管理をしていました。しかしながらWebサイト運用や障害対応といった技術やノウハウは当然本には書かれていなく、それらは経験してみないとわからないことだらけでした。エラーを読み解き、関係性を調べ、障害ポイントやボトルネックを洗い出します。手探りで試行錯誤していくことで、だんだんと類似点が見つかり対応手順はフレームワーク化するのですが、新たなサービスごとに新しい問題が出てきたりもします。

運用の中でどのようなときにどう対処したらいいのか? これは当然システムによって異なります。本書では現場での苦悩を味わいながらもサービスをより良いものへと試行錯誤するエンジニアの思いやテクニックが紹介されています。技術的なこともたくさん出てくるのでいろいろと身につくと思いますが、本書を読むときは登場するエンジニアの視点で読み、そのとき何を思っていたのかを考えてもらいたいです。ほとんどのエンジニアはサービスに対する強い意識を持っています。このことはそのサービスをより良いものに変えていく原動力になっており、たとえ素晴らしい技術を持っていたとしても、この意識がなければ良い運用はできないと思います。たとえばサイトの表示が重くなればサイトの信頼性にもかかわりますが、運用者に意識がなければ放置してしまうと思います。イ

PROFILE

元井正明（もといまさあき）　株式会社マイクロアド（http://www.microad.co.jp/）

新卒での就職先ではシステム管理を任され、その後青年海外協力隊にて南国でシステム管理者となる。現在は株式会社マイクロアドにてプロダクトから社内ネットワークまで、システムに関するインフラ全般を担当。マイクロアド創業当時はすべて1人で担当し、現在はインフラチームの中核を担う。

ンフラエンジニアを目指す方にとっては彼らの思考や行動が参考になるはずなので、ぜひ彼らのように試行錯誤してもらいたいものですが、今や安定したサービスが多いこの時代にこのような経験できないと思いますので、本書でこんな世界もあるのかと感じてもらいたいです。

全体像を見て理解を加速！

インターネットのカタチ――もろさが織り成す粘り強い世界

あきみち、空閑洋平（著）、オーム社、2011年、1,995円

あって当然のインフラになってきたインターネットですが、実際どのようにつながっているのか、そしてその中では何が起きているのか？

ウェブ上でコンテンツを見るという日々、そんなインターネットの裏側のディープな世界を本書を通じてぜひ味わってもらいたいです。

よくたとえでメールはどのようなしくみで配信されているかというのがあり、メーラから送信サーバに接続して、いや接続する前にDNSでサーバのIPアドレスを解決して、なんとかっていうプロトコルを使って別の相手のサーバに送信して……、でもそれは物理的にどうなっているのでしょうか？

本書にも書かれていますが、技術書というよりは読み物に近くとても読みやすく、インターネットの裏側ではこんなことが起こっていましたといったことがわかりやすく描かれています。つながっているからこその壊れ方があり、何気なく設定したものがあるところではつなげなくしてしまう設定だった、しかし中の人たちによってすぐに復旧したといった話や、つながっているはずなのに、あの国ではあのサイトが見られないといった話があります。

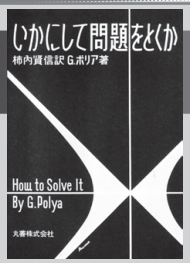
また技術を覚える前に読んでおくと、より理解が深まると思います。昔、TCP/IPの基礎本を何度も繰り返し読んだのですが、全体感をとらえるのが難しかった記憶があります。しかし本書のように全体感を把握することができていれば、そのあとに技術書を読むことで簡単に理解が深まるはずです。なぜ昔にこのような本がなかったのか悔やまれます。そしてインターネットでのサービスを扱うということはインターネットのカタチの一部になることなので、ぜひ自分の力でインターネットのカタチを作ってください。

SD

＋α　突破口の見つけ方

いかにして問題をとくか

G.ポリア（著）、柿内賢信（訳）、丸善、1975年、1,575円



数学の問題を解くとき、何をどのように考えて回答までたどり着くのかということを文章にしており、自分の中でなんとなくたどりついていた過程の思考を説明してくれています。数学だけに限らずすべての問題解決に通ずるところがあり、例えば何かシステムを作る場合においても過程はどうであれ、最終的にできあがるシステムは同じもので、問題を解けないということは問題自体を理解していないことなのです。

これからいくつもの壁にぶち当たると思いますが、そんなときは一度立ち止まり、問題文を読み返すと突破口が見つかるかもしれません。

Date. 2013 . 5 . 18

18 クラウド時代を読み解き、仕事のスタイルを確立する

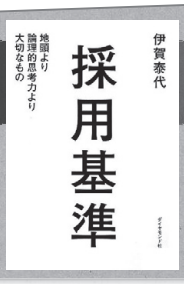
IT，Cloud Computing Engineer

今、クラウドがIT業界を大きく揺さぶっていることは皆さんご存じだと思います。これまで「IT業界に入る」ということは「技術者となり、作り方を学ぶ」ことを意味していましたが、とくにシステム開発の現場では「作るのではなく、使うことが正義」に変わりつつあります。こうした変化は、みなさんがこの春から入社する会社の先輩・上司も未体験のもので、もしかすると時代にあった正しいアプローチを教えてくれない可能性がある、ということを認識しておかなくてはいけません。筆者からは、なぜクラウドに大きなインパクトがあるのか、クラウド時代の仕事とはどういうものなのかをつかめる良書を紹介したいと思います。

必読書

採用基準

伊賀泰代(著)、ダイヤモンド社、2012年、1,575円



タイトルだけ見ると「人事担当者でもない新入社員が、何でこんな本を？」と思ってしまうかもしれませんが、これから社会人としてデビューを飾る皆さんにぜひ読んでもらいたい、人生のためになる1冊です。著者は外資系のコンサルティングファームで採用マネージャを務めたという経験から、日本人に足りない「リーダーシップ」について、その概念から丁寧に書かれています。リーダーシップというと、多くの方が「周りをぐいぐいと引っ張っていく力」というように考えると思いますが、そうではありません。どちらかというと「主体性」という言葉がしっくりくるような概念で、そうした力を持つことがこれからの社会人人生を歩む上でとても大切だと考えさせられる1冊です。

(会社によると思いますが)社会人として会社に入ると、実にさまざまな、理不尽なことを要求されます。筆者も新卒で入社した会社では「会社とは理不尽なところだから」とさんざん説明され、合理的とは言えない仕事をやらされてきました。これは犬のしつけと一緒で「役職が上の人のことには、無条件に従え」という精神を植え付けるには非常に合理的なしくみです。こういう仕事をしているうちに、業務命令の合理性などについて考えなくなるからです。これは、階層型組織で、会社として行うべき業務の範囲が狭く、かつ明確である場合にはとても適した教育システムだと思います。

ですが、残念ながら今は違います。変化が早く、役職が上の人の判断が正しいとは限らない。ややもすると「上の人の方が、現場よりも判断できない」という事態が起きつつあります。こうした環境においては、必然的に意志決定が分散化され、現場におけるリーダーシップの有無が会社の存亡を分けるような事態に直結しつつあります。このような時代を生き抜くためには、リーダーシップの存在が不可欠です。そして、その必要性は、会社だけが理解すればよいというものではなく、新入社員のみなさんを含めた一人一人が認識する必要があるということを繰り返し訴えています。

皆さんも、会社に入ると社長や現場の部長など、さまざまなリーダーに会うことになると思

PROFILE

大石良（おおいしりょう）　Twitter：@ooishi

1973年新潟市生まれ。株式会社サーバーワークス代表取締役。10歳のときにSHARP X1に触れてPCライフをスタート。総合商社丸紅でインターネット通信事業に従事した後、2000年に起業。現在はAWS専業のクラウドインテグレーターとして事業拡大に奔走する傍ら、多数の執筆・講演活動を行っている。

います。もちろん、そうした人が発揮するリーダーシップは大切ですが、誰かがリーダー、あとはフォロワーという組織がうまくいくことはありません。

本書では、そのことを次のように伝えています。

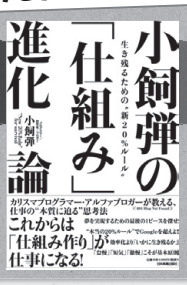
「現状を変えられるのは、神でもスーパースターでもありません。必要なのは、組織のあらゆる場所で、目の前の変革を地道に主導するリーダーシップの総量が、一定以上まで増えることです」（同書、P.182）

先行きが見えない世の中と言われます。ですが、リーダーシップを発揮して自分が自分のキャリアのリーダーになることで、この先に何が起きても自分の道を自分で決められる、自分のキャリアに責任の持てるエンジニアになれると筆者は思います。本書は、そのための入り口を照らしてくれる、とても貴重な一冊です。

クラウドを目の前にしたエンジニアに求められる、仕事への取り組み方を学べる書

小飼弾の『仕組み』進化論

小飼弾（著）、日本実業出版社、2009年、1,575円



著名なブロガー、プログラマである小飼弾さんが、エンジニアに求められる「仕組み」とその作り方、考え方について詳しく説明されています。同氏がプログラマだからということもあり、いわゆる「仕事術」にとどまらず、「プログラマがどのようにしくみについて考えるべきか」という視点で書かれており、エンジニアを目指すみなさんにとってとても読みやすいと思います。

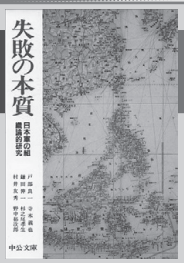
実は本書、クラウドのことを意識して書かれているわけではないのですが、「クラウドというプログラマブルなインフラの運用を、どのように自動化していくべきか」という点でも気づく点が多く、今筆者たちが読んでもとても勉強になります。

筆者たちの会社も、障害などを起こしたときに「次に同じことが起きないしくみ」をとても大切にしていますが、そうした「しくみ」を作るためにはどのような考え方が必要なのかといった点を、航空機業界のたとえ話で説明されているくだりなどは、耳の痛い方も多いのではないかと思います。SD

+α　なぜ失敗するのか

失敗の本質――日本軍の組織的研究

戸部良一、寺本義也、鎌田伸一、杉之尾孝生、村井友秀、野中郁次郎（著）、中央公論社、1991年、800円（文庫）



なぜ日本が勝算のない太平洋戦争に突き進み、そして破滅的な最後を迎えたのか、組織の観点から分析された名書中の名書です。非常に長く、難解で、展開も子細に渡っているので読解力と忍耐力が必要ではありますが、ぜひ何度も読んでほしい本です。これから社会人を迎えるにあたり、最初はどうしても上司・先輩から教わることが多く、そして自然と「空気を読む」ことに慣れていってしまうと思います。ところが、筆者たちが美徳としているこの「空気を読む」とか「阿吽の呼吸」といったものが、どれほど悲惨な結果を招くことになるのか、筆者たちはあまりにも知らな過ぎます。間違っていることを「間違っている」と勇気を持って言うことの大切さを教えてくれる、貴重な1冊です。

19 俺の屍を越えて行け!

Operating System, Infrastructure、CPU、Hardware

ドッグイヤーと言われているこのコンピュータ業界で、これからを支える若手の皆さんの勢いを見ていると、日々頭が下がる思いです。ただ、コンピュータは複雑化し続けており、9年前の筆者のときよりも、複雑さは増しています。抽象化されているとはいえ、何かを少し掘り下げてみれば、さまざまな抽象化レイヤの隙間からはすぐにコンピュータの素顔が現れます。今回は基礎力になる、「プロセッサ本」を2冊ほど選んでみました。ぜひ筆者たちの世代をブチ抜いてください。

x86アーキテクチャを極める

はじめて読む486

蒲池 輝尚(著)、アスキー、1994年、2,548円

現在、WindowsやLinux、またOS Xといったオペレーティングシステムはおもに Intel の x86 アーキテクチャ上で動作しています。現在の x86 プロセッサは 64 ビットにも対応し、複雑なものとなっています。しかし、ソフトウェアの側からみると、現在のプロセッサは 486 の延長線にあり、つまり、現在のプロセッサを理解したければ、まずは、当時の 486 のことを知るのが近道となるでしょう。

これまで、多くのノートパソコンやサーバなどは x86 アーキテクチャのプロセッサを搭載してきましたが、最近のスマートフォンやタブレット、今後はサーバでも低消費電力・高密度を特徴とする ARM アーキテクチャを採用する流れも感じます。それでもなお、直近で皆さんが見かけるコンピュータの多くは、本書で語られる、x86 アーキテクチャベースのものがほとんどだと言えるでしょう。

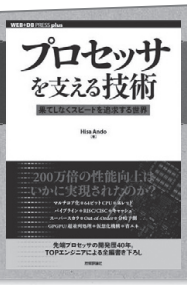
本書はとくに、ITインフラエンジニアと呼ばれるような職種になられる方、および低レイヤなソフトウェアの開発にかかわられる方で、ある程度のプログラミングの知識をお持ちの方にお勧めします。x86 アーキテクチャのサーバを構築・運用するとき、またシステムの構成を検討する立場であれば、リーダークラスのエンジニアならイメージとして持っておきたい知識ですし、また最近は仮想化技術の導入も盛んですが、いわゆるサーバ仮想化における仮想マシンとは多くの場合「ソフトウェア的に作り出された x86 アーキテクチャのコンピュータ」を指すことがほとんどですので、このレイヤの知識を持っていて損をすることはないはずです。

情報系の勉強をされた方なら、ある程度はすでにご存じの内容でしょう。もし本書を“難しい”と感じた場合には、『はじめて読む 8086』という書籍も手に取ってみてください。

ボトルネックを見つける本

プロセッサを支える技術

Hisa Ando(著)、技術評論社、2011年、2,709円



40年にわたるプロセッサの開発経験を持つ著者が、プロセッサのしくみから、現行のプロ

PROFILE
長谷川猛（はせがわたけし）　Twitter：@hasegaw
1981年、マイコン「COMPO BS/80」がある家庭に生まれ、幼少期からパソコンに囲まれた日々を過ごす。迷いなくIT業界を志し、SRAにて7年間のシステム構築および提案経験ののち、現在はFusion-ioのSEとして活動中。OSなどに主な関心を持つ。『萌え萌えうにっくす！ UNIXネットワーク管理ガイド』『Xen徹底入門』共著、本誌やgihyo.jpへの寄稿も多数。

セッサで採られているさまざまな手法までを1冊にまとめた書籍です。この著者は、プロセッサに関しての書籍をいくつも出されていますが、その中では一番難易度が低めで、情報が体系的にまとめられており読みやすい書籍です。

本書の前半（とくに第1章、第2章）は、プロセッサや歴史や基本的なしくみを、特定のアーキテクチャに偏らず、体系的に解説しています。また、これまで各社がどのようなプロセッサを出してきたかなどの解説もありますので、各種プロセッサの歴史や特性について初見の方、もしくはざっくりとおさらいしたい方にも有用でしょう。第3章では、おもにプロセッサ内部の性能向上テクニックが解説されており、光や電気の速度という制限の中でプロセッサの開発者らが、性能を押し上げてきたのかを伺い知れるでしょう。第4章以降では仮想化マシン技術のしくみ、マルチスレッド、マルチプロセッサ、さらには、PCI Expressなどにも軽く触れられており、コンピュータの概要をつかみやすい構成です。

本書で解説されている内容は、コンピュータのしくみから、今どきのコンピュータであれば当たり前のように使われている技術に焦点が絞られています。このため、本書で今どきのコンピュータのしくみを理解しておけば、やはりITインフラエンジニアや開発者としての仕事にきっと役に立つ知識になると思います。

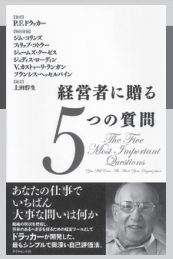
筆者は現在Flashメモリによるシステムの高速化ソリューションを提供するベンダで仕事をしています。この会社が提供する製品は、コンピュータのI/O性能を劇的に向上させますが、コンピュータの1台あたりの性能が底上げされたときに引っかかるポイントは、アルゴリズムやソフトウェアの設計上の問題でなければ、だいたいは本書の中盤以降に書かれている内容とかかわりがあるように思います。このため、将来、性能問題に取り組むエンジニアの皆さんにはとくに目を通していただきたい内容です。

また、同じ執筆者による『コンピュータ設計の基礎』、『高性能コンピュータ技術の基礎』（毎日コミュニケーションズ）では、プロセッサ内の動作がわかりやすく解説されています。もともとのターゲットはプロセッサ設計にかかわられる方向けとのことですが、今回紹介した本書を読み終わったあとに、これらのシリーズを読み進めてみてはいかがでしょうか。SD

＋α　利益を最大化する方法を考えていますか

経営者に贈る5つの質問

P.F.ドラッカー（他著）上田惇生（訳）、ダイヤモンド社、2009年、1,575円



いわずもがな、20世紀～21世紀にかけて大きな影響をもたらした経営思想家による、晩年の1冊です。ドラッカーといえば『マネジメント』または『もし高校野球の女子マネージャーがドラッカーの「マネジメント」を読んだら』などがとくに有名ですが、成果の対価をいただく「社会人」としての第一歩に、本書はお勧めできると思います。自身への5つの問いかけを通して、自身が誰から利益を受けているのか、他者にどんな利益を提供できるのか、本書を通じて自身の考えを整理できるでしょう。

コンピュータエンジニアとして活躍していくうえでは技術スキルがもちろん大事です。しかし、あなたは、いつでもどこでも人や顧客に囲まれることでしょう。本書は100ページ程度と手軽ですし、また、“経営者に贈る～”とタイトル付けされていますが、経営者に限らず誰にでも役立つ1冊だと確信しています。

I 楽しいネタを盛り込むのも エンジニアの流儀

PROFILE　田中邦裕（たなかくにひろ）

1978年大阪生まれ。舞鶴高専でロボコンに明け暮れる日々のなか、インターネットに魅せられ1996年にさくらインターネットを創業してサーバ事業を開始。日本Apacheユーザ会コアメンバーほか、OSS系にもかかわる。趣味は、旅行、電子工作、アニメ、DTMほか、週末プログラマとして「とあるジェネレータ」などのWebサービスも開発する。

IT業界ではなぜかラノベやアニメ好きの人が多く、社内での会話やイベントのプレゼンテーションなどで、知らずにはついていけないという状況に出くわします。逆に言うと、そのような人たちをクスっと笑わせるコミュニケーションを行うことによって、急速に距離を縮められるというメリットがあります。

以前、「ITインフラ業界にとっての深夜アニメは『接待ゴルフ』のようなものなのか？」というブログ記事がはてなブックマークで話題になりましたが、大してアニメ好きでないにもかかわらず、仕事の一環としてアニメを見ている人々の話が紹介されていました。ちなみに、はてなブックマークでは「アニメネタ気持ち悪いのでやめてほしいです」というコメントに多くのスターが付与されており、どうやら諸刃の剣であることは間違いないようですが、それは覚悟しつつ技術系勉強会のプレゼン資料で役立つラノベ・漫画を紹介したいと思います。

タイトルロゴが特徴的で見栄えのする『とあるシリーズ』

“とあるシリーズ”は『とある魔術の禁書目録（インデックス）』と『とある科学の超電磁砲（レールガン）』という2つの作品を総称したもので、科学と魔術が存在する世界を舞台に、少年少女がさまざまな大人の思惑に翻弄される様子を描いた人気作です。元々はラノベとして一世を風靡（ふうび）し、そのあとアニメ化されてからさらに人気が広がりました。

ただ今回お勧め候補としてあげた理由は物語の内容ではなく、その特徴的なタイトルロゴにあります。とあるシリーズを見たことがない人でも、このタイトルロゴはいろいろなところで目にされたという方は多いのではないでしょうか。とある○○の△△△△（××××）という型にはめるだけで、なんとなく「とある風」になるのが不思議なところです。

とある魔術の禁書目録（インデックス）（1）

アスキー・メディアワークス／鎌池和馬 著、灰村キヨタカ イラスト／A6判・312ページ／599円

たとえば、「とある技評の月刊雑誌（ソフトウェアデザイン）」といったように語呂遊びを行うことができ、まるで5－7－5の俳句を詠むかのごとく、粋なキャッチを作ることができます。

ちなみに手前味噌ながら、上記のような言葉を入れるだけでロゴ画像を作成するというサイトを運営しており、まるでアニメのタイトルロゴのような画像を簡単に作ることができます（下図参照）。秋葉原や日本橋の電気街などを歩いていると、商品説明POPなどに拙作のロゴジェネレータで作成したと思しき画像が貼ってあり、密かに喜んだりすることもあります。

キャラクターのしゃべり方が特徴的な『侵略！イカ娘』

『侵略！イカ娘』は、海を汚す人間たちを征服しようと考え陸に上がったイカ娘によるドタバタ劇にフォーカスした、コメディ作品です。チャンピオンで連載中の漫画であり、アニメ化されてさらに人気を博しました。

この作品の特徴はイカ娘のかわいさに集約さ

れるのですが、そのうえでしゃべり方が独特の“イカ語”であり、それをまねることでプレゼン資料に華を添えることができます。

ここからはイカ語で話したいのでゲソが、語尾にゲソとつけるだけで、いい感じになるんでゲソ。そんな状況にスルメイカという意見は多いでゲソが、プレゼン資料などでイカとかゲソとか付けておくとそれなりにウケるんじゃなイカ。1つ問題があると言えば、イカ語ばかりを変換しているといつのまにか仮名漢字変換の辞書が侵略されてしまうでゲソ。実際にTwitterなどで「なイカと思います」とか、イカの部分だけ独立して変換されている恥ずかしい人に出会うことがあるから、気をつけるでゲソ（てな具合です）。

一世を風靡した神アニメ「魔法少女まどか☆マギカ」

『魔法少女まどか☆マギカ』は、2011年に放送されたアニメオリジナル作品です。最近は多くの深夜アニメが放送され、毎シーズンごとにどのアニメを見ればいいのかと前評判を調べるわけですが、良い意味で前評判を裏切ったのが本作品です。魔法少女という設定と、ほのぼの系に強みのある蒼樹うめ先生のキャラクターデザインということもあり、当初は単なる萌えアニメだろうと思われていたのですが、途中からのシリアスで迫力のある脚本によって、一挙に神アニメとして一世を風靡しました。

ストーリーもさることながら、今回は「僕と契約してよ」という有名な台詞が、勉強会界隈で多用されたことから、一般の方々に知られるところとなりました。

たとえば、会社の肩書きでプレゼンするときなど、どうしても会社の宣伝をしないといけないことがあると思います。そんなときに露骨に宣伝をしてしまうと場がしらけてしまうわけですが、上記の言葉を活用して、

「僕と契約してクラウド使いになってよ」

とか

「僕と契約して紹介実績をちょうだいよ」

とか、クスッと笑いを誘いつつ意図を伝えるという活用が可能になります。応用編としては、「僕と契約して社員になってよ」とか、求人に使う人もまま見受けられます。

◆ ◆ ◆

以上、お勧め図書という趣旨から言うとずいぶんと冒険した感がありますが、息抜き企画担当として3作品を紹介させていただきました。

なお、どうしても流行り廃りがありますので、常に新しい情報をキャッチすることも重要です。現在のところ、劇場版を公開中の『とある魔術の禁書目録』や、4月から第二期放送中の『とある科学の超電磁砲』、劇場版の予定がある『魔法少女まどか☆マギカ』は盛り上がる可能性があるものの、『侵略！イカ娘』については先行きが不透明です。

逆に寿命の長い鉄板ネタを使うとはずす可能性が減り、たとえばガンダムネタなら40代に、エヴェンゲリヲンネタなら30代にウケやすいと言えます。そのほか、『ゲームセンターあらし』や、『ハイスコアガール』、またアニメ漫画からは外れますが、『ベーマガ（マイコンベーシックマガジン）』などは、そのタイトルを出すだけでも一定の年齢層以上にはウケやすいという傾向がありますので、活用するのもよいでしょう。

良い情報と、楽しいネタで、今までよりもさらに楽しいプレゼンにつながると思いますので、ぜひ成功を祈っています。SD

魔法少女まどか☆マギカ
※写真はBlu-ray 1巻
アニプレックス／新房昭之 監督、虚淵玄 脚本、蒼樹うめ キャラクター原案、シャフト アニメーション制作／Blu-ray（1巻）7,350円
©Magica Quartet／Aniplex・Madoka Partners・MBS

侵略！イカ娘（1）
秋田書店／安部真弘 著／コミックス・162ページ／440円

II 20代のためのガンダム入門

PROFILE　砂金信一郎（いさごしんいちろう）　Twitter：@shin135
マイクロソフトでWindows Azureなどのクラウド活用を啓蒙するエバンジェリスト。東工大卒業後、日本オラクルで新規事業開発などを歴任した後、戦略コンサルタント、スタートアップ企業マザーズ上場を経て現職。ガンダムに限らずアニメ、サブカル方面の造詣が深い。クラウディアの仕掛け人でもある。

第1特集　新しい季節に君へ　IT業界ビギナーのためのお勧め書籍55冊+α……大好評発売中!!

アラフォーの先輩とチームを組んだとき、勇気を出して言ってみよう。

「好きなモビルスーツは何ですか?

ちなみに僕は、グフ・カスタムです（ｷﾘｯ）と。

多様化が進んだ新社会人の皆さんにはにわかに信じがたいことかもしれないが、IT業界の中核たる我々世代は多感な時期にガンダム、エヴァンゲリヲン、攻殻機動隊をはじめとする社会現象化したアニメ作品に魂を揺さぶられる共通の原体験を持っている。断言しよう。IT業界において、ガンダムはゴルフや飲み会より優れたビジネスコミュニケーションツールなのである。

ここでは、ガンダムに絞って最短でその基礎を身につける方法を伝授したい。

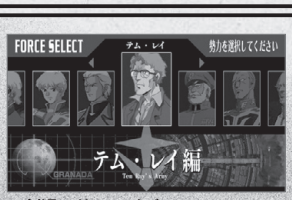
立てよ新人!

「何から見ればよいですか？」

煙たがられながらもガンダムの重要性を若者に説いた際の代表的な反応である。原点に立ち返ってファーストガンダム※劇場版三部作か、最新作で作画がきれいなガンダムユニコーン※あたりが妥当なところで、気の利いた先輩なら手持ちのDVDも貸してくれるだろう。しかし、ガンダム世界観の本質を最短で網羅的に理解する方法は『ギレンの野望※』を自力でクリアすることであると私は考えている。

『ギレンの野望』は三国志や大戦略とほぼ同じ戦略シミュレーションゲームで、ガンダムの宇宙世紀を舞台に兵器であるモビルスーツを開発し、パイロットを育て、戦闘で勝利して資源と資金を稼ぎ、最終的に地球圏を統一することを目的にしている。正直なところ難易度は高く初心者向けではないので、PSPや時間のない人は攻略本を読むだけでもよいかもしれない。

『ギレンの野望』から体感できる重要なポイントのひとつは、主役機のガンダムが唯一無二の無敵スーパーロボットではなく兵器として扱われていることの理解である。『ギレンの野望』においては、RX-78-2ガンダム1機の建造にかかる資金はおよそMS-06F量産型ザク20機と同じである。闇雲にガンダムを量産すれば勝てるというものではなく、自軍の状況に応じた判断をしてゆかないと勝つことはできない。また、連邦軍と対立するジオンが悪という構図ではなく、互いに信じる正義がある点も重要なラーニングである。映像作品でも随所で描かれてはいるが、『ギレンの野望』をジオン勢力でプレイすることで独立戦争の正義や背景をより正しく理解できる。ガンダムが登場するのは一年戦争※後半であり、それまではジオンが開発した画期的な新兵器であるモビルスーツ、ザクは圧倒的だ。時代の趨勢（すうせい）を一変するザク開発の成功があってこそ、連邦軍に対し国力が1/30にしか満たないジオンは独立戦争の開戦を決意したのである。現実世界でも、一見大手企業が独占的なビジネスを展開しているように見えて、その状況を覆す技術やサービスがないわけではない。IT業界



©創通・サンライズ

機動戦士ガンダム ギレンの野望 アクシズの脅威V
バンダイナムコゲームス／2009年2月12日発売／PSP用ソフト（5,500円）、PS2用ソフト（6,800円）

※一年戦争…ファーストガンダムの舞台となったジオン公国軍対地球連邦軍の戦い。開戦後約1年で終戦となったので後に一年戦争と呼ばれることになる。
※パチ組み…塗装などせずキット品のプラモデルをそのまま組み立てること。HG（1/144）シリーズであれば数時間で完成させられる。

のガンダムファンにジオン派が多いのは、このあたりのロマンに共感しているためであろう。

通常の3倍早く、職場に馴染むには

書籍では、『評伝シャア・アズナブル(上/下)』がお勧めだ。ガンダムシリーズで最高の人気を誇るシャアの苦悩を、作品上の発言や行動を細かく拾って解説している。シャアに自分自身を投影し続けてきたアラフォー世代の悩みが手に取るように理解できるようになるだろう。ジオン公国軍エースの赤い彗星ではあるが、建国者ジオンを父に持つ出自や、ニュータイプとしての能力ではアムロに敵わないことを理解し、客観的な見識の高さゆえ最後まで苦悩する。

また、本書を読むことで身につく大事な素養は、いわゆる「名ゼリフ」である。

「ランバ・ラル、設計の中で仕様を忘れた」「悲しいけどこれ仕様なのよね」「バグとは違うのだよ! バグとはっ」「エンドユーザーは無理難題をおっしゃる」「すまんが、みんなの土日をくれ」「ぜ、全滅? 12機のXLインスタンスが全滅? 3分もたたずにか?」

いずれもTwitterから拾ってきたIT技術者の魂の輝きである。後ろ向きな発言になってしまいがちだが、出典を知る方の共感を得られるだろう。意味のない機能追加を要求された際に「あんなの飾りです。偉い人にはそれがわからんのですよ」と切り返すことができれば一人前だ。仕事を任せられるようになれば「認めたくないものだな、自分自身の若さゆえの過ちというものを」と言いたくなる場面も多々あるだろう。社内のプレゼンや、コミュニティの勉強会でこれらのセリフを散りばめたLTを展開すれば、アラフォー世代からは良好な反応が期待できる。

ガンダムで先輩とわかり合える最も簡単な方法は、パチ組み※でもよいので机にガンプラ置くことである。完成済みのフィギュアでも構わないのだがモビルスーツの選択が重要で、素人っぽい媚びた行為は逆効果である。冒頭で例に挙げたグフ・カスタムは、漢(おとこ)の機体として人気が高く先輩の評価が上がること請け合いだ。よくわからなければサザビーや百式などシャアの愛機を選んでおけば間違いは少ない。

評伝シャア・アズナブル
※新書判は絶版のため、データは文庫判のもの

講談社/皆川ゆか著、サンライズ監修/文庫判・544ページ/880円

ぼくには帰れる職場があるんだ

今を生きる若者達に、ガンダムを理解しておいてほしいのは先輩とのコミュニケーションを円滑化できるからというだけの理由ではない。ガンダムで描かれる壮絶な戦闘シーンは、まさにプロジェクト現場の縮図であり、生き延びるヒントがいろいろ詰まっているのである。突然現れる強敵、無理な納期、未経験者ばかりの組織、未熟な指揮官、それでもみなさんは無事にプロジェクトから生還しなくてはならないのである。たとえば、ガンダムには、脱出用に利用可能なコアブロックシステムと教育型コンピュータと呼ばれる戦闘支援AIが搭載されており、パイロットの生存に大きく寄与していた(といわれている)。実際のプロジェクト現場でこれにあたる技術は何か、考えておいたほうがよい。一般的には実行環境を自動管理するクラウドや開発環境、コード管理のしくみなどが皆さんを助けてくれるだろう。デバイスやクラウドを使いこなすニュータイプ素養の高い皆さんに、都合よく最初からガンダムがあてがわれるとは限らない。どんな理不尽な現場からでも逞しく生還する術を身につけてほしい。

この原稿は、『ガンダムユニコーン Episode6』を見ながら書いている。最終決戦を前にフルアーマー装備で発進するユニコーンガンダムを見送るオットー艦長に感情移入しながら、最後にこの言葉を皆さんに贈りたい。

「IT業界人として、一人の大人として。諸君の健闘に期待する!」SD

※ファーストガンダム…すべての原点となったアムロ達の活躍を描くTVシリーズおよびその劇場版を、その後のシリーズ作品と対比して「ファースト」と呼ぶ。SEEDなどの非宇宙世紀シリーズを認めないファースト原理主義もいるので注意。
※ガンダムユニコーン…ジオンと連邦の対立の根源を描く宇宙世紀シリーズ最新作。OVAは全7巻構成で執筆時点での最新はEpisode6。
※ギレンの野望…ガンダムの宇宙世紀を舞台にした戦略級シミュレーションゲーム不朽の名作。初心者用に難易度を下げた『新ギレンの野望』ではなく、少し古いがPSP版の『アクシズの脅威V』がお勧め。

Ⅲ IT業界にある夢と希望と絶望を知っておくために

PROFILE 前佛雅人（ぜんぶつまさと）
Twitter：@zembutsu／URL：http://pocketstudio.jp/
株式会社リンク at+link事業部サービス開発部所属。インフラエンジニア的な仕事がメイン。ホスティングサービスの運用・サポート・保守対応を一筋。最近はソーシャルゲーム系の監視や対応で疲弊する心を、アニメ鑑賞で癒す日々を送る。個人では、クラウド系などのコミュニティ活動にも参加。

この業界のエンジニア界隈では、アニメやコミックなど、アキバ系の文化を趣味としている方が多いように思う。もちろん、すべての方がそうではないが、作品を理解しておけば、勉強会できっと役に立つシーンも出てくる。発表に盛り込まれるネタがわかれば、内容の理解を深める手助けになり、会場のお堅い雰囲気を和らげるスパイスにも。業界用語の基礎知識として理解するだけでなく、さらに自らのネタとして使いこなせれば「新人なのに、こいつ……デキる！」と会場をざわつかせられるはずだ。

ネット世界を仕事場にするなら『攻殻機動隊』でダイブ!

企業のネットが星を覆い、電子や光が駆け巡っても、国家や民族が消えてなくなるほどは情報化されていない近未来の日本。舞台は、公安9課の草薙素子（くさなぎもとこ）少佐ほか、9課のメンバー達とタチコマと呼ばれる人工知能搭載型戦車（アニメ版ではタチコマと呼称。ちなみに原作ではフチコマ）とともに、さまざまな反社会勢力と戦うSF作品。今から20年以上前の作品でありながら、インターネットが普及する前に士郎正宗氏によって構築された本作品の世界観は今も色あせていない。作中で描かれるネットワークとロボットが融合するような技術はまだ発展途上だからだ。

コミック中には「ネットは広大だわ」や「私のゴーストが囁（ささや）くのよ」などの名台詞も多く、これらは勉強会のスライドで使えるくらい汎用性が高い。また、TVシリーズ版でも名台詞が見受けられる。とくにTV版1作目の『STAND ALONE COMPLEX』では、組織としてどうあるべきか考えさせられる台詞も多い。

私にとって攻殻機動隊とは、今の業界に進むきっかけになった作品である。ネットワークが普及した世界において、自分がどのように働き、暮らしていくのか考えさせられた。ビジュアル的な印象としての格好良さだけでなく、「お前にだってゴーストがあるだろ　脳だってついてる　電脳にもアクセスできる」「未来を創れ」など、印象深い言葉も多い。まだまだ発展する未来のために、自分自身がどうしていきたいか、ワクワクしながら考えるきっかけになる作品だ。

攻殻機動隊（1）

講談社／士郎正宗 著／A5判・347ページ／1,020円

IT業界の悲喜こもごもを萌えながら疑似体験『なれる! SE ― 2週間でわかる? SE入門』

新卒社会人の桜坂工兵が、株式会社スルガシステムという小さな開発会社で働き始めるシーンからスタートする。過酷な就職活動の果て、希望を持って入社した開発会社はブラック会社だった……。失望と挫折を味わいながらも、周りのメンバや上司に支えられ、エンジニアとして成長していく物語。表題にSE（システムエ

ンジニア)とあるが、対象はSEだけでなく、SI業界やプログラマのほか、日本のIT業界で働くすべてのエンジニアにとってあてはまるであろう内容だ。

一見、ライトノベル風の軽めの読み物と思われるかもしれない。だが、本書に書かれている内容は決して誇張でもギャグでもない。現場的な視点としては「だいたい合ってる」としか言いようのないほど。これは、作者である夏海公司氏自身が、大学卒業後にシステムエンジニアとして働いているという経歴の持ち主であるからだ。その経験を元に構成された本作品シリーズは、巻を追うごとにさまざまな現場の状況が描かれる。丁寧かつ精緻だ。開発の話だけではなく、無茶な要求を突きつける顧客とのやりとりや、関連企業の交渉など、この業界は技術以外の要素も欠かせないと気がつかされる。正直読んでいて胃が痛くなるシーンも多い。

だが、この作品を通して読めば、どんなに難しい問題が起こったとしても、仲間達と共に困難に立ち向かう自信がつくであろう。本格的な業務に入る前に、業界の様子を知っておくべき作品として、本書はオススメである。

『魔法少女まどか☆マギカ』でエンジニアの魂のありかを考える!?

「僕と契約して、魔法少女になってよ！」という台詞で一世を風靡したアニメ。契約して——というのは、少女の願いを叶えてあげる代償として、魔法少女になって魔女と戦ってもらうことを指す。ヒロイン達は女子中学生。普通なら、ほのぼの魔法少女アニメのはずが、この作品は少し違う。

物語は、一見したところ、鹿目まどかほかヒロインの中学生達が魔女と戦うような勧善懲悪な作品に見える。どこにでもいる普通の女子中学生が契約によって魔法少女になれる。そして、多くの人を救うことができる。けれど少女達は、いずれ絶望として魔女となり、この世に悪をもたらしてしまう宿命を持っているのだった……。

そんな、少女達を導く存在が「キュゥべえ」である。キュゥべえは人間的な感情や常識が一切通用しない。作中で「こんなの絶対おかしいよ！」や「僕と契約して魔法少女になってよ！」など名言が多い。しかも、魔法少女をエンジニアに置き換えても通用するような世界観を持ち合わせているので、台詞を勉強会での発表のネタとして使える個所も多い。気に入った台詞を見つけたら、さりげなくスライドに滑り込ませてみるべきだ。

もっとも、作品としてのメッセージは明快だ。単純な娯楽作品として見ても良し、エンジニアの働き方と重ねて見ても良し。ぜひとも一度は見ておくべきだ。たとえどのような辛い状況に置かれたとしても、ヒロインのように最後まで諦めない心、そして戦う意志は、エンジニアとして見習いたい。そんな勇気の出る作品である。

最後にあえて言おう、**ほむほむは俺の嫁！**であると。SD

魔法少女まどか☆マギカ
※写真はBlu-ray 1巻
©Magica Quartet／Aniplex・Madoka Partners・MBS

アニプレックス／新房昭之 監督、虚淵玄 脚本、蒼樹うめ キャラクター原案、シャフト アニメーション制作／Blu-ray（1巻）7,350円

なれる！ SE ——2週間でわかる？SE入門

アスキー・メディアワークス／夏海公司 著、Ixy イラスト／A6判・328ページ／620円

Ⅳ 現代の魔力を操るエンジニアに贈る本

PROFILE　小飼 弾（こがい だん）　Twitter：@dankogai
blog：404 Blog Not Found／http://blog.livedoor.jp/dankogai/
1969年生まれ。1999～2001年、株式会社オン・ザ・エッヂ（旧：ライブドア）でCTOとして勤務。現在は、執筆、講演、プログラミング、Encode Moduleのメンテナンス、投資などマルチにこなす。著書『弾言』『小飼弾の「仕組み」進化論』『「中卒」でもわかる科学入門』『本を読んだら、自分を読め』など。

物語世界

本作「新世界より」の世界に、コンピュータは一切登場しません。本作の舞台である1000年後の未来の日本、すべての成人は「呪力」と呼ばれる超能力の持ち主であり、我々が文明の利器にやらせていたことは念じるだけでやることができます。それでも足りないことは、バケネズミという、ハダカデバネズミから進化したと思われる知的生命体がやってくれます。

本作の世界における人間とは、現代人にとってはむしろ「神」と呼ぶにふさわしい人々で、実際バケネズミたちは彼らを神として崇め怖れています。たった一人の人間が念じるだけで、コロニーを一瞬にして消滅させることができるのですから、むしろ当然といえます。

よって彼ら人間が最も怖れているのは、人間が人間を攻撃すること。意図せずに呪力を外に漏らすものを業魔、意図的に他者を攻撃するものを悪鬼と彼らは呼んでいますが、この二者は同作の世界では神話化しています。滅多なことで本物の業魔や悪鬼に出くわすことはないからです。

それもそのはず、彼らは自らに愧死（きし）というしくみを組み込みました。他の人間を攻撃しようとしたら、それだけで不快感がこみ上げ、それでも攻撃したら、自らの呪力で死んでしまうのです。「GANTZ」の脳内爆弾のようなものですね。

同胞は敵にならず、そして異邦に対しては無敵。本作の主人公、渡辺早季は、機械も暴力もない神栖66町で仲間達と一緒に育っていきます。諍（いさか）いがありえないはずのこの世界で、瞠目に値する事件など起こりうるのでしょうか？

みどころ

本作のまずすごいところは、すべての成人が超能力者という設定にあります。

超能力というのは、現代フィクションにおいて最もよく登場する小道具とも言えます。むしろそれが登場しない作品を探すほうが難しい。『ドラゴンボール』、『Death Note』、『HUNTER × HUNTER』……「友情、努力、勝利」以上に少年ジャンプに欠かせないのは超能力かもしれません。アニメの世界はさらに顕著で、同世界ではむしろ「魔力」と呼ばれることの多いこの分野は、「魔法少女もの」だけで一大ジャンル。魔法の類いに無縁に思える『機動戦士ガンダム』シリーズですら、ニュータイプという一種の超能力者が登場します。

もちろん超能力者の蔓延は、邦作にとどまりません。『ハリー・ポッター』に『スター・ウォーズ』……まさに「フォースとともにあらんことを」といわんばかりです。

そのこともあってか、超能力発祥の「地」であるサイエンス・フィクションは昨今むしろ超能力を避けています。野尻抱介も小川一水も円城塔も小林泰三も、私の知るかぎり超能力ものは一切手をつけていないのです。

彼らSF作家が超能力を避けるようになった理由は、たとえば少しも科学的でない『とある科学の超電磁砲（レールガン）』を見ればわかります。人智を超越した力であるがゆえに、話がグダグダになりやすいのですね。ボム数無限の撃ちゲーのよ

※リアル魔法…http://blog.livedoor.jp/dankogai/archives/51616088.html

うに、「そのふざけた幻想をぶちのめす」のは最初は気持ちよくてもすぐ飽きてしまうのです。誓約と制約を念能力の枷にした『HUNTER × HUNTER』はさすがです。締め切りは作者の誓約と制約には入ってないようですが:-)

しかし、結局のところ超能力は能力を超えているからこそ超能力なのです。だからこそ、作中では超能力者というのは羨望と嫉妬の対象で、『地球(テラ)へ…』のように抑圧の対象となるか、『超人ロック』のように非超能力者にとって便利で都合のよい存在として描かれるかのどちらかでした。

ところが本作における呪力は、前述のとおり「能力を超えた力」ではもはやないのです。読者にとってFizzBuzzを2分で書くほど当たり前の力なのです。他作品の終着点が、本作の出発点なのです。

しかしそれ以上に、本誌の読者から最も遠そうな本書を本誌の読者が最も読むべき一作としているのは、他者が持ち得ぬ力を当たり前に持っているということがどういうことなのかということを、本作ほど上手に物語った作品がないから。

そう。実は読者のみなさんは、本作における人間に近い存在なのです。読者のみなさんは呪力の代わりにソフトウェア開発力をお持ちです。呪力ほど洗練されてはいなくとも、呪文を唱えたら動くという意味で、プログラミングというのはまさに“リアル魔法※”。にもかかわらず我々は愧死機構に相当する自制を欠いています。ある意味現代は業魔と悪鬼が跋扈(ばっこ)する時代でもあるのです。

しかしすべての人が力と、そしてそれを制御する自制を得さえすれば、平和は訪れ、長らえるのか?

本作で、その答えを見つけてください。SD

新世界より(上・中・下)

講談社／貴志祐介 著／文庫判・(上)482ページ、(中)442ページ、(下)552ページ／(上)760円、(中)710円、(下)830円

推薦図書プレゼント

各出版社様のご厚意により、本特集で取り上げた推薦図書の一部を一挙プレゼント！ふるってご応募ください。応募の方法は、16ページの読者プレゼントと同様に行っていただき、プレゼント番号にこちらの番号をご記入ください。

プレゼント番号※	書名	プレゼント冊数	出版社	著者	参照記事掲載ページ
8	はじめて読む486	2冊	アスキー	蒲池輝尚 著	P.54-55
9	インターネットのカタチ	1冊	オーム社	あきみち、空閑洋平 著	P.50-51
10	マスタリングTCP/IP 入門編 第5版	1冊	オーム社	竹下隆史、村山公保、荒井透、苅田幸雄 著	P.38-39
11	リーダブルコード	2冊	オライリージャパン	ダスティン・ボズウェル、トレバー・フォシェ 著／角征典 訳	P.32-33
12	詳解 Linuxカーネル 第3版	2冊	オライリージャパン	Daniel P.Bovet、Marco Cesati 著／高橋浩和 監修／杉田由美子、清水正明、高杉昌督、平松雅巳、安井隆宏 訳	P.30-31
13	2分以内で仕事は決断しなさい	1冊	かんき出版	吉越浩一郎 著	P.40-41
14	[Web開発者のための]大規模サービス技術入門	2冊	技術評論社	伊藤直也、田中慎司 著	P.48-49
15	プロセッサを支える技術	2冊	技術評論社	Hisa Ando 著	P.54-55
16	増補改訂版イノベーションのジレンマ	2冊	翔泳社	クレイトン・クリステンセン 著／玉田俊平太 監修／伊豆原弓 訳	P.20-21、P.24-25
17	増補改訂版 Java言語で学ぶデザインパターン入門	1冊	ソフトバンククリエイティブ	結城浩 著	P.22-23
18	体系的に学ぶ安全なWebアプリケーションの作り方	1冊	ソフトバンククリエイティブ	徳丸浩 著	P.26-27
19	ドラッカー名著集1　経営者の条件	1冊	ダイヤモンド社	P.F. ドラッカー 著	P.20-21
20	経営者に贈る5つの質問	1冊	ダイヤモンド社	P.F. ドラッカー他 著／上田惇生 訳	P.54-55
21	採用基準	2冊	ダイヤモンド社	伊賀泰代 著	P.52-53
22	新版 問題解決プロフェッショナル	1冊	ダイヤモンド社	齋藤嘉則 著	P.38-39
23	失敗の本質	1冊	中央公論新社	戸部良一、寺本義也、鎌田伸一、杉之尾孝生、村井友秀、野中郁次郎 著	P.52-53
24	理科系の作文技術	1冊	中央公論新社	木下是雄 著	P.20-21、P.24-25
25	小飼弾の「仕組み」進化論	2冊	日本実業出版社	小飼弾 著	P.52-53
26	改訂版 アサーション・トレーニング―さわやかな〈自己表現〉のために	1冊	日本・精神技術研究所	平木典子 著	P.28-29
27	ガロアの生涯―神々の愛でし人(新装版)	2冊	日本評論社	レオポルト・インフェルト 著／市井三郎 訳	P.42-43
28	ファスト&スロー(上)	1冊	早川書房	ダニエル・カーネマン 著／村井章子 訳	P.36-37
29	ファスト&スロー(下)	1冊	早川書房	ダニエル・カーネマン 著／村井章子 訳	P.36-37
30	UNIXネットワークプログラミング 第2版 Vol.1	2冊	ピアソン・エデュケーション	W.リチャード.スティーヴンス 著／篠田陽一 訳	P.42-43
31	計算機プログラムの構造と解釈	2冊	ピアソン・エデュケーション	ジェラルド・ジェイ サスマン、ジュリー・サスマン、ハロルド・エイブルソン 著／和田英一 訳	P.36-37
32	分散システム　第二版	2冊	ピアソン・エデュケーション	アンドリュー・S・タネンバウム、マールティン・ファン・スティーン 著／水野忠則、佐藤文明、鈴木健二、竹中友哉、西山智、峰野博史、宮西洋太郎 訳	P.34-35

※16ページにある読者プレゼント番号の続きからになっています。

第2特集

覚えておきたい、ちゃんと使いたい!

正規表現をマスターしていますか?

正規表現はコマンドライン、エディタ、プログラミング言語などいろいろな場面で使えます。日々、利用しているという読者も多いのではないでしょうか。

しかし、完璧に使いこなすには、正規表現特有の書き方で文字列のパターンを正しく表記できることと、扱うテキストデータや利用環境の特徴をきちんと理解しておくことが大切です。本特集ではそれらのポイントを解説しつつ、処理速度の要である正規表現エンジンについても紹介します。

日常でよく利用するものだからこそ、この機会に正しい使い方を見直してみましょう。

第1章

これだけは覚えておきたい!

誰にでも役立つ正規表現

本章では正規表現の中でも最も基本の部分の解説をします。最初から完全にマスタしようとせずに、ポイントに沿って実際に試してみてください。だんだん何ができるか、役に立つということが判っていただけると思います。

著者:石田 つばさ　ISHIDA Tsubasa

正規表現とは? ～どこで使えるの?～

世の中には、マスタできれば非常に役に立つことがわかっているのだけれど、どうも手を出しにくい、またそれほど難しくはなさそうに見えるにもかかわらず、なぜかなかなかマスタしにくいものがあります。正規表現もその1つではないかと思います。かく言う筆者も、UNIXを使い始めて間もないころからおもしろそうだと正規表現にチャレンジしてみたものの、どうしても思うようにうまく使いこなすことができず、チャレンジと挫折を繰り返すうちに、何とかある程度使いこなせるようになったという経験があります。まず最初に、そんな過去の経験から、「このあたりから少しずつマスタしていくといいのではないか」というポイントを解説していきたいと思います。

正規表現は**「特定の文字列そのものを完全に表現するのではなく、ある条件を満たす文字列を表現するための標準的な記法」**ということになります。これだけだと何のことかまったくイメージできないと思いますので、言葉の意味や歴史などはおいておいて、どういう表現ができてどのように使えるのかを、具体的に説明していくことにします。

正規表現は、もともとの言葉では「Regular Expression」といいます。まさにそのままコマンド名の一部になっているUNIXのgrep(Global Regular Expression Print)コマンドが正規表現をサポートしているコマンドの代表です。そのほかにも、さまざまなコマンドやツールの検索機能などで利用されるようになり、UNIXではsedなどのコマンド、vi、emacsのようなエディタ、そしてプログラミング言語ではPerl、PHP、Java、Ruby、JavaScriptなどでサポートされています。さらに、正規表現はUNIX以外にも実装されるようになり、Windows上でも秀丸エディタなど各種のアプリケーションソフト上で検索機能などで利用可能となっています。

正規表現は、今やUNIXユーザで、特定のコマンドを利用する人だけのものではなく、より多くの人に利用してもらえるものとなっているわけです。

正規表現の機能 ～ワイルドカードとどう違う?～

正規表現の用語

それでは、正規表現はどのような機能を持っていて、どのように使えるのかを説明していきましょう。

正規表現では、指定したい文字列の条件を表現するための「メタキャラクタ」と呼ばれる特殊

な文字を使います。ワイルドカードを使ったことがある人ならば、「そういえばワイルドカードでも“*”とか“?”とかいうメタキャラクタを使ったな」と思い出すのではないかと思います。言葉としては似たようなものと思っていても良いのですが、その働きは異なるのであまり同じものだと思い込み過ぎ、はまってしまわないように注意してください。

そして、メタキャラクタで記述した正規表現の条件に該当する文字列が存在した場合、それを「正規表現にマッチした」といいます。このほかの用語については、説明していく中で解説します。

正規表現のメタキャラクタ

次に、正規表現のメタキャラクタとその記述方法を説明していきましょう。正規表現の代表的なメタキャラクタには、**表1**のようなものが

▼表1　主なメタキャラクタ

メタキャラクタ	機能、意味	備考
.	任意の文字1文字(改行以外、ただし場合によっては改行を含む)	ピリオド
*	直前に指定された文字の0回以上の繰り返し	アスタリスク
+	直前に指定された文字の1回以上の繰り返し	基本正規表現では¥+とエスケープした形で指定する。
?	直前に指定された文字の0回または1回の繰り返し	基本正規表現では¥?とエスケープした形で指定する。
{n}	直前に指定された文字のn回の繰り返し	基本正規表現では{}の代わりに¥{¥}とエスケープした形で指定する
{n,}	直前に指定された文字のn回以上の繰り返し	基本正規表現では{}の代わりに¥{¥}とエスケープした形で指定する
{n,m}	直前に指定された文字のn回以上m回以下の繰り返し	基本正規表現では{}の代わりに¥{¥}とエスケープした形で指定する
[]	“[”と“]”との間に指定された文字のいずれか1文字	
-	その前後に指定された文字の範囲(“[”と“]”との間に指定された場合)	
|	その前後に指定された文字(列)のいずれかの文字(列)	基本正規表現では¥|とエスケープした形で指定する。
^	行頭	
^	否定(続けて指定された文字以外の文字)(“[”に続けて指定された場合)	
$	行末	
()	“(”と“)”とでくくられた正規表現を1つのまとまりとして解釈する	基本正規表現では()の代わりに¥(¥)(エスケープした形)で指定する
¥<	単語の先頭境界	
¥>	単語の末尾境界	
¥b	単語境界の空文字	Perlで文字クラス内で使用する場合はバックスペースを表す
¥d	数字1文字	sedではサポートされない。grepでは-Pオプションでで Perl互換の正規表現を有効にした場合は使用可
¥D	数字以外の文字1文字(¥d以外の文字1文字)	sedではサポートされない。grepでは-Pオプションでで Perl互換の正規表現を有効にした場合は使用可
¥s	空白文字1文字(スペース、改行、タブなど)	grepでは-Pオプションで Perl互換の正規表現を有効にした場合は使用可
¥S	空白文字以外の文字1文字(¥s以外の文字1文字)	grepでは-Pオプションで Perl互換の正規表現を有効にした場合は使用可
¥w	英数字1文字	
¥W	英数字以外の文字1文字(¥w以外の文字1文字)	

あります。

今回は誌面の制約もありすべてのメタキャラクタについて詳細に説明することはできませんが、いくつかのメタキャラクタを例に具体的な正規表現の書き方を見ていくことにしましょう。

基本正規表現と拡張正規表現

正規表現には、「基本正規表現」と「拡張正規表現」があります。現在では拡張正規表現の方が標準であり、基本正規表現は後方互換性のために残されていると考えられます。名称から受ける印象としては、まず基本正規表現が標準としてサポートされていて、拡張正規表現がオプション機能としてサポートされるものであるかのように思えてしまいますが、実際には逆ですので注意してください。

grepコマンドは、grepコマンドとして実行すると基本正規表現が、egrepコマンドとして実行すると（またはgrepコマンドに-Eオプションを指定して実行すると）拡張正規表現が、それぞれサポートされるようになります。基本正規表現は、一部の古いコマンドの実装でサポートされている場合があるので、そのようなコマンドを使う場合には注意してください。

表1に、基本正規表現と拡張正規表現でサポートされるメタキャラクタの有無、機能の相違についても簡単に記載してあるので、参考にしてください。

以降の説明では、拡張正規表現を中心に説明します。

繰り返しの指定

まず、ワイルドカードでも正規表現でもメタキャラクタとなっている“*”です。一見同じものと考えてしまいそうですが、正規表現においては“*”は「直前に指定された文字列の0回以上の繰り返し」と説明されていることに注意してください。ワイルドカードでは“*”はあらゆる文字（列）にマッチします。たとえば“ab*”という指定は、ワイルドカードと正規表現とでは次のように異なる意味になります。

・ワイルドカード

文字列“ab”に続いて何らかの文字が0文字以上何文字か続いている文字列

例：ab、abcde、abbb、abab、ab123

・正規表現

文字aに続いて文字bが0回以上繰り返されている文字列

例：a、ab、abb、abbb

どちらの場合でも、文字列“ab”はマッチしていますが、ワイルドカードの場合は「文字列“ab”に続いて0文字が続いている文字列」としてマッチしているという解釈になるのに対して、正規表現の場合は「文字列“a”に続いて、文字“b”が1回繰り返されている文字列」としてマッチしているという解釈になります。この違いを理解しておかないと最初からつまづいてしまうので、しっかりと理解するようにしてください。

“*”の使い方がわかれば、“+”と“?”は繰り返しの回数の違いだけですから、簡単に理解できるでしょう。

“+”は「直前に指定された文字の1回以上の繰り返し」を表すので、たとえば、正規表現「ab+」は「ab*」と違って文字列“a”にはマッチしなくなりますが、“b”が1回以上繰り返されているab、abb、abbbには同じようにマッチします。

“?”は「直前に指定された文字の0回または1回の繰り返し」を表します。繰り返しといっても2回以上にはマッチしないので、繰り返しというよりも、その直前の文字が一文字だけあってもなくても良いというオプションであることを表しているとも考えられます。たとえば、正規表現「ab?」は文字列a、abのいずれかにマッチします。abb、abcdといった文字列の場合は、その文字列全体にはマッチしないのですが、一部分であるabの部分にマッチします。

繰り返しの回数指定

メタキャラクタ“*”、“+”では、繰り返しの最低の回数は0回または1回と決まっていますが、繰り返しの回数の上限のほうは不定です。これに対して、繰り返しの回数がより厳密に決まっている場合には、メタキャラクタ“{}”を使って具体的な繰り返し回数を指定できます。「繰り返し回数がn回」とピンポイントで決まっているのであれば、「{n}」のように“{}”でくくった中に回数を記述します。「n回以上m回以下」というように、繰り返し回数の上限と下限が決まっているのであれば、「{n,m}」のように“{}”でくくった中に上限下限の数値をカンマで区切って記述します。また、「下限だけが決まっている」のであれば「{n,}」のように下限をカンマの前に記述します。たとえば、「文字“a”に続いて文字“b”が1回以上3回以下繰り返されている文字列」は、正規表現では「ab{1,3}」と記述します。これで文字列ab、abb、abbbにマッチしますが、文字列aやabbbbはbの繰り返し回数が異なるためマッチしなくなります。

それぞれの繰り返しのメタキャラクタを使って、実際にどのような文字列にマッチするかを実行した例を**図1**に挙げておくので、参考にしてください。

任意の文字の指定

さて、ある文字の繰り返しということで、固定で決まった文字を指定する場合を説明しましたが、ワイルドカードに近い指定のしかたで「任意の文字」を指定したいことは多々あります。このような場合にはメタキャラクタ“.”(ピリオド)を使います。“.”は改行を除く任意の文字を表すことができます。たとえば、正規表現「a.」は、「文字aに続けて任意の文字が一文字続く文字列」

▼図1　繰り返しの例

```
[root@genesis list1]# cat test.txt
a
ab
abb
abbb
abcde
abab
ab123
[root@genesis list1]# perl -pe 's/ab*/
MATCHED/g' test.txt
 ↑正規表現「ab*」にマッチした部分を文字列“MATCHED”に置き換え
   てどこが正規表現にマッチしているのかを確認してみる
MATCHED
MATCHED
MATCHED
MATCHED
MATCHEDcde
MATCHEDMATCHED
MATCHED123
[root@genesis list1]# perl -pe 's/ab+/
MATCHED/g' test.txt
 ↑メタキャラクタ“*”の代わりに“+”を指定してみる
a ←今度は「bの0回の繰り返し」にはマッチしなくなった
MATCHED
MATCHED
MATCHED
MATCHEDcde
MATCHEDMATCHED
MATCHED123
[root@genesis list1]# perl -pe 's/ab?/
MATCHED/g' test.txt ←さらに“+”を“?”に変更してみる
MATCHED
MATCHED
MATCHEDb
 ↑今度は「bの2回以上の繰り返し」にはマッチしなくなった
MATCHEDbb
MATCHEDcde
MATCHEDMATCHED
MATCHED123
[root@genesis list1]# perl -pe 's/ab{2,}/
MATCHED/g' test.txt ←メタキャラクタ“{}”を使って2回
a                    以上の繰り返しを指定してみる
ab ←今度は「bの2回未満の繰り返し」にはマッチしなくなった
MATCHED
MATCHED
abcde
abab
ab123
[root@genesis list1]# perl -pe 's/ab{1,2}/
MATCHED/g' test.txt
 ↑1回以上2回以下の繰り返しを指定してみる
a
MATCHED
MATCHED
MATCHEDb ←「bの3回以上の繰り返し」にはマッチしなくなったので
          3個目のbはそのまま表示された
MATCHEDcde
MATCHEDMATCHED
MATCHED123
```

を表し、ab、ac、az、a0、a1、a%などの文字列にマッチします。

ここまで説明してくるとピンときた方もいると思いますが、ワイルドカードの“*”と同じく「0文字以上の任意の文字列」を正規表現で表すには「.*」とすれば良いということになります(**リスト2**)。

文字クラスの指定

いろいろな文字が当てはまる「任意の文字」ということでは条件としてあいまい過ぎて、ある特定の文字の集合(文字クラス)にだけマッチするかどうかを調べたい場合は多々あるでしょう。そのような場合は、任意の文字にマッチする“.”ではなくて文字クラスを記述するメタキャラクタ“[]”を使って、マッチさせたい文字の集合を指定します。正規表現の中で“[]”を指定すると、「その括弧の中に記述された文字の集合のうち、いずれかにマッチする一文字」を意味することになります。

たとえば、正規表現「[abc]」は文字列“abc”にではなくて「文字a、b、cのいずれか一文字」にマッチします。a、b、cという3つの選択肢から選ばせるときに、入力された値が正しいかをチェックするのに、正規表現「[abc]」とマッチするかをチェックする、といった使い方ができます。

特定の文字数文字であれば、“[]”で文字をそのまま並べていけば良いのですが、もっと指定したい文字が増えてくると指定するのはたいへんです。この場合、文字コードで連続しているある範囲の文字を指定するのであれば、その範囲の先頭の文字と最後の文字とを“-”でつないで指定できます。

たとえば、先ほどの「文字a、b、cのいずれか一文字」は範囲指定を使って、「[a-c]」と記述することもできます。

この範囲指定を応用してもう少し指定する範囲を広げると、ある文字種を指定できます。たとえば、「大文字アルファベット」は「[A-Z]」、「小文字アルファベット」は「[a-z]」、数字は「[0-9]」というように指定できます。

また、この範囲指定は複数の範囲指定を続けて記述することで複数の範囲をまとめて指定することもできます。たとえば、「アルファベットすべて(大文字小文字両方)」は「[A-Za-z]」、「英数字」は「[A-Za-z0-9]」というように指定できます。

ここで、文字クラスを記述するときに、文字“-”そのものをその文字クラスに含めるにはどのようにしたら良いのか、という疑問が出てくると思います。このような場合は、“-”は指定したい文字の間ではなくて先頭に指定します。たと

▼図2　任意の文字の指定例

```
[root@genesis list2]# cat test.txt

a
ab
ab13
123
456
@#$%
[root@genesis list2]# grep '.*' test.txt  ←任意の文字の0回以上の繰り返しを指定してみる
     ←0回以上の繰り返しということで空行も含めてすべての行が出力された
a
ab
ab13
123
456
@#$%
[root@genesis list2]# grep 'a.*' test.txt     ←文字“a”に続いて任意の文字が0回以上という条件を指定してみる
a   ←文字“a”が含まれる行だけが出力された
ab
ab13
```

えば、「文字a、c、-の3文字のいずれか一文字」を表す正規表現は、「[-ac]」と記述すれば良いのです(**図3**)。

POSIXブラケット表現

文字クラスを使って「数字」といった文字の種類をまとめて記述する方法を説明しました。このような文字の種類を正規表現で記述するには、もうひとつ「POSIX(Portable Operating System Interface for UNIX)ブラケット表現」を使う方法があります(**表2**)。

ブラケット表現では、具体的などの文字というのを指定していくのではなく、あらかじめ決められた文字クラスに付けられた論理的な名称をブラケット(“[:”と“:]”)でくくって表します。POSIXブラケット表現で定義されている文字クラスの論理名には、表2のようなものがあります。

ここで注意しなければならないのは、ブラケット表現で使われている“[:”、“:]”の“[]”は、正規表現の文字クラスを指定するメタキャラクタではない、ということです。正規表現の記述の中にPOSIXブラケット表現の部分のみを記述すると、単なる文字クラスの指定として解釈されてしまうので、さらに文字クラスを指定する“[]”で

▼表2　POSIXブラケット表現

ブラケット表現	機能、意味
[:alnum:]	英数字
[:alpha:]	英字
[:blank:]	スペースまたはタブ
[:chtrl:]	制御文字
[:digit:]	10進の数字
[:graph:]	空白以外の文字
[:lower:]	英小文字
[:print:]	表示可能な文字
[:punct:]	英記号
[:space:]	スペース、タブ、改行
[:upper:]	英大文字
[:xdigit:]	16進の数字(AからFは大文字小文字両方を含む)

▼図3　文字クラスの指定例

```
[root@genesis list3]# cat test.txt

a
ab
abcde
ab-cde
ab13
123
456
@#$%-
[root@genesis list3]# grep '[0-9]' test.txt
 ↑文字クラスを使って数字を含む行を抽出してみる
ab13 ←英字のみ記号のみの行は出力されない
123
456
[root@genesis list3]# grep '[1-3]' test.txt
 ↑1から3までの数字を含む行を抽出してみる
ab13
123 ←4以上の数字しか無い行も出力されなくなった
[root@genesis list3]# perl -pe 's/[a-c]+/
MATCHED/g' test.txt
 ↑aからcまでの文字の繰り返しの部分にマッチした部分を文字列
“MATCHED”に置き換える

MATCHED
MATCHED
MATCHEDde
MATCHED-MATCHEDde
 ↑文字列“ab”と文字“c”にマッチしたが文字“-”にはマッチしていない
MATCHED13
123
456
@#$%-
[root@genesis list3]# perl -pe 's/[-ac]+/
MATCHED/g' test.txt
 ↑文字“-”を文字クラスの先頭に指定して文字"a"、"c"、"-"に
マッチさせてみる

MATCHED
MATCHEDb ←今度は文字“b”にはマッチしなくなった
MATCHEDbMATCHEDde
MATCHEDbMATCHEDde
 ↑今度は文字列“-c”にもマッチしている
MATCHEDb13
123
456
@#$%MATCHED ←文字"-"だけにもマッチしている
[root@genesis list3]# perl -pe 's/[-a-c]+/
MATCHED/g' test.txt
 ↑文字“a”から“c”と“-”の繰り返しにマッチさせてみる

MATCHED
MATCHED
MATCHEDde
MATCHEDde
 ↑今度は“b”にも“-”にもマッチして文字列“ab-c”がひとまとまりで
マッチしている
MATCHED13
123
456
@#$%MATCHED
```

くくる必要があります。たとえば、「[:alnum:]」とだけ記述するとこれは「文字“a”、“l”、“n”、“u”、“m”、“:”のいずれか一文字」という正規表現になってしまうので、「英数字のうちのいずれか一文字」を表したい場合は「[[:alnum:]]」と記述しなければなりません(**図4**)。

▼図4　POSIXブラケット表現の指定例

```
[root@genesis list4]# cat test.txt

a
ab
abcde
ab-cde
ab13
123
456
@#$%-:
[root@genesis list4]# perl -pe ➘
's/[:digit:]+/MATCHED/g' test.txt
 ↑ブラケット表現の部分のみを記述してみる

a
ab
abcMATCHEDe
ab-cMATCHEDe
ab13
123
456
@#$%-MATCHED
 ↑単なる文字クラスと解釈され文字“d”、“:”にマッチしてしまっている
[root@genesis list4]# perl -pe ➘
's/[[:digit:]]+/MATCHED/g' test.txt
 ↑ブラケット表現を“[]”でくくって記述してみる

a
ab
abcde
ab-cde
abMATCHED ←今度は数字1文字の繰り返しにマッチしている
MATCHED
MATCHED
@#$%-:
```

否定の指定

これまでの説明とは逆に「指定した文字以外の任意の文字」を表したいこともあります。このような場合には、範囲指定の“[”に続けて“^”を指定し、その後に除外したい文字を記述します。たとえば、「&%以外の任意の文字一文字」を表す正規表現は、「[^&%]」となります。

そして、もし文字“^”そのものを“[]”の中で指定したい場合は、先頭以外に記述します。たとえば、「文字^、&、%のいずれか一文字」を表す正規表現は、「[&%^]」となります(**図5**)。

行頭、行末の指定

単にデータの中にどの文字があるかということだけでなく、その文字がデータ中のどの位置にあるのかということをチェックしたい場合があります。このような場合に使えるメタキャラクタとして、“^”、“$”があります。“^”は行頭を、“$”は行末を、それぞれ表します。改行などがなく「行」という形がない一連のデータに対しては、それぞれデータの先頭、データの末尾を表します。“^”は文字クラスの先頭で用いら

▼図5　否定の指定例

```
[root@genesis list5]# cat test.txt
abc!@#$=^890
[root@genesis list5]# perl -pe 's/[a-z0-9]+/MATCHED/g' test.txt
 ↑英小文字と数字にマッチさせてみる
MATCHED!@#$=^MATCHED
[root@genesis list5]# perl -pe 's/[^a-z0-9]+/MATCHED/g' test.txt
 ↑英小文字と数字以外にマッチさせてみる
abcMATCHED890MATCHED
 ↑記号の部分にマッチし(注)、英小文字と数字以外ということで最後の改行にもマッチしている
[root@genesis list5]# perl -pe 's/[a-z0-9^]+/MATCHED/g' test.txt
 ↑否定のメタキャラクタ“^”を文字クラスの先頭以外に指定してみる
MATCHED!@#$=MATCHED
 ↑今度は英小文字と数字と文字“^”にマッチした(“^”は否定のメタキャラクタとは解釈されなくなった)
[root@genesis list5]# perl -pe 's/[^a-z0-9¥n]+/MATCHED/g' test.txt
 ↑英小文字と数字と改行以外にマッチさせてみる
abcMATCHED890
 ↑改行の部分にはマッチしなくなり“MATCHED”"と出力されなくなった
```

※注)“^”にもマッチしていますが、これはあくまで英小文字と数字には含まれていない文字ということでマッチしています。

れると否定の意味を持つメタキャラクタであると説明しましたが、文字クラスの中以外で用いられると行頭を表すという、複数の意味を持つメタキャラクタです。

たとえば、ソースコードや設定ファイルなどで、行の先頭が“#”の行はコメント行であるとして、コメント行を削除して有効なコード(あるいは設定パラメータ)の行だけを抽出したいとします。ここで、単純に「grep -v '#' hoge.txt」などとすると、行の途中に“#”が含まれている行なども削除されてしまいます。このような場合は、“^”を使って「grep -v '^#' hoge.txt」とすれば、「行頭に文字“#”がある」という条件を正規表現でうまく指定でき、望む結果を得ることができます(**図6**)。

選択の指定

複数の候補がある文字、文字列のいずれかにマッチしているかを指定したいという場合があったとします。たとえば、ある文字列を検索したいのだけれども、先頭の一文字だけは大文字、小文字いずれの場合でも良いというような場合です。このような場合には選択のメタキャラクタ“¦”を使います。

たとえば、「文字“a”、文字“b”のいずれか」という正規表現は「a¦b」となります。“¦”は、その前後一文字だけでなく、その前後に記述された正規表現全体の選択の対象として解釈するので、「文字列“linux”、文字列“Linux”のいずれか」という正規表現は、「linux¦Linux」となります。

ここで注意しなければならないのは、さらにその前後に正規表現がある場合です。たとえば、「“SElinux”と“SELinux”のいずれか」という条件を指定しようとして、「『“linux”、“Linux”のいずれか』が『linux¦Linux』なんだから、その前に“SE”と付け加えればいいはず」と考えて、「SElinux¦Linux」としてしまうと、これは「“SElinux”と“Linux”のいずれか」と解釈されてしまい、期待する結果が得られません。

このような場合は、どこからどこまでを“¦”による選択の範囲として解釈するのかを指定するため、その範囲を“()”でくくらなければなりません。たとえば、前述の例でいえば、期待する結果を得るためには、「SE(linux¦Linux)」としなければなりません。また、この“()”の指定をうまく使えば、この正規表現は「SE(l¦L)inux」とさらに短く記述することもできます(**図7**)。

制御文字の指定

英数字や記号のような可読文字だけでなく、条件としてTABなどの制御文字を指定したい場合もあるでしょう。正規表現では、制御文字など特殊な文字を表すエスケープ表現がサポートされています(**表3**)。たとえば、TABコードは“¥t”ですので、「TABで区切られた2組の数字文字列」は正規表現で「[0-9]+¥t[0-9]+」と表すことができます(図5の最後のコマンド実行例を参照)。

メタキャラクタのエスケープ

これらのメタキャラクタを見ていると、「メタ

▼図6　行頭の指定例

```
[root@genesis list6]# cat test.txt
# This line is commented out.
To comment a line out in PERL, the # symbol is used.
Is it OK?
[root@genesis list6]# grep '#' test.txt   ←単に文字“#”を含む行を出力してみる
# This line is commented out.
To comment a line out in PERL, the # symbol is used.   ←行の途中に“#”がある行も出力された
[root@genesis list6]# grep '^#' test.txt   ←正規表現で行頭に“#”がある行のみを出力してみる
# This line is commented out.   ←行の途中に“#”がある行は出力されなくなった
[root@genesis list6]# grep -v '^#' test.txt   ←行頭に“#”がある行以外の行(コメント行以外)を出力してみる
To comment a line out in PERL, the # symbol is used.   ←コメント行以外が出力された
Is it OK?
```

キャラクタとされている文字を、メタキャラクタではなくてその文字そのものとして正規表現で指定したいときはどうするのだろう」と疑問に思うのではないかと思います。たとえば、「1+1」は正規表現だと「文字“1”の1回以上の繰り返しに続いて文字“1”が続く文字列」という意味になり、11、111、111、……といった文字列にマッチするのですが、これを足し算を表す文字列「1+1」そのものとして正規表現で指定したいときはどのようにしたら良いでしょうか。このような場合は、メタキャラクタの前に“¥”を指定することでメタキャラクタとしての機能を打ち消すことができます。これを「エスケープする」といいます。つまり、「1¥+1」と指定することにより、メタキャラクタ“+”の機能が打ち消されて、この正規表現は足し算を表す文字列「1+1」にマッチするようになります(**図8**)。

▼表3　制御文字を表すメタキャラクタ

メタキャラクタ	機能、意味
¥a	ビープ
¥b	バックスペース
¥e	エスケープ(ESC)
¥f	フォームフィード
¥n	改行
¥r	復帰
¥s	空白文字
¥t	(水平)タブ

前方参照

正規表現の中で、正規表現のある部分でマッチした文字列と同じ内容を使ってマッチさせたい場合もあるでしょう。また、文字列を検索置換するときに、置換する文字列の一部に検索条件でマッチした文字列を再利用したい場合もあります。正規表現にはこのような場合に利用可能な「前方参照」と呼ばれる機能があります。

前方参照を利用するには、まず再利用したい部分の正規表現をグループ化のメタキャラクタ“()”でくくります。そして、前方参照でその部分を参照したい場所で“¥n”(ただし、ここで“n”は文字“n”ではなく任意の数字を意味します)と指定します。“()”でくくった部分が複数ある場合は、“(”が現れた順番に¥1、¥2……のように参照できます。

たとえば、「[0-9]¥+[0-9]」は任意の1桁の数の足し算にマッチしますが、「([0-9])¥+¥1」とすると任意の同じ数同士の足し算(1+1、2+2など)にマッチすることになります。検索置換をする場合であれば、検索条件に「([a-z])+¥.text」、置換文字列として「¥1¥.txt」のように指定すると、「任意の英小文字何文字かからなる文字列に文字列“.text”が続く文字列を、“.text”の前の部分でマッチしたのと同じ文字列に文字列“.txt”が続

▼図7　選択の指定例

```
[root@genesis list7]# egrep 'linux|Linux' test.txt ←“linux”または“Linux”を含む行を出力する
linux
Linux
SElinux
SELinux
[root@genesis list7]# egrep 'SElinux|Linux' test.txt ←“SElinux”または“Linux”を含む行を出力する
Linux ←"linux"の行にはマッチせず出力されなくなった
SElinux
SELinux
[root@genesis list7]# egrep 'SE(linux|Linux)' test.txt ←文字列“SE”に続いて“linux”または“Linux”が続く行を出力する
SElinux
SELinux
 ↓文字列“SE”に続いて“l”または“L”が来てさらに文字列“inux”が続く行を出力する
[root@genesis list7]# egrep 'SE(l|L)inux' test.txt
SElinux
SELinux
```

※注) 選択のメタキャラクタ“|”は基本正規表現ではサポートされていないため、この例では拡張正規表現を使うようにegrepコマンドを使っています。

く文字列に置換する」、つまり拡張子.textのファイル名の文字列を、同名のファイルで拡張子を.txtに置換するという条件を指定できるようになります(**図9**)。

最長一致

メタキャラクタ“*”など回数が可変の繰り返しの指定をした場合に、正規表現ではできる限り長く一致する部分をマッチさせようとします。

たとえば、ドメイン名の一部にマッチさせようとして「.+¥.」という正規表現を考えたとします。これは「任意の文字の1回以上の繰り返しに続いてピリオドが指定されている文字列」(2つめのピリオドはエスケープされているので、任意の文字を表すメタキャラクタではなくてピリオドそのものを表します)を表します。では、この正規表現は文字列“www.example.co.jp”のどの部分にマッチするでしょうか。普通に考えると最初にピリオドが現れるところまででマッチするのは“www.”だろうと思うことでしょう。しかし、実際にマッチするのは“www.example.co.”になります。

正規表現は、まず「.+」でマッチできるところまでマッチさせていきます。すると「.+」は任意の文字の繰り返しですので最後の"jp"のところまで進んでしまいます。するとその先には文字が残っていないので、続く「¥.」の部分の正規表現にマッチできなくなります。ここで今度は逆にたどっていってピリオドがあるところまで戻ります。すると“co.jp”のところのピリオドが見つかるので、ここのピリオドのところまでマッチするものとして“www.example.co.”にマッチしてしまうのです。

これは便利なようでいて、正規表現を記述する側の感覚と合わずとまどうポイントでもあります。Perlなどではこの最長一致をキャンセルする機能が用意されており、最長一致させたくない場合は繰り返しのメタキャラクタに続けて“?”を指定します。この機能を使って「.+?¥.」と書き換えれば、この正規表現は“www.example.co.jp”のうちの“www.”の部分にマッチするようになります。

最長一致をキャンセルする機能がない場合は、正規表現の工夫で回避することになります。今

▼図8　メタキャラクタのエスケープの例

```
[root@genesis list8]# cat test.txt
11121+120=11241
[root@genesis list8]# perl -pe 's/1+1/MATCHED/g' test.txt
MATCHED21+120=MATCHED241   ← “1” の2回以上の繰り返しにマッチしている
[root@genesis list8]# perl -pe 's/1¥+1/MATCHED/g' test.txt
                        ↑メタキャラクタ “+” をエスケープして文字 “+” として指定する
1112MATCHED20=11241   ←文字列 “1+1” にマッチするようになった
```

▼図9　前方参照の指定例

```
[root@genesis list9]# cat test.txt
dataa.html
datab.text
datac.txt
[root@genesis list9]# perl -pe 's/([a-z]+)¥.text/MATCHED/g' test.txt
                            ↑英小文字の繰り返しに “.text” が続く文字列にマッチさせ、全体を文字列"MATCHED"に置換する
dataa.html
MATCHED   ←文字列 “datab.text” にマッチして置換された
datac.txt
[root@genesis list9]# perl -pe 's/([a-z]+)¥.text/¥1.txt/g' test.txt
                                        ↑拡張子前の英小文字の繰り返し部分を前方参照して拡張子を “.txt” に置換する
dataa.html
datab.txt   ←拡張子が置換された
datac.txt
```

回の例でいえば、否定を使って「[^.]+¥.」のように記述すると「ピリオド以外の任意の文字の1回以上の繰り返しに続いてピリオドが指定されている文字列」という意味になり、「[^.]+」は最初のピリオドでもうマッチしなくなるので、結果として「[^.]+¥.」は文字列“www.example.co.jp”の“www.”にマッチするようになります(**図10**)。

以上、たいへん駆け足で代表的な正規表現のメタキャラクタの機能とその使い方を簡単に説明しました。ここではすべてのメタキャラクタなどについて詳細に説明することはできませんでしたが、表1に挙げたその他のメタキャラクタについてもいろいろな情報を参照して、正規表現の知識を増やしていってください。

実際に使ってみよう

代表的なメタキャラクタの機能についてはある程度説明してきましたが、文章での説明だけだとなかなかどのように正規表現が使えるのかはイメージしづらいところがあります。次に、実際にコマンドを実行する際の指定でどのように正規表現が使えるのか、指定したい条件をどのようにして正規表現に置き換えていくのか、具体的な例題をもとに説明していきたいと思います。

例題1　ファイル中の空行を削除する

ソフトウェアの設定ファイルなど、見やすさのために空行を挿入してあるファイルから、空行を取り除いて有効な記述がある行のみを取り出したい、ということが時々あります。これをegrepコマンドと正規表現でうまく処理してみましょう。

「空行を取り除く」というのを「空行ではない行を取り出す」と考えると、egrepコマンドの抽出条件として「空行ではない」という条件を指定すれば良いでしょう。空行ではないというのは、「任意の文字が(一文字以上)存在する」と考えら

▼図10　最長一致の例

```
[root@genesis list10]# cat test.txt
www.example.co.jp
www.example.jp
[root@genesis list10]# perl -pe 's/.+¥./MATCHED/' test.txt
                        ↑単に“.”で終わる任意の文字の繰り返しにマッチさせてみる
MATCHEDjp
MATCHEDjp  ←“.”が何個出てきても最後の“.”までマッチしてしまっている
[root@genesis list10]# perl -pe 's/.+?¥./MATCHED/' test.txt
                        ↑繰り返しのメタキャラクタに“?”を付加して最長一致をキャンセルする
MATCHEDexample.co.jp    ←最長一致がキャンセルされて最初の“.”までにだけマッチした
MATCHEDexample.jp
[root@genesis list10]# perl -pe 's/[^.]+¥./MATCHED/' test.txt
                        ↑条件を“.”以外の任意の文字の繰り返しに変更してみる
MATCHEDexample.co.jp    ←最長一致をキャンセルしなくても最初の“.”までにだけマッチさせられた
MATCHEDexample.jp
```

▼図11　例題1：空行を削除する

```
[root@genesis list11]# cat test.txt
#!/bin/bash  ←空行がある
date

[root@genesis list11]# egrep '.+' test.txt  ←任意の文字がある行を出力する
#!/bin/bash
date  ←空行が削除された
[root@genesis list11]# egrep -v '^$' test.txt  ←行頭と行末の間に何も無い行(空行)は出力しない指定に変更してみる
#!/bin/bash
date  ←やはり空行が削除された
```

れるので、これを表す正規表現は「.+」となり、実際にegrepコマンドのオプションに指定して「egrep '.+' test.text」のように実行してみると、空行が削除できることが確認できます。

また、egrepコマンドには「指定した条件にマッチする行を出力しない」という-vオプションがあるので、これを使って条件に空行を指定することもできるでしょう。ここでは行頭と行末を表すメタキャラクタを使って表現できます。「空行」→「その行になんの文字も存在しない」→「行頭に続いて何の文字も存在せずいきなり行末になっている行」というように考えていくと、これは正規表現では「^$」と表せるので、「egrep -v '^$' test.text」のように実行してみると、これでも前の例と同じように空行が削除できることが確認できます(**図11**)。

例題2 郵便番号の書式チェック

次に、郵便番号の書式をチェックするための正規表現を考えてみましょう。

郵便番号は「数字3桁+"-"(ハイフン)+数字4桁」という形式になっています。まず最初に正規表現での「数字3桁」の記述方法を考えてみましょう。「数字」を正規表現で記述するにはいくつかの方法があります。1つは、通常の文字クラスを使う方法で、この場合は「[0-9]」と記述します。POSIXブラケット表現が使える場合には、「[[:digit:]]」と記述することもできます。また、perlなどでは「¥d」と記述することもできます。ここでは、文字クラスを使ってみることにしましょう。

次に「数字3桁」ですので、繰り返しが3回ということで繰り返しの回数指定のメタキャラクタ“{}”を使って「[0-9]{3}」と記述できます。ここまでできればあとは簡単です。ハイフンは文字クラスの中で使うと範囲を指定するメタキャラクタですが、文字クラスの外に記述すれば単なる文字“-”そのものとして解釈されます。次に「数字4桁」は繰り返しの回数の指定を変えればいいだけですから、正規表現で郵便番号を表すには「[0-9]{3}-[0-9]{4}」と記述すれば良いことがわかります。

これだけだとちょっと簡単過ぎるので、もう一工夫だけした書式チェック方法を考えてみることにしましょう。ハイフンはあったほうが見やすいのですが、これがなくても数字7桁が正しく指定されていれば、郵便番号として使えないことはありません。数字3桁に続いてハイフンがないだけであればチェックOKということにして、“-”はあってもなくてもマッチするように正規表現を変更してみましょう。この場合には、0回または1回の繰り返しを表すメタキャラクタ“?”を使い、「[0-9]{3}-?[0-9]{4}」と記述すればいいということはすぐにわかりますね(**図12**)。

この正規表現と変数に入力された文字列とを比較すれば、入力フォームで入力された郵便番号の書式チェックとして使うことができます。SD

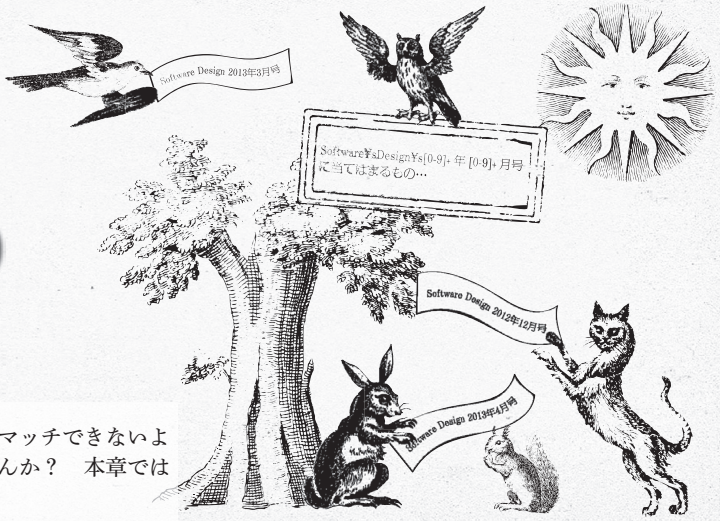
▼図12 例題2:郵便番号形式チェック

```
[root@genesis list12]# cat test.txt
100-0009
100-123
1234-432
10A-123A
1000009
[root@genesis list12]# egrep '[0-9]{3}-[0-9]{4}' test.txt ←正規表現で繰り返し回数を指定する
100-0009 ←郵便番号の形式に合うものだけが出力された
[root@genesis list12]# egrep '[0-9]{3}-?[0-9]{4}' test.txt ←"-"に"?"を付加して有無にかかわらずマッチさせる
100-0009
1000009 ←"-"なしの場合にもマッチして出力された
```

第2章

もう一歩先へ

正規表現の落とし穴



正規表現の意味がわかっていても、思ったようにマッチできないような場面に遭遇し、悩んでしまうことはありませんか？　本章では正規表現で陥りがちな勘違いを取り上げます。

著者：石田 つばさ　ISHIDA Tsubasa

はじめに

正規表現を勉強し始めて、メタキャラクタの意味は理解できたところで、いろいろと実際のデータで試してみると、どうも思ったような条件の文字列にマッチしてくれないとか、思ったとおりにマッチしたかと思うと全然違う文字列にもマッチしてしまって悩む、ということがあります。メタキャラクタの意味を取り違えているような間違えであれば、しっかり見直せばすぐ気づくのでしょうが、どう考えてもメタキャラクタそのものの理解には間違いがなさそうなのにどうしても思うようにいかない、と悩んだあげくに挫折するというパターンは多いのではないでしょうか。

代表格の最長一致については第1章ですでに触れていますが、ほかにどんな落とし穴があるのか代表的なものを説明したいと思います。

実際にマッチしている文字列の勘違い

第1章で、ワイルドカードと正規表現でメタキャラクタ“*”を使った場合の相違の例を紹介しました。この例をもう一度見直してみましょう。

1章の図1の例では、たまたま用意したテストデータが、ワイルドカード、正規表現ともに文字列「ab」と「abbb」にマッチしています。このように実は正しく正規表現が記述できていたわけではなく、ワイルドカードと同じ使い方をしてしまっていたのに、たまたま結果が想定どおりの場合もあります。こんな場合、正しくない正規表現の記述を正しいものと思い込んでしまい、誤った理解をしてしまうことがあります。そして、あとから別のパターンのデータに対して誤った正規表現を適用して、うまくいかずに悩むということになります。このようなこともあるので、正しい正規表現を習得するにはしっかりとした勉強を積み重ねていくしかありません。

また、この例ではワイルドカードでは「abcde」にはマッチしますが正規表現では「abcde」という文字列のうちの「ab」の部分にだけマッチします。この違いは、検索置換した場合や検索時にマッチした文字列部分を反転表示するような機能を持ったエディタだとはっきりわかります。しかしたとえばgrepでは、マッチした文字列が含まれる行全体が抽出（表示）されますから、文字列「abcde」を含む行もそのうちの「ab」にマッチするため抽出されてしまいます（**図1**）。よく考えればわかるのですが、「なんでワイルドカードと違うはずなのに『abcde』も抽出されてしまうのか？」と悩む場合があります。これも、文字列のどの部分がマッチしているのかをよく考えて理解していくようにしましょう。

シェルがメタキャラクタを展開することを忘れずに

第1章で正規表現の使い方の例としてgrepコマンドを取り上げました。grepコマンド以外にも正規表現はいろいろなコマンドでが使え、たとえばfindコマンドでのファイル名の指定にも使えます。しかし、オプションに検索条件として正規表現を記述してこれらのコマンドをコマンドラインから実行してみると、なかなかうまくいかずに悩むことがあります。検索された内容が想定したものと違うというのであれば、正規表現の書き方がまずいのかと考えることができますが、なぜか「no match」と怒られたりして、「『no match』て、確かにこの正規表現でマッチするはずなのに？」と悩むことがあります。

このような場合は、コマンドを実行する際にシェルが関与していることに注意する必要があります。シェルには、ファイル名の展開機能があり、この展開機能が解釈できるメタキャラクタがあると、コマンドを実行する前にそのメタキャラクタをファイル名として展開しようとします。

▼図1　マッチ部分の違いの例

```
[root@genesis list13]# cat test.txt
a
ab
abb
abbb
abcde
abab
ab123          ↓正規表現「ab*」にマッチする行を出力する
[root@genesis list13]# grep 'ab*' test.txt
a
ab
abb
abbb
abcde
↑ワイルドカードと同じように“abcde”にもマッチしているように見える
abab
ab123
[root@genesis list13]# perl -pe 's/ab*/ア
MATCHED/g' test.txt
MATCHED
MATCHED
MATCHED
MATCHED
MATCHEDcde  ←実はマッチしているのは頭の“ab”の部分だけ
MATCHEDMATCHED
MATCHED123
```

bashはデフォルトではファイル名の展開に失敗すると、展開前のコマンドラインの文字列をそのままコマンドに渡すので、たまたまマッチする名前のファイルがないと一見問題なくうまく実行できてしまいます。しかし、たまたまマッチする名前のファイルが存在してしまうと、そのファイル名に展開されたあとの文字列がコマンドに渡されるため、また違った結果になり、混乱してしまうことになります。また、bashのオプション変数でfailglobがonの場合やtcshでは、ファイル名の展開に失敗すると、「no match」というエラーメッセージを出力するという、正規表現自体のエラーとは違う現象が起こるのでますます混乱することになります。

たとえば、1章の図1と同じ正規表現「ab*」をgrepコマンドで指定してみたとします。「grep ab* test.txt」と実行してみると、bashのデフォルトではうまくいきます。しかし、ここで同じディレクトリに「abcfile.txt」のように“ab”から始まるファイル名を持つファイルがあると、今度は何も検索されなくなります。

これは「ab*」がファイル名「abcfile.txt」に展開されてgrepコマンドに渡されてしまい、正規表現「ab*」ではなくて文字列「abcfile.txt」でファイルの中を検索して何もマッチしなかったからです。failglobが設定されている場合はまた違ってきます。abcfile.txtがないとファイル名の展開に失敗するため「no match」とエラーになってコマンドは終了してしまいます。abcfile.txtがあるとファイル名が展開されて、何も検索されなくなります。

このようなシェルによるファイル名の展開の影響を避けるため、正規表現を指定する場合はシェルに展開させずにそのままコマンドに渡すように“'”（クォーテーション）でくくって指定するか、あるいはファイル名の展開のメタキャラクタをエスケープするかする必要があります。実はこれまでの例ではとくに説明せずにコマン

ドに指定する正規表現をクォーテーションでくくっていたのですが、それはこのような意味があるためだったのです(**図2**)。

エスケープの指定ミスに注意

メタキャラクタとされている文字もけっこうありふれた文字ですので、文字そのものとして正規表現の中に含めたいことは多々あります。このような場合はエスケープすれば良いことはすでに説明しました。

エスケープしなければならない文字が一文字しかなければ問題にはならないのですが、ちょっと複雑で長い正規表現を記述しようとすると、何文字もエスケープしなければならなくなります。さらに、"¥t"のようなエスケープシーケンスによる表現もあれば、コマンドによってはメタキャラクタをエスケープした形式で記述するものもあります。

また、viなどで検索条件を指定する場合には、検索条件指定の区切り文字("/"など)が検索したい文字列に含まれていると、この区切り文字もエスケープしなければなりません。

あれやこれやで正規表現を使っていると、エスケープ文字"¥"を多用しなければならない局面に遭遇します。すると、エスケープ文字"¥"を指定するところを間違えて、思った文字列が検索置換できない、ということが起こります。

この問題に対しては、とにかく注意深くエスケープしている個所が正しいかよく考え確認するしかありません。何ヵ所もエスケープしている正規表現を記述して、思うようにマッチしないという場合には、正しくエスケープの指定が記述できているかよく見直してみてください。

また、viなどでは区切り文字は"/"固定ではなく、別に文字を区切り文字として指定できます。デフォルトの区切り文字が検索したい文字列に含まれているような場合は、区切り文字を変更することによりわずかでもエスケープする個所を減らして正規表現を見やすくする工夫をするのも良い方法でしょう。

▼図2　ファイル名の展開例

```
[root@genesis list14]# cat test.txt
a
ab
abb
abbb    ↓正規表現をそのまま記述してもうまくマッチする
[root@genesis list14]# grep ab* test.txt
a
ab
abb
abbb            ↓ファイルabcfile.txtを作成する
[root@genesis list14]# touch abcfile.txt
↓すると同じ正規表現で何もマッチしなくなる
[root@genesis list14]# grep ab* test.txt
↓実は正規表現(のつもり)の部分がファイル名に展開されている
[root@genesis list14]# echo grep ab* test.txt
grep abcfile.txt test.txt
↓正規表現を"'"でくくるとうまくマッチする
[root@genesis list14]# grep 'ab*' test.txt
a
ab
abb
abbb
↓ファイル名の展開のメタキャラクタをエスケープしてもうまくマッチする
[root@genesis list14]# grep ab¥* test.txt
a
ab
abb
abbb
[root@genesis list14]# rm abcfile.txt
rm: remove regular empty file `abcfile.txt'?
yes
[root@genesis list14]# shopt failglob
failglob        off     ↓failglobをonにする
[root@genesis list14]# shopt -s failglob
[root@genesis list14]# shopt failglob
failglob        on
[root@genesis list14]# ls
index.html  test.txt  test1.txt  test2.txt
test3.html          ↓再度正規表現をそのまま記述してみる
[root@genesis list14]# grep ab* test.txt
-bash: no match: ab*  ←今度はエラーになった
[root@genesis list14]# touch abcfile.txt
↑ファイルabcfile.txtがあるとやはり何もマッチしなくなる
[root@genesis list14]# grep ab* test.txt
[root@genesis list14]# echo grep ab* test.txt
grep abcfile.txt test.txt
↑やはり正規表現(のつもり)の部分がファイル名に展開されている
↓この場合でも正規表現を"'"でくくるとうまくマッチする
[root@genesis list14]# grep 'ab*' test.txt
a
ab
abb
abbb
```

コマンドによるサポートの相違

第1章でコマンドによって基本正規表現と拡張正規表現とどちらをサポートしているか異なる場合があることに触れました。また、1章の表1の備考にも書いておきましたが、正規表現はコマンドなどによってサポートしているメタキャラクタのサポートの有無、その意味が異なっている場合があります。

たとえば、ある文字列のグループ化を指定するメタキャラクタは一般的には“(”と“)”ですが、sed、viなどではエスケープ表現のように前に“¥”を付加した“¥(”、“¥)”がグループ化のメタキャラクタとなっています。また、メタキャラクタ“¥d”、“¥s”などはPerlではサポートされているのですが、grepでは-Pオプションで Perl互換の正規表現を有効にしないと使用できません。

このため、ある正規表現がうまく使えているからと、その正規表現をそのまま別のコマンドや言語で実行してみると、想定した条件でマッチしなかったりエラーになってしまったりすることがあります。異なる環境で正規表現を使うときには、もう一度その環境の正規表現で各メタキャラクタが同じくサポートされているのかを確認しておく必要があります。

範囲指定は文字コードに依存する

メタキャラクタ“[]”を使うと、ある範囲の複数の文字を指定できるということを第1章で説明しました。このとき、指定したい範囲の先頭の文字と最後の文字を指定することによって、その間にある文字すべてを表すことができるのですが、その先頭から最後までというのは文字コードの並び順に依存するということを忘れないようにしましょう。

変な例ですが、「[A-Z]」でも「[Z-A]」でも人間だと「大文字のアルファベットを表したいのだな」と考えることができるかもしれませんが、正規表現としてはまったく意味が違ってしまいます。ASCIIコード体系では「[A-Z]」はまさに大文字アルファベットを表しますが、「[Z-A]」ではまったく意味をなさない(該当する文字がない)ことになってしまいます。

情報処理の世界では「ゼロオリジン(数え始めをゼロとすること)」は常識的なことですので、「数字10文字」というと「0から9まで」と出てくる人が多いと思いますが、ゼロオリジンではない自然数の並びで考えて「1から0まで」と考えてしまう人もいます。これをそのまま正規表現に当てはめて、数字という範囲を「[1-0]」と記述してしまうと、これまた意味をなさないことになってしまいます。

ASCIIコード体系ではアルファベットは連続したコードが割り当てられているので「[A-Z]」「[a-z]」で大文字、小文字のアルファベットを表せますが、EBCDICコード体系ではアルファベットも非連続でコードが割り当てられているので、もしEBCDICコード体系上で正規表現が使用できる実装があったとすると、大文字アルファベットは「[A-IJ-RS-Z]」、小文字アルファベットは「[a-hi-pqrst-z]」のようにしなければ文字クラスでは表せません。

異なるコード体系でのポータビリティーを意識した正規表現を書くのであれば、もし対象となる各コード体系上のシステムでPOSIXブラケット表現がサポートされているのであればPOSIXブラケット表現を用いるのも良い方法でしょう。

◆◆◆

第1章と本章で正規表現の大まかな使い方と、陥りがちな問題について説明しました。正規表現は追求していくとまだまだ深い世界があり、またメタキャラクタの使い方を工夫するといろいろな記述方法、応用が考えられるので、今回の説明を参考にどんどんと正規表現にチャレンジしていってもらいたいと思います。SD

第3章

JavaScriptで学ぶ

プログラミングでも威力を発揮する正規表現

プログラムの中で、テキストを扱う機会も非常に多いです。そこで正規表現が使えると、チェック、変換、パースなどのテキスト処理が効率的に行えます。本章ではJavaScriptを例に、プログラミングでの正規表現の使い方を紹介します。

藤本 壱　FUJIMOTO Hajime

JavaScriptの正規表現の基本

プログラム言語は多数あり、それらの大半が正規表現をサポートしています。もちろん、JavaScriptも正規表現に対応していて、文字列の置換／検索を行う際に正規表現を使うことができます。

JavaScriptでの正規表現の表し方

JavaScriptでは、正規表現は「RegExp」というオブジェクトとして扱います。正規表現を生成する書き方は次の2通りあり、どちらを使っても動作は同じです。②のほうがシンプルに書けるので、②を使う機会が多いと思います。

```
①new RegExp('パターン', 'フラグ')
②/パターン/フラグ
```

「パターン」には、ほかの言語と同様の書き方で、マッチさせたいパターンを指定します。また、「フラグ」には、**表1**のいずれかの文字を指定できます。フラグを使わない場合は、指定を省略することもできます。

たとえば、次のような条件で文字列の検索／置換を行いたいとします。

①1文字目がアルファベット
②2文字目以降がアルファベットか数字（2文字目以降はなくても良い）
③アルファベットの大文字／小文字は区別しない

この場合の正規表現は、次のいずれかの書き方で表せます。

```
①new RegExp('[a-z][a-z0-9]*', 'i')
②/[a-z][a-z0-9]*/i
```

正規表現を使った検索

JavaScriptで正規表現を使って文字列を検索する場合、文字列に対して「match」というメソッドを実行することが多いです。書き方は次のようになります。

```
文字列.match(正規表現)
```

文字列が正規表現にマッチした場合、戻り値はマッチした部分を表す配列になります。一方、マッチしなかった場合はnullになります。

パターンの中で括弧を使って、その部分をキャプチャすることもできます。この場合、パターンにマッチすると、結果の配列の内容は**表2**の

▼表1　フラグの種類

フラグ	内容
i	大文字と小文字を区別しない
g	複数回マッチさせる
m	「^」「$」を文字列の各行の先頭／最後にもマッチさせる

ようになります。

たとえば、変数strに「0123-45678-910」という文字列が代入されているとします。この状態で次のmatchメソッドを実行すると、変数mは**表3**のような内容の配列になります。

```
m = str.match(/(¥d+)-(¥d+)-(¥d+)/);
```

また、matchメソッドの実行後に、RegExpオブジェクトのプロパティから、マッチ結果に関する情報を得ることもできます(**表4**)。

なお、検索対象文字列の中で、パターンがマッチした位置を調べたい場合は、matchメソッドの代わりに、「search」というメソッドを使います。マッチすればその位置が戻り値になり(文字列の先頭を0としてカウント)、マッチしなかった場合は-1が戻り値になります。

正規表現を使った置換

正規表現にマッチする部分を、ほかの文字列に置換することもできます。それには「replace」というメソッドを使います。書き方は次の通りです。

```
文字列.replace(正規表現, 置換先文字列)
```

パターン内でキャプチャした部分を、置換先文字列に使うこともできます。その場合、1番目にキャプチャした部分を「**$1**」で表し、以下同様に2番目は「**$2**」、3番目は「**$3**」……のように表します。

たとえば、文字列の中で、数字が1文字以上連続する部分を、角括弧で囲みたいとします。この場合、次のように書きます。

```
文字列.replace(/([0-9]+)/g, '[$1]')
```

関数での置換

正規表現による置換では、置換先の文字列を関数で指定することもできます。この場合、replace関数の2つ目のパラメータとして、置換先の文字列ではなく、置換を行う関数を渡します。

置換用の関数には、パラメータとして、パターンにマッチした文字列が渡されます。その文字列を関数で処理して、結果を戻り値として返します。すると、パターンにマッチした部分が戻り値の文字列に置換されます。

また、正規表現のパターンの中でキャプチャを使うと、置換用の関数には、パラメータとして、キャプチャにマッチした文字列も渡されます。

関数で置換するしくみを使うと、通常の正規表現では表せないような置換も行えます。具体的な例は、あとの節で紹介します。

正規表現での文字列の分割

文字列をコンマやタブなどで区切って、配列に代入して扱いたい場面も多いです。このようなときには、「split」というメソッド使います。

たとえば、次の文を実行すると、変数arは「apple」「grape」「banana」の3つの要素からなる配列になります。

```
ar = 'apple,grape,banana'.split(/,/)
```

▼表2　結果の内容の配列

番号	内容
0	マッチした部分全体
1	1番目のキャプチャにマッチした部分
2	2番目のキャプチャにマッチした部分
⋮	⋮

▼表3　matchメソッドの結果の配列の例

番号	内容
0	0123-45678-910
1	0123
2	45678
3	910

▼表4　RegExpオブジェクトのプロパティ

プロパティ	内容
$1～$9	1番目～9番目のキャプチャにマッチした部分
lastMatch	マッチした部分全体
leftContext	対象文字列の中で、マッチした部分よりも前の部分
rightContext	対象文字列の中で、マッチした部分よりも後の部分

JavaScriptの正規表現とPCREとの違い

前述したように、現在の多くの言語は正規表現に対応しています。中でも、「Perl互換の正規表現」(PCRE:Perl Compatible Regular Expressions)を採用している言語が多いです。

Perlもプログラム言語の一種で、かつてのインターネット普及初期には、CGIを書くためによく使われていました。そのため、Perlの正規表現が事実上の標準のようになっています。

ただ、各言語の正規表現がPerlと100%同じ仕様になっているかというと、そうではありません。言語によって細かな違いがあります。JavaScriptの正規表現も、Perlの正規表現に近いですが、まったく同じではありません。主な違いは次のような点です。

フラグの違い

Perlの正規表現では、フラグの種類が多くあります。一方、JavaScriptの正規表現では、フラグは前述したように3種類しかありません。PerlにあってJavaScriptにないフラグとしては、**表5**のようなものがあります。

ちなみに、Web系のプログラミングではPHPを使う機会も多いです。PHPにもPCREの正規表現処理関数があり、「s」や「x」のフラグにも対応しています。この点を忘れていて、JavaScriptでうっかり「s」などのフラグを使おうとしてしまうことも少なくありませんので、注意が必要です。

なお、JavaScriptで「改行を含むすべての文字」にマッチさせたい場合、「**.**」の代わりに「**[¥s¥S]**」を使う方法があります。「**¥s**」はホワイトスペースで、「**¥S**」はそれ以外ですので、そのどちらか(**[]**)にマッチさせると、改行を含むすべての文字にマッチします。

▼表5　JavaScriptにはないフラグ

フラグ	内容
s	「**.**」のメタキャラクタを改行にもマッチさせる
x	パターン内にスペースを入れて読みやすくできる
o	パターンを1回だけ評価する

エスケープシーケンス(メタキャラクタ)の違い

パターンの中で特殊な文字を表す際には、「**¥n**」などのエスケープシーケンス(メタキャラクタ)を使います。JavaScriptのエスケープシーケンスは、Perlのエスケープシーケンスと比べて種類が少ないです。

PerlにあってJavaScriptにはないエスケープシーケンスとして、**表6**のようなものがあります。

拡張正規表現

Perlの正規表現では、パターン内で「拡張正規表現」を使えます。拡張正規表現は「**(?**」から始まるパターンで、コメント/名前付きキャプチャ/動的正規表現/条件/再帰などの機能を使えます。

しかし、JavaScriptの正規表現では、これらの拡張正規表現には対応していません。

JavaScriptでの正規表現の事例

JavaScriptはプログラム言語ですので、エディ

▼表6　JavaScriptにはないエスケープシーケンス

エスケープシーケンス	内容
¥a	ベル(16進数で07)
¥e	エスケープ文字(16進数で1B)
¥l	次の文字を小文字にする
¥u	次の文字を大文字にする
¥L……¥E	¥Lと¥Eの間を小文字にする
¥U……¥E	¥Uと¥Eの間を大文字にする
¥Q……¥E	¥Qと¥Eの間に含まれるメタ文字を通常の文字として認識させる
¥A	文字列の先頭
¥Z	文字列の末尾(省略可能な改行の前)
¥z	文字列の末尾
¥G	前回のgフラグ付きマッチの末尾

タなどでの単純な正規表現とは異なり、より幅広い処理に使えます。ここでは事例として、「全角英数字を半角英数字に変換」と、「HTMLのパース」を取り上げます。

全角英数字を半角英数字に変換する

会員登録などのフォームで、住所や電話番号の入力の際に、「英数字は半角で入力してください」といった指示を見かけることがよくあります。そして、その指示に従わずに全角で入力すると、再入力させられることも多いです。しかし、ユーザに半角文字を再入力させるのではなく、システム側で自動的に半角に置換する方が、ユーザにとって使いやすいです。

JavaScriptには、全角英数字を半角に置換するような関数は標準装備されていません。しかし、正規表現を活用すれば、全角英数字を半角に置換することは、そう難しくはありません。

個々の文字には、「文字コード」という番号が付けられています。文字列中に全角の英数字が含まれている場合、その文字コードから、対応する半角英数字の文字コードを計算で求めることで、半角文字に置換できます。

JavaScriptでは、文字列を内部的にUnicodeで扱っています。全角英数字の文字コードは、Unicodeの65296番(全角の0)～65370番(全角のz)に連続的に割り当てられています。一方、半角英数字の文字コードは、48番(半角の0)～122番(全角のz)に割り当てられています。

そこで、文字列中の個々の全角英数字に対して、次の処理を行うことで、半角英数字に置換します。

①個々の全角英数字の文字コードを求めます。
②①の文字コードから、全角の「0」の文字コードと半角の「0」の文字コードの差を引きます。
③②の文字コードから、全角英数字に対応する半角文字を得ます。

関数の実際のコードは、**リスト1**のようになります。3行目のreplaceメソッドで、文字列内に含まれる全角英数字(**[０-９Ａ-Ｚａ-ｚ]**)にマッチしたときに、置換の関数を実行しています。

置換の関数の中では、上に挙げた手順のとおりに、元の全角文字のコードから対応する半角文字のコードを求め(4～5行目)、そこから半角文字を得ています(6行目)。

フォームのsubmitイベントのハンドラなどで、この関数を使って全角英数字から半角英数字に置換すると良いでしょう(**図1**)。また、リスト1のconvZenHan関数をjQueryのプラグインにして、jQuery(セレクタ).convZenHan()のように書けるようにしておくのも良いでしょう(**リスト2**)。

なお、リスト1の3行目を次のように書き換えることで、英数字だけでなく、記号も変換できます。

▼リスト1　全角英数字を半角英数字に変換する関数

```
function convZenHan(str) {
  var diff = '０'.charCodeAt(0) - '0'.charCodeAt(0);
  str = str.replace(/[０-９Ａ-Ｚａ-ｚ]/g, function(c) {
    var code = c.charCodeAt(0);
    code -= diff;
    return String.fromCharCode(code);
  });
  return str;
}
```

▼図1　リスト1の関数を使って全角英数字を半角に置換した例

変換前
０１２３４５６７８９ＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯＰＱＲＳＴＵＶＷＸＹＺａｂｃｄｅｆｇｈｉｊｋｌｍｎｏｐｑｒｓｔｕｖｗｘｙｚ

変換後
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

変換

```
str = str.replace(/[！-～]/g, function(c) {
```

HTMLをパースする

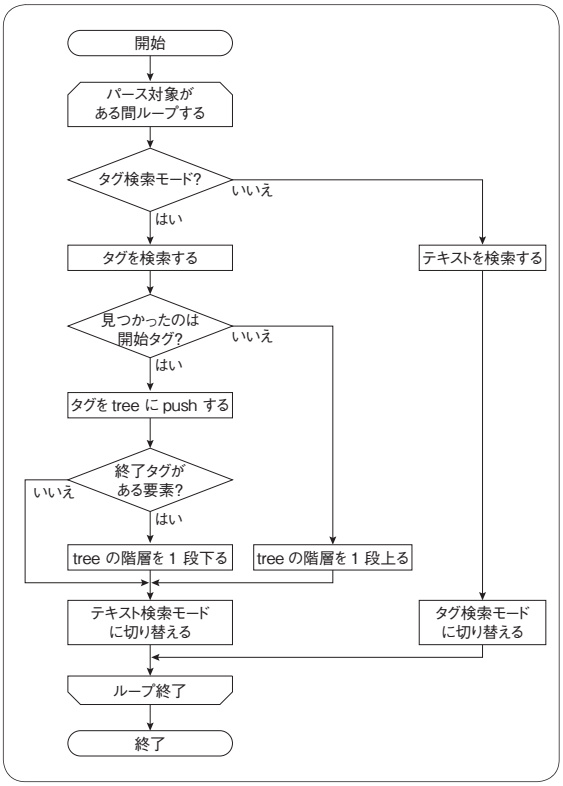
正規表現は、各種の構文解析を行う際にも威力を発揮します。その例として、HTMLをパース(構文解析)して、その中にある要素を得る例を紹介します。

次の条件を満たす場合、HTMLをパースする手順を流れ図で表すと、**図2**のようになります。

①HTMLが文法的に正しい(タグの対応が正しく、タグに関係ないところで「<」や「>」が登場しない)
②img要素など、終了タグがない要素では、タグの最後を「/>」にしている(例：)

HTML内のタグを検索する際に、正規表現を活用します。開始タグは、「『<』の文字→タグ名→属性→『>』の文字」の流れになります。また、終了タグは、「『</』の文字→タグ名→『>』の文字」の流れです。これら両方にマッチ

▼図2　HTMLをパースする手順



▼リスト2　convZenHan関数をjQueryのプラグインとして使えるようにしたもの

```
(function($) {
  $.fn.convZenHan = function() {
    var diff = '０'.charCodeAt(0) - '0'.charCodeAt(0);
    $(this).each(function() {
      var str = $(this).val();
      str = str.replace(/[０-９Ａ-Ｚａ-ｚ]/g, function(c) {
        var code = c.charCodeAt(0);
        code -= diff;
        return String.fromCharCode(code);
      });
      $(this).val(str);
    });
  }
})(jQuery);
```

▼表7　「<((/?)[a-z][a-z0-9]*)¥s?([^>]*?)>」の正規表現の意味

部分	意味
/?	「/」の文字が0回または1回(終了タグの「/」があるかどうかを判断するために利用)
[a-z][a-z0-9]*	1文字目がアルファベットで、2文字目以降がアルファベットか数字(「h1」などの数字を含むタグも検索するため)
¥s?	0文字または1文字のホワイトスペース
[^>]*?	「>」以外の任意の文字の連続(要素の属性を検索するために利用)

▼リスト3　HTMLをパースする関数

```
function parse(html) {
  var re_tag = new RegExp('<((/?)[a-z][a-z0-9]*)¥s?([^>]*?)>', 'i');
  var SEARCH_TAG = 1, SEARCH_TEXT = 2;
  var mode = SEARCH_TAG, tag, attr, found_text;
  var tree = [], treeStack = [];

  // html要素の外にある部分(DOCTYPE宣言等)を削除
  html = html.replace(/^.*(<html)/i, '$1');
  html = html.replace(/(<¥/html>).*$/i, '$1');
  // すべてをパースし終わるまで繰り返す
  while (html.length) {
    // タグ検索モードの場合
    if (mode == SEARCH_TAG) {
      // タグを見つける
      if (html.match(re_tag)) {
        // タグ／属性／残りのHTMLを得る
        tag = RegExp.$1;
        isStartTag = (RegExp.$2 != '/');
        attr = RegExp.$3;
        html = RegExp.rightContext;
        // 開始タグの場合
        if (isStartTag) {
          // ツリーにタグをpushする
          tree.push({ tag: tag, attr: attr, nodes: [] });
          // 対になる終了タグがあれば、
          // treeの階層を一段下る
          if (!attr.match(/¥/$/)) {
            treeStack.push(tree);
            tree = tree[tree.length - 1].nodes;
          }
        }
        // 終了タグの場合
        else {
          // treeの階層を一段上る
          tree = treeStack.pop();
        }
        // テキスト検索モードに切り替える
        mode = SEARCH_TEXT;
      }
    }
    // テキスト検索モードの場合
    else if (mode == SEARCH_TEXT) {
      // 次のタグまでの部分を検索する
      if (html.match(/([^<]*)/)) {
        // 見つかったテキストをtreeにpushする
        found_text = RegExp.$1;
        if (found_text.length) {
          html = RegExp.rightContext;
        }
        tree.push({ tag: 'TEXT', text: found_text });
        // タグ検索モードに切り替える
        mode = SEARCH_TAG;
      }
    }
  }
  return { nodes: tree };
}
```

するパターンを作って、タグを検索するようにします。

図2の流れを元にJavaScriptを組むと、**リスト3**のようになります。2行目にある次の正規表現が、HTMLのタグを表します。

```
<((/?)[a-z][a-z0-9]*)¥s?([^>]*?)>
```

一見わかりにくい正規表現ですが、部分ごとに分解すると、**表7**のような内容になっています。

図3はリスト3の関数を利用してHTMLをパースし、見つかったHTMLのタグのツリー構造を表示した例です。フォーム入力したHTMLから、正しいツリー構造が得られていることがわかります。

まとめ

正規表現はそれ単体でも強力で便利なものですが、プログラムと組み合わせることで、より威力を発揮します。本章ではそのごく一部を紹介しました。ほかにもさまざまな活用が考えられますので、みなさんも正規表現をプログラムの中で使ってみてください。SD

▼図3　リスト3の関数を利用してHTMLをパースした結果

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTMLの例</title>
  </head>
  <body>
    <section>
      <h1>Software Design</h1>
      <p><a href="http://gihyo.jp/magazine/SD">Software Design</a>は絶賛発売中です。</p>
      <p><img src="http://image.gihyo.co.jp/assets/images/cover/2013/641303.jpg" /></p>
    </section>
  </body>
</html>
```

パース

- html
  - head
    - title
  - body
    - section
      - h1
      - p
        - a
      - p
        - img

第4章

処理速度にも影響する!?

正規表現エンジンの種類としくみ

本章では、正規表現のマッチングのしくみについて、実装の話も交えつつ詳しく見ていきます。しくみがわかれば、正規表現を書く際に役立つだけでなく、処理速度を考慮して利用できるようになります。

東京工業大学
新屋 良磨　Sin'ya Ryoma
Twitter @sinya8282

可能性のたどり方——マッチしているのかどうか

簡単な正規表現の例から始めてみましょう。なお、本章では正規表現は常にスラッシュ"/"で囲んでいます。

```
/[ab]*a[ab][ab]/
```

は、「aかbで作られた文字列で、最後から3文字目がaである文字列」に**完全一致**します。文字列が正規表現**/regex/**に「完全一致」するというのは、文字列が**/^regex$/**にマッチすることを言います。「行頭から行末まで完全に一致」というわけですね。完全一致以外にも、**/^regex.*/**にマッチする**前方一致**、**/.*regex$/**にマッチする**後方一致**、**/.*regex.*/**にマッチする**部分一致**があります。

話を**/[ab]*a[ab][ab]/**に戻します。この正規表現に対して、文字列baabaを入力するとどう判定されるのでしょうか？　完全一致することは明らかですが、「先頭から1文字ずつ文字列を読める」という前提で、マッチングを考えてみましょう。

・1文字目 b

先頭1文字目bが入力された時点では、完全一致する気配はありません。なぜなら、今考えている正規表現にとってはaがどこにあるのかが重要なのですから。

・2文字目 a

期待していた文字aが入力されました。どうやら期待できそうです。

・3文字目 a

ここまで来ると次の入力が気になります。なぜなら現段階でbaaまでわかっており、次の入力がbかaならば、現時点での真ん中のaが「最後から3文字目のa」となるためです。

・4文字目 b

これでめでたく完全一致が決定しました！

```
   b   a   ab
/[ab]*  a  [ab][ab]/
```

のように、文字列が正規表現に完全一致することがわかります。

……しかし喜ぶも束の間、どうやら入力に続きがあることが発覚しました。

・5文字目 a

どうやら先ほどまでの解釈は間違っていたようです。解釈をリセットし、もう一度3文字前を見直すことで、

```
   ba  a  ba
/[ab]*  a  [ab][ab]/
```

という形で完全一致することがわかりました。

このように、正規表現によるマッチングでは「どのようにマッチするのか」という複数の可能

性を考慮しなければなりません。

現代の正規表現エンジン実装たちは、どのようにしてマッチングを行っているのでしょうか？

ここでは、最も実装が多い「DFA型」と「VM型」の2つを紹介したいと思います。

DFA型——可能性を幅優先でたどる

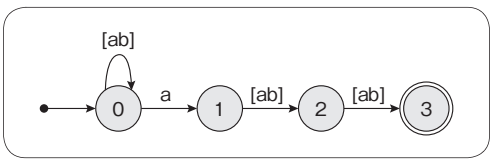
実は、正規表現は常に「とある有限状態の計算モデル」でマッチングを行えることが知られています。その計算モデルが、次に紹介する**有限オートマトン**です。

有限オートマトン

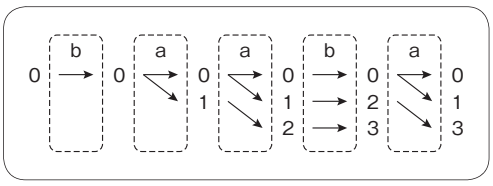
天下り的になりますが、**図1**が正規表現**/[ab]*a[ab][ab]/**に対応する有限オートマトンです。なお、正規表現から有限オートマトンへの変換は簡単に(正規表現の大きさに対して線形時間で)行えます。

図1の有限オートマトンが何を表しているのかを説明します。まず、番号の振られた円はそれぞれ状態を、文字が上についた矢印は遷移規則を表しています([ab]という遷移規則はaとbの2つの遷移規則をまとめたものです)。状態の中でもとくに重要なのが**初期状態**と**受理状態**と呼ばれるものです。ここでは初期状態は「黒点始端の矢印で指された状態」を、受理状態は「二重円の状態」で表すことにします。図1の初期状態は状態0で、受理状態は3です。

▼図1　/[ab]*a[ab][ab]/に対応する有限オートマトン(NFA)



▼図2　図1の有限オートマトンでbaabaをシミュレーション



簡潔に言うと、「入力文字に対して、初期状態から遷移規則を適用した結果、受理状態に至るかどうか」をシミュレーションすることで、有限オートマトンを使った正規表現マッチングが行えます。百聞は一見にしかず、最初の例を図1によってシミュレーションしたものが**図2**です。

初期状態0から始めて、文字列baabaを読むことで状態0,1,3に遷移します。3は受理状態のため、文字列baabaは「受理＝正規表現に完全一致」というしくみです。

状態0から文字aを読むと、状態0と状態1に遷移が分岐するわけですが、このように遷移先が複数ある遷移規則を**非決定な遷移規則**と呼びます。さらに、非決定な遷移規則を持つ有限オートマトンを**非決定性有限オートマトン(NFA)**と呼びます。図1の有限オートマトンはNFAになります。

NFAからDFAを作る——可能性を全列挙

非決定的な遷移はシミュレーション的には少々厄介です。遷移の分岐がある限り、遷移先は複数ありえるので「現在の状態」を集合的に扱う必要があります。プログラムに起こす場合は、「遷移状態の集合」をリストや配列などのデータ構造で格納しておき、入力文字を読むたびに状態のデータ構造を舐めるといった処理が必要になるでしょう。

しかし、ちょっとした発想の転換で有限オートマトンから非決定性をなくすことが可能です。「遷移状態の集合」そのものを状態と見てやれば良いのです。先ほどの例では状態0から文字aを読むと「状態0と状態1」に遷移しました。この「状態0と状態1」という「遷移状態の集合」を1つの状態とみるのです。非常に単純な発想ですが、実際これでうまく非決定性をなくすことができます。この「遷移状態の集合を1つの状態とみる」という構成法は**部分集合構成法**と呼ばれています。

たとえば、(**図3**)は図1からこの手続で非決定な遷移規則を取り除いた決定性オートマトンです。このように、非決定な遷移規則を持たず、初期状態が1つの有限オートマトンを**決定性有限オートマトン(DFA)**と呼びます。

図1(状態数4)と図3(状態数8)を見比べると、どちらも**/[ab]*a[ab][ab]/**に対応する有限オートマトンですが、状態数が異なっていますね。一般に、NFAから部分集合構成法によってDFAを作ると状態数は増えてしまいます。DFAの状態数については、あとでもう少し踏み込んで説明したいと思います。

DFA型の実装

わざわざ状態数を増やしてまで、NFAからDFAを作る価値はあるのでしょうか？ 答えは「YES」です。なぜなら、状態遷移のシミュレーションが格段にシンプルになるためです。DFAの場合、「どんな文字列でも、遷移状態はただ1つ」ですので、

```
while (str != NULL)
  state = transition[state][*str++];
```

のような単純な2段の表引きのループでDFAのシミュレーションができてしまいます。

実際、**リスト1**のコードはGNU grep[注1]2.14(最新版)から該当部分を抜粋したものです(最初のコメントは付け足しました。インデントもそのままです)。DFAの遷移を2段の表引きで行う、とてもシンプルなコードです。

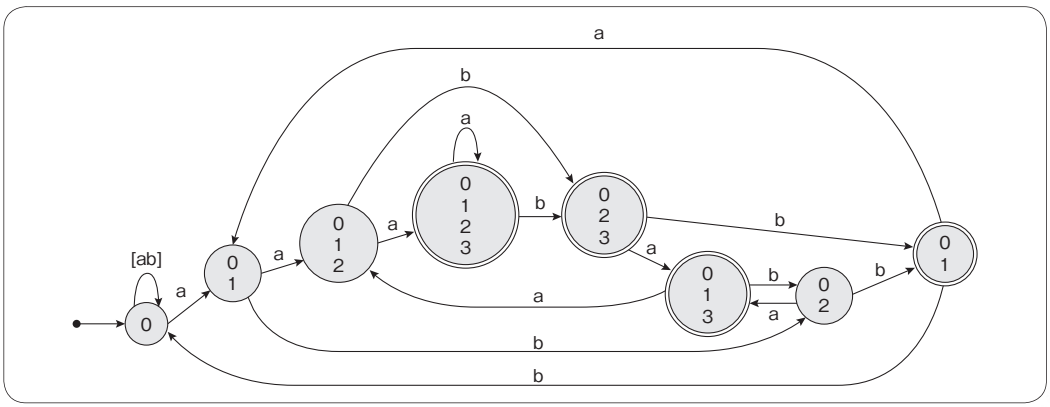
s、s1の2つの変数で、現在と1つ前の状態を保持しています。わざわざ2つの状態変数を用いて、1行目のt = trans[s]と2行目のt = trans[s1]とループ展開しているのは、涙ぐましい高速化の試みと読み取れます(GNU grep 2.9までは「/* hand-optimized loop */」というコメントがありました)。

注目すべき点はtransの表引きの際のNULLチェックで、これは「状態遷移表が完全には埋められていない」ことを表しています。実際、GNU grepは正規表現から最初にNFAを作りますが、そこからDFA(の遷移)を完全に構成することは

▼リスト1　GNU grep 2.14のDFAをシミュレートするコード(src/dfa.cのdfaexec()関数より)

```
/* state_num s, s1, tmp; //状態を表す整数変数
   unsigned char* const p;//入力文字列へのポインタ
   state_num **trans, *t; //遷移規則を格納した配列
*/
while ((t = trans[s]) != NULL)
  {
    s1 = t[*p++];
    if ((t = trans[s1]) == NULL)
      {
        state_num tmp = s;
        s = s1;
        s1 = tmp;     /* swap */
        break;
      }
    s = t[*p++];
  }
```

▼図3　図1から部分集合構成法によって得られた有限オートマトン(DFA)



注1) http://www.gnu.org/software/grep/

せず、マッチング時に必要になった状態だけ状態遷移表を計算していきます。リスト1には載せきれませんでしたが、trans[s]がNULLだったときは、ループから抜けて「状態sから文字*pを読んだらどの状態に行くべきか」を計算し、状態遷移表を埋める処理をします(そのためには、DFAの状態がどの「NFAの状態の集合」に対応するかを裏で保存している必要があります)。このように、DFAの状態とその遷移規則を必要なときに(言葉を変えれば"lazyに")構成するテクニックをDFAの**on the fly構成法**と呼びます。on the fly構成法は現代のDFA型正規表現エンジンには必須と言えるでしょう。DFA型の正規表現エンジン実装としては、GNU grepのほかにGoogleのRuss Cox氏によるRE2[注2]があり、RE2もon the fly構成法を採用しています。

◆ ◆ ◆

ページ数の都合上、正規表現からNFA/DFAを作る細かい手順は割愛させていただきました。このあたりの話はWeb上に優れた資料がいくつかあります。日本語ではhiratara氏の記事[注3]を、英語ではRE2の開発者Russ Cox氏の記事[注4]を参照すると良いでしょう。DFA型エンジンの奥は深く、まだまだ書きたいことは無数にありますが、そろそろもう1つの方式である「VM型」に移りましょう。

VM型——可能性を深さ優先でたどる

ある意味、現代の正規表現エンジン実装の主流である「VM型」のマッチング方式について説明します。

DFA型ではマッチング対象の文字列は「先頭から1文字ずつ順方向に読む」ことでマッチングを行います。VM方式はまったく異なる方法を使います。文字列を進めたり遡ったりを繰り返すのです。ところで、何が「VM」なのでしょうか?

30行のVM型実装

VMとは「与えられた命令列をエミュレートするプログラム(仮想機械)」のことを指します。VM型エンジンでは正規表現が「命令列」、マッ

注2) http://code.google.com/p/re2/
注3) http://codezine.jp/article/detail/3039
注4) http://swtch.com/~rsc/regexp/regexp1.html

COLUMN これは正規? あれは正規?

現在世の中で使われている正規表現は、多くの拡張機能が存在します。その中には、理論的には正規表現と呼べないような機能も多くあります。たとえば、Perlで有名になった「後方参照」「再帰」などがあります。ところで、「理論的には正規表現とは呼べない」という基準は何なのでしょうか? 実は、「選択演算子|と繰り返し演算子*と正規表現をつなげる演算(**/re/**と**/gex/**をつなげて**/regex/**にするなど)のみを用いた正規表現」で書けなかったり、ここで紹介した「有限オートマトン」で計算できないようなモノは正規表現でないと言い切ることができます。**+**、**?**、**{}**、**^**、**$**などの拡張演算子は広く親しまれていますが、これらはすべて正規です。ほかに証明は少々複雑になりますが先読み演算や否定先読み(参考文献[5])も、実は正規です。

この話題はとてもおもしろいので、興味があれば「正規言語の反復補題(pumping lemma)」というキーワードを調べてみてください。また、筆者が過去にShibuya.pmで発表した「正規表現の限界」というスライドでも扱われています[注5]。

注5) http://swatmac.info/etc/shibuya_pm

チング対象文字列が「入力」に対応します。VMは命令をどのように解釈し、入力に対してどう振る舞うのでしょうか？　実は、簡単なものであればたった30行程度で「正規表現と文字列を受け取り、マッチングを行うVM型エンジン」が実装できてしまいます。

リスト2に示すコードは『プログラミング作法』(参考文献[2])から抜粋したRob Pike氏によるコードです。このコードは正規表現の**.**、**^**、**$**、*****演算に対応しています。

さて、リスト2のコードはどのように実装されているのか、簡単に解説したいと思います(シンプルなコードのため、読者のみなさんには必要ないかもしれませんが)。

match関数では、正規表現regexpの先頭が**^**であれば直接matchhere(regexp+1,text);を、そうでなければmatchhere(regexp, text);が真になるまでtextをインクリメントしつつマッチングしています。正規表現が**/^regex/**という形でなければ自動的に**/.*regex/**と解釈するわけで、部分一致をデフォルトとしたgrepの振る舞いです。

matchher関数では、正規表現を命令として解釈している様子がよくわかるでしょう。

```
if (*text!='\0' && (regexp[0]=='.'
                    || regexp[0]==*text))
  return matchhere(regexp+1, text+1);
```

最後のif文は現在注目している命令regexp[0]が**.**の場合は無条件に、そうでない場合は文字列と一致している場合に、文字列と命令を消費して次に進みます。matchherは再帰で処理を進めていますが、すべて末尾再帰となっています。いきなり最後のif文を説明しましたが、このif文に行き着くまでに特別な命令(演算子)について処理を切り分けています。最初のif文では、命令がこれ以上ない場合に無条件で1を返していますが、これはまさに部分一致の振る舞いですね。また、3番目のif文では、正規表現が**regexp$**の形のときは後方一致(文字列の最後まで一致する)の判定を行っています。さて、ようやく2番目のif文まで来ました。

```
if (regexp[1] ==  * )
    return matchstar(regexp[0], ↵
regexp+2, text);
```

1つ先の命令が*****の場合は、どうやら専用の関数呼び出しを行うようです。引数の形が少々特別ですが、正規表現が**/a*regex/**の形の場合に、(/a/, /regex/, text)という引数で呼び出しを行っているだけです。どうやら、繰り返し演算*****の対象は一文字に限定されているようです。

matchstar関数が*****を解釈する関数です。この部分がVM型実装のキモと言えるで

▼リスト2　30行程度の正規表現VM実装(『プログラミング作法』より)

```
int match(char *regexp, char *text)
{
  if (regexp[0] == '^')
    return matchhere(regexp+1, text);
  do { /* 文字列が空の場合でも調べる必要あり */
    if (matchhere(regexp, text))
      return 1;
  } while (*text++ != '\0');
    return 0;
}
int matchhere(char *regexp, char *text)
{
  if (regexp[0] == '\0')
    return 1;
  if (regexp[1] == '*')
    return matchstar(regexp[0], regexp+2, text);
  if (regexp[0] == '$' && regexp[1] == '\0')
    return *text == '\0';
  if (*text!='\0' && (regexp[0]=='.'
                    || regexp[0]==*text))
    return matchhere(regexp+1, text+1);
  return 0;
}
int matchstar(int c, char *regexp, char *text)
{
  do { /* 「*」は「0 回以上の繰り返し」であることに注意 */
    if (matchhere(regexp, text))
      return 1;
  } while (*text != '\0' && (*text++ == c
                            || c == '.'));
  return 0;
}
```

しょう。と言っても、しくみは単純です。条件式

```
(*text != '¥0' && (*text++ == c || c == '.'))
```

は繰り返し対象の文字と文字列が一致するかを判定しているだけで、一致する場合に内部でmatchhere(regexp, text)を呼び出します。つまり、「*で先頭が消費できる範囲で、文字列の部分一致を判定する」わけです。なお、コメントにもあるように*は「0回以上の繰り返し」ですので、最初にmatchhereを無条件で呼び出します。

わずか30行程度のコードですが、VM型のエッセンスに触れる良い例でしょう。「再帰の強力さを存分に活用した美しいコード」と言ったところでしょうか。なお、リスト2のコードは『ビューティフルコード』(参考文献[1])でも解説されています[注6]。

より完全なVM型の実装

30行のVMコードは本質的なコードですが、より実用的なVMを実装するにはさらにコード量を増やす必要があるでしょう。とくに正規表現のグループ化——括弧への対応は必須でしょう。先ほどのコードでは正規表現を直接命令として扱っていましたが、括弧への対応も考えると、正規表現をパースして命令列に変換するのが楽でしょう。一般的なVM型実装は、正規表現を**リスト3**のような命令語に変換します。変換規則を**表1**に示します。

VM型エンジンは、「有限状態で文字は先頭から順番に読む」というDFA型と違って制限がほとんどありません。なんと言ったって正規表現を機械語に変換して、バックトラックまで使えるのですから。そのためVM型エンジンはコラム「これは正規？ あれは正規？」で説明したような「非正規」な演算に対応することも容易です。そのため、PerlやRuby、PythonやJavaScriptなどの言語組込みの正規表現エンジンはほぼすべてVM型実装と言えるでしょう。PCREもVM型実装です。実装のより細かい話は、Russ Cox氏による記事[注7]が参考になるでしょう。一流の開発者が書く記事はとてもタメになります。

注6) http://www.oreilly.co.jp/editors/archives/Btfl_Cd_01.pdf

▼リスト3 VM型実装の命令語(Russ Cox氏の記事より)

```
struct Inst {
  //Char,Match,Jmp,Split
  int opcode;
  int c;
  Inst *x;
  Inst *y; };
```

あいまいさのたどり方——どこがどうマッチしたか?

ここまでページをたっぷり使って、文字列が正規表現にマッチするか否か——「可能性のたどり方」を、DFA型とVM型それぞれの方法を説明しました。

正規表現好きにとってのバイブルである『詳説正規表現 第3版』(参考文献[3])から、本質をとらえた次の記述を紹介します。

ガソリンエンジンのNFAは"正規表現主導型"、電動エンジンのDFAは"テキスト主導型"と言うことができる。

ここで言っている「ガソリンエンジンのNFA」

注7) http://swtch.com/~rsc/regexp/regexp2.html

▼表1 正規表現からリスト3の命令語への変換規則(Russ Cox氏の記事より)

/a/	Char a
/e1e2/	L1: /e1/のコード /e2/のコード
/e1 | e2/	Split L1, L2 L1: /e1/のコード Jmp L3 L2: /e2/ のコード L3:
/e * /	L1: Split L2, L3 L2: /e/のコード Jmp L1 L3:

とは、本章で扱っているVM型実装のことです。本章と『詳説 正規表現 第3版』でNFA/VMと記述が異なっている理由はコラム「NFA？ VM？ バックトラック」を御覧ください。

DFA型（テキスト主導）か VM型（正規表現主導）か

以降では、DFA型実装の代表としてGNU grep 2.14を、VM型実装の代表としてPCRE 8.32の標準ツールであるpcregrepを使って説明を行っていきます。PCREのようなメジャーなVM型実装が、標準でgrepフロントエンドを提供してくれているのはとてもありがたいことです！

VM型が困る例

さて、正規表現エンジンがDFA型かVM型かを判定するのに、てっとり早い方法はあるでしょうか？　もちろんソースコードを読めばわかりますが、もっと簡単な方法があります。次の例を見てみましょう。

```
$ perl -e 'print "a"x30' > hoge.txt
$ egrep -c '(a?){30}a{30}' hoge.txt
1
$ pcregrep -c '(a?){30}a{30}' hoge.txt
pcregrep: pcre_exec() gave error -8 while
matching this text:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
0
pcregrep: Error -8, -21 or -27 means that a
resource limit was exceeded.pcregrep: Check
your regex for nested unlimited loops.
```

egrepでは一瞬で結果が返ってくるのに対し、pcregrepでは0.5秒ほど時間をかけて「resource limit was exceeded」というエラーメッセージが返ってきました。正しい結果も得られていません（grepの-cオプションは「部分マッチした行数」を返すオプションです）。なぜでしょうか？

使用した正規表現

```
/(a?){30}a{30}/
```

は、**{30}**を展開すると

$$\underbrace{\texttt{/(a?)}\cdots\texttt{(a?)}}_{30}\underbrace{\texttt{a}\cdots\texttt{a/}}_{30}$$

という正規表現になります。**/a?/**は「1文字の“a”かε（空文字）」にマッチします。よって、VM型の正規表現エンジンは「**/a?/**でテキストを1文字消費するか否か」という決断を30回行わなければいけません。決断が間違っていれば、バックトラックを行えば良いのですが、組み合わせ的には10億通りを超えます！　なお、一般的なVM型実装の仕様では**a?**は可能なら「1文字消費する」という決断を優先的に行います。今回の例でのマッチング対象の文字列は、「文字“a”の30回の繰り返し」ですので、

$$\underbrace{\varepsilon}_{\texttt{/(a?)\{30\}}}\ \underbrace{\texttt{a}\cdots\texttt{a}}_{\texttt{a\{30\}/}}$$

COLUMN
NFA？VM？バックトラック

『詳説 正規表現 第3版』では、バックトラックを使う実装を**従来型NFA**として説明していました。オートマトン理論が専門である著者としては、この記述は受け入れ難いです。本来、NFAの本質である「非決定性」はバックトラックとは無関係であり、バックトラックはあくまで「非決定性」を計算するための手段の1つにすぎないからです。なお、『詳説 正規表現 第3版』では図2のような（幅優先探索な）NFAのシミュレーション方式を**POSIX NFA**と呼んでいます。既存エンジンの実装の多くは、正規表現を独自の命令語に変換し、バックトラックを主体とした評価機（VM）をベースに実装されているため、本章では「VM型」という表記で一貫させました。

という解釈以外では正規表現と文字列はマッチしません。よって、VM型エンジンは**(a?)**の決断に恐ろしく時間がかかってしまうというしくみです。なお、これはPCREに限ったことではなく、PerlやRuby、Python組込みの正規表現エンジンでも同様です。なお、この問題は括弧をキャプチャしない括弧**/(?:a?)/**に置き換えても解決しません。

正規表現を工夫することで、このような問題を回避可能な場合がいくつかあります。たとえば、先ほどの正規表現を

```
/(a??){30}a{30}/
```

に変えてみればうまく動きます。

```
$ pcregrep -c '(a??){30}a{30}' hoge.txt
1
```

/a??/は、**/a?/**と同じく「1文字の"a"かε(空文字)」ですが、「1文字も消費しない(εにマッチする)」という決断を優先する演算子です。VM型特有の演算子と言えるでしょう。このように、「文字を消費するか否か」という状況において消費を優先する方式を**強欲(greedy)**、消費しない(εにマッチ)方式を**非強欲(non-greedy)**な演算と呼びます。**?**、**+**、*****という強欲な演算子に対して、**??**、**+?**、***?**が非強欲な演算子に対応するのが標準なようです。

DFA型が困る例

もちろん、DFA型にも苦手な正規表現はあります。正規表現の長さに対して、DFAの状態数が極端に大きくなってしまう例です。実は、本章で一番最初に扱った

```
/[ab]*a[ab]{n}/
```

という正規表現(nは自然数)がそのような例で、繰り返し回数nに対してDFAの状態数が指数的に大きくなってしまいます！　このような状況をDFAの**状態爆発**と言います。

では、**/[ab]*a[ab]{100}/**のような正規表現をGNU grepに食べさせるとどうなるのでしょうか？　この場合、DFAの状態数は10の30乗を超えるほど巨大になってしまいます。試してみましょう。

```
$ perl -e 'print "a"x101' > hoge.txt
$ time egrep -c '[ab]*a[ab]{100}' hoge.txt
1
egrep -c '[ab]*a[ab]{100}' hoge.txt  0.00s
user 0.00s system 31% cpu 0.017 total
```

一瞬で計算できてしまいました！　というのも、DFA型の節で触れたように、GNUgrepはDFAの構成を「必要になったら」計算するon the fly構成法を基に実装されているためです。「DFA型が苦手な例」と言いましたが、on the fly構成法によってある程度は状態爆発に対応できるという話でした。

DFA型とVM型——マッチの解釈

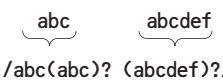
先ほどは「DFA型とVM型を見分ける例」として**/(a?){30}a{30}/**を扱いました。マッチング方式の違いを見分ける本質的な例です。ここで、もう1つDFA型とVM型を見分ける本質的な例を紹介しましょう。

```
/abc(abc)?(abcdef)?/
```

という正規表現に対して、文字列abcabcdefをマッチングさせてみましょう。grepに-oオプションをつければ「正規表現にマッチした部分だけ出力」させることができます。

```
$ echo 'abcabcdef' > hoge.txt
$ egrep -o 'abc(abc)?(abcdef)?' hoge.txt
abcabcdef
$ pcregrep -o 'abc(abc)?(abcdef)?' hoge.txt
abcabc
```

おもしろい結果になりました。この正規表現と入力文字列では、マッチングは



と解釈すれば、完全一致となります。しかし、

前述したように、VM型実装では**/(abc)?/**に対して「文字列を消費する」決断を優先させるので、

```
abcabc      ε      def
/abc(abc)? (abcdef)?/
```

という結果(前方一致)になるわけです。

なお、DFA型エンジンでは多くの実装で**最左最長**という「最も左側で始まり、最も長いマッチ文字列を採用する」という方式を採っています。多くの場合、最左最長という規則はユーザの目的に見合った規則と言えるでしょう。

ここで扱ったDFA型とVM型の挙動の違いは、『詳説 正規表現 第3版』では丸々1章(第4章)使って説明されているので、より深く知りたい方は参照してみてください。

grepのパフォーマンス

残りページは少ないですが、ここで既存のDFA型エンジンとしては最速レベルのGNU grepのパフォーマンスとその秘密について、いくつか説明を行いたいと思います。ここでは、実用的かつ複雑な例ということで、RFC定義のURIを正規表現を用いて速度比較を行ってみました。URIとはUniform Resource Identifierのことで、HTTP-URIなどを含んだ識別子のことです。「http://gihyo.jp/」などはもちろんURIの1つです。RFC定義のURIが正規表現で書けるという事実は、あまり知られていないようです。コラム「厳密なHTTP URIは正規表現で書ける」も御覧ください。

図4に、Wikipedia日本語版全文(6.9GB)に対して、RFC定義のURIおよびHTTP-URI(スキームをhttpで固定したもの)をgrepした場合の実行時間(timeコマンドによるreal時間[sec])を載せました。なお、実験環境はIntel(R) Core(TM) i7-2600 CPU@3.40GHz、DDR3-1333 8GBのメモリを搭載したマシンで行いました。Wikipedia日本語版全文はメモリ上に載っている状況で、このベンチマークによるディスクI/Oは発生していません。

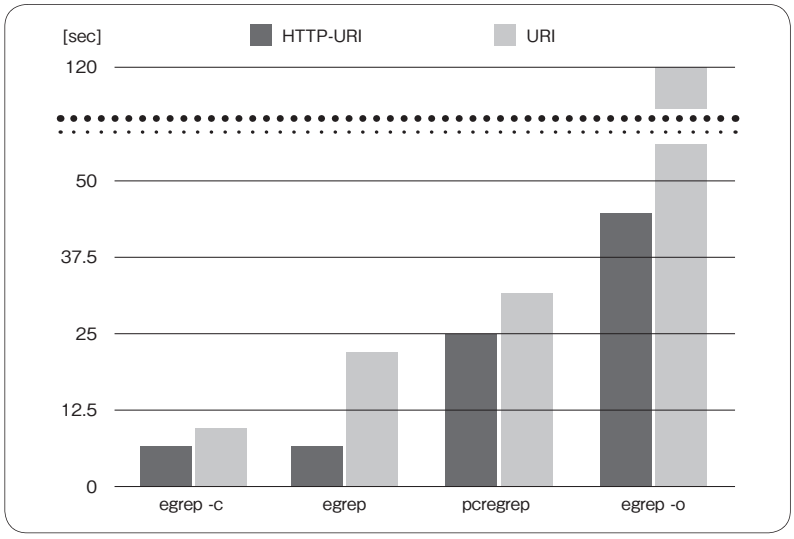
「行数を数えるだけ」の-cオプションのgrepが最も高速という結果が出ました。行の出力にかかる処理が必要ないため、これは当然の結果でしょう。ちなみに、HTTP-URIを含む行は1,833,126行(544MB)、URIを含む行は17,897,823行(1.6GB)という結果でした。

GNU grepにはおよびませんが、pcregrepも奮闘しています。VM型実装では驚異的な速度と言えるでしょう(PCREはDFAエンジンも内部で実装しているので、部分的にDFAを使っている可能性もあるかもしれません)。

「マッチした部分だけ出力する」-oオプションを付けた場合が最も遅く、純粋にDFAを走らせる通常のマッチングよりも、最左最長マッチを補足するための処理によるオーバーヘッドが読み取れます。

grepのパフォーマ

▼図4　URIによるgrep速度比較(GNU grep 2.14, pcregrep 8.32を使用)



ンスについて、まだまだ書くべきことが多いのですが、ページ数の都合上このあたりで考察を留めておきましょう。grepの実装内部に関しては多治見寿和氏の『プログラミングテクニックアドバンス』(参考文献[4])で詳細な解説が行われています。とくに固定文字列探索による高速化の説明がしっかりしており、grepを深く知りたい人にとっては必読と言えるでしょう。

最後に

DFA型とVM型という異なる2つのしくみを説明してきました。DFA型はしくみ的に高速ですが、VM型はしくみ的にキャプチャへの対応や機能拡張が容易です。しかし、実用的に進んだ実装になるほど両者の境界はあいまいになってきます。たとえば、VM型実装のPCREもDFA型実装を内部に持っていますし、GNU grepも純粋なDFAでは不可能な後方参照に対応(バックトラックに切り替える)しています。grep1つとっても、オプションの違いで実行速度が大きく変わるので、ユーザが状況に応じて、最適なエンジンの種類やオプション、正規表現の書き方を工夫する余地があるでしょう。

エンジンのしくみやそれぞれのメリット／デメリットを把握したうえで使いこなして初めて、「正規表現をマスターした」と言えるのではないでしょうか！SD

＜参考文献＞
[1] Brian,K.『ビューティフルコード』, オライリージャパン,2008.
[2] Brian,K.Rob,P.『プログラミング作法』アスキー,2000.
[3] Friedl,J.E.『詳説 正規表現 第3版』, オライリージャパン,2008.
[4] 多治見寿和『プログラミングテクニックアドバンス』, アスキー,2004.
[5] 森畑明昌『先読み付き正規表現の有限状態オートマトンへの変換』, コンピュータソフトウェア,Vol.29,No.1(2012),pp.147-158.

COLUMN
厳密なHTTP URIは正規表現で書ける

URIは1998年にRFC2396で規定され、その後2005年にRFC3986で改訂されました。RFCによるURIの定義には、厳密な文法(拡張BNF)も定義されています。ところが、ここで定義されている文法は正規表現に変換可能なのです(一般的にBNFで定義される文法は正規表現では表せません)。

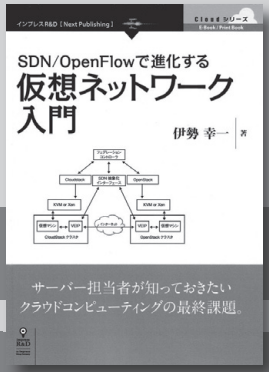
RFC3986定義のURIは著者のGitHubで公開しています[注8]。なお、この正規表現はRFCのABNFからTanaka Akira氏のabnf[注9]というツールを用いて変換した結果を、egrep用に変換したものです。

さて、現在のRFC 3986定義のURIはいわば「バージョン2」なわけですが、バージョン1(RFC2396)に比べてどれほど仕様が複雑になったのでしょうか？　意味的な複雑さを評価するのは難しいですが、正規表現的に複雑さを計ることは可能です。DFAの状態数によって複雑さを比べてみましょう。RFC3986定義のURIは、状態数が180状態のDFAが対応します。一方のRFC2396定義のURIは、なんと状態数がたった13状態のDFAが対応するのです[注10]。7年越しのバージョンアップによって、URIは正規表現(DFA)的には10倍以上も複雑なものになってしまったのです！

注8) https://github.com/sinya8282/RANS/blob/master/test/uri.rfc3986.posix.regex
注9) http://www.a-k-r.org/abnf/
注10) 著者のプロジェクトページ http://sinya8282.github.com/RANS/index.html#uri_image で180状態のDFAの図を見ることができます。

BOOK no.1 SDN/OpenFlowで進化する 仮想ネットワーク入門

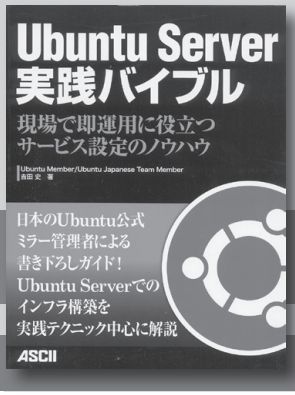
伊勢 幸一 【著】
A5判、 198ページ／価格＝2,520円 （税込） ／発行＝インプレスR&D／ISBN＝978-4-8443-9575-1
※上記価格は印刷書籍版のもの。 印刷書籍はAmazon.co.jpや三省堂書店のプリント・オン・デマンド （POD） での販売になります。



SDN/OpenFlowはクラウドコンピューティングの基盤として必然なのか。本当に必要なのか。これは、今までケーブリングや設定を仕事としてきたネットワークエンジニア、インフラエンジニアにとって大きな問題である。本書は、SDN/OpenFlowの存在理由を明らかにする。ネットワークの物理層の説明から始まり、仮想LAN、VXLAN、仮想スイッチ、広域ネットワーク……と視野を広げていく。筆者の現場経験の豊富さゆえか、それぞれに核心をついた説明がされているので、仮想ネットワーク技術の流れが大局的に見えてくる。OpenFlowについて知りたい、その問題点は何かと考えていきたい方にとって基準となる本である。

BOOK no.2 Ubuntu Server 実践バイブル

吉田 史 【著】
B5変形判、 416ページ／価格＝3,360円 （税込） ／発行＝アスキー・メディアワークス
ISBN＝978-4-04-886687-3



本書は、本誌連載「Ubuntu Monthly Report」でもお馴染みの吉田史氏によるUbuntu Server構築の実践的な入門書だ。

Ubuntu Serverは自動的にLAMP環境が組み込まれるため、サーバ環境を簡単に構築できる。しかし、操作はCUIで行う必要があり、Ubuntu DesktopのGUI操作と比べると初心者にはとっつきにくい。本書では、サーバ構築におけるLinuxコマンドやサーバに関する基礎知識などの説明も交えながら、実際の運用経験からくる暗黙知や作法などのより深い解説がなされている。

Linuxによるサーバ構築の初心者から、より深い設定をしたい方まで参考になるだろう。

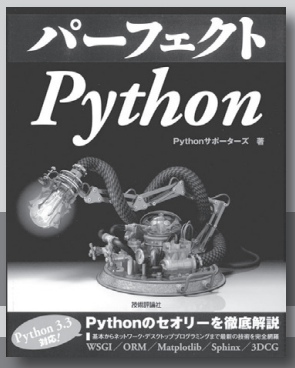
BOOK no.3 入門Chef Solo Infrastructure as Code

伊藤 直也 【著】
124ページ （A4 PDF版換算） ／価格＝890円 （税込）
※本書は電子書籍です。 Amazon.co.jp （Kindle版） と達人出版会 （EPUB、 PDF） での販売になります。

Chefとはサーバの構築／運用を自動化するツール。クライアント／サーバモデルで利用するものだが、小規模利用向けに管理対象のサーバのみで扱えるように用意されているのがChef Soloだ。本書はChefを利用するとっかかりとして、Chef Soloの基本的な使い方を紹介する。実際に試せるように試験環境の導入から解説されている。書名にChef Soloと銘打っているが、最後に大規模向けのChef Serverの機能紹介もある。そこまでのChef Soloの解説をしっかり読んでいれば、ServerとSoloの違いや、システムにどちらを適用すべきかの判断基準も見えてくる。大規模への導入を検討中の人も、まずは本書からChefに入ることをお勧めする。

BOOK no.4 パーフェクトPython

Pythonサポーターズ 【著】
B5変形判、 464ページ／価格＝3,360円 （税込） ／発行＝技術評論社
ISBN＝978-4-7741-5539-5



Python 3を基本から実践まで網羅的に解説した1冊。取り扱う内容は、言語の特徴や仕様の解説に始まり、テキスト／ファイル処理、アプリケーション開発、アプリケーションやライブラリの配布、テスト、Webプログラミングなどの開発手法の説明、専門分野ごとのサードパーティ製ライブラリの紹介など幅広い。環境構築、ライブラリの導入方法は最後にまとめて掲載しているため、Pythonの使い方に専念して読み進められ、学習も捗る。分量はかなり多いが、内容はほかの言語でプログラミングの要点を理解していれば、十分理解できる。プログラミング経験者なら中途半端な入門書よりも、本書のほうが体系的に学べるだろう。

OpenFlowを実装してみた!

OpenFlow/SDNフレームワーク

バーチャルネットワークコントローラ2.0開発の実際

㈱NTTデータ　基盤システム事業本部
システム技術方式ビジネスユニット
小島俊範(こじまとしのり)
前田繁章(まえだしげあき)

昨今ネットワーク業界では、Software-Defined Networking(SDN)というコンセプトと、それを実現する技術の1つであるOpenFlowが注目されています。この状況下、当社ではSDNビジネスを推進していくにあたり、OpenFlowを簡易に導入できる「バーチャルネットワークコントローラ 2.0(VNC2.0)」のフレームワークの提供を2013年2月より開始しました。本稿では、OpenFlow/SDNの復習から始め、VNC2.0とはどういうものか、その特徴や開発秘話を交えながら解説します。

Suirokaku (Aqueduct Bridge) / Hyougushi

ソフトウェアでネットワークの機能を定義するOpenFlow

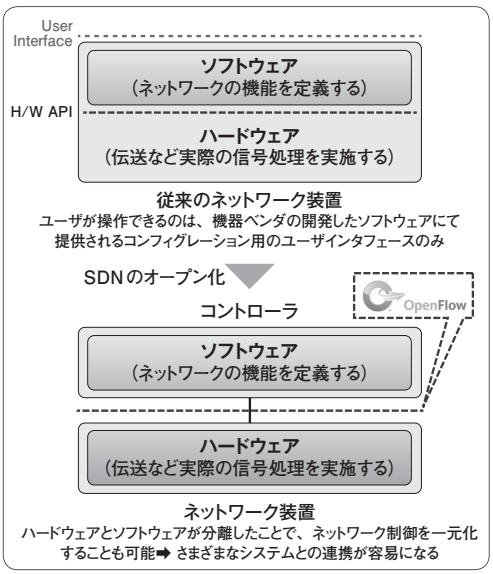
SDNとはSoftware-Defined Networkingの略であり、文字どおりソフトウェアでネットワークの機能を定義します。そもそも、このSDNはネットワークを構成する概念やシステムアーキテクチャといった類のものであり、特定の技術や機能を示す言葉ではありません。このキーワードは2012年からとくに注目を集めるようになりましたが、アーキテクチャ自体は目新しいものではなく、従来のネットワーク装置でも同様なアーキテクチャは見られました。ただし、従来のネットワーク装置では機器ベンダがハードウェアを制御するためのAPIを非公開としていたため、機器ベンダ以外が独自にネットワーク機能を開発することは事実上不可能であり、ユーザがSDNのメリットを享受することがなかったというのがこれまでのネットワーク業界の特徴でした(**図1**)。

ユーザは欲しい機能があってもベンダがNoというネットワーク機能を実現できませんでした。ベンダがYesといえば機能を実現できるようになりますが、時間がかかるものでした。例として、複数データセンタ間でのマルチテナントのブロードキャストドメインを組む例では、VXLANのような標準技術ができるまで待たなければなりませんでした。

この長年にわたり続いてきたネットワーク業界の慣習からから脱却し、ベンダから提供を受けるのはハードウェアだけで、ネットワーク機能を実現するためのソフトウェアはユーザ自身で開発するといった流れが起きています。つまり今ネットワーク業界で起きているのはSDNアーキテクチャのオープン化と言えるでしょう。

OpenFlowとは、SDNを実現するための技術であり、ソフトウェアからハードウェアを制御するためのインタフェース※を定義したもので

▼図1　SDNの概念とOpenFlowの関係



※注)本稿では筆者らの意向により「インタフェース」と記載しています。

す。これまで機器ベンダが非公開としていたこのインタフェースが標準化されたことで、機器ベンダ以外でも独自のネットワーク機能を定義できるようになりました。つまり、市販の機器では実現できないようなオリジナルな機能／しくみについても、OpenFlowを活用してソフトウェアの領域で実現できるようになったのです。

SDNコントローラ開発フレームワークの構造と機能

当社ではこのOpenFlowの技術を使い、ソフトウェアの領域でユーザが作りたいネットワークを実現するためのSDNコントローラを開発するフレームワークとしてVNC2.0を開発しました。

たとえば、端末がウィルス感染した場合について考えてみます。従来技術ではそのような端末はいったんネットワークから隔離し、DVDメディアなどを使ってウィルスを駆除・パッチあてなどをしたのちに再度ネットワークに接続する必要がありました。OpenFlowでは、そのような端末に対して個別の通信経路を設定し、ソフトウェアのアップデート等のために必要なサーバとのみ通信し他との通信をシャットダウンするようにできます。そして端末のウィルス駆除・パッチあてなどを行った後、元のネットワークに所属させるということもできます。このようにすることで、より安全に柔軟にネットワークレベルで制御できるようになります。VNC2.0を利用し、そのうえでソフトウェアを開発することで、仮想ネットワークを構築制御する機能や、複雑な経路制御を行う機能など、ネットワーク機能の高度化が実現可能になります。

VNC2.0の内部アーキテクチャ「VNC-NOSとVNC-AP」

VNC2.0でどのようなことができるか、ソフトウェアエンジニアにとってどのような利点があるのか、その理解を深めていただくため、VNC2.0の内部アーキテクチャについてもう少し詳しく見ていきます。

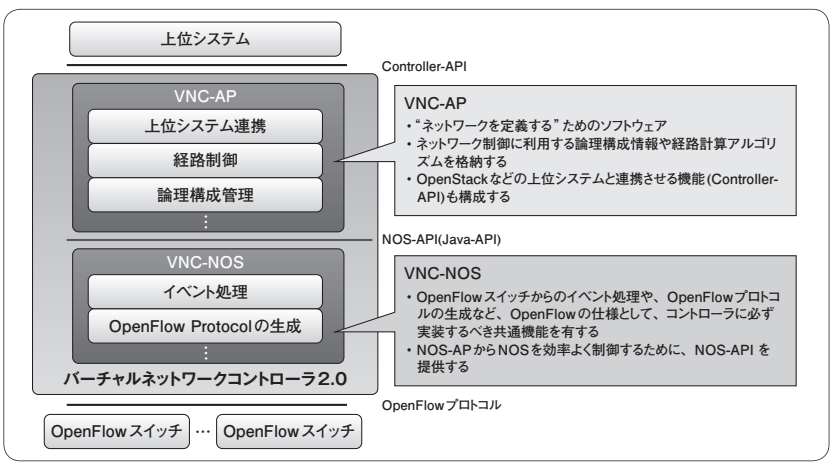
◆VNC-NOSとVNC-APそれぞれの役割

VNC2.0は大きくVNC-NOSとVNC-APとに分かれています(**図2**)。VNC-NOSではOpenFlowスイッチからのイベント(セキュアチャネルの確立、Packet-In、フローの削除通知)を処理したり、OpenFlowプロトコルのフォーマットに沿った電文の生成・解析をしたりします。OpenFlowの仕様としてコントローラに必ず実装すべき機能を有することになります。

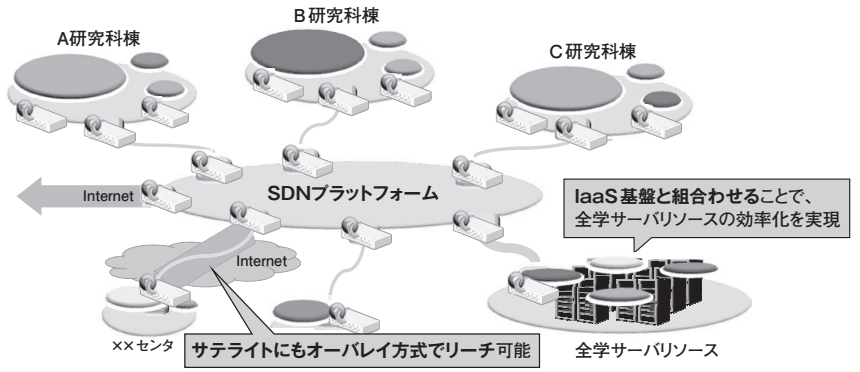
また、VNC-APからVNC-NOSを効率よく利用しやすい形で制御するためのAPIとしてVNC-NOS APIを提供します。このAPIはJava APIとして提供されています(VNC-NOSはFloodlightやtremaといったOSSなどを流用せず、Javaを開発言語としてフルスクラッチで開発されています)。

一方、VNC-APではユーザが実現したい機能を自由に開発できます。この部分は、ネットワークを定義するためのソフトウェアです。基本的には本フレー

▼図2　VNC2.0の構成



▼図3　さまざまなネットワーク機能の例「キャンパスネットワーク」



SDNプラットフォームに、これまでの自治ドメインごとにネットワークセグメントを提供するのはもちろん、**より細かいレベルでのポリシーベースのネットワークの提供が可能**になる。所属するドメインにあるリソースに対して、**SDNプラットフォーム上のどこからでも、制約を受けることなくアクセスすることが可能**となる。

ムワークを使ってVNC2.0のユーザが独自にプログラムを開発することになります(なお、当社では後述のように、論理L2ネットワークを実現するためのVNC-APとして、VNC-Standard APを開発／商品化しました。)。

VNC-APとして、論理構成管理情報や経路計算アルゴリズム、OpenStackなどの上位システムと連携させる機能を実装することで、VNC2.0のユーザが自由なネットワークを実現できるようになります。

VNC-AP開発によって「NWの機能を創る」という発想

VNC2.0ではVNC-APを開発することで、データセンタ事業者が運用効率化のためのしくみを作ったり、サービスプロバイダが顧客の求めるネットワーク構成に合わせて通信路を確保するしくみを作ったり、通信キャリアが独自プロトコルの機能を具備したりすることが可能になります(**図3**)。

このような「ネットワークの機能を創る」という発想は、ネットワーク機器は設定するもので装置そのものはソフトウェア開発するものではないという立場の方(多くの読者がそうでしょう)からすれば、今までにない新しいものだと感じられるでしょう。

VNC2.0では、スイッチからの通信のハンドリングがOpenFlowプロトコルで規定されている仕様に沿ったビット列の解析など、アプリケーション開発技術者が苦手としがちな低レイヤの部分についてはVNC-NOSに処理をゆだねてしまいます。そのうえでソフトウェア開発の技術者が得意とする経路計算のためのアルゴリズムの実装やスイッチからのメッセージを契機に実行する再計算やフローの再読み込み処理などの実装をVNC-APで実現するアーキテクチャになっています。つまり、一からコントローラを開発するよりもかなりとっつきやすくなっています。ネットワークを創るということをより実現しやすいアーキテクチャになっているのです。

マルチベンダスイッチ対応

VNC2.0ではいろいろなスイッチとの接続試験を通して、マルチベンダスイッチ対応を実現しました。そのために、いくつものOpenFlowスイッチとの接続性を確認、VNC2.0の実装への反映を行ってきました。ここではプログラムを実際に開発し、VNC2.0を実装した担当者の立場から、エピソードをいくつか紹介します。

◆フィールドの書き換えができないOpenFlowスイッチ

OpenFlowプロトコルでは、パケットの特定のフィールド(送信元のMACアドレスや送信先のIPアドレスなど)を書き換えるためのメッ

セージを、コントローラからスイッチに対して送ることができるようになっています。しかしながら、すべてのOpenFlowスイッチがすべてのパケットの要素を書き換えられるとは限りません。OpenFlowスイッチAは送信元MACアドレスを書き換えられるけれど、OpenFlowスイッチBは送信元MACアドレスを書き換えられないといった具合です。

このようなOpenFlowスイッチごとにみられる機能の差を吸収するために、VNC2.0ではOpenFlowスイッチで書き換えができないフィールドがあれば、そのフィールドへの書き換え命令は出さないようなしくみに実装しました。こうすることでOpenFlowスイッチからのエラーの発生を防ぎ、スムーズに処理を進めるようになりました。

◆仕様上明確に定義されていないビットの存在

OpenFlowプロトコルでは「Aの処理を行う場合には、aという構造体の中のとある部分に物理ポートを設定する」のような記述がされている個所がいくつかあります。こうした場合に、「A以外の処理を行う場合に、aという構造体に何を入れるべきか」が記載されていないことがよくあります。

たとえば、スイッチの10番ポートでバッファリングされているパケットをそのポートから送出する際には、「ポート番号10のパケットを送出」と記述します。ところが、コントローラで生成された情報を送出する際には「ポート番号xのパケットを送出」のxに何を入れるべきかは記載されていません。こういったケースでは、A以外の処理の場合にaという構造体にどういった値を入れればよいかというのはOpenFlowスイッチの開発元によってばらばらになっています。このようなOpenFlowスイッチごとの細かな差異を意識しつつコントローラであるVNC2.0を開発していくことは非常にたいへんでした。それと同時にOpenFlowプロトコルに対する理解がより深まるものでもありました。

外部システム連携

OpenFlowを単体で利用するのではなく、いろいろなシステムと連携させることでよりその用途が広がります。ここではVNC2.0と外部とのシステムとを連携させるケースについて説明したいと思います。

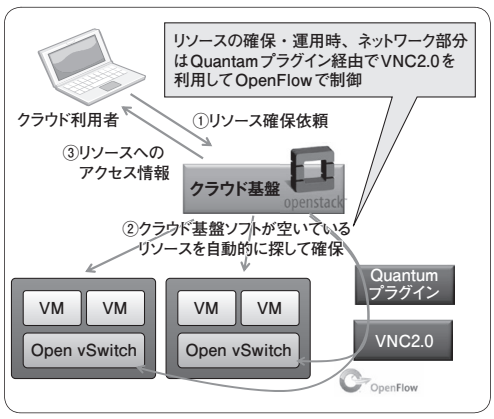
◆OpenStack(Quantum)との連携

NTTデータでは、VNC2.0とOpenStackとを連携させるため、OpenStackで定義されているQuantum APIを通してVNC2.0を制御するためのQuantumプラグインを開発しました。その構成イメージは**図4**のようになります。

OpenStackからQuantumのAPIを経由して、仮想のネットワークを作る、ネットワークにポートを追加するなどの命令をVNC2.0が受け取ります。その結果、受け取った命令に応じてOpenFlowを利用してネットワーク層の制御を実現することが可能になります。

この開発をする際には、OpenStackが管理するtapという情報と、OpenFlowで管理するportという情報の扱いが1つの課題でした。OpenStackからは、VMの各ネットワークインタフェースはtapに接続したりtapから外したりというふうに処理することになります。OpenFlowからすればスイッチのportが接続されたり、portがシャットダウンされたりという

▼図4　OpenFlowとOpenStackとの連携例



ふうに見えます。この際のportとtapとをどう紐付けするか、管理はどうするかといったところが開発をするうえでのポイントになりました。

VNC-Standard APでSDNをスムーズに導入

VNC2.0は、前述のようにVNC-APとVNC-NOSで構成されます。VNC-APは基本的にユーザの方々が自由にアプリケーションを開発するものです。VNC-Standard APは、ユーザの代わりに当社が基本的な機能をパッケージしたものです。これを用いれば、VNC-APをユーザが開発することなく(ユーザに代わって「当社で機能定義して、その世界観のVNC-APを開発した」といったほうが正確かもしれません)、SDNコントローラを利用できます。そのしくみを紹介しましょう。

論理L2ネットワークサービスを実現

VNC-Standard APでは論理L2ネットワークサービスを提供します。複数のスイッチの物理スイッチを使ってポートVLANのように論理分割したり(しかもVLAN数制限のような上限はありません)、MACアドレスのリストで論理的なL2セグメントを構成したりできます。

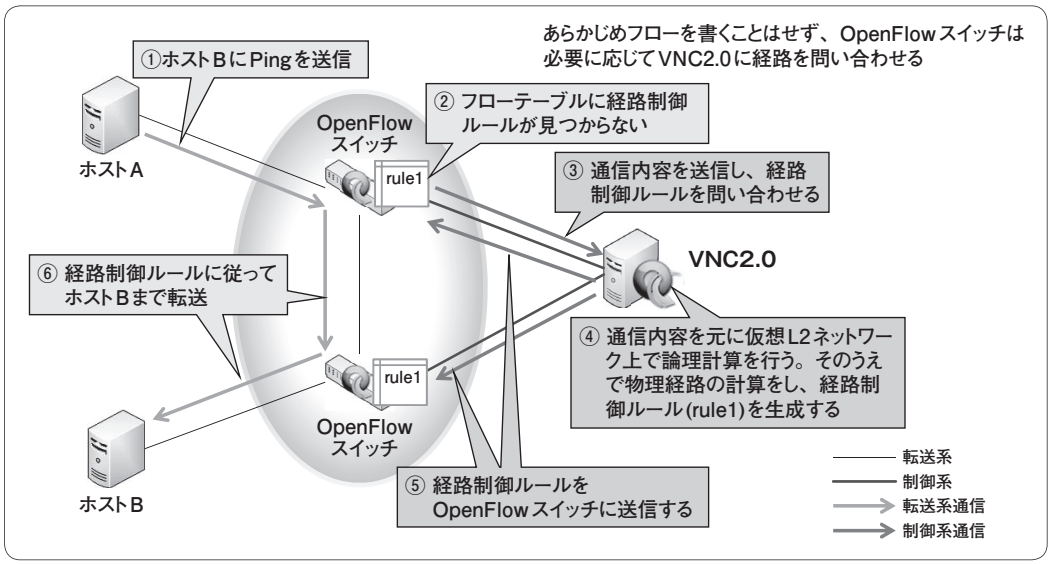
基本的なネットワークはL2で事足りることが多いので、L3スイッチングなどの機能についてはVNC-Standard APでは実装されていません。そのような機能をユーザが使いたい場合には、VNC-APを独自に開発することになります。

◆VNC-Standard APの基本的な動作

VNC-Standard APの基本動作(**図5**)は次のようになります。

①ホストが通信を開始する
②OpenFlowスイッチから問い合わせを受ける
③問い合わせ時の情報(Packet-Inとして受け取ったホスト間通信のMACアドレスやスイッチのポート番号の情報)を内部管理情報に追加する
④上記の問い合わせ時の情報を元に論理L2ネットワークを特定する
⑤特定した情報と、内部的に管理しているホストの接続情報から届け先のOpenFlowスイッチの物理ポートを特定する
⑥特定したOpenFlowスイッチの物理ポートまで届けるためのフローを各OpenFlowスイッチに書き込む

▼図5　VNC-Standard APの基本動作



⑦ホスト間の通信ができるようになる

このように、VNC-Standard APではOpenFlowプロトコル1.0の基本的な動作を踏襲して、OpenFlowスイッチからの問い合わせを受けて動作する方式(リアクティブ方式)を採用しました。

◆OpenFlowスイッチの性能・機能制限を意識

VNC-Standard APを実装するにあたり、OpenFlowスイッチのPacket-Inの性能や、保持可能なフローエントリ数を意識しながら開発しました。この際には、できるだけ無駄なPacket-Inを防ぐ(Packt-Inが頻発するとOpenFlowスイッチにもコントローラであるVNC2.0にも負荷がかかるのでなるべく減らしたい)ためには、どういうタイミングでアドレス学習などを行えば良いか、複数のOpenFlowスイッチにフローを書き込むときはどういう順序でOpenFlowスイッチにフローを書き込めば良いかが、重要なポイントになりました。

フローを書き込む際も、可能な限りフローテーブルのフロー数が溢れないようにしつつ、論理的なセグメンテーションは担保する(異なる論理L2ネットワークどうしはブロードキャストも含めて遮断する)ことが開発の障壁となりました。できるだけマッチングルールとしてワイルドカードを使えばフロー数は減らせますが、減らし過ぎると論理的な破綻を起こします。たとえば「宛先MACアドレスがブロードキャストなら、すべてのポートからパケットを送る」とすると、ブロードキャストはされますが異なる論理L2セグメントにも届いてしまいます。そういったことが発生しないようにフローを考え、そのフローが書き込まれるようにコントローラを開発するところが大きな技術的課題であり、逆に面白いところでもありました。

外部ネットワーク接続

VNC-Standard APは、外部ネットワークとの接続性も考慮して開発しました。これは、既存ネットワークからすべてOpenFlowのネットワークに変わるのではなく、徐々にOpenFlowのネットワークに置き換わっていくであろうというコンセプトに基づくものでした。

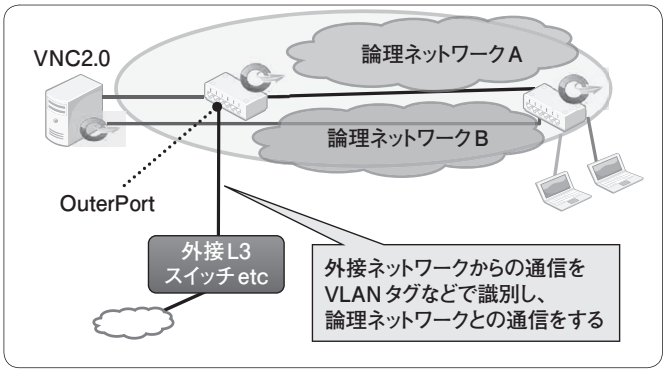
◆既存ネットワークとの接続性について

既存ネットワークとの接続点のことをVNC2.0ではOuterPortと呼んでいます。この時の構成イメージは**図6**のようになります。

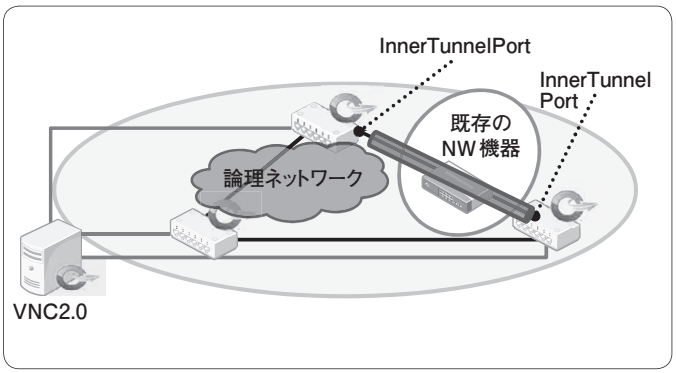
このようなネットワークを制御するしくみを開発した際にポイントになるのは、物理的には1つのポートなのにもかかわらず、そのポートにはさまざまな論理L2ネットワークのいろいろなホストが接続されるということです。

開発を進めるうえでは、このような特殊なポートの制御を矛盾なく管理するしくみの実装が不可欠でした。一般的には、1つのポートに1つのホストが存在するというデータ構造になりますので、この既存ネットワークとの接続点のデータ構造だけが特殊になります。それによって論理情報の保持の仕方や、経路計算方法も既存ネットワーク経由のものとそれ以外のものとでは異なった形になり、より複雑な処理が要求されました。

▼図6　OuterPortを利用した構成例



▼図7　トンネルを利用した構成例（InnerTunnel）



GREトンネリング対応

VNC-Standard APでもう1点特徴的なのが、トンネルを利用したオーバレイ方式にも対応した点です。これには2パターンがあります。

◆InnerTunnel

1つめは、同一拠点内（1つのVNC2.0で制御される）でトンネルを利用するケースです（**図7**）。この場合、トンネルの中の制御は既存の装置にゆだね、トンネルの外の制御はVNC2.0で行います。1つのコントローラで完結しますので、比較的実装がしやすいものでした。

◆OuterTunnel

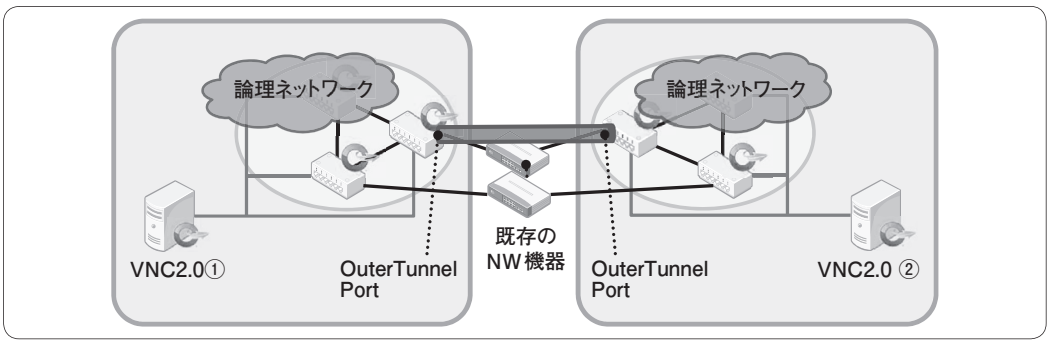
もう1つは複数拠点（複数のVNC2.0で制御される）でトンネルを利用するケースです。この場合もトンネルの通信は既存の装置にゆだね、トンネル外の制御はVNC2.0で行います。

ただ、InnerTunnelの場合と異なり、複数のVNC2.0で制御されるケースでは、トンネルの相手側からの通信が同じ論理L2ネットワークに所属するか否かの判断を行う必要があります。VNC2.0ではその際に、トンネリングプロトコルのIDを使って論理L2ネットワークの識別をしています。**図8**のケースではVNC2.0①で処理する際のトンネルIDとVNC2.0②で処理するトンネルIDとをそれぞれ識別し、同一の論理L2ネットワークの識別を行っています。

◆トンネルそのものはOpenFlow 1.0の仕様外

VNC2.0はGREプロトコルを使ってオーバレイを実現しています。このプロトコルはOpenFlowプロトコルとしては規定されていないため、ベンダ拡張機能を使って実現しました。ベンダ拡張機能を使うためどうしても対向となるOpenFlowスイッチとの連携が不可欠です。開発を進めるうえでは、これがポイントでした。スイッチとどのように連携させ、ベンダ拡張機能を有効に利用するにはどういった仕様・実装にすべきか、ベンダ拡張機能に対応していないOpenFlowスイッチに対しては、どのような処理をすべきかを考慮しながら実装を進めるところが、エンジニアとして工夫のしどころでした。

▼図8　トンネルを利用した構成例（OuterTunnel）



Sample-AP解説

さて、ここからは、VNC-APのサンプル実装としてVNC2.0お試し開発版に付属するSample-APについて解説します。Sample-APが実現しているネットワークの機能は、VNC-Standard APと同じく論理的なL2制御機能です。ただし、あくまでサンプルのため、外部ネットワークとの接続機能は備えておらず、トンネリングプロトコルによるオーバレイ方式にも対応していません。その代わり、試験的に実装しているオリジナルの機能として、ネットワークのトラヒック状況に応じた負荷分散機能を備えています。

開発コンセプト「L2スイッチの論理化」

Sample-APの開発経緯について少しお話しします。

◆システムの配線をリモートで行えるようにしたい!

開発のきっかけは、2011年夏頃の社内通達でした。都内の電力不足が深刻となり、職場が保有する実験機材を、西日本のデータセンタに移設することが決定したのです。当チームではOpenFlowを含むネットワークに関する基盤技術の研究開発を行なっていますので、実験環境としては「各実験機材間の配線を変更する」といったことがしばしば必要となります。

これまでは、これらの実験機材はすべて手元にあったので、配線変更は必要に応じて都度自由に行えたのですが、これら機材が遠隔地にあるとなるとそうもいきません。「ちょっとLANケーブルを1本差し替えたいから新幹線で片道数時間掛けて作業しに行ってきます」というのでは、あまりにも非効率です。

そこで出てくる発想が、「あらかじめすべての機材のネットワークインタフェースをL1フラットに接続しておき、必要に応じてリモート操作で任意のネットワークインタフェースどうしをL2接続できるようにする」というしくみでした。そして、おあつらえ向きなことに、このしくみを簡単に実現できる技術として、OpenFlowがそこにあったのです。

独自機能「トラヒックベースの負荷分散」

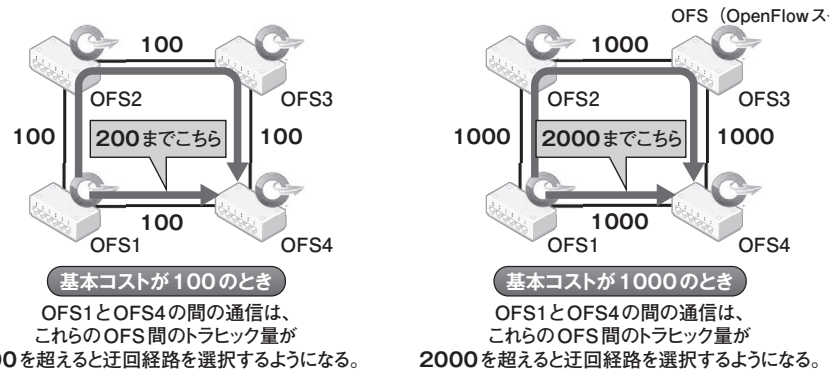
さて話を戻します。ここでは独自の機能として実装したトラヒックベースの負荷分散機能について解説します。OpenFlowネットワーク上のあるホスト間で通信が発生した際、基本動作の⑥にある「特定したOpenFlowスイッチの物理ポートまで届けるためのフローを各OpenFlowスイッチに書き込む」という処理が行われます。このとき、フローの始点・終点となるOpenFlowスイッチは必然的にそれぞれのホストが接続されているスイッチに決まりますが、その間を経由するOpenFlowスイッチはVNC-APに実装されている経路決定ロジックによって決まります。最も簡単な経路決定ロジックは、「経由するOpenFlowスイッチの数(すなわちホップ数)が最小となる経路」でしょう。Sample-APでは、ここに少し工夫を加え、フローを書き込む時点でのネットワーク上のトラヒック量に基づいて経路を決定するようにしています。

このしくみを実現するために、Sample-APではOpenFlowスイッチの持つ統計情報を利用しています。この統計情報はOpenFlowのプロトコルにおいて定義されており、ポート単位・フロー単位等、いくつかのレベルでのデータ転送量等の情報があります。VNC-NOSではこうした統計情報をOpenFlowスイッチから取得するためのAPIも備えているので、Sample-APはその中のポートの統計情報を使い、上記のような経路決定ロジックを実現しています。

◆統計情報をどのように利用しているか

さらに具体的に説明します。Sample-APは、定期的にすべてのOpenFlowスイッチに対してポートの統計情報の問い合わせを行い、その結

▼図9　基本コストの違いによる挙動の変化



果に基づきOpenFlowスイッチ間の各リンクの「コスト」を更新します。このコストの値は、各リンクに設定されている基本コスト値に、問い合わせたポートの統計情報から算出される各リンクの単位時間あたりのデータ通信量を加えたものになります。そして、Sample-APは新たにフローを書き込む際、「始点から終点までを結ぶ経路のうち、現在のコストの総和が最小となる経路」を選択します。

このようにすることで、現在多くのトラヒックが流れているリンクを避け、より帯域が空いているリンクを使って通信が行われるようになります。

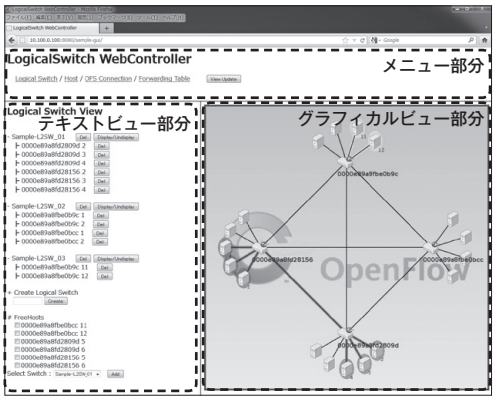
ここで用いる基本コスト値は、ホップ数と負荷分散のバランスを取るためのパラメータであり、これを調整することで「帯域をどれぐらい使っていたら迂回経路を選択するようになるか」をある程度操作できます。

図9において、基本コストが100のとき、OFS（OpenFlowスイッチ）1とOFS4の間の通信は、トラヒックが流れていない状態では最短経路（OFS1-4）の総コストが100、迂回経路（OFS1-2-3-4）の総コストが300なので、OFS1-4の間のトラヒック量が200を超えた（基本コストの100と合わせて最短経路の総コストが300を超えた）時点で迂回経路が使われるようになります。一方、基本コストが1000のときについて同様に考えると、OFS1-4の間のトラヒック量が2000を超えるまでは最短経路を使うので、よりホップ数を重視した経路選択をしていると言えます。

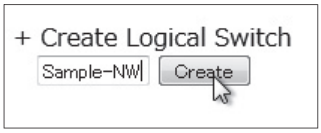
GUIクライアントの操作イメージ

このSample-APには、Sample-APをリモートのブラウザ上から操作するためのSample-GUIがセットになっています。**図10**はSample-GUIの画面イメージです。画面は大きく3つに分かれており、上部がメニュー部分、左下部が選択したメニューに応じた内容が表示されるテキストビュー部分、そして右下部がSample-APが管理するOpenFlowネットワークの物理トポロジと論理トポロジを表示するグラフィカルビュー部分です。以降では、このSample-GUI上から実際に論理的なL2スイッチを作成し、そ

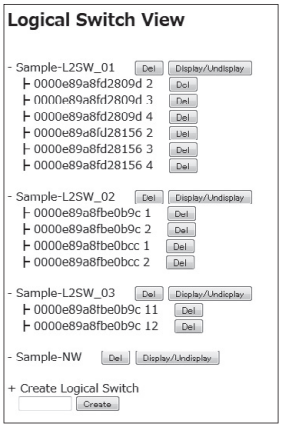
▼図10 Sample-GUI画面イメージ



▼図11　論理L2スイッチの作成



▼図12　論理L2スイッチ追加後のLogical Switch View



▼図13　論理L2スイッチへのホストの追加

▼図14　ホスト追加後の論理L2スイッチ

こにホストを追加する手順を解説します。

◆名前を決めるだけで論理L2スイッチを作成

論理L2スイッチの作成手順は、メニューから[Logical Switch]を選択し、表示されるテキストビューの中の[Create Logical Switch]欄のテキストボックスに任意の論理L2スイッチ名を入力し、[Create]をクリックするだけです(**図11**)。これで、Logical Switch Viewの中に入力した名前の論理L2スイッチが追加されます(**図12**)。

作成した論理L2スイッチを削除する場合には、表示されている論理L2スイッチ名の横にある[Del]ボタンをクリックすればOKです。

◆クリック操作だけで自由にホストを追加／削除

続いて、作成した論理L2スイッチにホストを追加します。Logical Switch Viewの下方に、[Free Hosts]という欄があります。これらが現在いずれの論理L2スイッチにも所属していないホスト(厳密にはネットワークインタフェース)です。ホストは、OpenFlowスイッチの識別子であるデータパスID(16桁の16進数)と、その物理ポート番号で表示されます。たとえば「0011223344aabbcc　10」と表示されている場合、「データパスIDが0011223344aabbccであるOpenFlowスイッチの10番ポートと接続されているホスト(ネットワークインタフェース)」を意味しています。

これらのホストにはチェックボックスが付いているので、論理L2スイッチに追加したいホストを任意の数だけチェックし、最下部にある[Select Switch]欄のプルダウンメニューから所属させる論理L2スイッチ名を選択し[Add]をクリックすればホストの追加が完了します(**図13**)。ホストを追加した直後から、同一の論理L2スイッチに所属するホスト間において通信が可能になります。追加したホストは、Logical Switch Viewにおいて追加された論理L2スイッチ名の下にぶら下がる形で表示され、Free Hosts欄からは削除されます(**図14**)。

追加したホストを論理L2スイッチから削除するには、削除したいホストの横にある[Del]ボタンをクリックします。削除された瞬間、Sample-APは削除されたホストが関連するフローをすべて削除するので、操作直後から削除されたホストは他ホストとの通信ができなくなります。削除されたホストはふたたびFree Hosts欄に表示されます。なお、何らかのホストが接続されている状態の論理L2スイッチを削除すると、その論理L2スイッチに所属していたホストはすべて自動的にFree Hosts欄に戻ります。

以上のように、Sample-APとSample-GUIを用いると、ごく簡単なテキスト入力とクリック操作だけで論理的なL2ネットワークの構築・運用管理ができます。SD

プログラム知識ゼロからはじめる
iPhoneブックアプリ開発

第1回
コード0(ゼロ)で写真集アプリを作ろう!

GimmiQ(ギミック;いたのくまんぼう&リオ・リーバス)
URL http://ninebonz.net/
URL http://www.studioloupe.com/

今号からはじまったiPhoneアプリ開発の入門連載。デジカメで撮った写真や自分で描いたイラスト、お気に入りの画像などを用意すれば、簡単にオリジナルアプリが作れちゃいます! 回を追うごとにブラッシュアップしつつ、アプリ開発の基礎を身につけましょう。

あなたのアプリ開発者人生はここからはじまる

カラーページ(P.4〜5)でも述べたとおり、ここからは実際のブックアプリ開発のはじまりです。今回のチュートリアルでは極力難しいことは避け、本当にシンプルな写真集アプリをとりあえず1本完成させるところをゴールとします。

この連載では、iPhoneアプリを作ろうと思っているけどプログラムはまったくしたことがない人や、書籍を買って勉強してみたけど結局アプリを作ることができなかったような方、時間がなくてなかなか重い腰をあげられなかったエンジニアの方たちにも「完成させる喜び」を味わっていただきたいと思っています。ですので初回のこの記事ではプログラムは一切書かずに写真集アプリを完成させたいと思います。

開発初心者でも迷わずにゴールにたどり着けるはずです。長い連載になりますが、まずは作ったものが動く喜びを感じましょう!!

開発環境を整えよう

ステップ1

まずは開発環境を整えるところからです。第一に、iPhoneを含むiOSアプリ開発にはApple製のMacが1台必要になります。ここ最近のものであればどんなMacでもかまいませんが、OSに関してはMac OS X 10.7.4以降が必要です。すでにこの環境が整っている方はステップ2へ。

ステップ2

次はアプリ開発をするために必要なアプリケーションのダウンロードです。「Dock(ドック)」もしくはメニューバーの「Spotlight」を使って「App Store」を見つけ、開きます(図step2-1、step2-2)。

App Storeが開いたら、右上の検索バーに「Xcode」と入力しましょう。そして、検索結果に出てきた「Xcode」をダウンロードします(図step2-3)。これがAppleが無料で提供している、アプリ開発者向けの開発ツールです。基本的

step2-1

or

step2-2

step2-3

にブックアプリのコンテンツ（画像など）の用意を除くと、このアプリ1つで開発ができてしまいます。

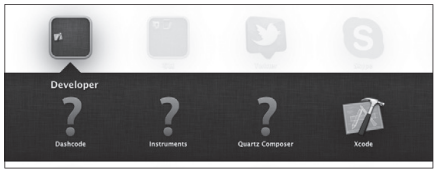
ステップ3

Xcodeがダウンロードできたらさっそく開きましょう。Dockから「LaunchPad」を選択し（図step3-1）、「Developer」フォルダの中にある「Xcode」を選択します（図step3-2）。これから先、使うことが多くなるアプリケーションなのでDockに入れておくと便利だと思います。

Xcodeを開くと、最初に出てくる画面の中から「Create a new Xcode project」を選択します（図step3-3）。

step3-1

step3-2



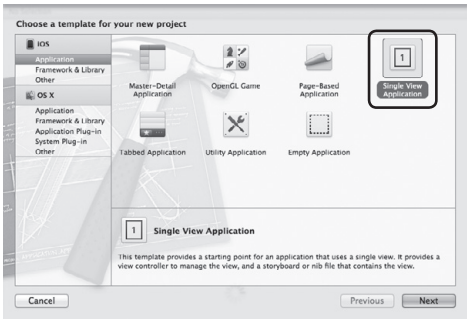
step3-3

プロジェクトの作成

ステップ4

「Create a new Xcode project」をクリックすると図step4-1のようなウィンドウが表示されます。このウィンドウで「Single View Application」アイコンをクリックし選択状態にして、[Next]をクリックします。

step4-1

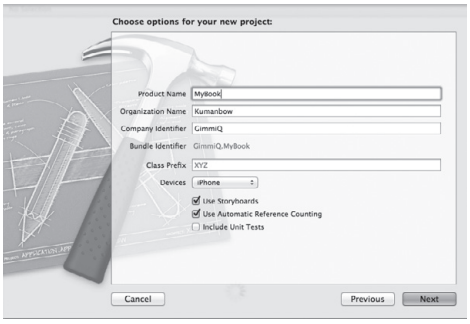


ステップ5

続いて表示されるウィンドウ（図step5-1）ではこれから作るアプリの名前を決めましょう。

「Product Name」欄にアプリ名を入力します。作例では「MyBook」としました。「Organization Name」欄には組織名やご自分の名前を、「Company Identifier」欄には会社名やブランド名をそれぞれ入力します。今回はiPhone用アプリを作るので「Devices」には「iPhone」を指定します。「Use Storyboards」と「Use Automatic Reference Counting」にチェックが入っていることを確認し[Next]をクリックします。

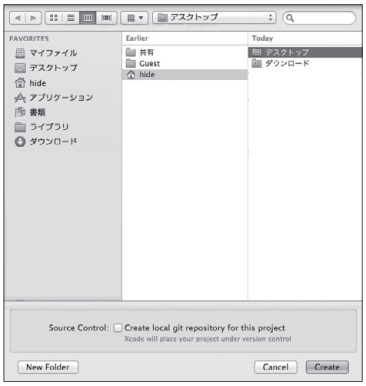
step5-1



ステップ6

プロジェクトファイルを保存する場所を選択し[Create]をクリックします。作例ではデスクトップを指定しました(図step6-1)。図step6-2のウィンドウが表示されれば無事に新規プロジェクトは作成されています。

step6-1



step6-2

基本的な設定

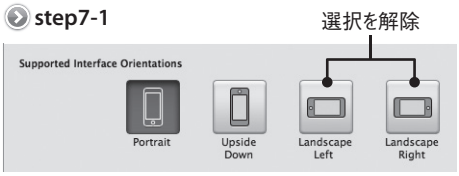
ステップ7

図step6-2の画面で、作成するアプリの初期設定を行います。

まずは対応する画面の向きを設定します。iPhoneアプリでは端末の向きに合わせてアプリの画面を回転する機能がありますが、今回はシンプルなアプリにするため使用しません。そのため図step6-2の中ほどにある「Supported Interface Orientations(アプリが対応する画面の向き)」を図step7-1のように設定します。

初期状態では「Portrait」「Landscape Left」「Landscape Right」が選択された状態になってるので、「Landscape Left」と「Landscape Right」をクリックし非選択の状態にします。これで、このアプリが対応する画面の向きはホームボタンを下にした縦の正位置の対応のみになりました。

step7-1



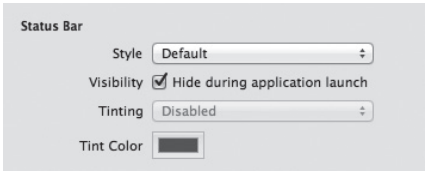
ステップ8

続いてステータスバーを非表示にします。ステータスバーとはiPhoneの画面上部に表示されている時間やバッテリー残量が表示されている部分です(図step8-1)。写真集アプリでこの部分が表示されたままですと閲覧の邪魔にもなりますし、写真の雰囲気を壊してしまうかもしれませんので、今回のアプリでは非表示にしたいと思います。

「Status Bar」の設定項目にある「Visibility」にチェックを入れます(図step8-2)。これで「Hide during application launch(アプリ起動中はステータスバー非表示)」になりました。

step8-1

step8-2

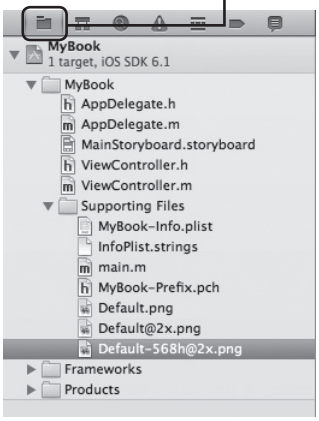


ステップ9

次に対応する画面サイズの設定を行います。現在iPhoneには2種類の画面サイズが存在します。iPhone 5に採用されている4インチディスプレイ、iPhone 4S以前の機種に採用されている3.5インチディスプレイです。これからの作業を簡単にするために、今回はより多くの機種に採用されている3.5インチのみ対応とします。

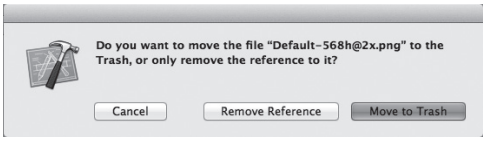
Xcodeで作ったプロジェクトは初期状態で4インチと3.5インチの両方に対応した状態ですので、3.5インチ対応のみに変更します。4インチ非対応にするには、Xcodeの左カラムにあるフォルダー一覧「Project Navigator（プロジェクトナビゲーター）」エリアから[MyBook]－[Supporting Files]にある「Default-568h@2x.png」という名前のファイルを削除します（図step9-1）。

step9-1 Project Navigator



このファイルをクリックし選択状態にした後にdeleteキーを押して削除します。削除の際に図step9-2のようなダイヤログが出ますので「Move to Trash（ゴミ箱へ移動）」をクリックします。

step9-2



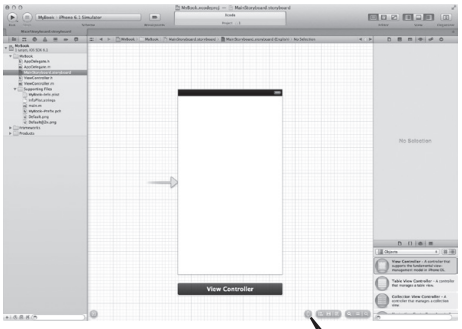
「Default〜.png」というファイルはアプリ起動時に表示される画像です。プロジェクトの初期状態で黒色で塗りつぶされた画像が各ディスプレイごとに用意されているのですが、「Default-568h@2x.png」が4インチ用の画像になります。この画像の有無で4インチ対応アプリか否かを判断していますので、この画像を削除することによって4インチ非対応にすることができます。

ステップ10

最後にStoryBoardの設定をしましょう。StoryBoard（ストーリーボード）とはiOS 5から採用された新機能で、画面の移行などをグラフィカルに作成できるものです。今回はこのストーリーボードのみを使ってアプリを作成します。

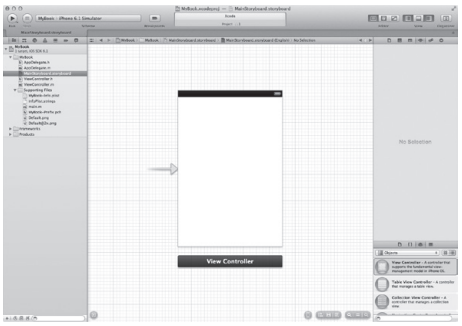
プロジェクトナビゲーターの[MyBook]内にある「MainStoryboard.storyboard」をクリックすると図step10-1のようにストーリーボード画面になります。すでにストーリーボード上には1つ画面が配置されていますが、4インチサイズになっていますので3.5インチのものに切り替えます。

step10-1



step10-2

step10-3



中央カラムの右下に並んだアイコンから画面サイズ切り替えアイコン（図step10-2）をクリックします。ストーリーボード上に配置されていた画面の大きさが少し小さくなって、3.5インチ用のストーリーボードになったのがわかるかと思います（図step10-3）。

以上で基本的な設定は終わりです。次からいよいよアプリ自体を作成します。

1ページ目を作る

ステップ11

まずは、写真集アプリに使う画像をプロジェクトに追加します。

用意する画像のサイズは3.5インチディスプレイ用なので320×480ドットです。実は3.5インチディスプレイにも非Retinaディスプレイ(320×480ドット)とRetinaディスプレイ(640×960ドット)の2種類ありますが、今回はシミュレータで動作確認するのと、話をシンプルにするために非Retinaディスプレイのみ対応とします。

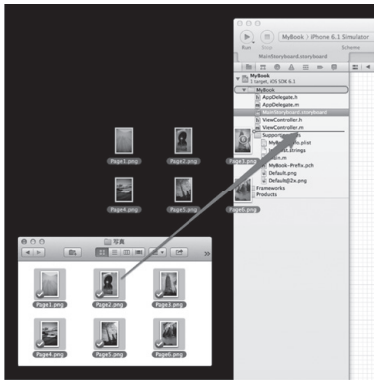
アプリで表示したい写真を320×480ドットにリサイズして、PNGもしくはJPGフォーマットで用意してください。作例ではPNGフォーマットの写真を6枚用意しました。ファイルネームは「Page1.png〜Page6.png」としました。

用意した画像をプロジェクトに追加します。画像ファイルをドラッグして、Xcodeのプロジェクトナビゲーターまで移動します(図step11-1)。プロジェクトナビゲーターにファイルをドロップすると図step11-2のようなウィンドウが表示されます。

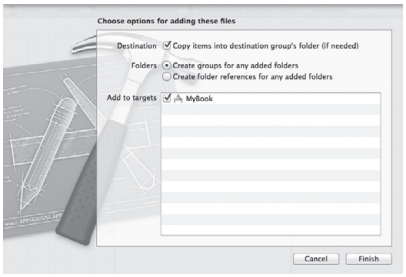
「Destination」と「Add to targets」の項目にチェックを入れ[Finish]をクリックします。

プロジェクトナビゲーターに画像ファイルが追加されていることと思います(図step11-3)。ステップ6で作成したプロジェクトのフォルダ(作例では「MyBook」)内に画像ファイルがコピーされていることを確認してください。

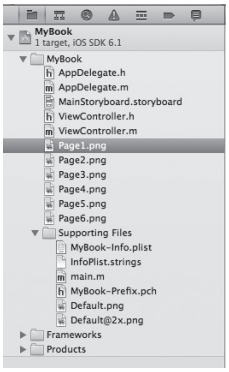
step11-1



step11-2



step11-3



ステップ12

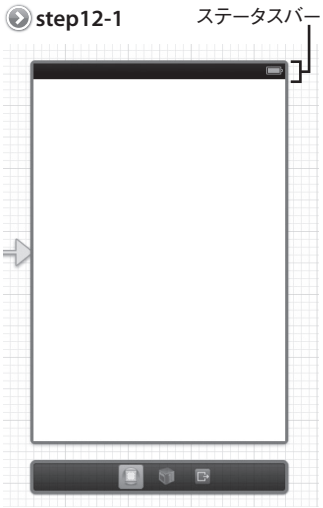
プロジェクトナビゲーターの「MainStoryboard.storyboard」をクリックしストーリーボード画面に戻り(切り替え)ます。

ストーリーボードには1ページ目となる画面の「ViewController」が表示されています(図step12-1)。ViewControllerとは画面を管理するものだと考えてください。基本的にiOSアプリではViewController単位で画面を管理します。

この1ページ目のViewControllerをよく見ると画面上部にステータスバー表示領域が確保されたままとなっています。まずは、これを取り去りましょう。

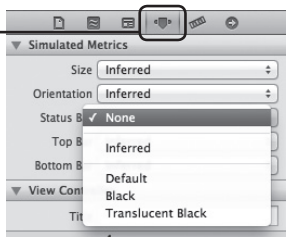
Xcodeの右カラムにある「Attributes inspector(アトリビュートイ

step12-1



ンスペクター）」エリアにある「Status Bar」の項目をクリックし、リストから「None」を選択してください（図step12-2）。ステータスバーがなくなり表示領域が広がったのが確認できればOKです。

step12-2

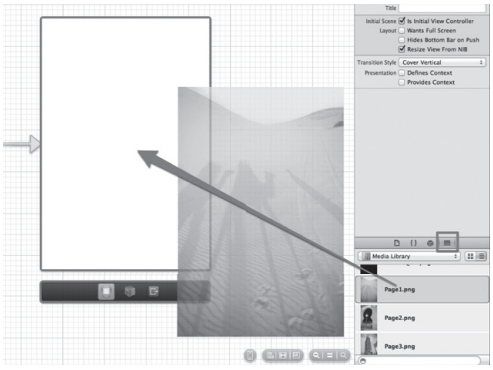


ステップ13

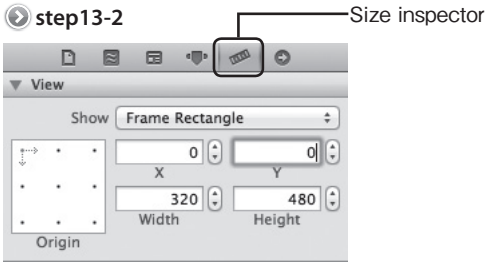
それでは、いよいよ1ページ目の写真をページに貼り付けます。Xcodeの右下カラムの「Media library」エリアから、1ページ目の画像をViewControllerまでドラッグ＆ドロップします（図step13-1）。

貼り付ける位置と大きさはある程度自動で調節されますが、正しい位置に貼り付けられているか確認します。ViewControllerに貼り付けた画像をクリックし選択状態にしたまま、Xcodeの右カラムの「Size inspector（サイズインスペクター）」のアイコンをクリックすると画像の位置（X、Y座標）と大きさ（Width：幅、Height：高さ）が確認できます。図step13-2と同じ値になっているか確認してください。なっていなければこれらの数値を直接編集して修正します。

step13-1



step13-2



ステップ14

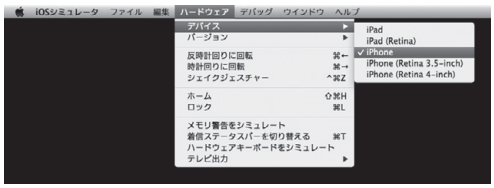
ここまでできたところで一度実行してiOSシミュレータで動かしてみましょう。

Xcodeのウィンドウ左上にある「Run（実行）」のアイコンをクリックしましょう（図step14-1）。図step14-2のように表示されたでしょうか？　シミュレータの枠が図step14-2と違うものが表示された場合はiOSシミュレータのデバイス設定を変更してください。一度Xcodeの「Stop（停止）」ボタンを押してシミュレータの実行を止め、メニューバーの［ハードウエア］－［デバイス］－［iPhone］と選択してください（図step14-3）。これでデバイス設定が3.5インチ、非Retinaディスプレイ搭載のiPhoneに設定されました。

step14-1

step14-2

step14-3



ページを増やして画面切り替えボタンを付ける

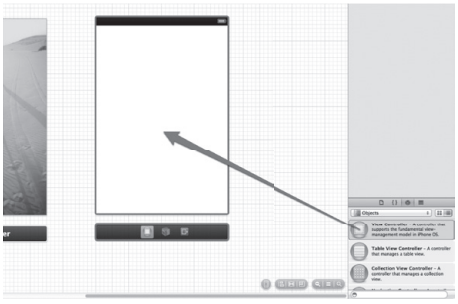
ステップ15

それでは、この調子でページ数を増やしていきましょう。Xcode右下カラムの「Object library」エリア（図step15-1）から「ViewController」をストーリーボードの1ページ目の横の位置にドラッグ＆ドロップします（図step15-2）。

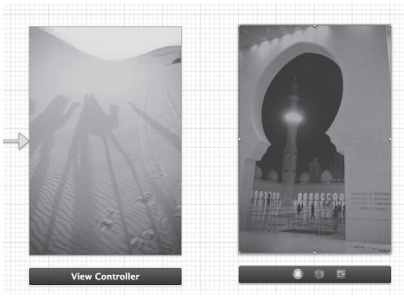
このViewControllerが2ページ目になります。ステップ12、ステップ13で行ったのと同じようにこのViewControllerのステータスバー表示領域を取り去り、写真を貼り付けます。これで2ページ目もできました（図step15-3）。

step15-1

step15-2



step15-3

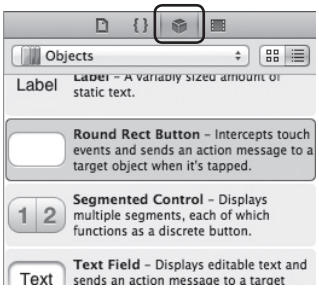


ステップ16

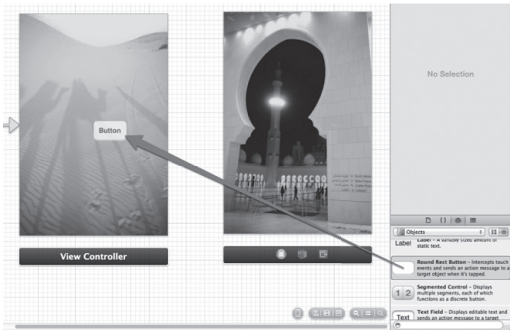
ページを移動するためのボタンを設置します。ステップ15で使ったObject libraryから「Round Rect Button」（図step16-1）を1ページ目にドラッグ＆ドロップします（図step16-2）。Round Rect Buttonはその名のとおりボタンパーツで、タップすることによってさまざまなアクションを起こすことができます。今回はこのボタンを押すことで次のページに行くようにします。

画面全体をボタンの反応領域としたいと思いますので、画面全体を覆うボタンにします。貼り付けたRound Rect Buttonをクリックして選択状態にし、右下カラムのサイズインスペクターのアイコンをクリックします。図step16-3と同じ値になるように各値を入力してください。

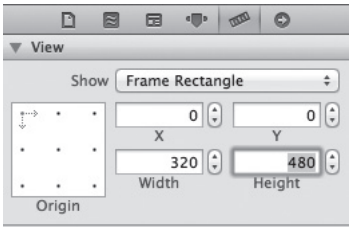
step16-1



step16-2



step16-3

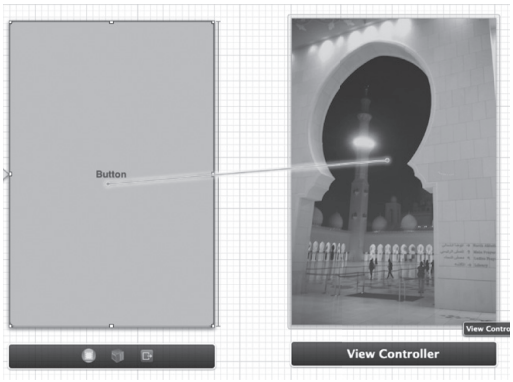


ステップ17

1ページ目と2ページ目をつなげます。画面を覆う状態になった1ページ目のボタンをcontrolキーを押しながらドラッグすると、ボタンから青い線が延びます。この線が画面のつなぎを表しています。

ドラッグしたまま2ページ目まで線を引っ張り、2ページ目が青い枠で囲まれたらマウスのボタンを放してください（図step17-1）。すると図step17-2のようなメニューが出ますので、「modal」をクリックします。これでページがつながりました。1ページ目と2ページ目のViewControllerの間にラインが引かれたことを確認してください（図step17-3）。

step17-1



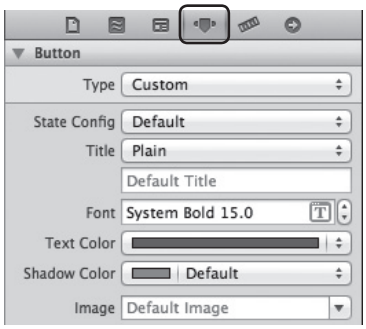
step17-2

step17-3

ステップ18

1ページ目の写真がボタンで隠れてしまっていますので、ボタンを透明にします。ボタンをクリックして選択状態にし、右カラムのアトリビュートインスペクターアイコンをクリックします（図step18-1）。「Type」欄を「Custom」に設定し、「Title」欄に入力されている「Button」という文字列を削除します。これでボタンが透明になったと思います。

step18-1



ステップ19

では、実行してページの移動を確認します。「Run」ボタンをクリックし、実行されたiOSシミュレータの画面部分をクリックしてみましょう。無事に2ページ目に移動したでしょうか？

ステップ20

今回覚えなければいけないことはこれですべてです。あとは、ステップ15からステップ18を繰り返してページを増やしていけば(P.4〜5にある一番下の図)、立派な写真集アプリになることでしょう。

第1回のまとめ

いかがでしたか？　まだこれはほんの触りの部分とはいえ、単純に画像を切り替えていくアプリを作るだけならこんなにも簡単にできてしまいます。ここに自分で用意した画像をはめ込んでいけば、自分だけの写真集アプリにできるでしょう。

もちろん、もっと細かく操作にこだわったり、機能を追加したいという要望もあるでしょう。また、今回の作例ではページを戻ることができなかったり、メモリ管理などの課題も残されています。これからの連載ではそのような要望に応えられるように、1回1回、新しいことを覚えつつ、より高機能で洗礼されたブックアプリに仕上げていくことがテーマです。次回からは少しづつプログラムも書きはじめ、回を重ねるごとにインタラクティブ性の高い本の作り方などにも触れていく予定です。

また、連載記事にはページの制限があることと、連載の途中から読みはじめた方でも参加しやすいように、本連載用の専用「まとめサイト」を開設しました。ここでは基本的な環境の準備から、開発者登録の手順、シミュレータや実機を使ったテストの方法、そしてApp Storeでアプリを販売するためにサブミット(Appleに審査をしてもらうために提出)する流れなど、連載では書ききれない部分をまとめていきます。今回作ったアプリの復習がしたいという方がいれば、さっそく次のアドレスからアクセスしてみてください。

URL http://www.gimmiq.net/p/sd.html

iOSは毎年のようにOSがメジャーアップデートされるので、それに伴って開発ソフトも変化することがあります。ですから、これから1年の連載期間中に開発環境の変化が起こることも十分に考えられます。本誌では一度掲載してしまった後では開発手順などに大きな変更があったとしても修正はできませんが、「まとめサイト」のほうではなるべく変化に合わせて最新の環境での情報へと更新していく予定なので、ぜひご利用いただければと思います。

全12回で伝えたいことのすべてを詰め込むことは困難ではありますが、少なくともこの全12回のチュートリアルをこなしたあとには、他のiOS開発本を手に取ったときに最初のページから何が何だかわからない、ということにならないくらいの知識を身につけてほしいと願います。脱・初心者を目指して頑張りましょう！SD

●いたのくまんぼう／Itano Kumanbow　Twitter @Kumanbow

神奈川工科大学非常勤講師。リオさんとはGimmiQ名義で「MagicReader」(手を使わずにページがめくれる電子書籍ビューワ)をリリース。個人ではNinebonz名義で「Crop It Cam!」(おしゃれな切り抜き写真カメラ)、「i列車の車窓から—そうだ!京都行こう!—」(バーチャル旅行アプリ)など。アプリ紹介サイト「あぷまがどっとねっと(http://appmaga.net/)」の技術サポータ。

●リオ・リーバス／Leo Rivas　Twitter @StudioLoupe

iOSアプリ開発を中心に電子絵本作家・漫画家として活動中。個人ではスタジオルーペとして数字を指でドラッグ＆ドロップ保存できる「フュージョン計算機(FusionCalc)」が代表作。電子絵本はiBookstore/Kindleストア共に児童書カテゴリ総合1位を獲得。現在HPにてWeb漫画「HELL BASEBALL」を連載中。

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

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

2013年4月号

第1特集
僕（私）の言語の学び方
裏口からのプログラミング入門

第2特集
オブジェクト指向再入門
ソフトウェア開発に効くSmall Objectをご存じですか?

特別企画
・スクウェア・エニックス＋Skeed
「ゲーム開発の舞台裏」

1,280円

2013年3月号

第1特集
オープン環境でスキルアップ!
もっとクラウドを活用してみませんか?

第2特集
光、ギガビット、高速ネットワークを体験!
実践! ワイヤリングの教科書

一般記事
・「SSDストレージ」爆発的普及の理由

1,280円

2013年2月号

第1特集
UNIXコマンド、fork、pipeを復習し、高度なスクリプティングへ
シェルスクリプティング道場

第2特集
忙しいITエンジニアのための
超効率的勉強法

一般記事
・Samba 4.0.0ファーストインプレッション

1,280円

2013年1月号

第1特集
いざというときに備える
システムバックアップ

第2特集
IT業界のキーパーソンに聞く
2013年に来そうな「技術」・「ビジョン」はこれだ!

特別付録
法輪寺電電宮情報安全護符シール

1,380円

2012年12月号

第1特集
判断をあおぐ／経緯を説明する／手順の理解を得る文章を書くためのアタマの整理術
なぜエンジニアは文章が下手なのか?

第2特集
高速・高機能HTTPサーバ
Nginx構築・設定マニュアル

一般記事
・エブリデープログラマの発想と実践

1,280円

2012年11月号

第1特集
もし、OpenFlowでやれと言われたら?
SDN、仮想化でネットワークはどうなる

第2特集
サーバの運用支援に
グラフィカルなリソース監視ツールを!
Muninが手放せない理由

一般記事
・SkeedSilverBulletとは?

1,280円

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

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

デジタル版のお知らせ　DIGITAL

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

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

ハイパーバイザの作り方

ちゃんと理解する仮想化技術

第8回

Intel VT-xを用いたハイパーバイザの実装その3「vmm.koによるVMEntry」

浅田 拓也（ASADA Takuya）Twitter @syuu1228

はじめに

前回は、/usr/sbin/bhyveの初期化とVMインスタンスの実行機能の実装について解説しました。今回はvmm.koがVM_RUN ioctlを受け取ってからVMEntryするまでの処理を解説します。

解説対象のソースコードについて

本連載では、FreeBSD-CURRENTに実装されているBHyVeのソースコードを解説しています。このソースコードは、FreeBSDのSubversionリポジトリから取得できます。リビジョンはr245673を用いています。

お手持ちのPCにSubversionをインストールし、次のようなコマンドでソースコードを取得してください。

```
svn co -r245673 svn://svn.freebsd.org/base/head src
```

/usr/sbin/bhyveによる仮想CPUの実行処理の復習

/usr/sbin/bhyveは仮想CPUの数だけスレッドを起動し、それぞれのスレッドが/dev/vmm/${name}に対してVM_RUN ioctlを発行します（**図1**）。vmm.koはioctlを受けてCPUをVMX non root modeへ切り替えゲストOSを実行します（VMEntry）。

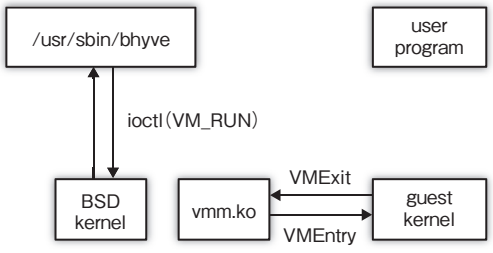
VMX non root modeでハイパーバイザの介入が必要な何らかのイベントが発生すると制御がvmm.koへ戻され、イベントがトラップされます（VMExit）。

イベントの種類が/usr/sbin/bhyveでハンドルされる必要のあるものだった場合、ioctlはリターンされ、制御が/usr/sbin/bhyveへ移ります。イベントの種類が/usr/sbin/bhyveでハンドルされる必要のないものだった場合、ioctlはリターンされないままゲストCPUの実行が再開されます。今回の記事では、vmm.koにVM_RUN ioctlが届いてからVMX non root modeへVMEntryするまでを見ていきます。

vmm.koがVM_RUN ioctlを受け取ってからVMEntryするまで

vmm.koがVM_RUN ioctlを受け取ってからVMEntryするまでの処理について、順を追って見ていきます。**リスト1**、**リスト2**、**リスト3**、**リスト4**にソースコードを示します。白丸の数字と黒丸の数字がありますが、ここでは白丸の数字を追って見ていきます。

▼図1　VM_RUN ioctlによる仮想CPUの実行イメージ



VMExit時の再開アドレス

⑲でCPUはVMX non-root modeへ切り替わりゲストOSが実行されますが、ここからVMExitした時にCPUはどこからホストOSの実行を再開するのでしょうか。直感的にはvmlaunchの次の命令ではないかと思いますが、そうではありません。VT-xでは、VMEntry時にVMCSのGUEST_RIPからVMX non-root modeの実行を開始し、VMExit時にVMCSのHOST_RIPからVMX root modeの実行を開始することになっています。GUEST_RIPはVMExit時に保存されますが、HOST_RIPはVMEntry時に保存されません。

このため、VMCSの初期化時に指定されたHOST_RIPが常にVMExit時の再開アドレスとなります。では、VMCSのHOST_RIPがどのように初期化されているか、順を追って見ていきます。**リスト1**、**リスト2**、**リスト3**、**リスト4**にソースコードを示します。今度は黒丸の数字を追って見ていきます。

リスト1の解説

vmm_dev.cは、sysctlによる/dev/vmm/${name}の作成・削除と/dev/vmm/${name}に対するopen(), close(), read(), write(), mmap(), ioctl()のハンドラを定義しています。ここでは/dev/vmm/${name}の作成とVM_RUN ioctlについてのみ見ていきます。

リスト2の解説

vmm.cは、Intel VT-xとAMD-Vの2つの異なるハードウェア仮想化支援機能のラッパー関数を提供しています(このリビジョンではラッパーのみが実装されており、AMD-Vの実装は行われていません)。Intel／AMD両アーキテクチャ向けの各関数はvmm_ops構造体で抽象化され、207～210行目のコードでCPUを判定してどちらのアーキテクチャの関数群を使用するかを決定しています。

リスト3の解説

intel/ディレクトリにはIntel VT-xに依存したコード群が置かれています。vmx.cはその中心的なコードで、vmm.cで登場したvmm_ops_intelもここで定義されています。

▼リスト1　sys/amd64/vmm/vmm_dev.c

```
  ……(省略)……
137: static int
138: vmmdev_ioctl(struct cdev *cdev, u_long cmd, caddr_t data, int fflag,
139: struct thread *td)                    ←①vmmdev_ioctlは/dev/vmm/${name}のioctlハンドラ
140: {
  ……(省略)……
234:   switch(cmd) {
235:   case VM_RUN:                        ←②/usr/sbin/bhyveから送られてきたVM_RUN ioctlを受け取る
236:     vmrun = (struct vm_run *)data;
237:     error = vm_run(sc->vm, vmrun);     ←③vm_run()を呼ぶ
238:     break;
  ……(省略)……
371:   }
  ……(省略)……
382: }
  ……(省略)……
471: static int
472: sysctl_vmm_create(SYSCTL_HANDLER_ARGS)  ←❶hw.vmm.create sysctlにより/dev/vmm/${name}を作成しVMインスタンスを初期化する時に呼び出される
473: {
  ……(省略)……
490:   vm = vm_create(buf);                 ←❷vm_createを呼び出す
  ……(省略)……
517: }
```

▼リスト2　sys/amd64/vmm/vmm.c

```
  ……(省略)……
119: static struct vmm_ops *ops;
  ……(省略)……
123: #define VMINIT(vm) (ops != NULL ? (*ops->vminit)(vm): NULL)

124: #define VMRUN(vmi, vcpu, rip) \
125:   (ops != NULL ? (*ops->vmrun)(vmi, vcpu, rip) : ENXIO)
  ……(省略)……
195: static int
196: vmm_init(void)
197: {
  ……(省略)……
207:   if (vmm_is_intel())
208:     ops = &vmm_ops_intel;
209:     else if (vmm_is_amd())
210:       ops = &vmm_ops_amd;
  ……(省略)……
216:     return (VMM_INIT());
217: }
  ……(省略)……
261: struct vm *
262: vm_create(const char *name)
263: {
  ……(省略)……
275:   vm->cookie = VMINIT(vm);
  ……(省略)……
287: }
  ……(省略)……
672: int
673: vm_run(struct vm *vm, struct vm_run *vmrun)
674: {
  ……(省略)……
681:   vcpuid = vmrun->cpuid;
  ……(省略)……
688:   rip = vmrun->rip;
  ……(省略)……
701:   error = VMRUN(vm->cookie, vcpuid, rip);
……(省略)……
757: }
```

❹VMINIT()マクロはvmm_ops構造体上の関数ポインタops->vminitを呼び出す

⑤VMRUN()マクロはvmm_ops構造体上の関数ポインタops->vmrunを呼び出す

❸vm_create()はVMINIT()マクロを呼び出す。

④vm_run()はVMRUN()マクロを呼び出す。

▼リスト3　sys/amd64/vmm/intel/vmx.c

```
  ……(省略)……
666: static void *
667: vmx_vminit(struct vm *vm);
668: {
  ……(省略)……
725:   for (i = 0; i < VM_MAXCPU; i++) {
  ……(省略)……
735:     error = vmcs_set_defaults(&vmx->vmcs[i],
736:             (u_long)vmx_longjmp,
737:             (u_long)&vmx->ctx[i],
738:             vtophys(vmx->pml4ept),
739:             pinbased_ctls,
740:             procbased_ctls,
741:             procbased_ctls2,
742:             exit_ctls, entry_ctls,
743:             vtophys(vmx->msr_bitmap),
744:             vpid);
```

❻結果として、vm_createはvmx_vminitが呼び出される

❼vmx_vminitでは仮想CPUごとのVMCSの初期化処理が行われる。その過程でいくつかの関数が呼び出されるが、ここではvmcs_set_defaultsにのみ着目する

❽vmcs_set_defaultsの第2引数にvmx_longjmpを指定して呼んでいる

```
  ……(省略)……
770: }
771:
772: return (vmx);
773: }
  ……(省略)……
1354: static int
1355: vmx_run(void *arg, int vcpu, register_t rip)
1356: {
  ……(省略)……
1366:    vmxctx->launched = 0;
  ……(省略)……
1375:    VMPTRLD(vmcs);
  ……(省略)……
1385:    if ((error = vmwrite(VMCS_HOST_CR3, rcr3())) != 0)
1386:      panic("vmx_run: error %d writing to VMCS_HOST_CR3", error);
1387:
1388:    if ((error = vmwrite(VMCS_GUEST_RIP, rip)) != 0)
1389:      panic("vmx_run: error %d writing to VMCS_GUEST_RIP", error);
1390:
1391:    if ((error = vmx_set_pcpu_defaults(vmx, vcpu)) != 0)
1392:      panic("vmx_run: error %d setting up pcpu defaults", error);
1393:
1394:    do {
1395:      lapic_timer_tick(vmx->vm, vcpu);
1396:      vmx_inject_interrupts(vmx, vcpu);
1397:      vmx_run_trace(vmx, vcpu);
1398:      rc = vmx_setjmp(vmxctx);
  ……(省略)……
1402:      switch (rc) {
1403:      case VMX_RETURN_DIRECT:
1404:        if (vmxctx->launched == 0) {
1405:          vmxctx->launched = 1;
1406:          vmx_launch(vmxctx);
1407:        } else
1408:          vmx_resume(vmxctx);
1409:        panic("vmx_launch/resume should not return");
1410:        break;
……(省略)……
1458:    } while (handled);
……(省略)……
1480:    VMCLEAR(vmcs);
1481:    return (0);
……(省略)……
1490: }
……(省略)……
1830: struct vmm_ops vmm_ops_intel = {
1831:  vmx_init,
1832:  vmx_cleanup,
1833:  vmx_vminit,
1834:  vmx_run,
1835:  vmx_vmcleanup,
1836:  ept_vmmmap_set,
1837:  ept_vmmmap_get,
1838:  vmx_getreg,
1839:  vmx_setreg,
1840:  vmx_getdesc,
1841:  vmx_setdesc,
1842:  vmx_inject,
1843:  vmx_getcap,
1844:  vmx_setcap
1845: };
```

⑦結果として、VM_RUN ioctlはvmx_run内で実際に処理される

⑧VMCSにゲストのRIPレジスタをセットする

⑨ユーザランドでハンドルしなければならないイベントが来るまでカーネル内でループし、VMEntryを繰り返す

⑩アセンブリコードのvmx_setjmpを呼び出し、vmxctxへホストレジスタを退避、VMX_RETURN_DIRECTがリターンされる

⑫rcはVMX_RETURN_DIRECT

⑬1度目のループではlaunched == 0でvmx_launchが呼ばれる。二度目以降はvmx_resumeが呼ばれる

⑭アセンブリコードのvmx_launchを呼び出してVMX non-root modeへCPUを切り替える

❺ops->vminitはvmx_vminitを指している

⑥ops->vmrunはvmx_runを指している

リスト4の解説

vmcs.cはVMCSの設定に関するコードを提供しています。ここではHOST_RIPの書き込みに注目しています。なお、vmwriteを行う前にvmptrld命令を発行していますが、これはCPUへVMCSポインタをセットしてVMCSへアクセス可能にするためです。同様に、vmwriteを行った後にvmclear命令を発行していますが、これは変更されたVMCSをメモリへライトバックさせるためです。

リスト5の解説

vmx_support.SはC言語で記述できない、コンテキストの退避／復帰やVT-x拡張命令の発行などのコードを提供しています。ここでは、ホストレジスタの退避（vmx_setjmp）とVMEntry（vmx_launch）の処理について見ています。

▼リスト4　sys/amd64/vmm/intel/vmcs.c

```
  …… (省略) ……
309: int
310: vmcs_set_defaults(struct vmcs *vmcs,
311:   u_long host_rip, u_long host_rsp, u_long ept_pml4,
312:   uint32_t pinbased_ctls, uint32_t procbased_ctls,
313:   uint32_t procbased_ctls2, uint32_t exit_ctls,
314:   uint32_t entry_ctls, u_long msr_bitmap, uint16_t vpid)
315: {
  …… (省略) ……
325:   /*
326:   * Make sure we have a "current" VMCS to work with.
327:   */
328:   VMPTRLD(vmcs);
  …… (省略) ……
416:   /* instruction pointer */
417:   if ((error = vmwrite(VMCS_HOST_RIP, host_rip)) != 0)
418:     goto done;
…… (省略) ……
446:   VMCLEAR(vmcs);
447:   return (error);
448: }
```

❾VMCSのHOST_RIPにhost_ripを指定している。vmx_vminitではvmx_longjmpが指定されているため、結果としてVMExitすると常にvmx_longjmpから再開されることとなる

▼リスト5　sys/amd64/vmm/intel/vmx_support.S

```
  …… (省略) ……
69: #define VMX_GUEST_RESTORE     \
70:   movq %rdi,%rsp;          \
71:   movq VMXCTX_GUEST_CR2(%rdi),%rsi; \
72:   movq %rsi,%cr2;           \
73:   movq VMXCTX_GUEST_RSI(%rdi),%rsi; \
74:   movq VMXCTX_GUEST_RDX(%rdi),%rdx; \
75:   movq VMXCTX_GUEST_RCX(%rdi),%rcx; \
76:   movq VMXCTX_GUEST_R8(%rdi),%r8;   \
77:   movq VMXCTX_GUEST_R9(%rdi),%r9;   \
78:   movq VMXCTX_GUEST_RAX(%rdi),%rax; \
79:   movq VMXCTX_GUEST_RBX(%rdi),%rbx; \
80:   movq VMXCTX_GUEST_RBP(%rdi),%rbp; \
81:   movq VMXCTX_GUEST_R10(%rdi),%r10; \
82:   movq VMXCTX_GUEST_R11(%rdi),%r11; \
83:   movq VMXCTX_GUEST_R12(%rdi),%r12; \
84:   movq VMXCTX_GUEST_R13(%rdi),%r13; \
85:   movq VMXCTX_GUEST_R14(%rdi),%r14; \
86:   movq VMXCTX_GUEST_R15(%rdi),%r15; \
87:   movq VMXCTX_GUEST_RDI(%rdi),%rdi; /* restore rdi the last */
88:
```

⓱vmxctx上のゲストレジスタ値を実レジスタに展開。これらのレジスタはVMEntry/VMExit時にVMCSからセーブ・リストアされないのでハイパーバイザがVMCSとは別の領域にセーブ・リストアする必要がある。BHyVeではこれにvmxctx構造体が用いられている

```
89: #define VM_INSTRUCTION_ERROR(reg) \
90:  jnc 1f;           \
91:  movl $VM_FAIL_INVALID,reg; /* CF is set */ \
92:  jmp 3f;           \
93: 1: jnz 2f;          \
94:  movl $VM_FAIL_VALID,reg; /* ZF is set */ \
95:  jmp 3f;           \
96: 2: movl $VM_SUCCESS,reg;     \
97: 3: movl reg,VMXCTX_LAUNCH_ERROR(%rsp)
  …… (省略) ……
107: ENTRY(vmx_setjmp)
108:   movq (%rsp),%rax /* return address */
109:   movq %r15,VMXCTX_HOST_R15(%rdi)
110:   movq %r14,VMXCTX_HOST_R14(%rdi)
111:   movq %r13,VMXCTX_HOST_R13(%rdi)
112:   movq %r12,VMXCTX_HOST_R12(%rdi)
113:   movq %rbp,VMXCTX_HOST_RBP(%rdi)
114:   movq %rsp,VMXCTX_HOST_RSP(%rdi)
115:   movq %rbx,VMXCTX_HOST_RBX(%rdi)
116:   movq %rax,VMXCTX_HOST_RIP(%rdi)
117:
118:   /*
119:   * XXX save host debug registers
120:   */
121:   movl $VMX_RETURN_DIRECT,%eax
122:   ret
123: END(vmx_setjmp)
  …… (省略) ……
223: ENTRY(vmx_launch)
224:   VMX_DISABLE_INTERRUPTS
225:
226:   VMX_CHECK_AST
227:
228:   /*
229:   * Restore guest state that is not automatically loaded from thevmcs.
230:   */
231:   VMX_GUEST_RESTORE
232:
233:   vmlaunch
234:
235:   /*
236:   * Capture the reason why vmlaunch failed.
237:   */
238:   VM_INSTRUCTION_ERROR(%eax)
239:
240:   /* Return via vmx_setjmp with return value of VMX_RETURN_VMLAUNCH */
241:   movq %rsp,%rdi
242:   movq $VMX_RETURN_VMLAUNCH,%rsi
243:
244:   addq $VMXCTX_TMPSTKTOP,%rsp
245:   callqvmx_return
246: END(vmx_launch)
```

⑪vmctx（VMCSではない）にホストレジスタを退避してVMX_RETURN_DIRECTをリターンする

⑮ゲストレジスタをリストアしてVMEntryする

⑯ゲストレジスタリストアを行うマクロを展開

⑱vmlaunch命令でVMX non-root modeへVMEntryする

⑲ここにはVMEntryが失敗したときのみ到達する。正常にVMEntryしたのちVMExitした場合は、前述のとおりvmx_longjmpへジャンプする

まとめ

vmm.koがVM_RUN ioctlを受け取ってからVMEntryするまでにどのような処理が行われているかについて、ソースコードを解説しました。次回はこれに対応するVMExitの処理について見ていきます。SD

テキストデータならお手のもの

開眼シェルスクリプト

㈲ユニバーサル・シェル・プログラミング研究所　http://www.usp-lab.com
上田 隆一　UEDA Ryuichi　Twitter @uecinfo

第17回 端末上で扱いづらいテキストの対処法――AWKで乗り切れ!

はじめに

今回は、作りものを休んでテキスト処理のパズルを扱います。内容はUSP友の会の「シェル芸勉強会」で出題しているような問題です。もし勉強会にご興味があれば、筆者自らビシビシ鍛えますので、ぜひ遊びに来てください。参加費500円也。

そのシェル芸勉強会でも本連載でも再三にわたって訴えていることですが、シェルでコマンドの使い方を覚えると、わざわざプログラム(シェルスクリプトを含む)を書いたりせず、電卓を叩くようにテキスト処理ができるようになります。

ただし、端末上でテキストをいじることには1つの「壁」があります。その壁とは、「本当に自分の欲しい出力が得られるのだろうか?」という恐怖というか、時間を無駄にするかもしれないことに対する抵抗感です。UNIXができた当初ならいざ知らず、今はいろんな「逃げ方」があります。抵抗感を持つと、筆者が「この方法が一番手っ取り早い」と言っても、なかなかやってみようという気にはならないでしょう。

この壁を少しでも低くするために、筆者は度々AWKを使ってもらうように宣伝します。AWKの場合、たとえば`cat hoge | awk '{print $1}'`だけでも役立つ場面が多いので、シェルスクリプトが云々と言うよりとっかかりが早いのです。

すでに「こちらの世界」に入ってしまった人にとっても、AWKのスキルはツブしを効かせることに絶大な効果があります。コマンドを忘れても、ややこしいテキストを処理しだして袋小路に追い込まれても、AWKで力ずくで押し切ってしまう腕があれば、HDDのゴミになるようなプログラムなど書く気にもならなくなるでしょう。

ということで、今回は比較的端末上で扱いづらいテキストを、AWKで乗り切る方法について扱いたいと思います。

毎回定番の格言ですが、「壁」と言えばこの方ですので、学習のヒントとして挙げておきます。

- すでにやってしまった以上は、その結果がよいほうに向かうように、あとの人生を動かすしかない。――養老孟司
- すでに端末上でやってしまった以上は、その結果がよいほうに向かうように、AWKを動かすしかない。――筆者

実行環境

今回も前回に引き続き、Macを使っています。**図1**に環境を示します。ただ、今回も環境が違うからと言ってそんなに困ることはないと思います。むしろ、どの問題もいろんな処理の方法があるので、うまくいかなかったら別の方法を試してみましょう。

問題4で必要ですので、Macにデフォルトで入っているsedのほかに、gsedをインストールしました。Macの人はmacportsやhomebrewを

使ってインストールしてみてください。FreeBSDの場合も、portsなどでインストールする必要があります。Linuxの場合は、sedでそのまま問題なく日本語を処理できるはずです。

問題1　前の数字との差を求める

図2のようなデータがあったとします。やりたいことは、各行の数字に対して、その1つ上の数字との差を求めるということです。出力は、自分で見るためのものですので適当でもかまいません。

この問題は、AWKに慣れている人ならすぐにできると思います。**図3**のように、変数を使って行またぎの処理をします。AWKのアクションの中は、今のレコードとaの差を求めて出力し、その後で今のレコードをaに代入するというものです。次のレコードに処理が移ったとき、aには前のレコードの値が入っています。

ところで、この計算では一番最初のレコードの扱いが雑です。最初のレコードの**$1-a**は、aは0で初期化されるので、2-0で2がそのまま出力されます。雑なら雑でよいのですが、もし最初のレコードがいらないなら**図4**のようにtailで取り除きます。

AWKだけでやるとすると、もう少しエレガントな方法もあります。**図5**に示します。ただまあ、これは考え過ぎです。

▼図3　問題1の解答①（AWKによる行またぎの処理）

```
$ cat Q1.INPUT | awk '{print $1-a;a=$1}'
2
51
112
-159
893
-890
```

▼図4　問題1の解答②（1レコード目を出力しない）

```
$ cat Q1.INPUT | awk '{print $1-a;a=$1}' | tail -n +2
51
112
-159
893
-890
```

▼図5　問題1の解答③（図4をAWKだけで実現）

```
$ cat Q1.INPUT | awk 'NR!=1{print $1-a}{a=$1}'
51
112
-159
893
-890
```

問題2　キーごとに数字を足す

次は、つい表計算ソフトウェアでやってしまいそうな足し算の問題です。**図6**のファイルQ2.INPUTについて、キー（この例では**001 AAA**や

▼図2　問題1の入力ファイル

```
$ cat Q1.INPUT
2
53
165
6
899
9
```

▼図6　問題2の入力ファイル

```
$ cat Q2.INPUT
001 AAA 0.1
001 AAA 0.2
002 BBB 0.2
002 BBB 0.3
002 BBB 0.4
```

▼図1　実行環境

```
$ uname -a
Darwin uedamac.local 12.2.1 Darwin Kernel Version 12.2.1: Thu Oct 18 16:32:48 PDT 2012; ➪
root:xnu-2050.20.9~2/RELEASE_X86_64 x86_64
uedamac:201305 ueda$ awk --version
awk version 20070501
$ gsed --version
GNU sed version 4.2.1
Copyright (C) 2009 Free Software Foundation, Inc.
（以下略）
```

002 BBBのこと）ごとに数字を足してみましょう。

この問題を一番素直に解くと、**図7**のように連想配列を使ったものになるでしょう。1列目と2列目の文字列を空白をはさんで連結してキーにして、配列sumの当該のキーに値を足し込みます。sumの各要素はゼロで初期化されているので、最初から**+=**で足してかまいません。そして、前号の画像処理の際も説明しましたが、AWKの配列は連想配列で、インデックス（角括弧の中）に初期化せずになんでも指定できます。ですので、このように文字列を丸ごとインデックスにできます。

for(k in sum)は、配列sumの全要素のインデックスを1つずつkにセットしてfor文を回す書き方です。

ところでこの方法だと、ソートしていないデータでも足し算してくれる一方、入力されたデータをAWKが一度全部吸い込んでから出力するので、パイプの間に挟むとデータが一時的に流れなくなります。最後に順番に出力してくれるとも限りません。さらに、連想配列を使っているので、もう少しデータが大きくなると遅くなります。

ということで、入力レコードがもうちょっと多くなったときの書き方を**図8**に示します。データはソートされていることが前提となります。念のために言っておくと、図7の方法で済むうちは、わざと難しく書く必要はないので、図7の方法でやってください。

1列目と2列目が一対一対応でないことがある場合は、**図9**のようにしましょう。

この方法だと、キーの境目で出力がありますし、配列にデータを溜め込むということもありません。

この手のAWKプログラミングでは、まずパターンがどれだけあるか考え、その後に各パターンで何をしなければならないのかを考えると、すんなり問題が解けることがあります。この例では、

- キーが変化するレコードで行う処理（キーと和を出力）
- 通常の処理（キーを記憶し、数字を足す）
- 最後の処理（一番最後のキーの和を出力）

と3つのパターンとアクションを考えることで、if文を使わずに目的の計算を実装しています。

ちょっと脱線しますが、同じ処理をPythonで素直に書くと、**図10**のようになります。一概に長い短いを比較することは乱暴ですが、変数の初期化の方法、行の読み込み方、パターン v.s. if文、という3つの点において、AWKの方が、筆者の出している問題に対して近道であることがわかります。

もう1つ。Open uspTukubaiのsm2を使えば、**図11**のようにコマンド一発で終わりです。詳細については **https://uec.usp-lab.com**をご覧ください。

このAWK、Python、sm2（あとはExcel）の例を見比べていると、道具の選び方について考えさせられます。普段、端末を使わない人がわざわざAWK一行野郎（or perl一行野郎）やsm2を覚

▼図7　問題2の解答①（連想配列を使う）

```
$ cat Q2.INPUT | awk '{sum[$1 " " $2] +=$3}END{for(k in sum) ➡
{print k,sum[k]}}'
001 AAA 0.3
002 BBB 0.9
```

▼図8　問題2の解答②（配列を使わずに実現）

```
$ cat Q2.INPUT | awk 'NR!=1 && k1!=$1{print k1,k2,sum;sum=0}¥
                {k1=$1;k2=$2;sum+=$3}END{print k1,k2,sum}'
001 AAA 0.3
002 BBB 0.9
```

▼図9　問題2の解答③（キーの判定を改善）

```
$ cat Q2.INPUT | awk 'NR!=1 && k!=$1" "$2{print k,sum;sum=0}¥
                {k=$1" "$2;sum+=$3}END{print k,sum}'
001 AAA 0.3
002 BBB 0.9
```

える必要があるかと言われると正直疑問です。おまけに、問題に特化したツールほど数が多くなるので覚えるのがたいへんです。ただし一行野郎もコマンドも知らないと、もしテキストの数字を足せと言われたとき、遠回りを余儀なくされます。

少なくとも言えることは、この「汎用的なものを少し覚える v.s. 特化したものを多く覚える」には優劣がないということと、とくに若い人は、思っているより人生は長いので、どっちもたくさん勉強しておくと後から時間が稼げるということです。

もう1つ言っておくと、「コマンド派」には、「組み合わせ」という強みがあります。Open usp Tukubaiのコマンドは今のところ50足らずですが、組み合わせることで無数の仕事をこなせます。組み合わせることで数が爆発することは、ご存じかと思います。この点において、言語のライブラリをいちいち調べることよりも、コマンドを覚えるほうが学習の密度は高く、優先度も高いのかなと考えています。

問題3　特定期間中に対象日付が何日あるか数える

図12のファイルを考えます。Q3.SPANの各レコードは日付の範囲で、Q3.DAYSは日付のつもりです。やりたいことは、Q3.SPANの各日付の範囲に、Q3.DAYSが何日ずつ含まれているかを調べるということです。

筆者が「AWK1個だけ」という制限のもと、最初に書いたワンライナーは図13のようなものです。

これは完全にライトオンリーの(=全く読めない)コードですので読む必要はありません。どんな処理か説明すると、次のようになります。

- 最初のパターンでQ3.SPANの各レコードを配列f(from)、t(to)に代入
- 次のパターンでQ3.DAYSの日付とf、tの内容を比較して、日付が期間中にあれば、期間に対応するカウンタ(sum[<期間>])に1を足す
- ENDパターンで、各期間とsumの内容を出力

▼図10　問題2 pythonでの解答

```
$ cat ./sum.py
#!/usr/bin/python

import sys

key = ""
n = 1
s = 0.0
for line in sys.stdin:
        token = line.rstrip().split(" ")
        k = " ".join(token[0:2])

        if n != 1 and key != k:
                print key, s
                s = 0.0

        s += float(token[2])
        key = k
        n += 1

print key, s

$ cat Q2.INPUT | ./sum.py
001 AAA 0.3
002 BBB 0.9
```

▼図11　問題2 Open usp Tukubaiを使った解答

```
$ sm2 1 2 3 3 Q2.INPUT
001 AAA 0.3
002 BBB 0.9
```

▼図12　問題3の入力ファイル

```
$ cat Q3.SPAN
20130101 20130125
20130126 20130212
20130213 20130310
20130311 20130402
$ cat Q3.DAYS
20130102
20130203
20130209
20130312
20130313
20130429
```

▼図13　問題3の解答①(AWKだけで実現)

```
$ awk 'FILENAME~/SPAN/{f[FNR]=$1;t[FNR]=$2}¥
        FILENAME~/DAYS/{for(k in f){¥
        if($1>=f[k] && $1<=t[k]){sum[f[k]" "t[k]]++}}}¥
        END{for(k in sum){print k,sum[k]}}' Q3.SPAN
Q3.DAYS
20130126 20130212 2
20130311 20130402 2
20130101 20130125 1
```

変数FILENAMEにはオプションで指定したファイル名、FNRには各ファイル内でのレコード番号があらかじめ代入されており、このコードではそれをパターンや配列のインデックスに使っています。

筆者は答えが出ればそれでOKという立場ですが、もうちょっときれいに解いてみましょう。シェルスクリプトでデータ処理を行うときは、1レコードに計算する対象がすべて収まっていると楽な場合が多くなります。ということは、あらかじめそのような状態を作りにいけばよいということになります。

ということで、まず、**図14**のようにQ3.DAYSとQ3.SPANのレコードの組み合わせを全とおり作ります。

こうなると、3列目の日付が1、2列目の日付の範囲に含まれているものを出力し、数を数えるとよいということになります(**図15**)。このほうが、1個のAWKで頑張るよりすっきりしますね。

これもOpen usp Tukubaiを使うともっと楽になります。**図16**に解答を示します。loopxは、図15の最初のAWKと同じ処理をしています。

問題4　文章を特定の文字数で折り返す

最後に、文章の処理を扱ってみましょう。さばくのは**図17**のファイルです。文章の内容は信じないようにしましょう。大嘘です。

このファイルを、何かのフォームに貼付けるために、20文字で折り返すというのが問題です。さっそくやってみましょう。

図18では、最初のgsedで1文字1文字、後ろに改行を入れて出力し、次のAWKで20文字ずつまとめています。単に文字列を改行しないで出力したい場合は、この例のようにprintfを括

▼図14　期間と日付の組み合わせを作る

```
$ awk 'FILENAME~/SPAN/{key[FNR]=$1" "$2}FILENAME~/DAYS/{for(k in key){print key[k],$1}}' ➲
Q3.SPAN Q3.DAYS
20130126 20130212 20130102
20130213 20130310 20130102
20130311 20130402 20130102
20130101 20130125 20130102
20130126 20130212 20130203
20130213 20130310 20130203
（以下略）
```

▼図15　問題3の解答㈱（図14の情報をもとに集計）

```
$ awk 'FILENAME~/SPAN/{key[FNR]=$1" "$2}FILENAME~/DAYS/{for(k in key){print key[k],$1}}' ➲
Q3.SPAN Q3.DAYS | awk '$1<=$3&&$3<=$2{print $1,$2}' | uniq -c
   1 20130101 20130125
   2 20130126 20130212
   2 20130311 20130402
```

▼図16　問題3 Open usp Tukubaiを使った解答

```
$ loopx Q3.SPAN Q3.DAYS | awk '$1<=$3&&$3<=$2{print $1,$2}' | uniq -c
   1 20130101 20130125
   2 20130126 20130212
   2 20130311 20130402
```

▼図17　問題4の入力ファイル

```
$ cat Q4.MEMO
「コロラド大ダンゴ虫」は、直径20cmになる世界最大のダンゴ虫。ダンゴ虫を転がし、トゲを抜いた柱状サボテン ➲
を倒して遊んだのが、ボーリングの始まり。
```

弧なしで変数を指定すれば大丈夫です。

この出力、少々問題があります。読点(、)が1個、行頭に来ています。読点を21文字目にくっつけてよいなら、**図19**のようにコードを書けばよいでしょう。

わかりにくいと思いますが、このAWKは「先読み」というプログラミングの定石を使っています。実は問題1、2でも使っています。**図20**のコードのように、1行読み込んで1行前の行を出力するコードを書いて、そこから必要なコードを足します。そうすると、aの出力をその次の行を見て操作できます。

読点も含めて20字で収めなければならないなら、話はもっとややこしくなります。コードだけ示しておきます(**図21**)が、ここまで実装するとバグを取るのがたいへんでした。こうなったら、字の折り返しのコマンドをちゃんと作ろうかという気になってきます。もちろんコマンドにするなら、読点だけでなく句点などにもちゃんと対応して汎用性を目指すことになります。

最後にどうでもよいことを述べると、この出力の1字1字にカンマ(,)を挟み込んでcsvにすると、あの悪名高き「Excel方眼紙」に貼り付けできます。やむなくExcel方眼紙に字を書く羽目になったら、お試しください。スジのいい人なら、1字ずつ方眼紙で書く時間より、AWKを覚える時間のほうが短いです。そりゃ言い過ぎですね。失礼しました!

最後に

今回はシェルスクリプトでテキスト処理する際につまずきやすい問題を、AWKで解決してみました。このような類いの処理は無数にありますが、先読みが使えるかどうかなど、パターンはそれほどないので、慣れてしまうとシェル上でさばくことに抵抗が薄れてきます。とくに先読みは多くの集計処理に登場します。

次回は、Twitterでリクエストを受けたので、netcatなどを使ってbashで通信を行ってみたいと思います。SD

▼図18　問題4の解答①(gsedとAWKで実現)

```
$ cat Q4.MEMO | gsed 's/./&¥n/g' |
        awk '{printf $1}NR%20==0{print ""}END{print ""}' > tmp
$ cat tmp
「コロラド大ダンゴ虫」は、直径20cmに
なる世界最大のダンゴ虫。ダンゴ虫を転がし
、トゲを抜いた柱状サボテンを倒して遊んだ
のが、ボーリングの始まり。
```

▼図19　問題4の解答②(行頭の読点対応)

```
$ cat tmp | awk 'NR!=1{¥
        if($1~/^、/){print a"、"}else{print a}}¥
        {a=$1}END{print a}' | sed 's/^、//'
「コロラド大ダンゴ虫」は、直径20cmに
なる世界最大のダンゴ虫。ダンゴ虫を転がし、
トゲを抜いた柱状サボテンを倒して遊んだ
のが、ボーリングの始まり。
```

▼図20　先読みの骨組み

```
$ cat tmp | awk 'NR!=1{print a}{a=$1}END{print a}'
「コロラド大ダンゴ虫」は、直径20cmに
(以下略)
```

▼図21　問題4の解答③(読点も含めて20字以内にする)

```
$ cat Q4.MEMO | gsed 's/./&¥n/g' |
awk 'BEGIN{c=0}¥
NR!=1{if(c==19 && $1=="、"){print "";printf a;c=1}else{printf a;c++}}¥
c==20{print "";c=0}{a=$1}END{print a}'
「コロラド大ダンゴ虫」は、直径20cmに
なる世界最大のダンゴ虫。ダンゴ虫を転が
し、トゲを抜いた柱状サボテンを倒して遊ん
だのが、ボーリングの始まり。
```

第37回 Androidの取り巻く環境とその進化

モバイルデバイス初のオープンソースプラットフォームとして、エンジニアから高い関心を集めるGoogle Android。いち早くそのノウハウを蓄積したAndroidエンジニアたちが展開するテクニックや情報を参考にして、大きく開かれたAndroidの世界へ踏みだそう！

日本Androidの会 幹事
嶋 是一　SHIMA Yoshikazu

はじめに

2007年11月にGoogle Androidが発表されてから今年で6年目です。連載を開始した3年前にはまだ少数派だったAndroidも、いまや搭載台数でiOSを抜き、スマートフォンで最もメジャーなOSとなりました。自身の進化とともに快進撃を続けるAndroidですが、ここに来てHTML5というキーワードのもと、Firefox OS、Ubuntu、そしてTizenといった新しいスマートフォンOS勢が名乗りをあげて来ました。Androidがこれら勢力に脅かされるのか、しばらくは動向が混沌としそうです。

ただ混沌はIT業界ではチャンスであり、プログラマとしてはアプリを開発を行える、新天地の開拓に他なりません。新しいモバイルのフロンティアを目指す皆さんへ、現在Androidを取り巻くモバイルOSの状況を紹介します。

デファクトスタンダードOS；Android

Androidはもっとも普及したOSとなりました。世界シェアで見ると2012年末の段階で、実にスマートフォン出荷の70%がAndroid端末であり、続いてiOSが19.4%となっています[注1]。台数で見てみると、全世界でアクティベート（電源を入れてアカウント登録した）端末数が7.5億台を越え（2013年3月[注2]）、Androidを搭載した端末を提供するメーカーは65社、アプリはマーケットから、累計250億ダウンロードされるほどに広がっています。途方もない数字をAndroidはこの5年間で達成しました。

そもそもAndroidは「オープンな携帯電話OS」という、誰でもモバイル端末の開発ができるという従来になかったコンセプトを広めました。さまざまなメーカーや、さまざまな機種の上で動作するOSとして複数の端末が発売されました。iPhoneのiOSはAppleの端末とセットですし、フィーチャーフォン（ガラケーとも言われます）はOS自体の存在を気にすることがありません。スマートフォンにOSがあることを、利用者が意識しはじめたOSがAndroidだったのではないでしょうか。同時に、スマートフォンを購入する際「メーカー」+「OS」で端末を選択する文化の定着もさせました（旧来Windows CEなどもありましたが、台数が少なかった）。このオープンOSのしくみも一因して、Androidがもっとも普及したスマートフォンOSとなりました。

このようなAndroid OS搭載スマートフォンの増加とオープンな開発環境は、開発環境としても、大きなメリットをもたらしています。

注1）Strategy Analytics: Android and Apple iOS Capture a Record 92 Percent Share of Global Smartphone Shipments in Q4 2012 URL http://blogs.strategyanalytics.com/WSS/post/2013/01/28/Android-and-Apple-iOS-Capture-a-Record-92-Percent-Share-of-Global-Smartphone-Shipments-in-Q4-2012.aspx

注2）URL http://www.androidpolice.com/2013/03/13/google-ceo-larry-page-750-million-android-devices-activated-to-date-more-than-250-million-in-the-last-6-months/

私たちソフトウェア開発者は自らのアイデアで作成したプログラムを、自分の分身でもある携帯電話に気軽に入れて動かす環境が実現できました。また、作成した便利プログラムは、自分のスマートフォンに入れて肌身離さず持ち歩けるようになりました。そしてなによりも、自分で作ったアプリが他人のAndroid OS上で動かせるよう配布することもできるのです。実に世の中には7.5億台というAndroid端末がアクティベートした実績があるのです。どれだけ多くの「他人」に、自分のアイデアであるアプリを提供できるか考えると、ワクワクしませんか？

面白いと感じたならば、ぜひ、この後に紹介するAndroid SDKをダウンロードしてアプリ開発にチャレンジしてみましょう。

Androidの進化

開発環境「バンドル」に変化

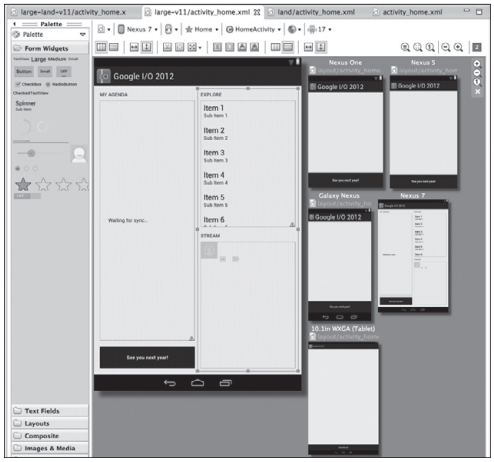
ここ1年の間にAndroidの開発環境が変化しました。

Android本体のバージョンは2012年6月にJB(Jelly Bean)が発表されました。画面のスクロールや動作のチューニングなどがされ「目指せiPhone画面の滑らかさ！」といった意気込みを感じさせるバージョンでした。ただ開発者からすると、もっと大きい変化がありました。開発環境であるAndroid SDKの配布形態が変わったのです。

Androidのアプリケーション(以下、アプリ)を開発する際には、Android Developersサイトから Android SDKを入手します。2012年11月に提供された「Android SDK Tools, Revision 21」より、これまで別々にインストールしていた「eclipse」や「ADT(Android Developer Tools)」が「バンドル(bundle)」という形で1つにまとめられ、一発インストールが可能になっています。

これまではアプリ開発環境をインストールするのにネットワーク環境が必須でした。ネットワークがつながらないとか、proxyがhttpsを通さないなどの条件があると正しくインストールできませんでした。または、Androidのプログラム講習会などで一斉にAndroid SDKをインストールすると、通信が混雑してインストールが失敗することもありました。今回のバンドルにより、これらの問題が解決しています。

▼図1　マルチビューで複数のAndroid画面を編集可能



また、UI画面を作成するためのオーサリングツールが複数の画面を編集できるようになりました(**図1**)。異なるAndroid端末の画面サイズや、異なる言語の画面を同時に表示し、編集することができるようになったのです。

もし、Androidの開発に興味ある方でSDKをインストールしていない方は、Android Developersサイト[注3]に訪れて「Download the SDK」から開発環境を取得しましょう。PC上だけで開発環境を作ることができます。また、ここ半年ほど開発をサボっていた方も新しいSDKに入れ替えましょう。

Chrome for AndroidとAndroidの統合

2013年3月13日に、GoogleのAndroid開発責任者のトップが交代しました。これまでGoogleになる以前からAndroidを率いてきたAndy Rubin氏が退き、現在GoogleでChromeの責任者であるSundar Pichai氏が兼任することとなったのです。

Androidの標準ブラウザのHTML5対応スコアや、不具合対応や機能追加などは、他の競合

注3) URL http://developer.android.com/intl/ja/sdk/index.html

◀写真1　Googleの最新Chrome OS搭載Chrome BOOK「Chromebook Pixel」

ブラウザと比較すると遅れを取っている現状があります。一方、Chrome for AndroidはPC版で定評のあるChromeブラウザのAndroid版として開発されています。

Android端末はデフォルトこそ標準ブラウザですが、Chrome for Androidブラウザも ICS（Ice Cream Sandwich）から標準搭載されるようになりましたし、Nexus 7についてはChrome for Androidのみ搭載されています。このルールは、Googleと端末メーカーの間で定められたGMS（Google Mobile Service）を用いる契約により決められていると想定されます[注4]。

AndroidがHTML5の雄となるためには、これまで以上にChrome for Androidの標準ブラウザ化を進めることになります。そして、Chrome開発責任者がAndroidを兼務することで、この動きが促進されると予想されています。

ただし3月21日に「ChromeとAndroidを統合することはない」とGoogleのエリック・シュミット会長が語ったとのニュースもありました。しかしこれは、Chrome OSとAndroid OSの話題であり、ブラウザの議論とは別です。Chrome OSを搭載した最新ノートは「Chromebook Pixel」として2月21日に発表されています（**写真1**）。

新しいOS時代

2011年から2012年にかけて、新しいモバイルOSたちの声が届きはじめました。とくに今年のスペイン・バルセロナで開かれたモバイル関連の展示会「MWC（Mobile World Congress）2013」で　は、Firefox OS、Tizen、Ubuntu、Sailfishなどのまだ商用化されていないモバイルOSが発表され、話題となりました。

Firefox OS

Firefox OSはMozilla社が開発を行っているFirefoxブラウザのスマートフォン向けOSです[注5]。一部ではFxOS、FFOSなどと略されます。FirefoxのHTMLレンダリングエンジンはGeckoというNetscape時代からのエンジンを用いています。もともとはこのGeckoをスマートフォン上でブラウザとしてではなく、直接ブート（起動）させる試みとしてB2G（Boot to Gecko）と呼ばれていました。これが昨年改名されてFirefox OSとなっています。

一般的に、ブラウザ上でネイティブアプリ同等のアプリを、JavaScriptなどを活用して制作したものをWebアプリと呼びます。

Firefox OSのアプリはすべてHTML5で記述し、Webアプリとして作成されます。そのため、特別にネイティブの言語を覚えたりする必要がありません。専用のSDKもなく、Webのオーサリングツールをそのまま用いて開発することができます。そういう意味で、Webクリエイタも開発しやすい環境となっています。

Firefox OSの大きな特徴として、ハードウェアレイヤからLinux Kernelなどのハードウェア環境周りにはAndroidの資産を用いている点です（**図2**）。つまり、Androidが動作する端末であるならば、Firefox OSが搭載しやすい環境となっています。実際に、現在はNexus S、Galaxy S2そしてNexus 7などでも動作させることができています。これはAndroidが動作するためのLinux／ドライバの層をHALのように位置づけたと言えます。実はこの後出てくるUbuntuも同じような位置づけとなっています。Androidの開発を経験したことがある人ならば、「Firefox OS搭載端末へのPCからのアプリインストールなどはadbを用いる」と言えば理解しやすいかもしれません。なお、国内動向としてはKDDIがMWCで参画を

注4）URL http://www.soumu.go.jp/main_content/000143085.pdf ページ8

注5）URL https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS

表明しています。国内のコミュニティーとしては、Googleグループ；Firefox OS[注6]があります。

Tizen OS

「タイゼン」と発音するこのOSは、携帯電話のプラットフォームであったLiMoプラットフォームと、インテルやノキアのMeeGoのプラットフォームが統合して作られたOSです[注7]。Androidが登場した2008年当時は、Linuxベースの携帯電話プラットフォームとしてLiMoも話題になっており、NTTドコモから対応端末も発売されていました。フィーチャーフォンからすると、由緒正しき携帯電話のプラットフォームということになります。現在SAMSUNGのSLPというプラットフォームに基づいており、ライセンス条件もSAMSUNGのものとなっています。

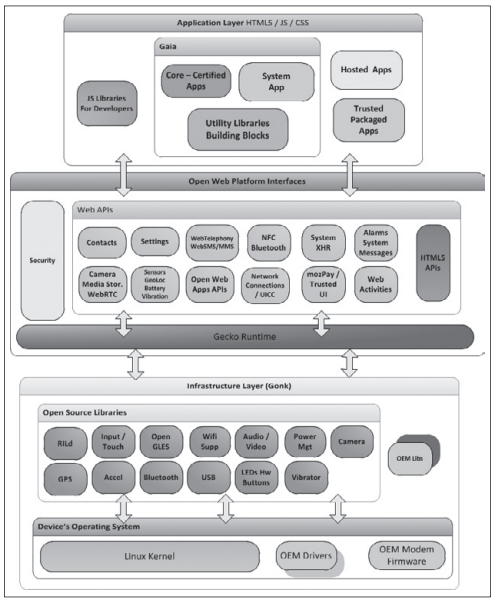
アプリはHTML5のWebアプリだけではなく、C/C++で記述するネイティブアプリの双方が開発できるようになっています。eclipseベースで構築されているTizen SDK[注8]を用いてアプリを開発できます。国内ではNTTドコモが参画を表明しています。

Ubuntu for phones

「Ubuntu」のスマートフォン向けOS「Ubuntu for phones」[注9]が2013年1月に発表され、世界最大の家電展示会「2013 INTERNATIONAL CES」にて動作展示がされました(**写真2**)。Ubuntu for phonesもFirefox OSと同じく、Linuxなどのハードウェア周りはAndroidの資産を用いています。

Ubuntu for phonesの最大のポイントは、スマートフォンとPCのOSが融合されている点です。つまり、高機能端末の場合は、ドックに挿すことでスマートフォンがUbuntu OSが搭載されたPC本体となり、利用することができます。しかもアプリは、スマートフォンとPCを共用で作ることもできます。

▼図2　Linux KernelやDrives部分にはAndroidのリソースを利用



［出 典］Firefox OS architecture URL https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture?redirectlocale=en-US&redirectslug=Mozilla%2FFirefox_OS%2FArchitecture

写真2▶ Ubuntu for phones

Androidと新しいOS群の関係

新たなOSが今すぐAndroidの牙城を崩すとは考えにくいですが、新しいOSがもたらすアイデアやコンセプトは、きっとこのモバイルOSの世界に影響を与えることでしょう。そのようなものに触れることはまさにモバイルのフロンティアを体感することにほかなりません。

もう1つ面白いのが、新しいOSが数々登場しても、下回りの部分としてAndroidの資産が使い続けられそうな動向となっている点です。万が一Androidがなくなったとしても、Androidのオープンソースの一部や、そこで決められた仕様などは、このモバイルの世界に残ることでしょう。SD

注6）URL https://groups.google.com/forum/#!forum/firefoxos
注7）URL https://www.tizen.org/
注8）URL https://developer.tizen.org/downloads/tizen-sdk
注9）URL http://www.ubuntu.com/devices/phone

the JBoss Way

皆本 房幸
MINAMOTO Fusayuki
レッドハット(株)
JBossプリンシパルコンサルタント

今月のテーマ

はじめまして、レッドハットでJBossコンサルタントをしている皆本です。コンサルタントというと難しいイメージをお持ちの方もいらっしゃるかと思いますが、JBossのコンサルタントはJBossエンタープライズミドルウェアを採用したお客様の開発の支援をする仕事です。具体的には、お客様のプロジェクトの背景を理解したうえで、開発現場でJBossの導入や開発のお手伝いをさせていただいています。筆者はJBossがそれほど世の中に知られていなかったころからJBossに注目してきました。恵比寿通信でJBossが話題になるのも今回で3回め。今回は、JBossのアーキテクチャの変遷を振り返り、JBossコミュニティでのthe JBoss Wayについて紹介したいと思います。

JBossとのかかわり

JBossに初めて出会ったのは2001年ごろだったと思います。当時、レッドハットに転職する前の会社でEnterprise JavaBeans(EJB)を使った社内システムの開発をしていた筆者は、EJBが使えるオープンソースのJavaアプリケーションサーバがないか探していました(商用のアプリケーションサーバはライセンス料金や保守料金が高く、どこででも使えるという状況ではありませんでした)。自宅のPC上で簡単に動く、軽いサーバというのがJBossに対する第一印象でした。好印象ではありましたが、JBossに関する情報はあまり手に入りませんでした。

JBossを使い始めると、JBossの情報を求めてjboss.orgという開発サイトに頻繁にアクセスするようになりました。初期のJBossプロジェクトでは"Coding the future"や"Beyond J2EE"というのがキャッチフレーズでした[注1]。筆者を含む当時のJBossユーザを惹きつけたのは「自分たちがオープンソースによって世の中を変える」という強烈なメッセージだったと思います。

JBossのやり方とは

JBossの開発者は、jboss.orgのフォーラムやメーリングリストに積極的に参加し、コミュニティを盛り上げていました。大手商用ベンダの製品では、ユーザが製品開発者と直接会話ができるというのはあまりないことです。筆者はフォーラムでのやりとりを通じて、JBossの開発者は新しいアイデアをとても尊重し、それを実現できる人に任せることを知りました。

jboss.orgは複数のサブプロジェクトから構成されていて、各プロジェクトではリーダーが存在してプロジェクトを牽引していました。筆者はコミュニティに貢献する優秀なエンジニアがJBossの開発者としてプロジェクトに参画し、新しいアイデアを実現していくのを目の当たりにしていました。次に何が起こるのかわからない、ワクワクさせてくれるものがJBossでした。

オープンソース開発ですのでユーザからのコー

注1) 当時はエンタープライズJavaの標準はJ2EEと呼ばれていました。

ドやドキュメントの貢献はJBoss本体に取り込まれていきました。自分でも何かできることはないかと考え、jboss.orgフォーラムで書き込みをしたり、JBossのソースを読んだり、JBossマニュアルの日本語翻訳をブログ上で公開したりしていました。まだ、jboss.orgの規模も小さいJBoss 2のころの話です。

JBossアーキテクチャの歴史

JBossはオープンソース開発によって急速に機能を充実させていきました。JBossはEJBコンテナとして開発が始まりましたが、OSSのWebコンテナやメッセージングミドルウェアなども次々にJBossに統合され、1つのエンタープライズJavaアプリケーションサーバとして発展してきました。このような統合を可能にしたのはJBossのカーネルアーキテクチャでした。JBossアプリケーションサーバのアーキテクチャの変遷を見ると、いくつかの大きなアーキテクチャ上の変更があったので振り返ってみましょう(**表1**)。

JBoss 3の「マイクロカーネル」は、モジュールをJMX(Java Management Extensions)のしくみを使ってモジュールを統合していました。JBoss 5の「マイクロコンテナ」は、マイクロカーネルを発展させ、JMXへの依存性をなくした軽量コンテナでした。しかし、JBossが発展するにつれ、搭載されるモジュールもそれまでとは比べ物にならないくらいに増え、もはやJBoss 3のころのような「軽いJBoss」ではなくなってきました。

JBoss AS 7で導入された「モジュラサービスコンテナ(MSC)」は、クラウド上での利用に対応できるように、従来のアーキテクチャを軽量・省リソースなものに刷新したものです。サーバ上で起動するサービスやアプリケーションはJBossモジュールという単位で管理され、クラスローダのしくみも変更されました。また、並行してモジュールのデプロイが可能になり、マルチコアのCPUを利用できるようになったことで、これによりサーバの起動時間が飛躍的に短縮されました。

こうしてJBossアーキテクチャの歴史を振り返ってわかることは、JBossでは常に新しいアイデアを掲げた開発者によって既存の技術が変

▼表1　JBossアーキテクチャの変遷

時期	バージョン	特徴	備考
1999年	EJBOss	EJBコンテナのみ	Marc Fleury氏によってプロジェクトが開始された。 EJBOssの先頭から"E"と取ったものがJBoss
2000年	JBoss 2	EJBのホットデプロイ	"Coding the Future"(未来をコードで築く)
2002年 ～2003年	JBoss 3	JMXマイクロカーネル EJB 2.0/CMP2.0 J2EE 1.3相当	"Professional Open Source" や "Beyond J2EE"(J2EEを超えた) JBoss Groupによる商用サポート
2004年 ～2007年	JBoss 4	アスペクト指向ミドルウェア JBoss AOP Hibernateの統合 J2EE 1.4	2004年JBoss Inc.設立(JBossサポート会社) 2006年Red Hat Inc.がJBoss Inc.を買収 JBoss EAP 4.x リリース
2008年 ～2009年	JBoss 5	DIコンテナベースのマイクロコンテナ Java EE5 EJB 3.0/JPA	JMXに依存しないコンテナアーキテクチャ JBoss EAP 5.x リリース
2010年	JBossAS 6	Java EE6	JBossAS 6はコミュニティ版のみリリース
2011年	JBossAS 7	モジュラサービスコンテナ(MSC) Java EE6 Webプロファイル	クラウド向けの軽量、省リソースアーキテクチャに刷新 モジュールの並行デプロイが可能
2012年	JBossAS 7.1	Java EE6 Fullプロファイル EJB 3.1/JPA 2.0	JBoss EAP 6.0リリース

えられてきたということです。開発者は自らブログなどを使って新しい技術を発信し続けています。しかし、ユーザやコミュニティによって受け入れられて成功したプロジェクトは継続しますが、逆に新しい技術によって置き換えられるものもあるのです。新しいアイデアをコミュニティに問い、そのフィードバックを得て改善していく。JBossの発展の活力はこうしたところから生み出されているのだと思います。そして、それが昔から変わらないJBossのやり方です。

もっとJBoss Way

JBossアプリケーションサーバには、レッドハットがサポートするJBoss Enterprise Application Platform(EAP)と、コミュニティ版のJBossAS(JBoss Application Server)という2つのリリース形態があります。製品としてのJBoss EAPには、レッドハットによるサポート、トレーニング、コンサルティングというしっかりとした支援体制がありますが、JBossASというコミュニティ版についてはコミュニティでのサポートになります。そんなことから、JBoss EAPだけを使っている企業ユーザにはJBossコミュニティというのは無縁の存在と思っている方も多いかもしれません。

しかし、JBossコンサルタントとして仕事をしてきて思うことは、JBossを十分活用していただくためには開発現場の技術者の方にJBossについてもっと知ってもらうことが重要だということです。そのためにはJBossの豊富な技術情報を持つjboss.orgにアクセスするのが近道です。JBossに詳しくなるための最初の一歩はJBossのサンプルプログラムを動作させてみることです。筆者も最初にJBossを触ったときには、まずサンプルプログラムを動かしてみました。JBossはオープンソースですのでサンプルコードとサーバ本体の間に垣根はありません。サーバの内部動作を知りたければデバッガでどこまでも深く追求することもできます。

jboss.orgではthe JBoss Way(http://www.jboss.org/developer/)と題して、JBossを使ったシンプルでより生産性の高い開発スタイルを提唱しています。the JBoss Wayの一環として、jboss as quickstartsがGitHub上で公開されています。これはJBoss上で利用可能なJava EEなどの技術のサンプルプログラム集です。JBossに興味をもたれた方は、JBossASをダウンロードしてquickstartsを動かしみてください。もし、少しでも気にいってもらえたならブログで感想を書いてください。それがJBoss流ですから。

JBoss EAPの公開

2013年3月17日にJBoss CTOのMark Littleから「EAPのバイナリをすべて開発者に公開する」という発表があり、実際にJBoss EAP 6.1.0 Alpha版がダウンロードできるようになりました。従来JBoss EAPは、レッドハットのFedora/Red Hat Enterprise Linuxのモデルと同じで、製品バイナリの利用にはサブスクリプションの購入が必要でした。この動きによって、今後、より多くのJBossユーザが開発時にEAPを使えるようになるとともに[注2]、製品としても多くのフィードバックが得られるようになるでしょう[注3]。ますますjboss.orgの動きに目が離せなくなるのは間違いありません。

最後に本の紹介をさせてください。現在、NTT OSSセンタ有志の皆さんと『JBoss Enterprise Application Platform 6完全ガイド(仮)』なるJBoss本を執筆中です。もちろんJBoss EAPの最新アーキテクチャの解説も満載です。請うご期待！SD

注2) Beta/GAバイナリの本番環境での利用およびレッドハットによる製品サポートには従来どおりサブスクリプションの購入が必要です。

注3) 詳細は、http://www.jboss.org/jbossas/faq

3

Debian Developer　やまねひでき　henrich@debian.org

PostgreSQL公式リポジトリの使い方とパッケージングツール「dh」の紹介

Debian Hot Topics

はじめに

先日、本連載について読者の方から要望のメールをいただきました。こうして声が伝わってくるのはたいへんありがたいことですね。さっそく今号に反映……は難しいですが、いただいた要望(開発コミュニティへの参加方法や定例イベント情報など)についてはこちらも取り上げたい事柄ですので、ページとスケジュールが許す限り対応していく予定です。

そして、そろそろDebian7.0のリリース……のはずなのですが、今回の連載への掲載は、スケジュールから考えて原稿締め切りが許してくれそうにありません。次号で取り上げられることを期待しましょう。

pgapt.debian.net廃止、apt.postgresql.orgへ

若干旧聞に属しますが、紹介する価値があると思いますのでPostgreSQL関係の話を。

少し前までpgapt.debian.netというサイトでDebian開発者がPostgreSQLの各バージョンのパッケージを公開していましたが、これがPost greSQL公式リポジトリのapt.postgresql.orgへと完全に移行しました。こちらを追加することでPostgreSQL 8.3、8.4、9.0、9.1、9.2が、Debian/Ubuntuの各バージョン(squeeze、wheezy、sid、lucid、precise)で利用可能になります[注1]。

Debian Squeezeではは PostgreSQL 8.4が、Wheezyでは9.1が提供されていますが、これ以外のバージョンを使いたい場合には、このリポジトリが助けとなるでしょう。利用方法は**図1**のようになります。ここまでできたら**リスト1**を/etc/apt/preferences.d/pgdg.prefに記入してください。これで**apt-get install postgresql-9.2**などとすれば、Squeeze/Wheezyなどの環境でもPostgreSQL 9.2が利用できるようになります。これまでpgapt.debian.netのリポジトリを利用されていた方は変更をお忘れなく。

注1) URL https://wiki.postgresql.org/wiki/Apt

▼図1　apt.postgresql.orgの利用方法

```
GPG公開鍵を取得してaptが信用する鍵一覧に追加
$ wget -O - http://apt.postgresql.org/ ➥
pub/repos/apt/ACCC4CF8.asc | sudo apt- ➥
keyadd -

apt lineに追加してパッケージデータベースを更新
squeezeとなっているところは適宜wheezyなどに読み替え
$ sudo sh -c 'echo "deb http://apt. ➥
postgresql.org/pub/repos/apt/ squeeze- ➥
pgdgmain"' >> /etc/apt/sources.list.d/ ➥
postgresql.list
$ sudo apt-get update
```

▼リスト1　postgresql.orgからパッケージを優先してとってくるように指定

```
Package: *
Pin: release o=apt.postgresql.org
Pin-Priority: 500
```

パッケージメンテナンスのトレンド―「dh」

Lucas Nussbaumさんのブログで「Debian is (still) changing」[注2]というタイトルでDebianでのパッケージのメンテナンス体制、使われているバージョン管理システム、パッケージツールなどの推移について考察が行われています。これをかいつまんでまとめると、「Debianのパッケージは、チームでメンテナンスを行い、バージョン管理はGit、パッケージングツールはdhを使うのが多数を占める」ということになります。

さて、みなさんGitはすでにご存じでしょうが、dhについては初耳だと思いますので解説をしましょう。Debianパッケージを作成するためのツールとして「debhelper」というものがあるのですが、このdebhelperのバージョン7から導入された、よりモダンなパッケージングが可能なツールが「dh」です。モダンなパッケージングとは何か、何が良いのかというと、一言で言えば**「設定より規約(convention over configuration)」**[注3]ということに尽きます。

Debianパッケージの肝はソースコードファイル群のdebian/rulesというMakefileです。ここでdebhelperというパッケージ用ツールコマンド群を呼び出して、パッケージ化に必要なさまざまな処理を行っていきます。初期のdebhelperは**リスト2**のように「やることを列挙して指定」していました(これを「debhelperスタイル」と呼びます)。

誌面の都合上省略しましたが、リスト2は全部で52行あります。これは、順に目で追っていけば、「何をしているのか」が理解できるという点では良いのですが、「何をしたいのか」が埋没することになります。

debhelper 7以降では「dh」という形で大幅にこのrulesファイルを省略できます。これはrulesファイルで指定していたdh_というprefixの各種プログラムが、そもそも決まりきった順序で呼び出されることに着目して、rulesファイルについて大きく抽象化した記述を行います(後述します)。そして、rulesファイル中で明示的に記述を行う代わりに、ソースコードファイル群のdebianディレクトリ以下の各ファイルに処理を行いたい対象を記述しておく、という手法を行っています。

たとえば、debian/cleanファイルを作成して

▼リスト2　poppler-data パッケージの debian/rules ファイル(一部省略)

```
#!/usr/bin/make -f
# Uncomment this to turn on verbose mode.
#export DH_VERBOSE=1

configure: configure-stamp
configure-stamp:
        dh_testdir
        # Add here commands to configure
the package.
        touch configure-stamp
(中略)
install: build
        dh_testdir
        dh_testroot
        dh_clean -k
        dh_installdirs
        # Add here commands to install ➥
the package into debian/poppler-data.
        $(MAKE) DESTDIR=$(CURDIR)/debian/➥
poppler-data prefix=/usr install
                ↑インストール先を変更してmake install
# Build architecture-independent files ➥
here.
binary-indep: build install
        dh_testdir
        dh_testroot
        dh_installchangelogs
        dh_installdocs
        dh_link
        dh_strip
        dh_compress
        dh_fixperms
        dh_installdeb
        dh_gencontrol
        dh_md5sums
        dh_builddeb
(以下略)
```

注2) URL http://www.lucas-nussbaum.net/blog/?p=751
注3) URL http://ja.wikipedia.org/wiki/設定より規約
JavaのMavenのディレクトリレイアウトや、Railsが同様のアプローチを採っています。難しそうに聞こえますが「いちいちやることを書くのではなく、最初に『お約束』のファイル配置とデフォルト値を決めて、それに従って処理を行う。例外事項が発生したらその都度記述する」こと、とご理解ください。「まず基本の型、ところどころで臨機応変」のほうがわかりやすいでしょうか。

.bakと書けば、それは「dh_cleanコマンドを使って.bakにマッチングするファイルを消去する、と指定する」ということとイコールになります。パッケージングの肝であるdebian/rulesも基本的に規約のままで済ませ、規約(＝デフォルトの動作)からはみ出るイレギュラーな事柄に対しては「override」を行ってその部分だけ動作を変える、という考えです。

すると、debian/rulesは本ページの下部に掲載した**リスト3**のような形になります。

まったく同一ではありませんが、先ほど挙げたdebhelperスタイルのものと比較すると大幅に少なくなっていること(52→13行、1/4に減少)がわかるかと思います(ちなみに最短のものですと、shebang行を含め、3行で済みます)。肝は

```
%:
	dh $@
```

という行で、Makeのターゲットの抽象化を行うことで大幅な省略を実現しています。そして「意図的に変更を加えたい部分」が明示的にあぶり出されていることが、わかるのではないでしょうか？

dhを採用するにあたって「いちいちその規約を覚えるのかよ……たいへんだな」という不満があると思いますが、常に使うファイルは限られており、さほど苦労はしないというのが筆者の実感です。また、dh_○○○というネーミングでdebhelperの各コマンド群は存在しているので、ある程度覚えたら、ほかの動作もだいたい推測できます。manも充実しているので、どのファイルに何を書けば良いのか迷うことは少ないでしょう。

パッケージングという「動き」には共通の「型」があることを理解し、そのうえで型に収まらない部分を明示的に書く、ということでそのパッケージの作りに対する理解を大幅に早めることが可能になります。変更を加える場合も容易で、かつ既存の動作に副作用が少なくなります。そもそも理解せずに処理を記述して我流のパッケージングをやっていたら、「技術的負債」が山盛りで、あとからいじる人にしたら魔窟でメンテできません……。

ちなみに、自分の知る限り[注4]では、RPMパッケージのspecファイルは「設定より規約」にはなっておらず、かつ独自マクロなどをてんこ盛りにできるので、長く複雑になりがちです。(dh以降の)Debianパッケージのほうが一般的に可読性や保守性に優れているのでお勧めだ、と思っているのですがいかがでしょうか？ SD

注4) YouTubeでRed Hat社の人の講演(https://www.youtube.com/watch?v=K_A6U2nWntE)を聞いただけ、とも言います。小一時間で見られるように、ざっとRPMパッケージングのしかたをまとめてあり、わかりやすくてお勧めです。

▼リスト3　dhスタイルで書き直したdebian/rulesファイル(全体)

```
#!/usr/bin/make -f
# Uncomment this to turn on verbose mode.
# export DH_VERBOSE=1

%:
	dh $@

override_dh_auto_install:	←dh_auto_installターゲットを上書き
	$(MAKE) DESTDIR=$(CURDIR)/debian/poppler-data prefix=/usr install
	dh_install	↑インストール先を変更してmake install
	↑debian/installファイルに書いてあるファイルをインストール
override_dh_builddeb:
	dh_builddeb -- -Zxz	←圧縮方法にxzを使ってdebパッケージを作成
```

第37回 Ubuntu Monthly Report

CMISサーバとLibreOffice 4.0の連携

今回はCMISサーバを準備し、LibreOffice 4.0と連携して使用する方法の紹介です。

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

CMISとは

CMISはContent Management Interoperability Services(「コンテンツ管理の相互運用性サービス」とでも訳すのでしょうか)の略で、これ自体はその名のとおりのサービス、つまり規格です。この規格はOrganization for the Advancement of Structured Information Standards(OASIS)によって承認されており、オープンスタンダードとなっています。なお、LibreOffice(以下LibO)が採用しているフォーマット(ODF)も、同じくOASISに承認されています。つまりは、オープンスタンダードなサービスなので複数の実装(サーバ)があるということです。

LibO 4.0(4.0.0～4.0.xをまとめてこう表記)からCMISをサポートするようになったので、Ubuntuでサーバの構築から連携までの方法を紹介します。

CMISサーバは、一般的にはドキュメントサーバや文書管理システムなどと呼ばれており、具体的にはMicrosoftのSharePoint[注1]が有名です。

CMISのいいところ

CMISサーバを使ってみていいなと思ったのは、ドキュメントをドキュメントとして扱うところです。通常ドキュメントは手元のフォルダないしファイルサーバに置きますが、これはあくまでファイルであってドキュメントとして扱うわけではありません。ドキュメントとして扱うと開かなくてもプレビューが表示できるとか、更新履歴が残るとか、権限をきちんとすれば誰が変更したのかがすぐにわかりますし、差分も取れます[注2]。確かに変更履歴はLibOも機能として持っていますが、正しく運用するのはなかなか難しい上にサイズも大きくなってしまいます。ただし、これから紹介するCMISサーバは個人で運用するには大規模向け過ぎるので、もう少し手軽な実装があるといいと思いました。

AlfrescoかNuxeoか

ドキュメントのアクセスにはURIを使用するので、実装によって違いが出てしまいます。LibO 4.0.1が対応しているCMISサーバは**図1**[注3]のとおりで[注4]、さらにUbuntuで動作するもので絞られ、その結果Alfresco[注5]かNuxeo[注6]かというのが現実的な選択肢となります。どちらもUbuntuのリポジトリにはなく、オフィシャルのバイナリを使用することになる

注1) SharePointもCMISサーバになるのですが、当然のことながらUbuntuでは動作しないので本連載の範疇からは外れるものの、現実的には一番よく使われる気もします。

注2) LibOの機能でです。

注3) 今回インストールしたのはNuxeo 5.6ですが、問題なく動作しています。

注4) 今後増える可能性はあります。といいますか、4.0.1へのバージョンアップでも増えました。

注5) http://www.alfresco.com/jp/

注6) http://www.nuxeo.com/en/

のですが、今回はNuxeoを使用することにしました[注7]。AlfrescoはNuxeoと比較して多機能で日本語化も進んでいますが、インストーラからインストールする関係で運用がたいへんそうという印象を受けました[注8]。Nuxeoはaptでインストールできるので、運用が比較的楽です。また、バックエンドにLibOを使用するのもおもしろいです。英語ではあるものの（そして読みにくいものの）詳細なマニュアルがあるのもありがたいです。

というわけで、今回はUbuntu 12.04にNuxeoをインストールします。Ubuntu Serverではなくデスクトップを使用してください。可能な限りまっさらの環境が望ましいです。

Nuxeoのインストール

Nuxeoのインストールは簡単です。**図2**のとおりにコマンドを実行してください。

実行すると大量のパッケージがダウンロードされるので、しばらくお待ちください。インストールが開始されるると、「Bindアドレス」「Listenするポート番号」「バックエンドのデータベース」の質問がありますが、いずれもデフォルト値で問題ありません。あえて言えば、デフォルトのポートが8080ですので、これを使用しているほかのもの[注9]が動いている場合は別ポートにするといいでしょう。本誌の読者には釈迦に説法ですが、ポートが80以外のときはURIでポート番号を指定する必要があります。これが割に落とし穴になったりします。

Nuxeoのウィザード

Firefoxを起動し、“localhost:8080”にアクセスするとNuxeoのウィザードが開始します。ほかのUbuntuからアクセスする場合は“（ホスト名）.local:8080”とすると、いちいちIPアドレスの確認をしなくてもいいので便利です。ウィザードで聞かれる内容は次の5つです。

- HTTPのProxy
- SMTP転送
- Nuxeo ConnectとNuxeo Studioの設定
- モジュール
- ダウンロード

Nuxeo ConnectとNuxeo Studioは、ひとまず［Or skip and don't register］で飛ばします。それ以外も、次に進んでしまっていいでしょう。どうしてもここでしかできない設定はありません。本格運用を検討するようになったら設定を変更すればいいと思います。

すべて完了し、しばらく待っているとログイン画面になります。ウィザードにも書かれていましたが、デフォルトのユーザ名とパスワードはどちらも“Administrator”です。

ログイン後の設定

ログイン後、［Default Domain］に［Sections］［Templates］［Workspaces］の3つのフォルダーが表示されます。このいずれかのフォルダに、さらにフォルダを作成しないとLibOとのドキュメント共有が行えません。今回は［Workspaces］に［test］という

注7）その程度の理由なので、Alfrescoを使ってみるのもおもしろいと思います。
注8）今回紹介しないだけで、動作の確認は行なっています。
注9）tomcatやglassfishなどです。ちなみにNuxeoもtomcatで動作しています。

図1　LibO 4.0.1が対応しているCMISサーバ

Alfresco 4
IBM FileNet P8
Lotus Live Files
Lutus Quickr Domino
Nuxeo 5.4
OpenDataSpace
OpenText ELS 10.2.0
SharePoint 2010

図2　Nuxeoのインストール

```
$ wget -q -O- http://apt.nuxeo.org/nuxeo.key | sudo apt-key add -
$ sudo add-apt-repository "deb http://apt.nuxeo.org/ precise releases"
$ sudo apt-get update
$ sudo apt-get install nuxeo --no-install-recommends
```

フォルダを作成します。[Workspaces]をクリックし、[ワークスペース新規作成]をクリックします。[題名]に[test]と入力して[作成]をクリックすると、testフォルダが作成されます。Nuxeo側の設定はこれで完了です。

LibreOffice 4.0の準備

Ubuntu 13.04のLibOはバージョン4.0なので、そのままで大丈夫ですが、Ubuntu 12.04と12.10の場合[注10]はアップデートする必要があります。**図3**の方法でPPAにあるLibO 4.0にアップデートしてください。

連携方法

まず、ファイルダイアログをLibOのものに変更する必要があります。LibOを起動し、[ツール]-[オプション]-[LibreOffice]-[全般]-[「開く」ダイアログと「保存」ダイアログ]の[LibreOfficeダイアログを使用する]にチェックを入れてください。

連携する適当なドキュメントを作成します。最も簡単に適当なドキュメントを作成する方法は、Writerを起動して"dt"と2文字入力し、F3キーを押すことです。このdtとはダミーテキストのことで、LibOに内蔵されているテキストがサクッと入力されます。保存アイコン[注11]をクリックすると保存ダイアログが表示されます。パスの横に3つのアイコンがありますが、一番左の[...]をクリックします。

[サーバーに接続]ダイアログが表示されるので、[名前]を任意に(今回は[Nuxeo]としました)、[種類]を[CMIS]に、[サーバーの種類]を[Nuxeo 5.4]にします。すると[バインドするURL]が自動的に挿入されるので、[<host>]を書き換えます。今回は12.04のLibO 4.0を使用するので、[localhost:8080]とします。くれぐれもポート番号を忘れないようにしてください。

[リポジトリ]の右側の読み込みアイコンをクリックすると[認証が要求されました]というダイアログが表示されるので、ユーザ名とパスワードを入力します。ログイン時と同じくともに[Administrator]と入力し、[OK]をクリックします。今度はマスタパスワードを入力するダイアログが表示されるので、任意のパスワードを入力してください。成功すると[リポジトリ]に[Nuxeo Repository Default]が表示されます。もし表示されない場合は、一度キャンセルしてもう一度ユーザ名とパスワードを入力するところからやりなおしてください。[パス]と[ログイン]は空白のままにします。そして[OK]をクリックします(**図4**)。

ここまでいくと、[場所]に[Nuxeo]が追加されています。これをダブルクリックすると再び認証ダイアログが表示されるので、ユーザ名とパスワードを入力してください。

あとは[Default Domain]-[Workspaces]-[test]フォルダーに適当な名前を付けて保存するだけです。今回は[テスト.odt]としました。

確認

Nuxeoにログインし、先ほど作成したファイルを

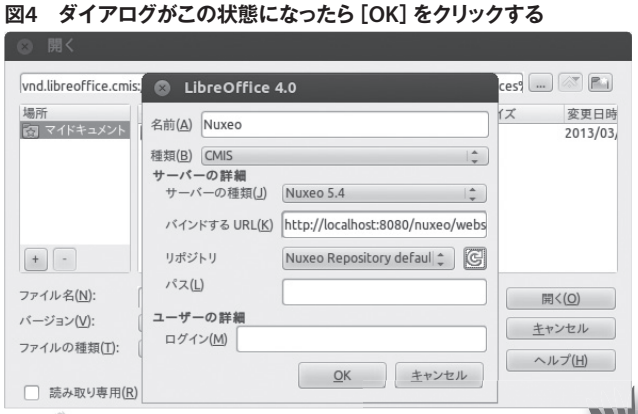
注10) Ubuntu 10.04でも4.0が同じ方法でインストールできるようですが、テストを行なっていない上にサポートも4月で切れてしまうので推奨しません。

注11) 4.0でフロッピーアイコンに戻されました。

図3　LibreOfficeを4.0にアップデートする

```
$ sudo add-apt-repository ppa:libreoffice/libreoffice-4-0
$ sudo apt-get update; sudo apt-get dist-upgrade
```

図4　ダイアログがこの状態になったら[OK]をクリックする



見てみます（**図5**）。名前の右にあるPDFアイコンをクリックすると、PDFで表示ができます。また、右端の［More］の横にある目のアイコンをクリックすると、プレビューが表示できます。これらはLibOの機能を使用しています[注12]。

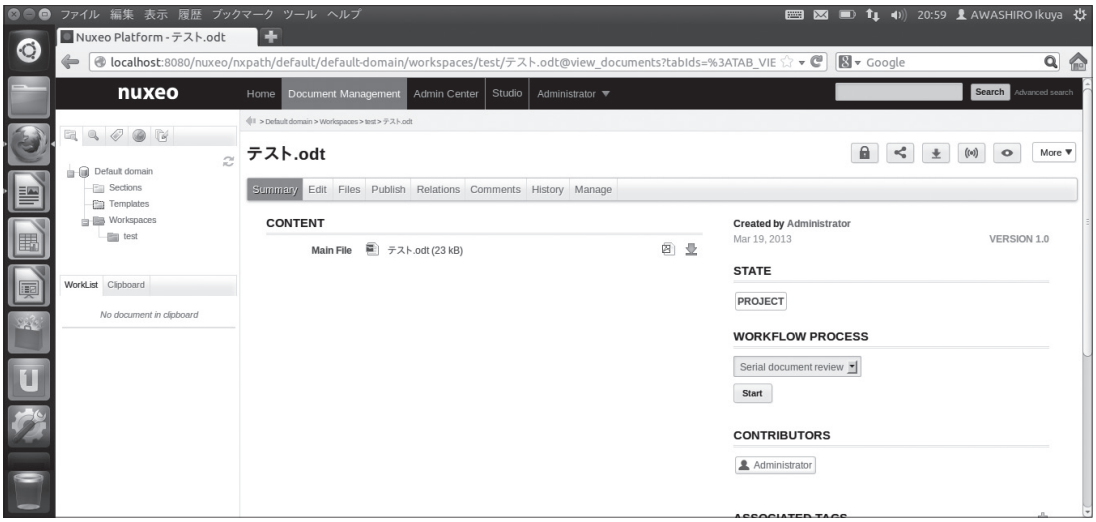
［History］をクリックすると、ドキュメントの履歴が見られます。リビジョン番号は［Edit］の下方にある［Update versions］を［Increment minor version］ないし［Increment major version］を選択し、［Save］をクリックすると上げることができます。

より詳しい活用方法はドキュメント[注13]をご覧ください。機能が膨大で、本連載内では紹介しきれませんので。SD

注12）ちなみにLibOのバージョンを12.04既定の3.5のままにしておくと、プレビューやPDFのフォントに問題が生じますので、4.0へのアップデートを推奨します。

注13）http://doc.nuxeo.com/display/ADMINDOC56/Installation+and+Administration

図5　先ほど保存したファイルを表示したところ



Ubuntu 12.04.2

今から12.04をインストールする場合は、2月にリリースされた12.04.2になるのではないかと思います。このバージョンからカーネルとXスタックが12.10をバックポートしたものになります。具体的には、カーネルのバージョンが3.5に、Xのバージョンが1.13になります。ほとんどの場合はこれで問題ないと思いますが、Xのバージョンに依存するようなパッケージを使用している場合はインストールできないということもあります。

そのような場合は、12.04.1のインストーラ[注1]からインストールし、最新の状態に更新してください。明示的にインストールしない限り、カーネルもXもバックポートしたものは使用されません。

注1）http://old-releases.ubuntu.com/releases/12.04.1/から取得してください。

第14回 Linux 3.9の新機能 ～Intel PowerClampの紹介～

Text：青田 直大　AOTA Naohiro

今回は開発が進められているLinux 3.9の新機能について紹介します。Linux 3.9は3月10日にRC2が出ており、新しく入ってくる機能は出そろっています。

- Intel PowerClampドライバの追加
- 新しいスリープPM_SUSPEND_FREEZEの実装
- intel_pstateドライバの追加
- VSOCKの追加
- SO_REUSEPORTの追加
- Goldfishのサポート
- ARM上でKVMが動作するように
- BtrFSがRAID5/RAID6に対応
- ftraceにsnapshotの実装

今回はこの中からIntel PowerClampを中心に紹介します。また、話の流れの中で、PM_SUSPEND_FREEZEとintel_pstateについてもふれます。

Intel PowerClampドライバ

最近、電力消費を抑えつつ高いパフォーマンスを実現することは1つの目標となっています。今までは、電力を抑えるために、CPUを使用していない時にCPUの動作周波数を落とす、あるいはCPUを完全にオフラインにする、などの方法がとられてきました。Intel PowerClampドライバは消費電力を抑えるための新たな方法を実現するドライバです。

ACPIにおける状態

Intel PowerClampの動作の解説に入る前に、ACPIの解説から始めましょう。ACPIはAdvanced Configuration and Power Interfaceの略で、その名のとおり電源管理のためのインターフェースです。このACPIではいくつかの電源の状態が定義されています(**図1**)。

まず、G0からG3の状態(G-state)があります。G-stateはマシンのグローバルな状態です。G0はコンピュータが動作中の状態、G1はサスペンドやハイバネートなどのスリープ状態、G2はソフトウェア的に電源オフの状態、G3は機械的に電源オフの状態です。G1では起動していたプロセスなどの動作状態が保存されているので、後でG0から移行してきたときの作業を継続できます。一方、G2でもG3でも動作状態は保存されていませんが、G2であればキーボードやWake On LANなどを使ってコンピュータを起動できるのに対して、G3では機械的に電源が切れているのでそういった手段は使うことができません。

これらグローバルな状態の下により、細かい状態が定義されています。G1の下にはS1から

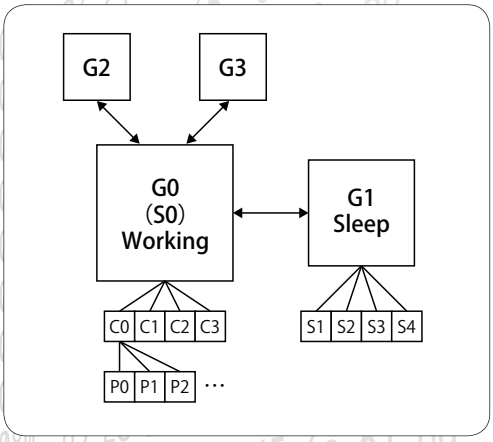
S4の状態(S-state)が定義されています。S-stateはマシンのスリープ状態です。

S1ではCPUの動作が止まっていますが、CPUは通電されています。S2ではCPUへの電力も停止しますが、メモリへの電力供給は残っています。このとき、キャッシュなどの設定が消えてしまっているので、復帰にあたってOSがそれを考慮して設定を戻す必要があります。S3はS2とほとんど同じです。S2よりも多くのキャッシュがクリアされたり、S2では復帰に使うことができるデバイスが使えないなどとS2よりも電力消費が小さくなっています。このS3が一般にサスペンドや、メモリへのサスペンドと呼ばれているものです。

S4ではメモリからも電力がカットされ、メモリの中身も消えてしまいます。つまり、復帰できるようにするにはメモリの内容をディスクに書き出しておいて、あとで戻す必要があります。一般にハイバネートやディスクへのサスペンドと呼ばれているものです。また、G0のことをS0、G2のことをS5と呼ぶこともあります。

G0の下にもC0からC3の状態(C-state)が定義されています。C-stateはCPUの動作状態を示しています。C0は通常の動作状態です。C1はCPUの動作が止まった状態です。この状態はすべてのCPUがサポートしています。C2、C3ではその順にC1より消費電力が少ないCPUのアイドル状態です。C2とC3との違いは、C2ではキャッシュのコヒーレンシが保たれているのに対して、C3では保たれていないことです。

▼図1　ACPIの状態



コヒーレンシ

コヒーレンシというのはデータの整合性ということです。たとえば、図1のようなマルチプロセッサのシステムを考えてみましょう。メインメモリはCPU間で共有されていますが、各CPUがそれぞれキャッシュを持っています。ここで、CPU0がメモリを読み込みそのデータをキャッシュに乗せたあとで、CPU1がその部分のデータを更新したとします。すると、CPU0のキャッシュの内容は実際のメモリの内容と変わってしまいますね。そこで、たとえばCPU1からメモリに書き込むと同時にCPU0に書き換えたことを通知します。通知を受け取ったCPU0は自分のキャッシュからデータを削除し、キャッシュとメモリの内容が不一致になることを防ぎます。

この機能を無効にすることで、C3ではC2よりも消費電力をおさえているというわけです。プロセッサによってはC4、C5などより消費電力が小さいアイドル状態をサポートしているものもあります。いずれにせよ状態の番号が大きくなるほど、消費電力は小さくなり、C0の状態に復帰し、動作を開始するまでには時間がかかるようになります。

G0の下、C0の下にさらにP0、P1といった状態(P-state)が定義されています。P-stateはCPUの動作周波数の状態を示しています。P0では一番多くの電力を使い、周波数も大きく、すなわちパフォーマンスが一番大きい状態になっています。そして、P1、P2と番号が大きくなるごとにより消費電力と周波数とが小さくなり、パフォーマンスが低下していきます。

最後にもう1つ、CPU以外のデバイスについて、D0からD3の状態(D-state)があります。D0は完全に電源が入っている状態で、逆にD3では完全に電源が入っていない状態で、再度デバイスを使用するには初期化しなおす必要があ

ります。D1、D2については明確な定義はありません。それぞれのデバイスクラスによっては、これらの状態が存在しないこともあります。いずれにせよ、今までの状態と同様に番号が大きくなるほど、消費電力は小さくなりますし、復帰し使用できるようになるまでの時間は大きくなっていきます。

Linuxでのインターフェース

ここからは、これらACPIでの電源状態(S-state、C-state、P-state)が今どのようにLinuxで使われているのかを見ていきます。

まず、S-stateの中ではS1、S3、S4がサポートされています。/sys/power/stateを読み出すとそのマシンでサポートされているスリープ形式が、“freeze standby mem disk”のように列挙されます。ここで“standby”をstateに書くとCPUがS1状態、その他デバイスが(可能であれば)D1状態にされます。同じように“mem”ではS2状態とD3状態になり、いわゆるメモリへのサスペンドが行われ、“disk”ではS4状態になり、ディスクへのサスペンドが行われるので電源を完全に切ってしまうことも可能となります。

さて、ここで“freeze”が説明されていないことに気がついたかもしれません。実はこれも3.9で追加された[注1]ACPIを使わない「疑似サスペンド」です。この方法ではすべてのプロセスをフリーズし、デバイスをサスペンドして、CPUをなるべく深いC-state(使えるC-stateの中で一番数が大きいもの)におきます。この方法はちょうどサスペンドと通常の動作状態の中間にあるようなイメージで、電力消費も通常よりは低く、サスペンド中よりは高くなります。これはACPIのサスペンド機能を使っていないので、たとえばACPIでサスペンドが実装されていなかったり、あるいは実装されていても壊れているようなマシンでも電力を抑えることができるというメリットがあります。

wakeup source

また、/sys/power/stateの代わりに/sys/power/autosleepに同様に文字列を書くと、スリープを妨げるロック(wakeup source)がなくなった時点で自動的に対応するスリープ状態に入るようになります。wakeup sourceはカーネル内部だけでなく、ユーザ空間からも作ることができます。/sys/power/wake_lockに“<名前><ナノ秒>”(またはナノ秒の部分を省略)の形式で書き込むとその名前のロックが作られます。このwakeup sourceは指定したナノ秒数が過ぎるか、/sys/power/wake_unlockに名前を書くことで削除されます。なお、ナノ秒を省略している場合は、時間経過で削除されることはありません。

C-state

次にC-stateについて見ていきます。C-stateはcpuidleというサブシステムで取り扱われています。復習となりますが、C-stateは番号が大きくなるほど(深いほど)消費電力は小さくなりますが、その代わりC0に戻り動作を開始するまでの時間はかかるようになります。つまり、CPUがすることがなくなったからといって、すぐさま一番深いC-stateに入っていては、ケースによってはかえって1ワットあたりのパフォーマンスは下がってしまうことにもありえます。そこで、どの程度の消費電力になるのかと、復帰までにどの程度の時間がかかるのかなどの情報をもとにしてどのC-stateに入るのかを決めます。

このアルゴリズムには、現状“ladder”というものと“menu”というものとがあります。ladderのほうは割とシンプルなアルゴリズムで、ladder(はしご)という名前のとおり、最初CPUがアイドルになるとC1に入り、その後1段1段はしごを降りていくかのように、アイドル時間が長くなっていくごとに1つ深いC-stateへと入っていきます。

もう一方のmenuはより複雑なアルゴリズムとなっています。こちらはladderのように順次切り

注1) commit 7e73c5ae6e7991a6c01f6d096ff8afaef4458c36

替えていくのではなく、現在のシステムの状態を考慮して、そのときに最適なC-stateを選択していきます。具体的には、まず次のタイマー割り込みが起こる時間を考えます。次のタイマー割り込みが遠ければ深いC-stateに入っても元がとれるでしょう。しかし、単純な次のタイマー時刻だけでは、実際には次の割り込み時刻の良い予想にはなりません。これはタイマーよりも前にほかの外部からの割り込み(たとえば、キーボード入力など)が入ってくるからです。こうした実世界の状況に対応するために、履歴をもとにして次のタイマーまでの時間に修正がかけられます。たとえば、前回次のタイマーまでの時間の半分が過ぎたところで割り込みが入っていたのであれば、次回のタイマーまでの時間に0.5をかけて予測値とします。

これだけではなく、I/Oが実行されているか、定期的な短い間隔での割り込みが行っているかどうか、状態の移行にかかる時間がシステムの現状の負荷を考えてみて大き過ぎないか(システムに負荷がかかっているときに、移行に時間をとられる状態へと移るとパフォーマンスを下げる要因となってしまいます)を考慮しています。

P-state

最後にP-stateについて見てみましょう。P-stateはcpufreqというサブシステムで取り扱われています。周波数が大きくなるほど、CPUのパフォーマンスも上がりますが電力も多く消費するようになります。省電力とパフォーマンスのバランスをとるには、CPUの実行状況に応じて最適なP-stateを選んでいく必要があります。cpufreqでのP-stateの選び方には“performance”、“powersave”、“ondemand”、“conservative”という4つの方法があります[注2]。

注2) 正確にはユーザ空間のプログラムに任せる“userspace”という方法もあります。

“performace”は常に最大の周波数を使い、逆に“powersave”は常に最小の周波数を使います。“ondemand”はある一定期間のCPU使用率がしきい値(デフォルトで80%)を越えると、一気にCPUを最大の周波数へと変更します。その後、負荷が落ち着いてくればCPU使用率がしきい値からさらにある一定の値(デフォルトでは10%)を下まわるような周波数を選択します。“conservative”も“ondemand”と同様に動作状況にあわせて周波数を変えていく方法です。ただし、こちらは負荷がかかったときにondemandのように一気に最大の周波数までいくのではなく、最大周波数の5%ずつ周波数を変えていきます。バッテリーで動作しているようなマシンではondemandよりかはconservativeのほうが適しているようです。

intel_pstate

さて、ここで少し寄り道をしてintel_pstateについてふれましょう。これもLinux 3.9で新しく追加されるcpufreq用のドライバです。このドライバではperformanceかpowersaveのどちらかのみを選択できます。performaceのときは最大の周波数に固定し、powersaveのときはCPUの状態に応じて周波数を変化させます。これはその名のとおり、IntelのCPUの機構に依存しています。今のところIntel Sandy BridgeのCPUで使うことができます。conservativeではカーネルの統計情報からCPUの使用率を算出していますが、intel_pstateではモデル固有レジスタ(MSR)を読むことでより正確にCPUの使用率を算出しています。この使用率をbusyとして次のような式を使って、P-stateの変更度合いのstepを計算します(**図2**)。

つまり、setpointが理想のbusy率となっており、errがその理想までの差分になります。integralは理想状態との差分のこれまでの和と

▼図2 P-stateの変更度合いのstepの計算

```
err = setpoint - busy
integral += err
step = err*p_gain_pct + integral*i_gain_pct + (err-last_err)*d_gain_pct
last_err = err
```

なります。さらに、前回のerrと今回のerrとの差分もstepの計算に使われています。細かい値については、/sys/kernel/debug/pstate_snb/にて**図3**のように設定／参照できます。

この値であれば、p_gain_pctが一番大きく17なので理想状態との差が重視されています。i_gain_pctが4であるので継続的に理想状態との差が大きくなっていれば、より速く周波数が上昇するようになっています。一方で、d_gain_pctは0なので前回のerrとの差については考慮されていません。

さらに、intel_pstateでは周波数が最小の状態が5回続くとnormal modeからidle modeへとmodeを変化させ、周波数を最大にします。idle modeでは、上の式をbusyの代わりに100-busyを使って計算します。つまり、idle率がsetpointになるように周波数を調整していきます。idle modeに入っているときというのはCPUにやることがないような状況なので、周波数を大きくしてもidle率が大きくなり、より長く深いアイドル状態(C-state)に入るようになるので、消費電力が低下することが期待できます。

さて、LinuxでのS-state、C-state、P-stateの使い方見てきました。これらの使い方を省電力の観点から見てみると、積極的／自発的に消費電力を抑制する方法と、機会があれば消費電力を抑制する消極的な方法とに分けられます。たとえば、suspendやpowersaveはマシンの能力を制限することで積極的に省電力を行っています。一方でcpuidleやondemandなどはCPU状況から予測して機会があればという消極的な省電力となっています(**表1**)。

▼図3　/sys/kernel/debug/pstate_snb/の表示

```
# cd /sys/kernel/debug/pstate_snb/
#  grep . *
d_gain_pct:0
deadband:0
i_gain_pct:4
p_gain_pct:17
sample_rate_ms:10
setpoint:109
```

▼図4　cur_state、max_state、typeの値

```
# pwd
/sys/class/thermal/cooling_device13
# grep . cur_state max_state type
cur_state:-1
max_state:50
type:intel_powerclamp
```

▼表1　S-state、C-state、P-stateの積極的／消極的な使い方の分類

	S-state	C-state	P-state
積極的	suspendなど	Intel PowerClamp	powersave
消極的	autosleep	cpuidle	ondemandなど

Intel PowerClampドライバのしくみ

それではIntel PowerClampドライバのしくみについて見ていきましょう。これは一言でいえば、積極的にCPUをアイドル状態にすることでCPUの電力消費を抑えるドライバになっています。このドライバはthermalシステムの一部として動作します。ドライバを読み込むと、/sys/class/thermalの下にcooling_device<n>というディレクトリができます。わたしのシステムの場合は、cooling_device13となっています。typeファイルが“intel_powerclamp”となっているのでそれで判別できるでしょう。

図4を見ると、今はcur_stateが-1となってPowerClampは無効になっています。max_stateに50とあるように、cur_stateに1から49までの数を書きます。ここで指定されたものが目指すアイドル状態の割り合いとなります。すると、CPUごとにそのCPUに固定された“kidle_inject/<CPU番号>”というカーネルスレッドが起動されます。このスレッドは通常よりもかなり高い優先度でスケジュールされ、以下のループを実行します。

- intervalの倍数となる時刻までスレッドをスリープする
- 6ミリ秒の間、一番深いC-stateに入る

intervalというのはC-stateに入っている期間が設定された割り合いになるような期間のことです。たとえば、cur_stateに30を書いている場合、

C-stateに入っている期間が30%となるように、このようにintervalが計算されます。

```
interval = 6 (msec) * 100 / 30
```

基本的な動きとしては以上となりますが、実際にはもう少し実世界用に調整が加えられています。

1つめはC-stateに入っている6ミリ秒の間に割り込みにより何度も起こされたときの扱いです。多くの割り込みが入ってきている場合、それらの割り込みで起こされてしまっている分、C-stateに実際にいた時間が短くなっているはずです。そこで、あるしきい値よりも多くの割り込みを受けている場合にはアイドルでいる割合が2倍になるように調整されています。

もう1つは誤差の調整です。C-stateへの移行および復帰にも時間をとられます。CPUのMSRから実際にC-stateにいた時間を知ることができますから、それをもとにしてアイドル時間の割合の調整を行います。この調整値の状態を、

```
/sys/kernel/debug/intel_powerclamp/
powerclamp_calib
```

で知ることができます(**図5**)。confidenceは割り込みが多くなかった期間の回数(ただし3回までしか数えていません)、steadyはそのときの設定の割合と実際のC-stateの割合との差、dynamicは今のところ使われておらず、常に0になっています。設定中の割合およびそのまわりのconfidenceが3以上になっている(つまり、十分にサンプルが集まっている)ときに、それら3つのsteadyの平均を調整値とします。たとえば、48%に設定している場合には47、48、49のsteadyの平均をとって(3 + 5 + 4)／3 = 3となります。46%に設定している場合には、46のconfidenceがまだ1なので調整は行われません。

dynamicの部分が表示されているのに使われていなかったりとまだ未完成な部分もあるようですが、積極的にC-stateに入ることで省電力を達成するという今までとは変わった電力制御法となっています。コードと一緒にコミットされているドキュメント[注3]によれば、CPUをオフラインにするよりも1ワットあたりのパフォーマンスが40%も向上しているようです。

まとめとその他の追加機能

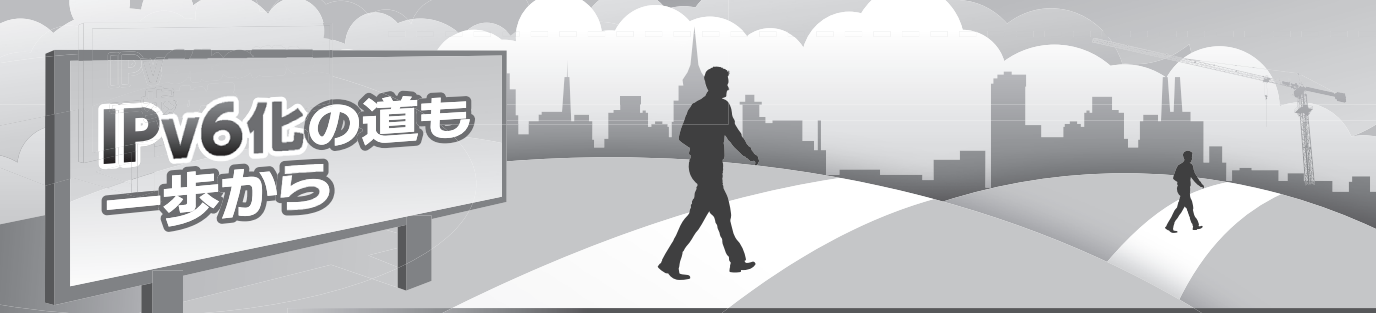
最初に挙げたようにIntel PowerClamp以外にもLinux 3.9にはいろいろな機能が導入されることになります。ここでいくつか簡単に紹介しておきましょう。VSOCKはVMWareでホスト⇔ゲスト間の通信、ゲスト⇔ゲスト間の通信に使われるソケットです。TCP、UDPと同じように使うことができます。Goldfishは Android開発に使われている仮想的なCPUです。このCPU上でLinux kernelを動かすパッチがメインラインへとマージされました。

今回はLinux 3.9に導入される新機能をIntel PowerClampを中心に見てみました。Power Clamp以外にもPM_SUSPEND_FREEZEやintel_pstateと電力関連でおもしろい機能が導入されています。PowerClampのdynamicが常に0なのがわかるようにまだまだ開発途上というところも見えますが、ワットあたりのパフォーマンスが良いといいますし、注目していきたい機能です。SD

注3)Documentation/thermal/intel_powerclamp.txt

▼図5　アイドル時間の割合

```
# cat /sys/kernel/debug/intel_powerclamp/powerclamp_calib
controlling cpu: 0
pct confidence steady dynamic (compensation)
0       0       0       0
1       0       0       0
2       2       1       0
...
45      3       2       0
46      1       5       0
47      3       3       0
48      3       5       0
49      3       4       0
```



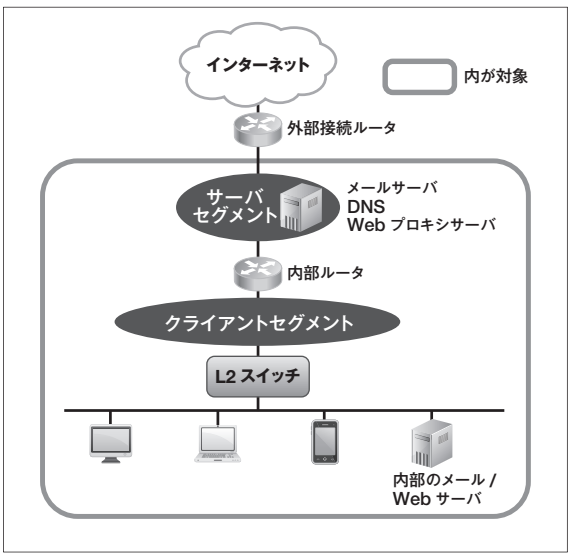
第6回 ネットワーク構築時の注意点と落とし穴

IPv6普及・高度化推進協議会　IPv4/IPv6共存WG　アプリケーションのIPv6対応検討SWG
廣海 緑里 HIROMI Ruri　**渡辺 露文** WATANABE Tsuyufumi　**新 善文** ATARASHI Yoshifumi　**藤崎 智宏** FUJISAKI Tomohiro

前回のおさらい

前回は、試験ネットワーク構築に向けたアドレス計画の話を進めました。今回は計画に従って機材にアドレスを設定し、設定内容を確認してみましょう。試験環境の構成を**図1**に、アドレスの割り当てを**表1**に示します。デュアルスタック端末を接続した場合に、ネットワークがIPv4だけで運用されている場合とデュアルスタックの場合では挙動が違うことを、実際の通信パケットをキャプチャしながら比較します。

▼図1　試験環境構成図



ルータの設定

サーバセグメントには2001:db8:0:1::/64、クライアントセグメントには2001:db8:0:2::/64を割り当てます。**図2**を参照してください。ルータのサーバセグメント側インターフェースには、2001:db8:0:1::1/64を設定します。サーバセグメントに接続する端末はアドレス自動設定は利用しないため、RA(ルータ広告)の設定は行いません。

ルータのクライアントセグメント側インターフェースには、2001:db8:0:2::1/64を設定します。このセグメントに接続される端末はアドレス自動設定を利用してノードのアドレス生成を行います。RAで2001:db8:0:2::/64のプレフィックスを流すようにします。

▼表1　試験環境のGUAアドレス設計の整理

	利用プレフィックス	割り当て	設定方法
試験ネット	(サーバセグメント)2001:db8:0:1::/64 (クライアントセグメント)2001:db8:0:2::/64	―	―
ルータ	(サーバセグメント)2001:db8:0:1::1/64 (クライアントセグメント)2001:db8:0:2::1/64	固定割り当て	手動設定
サーバ	2001:db8:0:1::10	固定割り当て	手動設定
クライアント	2001:db8:0:2:fffe:[EUI64アドレス]	動的割り当て	自動設定

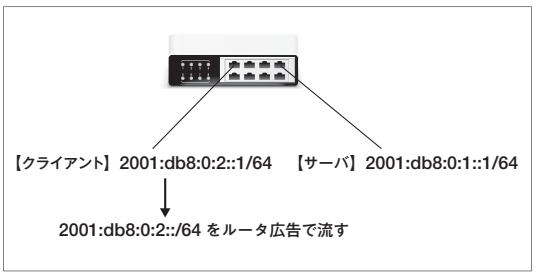
設定内容と方法

ルータによって細かいコマンドは違いますが、おおむね**図3、4**の設定を投入します。各種のルータのコマンドの例を掲載しましたので参考にしてください。これらのコマンドは各社のWebサイトのマニュアルページで確認できます。また、コマンドを投入するには設定モードを変更したり、付帯する情報設定が必要だったりします。バージョンによってコマンドの投入順序が変わっている場合もあります。実際に利用する場合には、コマンドリファレンスなどでご利用のルータOSで使えることを確認のうえ、正確な文法で設定してください。

確認方法

ルータの設定が完了したら、インターフェースの情報を見るコマンドなどで設定状況を確認し(**図5**)、それぞれのインターフェースに付与

▼図2　ルータの設定



▼図3　サーバセグメントのインターフェース設定例[注1]

①インターフェースのIPv6プロトコルを有効にする
`ipv6 enable`（Cisco IOS、NEC IXシリーズ、AlaxalA AXシリーズ）
②WANのインターフェースにIPv6アドレスを設定する
`ipv6 address 2001:db8:0:1::1/64`（Cisco IOS、NEC IXシリーズ、AlaxalA AXシリーズ）
`interface lan0 add 2001:db8:0:1::1/64`（IIJ SEIL）
`interface ge-0/0/0 unit0 family inet6 address 2001:db8:1:1::1/64`（Juniper Junos）
`ipv6 lan1 address 2001:db8:0:1::1/64`（YAMAHA RTXシリーズ）
③WAN側ではRAを動作させないようにする
`ipv6 nd ra suppress`,`ipv6 nd ra suppress all`（Cisco IOS ※allが使える場合はRSの受信も抑制される）
`ipv6 nd suppress-ra`（AlaxalA AXシリーズ）
④IPv6パケットの転送設定をする
`ipv6 unicast-routing`（Cisco IOS）
`security forwarding-options family inet6 mode flow-based`や`security policies`（Juniper Junos）

▼図4　クライアントセグメントのインターフェース設定例[注1]

①インターフェースのIPv6プロトコルを有効にする
`ipv6 enable`（Cisco IOS、NEC IXシリーズ、AlaxalA AXシリーズ）
②LANのインターフェースにIPv6アドレスを設定する
`ipv6 address 2001:db8:0:2::1/64`（Cisco IOS、NEC IXシリーズ、AlaxalA AXシリーズ）
③RAの設定をする
`ipv6 nd ra enable`（NEC IXシリーズ）
`rtadvd enable`（IIJ SEIL）
`protocols router-advertisement interface fe-0/0/2.0 prefix 2001:db8:0:2::/64`（Juniper Junos）
`ipv6 prefix 1 2001:db8:0:2::/64`、`ipv6 lan2 rtadv send 1`（YAMAHA RTXシリーズ）
④IPv6パケットの転送設定をする
`ipv6 unicast-routing`（Cisco IOS）
`security forwarding-options family inet6 mode flow-based`や`security policies`（Juniper Junos）

注1) 今回参考にしたルータの情報は次のとおり。 **AlaxalA AXシリーズ**(http://www.alaxala.com/jp/techinfo/archive/manual/AX6000S/html/11_7/CFGUIDE3/INDEX.HTM) **Cisco IOS**(http://www.cisco.com/cisco/web/portal/support/docs_listing.html?cid=282770988&locale=ja_JP&itag=prod_conf_guides_list) **IIJ SEIL**(http://www.seil.jp/support/tech/doc/command/) **Juniper Junos**(http://www.juniper.net/techpubs/en_US/junos12.3/information-products/pathway-pages/product/12.3/index.html) **YAMAHA RTXシリーズ**(http://jp.yamaha.com/products/network/routers/rtx1200/)

したアドレスに対して、ping(ping6)コマンドで応答を確認してみます。応答が返ってくることを確認できたら、クライアントを接続していきます。

ルータ設定の注意点

ルータ設定の注意点は、何と言ってもアドレス表記がIPv4とは変わっていることと長くなっていることで、タイプミスをしやすい点です。設定が問題なく完了しても、確認作業の際にタイプミスをして、「あれ？ pingが届かない」となることもしばしばあります。IPv6アドレスに慣れてくると多少はタイプミスも軽減されてきますが、大規模になるとどうにもならなくなります。

サーバであれば、DNSに登録してFQDNで日々の確認作業をしていくなどが可能ですが、DNSにあまり登録することのないルータなどはどうすればよいのでしょうか。これからあちこちで運用経験が積まれて、情報交換されてくるとよいアイデアが生まれるかもしれません。

IPv4のマスクの整合性確認などで苦労されている場合は、セグメントは/64で統一できるためマスクの計算をしなくてよい分、整合性に関係したミスは減ると思われます。

なお、IPv6アドレスの表記については当初は柔軟な書き方が可能でしたが、オペレーションの観点では柔軟性は混乱のもとになるということ で、RFC5952(A Recommendation for IPv6 Address Text Representation)で表現方法の基準が規定されています。たとえば、アルファベット部分は小文字にするといったことも含まれていますが、ルータベンダによってこのRFCの採用状況はまちまちです。さすがに、小文字でも大文字でも入力は受け付けられますが、設定を

▼図5 ルータのインターフェースの状態確認例(YAMAHA)

```
>show ipv6 address

LAN1 scope-id 1 [down]
 Received:    0 packet 0 octet
 Transmitted: 0 packet 0 octet

 global     2001:db8:0:1::1/64 (tentative)
 link-local ff02::1/64
 link-local ff02::2/64
 link-local ff02::1:ff00:1/64

LAN2 scope-id 2 [down]
 Received:    0 packet 0 octet
 Transmitted: 0 packet 0 octet

 global     2001:db8:1:1::1/64 (tentative)
 link-local ff02::1/64
 link-local ff02::2/64
 link-local ff02::1:ff00:1/64
```

▼図6 Windows 7のipconfigの結果

```
C:¥Users¥test>ipconfig

Windows IP 構成


Wireless LAN adapter ワイヤレス ネットワーク接続:

   メディアの状態. . . . . . . . . . . : メディアは接続されていません
   接続固有の DNS サフィックス . . . :

イーサネット アダプター ローカル エリア接続:

   接続固有の DNS サフィックス . . . : test.example.com
   IPv6 アドレス . . . . . . . . . . . : 2001:db8:0:2:19ef:8168:5b62:1098
   一時 IPv6 アドレス. . . . . . . . . : 2001:db8:0:2:180e:e1be:dd7b:b6e8

   リンクローカル IPv6 アドレス. . . . : fe80::19ef:8168:5b62:1098%2
   IPv4 アドレス . . . . . . . . . . . : 192.168.0.101
   サブネット マスク . . . . . . . . . : 255.255.255.0
   デフォルト ゲートウェイ . . . . . : fe80::20b:97ff:fe96:e34%2
                                        192.168.0.1
```

① Windowsが独自に生成

②ランダムな初期値をMD5でハッシュして生成

確認してみると大文字で表示されたりします。

クライアント端末の接続と確認

今回はクライアントとしてWindows 7を例にとります。Windows 7はIPv6はデフォルトONですので、とくにクライアント端末側での設定なしでそのまま接続してみます。GUIの「ネットワークと共有センター」でアダプタを指定して「ネットワークの詳細」にIPv6アドレスが設定されていることを確認してみましょう。コマンドプロンプトから「ipconfig」と入力して確認することもできます。**図6**はipconfigで得られた情報です。

前回説明したSLAACによるアドレス生成ですが、Windows 7では拡張EUI-64のインターフェースIDはついていません。「IPv6アドレス」に表示されるアドレスをよくみると、EUI-64由来のインターフェースIDではありません。Windows Vista以降のOSでは、独自に生成したインターフェースIDが使われています(図6-①)。

Windows 7では、「SLAACのプライバシー拡張」と呼ばれるRFC4941で規定されているインターフェースIDを用いています。SLAACのプライバシー拡張によるアドレスは、ipconfigの結果では、「一時IPv6アドレス」として表示されます(図6-②)。

これらは、EUI-64由来のインターフェースIDではMACアドレスが簡単にわかってしまい、プライバシーの漏洩につながる[注2]ことに配慮したものです。

比較として、Mac OS Xのifconfigの結果を挙げておきます(**図7**)。Mac OS Xでは、MACアドレスの先頭でUniversal(1)/Local(0) bitを反転させ、間にff:feを挿入した拡張EUI-64のインターフェースIDが利用されているのがわかります(図7-①)。EUI-64ベースのインターフェースIDの生成については、RFC4291のAppendix A: Creating Modified EUI-64 Format Interface Identifiersに詳細があります。

このようにインターフェースIDには、modified EUI-64方式／SLAACのプライバシー拡張／独自生成のいずれかが採用されています。**表2**にそれぞれの特徴をまとめておきます。

クライアントにIPv6アドレスが設定されたことを確認できたら、クライアント／ルータ双方からpingで疎通確認してみましょう。

最後にクライアント端末の挙動について、IPv6対応のネットワークがある場合とない場合で通信パケットの様子を比較してみます。

まず、IPv4のシングルスタックのネットワークにIPv4/IPv6デュアルスタックの端末を接続

注2) プレフィックスが変わってもインターフェースIDは同じですので、どこで接続していたかやアクセスログの解析と合わせると行動がわかってしまう。

▼図7　Mac OS Xのifconfigの結果

```
$ ifconfig

lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
  options=3<RXCSUM,TXCSUM>
  inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
  inet 127.0.0.1 netmask 0xff000000
  inet6 ::1 prefixlen 128

en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
  ether 10:40:f3:81:f1:d2
  inet6 fe80::1240:f3ff:fe81:f1d2%en0 prefixlen 64 scopeid 0x4   ←①拡張EUI-64のインターフェースID
  inet 192.168.0.108 netmask 0xffffff00 broadcast 192.168.0.255
  inet6 2001:db8::2:1240:f3ff:fe81:f1d2 prefixlen 64 autoconf
  inet6 2001:db8::2:119e:b748:5b43:a911 prefixlen 64 autoconf temporary
  media: autoselect
  status: active
```

した場合です。Wiresharkでのキャプチャ結果が**図8**です。

IPv6が有効になっているOSでは、ネットワークに接続すると近隣探索プロトコルのNeighbor Solicitation(NS)やRouter Solicitation(RS)のメッセージがマルチキャストアドレス宛てに送信されはじめます。Windows Vista以降のWindows OSでは、LLMNR(Link-Local Multicast Name Resolution、RFC4795)といったプロトコルもIPv4とIPv6を使っているため、IPv6のパケットを送信します。「ネットワークと共有センター」のGUIやipconfigコマンドを使ってネットワークの設定情報を見るとIPv6のアドレスとしてはリンクローカルアドレスが確認できるだけですので、一見するとIPv6に関するパケットが出ていることは想像できないかもしれません。しかし、IPv6がデフォルトでONになっているデュアルスタックの端末であるということは、IPv6のパケット送信ができるということです。

セキュリティに関しては別の機会に解説する予定ですが、IPv4だけでネットワーク運用しているからIPv6のセキュリティ対策はしなくてもよいと考えていると、このような状況で不正なIPv6ルータがネットワーク上に存在した途端にセキュリティホールとなってしまいます。デュアルスタックのOSを導入し始めたら、IPv6の通信の監視を開始することをお勧めします。

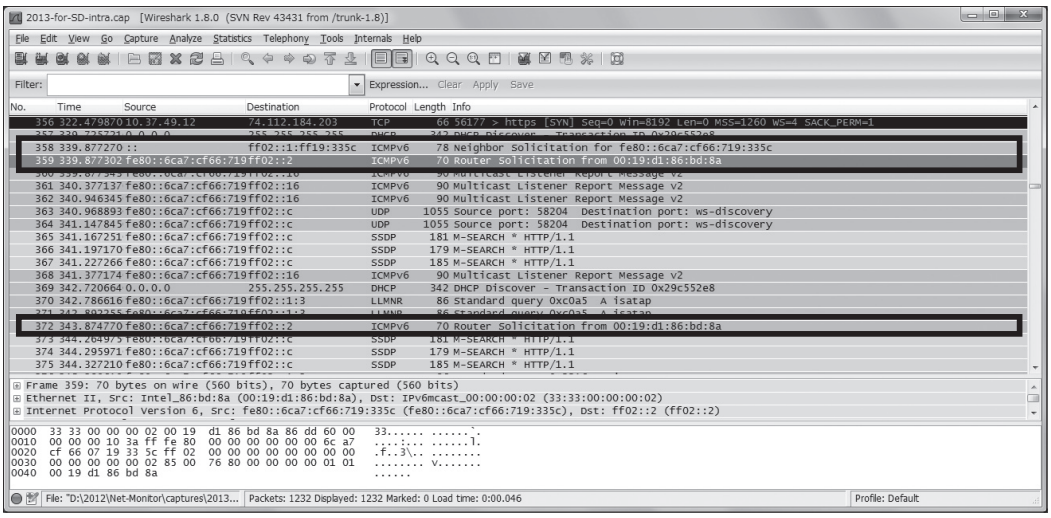
続いて、端末だけではなくネットワークもIPv4/IPv6デュアルスタックに対応している場合の挙動を見てみましょう(**図9**)。

ネットワークがシングルスタックの場合は、NSやRSを送信しても応えるノードがいないため、片方向の通信で終わっていました。しかし、ネットワークがIPv6対応していると、NSにはNeighbor Advertisement(NA)が、RSにはRouter Advertisement(RA)が返るようになり、それらの情報を受けて相手先端末とIPv6で通信するための準備作業が行われます。NDPのやりとりが正しく行われているかどうかを確認することは

▼表2　SLAACのアドレス

	参考RFC	推奨有効期限	最大有効期限	補足
modified EUI-64で生成したアドレス	4291	7日間	30日間	期限延長可
Privacy Extensions for SLAAC	4941	24時間	7日間	期限延長不可
Windows Vista/7が生成する独自のアドレス	—	7日間	30日間	期限延長可。再起動後もアドレスは変わらない

▼図8　デュアルスタック端末の振る舞い(IPv4ネットワーク)



とても大事な作業です。ここまで完了すれば、今回目標としている作業の半分は達成できたことになります。

外部との通信確認

外部IPv6ネットワークに接続している場合は、接続状況をモニターするサイトにアクセスしてIPv6での通信可否を確認できます。かつては、KAMEプロジェクト(http://www.kame.net/)で亀が踊るかどうかを見ていました。2チャンネルで「踊るひろゆき」が見えるかどうかも話題になったりしました。

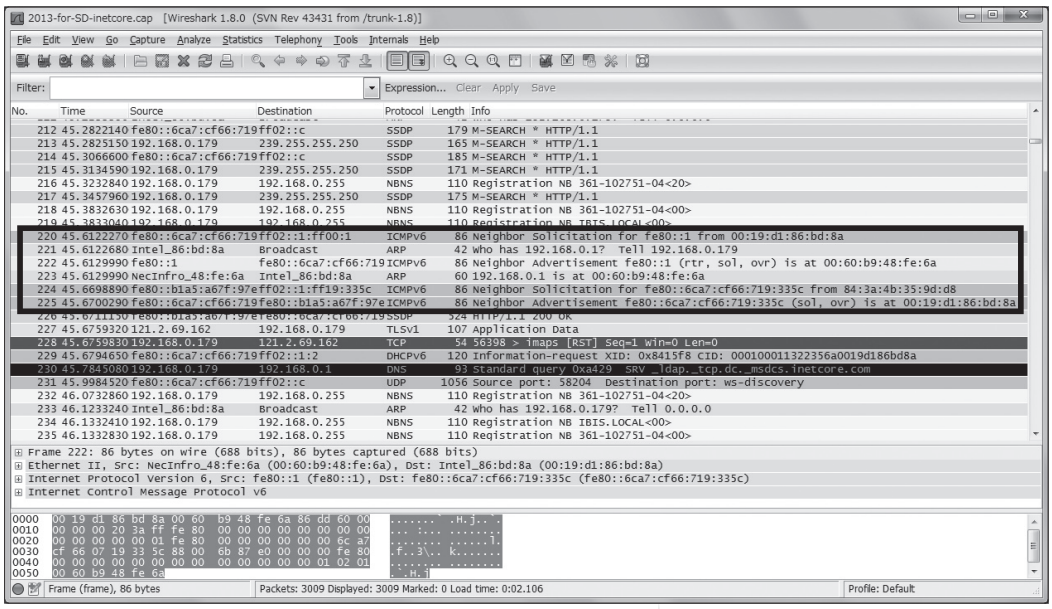
ここでは、もう少し情報量の多いサイトとして、TEST-IPv6のサイト(http://test-ipv6.com/)を紹介します。このサイトでは、IPv6対応状況を10段階で評価した結果を表示してくれます。IPv6アドレスの有無のほか、DNSでAAAAが参照できるかどうか、PathMTUディスカバリーに問題はないかなどが評価されます。Webブラウザを使ってアクセスが問題なくできるかどうか総合的に判断できます。

終わりに

今回はルータに設定を行い、クライアントを接続して確認するところまでを解説しました。実際に運用する際には、ルータにフィルタ設定を入れるなど安全に運用するための対策をとることになりますが、基本の設定は驚くほど簡単であることがわかっていただけたと思います。アドレス計画もLANには/64が基本となっているため、「マスク長を間違えた！」といった失敗はありません。端末のアドレス設定についてもルータ広告を出すだけで、受け取ったクライアントが自動でノードのアドレスを生成する過程も見ていただきました。このしくみはリナンバリングも簡単に行えると言われていますが、SLAACの特徴で見たように一定期間使うため、短期間にリナンバする場合には、端末の再起動を徹底するなど注意が必要です。

次回は、この環境にメールやWebサーバを立ち上げていきます。お楽しみに。SD

▼図9　デュアルスタック端末の振る舞い(IPv4とIPv6に対応したネットワーク)



NO.19

Monthly News from jus
Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
松澤 太郎　MATSUZAWA Taro　Twitter @smellman

グリーのJenkins導入事例にみるCIの本質

今回は、2月に行ったJenkinsの勉強会（**写真1**）について報告します。

2013年2月勉強会報告

■グリーでのJenkins導入2年間を振り返る
【日時】2013年2月19日（火）18:30～20:30
【会場】四谷アクセア会議室 第1会議室
【講師】岡崎 隆之（グリー）
【司会】松澤 太郎（日本UNIXユーザ会）

継続的インテグレーション（Continuous Integration、以下CI）は今や多くの開発者が取り入れているツールとなりつつあります。今回の勉強会ではグリー㈱におけるJenkinsの導入や、その普及について紹介していただきました。講師の岡崎さんはグリーにおいてソーシャルゲームの国際化に努めており、その中でもJenkinsを活用しています。

▲ 写真1　勉強会の様子

講演ではJenkinsの歴史や導入方法をふまえたうえで、どのようにCIを普及させていくかを中心に紹介し、最後の質疑応答では経験を用いて問題となる点を多く伝えていただきました。

■歴史

JenkinsはもともとHudsonという名前でSun Microsystems社（以下、Sun）の川口耕介さん（現在はCloudBees社に在籍）が始めたプロジェクトでしたが、SunがOracle社に買収されたのを機にフォークしたものです。なお、Hudsonは現在、Eclipse Foundationのもとで開発が進められています。岡崎さんはHudsonの時代から使い続けています。

■CIツール

Jenkins以外にもCIを行うツールは多くあります。その中でもTravisはGitHubに対するもので、GitHubで動かしているプロジェクトでCIが可能になります。

■Jenkinsの基本

Jenkinsは基本的にプログラムの変更をコミットしたら定期的にテストとビルドを行い、テストやほかの調査ツール（カバレッジ計測など）の結果をレポートし、ビルド結果を成果物として保持することができます。また、多くのプラグインがあり、シェルを実行することもできるため、さまざまな環境に柔軟に対応できます。プラグインを開発したい場合は、ほかのプラグインを参考にするとよいとのことでした。

■実行方法

JenkinsはWebアプリケーションですが、WARファイルとして配布されているため、実行はとても簡単です。OSによってはWARファイルをダブルクリックするだけでサーバが立ち上がります。Servletコンテナの利用も可能ですが、グリーのような会社ですらServletコンテナは使っていないそうです。

■導入から普及

岡崎さんはまず自分のチームでJenkinsを動かし始め、自分たちのプロダクトから自動化を始めました。その後、周りのエンジニアがビルドの自動化などで悩んでいたらJenkinsを紹介して、自分たちが用意したJenkinsを使ってもらうことで徐々に周りに普及していったとのことでした。

■導入のポイント

Jenkinsを導入するときのポイントとして「変えるのはツールではなくワークフロー」であると岡崎さんは語りました。たとえば、作業が属人的になっている部分などをJenkinsに代替させるなど、Jenkinsを導入することで「少しだけ変える」ようにするとよいとのこと。

また、導入時は無理強いなどはせず、開発者を見守りかつサポートしていくことが重要だと言います。そして、導入ができたらBTS（Bug Tracking System）などのしくみと合わせてワークフローを変えていくとよいというアドバイスもありました。

■成果物（ビルド）の自動生成

Jenkinsはプロダクトのビルドを自動生成することが可能です。例として、iPhoneアプリなどではxcodebuildを通してJenkinsからビルドを行い、ビルドで生成されたファイルをJenkinsに保持したり、Enterpriseであれば生成されたものを自動的に配布できます。

■測定

ビルドの方法まで確立したら、測定ツールを導入することで改善のプロセスを進めるとよいとのことでした。測定による品質基準を少しずつ変えていき、それを徐々にワークフローに入れていくことで多くの改善が行えるようになったそうです。測定ではコードのフォーマットやカバレッジ測定などが利用でき、多くのプラグインがあると説明されていました。

■マルチ構成

Jenkinsは設定しだいでマスターにもスレーブにもなります。そのしくみは非常に柔軟で、スレーブは常時つながっている必要がなく、たとえばノートパソコンにJenkinsを入れておいて、ネットワークに接続したときだけスレーブとして動かすといったことも可能です。また、マルチ構成を使っていろいろなブラウザでテストを回すといったことができます。

■質疑応答

最後の質疑応答において岡崎さんは、「昔、在籍していた会社で、ビルドのためだけに人がいたのがもったいなかった」と言っていました。大規模なプロダクトの開発になると、そういう役目の人はいまだにいると思います。このようなところから少しずつ改善していくためにCIはとても重要な考えだと思いました。

■終わりに

岡崎さんは現在このテーマで@ITにて次のような連載記事を執筆しています。

グリーはいかにしてJenkinsを導入したのか（1）
http://www.atmarkit.co.jp/ait/articles/1302/13/news030.html

また、同じテーマで別のところで話すこともあるそうですので、ぜひ足を運んでみてはいかがでしょうか。SD

Hack For Japan
エンジニアだからこそできる復興への一歩

Hack For Japan

第17回 International Open Data Hackathon Tokyo報告

“東日本大震災に対し、自分たちの開発スキルを役立てたい”というエンジニアの声をもとに発足された「Hack For Japan」。本コミュニティによるアイデアソンやハッカソンといった活動で集められたIT業界の有志たちによる知恵の数々を紹介します。

●Hack For Japanスタッフ
鎌田 篤慎 KAMATA Shigenori
Twitter @4niruddha

東日本大震災に対してITからの復旧復興支援を目指したボランティア団体Hack For Japanでは、震災から2年が経ち、引き続き社会の課題解決や次なる震災に備えた活動を継続しています。今回は本連載でもたびたび紹介してきたオープンデータに関して、去る2013年2月23日に実施され、筆者も参加した「International Open Data Day」に連動したハッカソン東京会場の様子を交えて、これからのオープンデータとその意義についてご紹介したいと思います。

オープンデータとInternational Open Data Day

International Open Data Day[注1]とは、産官学一体となってオープンデータ政策を盛り上げ、普及させていくためのイベントです。政府、地方自治体、公共機関などが、それぞれが持つデータの公開を進めているなか、それにかかわる政策や活動の促進を、市民や企業の側から支えることを目的としています。今回のイベントは「2013年2月23日」という日付で、世界34ヵ国、100都市以上で開催され、日本では「Open Knowledge Foundation Japan（OKFJ）」が運営の中心となり、青森、会津若松、千葉、東京、横浜、名古屋、鯖江、福岡の8都市の会場で大いに盛り上がりました。

そもそも、なぜ今このような動き、そしてオープンデータが求められているのでしょうか。オープンデータは、インターネットの双方向性を活かし、行政が持つ情報の発信と行政への市民参加を目的としたオープンガバメントの流れを汲んでいます。政府や地方自治体、公共機関が持つ大量のデータをマシンリーダブルな状態でインターネット上で公開し、そのデータは自由に再利用や再配布をすることが許されているというもので、近年、アメリカをはじめとする世界各国で活動が盛んになっています。また、その対象となるデータの内容も多岐にわたり、政府が公開する人口統計や意識調査、経済指標、その他の白書、地方自治体などではその地域の交通事情や犯罪件数、生活情報などさまざまです。

このようなオープンデータは我々のような開発者がインターネット上のさまざまなプラットフォームとマッシュアップすることで、さらなる価値を生み出します。しかし震災当時を振り返ると、Hack For Japanに参加したエンジニアの多くは、政府から公開されるデータがことごとくPDFなどのマシンリーダブルではないフォーマットであることに頭を悩まされていました。

エンジニア達が被災地の復旧、復興の支援にかける想いがある中、自らの力が発揮しづらかった状況を少しでも改善したく、今Hack For Japanではオープンデータの活動を積極的に支援しています。

それではここからInternational Open Data Hackathon Tokyoの様子を紹介していきましょう。

ハッカソン東京会場の様子

VOYAGE GROUPの会議室を借りた東京会場では、主催のHack For Japanからオーガナイザーとして参加した関 治之さん（@hal_sk）からのInternational Open Data Dayの説明と共に、今回ハッカ

注1）http://opendataday.org/

ソンで利用可能なオープンデータのリスト[注2]の内容が紹介されました。また、東京会場を後援してくださり、都内の自治体としてはオープンデータに積極的な千代田区が公開するデータもその中にあります（後述）。

東京以外の会場ではエンジニア以外の参加者が多かったこともあり、オープンデータの啓蒙活動に近いところが多かったのですが、東京会場ではエンジニアが多数集まったことで、純粋に開発を行うハッカソンとなりました。それでもエンジニアのほかに、経済産業省や千代田区の公務員の方々、山と渓谷社やITmediaといったメディア系の参加者、日本財団やピースボートといった社会貢献系の参加者などバラエティに富んだ参加者構成になっていたことから、オープンデータに対する注目度の高さがうかがえました。

参加者の自己紹介の後、千代田区職員の印出井一美さんによる千代田区が公開するデータの紹介がありました。公開されているデータは千代田区民に関するデータ、千代田区の経済データ、千代田区に存在する企業、労働者に関するデータなど多様な統計データです。加えて、ランナーに人気のある皇居周辺の交通量や事故が起きている危険個所のデータなどもありました。ランナーと歩行者、自転車などとの接触事故が多発している都心の中でも、数少ない自然空間である皇居周辺がオーバーユースになってしまっている現状への対策案として、この公開データを元にした皇居周辺の空間デザインを今回のハッカソン東京会場のテーマの1つとして提案くださいました。

また、基本的に政府や地方自治体が公開している情報はPDFであることが一般的です。しかし昨今のオープンデータの流れから、印出井さんは少しでも扱いやすいデータとして公開しようと、Excelデータによる千代田区の情報公開を押し進められています。まだ手探りの段階で、将来的なビジョンまで含めたかたちでの公開には至っていないというお言葉でしたが、今回はこの公開されたExcelデータを中心としたハッカソンを行ったことから、より成果を生み出しやすいデータのあり方を検討できたと思います。

◆写真1　ハッカソンの様子

このほかのテーマについては、事前にFacebook上で実施されたアイデアソンで検討されたアイデアと、当日会場に持ち込まれたアイデアの発表を行い、それらのアイデアに賛同するメンバーを募ってチーム分けを実施しました。このチーム分けにより、参加者は4チームに分かれました。

エンジニア以外の人が参加しやすかったオープンデータ「アイデアソン」チーム、「LocalWiki＆ランニングMAP」チーム、「災害発生後ダッシュボード」チーム。そしてもっとも賛同者が集まった「千代田区可視化」チームは、人数が多すぎることもあり、目的に合わせてデータを作る「Excelデータ位置情報化」チーム、データをGoogle Earth上にプロットする「Google Earth」チーム、データを可視化する「Data Rabbit」チームの3チームに分け、計6チームとなりました。

それぞれのチームの成果

さっそく各チームとも全員で協調して議論、開発に取りかかりました。各々の得意な領域の知識を活かしてオープンデータの利活用に取り組む姿勢は皆さん同じでした。会場にはお菓子や飲み物も用意され、お昼にはおにぎりも配られ、皆さん集中はしつつもリラックスした雰囲気で開発作業は進みました（**写真1**）。

それでは各チームの成果物をご紹介しましょう。

注2）https://www.facebook.com/groups/tokyo.opendataday/doc/451083498279204/

オープンデータアイデアソンチーム

このチームは開発者の方が少なかったこともあり、オープンデータに関する議論を行い、その成果をオープンデータアイデアボックス[注3]に提案することをゴールとしたチームでした。

オープンデータアイデアボックスとは、有識者や産業界、行政機関などからの意見だけではなく、国民からの声もオープンデータに反映させようとする試みです。このチームはそもそもオープンデータとは？というところからのメンバーが多かったため、オープンデータがどういうものか、今の課題は何なのかを有識者に共有してもらい、ゼロベースでオープンデータで何ができるか、何が求められているかを議論しました。

結論としては、(1) 行政はマシンリーダブルなデータを公開する (2) 開発者は開発したサービスの利便性を常に考える (3) 市民は困っていることの発信を行い知ってもらう、といった三者の活動により、オープンデータで築くより良い社会を目指すところに落ち着き、無事にアイデアもオープンデータアイデアボックスに投稿されました[注4]。

LocalWiki＆ランニングMAPチーム

このチームは記事と地図をリンクするWikipediaのようなサービスであるLocal Wikiを利用しています。エンジニアではないメンバーは、千代田区が公開するデータを元にLocal Wikiに千代田区地名由来のデータを入力。エンジニアメンバーはそのデータを用いてアプリを開発する、と作業を分担して行いました（**写真2**）。

◆**写真2　LocalWiki＆ランニングMAPチームの様子**

Local WikiでGoogle Mapsと連携して皇居周辺の地図とデータのマッチングを行い、皇居周辺のランニングコースや危険個所のデータが取り出せるようになりました[注5]。

災害発生後ダッシュボードチーム

このチームは災害発生時に災害復旧担当者を支援する目的の、ダッシュボードを開発するチームです（**図1**）。火災の発生現場や避難所の受け入れ状況、備蓄食料・資材の在処（ありか）などをソーシャルメディアに流れる情報などと連携し、Googleスプレッドシートからwebサイトに公開するしくみの開発にあたりました。

首都圏直下型地震を想定したリアルな設定に備えたアプリケーションで、311で経験された内容が事前に想定されたものとなり、ソーシャルメディアを上手く活用し、被災状況の可視化を行う方法としても興味深いものがありました[注6]。

千代田区可視化；Excelデータ位置情報化チーム

このチームはExcelデータで公開されている千代田区の情報の中から、位置情報に紐づくであろうデータ、たとえば保育園や小学校などの施設を見つけて抽出します。続いて施設名で検索して住所情報を見つけ、ジオコーディングにより緯度経度情報を付与することで地図上でプロットできるようにしました。これによって、他のチームでのプログラム利用に必要なデータを作成しました。

データ作成の中で、Excelデータもマシンリーダブルというにはほど遠い状況であるのと、プログラムで利用する際にあると便利な情報が欠落している点が見えてきました。そこで、オープンデータとしての利用を意識したXML形式でのデータ公開、プログラムでの利用を意識した情報の付与の提案も行いました[注7]。

注3) http://opendata.openlabs.go.jp/
注4) http://odhd13.okfn.jp/?p=134
注5) http://odhd13.okfn.jp/?p=121
注6) http://odhd13.okfn.jp/?p=137
注7) http://odhd13.okfn.jp/?p=159

千代田区可視化；Google Earthチーム

このチームは位置情報を3Dの棒グラフで表示できるようにすることを目指し、JSONからkmlを生成する変換プログラムを作成しました。ピンをある位置に表示し、長さを変えるという可視化を行いました。

上手くデータの可視化に成功し、またExcelデータ位置情報化チームにジオコーダーの提供などを行い、チーム間連携にも貢献してくれました[注8]。

千代田区可視化；Data Rabbitチーム

このチームはさまざまな種類の位置情報を重ねて表示することで、地域のさまざまな問題について横断的に見ることができるアプリを目指しました。ゼロからプロトタイプまでのアウトプットを目指すという開発に力を入れたエンジニア中心のチームで、HTML + JavaScript（Leaflet.jsとjQueryも利用）によって完全にゼロから作りました。

今回の開発でスキルの異なるメンバー間でペアプログラミングを実施し、スキルアップにもつながったことから、オープンデータを扱うハッカソンはプログラムの学習にも効果的であると発表してくれました[注9]。

終わりに

今回のハッカソンを通じて、オープンデータの可能性を各会場の参加者にも大いに伝えることができました。しかし、発表の中にもあったとおり、オープンデータ自体には公開形式の問題や、数が少ない問題などまだまだ課題が山積みです。読者の皆さんにも今後のオープンデータに注目し活用いただくことで、解決の方向が見えてくるでしょうし、普及にさらにはずみがつくと思われます。ぜひ今後の展開にもご注目ください！ SD

注8）http://odhd13.okfn.jp/?p=165
注9）http://odhd13.okfn.jp/?p=160

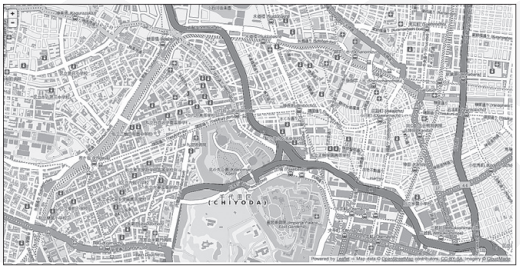
◆図1　災害状況確認ダッシュボードのイメージ図

◆図2　公開された千代田区の保育園・学童施設などのLOD※

※　LOD（Linked Open Data）とは、個々のデータを意味を持たせたリンクでつないでデータ全体を表現する、機械処理しやすい形式のこと。

◆図3　千代田区可視化；Google Earthチームの成果

◆図4　Data Rabbitチームのプロトタイプ



温故知新
IT むかしばなし

第22回

コンピュータ博物館

北山 貴広　KITAYAMA Takahiro　Twitter：@kitayama_t　kitayamat@gmail.com

はじめに

今回は、コンピュータに関連する歴史的な資料を集めて整理／展示したり、資料の調査／研究をしている博物館についての話です。

世界のコンピュータ博物館

Wikipedia 英語版の“Computer Museum”[注1]では、大きくはオンライン、北米、ヨーロッパ、中東、オセアニアと5つのカテゴリで博物館への URL リンクが集められています。残念ながら、地域カテゴリの中に日本を含むアジア地域はなく、日本に関するリンクは、オンラインのカテゴリの中に情報処理学会（IPSJ：Information Processing Society of Japan）があるのみです。

地域カテゴリに含まれるリンク数は、北米とヨーロッパへのリンクが13個と14個と大半を占めていて、中東とオセアニアはそれぞれ1個ずつにとどまっています。北米とヨーロッパは、コンピュータの歴史的な発展に中心的に関わり、コンピュータが博物館として認められている分野と認識されているのだと感じます。

Computer History Museum

1996年に設立された Computer History Museum[注2]（コンピュータ歴史博物館）は、米国カリフォルニア州マウンテンビューにあり、世界最大級の歴史的なコンピュータが展示されています。この博物館に所蔵されている約30台のコンピュータを独特の視点で撮影した図鑑「Core Memory －ヴィンテージコンピュータの美」[注3]がオライリー・ジャパンから発売されています。歴史的な ENIAC や教科書によく出てくる IBM System /360、またパソコンマニアとしては貴重な Apple I、そしてインターネットの歴史として Google の最初の運用サーバなどの写真が載っています。

この博物館にしかない貴重なコンピュータがあったり、実際に動作させているところを見学することができるようです。Web サイトには1日は時間をとって見に来てくださいとあります。

日本のコンピュータ博物館

日本国内には残念ながらコンピュータ専門の博物館はなく、上野にある国立科学博物館の科学技術の一部としてコンピュータが含まれているのみです。そのため、情報処理学会がコンピュータに関する歴史的な資料を「コンピュータ博物館」[注4]として Web 上にまとめて公開しています。先ほどの Computer Museum の中のオンライン博物館に含まれているのが、このコンピュータ博物館の英語版へのリンクです。

また、日本国内で歴史的なコンピュータを保存／展示している機関（企業や大学や研究所など）を情報処理学会が「分散コンピュータ博物館」[注5]として認定し、Web 上でリストを公開しています。

東京理科大学 近代科学資料館

情報処理学会の「分散コンピュータ博物館」に認定された中で、「計

注1）http://en.wikipedia.org/wiki/Computer_museum/

注2）http://www.computerhistory.org/

注3）http://www.oreilly.co.jp/books/9784873113579/

注4）http://museum.ipsj.or.jp/index.html

注5）http://museum.ipsj.or.jp/satellite/index.html

算機の歴史」という常設展でコンピュータに限らず計算という観点から幅広い分野で、数多くの資料が常に展示／公開されているのが東京理科大学の近代科学資料館[注6]です。東京の飯田橋駅近くと場所的にも訪れやすく、予約不要で公開日も多いのが特徴です。「日本一の計算機コレクション」と呼ぶにふさわしい圧倒的な展示物の豊富さで「コンピュータ博物館」と名前を付けたほうがよいのではと思いました。

もっとも豊富なコレクションは機械式計算機で、世界各国のものや日本の各メーカーのものがきれいにガラスケースに収められています。そのガラスケースが各部屋にいくつもずらっと並べられて展示されていて、計算尺やそろばん、またカシオミニなどの電卓もひととおりそろってます。パソコンは著名なAltair 8800、AppleII/TRS-80/PET-2001の米国8ビットパソコン御三家（**写真1**）や、日本の8ビット御三家、NEC PC-100やMSX、Apple IIcやMac、NEXT Cubeもありました。また、エプソンハンドヘルドコンピュータHC-20やHP 100LX/200LX、ゲーム機の展示など、身近に使っていたものも含めて幅広い分野で展示されていて、とても興味深いです。タイガー計算機を実際に触って実感することができます（**写真2**）。

大型の貴重な展示物は、情報処理技術遺産[注7]に認定されているパラメトロン計算機「FACOM201」[注8]と、「Bush式アナログ微分解析機」[注9]があります。つい最近、国内最初の真空管商用計算機「UNIVAC120」が加わりました。

近代科学資料館の展示物を中心に解説している書籍『実物でたどるコンピュータの歴史〜石ころからリンゴへ〜』[注10]が、筆者が訪問したときには現地で少しお得な価格で発売されていました。

理化学研究所

毎年4月の第3週は科学技術週間で、国内の科学技術系の施設が一般公開されています。筆者は昨年（2012年）4月の科学技術週間の一般公開日に和光市の理化学研究所を訪れました。

一番の目的である超伝導リングサイクロトロンの見学を終えたあと、コンピュータ（パソコン）の歴史展示に偶然出会いました。電子拡大鏡で4004マイクロプロセッサ回路基板を映して4004の刻印を見たり、磁気テープやパンチカード、理研電子計算室（現情報基盤センター）で1975年に制作した手作りパソコンの実機（Altair 8800と同等のスペック）が展示されていました。

終わりに

日本国内で実際に訪れることができる博物館の中から、今回分散コンピュータ博物館に認定されている東京理科大学を訪ねてみて、立派な展示に驚き、あらためて実物を見ることのよさを感じました。みなさんもぜひ近くの施設を尋ねられてみてはいかがでしょうか？

また、分散コンピュータ博物館に認定されている京都コンピュータ学院のコンピュータミュージアムのWebサイト[注11]で、DEC社製ミニコンを千葉から京都に移設するための寄付を募っていました。国レベルの施設としてコンピュータ博物館を運営して、こういった歴史的価値を残して後世に伝えてほしいと強く思いました。SD

注6）http://www.tus.ac.jp/info/setubi/museum/

注7）http://museum.ipsj.or.jp/heritage/index.html

注8）http://www.tus.ac.jp/info/setubi/museum/db/museum-yama/si/facom.html

注9）http://www.tus.ac.jp/info/setubi/museum/db/museum-yama/si/bibun.html

注10）http://www.tus.ac.jp/news/news.php?20120831134511

注11）http://www.kcg.ac.jp/museum/computer/index.html

▼写真1　米国8ビット御三家+Altair 8800など

▼写真2　実際に触れるタイガー計算機

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

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

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

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

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

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

Software

グレープシティ、Java開発コンポーネントセットの総合パッケージ「JClass DesktopViews 6.5［英語版］」を発売

グレープシティ㈱は、Javaデスクトップアプリケーション開発に適したコンポーネント群とJARファイル生成ツールをセットにした「JClass DesktopViews 6.5［英語版］」を3月29日に発売した。1開発ライセンス価格は462,000円(税込)。開発したアプリケーションの配布は無料で行える。

同製品は、データグリッドやチャート、レポート、各種UI部品などJava SEアプリケーション開発に必要となるコンポーネントとJAR生成ツールをセットにしたスイート製品。アプレット、スタンドアロン、Swing、SWTなど対応するフレームワークも幅広く、さまざまな分野のJavaシステムを開発できる。英語版でありながら、日本語環境での動作保証および日本語技術サポートも行っている。

新バージョンの6.5では、WindowsおよびLinux向けのSWTチャートコンポーネントを新たに収録。基本グラフに加え、ガントチャートやバブルチャートなどの拡張グラフを提供。軸や目盛、凡例などの各要素のカスタマイズも柔軟に設定できるようになった。

CONTACT グレープシティ㈱
URL http://www.grapecity.com/jp

Service

シーズホスティングサービス、プレミアムモデルのスペックを改定

シーズホスティングサービスは、ユーザからの「もっとハイスペックなサーバを使いたい」との要望に応えるため、「専用サーバーサービス」と「Windowsサーバーサービス」のプレミアムモデルについてスペックの改定を実施。3月4日より新スペックでのサービスを提供開始した。

今回、サービスが開始されたプレミアムモデルでは、6コアCPUを採用し、メモリも32GBに増設した。値段はそのままの据え置きで、高性能サーバを気軽に利用できるようになった。詳細なスペックは右のとおり。

■プレミアムモデルのスペック
サーバ：DELL PowerEdge R320
CPU：Xeon E5-2430L (2.0GHz/6Core/12Thread)
メモリ：32GB
HDD構成：300GB (SAS 15000RPM) ×2　RAID1
HDD容量：約300GB
バックアップHDD：1TB (SAS 7200RPM) ×1

CONTACT シーズホスティングサービス
URL http://www.seeds.ne.jp

Hardware

エバーグリーン、スマートフォン用車載ホルダー形FMトランスミッターを発売

インターネットショップ「上海問屋」を運営する㈱エバーグリーンは、スマートフォンを車のダッシュボードなどに設置できるホルダーと、FMトランスミッターが一体化した「車載ホルダー形FMトランスミッター」を発売した。

ホルダースタンドは、縦置き／横置き両方に対応し、画面の向きも上下左右に動かせる。

iPhoneなどの各種スマートフォンを、ステレオミニプラグでFMトランスミッターに接続して音楽を再生する。周波数は76.0～108.0MHzと広く、0.1MHzステップで最適な周波数を選択できる。

シガー電源ケーブルとMicro USB充電ケーブルを使えば、スマートフォンの車内充電も可能。価格は1,499円(税込)。

▶スマートフォン用車載ホルダー形FMトランスミッター DN-82576

CONTACT 上海問屋
URL http://www.donya.jp

Hardware

NECアクセステクニカ、新規格「Draft IEEE802.11ac」に対応のWi-Fiホームルータ「Aterm」の新製品を発売

NECならびにNECアクセステクニカは、無線LAN規格の一部であるIEEE802.11の新規格「Draft 11ac」に対応したWi-Fi（無線LAN）ホームルータ「AtermWG1800HP」と「AtermWG1400HP」を4月初旬より発売する。

▼新製品のラインナップ

商品名	型番	発売時期
AtermWG1800HP	PA-WG1800HP	4月初旬
AtermWG1800HP イーサネットコンバータセット（WG1800HP の2台セット）	PA-WG1800HP/E	4月初旬
AtermWG1400HP	PA-WG1400HP	4月初旬
AtermWG1400HP イーサネットコンバータセット（WG1400HP の2台セット）	PA-WG1400HP/E	4月初旬
AtermWG1400HP USB スティックセット（WG1400HP と WL900U のセット）	PA-WG1400HP/U	5月中旬
AtermWL900U（USB スティック子機）	PA-WL900U	5月中旬

新商品は5GHz帯にてDraft 11acに対応し、最大1300Mbps（AtermWG1400HPは最大867Mbps）の高速Wi-Fi通信を実現。2.4GHz帯でも11nで最大450Mbpsの高速Wi-Fi通信を実現している。

また、NEC独自の電磁ノイズ抑制技術「μ（マイクロ）EBG構造」を製品に適用した。これにより電気回路における電磁ノイズを大幅に削減し、Draft 11acの対応と合わせて、高速なWi-Fi通信を可能にした。従来時間のかかっていた高画質の動画閲覧などを快適に楽しむことができる。

販売価格はいずれもオープン価格となっている。

▲AtermWG1800HP（左）、AtermWG1400HP（右）

CONTACT NECアクセステクニカ㈱
URL http://www.necat.co.jp

Service

NTTPCコミュニケーションズ、新たなVPSサービス「WebARENA VPSクラウド」を提供開始

㈱NTTPCコミュニケーションズは、従来のVPSサービスプラットフォームに、監視、ロードバランサ、バックアップ&クローンなどの機能を追加した新たなVPSサービス「WebARENA VPSクラウド」を3月5日より提供開始した。サービスの特徴は次のとおり。

- 従来サービスと同様最短3分でサービスを開通
- インスタンスの起動／停止、ロードバランサの設定、マシンイメージのバックアップなどをWeb上のコントロールパネルからオンデマンドで操作できる
- 複数のインスタンスを1つのコントロールパネルで一元的に管理
- インスタンスごとにファイアウォールを導入可能
- 追加料金なしで、さまざまなサービスの監視（HTTP/HTTPS/FTPなど）を利用できる
- ロードバランサを利用可能
- マシンイメージのバックアップを取得可能。バックアップしたマシンイメージをテンプレートとして新たなインスタンスを起動可能
- 最大1Gbps共有回線でアップリンクした環境を提供

提供料金は基本サービスプランの場合、初期料金は無料で、月額料金が3,780円～52,920円。5月13日までの期間限定で基本サービスプランの月額料金が最大3ヵ月無料になるキャンペーンを実施している。

■基本スペック

①インスタンス
Sタイプ：ディスク40GB／メモリ2GB（仮想2コア）
Mタイプ：ディスク100GB／メモリ4GB（仮想4コア）
②OS：CentOS6
③ネットワーク
グローバルIPアドレス：1インスタンスあたり1つのIPアドレスを付与
アクセス回線：インターネット回線1Gbps共有
④コントロールパネル、ファイアウォール、監視などの機能は標準で利用可能

CONTACT ㈱NTTPCコミュニケーションズ
URL http://www.nttpc.co.jp

Letters from Readers

SD総集編【2001～2012】大好評発売中！

3月末に総集編を発売して以来、入手した方々がTwitterやブログでさまざまな感想や使い方を報告くださっていますね。各種タブレットでの動作確認に始まり、過去記事の内容分析や、PDFの編集方法まで……。編集部では想定もしていなかった使い方がいろいろあって驚きです。まさにSD総集編ハック！　ただいま【1990～2000】版を制作中です。こちらもどうぞご期待!

2013年3月号について、たくさんのお便りありがとうございました！

第1特集　もっとクラウドを活用してみませんか?

世にあるクラウドサービスはいずれも優れた操作性を持ち、システム運用が楽になる工夫がなされています。しかし、ITエンジニアなら、クラウドのしくみをきちんと理解したうえで利用したいものです。そこで、現在注目されているオープンソースのクラウド環境であるOpenStack、CloudStack、CloudFoundry、Scalr、Eucalyptusを紹介しました。

PaaSというと「インターネットの雲の向こうでなんだか高度ものが動いている」という印象だったのですが、仮想マシンで自分でも動かせると知ってびっくりしました。自前でPaaSを立ち上げてみると、いろいろできておもしろそうです。

埼玉県／猫棟梁さん

話題のOSSクラウド環境をざっくり比較できて良い。

神奈川県／中山さん

OpenStackしか知らなかったので、ほかのものにも興味がわきました。

東京都／真野さん

CloudStackやScalrはほとんど知らなかった。

東京都／荒木さん

クラウド環境を構築できるOSSがいろいろと出そろってきました。今後は与えられたクラウド環境をただ使うだけでなく、自ら構築してしくみを理解したり、検証環境として使ったりということができそうです。

第2特集　実践!ワイヤリングの教科書

ある統計によるとネットワーク障害の原因の7割は配線にある、と言われます。本来は専門の技術者がやるべき重要な作業ですが、社内LANなどは自社のエンジニアが見よう見まねで配線しているという例が多いのではないでしょうか。そこでそういう場合に役立つ配線のノウハウや注意点について、プロの視点から解説を行いました。

その昔、黄色の太いケーブルにトランシーバーを接続していたのが懐かしい。

東京都／エリックさん

ケーブルの作成はよくやったよ!　無駄になった長さのほうが長かったりして ^^;

埼玉県／南雲さん

むきだしのLANケーブルの写真がインパクト大でした。

千葉県／Tayuさん

「UTPケーブルは平行に配置してはいけない、スパゲッティ配線にすること」というのは、意外だったのではないでしょうか。なんでもきれいに配線するのが良いというのは、素人の勝手な思い込みなのだ、と思い知らされた一例でした。

一般記事　SSDストレージ爆発的普及の理由

以前に比べて価格も下がってきたことで、ストレージのボトルネックを解消するための手段としてSSDが導入され始めてきました。そのSSDの価格、性能、市場動向について、HDDと比較しつつ解説しました。

急速に容量単価が下がっており、HDDはまもなく抜かれると思っていた。

岩手県／ハヤブサさん

やはりMacBook Airの貢献度合いが大きい気がします。

東京都／husuiさん

SSDはエンタープライズ領域だけでなく、コンシューマ向け製品にも取り入れられてきていますので、いずれはSSDがHDDにとってかわる、と実感している読者も多いようですね。

一般記事
社内LAN撲滅運動!

サーバーワークス社は社内で利用するサーバのほとんどをAWSでまかない、新規SI案件はすべてAWSで実装するという方針で事業を進めています。その先進的な取り組みを紹介しました。

このような記事こそ、SDをIT系雑誌を越えたマネジメントの雑誌に格上げしていると思う。

長崎県／奇抜三十朗さん

逆に機械が手元になくて、若い人に教えられなくて困っている。

埼玉県／ぜーたさん

過激だけどありかも。

愛知県／kmさん

確かにこの事例を同じように実践できれば、サーバを調達／管理する手間、時間、人員はずいぶん削減されるでしょう。一方で、実機を使わないとなると、ぜーたさんのご指摘のように、ハードウェアまわりの技術の継承が課題となりそうですね。

連載

「はんだづけカフェなう」について単行本にならないものかと待っています。ぜひよろしくお願いします。

奈良県／本が増えるとなだれがおきるさん

ありがたい提案です。書籍化するには内容はもちろんのこと、市場規模や類書の売上状況など、さまざまな観点から検討しています。このようなご意見も判断材料の1つになります。今後ともご意見があれば、どしどしお送りください。

「iPhone OSアプリ開発者の知恵袋」の藤田武男さんの「少ないダウンロード数で大きく稼ぐアプリ開発術」に触発されました。が、手元にMacがないので、まずはAndroidアプリに挑戦しようと思っています。

大阪府／匿名さん

「iPhone OSアプリ開発者の知恵袋」は4月号で終了となりましたが、同じiPhoneの連載として、今月号より新たに「プログラム知識ゼロからはじめるiPhoneブックアプリ開発」が始まりました。いかがでしたでしょうか? こちらのご意見もお待ちしております。

エンジニアの能率を高める一品

仕事の能率を高める道具は、ソフトウェアやスマートフォンのようなデジタルなものだけではありません。
このコーナーではエンジニアの能率アップ心をくすぐるアナログな文具、雑貨、小物を紹介していきます。

ピコットフセン

420円（税込）／カンミ堂　http://www.kanmido.co.jp

※読者プレゼントあります。16ページ参照。

会議の要点はノートに手書きでメモし、白板に書かれた議論の内容はスマホのカメラで撮影して残す。そんなとき、ピコットフセンを使えば、手書きメモと写真を結び付けて管理できます。ピコットフセンはQRコードが印刷された付箋です。そのコードをスマホの専用アプリで読み取り、表示された写真選択画面でスマホ内の写真からリンクしたい写真を選択します。すると、次にそのQRコードを読み取ったときには、リンクした写真のみが即座に表示されます。ノートの手書きメモのそばに、白板の写真とリンクした付箋を貼っておけば、メモを見ながらすぐに写真も閲覧できます。スマホ内に写真が多いほど目的の写真を探すのも手間ですが、ピコットフセンを使えば簡単に見たい写真が見られ、便利さを実感します。3月末現在、iOS 6.0を搭載したiPhone 4/4s/5のみの対応ですが、4月末にはAndroid端末にも対応するようです。

▲QRコード読み取り画面

祝 3月号のプレゼント当選者は、次の皆さまです

①バックライト付キーボードSKB-WAR2 ..... 大阪府　河合勇輝様
②ストレッチングレーザーマウスDN-46382 ..... 東京都　山下佳寿様

★その他のプレゼントの当選者の発表は、発送をもって代えさせていただきます。ご了承ください。

次号予告

Software Design
June 2013

2013年6月号
定価1,280円　176ページ

5月18日発売

［第1特集］新しく「基礎」を学んで、プロ技を身につける

ちゃんとオブジェクト指向できていますか?

プログラミング言語の多くがオブジェクト指向に対応し、たとえスクリプト言語であってもオブジェクト指向を使いこなすことが重要なスキルになっています。しかし、いまだに苦手な方も多かったりします。オブジェクト指向に対する誤解もあるでしょうし、正しく習っていない場合もあります。

本特集ではオブジェクト指向を多面的に解説しマスターする手がかりを紹介します。

［第2特集］サーバ／ネットワーク管理

あなたの知らないUNIXコマンドの使い方

［一般記事］

- セミナー講師は見た「春の新人あるある物語」
- しくみを知りたい、学びたい「Stormではじめる分散処理」

休載のお知らせ

「システムで必要なことはすべてUNIXから学んだ」（第10回）は都合によりお休みいたします。

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。

■2013年4月号　連載「IPv6化の道も一歩から」

●P.141　図1
［誤］①近隣隣要請
［正］①近隣要請

●P.142　左段5行目
［誤］Oflag=1
［正］O flag=1

SD Staff Room

●東急東横線が副都心線と接続された。通勤ルートをいろいろ試せるようになったので、ためしに新宿三丁目経由で都営新宿線を使ってみたところ5分も時短!　しかし一度改札を抜けないとダメなので定期券の範囲内でなくて残念だった。新宿で買いものが楽になるけど、うっかり寝過ごすと小手指だよ!（本）

●会社から歩いて行ける範囲内に、メゾンカイザーやPaulがあり、ハード系パンはよく買っているのだが、最近は遠征してVIRONやル・プチメックなどを物色している。今一番はVIRONのバゲットレトロドール。小麦の香りが素晴らしい。エレシがないので、カルピスバターで。休日ブランチの幸せ。（幕）

●家族でボーリングに行きました。午前中が安い!ということで9時半に到着（8時半OPEN）。が、なんと1時間半待ち!?　家族連れや団体さんはまあ納得できますが、大学生らしき姿もちらほら。きみたち早起きして健康的だな……。気軽に楽しめる屋内スポーツとしての人気は健在ってことですかね。（キ）

●某テレビ番組でおならの臭いをおさえるには、ヨーグルト300gとサツマイモ100gを毎日食べると良い、と紹介していました。ヨーグルト300gは結構な量ですしサツマイモと一緒に食べるなんて……実践は無理だな、と思いました。いや、べつに自分のおならのことを気にしているわけではありませんよ。（よし）

●沖縄の代表料理チャンプルー。これにはコーレグース（島唐辛子）が欠かせない。私も必ずかけるがかけるときの分量調整が厄介だ。単なる島唐辛子の泡盛漬けなだけに液状はサラッサラ。それでもって容器の口の穴が大抵は大きすぎる。気がつくとドボドボ……メーカさん、穴の形状考えて～（中島んちゅ）

●スノボシーズンを終了したので次はお花見と思っていたら、今年は開花がかなり早まってしまい4月になる前にすでに満開。さすがに週末までもちそうもないので平日に夜桜見物へ。自宅や職場近くのお堀沿いもちょうど満開で見ごたえがあるが、きちんとライトアップされ水面に映る夜桜は幻想的で格別!（ま）

ご案内
編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［FAX］
03-3513-6173
※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

［E-mail］
sd@gihyo.co.jp

Software Design
2013年5月号

発行日
2013年5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
＊
細谷謙吾（書籍編集長）
取口敏憲

●編集アシスタント
松本涼子

●広告
中島亮太
北川香織

●発行所
㈱技術評論社
編集部
TEL：03-3513-6170
販売促進部
TEL：03-3513-6150
広告企画部
TEL：03-3513-6165

●印刷
図書印刷㈱

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

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