Special Feature 01
3Dプリンタ
Special Feature 02
バグを狙い撃つ技術
2013 August 08
[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8月18日発行
毎月1回18日発行
通巻340号
（発刊274号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
1,280円

理論から実践まで

3Dプリンタが未来を拓く理由

Special Feature 01

Makerbot
Replicator
3D Systems, Inc
RepRap
Ultimaker
Stratasys
Polyvinyl Alcohol
3D-CAD
Overhang
Numerical Control
Skeinforge+ReplicatorG
Kisslicer
Cura

3D Printer
Makes New Vision.

Special Feature 02
» ソフトウェアのバグを見抜く「システムを見通す力」の身に付け方

好評連載
» 「再発見の発想法」 結城浩
「enchant—創造力を刺激する魔法」 清水亮
「仮想ネットワークの落とし穴」 伊勢幸一
「Riak分散データベース未来工房」 上西康太
「自宅ラックのススメ」 tomocha
「プログラム知識ゼロからはじめるiPhoneブックアプリ開発」 GimmiQ

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56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。

次世代 Web ブラウザエンジン

揺れる Web ブラウザ市場

HTML5/CSS3 を中心とする Web 技術の急速な進化を支える最も重要な要素は、Web ブラウザに搭載されている Web レンダリングエンジンです。ここ数年は Apple 社と Google 社が中心となって開発し、Safari や Chrome（および Chromium）に搭載されている「WebKit」、Mozilla Foundation が開発し、Firefox に搭載されている「Gecko」、そして、Microsoft 社が開発し、Internet Explorer に搭載されている「Trident」が三大勢力となっていました。

とくに WebKit については 2005 年にオープンソース化されて以来、急速にそのシェアを拡大させており、現在 WebKit ベースの Web ブラウザが約半分のシェアを占めています。この急成長の背景には、最新 Web 技術への対応が極めて早いという技術面・機能面での強みに加えて、モバイル端末市場における iOS（標準ブラウザとして Safari を採用）と Android（同、Chrome を採用）の圧倒的なシェアの高さがあります。

2013 年 2 月には、Opera 社も Web ブラウザ「Opera」のレンダリングエンジンを自社開発の「Presto」から「WebKit」へ切り替えることを発表し、WebKit のシェアはますます盤石なものになると思われました。ところが 4 月に入って、この勢力図に大きく影響するであろう 2 つの衝撃的なニュースが伝えられました。1 つ目は、Mozilla が Samsung と共同で Gecko に変わる新エンジン「Servo」の開発を進めていくと表明したこと。そして 2 つ目は、Google も Chrome 向けに新エンジン「Blink」を開発し、WebKit からの移行を表明したことです。

Google 製新エンジン「Blink」

「Blink」は WebKit プロジェクトからフォークする形で開発され、コードベースの健全化と、マルチプロセスアーキテクチャへの最適化が行われる点が大きな特徴です。Blink が正式に採用されるのは Chrome 28 からで、間もなく正式版がリリースされる予定（本稿執筆時点）となっています。ただし、当面の間は内部的なアーキテクチャの改善とコードベースの簡素化が作業の中心になるため、Web 開発者に大きな影響を与えることはないとの話です。なお、WebKit の採用を決めたばかりの Opera でも Blink への移行が進められています。

Google が WebKit を捨てて独自のエンジン開発に方針を変えた理由としては、Chrome が他の Web ブラウザと違ってマルチプロセスアーキテクチャを採用していることが挙げられています。WebKit の開発は Chrome への対応のために煩雑化しており、逆に Chrome 側でも WebKit との整合性を保つために本来行いたい技術改良が阻害されるケースが増えていることから、袂（たもと）を分かつ苦渋の決断に至ったとのことです。

Mozilla 製新エンジン「Servo」

「Servo」はプログラミング言語 Rust をベースとして開発される新しいエンジンで、特徴的な機能としてはマルチコアやヘテロジニアスプロセッサへの対応、GPU による並列処理の活用、メモリ管理の大幅な改善によるセキュリティ強化などが挙げられます。Rust は Mozilla が独自に開発を進めているオープンソースの言語であり、ハードウェアリソースの厳格な制御機構やメモリセーフ機能、マルチコア CPU に最適化された設計などが特徴であり、Servo で要求されるひととおりの要素を備えています。

Servo は当初は Android + ARM というモバイル環境をターゲットに開発が進められる計画となっており、Gecko が完全に Servo に置き換えられるというわけではありません。Servo の位置づけは「新しいハードウェアに対応するためのまったく新しいエンジン」であり、PC 環境向けでは依然として Gecko がその中心的役割を担っていくことになります。ただし、将来的には Servo の成果が PC 向けブラウザにフィードバックされることも考えられます。

Blink と Servo という 2 つの新しいエンジンの登場は、Web ブラウザの新しい可能性を拡げる一方で、プラットフォームのさらなる多様化という問題を含んでいます。その影響を最初に受けることになる Web 開発者やデザイナは、各社の動向にとくに気を配る必要があるでしょう。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 Aug. 2013

1

第1特集 Special Feature 01

理論から実践まで

3Dプリンタが未来を拓く理由

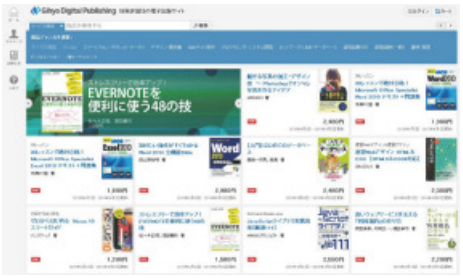
技術評論社の本が電子版で読める！

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。

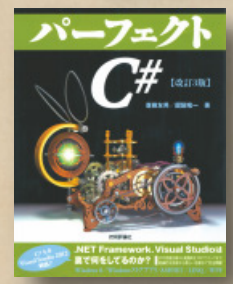
お問い合わせ
〒162-0846
新宿区市谷左内町21-13
株式会社技術評論社　クロスメディア事業部
TEL：03-3513-6180
メール：gdp@gihyo.co.jp

Contents

Software Design Aug. 2013

言語のセオリーを徹底解説

パーフェクトシリーズ



[改訂3版] パーフェクトC#

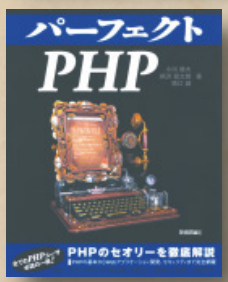
斎藤友男、醍醐竜一　著／B5変形判／608ページ　定価3,780円（本体3,600円）
ISBN 978-4-7741-5680-4

C#で.NET開発を行う人へのバイブル的1冊です。概要／基礎から実践までを幅広く学習でき、C#を扱ううえで知っておきたい知識は、この一冊に網羅されています。基本文法，Webアプリケーション開発はもちろん、C#5.0から可能になったWindowsストアアプリケーション開発の実践方法から、C#、.NETの内部動作まで幅広いテーマをあつかっており、この一冊でC#の知識は完璧といえる内容をめざします。C#5.0に対応。

パーフェクトJava

井上誠一郎、永井雅人、松山智大　著／B5変形判／640ページ　定価3,780円（本体3,600円）
ISBN 978-4-7741-3990-6

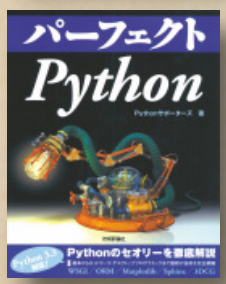
本書はJavaで開発を行う人へのバイブル的1冊です。Javaの基本から説明していますが、プログラミング一般の考え方や技法まで解説しています。なぜそれらが必要なのかを説明しながら、様々な技法やパターンについて、考え方や背景を含め理解できることを目的とした書籍です。本書の構成は3つのPartに分かれています。Part1でJava言語の基礎を、Part2はサーバプログラミング、Part3はJava GUIと代表的な応用分野の解説をしています。Part2とPart3は実践的な解説に力点を置いています。



パーフェクトPHP

小川雄大、柄沢聡太郎、橋口誠　著／B5変形判／592ページ　定価3,780円（本体3,600円）
ISBN 978-4-7741-4437-5

1冊で言語仕様から最新の技術までを網羅した内容。網羅的に解説されているだけでなく、各技術に関しては基本からしっかり解説し、フレームワークなどを利用したWebアプリケーション開発の解説などは、内部処理が裏で何をしているのかを掘り下げて解説してあるため、PHPを体系的に学びたい方はもちろん、より深い知識を得たい中～上級者にもお勧めの一冊です。



パーフェクトPython

Pythonサポーターズ　著／B5変形判／464ページ　定価3,360円（本体3,200円）
ISBN 978-4-7741-5539-5

1冊で言語仕様から最新の技術までを網羅した内容です。網羅的に解説されているだけでなく、各技術に関しては基本からしっかり解説し、必要な箇所では、内部処理が裏で何をしているのかを掘り下げて解説してあるため、体系的に知りたい初心者はもちろん、中級者にもお勧めの一冊です。最新のPython3.3に対応。

パーフェクトJavaScript

井上誠一郎、土江拓郎、浜辺将太　著／B5変形判／544ページ　定価3,360円（本体3,200円）
ISBN 978-4-7741-4813-7

1冊で言語仕様から最新の技術までを網羅した内容です。本書はJavaScriptで本格的なWebアプリケーションを作りたい人を対象に、前半でJavaScriptの言語仕様を掘り下げて解説し、後半で今求められるJavaScriptの応用分野として、クライアントサイドJavaScript、HTML5、Web APIの利用、サーバサイドJavaScriptの解説を丁寧に行っています。

8月上旬発売予定

パーフェクトRuby

Rubyサポーターズ　著／B5変形判
640ページ　定価3,360円（本体3,200円）

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

ソフトウェア・デザイン［目次］

Contents

Software Design Aug. 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	Mint Images - Frans Lanting/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。

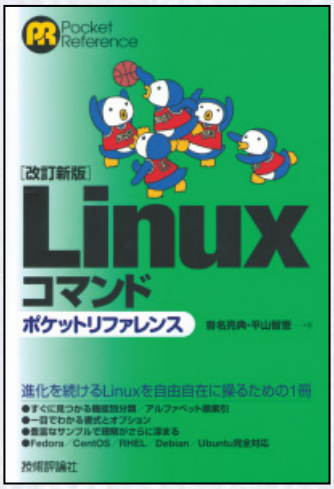
この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

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ーッと凝縮！
関連項目への参照ページもあって、
検索性もバツグン！



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

高橋晶ほか　著
四六判／528 ページ
定価 3,024 円（2,880 円+税）
ISBN 978-4-7741-5715-3

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

土井毅、高江賢、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

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

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

藤本壱　著
四六判／336 ページ
定価 2,604 円（2,480 円+税）
ISBN 978-4-7741-5486-2

涌井良幸、涌井貞美　著
四六判／288 ページ
定価 1,764 円（1,680 円+税）
ISBN 978-4-7741-5513-5

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

紙面版
A4判・16頁
オールカラー

電脳会議

DENNOUKAIGI

一切無料

新規購読会員受付中!

『電脳会議』は情報の宝庫、世の中の動きに遅れるな!

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

今が旬の情報満載!

新規送付のお申し込みは…

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

電脳会議事務局　検索

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

一切無料!

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

毎号、厳選ブックガイドも付いてくる!!

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

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

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4回

ラックの電源問題（その2）

イラスト：高野涼香

はじめに

前回は電源についてお話させていただきましたが、引き続き電源について解説します。

UPS選び

自分で使うラックが、どの程度の電力供給が可能かわかったところで、UPS（無停電電源装置）を検討してみましょう。UPSは、もはやなくてはならない必需品だと考えられます。ところで、なぜUPSは必要なのでしょうか？　それは、無停止を求めているためでしょうか？　決してそうではありません。UPSの本来の目的は、サーバや機器を安全にシャットダウンするまでの時間を確保することにあります。そのほかの得られる効果として、瞬停、ノイズ、スパイク、電圧調整など入力電源に対しての保護があります。こういった電源トラブルから守ってくれるのもUPSの役目なので、ぜひ導入を検討しましょう。

UPSには、大きく分けて3種類の給電方式があります。常時商用（オフライン）給電方式、ラインインタラクティブ方式および、常時インバーター（オンライン）給電方式です。

これらの違いについて簡単に説明してみましょう。

●常時商用給電方式

……通常は交流入力をそのままスルーで出力、停電などを検知した場合、瞬時にバッテリーからの給電へ切り換える。その際に、10ms程度の瞬断が発生する。シンプルでコンパクト、設定範囲内の入力電圧の補正はしない。安価

●ラインインタラクティブ方式

……通常時は交流入力をそのままスルーで出力するが、内部に商用トランスを持ち、入力電圧に応じて昇圧／降圧を行い出力電圧の調整をする。停電時や設定範囲外の入力電圧の場合、バッテリーからの給電へ切り換える。その際には10ms程度の瞬断が発生する。比較的シンプルで低コスト、電圧補正などを行う。また、電圧変動が激しいとバッテリーの消耗が早くなる

●常時インバーター給電方式

……交流入力をいったん直流へ変換し、バッテリー充電を行いながら、バッテリーから供給される。このため、常に安定した出力が得られる。電気の変換を常に行っているため、他の給電方式に比べて電力のロスが大きい

上記の違いから、常時インバーター給電方式がとてもよく見えますが、高価なUPSですので、ノンストップなシステムなど、非常に可用性が要求される場所で使用されることが多いようです。自宅では、まずオーバースペックでしょう。非常に重要な機器でない限り、常時インバーター給電方式のUPSを使っているケースはあまり聞きません。

また、家庭やSOHO向けや安価な常時商用給電方式のUPSの場合、電圧降下などに対応できません。これらから、中間的な位置づけのラインインタラク

ティブ方式のUPSが主流のようです。よく出回っている主要メーカーの多くはラインインタラクティブ方式です。もちろん、常時インバーター給電方式のUPSも販売していますよ！

容量の検討

UPSの方式が決まったら、容量を検討してみましょう。

容量は、ラックに設置し、使用する機器の合計消費電力から適合する機器を選べば良いでしょう。

UPSの容量はVAで記載されています。VAとは、ボルトアンペア（Volt Ampere）の略で、皮相電力を表す単位です。Wikipediaによると、皮相電力とは、電圧の実効値と電流の実効値との積で、その意味は名のごとく表向き（見かけ）の電力とあります。VAから、実際に使用可能な電力を知ることができますので、事前に確認しておきましょう。仕様などを見ると、VAとW両方で記載している場合があります。

実際に使用可能な電力は下記の式で導き出せます。

VA×力率＝W
（※一般的にサーバやネットワーク機器などの力率は0.7～0.8で計算する）

これが面倒なら、15Aの回路がまるっと使えるのでしたら、そのコンセントで使用可能な最大容量（1500VA）のUPSを選ぶのも良いでしょう（消費電力が少なければバックアップ時間は長くなるため）。うっかり、3000VA（3KVA）のUPSなどを買ってしまうと、コンセントのプラグがL5-30（30A用の回路およびコンセント）が必要など、工事が必要な場合もありますので、注意が必要です[注1]。家庭の場合の多くはNEMA 5-15（写真1）という形状が多く、エアコン用など、20A使用可能な回路の場合は、NEMA 5-20（写真2）の形状の場合が多いです。アースの端子はないことが多いですけど……200Vは、NEMA 6-XX/L6-XX（Xはアンペア）となります。

個人的には、市場に多く出回っているAPC Smart UPSのラックマウントモデルを愛用しています。SUA1500RMJ2U（写真3）は、ラックマウントモデルでラインインタラクティブ方式。中古であれば1～2万程度で入手可能です。バッテリーについても、比較的多く安価に出回っているので、お勧めです。

このあたりのモデルであれば、オプションとなりますが、ネットワーク管理モジュールなどの搭載もでき、UPSの温度や負荷など、SNMPで情報を取得することや、ネットワーク経由で、バッテリー動作時には、サーバの自動シャットダウンなどを行うことができます。必要な機能、容量、価格などを考慮し、自分に合ったモデルを選んでみましょう。SD

▼写真1　NEMA 5-15コンセント形状

▼写真2　コンセント形状L5-20（左）およびNEMA 5-20（右）

▼写真3　APC SmartUPS（SUA1500RMJ2U）

注1）http://en.wikipedia.org/wiki/NEMA_connector
http://en.wikipedia.org/wiki/File:NEMA_simplified_pins.svg
http://www.americandenki.co.jp/P/P-12/index.html

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 176 >>>>>

大人も楽しめる子供向けデジタルガジェット

デジタル機器を使いこなす子供たち

小さな子供たちがiPhone、iPadを使いこなす姿を見かけるようになりました。人気のキャラクタの映像を見たり、お絵描きアプリに夢中になって、おとなしくしている子供たち。物心ついたときからインターネットが使え、デジタルデバイスに親しみ、何の違和感もなく使いこなしています。なかにはテレビ画面を触って操作しようとしたり、放送中のテレビ番組が巻き戻したり検索できないことを不思議に思う子供もいるそうです。

「子供」といっても、Kidsとカテゴライズするときもあれば、Childrenというときもあります。ここでいうデジタルおもちゃが想定する「子供」とは12歳以下、まだサンタクロースを信じているであろう3歳くらいから12歳くらいがターゲットです。

インタラクティブな子供向けおもちゃを作る会社に1996年から続く老舗のAtomocom（http://www.atomo.com）があり、ここのCEOであるBill McIntyre氏のUX Week 2012での講演によると、子供に人気の出るスマートフォンアプリ、タブレットアプリとは次のようなものとのこと。

- おもちゃを画面の中に入れてしまったもの
- ディスプレイ画面をうまく組み合わせておもちゃに入れてしまったもの

いろいろな種類のデジタルおもちゃがありますが、結局夢中になるのは上記のどちらかとのことです。つまりはモノとして存在感のない完全なデジタルな概念よりも、やはり手で触ったり、見たりすることのできる「おもちゃ」を媒介として何かを感じ取っていることが解ります。

古くから楽しまれてきた幼い子供の手先の感覚などを進歩させる遊びにはいくつかの種類があるらしく、単にふざけているように見える仕草も、手先を鍛えるために無意識のうちに行動していることもあるそうです。

- ビリビリ紙を破く感覚
- カチカチと手に持ったもの同士を叩いて音を出す感覚
- 物を叩くという感覚。手に持ったもので、手に持っていないものを叩いて音を出す感覚
- 薄いものを適度な力でコントロールする力。ティッシュを箱から引っ張り出すなど
- 何かをつまむ。小さな不安定なものをつまんでどこかに移動する動作
- 何かを引っ張る。反発や反動のあるものをつかむ
- 何かの動きを真似る
- 隠れている何かを探し出す
- 手や顔を動かして、鏡が映し出すものの存在の意識化
- 物の揺れや回転を楽しむ
- 見て楽しむものと、触って感じるも

おもちゃを画面の中に入れたアプリの例「Digital Barbie」


画面をおもちゃの中に入れたアプリの例「HappiTaps」


画面をおもちゃの中に入れたアプリの例「Woogie」

※本記事で紹介しているものは国内未発表・未発売のものを含んでおります。

のの違いの意識化
- 物や、不安定な物のバランスを保つという感覚。何かを積み上げる感覚
- 目と手との連動。手で動かしたものが動いて目で認識できるということを知ること

そして、子供用のアプリやおもちゃを考えるときには、一般の大人のユーザとはまた違った観点で考えなければいけません。それは次のような事柄です。

- 子供たちがひと目で理解できるか?
- 年齢ターゲットを細かく決める。子供の振る舞いは数ヵ月の年齢の差でも大違いであることを認識
- 女の子と男の子を別々に分けて考える。または、女の子と男の子を混ぜた環境でも考える
- 複数人で使う場合、子供が1人で利用する場合、親と一緒の場合、それぞれどう違うか考慮する
- 慣れたら使えるのか? 学習して慣れることができるのか? 時間をかけても慣れることができないのか?
- 子供が予想だにしない事柄は何かないだろうか?

このような事柄を考えたうえで、アプリの企画やおもちゃとしての楽しみを考えますが、さらに実際に操作するときの細かな事柄を考慮しなければいけません。誰もが昔は子供だったはずですが、そのころのことは忘れてしまっていますし、何より、いまこの記事をご覧になっている読者の方々が幼児のころは、タブレット端末なんてなかったのですから、新しく考えなければいけないこと、学ばなければいけないことは数多くあります。さらに次のような点が、子供向けタブレットアプリのためのコツやポイントです。

- 子供たちは指が小さい。個人差はあるが、手がとても小さい
- 利用するとき、画面と顔との距離が近い場合が多い
- こういうふうに使ってもらえるだろうといった前提とか、仮定とかはまったく成り立たないことを認識する
- 親に助けてもらうことを前提にし、親がすること(設定や購入など)と、子供だけでできることに分離する
- 偶発的な成功を喜んだり、小さな達成感を積み重ねられるようにしくむ
- スキル向上の楽しさと、集中力の持続時間のバランスを考える(子供たちはごく短時間しか集中し続けられない)
- 時間が経っても、また再び遊びたいと思わせる何かを作り込む
- 変な方向から、どの指で触るのかわからないことを考慮する。右手の人差し指とは限らない
- 言葉より絵。多少の文字が読める、単語が理解できるとしても文字より絵を使う。知育玩具の場合はその限りではない
- タップとスワイプは得意だけど、ピンチインやオブジェクトをつかんだままドラッグなどの操作は無理
- LOADING中のプログレスバーなどで待たせてはいけない。少しでも待った時点で子供は使わなくなる

大人も子供の気持ちに

LEGOブロックなど品質が高く、長く使われているおもちゃは、子供用として素晴らしいだけでなく、大人も楽しむことのできるものがほとんどです。手軽に使えるとともに、習熟することでより使いこなすことのできる楽器系のおもちゃも子供も大人も楽しめる部類かもしれません。

ひとつ間違ってはならない事柄は、子供は何も大人が安易に考えるような「子供っぽいもの」で遊びたいわけではないのだということです。大人は「子供っぽい、子供用のもの」を使わせたがるかもしれませんが、「大人と同じもの」に憧れて、子供騙(だま)しではない、大人と同じものを使いたいのです。

子供たちが小さなころからデジタルデバイスに親しむことは素晴らしいことです。これからはデジタルデバイスを活用し、単に教科書や教材をデジタル化するというだけでなく、教育の形も違ってくることでしょう。でも子供たちにとっては、子供のころにしか体験できない、習得できない感覚も多いのではないでしょうか?

ふわふわ、ざらざら、ぽこぽこ、ぱらぱ

レオナルド・ダ・ビンチの手稿にある兵器などをモデル化したおもちゃ(童友社)

人が実際に着られるコスプレ用Iron Manスーツ。受注生産で180万円(ルービーズ)

LEGOスターウォーズのキャラクタ。実物大?(レゴ)

LINE TOWNマイタッチ。カラー液晶、近距離通信でスタンプ交換できる子供用デバイス(タカラトミー)

ら、ぺたぺた、さらさら、しわしわ、ぎざぎざ、べどべと、ちくちく

これらはモノの触感を言葉で表現したものです。将来的にはデジタルデバイスでもこれらのハプティックなパラメータを表現できるようになるかもしれませんが、現在はまだまだ研究段階のものばかりで、現実の世界に展開してはいません。

子供たちにはデジタルに親しむ前に、絵の具で手を汚したり、少しはケガもしながら工作したり、指先の触感や、素材に対する感性、物の強度や破壊に対する感覚などを磨いてほしい気がします。人間の根源的な感覚を磨くのは、子供の特権なのですから。子供のころに身につけた研ぎすまされた感覚がデジタル機器を使いこなしたり、デジタル機器用のサービスやアプリを作る際にも役立つのではないかと思います。

今年秋には、Digital Kids Summit（http://digitalkidssummit.com）という子供向けデジタルエンターテインメントを考える、初めてのカンファレンスが開催されます。子供向けデジタルデバイス、アプリやサービスの進化が楽しみです。

最後に、2013年6月13日から16日の4日間、ビッグサイトで開催された「東京おもちゃショー」から子供向けデジタルおもちゃを写真で紹介して結びとしましょう。SD

知育アプリ入り、子供向けタブレット。ラバーケースで頑丈に保護されている（メガハウス）

gadget 1

鳥オルガンハウス

※執筆時、該当サイト不明

手回しオルゴール

鳥オルガンハウスは紙の曲シートに専用パンチで音階となる穴をあけ、ハンドルを回しながら空気の力で笛をならす、手回しオルゴールです。ふいごの空気が笛の穴を通ることで音階が発せられ、小鳥が歌うような感覚で音が奏でられます。電池いらずのデジタルではないおもちゃですが、なぜか昔のコンピュータで利用されていたパンチカードを思い出したので紹介させていただきました。

gadget 2

AutoMee S

http://www.takaratomy.co.jp/products/automee/

ロボット掃除機

AutoMee Sはスマートフォンやタブレットをきれいにしてくれる、超小型ロボット掃除機です。色は全4色で定価は各1,575円（税込）。直径7cm、手のひらサイズのロボットがスマートフォンの上を縦横無尽に動き回って、指紋のあとや、皮脂汚れを自動的にまんべんなく拭き取ってくれます。スマートフォンやタブレット本体の端に来ると、自動的に方向転換して落ちることはありません。スマートフォンサイズで約4分、iPadなどのタブレットサイズで約8分のお掃除時間です。

gadget 3

ボイスそろばん

http://daiichi.c-syoku.com/kyouikusyouhin/voiceabacus.html

デジタル音声そろばん

ボイスそろばんはそろばんの操作を習得するための教育グッズです。数や数値、計算を日本語と英語で学ぶことができます。そろばんのことを知らない外国人がしくみを理解するのにも役立ちます。今までデジタルとは関係ないアナログ一辺倒と思っていたものも、新たな付加価値を得たうえでデジタル化されたデバイスの好例です。定価3,990円（税込）。そろばんの老舗メーカーの製品です。

gadget 4

ケロミン

http://www.keromin.com/

ぬいぐるみ型楽器

ケロミンは、カエル型のテルミンとも言えるパペット型の電子楽器です。手袋のように手を入れて、口をパクパクさせることで演奏します、定価が49,875円（税込）とおもちゃとしては少々高額ですが、楽器だと考えれば納得です。カエルの口をパクパクさせることで音程が変化します。初期バージョンよりも相当進化しており MIDIアウトを搭載、楽器の音のほかにもカエルの声を録音したリアルな音などで演奏することもできます。

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

結城浩の再発見の発想法 3

Consistent

Consistent—コンシステント

コンシステントとは

コンシステント(consistent)とは、日本語で**一貫している**という意味の形容詞です。名詞形はコンシステンシー(consistency)「一貫性」で、コンシステントの対義語はインコンシステント(inconsistent)です。

何かまとまったものを評価するとき「一貫性」は指標の1つです。

たとえば、Webサイトのデザインを評するとき、「デザインに一貫性がない」と表現することがあります。これがまさに、今回お話しする「コンシステントかどうか」です。

Webサイトのデザインが「一貫していない」と言われるのはどういうときでしょうか。よくあるのは、同じものが異なるデザインで表現されているときです。押しボタンが、こちらではフラットなデザインなのに、あちらでは三次元的なデザインになっているとき、私たちは「一貫していない」と感じるでしょう(**図1**)。

あるいはまた、誰かが書いたプログラムを「書き方に一貫性がないね」と評することがあります。これもまたコンシステントかどうかを気にしている状況です。

プログラムのソースコードが「一貫していない」のはどういうときでしょうか。よくあるのは、同じものが異なるコードで表現されているときです。ボタンを表す識別子にbtnとbuttonとbという3個のプレフィクスが混在していたら、一貫性がないと感じるでしょう(**図2**)。

ルールでコンシステントにする

さて、コンシステントかどうかを判断するには複数の構成要素が必要になります。構成要素がたった1つしかない場合には、一貫性を論じることはできません。英語でも“be consistent with...”という言い回しで、**何に対して一貫しているか**を明確に表現しています。

先ほどの例でも、ボタンのデザインが一貫してない例や、識別子の書き方が一貫していない

▼図1　一貫性のないWebサイトのデザイン

▼図2　一貫性のないプログラムの書き方

Consistent	Inconsistent
... button_reset = ... button_send = ... button_back = ...	... btn_reset = ... button_send = ... b_back = ...

例を述べました。一貫していないとき、必ず2つ以上のものの衝突があります(**図3**)。

一貫性を持つものを作り出すには、何らかの**ルール**が必要です。ルール、基準、ガイドライン、ポリシー……呼び名はさておき具体的なものを作るときに従う「約束」を用意するのです。

たとえば「フラットなデザインを使うこと」というルールがあれば、Webサイト作成者は迷いません。また、「ボタンを表すプレフィクスはbuttonにする」というルールがあればプログラマは迷いません。

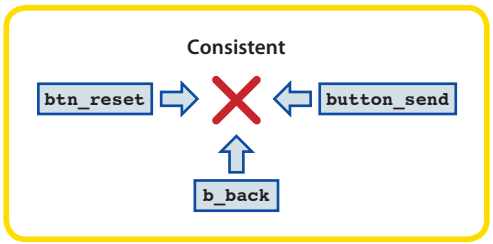
大きなものに一貫性を持たせたいなら、ルールが必要になります。ルールがあれば、従っているかどうかをチェックできるからです(**図4**)。

ルール自体の一貫性

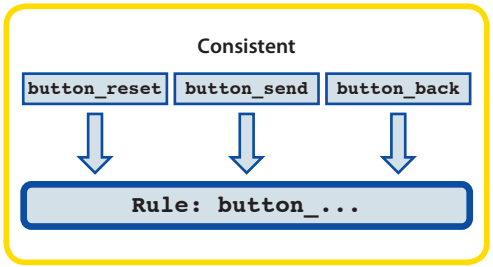
ところでルールそのものも、大きくなると一貫性を持たせることが難しくなります。

Webサイトの例でいえば、形、色づかい、振る舞いなど多くの要素に対してルールを定めるのはたいへんです。コーディングルールも、細かく定めれば実装者は迷わなくなりますが、ルールを定めること自体が難しくなるでしょう。

▼図3　一貫性を論じるには複数の要素が必要



▼図4　一貫性を持たせるにはルールが必要



一貫性と作成者

Webサイトでもソースコードでも、大きなものが一貫して作られているとき、私たちはそこに**ルールの存在**を感じます。「でたらめに作られたものではなく、あるルールに従って作られたものだ」という印象を受けます。そしてさらに、**作成者の存在**を感じます。実際には多くの作成者がいるのかもしれません。しかし、ルールに従い一貫性を生み出しているものは、誰か1人の作成者によって作られた印象を与えます。

人間はコンシステントか

さて、人間はコンシステントなのでしょうか。行動の予測が付く部分についてはコンシステントといえそうです。たとえば、映画の話をしたらすぐに食いつくけれど、ファッションの話には興味がないといった一貫性は、その人の「趣味」というルールなのかもしれません。

一見したところ一貫性がないように見えても、深いレベルの一貫性がある場合もあります。たとえば、ある知人は、折り紙と建築物と昆虫に興味を持っています(**図5**)。

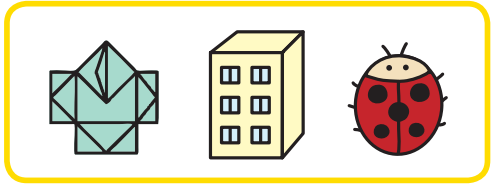
「折り紙」と「建築物」と「昆虫」に共通のものは何もないように見えます。でも筆者は「この人は、形に興味があるんだ！」と気づきました。

折り紙、建築物、昆虫。この人は、形を作ることや形を見ることが好きだったのです。そこには一貫性がちゃんとあったのですね。

◆ ◆ ◆

あなたの周りで「一貫している」と感じるものはありますか。その背後にルールはありますか。ぜひ考えてみてください。SD

▼図5　一貫性がないように見えても……



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4回　嵐の船出 ―― 一億総プログラマー国家計画

秋葉原にARC(アーク)という研究・教育組織を作った僕たちは、まず組織が取り組むべき課題を明確にする必要がありました。当面、ARCのミッションは新しいミドルウェアである「enchant.js(エンチャントジェイエス)」とゲーム開発コンテスト「9leap(ナインリープ)」の普及活動を行うこと。

しかし、まったく新しい組織であるがゆえに解決しなければならない問題はまだまだありました。

オープンスペースの難しさ

ARCの場所を秋葉原UDXの裏にある高級マンションと決めました。震災直後で家賃が安かったのと、手頃な広さだったのが決め手でした。28階にあるオフィスは、眼下に秋葉原とスカイツリーを一望できる絶好のロケーションです。

ところが規模が大きくなっていたUEI本体のセキュリティ基準では、こうした民家を改装したようなオフィスとVPNで接続するのが非常に難しくなってしまいました。

僕はARCの要員は数名の社員のほかはすべて学生を集めようと決めていました。そして学生が友達を自由に連れて来られるようなオープンスペースとして、ある程度セキュリティレベルを下げた場所が必要だったのです。しかしセキュリティレベルを下げるということは、すなわち情報管理が難しくなるわけで、結局のところネットワーク的にARCは孤立することになってしまいました。

はじめのうちは、伏見遼平が友達を連れて来たり、僕もいろいろな学生を紹介されてARCに呼んで話を聞いたりとオープン性を活かした使い方をしていました。ところがここに残るわずか数名の社員にとっては、ARCは居心地の良い場所ではなかったようです。稟議書1枚印刷するのに、わざわざ十数分かけて本社まで移動しなくてはならず、また、ミーティング場所が本社なのかARCなのかによってできることも変わってくるため、社員には大変不評だったのです。

「どうしてARCが必要なんですか？　この部署は何をやるんですか？　なんのためにやるんですか？」

あるとき、ついに不満を爆発させた辻秀美が僕に食ってかかってきました。普段寡黙な人物

秋葉原の高層マンションでスタートしたARC。伏見を中心とした大学生が意見交換しながら新しい発明を目指した

が突然怒りだすと周囲は戸惑うものです。僕もこのときは戸惑いを隠せませんでした。

なにより僕としては、この組織の必要性や意図をほかの誰でもない、学生時代から世界中駆け巡りながら一緒に仕事をしてきた辻にだけは説明せずとも解ってもらえる、と思っていたのです。それは僕の考えの甘かったところです。

そして、そういう不満を持っている社員はほかにもたくさんいました。こともあろうにそうした社員の多くは、ARCの所属として社内からは特別待遇と看做(みな)されている人間たちでした。「学生にお金をバラまいて、卒業後も返還の必要のない奨学金を与えて、それでうちの会社のどういう利益になるっていうんですか？」

彼らの主な主張はこういうものでした。そこで、僕は改めて、ARCの正社員を集めて説明する必要にかられたのです。

事業と教育

僕がなぜ、社内に教育と研究を専門とする部署を立ち上げようという考えに至ったのか。

確かにARCは社内のみならず業界全体を見渡しても、とても特殊な部署です。ARCの主たる事業である9leapでは、学生は上位入賞しさえすれば最新のPCを無料でもらえます。参加費はもちろんゼロです。enchant.jsにいたっては完全に無料です。

正社員は大学院卒、またはそれに相当する特殊な才能が認められることが入所条件で、その他のスタッフはすべて学生。しかも、厳しい面接と筆記試験をパスした学生だけです。学生はARCから給料をもらいながらも、事業としての明確なミッションは与えられず、ただやってきて好きなことをして帰って良い、というルールです。専用の最新PCが与えられ、必要ならばどんな道具も買い与えます。

無料のソフトを作り、無料でPCをバラ撒き、金を払って雇った学生には仕事を与えない。こんな場所をいったいなぜ作る必要があったのでしょうか。

普段のARC。フリーアドレスで各自が自由に座る場所を決めていたが、社員には不評だった。写真右は本社との往復で疲弊した辻

そのすべての目的は、教育です。

会社というのは、常に優秀な人間を獲得できるとは限りません。新卒一括採用では、中小零細企業にはまず優秀な人間は来ません。なにしろそれはARCを支えている辻、近藤誠といった人材自身が、証明しています。彼らの多くはアルバイトの名目で会社に入り、数年の月日をともに過ごし、UEIという会社を良く知ったうえで自ら進んで入社してきています。そして受け入れ側の僕たちも、彼らの人となり、性格、才能、そういったものを充分吟味したうえで採用に至っています。これは学生時代という多感な時代をともに過ごすことによってのみ得られる独特の信頼関係です。

UEIにとって重要な新人教育は、新製品を生み出し、戦略的な視点で売り出していける人物を育てることです。しかし、まったく新しい製品を生み出すためには、新卒で一括採用した人間にランダムに企画書を書かせるだけのやり方では、コンスタントに優れた企画者を見つけ出すのは不可能です。というのも、技術系企業における企画、つまり作戦立案とは、技術に明るいだけでなく、世の中に明るく、人々の心の動きの裏側を読み取り、目に見えない欲望を形にしていくような発想、視点が必要だからです。

そういう発想や視点は、むしろ純粋に技術だけを追求していては産まれません。しかし同時に、技術を中途半端に習得していても産まれません。

教養と発想、そして技術、その3つがすべてそろっていて初めて優れた企画者と呼べるのです。

さらに、より視野の広い戦略的なマーケティングまで含めると、それを構築できる能力を生まれつき持っている人はいません。そういうことを担当するチャンスが巡って来るケースも稀ですし、うちのような会社の場合、その数少ないチャンスをものにできなければ会社がなくなってしまいます。そうならないためには、日頃から訓練をする必要があるのです。

中小企業における競争力とは、人材がどれだけ業界標準から見てユニークであるか、ということです。ユニークな人々が集まっていると思ってもらえれば、我々にしかできない仕事をまわしてもらうことができます。その仕事が多少割高であっても、我々にしかできないなら発注をいただけるのです。これが我々の考える価値創造です。しかし、凡庸な人材しかないと思われれば、残っているのは価格競争だけです。インドやベトナムなど、オフショア開発が常態化してきた昨今において、人材のユニークさを武器にしないソフト会社はやっていけないと思います。

その会社の優れた人材を創りだすのは、教育です。ARCを設立した理由は、僕がただ教育に専念するためだったのです。

水滴と川

ここまで説明しても理解は得られませんでした。

「清水さんは会社のお金を自らの青い理想のために浪費しているだけではないですか」

辻と近藤は僕に面と向かって言いました。

「会社というのは、川の流れのようなものだ。人は川を流れる無数の水滴だ。誰でも永遠に生きられるわけではない。支流を伝ってほかの川に行くことを誰も止められない。以前、君たちがアルバイトに来たとき、もし僕が、“このバイトをやるならうちの会社に就職をしなさい”と言ったら、君たちはアルバイトを引き受けたのか？　まだいろんな世界が見たい、この会社に決めていいのかどうか迷うはずだ」

「それはそうかもしれませんが……」

「唯一、会社ができることは、才能ある人たちがその才能を発揮しようとするとき、最もやりやすい環境を提供すること。信頼できる仲間や先輩や指導者、自分の才能を何倍も引き出せるツールとしての組織を作ること。そのうえで選んでもらうのだ。僕たちではない。“彼ら”が選ぶのだ。以前、君たちがそうしたように」

それでも納得はしてないようでしたが、それから彼らは黙々と働きました。僕は彼らを説得することを諦めました。彼らもいずれ解ってくれる日が来るだろう、と祈るだけでした。彼らの多くは決して不満の色は隠さなかったものの、ともあれARCという船は荒波の中を漕ぎ出しました。

もう後戻りはできません。

一億総プログラマー国家計画

社内のゴタゴタとは裏腹に、ARCのスタートした新事業、enchant.jsと9leapの出だしは絶好調でした。毎月行われる「9Days Challenge」への応募も好調で、ARCの事業は早くも社外からの注目を集め始めました。

中心となったのは、東大の伏見、慶應の伊藤、電通大の杉本らが立ち上げた9leap。手軽にゲームプログラミングができる東大の田中諒によるenchant.jsと組み合わさって、すぐにアップロードしてユーザの反応が見られるという点がとくに好評でした。学生を中心に据えた組織編成が早くも効果を発揮してきたのです。彼らはアグレッシブで野心に燃えており、多少の困難をものともせず新しいことに挑戦していきました。

ほどなく週刊アスキー、月刊BestGearでの連載も始まり、多くのアマチュアプログラマたちがenchant.jsや9leapに入門するようになりました。

対する我々も、enchant.jsの公開と同時に行った東京のワークショップをはじめ、仙台、札幌、大阪と国内4ヵ所で無料のワークショップを開催、

さらに沖縄、九州、福井など、呼ばれればどこの県にも無料で講師を派遣しました。しばしば僕も開発者の伏見や辻を伴って、各地でワークショップを開催しました。時には、海外、たとえば世界で初めて開催されたHTML5ゲームのカンファレンスである「onGameStart」にも出向き、拙い英語ながらenchant.jsの存在をアピールしました。そのとき我々が配ったチラシにはこう書きました。

「ゲーム作りのたのしさ、無料で教えます。
全世界、どこへでも」

そしてそれから、本当に世界中の学校を巡りながらenchant.jsを布教する活動を行うことになります。1円も売上げの上がらないはずのオープンソースソフトウェアであるenchant.jsのためになぜ？と多くの人は疑問に思ったようです。

しかしこれもまた、僕の考える教育のあり方です。

本来、教育とは、教育者が与えると同時に、学習者は自ら進んで学ぶ姿勢を示すものです。ところがいつのまにか、学習者は自分の内側から沸き上がる知識欲や向上心ではなく、周囲から、時には教育者自身からのプレッシャーによって学ぶ姿勢を頑張って維持する、といういびつなものに変わりました。

僕は教育が持つ本来の姿に戻すべく、まず与え、それを学習者が自らの意志で進んで学ぶ、という形にこだわりました。学ぶことがその人本来の喜びになれば、その体験、学習体験には金銭を支払う価値が出てきます。ディズニーランドや映画館の入場料と同じです。

しかしその喜びが充分知られていないうちは、まず知ってもらうことが大切です。だから無償で、しかも全国的に飛び回ってワークショップを開いたのです。

そして多くの人々に無償で我々の成果であるenchant.jsを知ってもらった結果、我々が手にするのは何か。それは「親しみ」です。野球少年が監督やチームメイトと築くような親しみを、我々とプログラミングに対してももっていただきたいのです。

enchant.jsを普及させるため全国飛び回ってワークショップを開催。写真はクリプトンフューチャー社の協力で実現した札幌でのイベント

ARCではwise9というブログメディアのための取材旅行も行うことがあった

ポーランドで開催されたHTML5ゲームイベント「onGameStart」の様子。奥のほうにenchant.jsのロゴが見える

これもまた、お金だけでは決して手に入れることができない、非常に重要な感情です。プログラミングを教育することで事業を成立させ、受験のための塾のように不安をベースとした教育ではなく、喜びをベースとした趣味のための教育が実現すれば、プログラミングはより多くの人に親しまれることでしょう。すべての人々がプログラミングを覚えれば、世の中はもっとずっと変わったものになるはずです。

そこで、「一億総プログラマー国家計画」と称した活動を始めることにしました。プログラミングをもっと簡単に、親しみやすいものに変えて行く。そのための布石が、まずはenchant.jsと9leapだったのです。SD

コレクターが独断で選ぶ!

偏愛キーボード図鑑

株式会社 創夢
濱野 聖人 HAMANO Kiyoto
khiker.mail@gmail.com
Twitter：@khiker

第4回

キーがないキーボード!?

orbiTouch & COOL LEAF

写真1　orbiTouch

はじめに

前回は、エルゴノミクスキーボードの雄であるKINESISコンタードキーボードとMaltronキーボードを紹介しました。今回もその流れを組み、エルゴノミクスキーボードを紹介します。エルゴノミクスキーボードにはさまざまな種類があり、変わった形状を持つものも少なくありません。その中でもとくに変わったものの1つで、キーのないキーボードであるorbiTouch（オービタッチ）を紹介します。また、「キーがない」つながりで、COOL LEAF（クール リーフ）も紹介します。

orbiTouch

orbiTouch（**写真1**）は、Blue Orb社が生産しているエルゴノミクスキーボードです。キーのないキーボードとして有名です。

特徴

orbiTouchの特徴はキーがないことに加え、キーボード自体もかなり大きい部類に入ります。特殊な形状をしていますが、OSからは普通の英字キーボートと認識されます。

キーボードに2つのお碗があり、それを上下左右8方向に動かして（スライドさせて）操作します。お碗をつかめさえすれば操作ができるため、指や手首を動かす必要がほとんどありません。おもに障害を持つ方が使われているようです。

操作は、格闘ゲームのコマンド操作に似ていますが、少し違います。左側のお碗を8方向のうちどれか1方向に動かしたまま、右側のお碗を動かすと何らかのキーが押されたこととなります。

操作補助のために左側のお碗の各方向は赤・黄・緑……と色分けされており（**写真2**）、右側のお碗の各方向は、左側のお碗がその色の方向にあるときに入力できる文字が書かれています（**写真3**）。たとえば、左側のお碗が左（赤色の部分）にあるとき、右のお碗を左に動かすと「a」と入力できます。

SHIFT や Ctrl や Alt などの修飾キーも扱えます。orbiTouchはモードの概念を備えており、SHIFTであれば、左側のお碗を上に2度動かすことでShiftモードに入ります。この状態で普通の文字を入力するとSHIFTを付与して入力したものとなります。Ctrl、Altも同様です。

このモード機能はマウスモードも備えています。左側のお碗を下に2度に動かすとマウスモードとなります。マウスモードになると左側のお碗で左クリックと右クリックを、右側のお碗でカーソル移動を行います。マウススクロール機能は備えていません。

写真2　orbiTouch（左のお碗）

写真3　orbiTouch（右のお碗）

写真4　操作コマンドチートシート

どのように動かしたら、どの文字が入力できるかについては、orbi Touch標準でチートシートが付属しています(**写真4**)。

このように学習コストは相当大きく、お碗を8方向に動かすコストも大きいため、普通のキーボードと同様の速度で操作することは難しいと思います。ただ、学習をサポートするために、先ほど挙げたチートシートのほかに説明書や練習ソフトを収めたCDが付属しています。

接続はPS/2ですが、筆者が購入したときにはPS/2-USB変換アダプタが標準で付属していました。最近では、Bluetoothによる無線バージョンが発表されています。

入手方法

本家Webサイト[注1]やアメリカのamazon.comで販売されています。ただし、筆者購入当時では、日本への発送はしていませんでした。日本代理店も筆者の知る限り存在しません。また、オークションサイトにも稀にしか出品されていません。

最も簡単な購入方法は、海外輸入代行業者やamazon.com代行業者を通して買うことです。筆者はこの方法で入手しました。代行料は業者によってマチマチですので、よく調べてみることをお勧めします。値段は399ドルで販売されています。

COOL LEAF

COOL LEAFはミネベア㈱による全面フラットのキーボードです。

特徴

キーがなく、全面鏡面でフラットであることが特徴です。見た目こそ特徴的ですが、OSは普通のUSB日本語キーボードとして認識します。

キーボードをPCに接続していないと鏡面には何も表示されません(**写真5**)が、PCに接続するとLEDバックライトで文字が浮かび上がります(**写真6**)。その文字にタッチするとキーが入力されます。タッチした際に「ピッ」というビープ音がします。このタッチの認識は感圧式ではなく静電式です。そのため手袋をしていると押しても認識されません。

また、全面鏡面でフラットであるため、掃除が非常に簡単です。布でふきとるだけでよいです。その反面、入力はしづらいと言わざるを得ません。ホームポジションが取りづらく、タッチタイプはかなり練習しないと難しいです。

筆者の所感では、普通のキーボードの代替として使うことは難しく、清潔さが求められるような場所で、限定された用途で利用するためのキーボードであると思います。

入手方法

以前は、Amazonなど日本のWebショップで購入できましたが、今は取り扱っていないところが多いようです。秋葉原などで店頭を探せば見つかるかもしれません。値段は購入当時で約2万円でした。

◆ ◆ ◆

今回はキーのないキーボードを紹介しました。このような一風変わったキーボードはいくつか存在していますので、興味のある方は探してみるとおもしろいかもしれません。SD

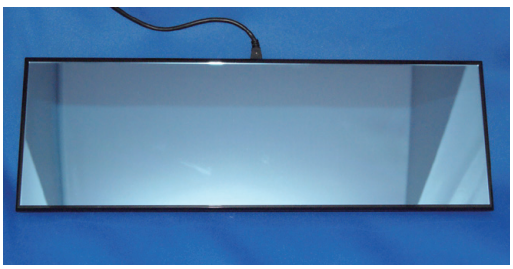


写真5　COOL LEAF(電源オフ)

写真6　COOL LEAF(電源オン)

注1) http://orbitouch.com/

はんだづけカフェなう

Maker Faireで見かけた3Dプリンタ

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/

前回紹介したMaker Faire Bay Areaには数多くの3Dプリンタ関連の出展がなされていました。今回の本誌の第1特集は「3Dプリンタが未来を拓く理由」ということで、前回は紹介しなかった3Dプリンタ関連の出展をいくつか紹介していきたいと思います。

筆者が会場で見かけたものの中で印象的だったものの1つに、大きなUltimaker Robotの出力があります（**写真1**）。なぜか会場隅の、飲食物の売店カウンターに置かれていたのですが、高さが20cmほどある巨大なものです。

今号の第1特集の第3、4章を担当した山田斉氏が3Dプリンタを選ぶときにUltimakerにした理由の1つに、昨年一緒に参加したOpen Hardware SummitでもらったこのUltimaker Robotのプリントの品質に納得がいったというものがあります。その思い出のロボットが大きな姿で出力されていることもあり、とても印象的でした。

123D Catchフォトブース

大手3D CADメーカーであるオートデスク社は、同社の123D Catch[注1]のテクノロジを使った、バストアップ3Dモデル撮影ブースを設置していました（**写真2**）。たいへんな人気で多くの人が並んで撮影してもらっていたのですが、筆者も撮影をしてもらってきました。

このブースの中にはたくさんの1眼レフが並べられており、人の上半身の写真をあらゆる方向から一度に撮影することが可能になっています（**写真3**）。

撮影してもらったデータは後日電子メールで送られてきます。Autodesk 3D modelという拡張子.3dpのファイルでしたが、これはPCとiOS用に提供されている123D Catchアプリケーションなどで開くことができます。

ちゃんとした3D写真が届いた方もいるのですが、残念ながら筆者が撮ってもらったデータは**写真4**のように残念なものでした。これはiPhone版の123D Catchでファイルを開いてガッカリしたときに撮ったスクリーンショットです。もちろん、3Dデータですので指で操作することで、この写真を回転させたりできます。

3Dデータですので、筆者は3Dプリンタで自分の胸像を出力してみようと思っていたのです

注1）　http://www.123dapp.com/catch

▼写真1　大きなUltimaker Robot

▼写真2　123D Catch フォトブース

▼写真3　ブース内部

▼写真4　筆者の3D写真

が、かなわず残念です。

オートデスク社は、先般よりMakerbot社との提携をすすめており、同社のブースには多くのReplicator 2が置かれていました。従来オートデスク社のソフトウェアはとても高価なプロ用の製品ばかりだったのですが、近年はホビーユーザ向けの無償のソフトウェアをリリースしています。先ほど紹介した123D Catchを始めとする123Dシリーズだけでなく、AutoCAD WSやInventor Fusionといった機能縮小版まで無償で利用できるように公開されています。OS Xユーザであれば、App Storeで“Autodesk”と検索するだけでこれらのソフトウェアを見つけることができます。

余談ですが、Maker Faireでは物販コーナーがあり、そこではReplicator 2やUltimakerといった3Dプリンタの現物が販売されていました(**写真5**)。Replicator 2などはオンラインで予約しないと買えない状況ですので、今お金を出せば買えるんだということで、筆者は一瞬買いそうになりました。持って帰るのが辛いと思い我慢した筆者をよそ目に3Dプリンタは飛ぶように売れており、家族連れのアメリカ人がこれをかついで帰って行く光景に驚愕しました。

SandBox

SandBox[注2]は開発中の粉末を使った、オープンソースの3Dプリンタです(**写真6**)。まだWebサイトにも詳しい情報が載っていないのですが、接着剤を使って粉末を固めて出力をする方法を採用している模様です。おそらく、第1特集の2章でも紹介している粉末固着積層法を採用しているのでしょう。材料の粉には、セメント、セラミック、木材、ガラス、砂糖などなど、さまざまな素材を選ぶことができるようです。

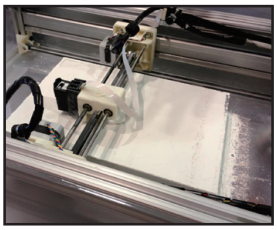
今年の秋にはリリースされる予定となっており、またソースもGitHubで公開するつもりのようです。興味を持った方はWebサイトでメールアドレスを登録しておいてはいかがでしょうか。

SandBoxで出力したものかはわかりませんが、SandBoxのWebサイトでも紹介されていた“UCB Department Architecture”が3Dプリントしたものを出展していてたいへん興味深かったです(**写真7**)。“UCB”と書くと古くからUnixに親しんでいる読者しかピンと来ないかもしれませんが、UCBはBSD(Berkeley Software Distribution)を生み出したカリフォルニア大学バークレイ校のことです。

注2) http://www.sandbox3dp.com/

▼写真5 物販スペースの在庫

▼写真6 SandBox



Type A Machines

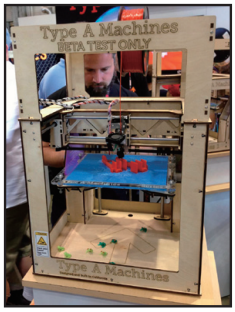
Type A Machines Series 1[注3]は、アメリカではよく知られているものの、日本ではあまり聞いたことがない3Dプリンタです(**写真8**)。展示されていたプリンタにはBETAと書かれていますが、彼らはこのプリンタをオンラインで$1,695で販売しています。

Type A MachinesはこのSeries 1を改良した新機種を開発しているそうで、この新機種、Series 1 Proが展示されていました(**写真9**)。

注3) http://www.typeamachines.com/

▼写真7 UCBの木製プリントの展示

▼写真8 Type A Machines Series 1



フレームをアルミ製に変更したりすることで、速度と信頼性を向上させているそうです。これはエンジニアリングコンセプトらしいのですが、Wi-Fi接続にも対応させ、今年の第三四半期のリリースに向けて開発中とのことでした。

3D Monstr

こちらも見たことのない3Dプリンタだったので興味深く見てきました。3D Monstr[注4]は、高精度な部品と改良型エクストルーダ（詳細は第1特集の第2章を参照してください）が売りの3Dプリンタのようです（**写真10**）。今のところあまり詳しい情報が公開されていないので詳細は不明です。

Kickstarterというクラウドファンディングサイトで資金を調達する予定のようで、プリンタの上にその旨を書いたパネルが置かれていました。まだKickstarterのキャンペーンも開始していないようで、調達を始める前に知名度を上げておくために展示をしていたのでしょう。

Afinia H-Series

アメリカのMake:誌やそのブログではこの3Dプリンタをよく見かけるのですが、Afinia H-Series[注5]は$1,599と安価な部類に入っており、完成度が心配になるうえ、ほかのプリンタと比べるとプリント速度が遅いので、正直筆者はあまり感心を持てずにいました（**写真11**）。

実際のところ出力のクオリティはどんなものなのだろうとブースを訪ね、サンプルを見ていたところ、想像していたよりは良いクオリティのものが置いてありました（**写真12**）。

出力サンプルを見ていると、隣の人が、自分はAfiniaを持っておりプリンタは遅いけど、使うのが簡単だし出力の品質にも満足していると語っていたのが印象的でした。この出力を見るまでは少しも欲しくなりませんでしたが、ちょっと良いかもしれないと思い始めた筆者です。

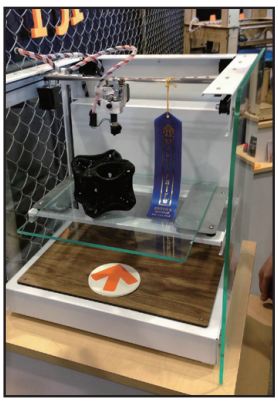
RepRap

RepRap[注6]はオープンソースの3Dプリンタなのですが、地元ベイエリアのユーザグループ“Bay Area RepRap”が自分たちが組んだ3Dプリンタを展示していました。

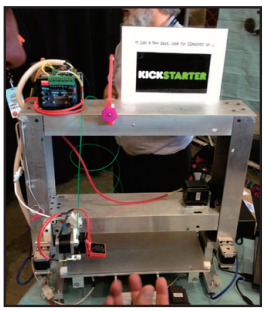
RepRapには多くのモデルが存在するのですが、ここではMendel 90[注7]（**写真13**）と、Prusa i3[注8]（**写真14**）を見かけました。RepRapについては、第1特集の第5章で紹介します。SD

注4） http://3dmonstr.com/
注5） http://www.afinia.com/
注6） http://reprap.org/
注7） http://reprap.org/wiki/Mendel90
注8） http://reprap.org/wiki/Prusa_i3

▼写真9 Series 1 Pro



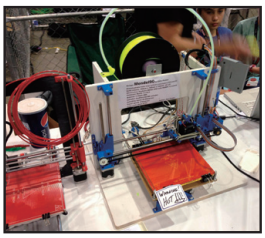
▼写真10 3D Monstr



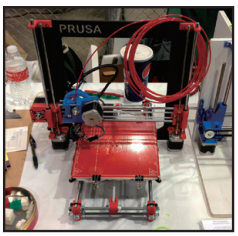
▼写真11 Afinia H-Series

▼写真12 Afiniaの出力例

▼写真13 Mendel 90



▼写真14 Prusa i3

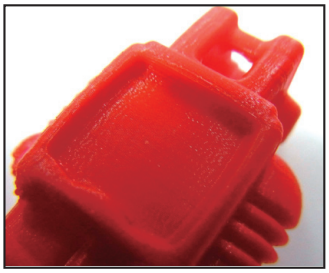


第1特集連動　3Dプリンタ関連の紹介

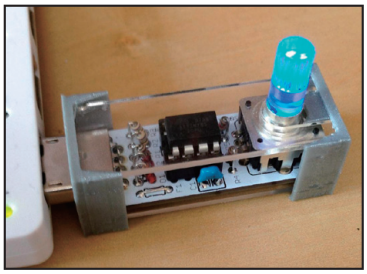
ここでは本誌p.17からの第1特集「3Dプリンタが未来を拓く理由」記事との連動ということで、記事中で、ぜひカラー写真で紹介したいものを集めてみました。

●第1特集　第1章

▼写真15　筆者の手元にあるUltimaker Robotの出力

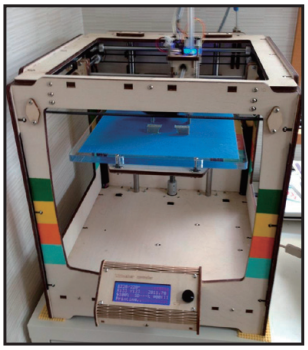


▼写真16　基板にアクリル板のカバーを付けるためのパーツ

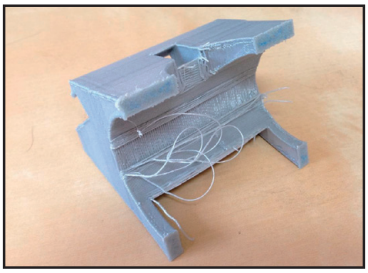


●第1特集　第4章

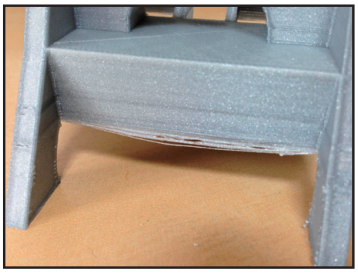
▼写真17　山田氏のUltimaker



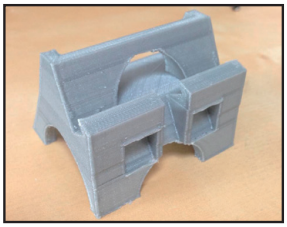
▼写真18　糸くず状にはみ出したフィラメント



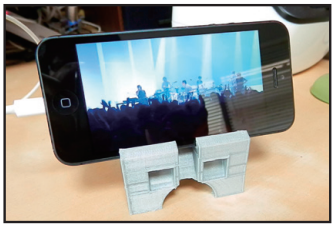
▼写真19　オーバーハングによる樹脂の垂下り



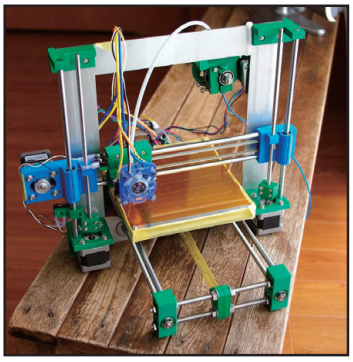
▼写真20　完成したiPhoneスタンド（最終形）



▼写真21　iPhone5を横においたところ（最終形）

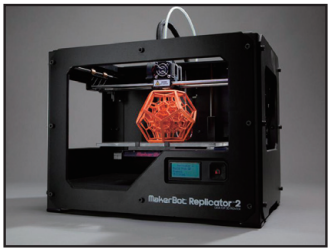


▼写真24　RepRap atom

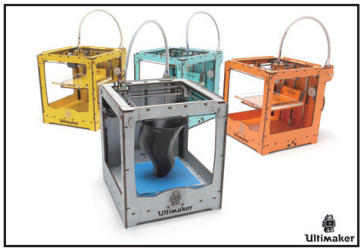


●第1特集　第5章

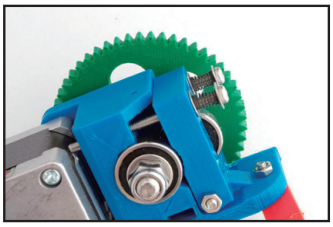
▼写真22　Replicator 2



▼写真23　Ultimaker



▼写真25　atomのエクストルーダ



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8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 タッチパネル操作対応デジタルペン MK-WTP1

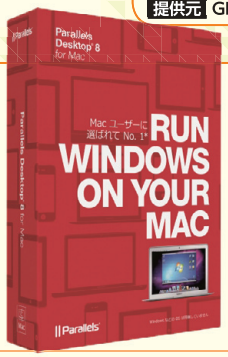
1名

タッチパネルに対応していないPCをタッチ操作可能にするデジタルペン。ベースユニットと呼ばれる機器をPCに取り付け、赤外線と超音波でタッチ位置を検出。ペンを使ってタブレットのようなタッチ操作ができるようになります。対応OSはWindows 7/8、17インチ以下のディスプレイで使用可能です。　※製品にPCは付属しません

提供元 GEANEE　**URL** http://www.geanee.jp

02 Parallels Desktop 8 for Mac

3名



Mac上でWindowsを動作させられる仮想化ソフトウェア。再起動することなくMacでWindowsとMacの両方のアプリケーションをシームレスに実行できます。Mac OS X Mountain Lion/Lion 10.7.4以上/Snow Leopard 10.6.8以上で動作可能。

提供元 パラレルス　**URL** http://www.parallels.com/jp

03 オライリーロゴ入りUSBハブ

5名

オライリーのロゴの入ったUSBハブ。USB 2.0対応のポートが4つとUSBポートのON/OFFを切り替えられるスイッチ1つが付いています。

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

04 自宅ではじめるモノづくり超入門

2名

水野 操　著／
B5判、288ページ／
ISBN = 978-4-7973-7354-7

3Dプリンタによるモノづくりの入門書。無償で使える CADソフト（Autodesk 123D Design）で3Dデータを作成する方法や、3Dプリンタで出力する方法、各種出力サービスについても解説します。

提供元 ソフトバンク クリエイティブ　**URL** http://www.sbcr.jp

05 入門 Android アプリケーションテスト

2名

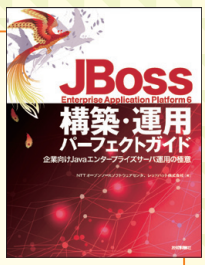
瀬戸 直喜、株式会社プリリアントサービス　著／
B5変型判、336ページ／
ISBN = 978-4-87311-578-8

Androidアプリケーション開発におけるテスト技法について解説します。とくに「必要なテストを必要な分だけ設計する方法」と「テストの自動化によってテスト工数を抑制する方法」を取り上げます。

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

06 JBoss Enterprise Application Platform 6 構築・運用パーフェクトガイド

2名



NTTオープンソースソフトウェアセンタ、
レッドハット株式会社　著／
B5変型判、448ページ／
ISBN = 978-4-7741-5794-8

基幹業務やミッションクリティカルなWebシステムで絶大な支持を受けているJBoss。インストールの方法からエンタープライズレベルでサービスを運用する際のプロの技まで総合的に紹介します。

提供元 技術評論社　**URL** http://gihyo.jp

07 データベースエンジニア養成読本

2名

データベースエンジニア養成読本編集部　編／
B5判、136ページ／
ISBN = 978-4-7741-5806-8

オープンソース系のRDBMSとNoSQLの最新活用術、データ分析やデータ可視化に関する基礎知識など、データベースを活用するエンジニアに求められる素養としての知識をわかりやすく解説します。

提供元 技術評論社　**URL** http://gihyo.jp

第1特集

理論から実践まで 3Dプリンタが未来を拓く理由

最近、何かにつけて話題にのぼる「3Dプリンタ」。モックアップ制作に使える、手に入らなくなった部品を作れる、複雑な作品が作れる、銃だって作れる……と話題ばかりが一人歩きしてしまっているようです。

本特集では、3Dプリンタが実際にどのようなしくみで作品を作ることができるのか、具体的な理論から、モデリング、3Dプリントまでを解説します。

何ができて何ができないのか、実際に制作の現場での試行錯誤なども交えながら多角的にとらえてみました。

理論から実践まで
3Dプリンタが未来を拓く理由

現在の3Dプリンタを取り巻く環境

第1章

坪井 義浩（つぼい よしひろ） ytsuboi@gmail.com
Twitter: @ytsuboi

本章では、最近注目を集めている3Dプリンタを取り巻く状況について解説します。また3Dプリンタだけではなく、3Dプリンタと競合するレーザー加工機やCNCミリングマシン、3Dプリンタのデータフォーマットや3Dスキャナについても触れます。

はじめに

近年、3Dプリンタが急に注目を集めています。この空前の3Dプリンタブームの背景には、やはり個人でも手が届く費用で3Dプリンタが入手できるようになり、さらには簡単に使い始められる環境が整いつつあるということが挙げられるでしょう。たとえば、Makerbot社は2012年に発売されたReplicatorというモデル以来、組み立てとテストが済んだ3Dプリンタを製品として供給しています。それまでの3Dプリンタはキットという形で提供されており、プリンタが届いてから何かを出力してみるまでには組み立てと調整という長いプロセスを経る必要があったのです。

3Dプリンタが安価になった

今では十数万円といった費用で3Dプリンタが入手できるようになりましたが、それまで3Dプリンタといえば自動車や航空機メーカーといった大企業がプロトタイピングのために数千万円で購入するような代物でした。

3Dプリンタのキーとなる特許を多数持っている3D Systems, Inc.（スリーディー・システムズ）は1984年に光造形技術の特許を取得し、1987年には世界初の光造形による3Dプリンタを商用化しています。この3Dプリンタという分野は、私たちに身近な例でたとえると、IT業界のように特許侵害訴訟が頻繁に起こされる、新規参入の非常に困難な業界だったようです。

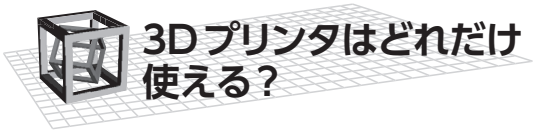
しかし、先の1980年代に出願された特許の有効期限が20年を経た昨今になって切れ始め、2005年にRepRapプロジェクトが始まるなど、今日のパーソナルな3Dプリンタの存在する世の中が始まりました。このRepRapプロジェクトは、Makerbot社のルーツとも言えるプロジェクトで、オープンソースでさまざまな3Dプリンタを生み出してきています。また、Makerbot社も当初はオープンソースで3Dプリンタのキットを販売していたのですが、コピー商品が数多く出たのもあってか、最近ではオープンソースなキットではなく、クローズドな完成品に商品を大きく切り替えてきました。今日では、3Dプリンタを買おうと思ったときに、どれを選んで良いのか迷うほど数多くの3Dプリンタが生み出され、販売されています。

MIT（マサチューセッツ工科大学）メディアラボのビット・アンド・アトムズセンター所長のニール・ガーシェンフェルド氏が記した『ものづくり革命 ― パーソナル・ファブリケーションの夜明け』[注1]でも触れられているのですが、「（ほぼ）あらゆるものを自分で作れる時代」が始まりつつあります。この本では、ビットからアトム

の時代への移り変わりや、消費者が設計者や製造者になれる時代であるということが語られています。

さらには、元Wired誌の編集長として知られる、クリス・アンダーソン氏がベストセラーとなった「フリー」のあとに著した『MAKERS — 21世紀の産業革命が始まる』(**写真2**)でも、「21世紀の製造業はアイデアとパソコンがあれば誰でも始められる」と世界の産業構造が変わるという話が語られています。

パーソナルなデスクトップ3Dプリンタは、本当に世界の産業構造を変えるのでしょうか。3Dプリンタで製品を作ることは可能なのでしょうか。本稿では3Dプリンタの現実を伝えることで、読者各位が3Dプリンタを購入するかどうかの判断をする一助になればと考えて記しました。



3Dプリンタはどれだけ使える？

仕上がりの問題

あらためて紹介する必要がないかもしれませんが、3Dプリンタは樹脂や石膏といった素材を平面に塗り、それを縦方向(Z軸といいます)に積み重ねていくという方法で立体物を作り上げます。

この積み重ねていく素材の高さを積層ピッチと言い、この積層ピッチの細かさが成果物の見た目の精細さに大きく影響します(**図1**)。

昨今では、ホビー向けの3Dプリンタでも積層ピッチ0.3mmとか0.1mmといったスペックがカタログに並んでいます。積層ピッチ0.1mmとなると、1mmの間に10層もの積み重ねがあるということで、信じられないくらい精細なものが出力されるような期待感があります。しかし、人間の感覚というのはとても優れたもので、0.1mm程度の積層では、素材を積み重ねて物を作っていることがすぐにわかってしまいます(**P.15写真15**)。

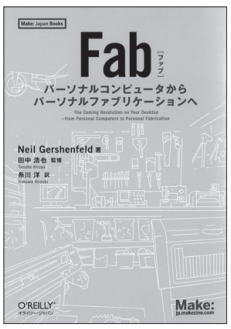
強度の問題

また、この積層をするという方法を使っている限り、層と層の結合は弱くなってしまいます。同じ樹脂を使っていても、樹脂を融かして型に入れて成型したものよりも、3Dプリンタで出力したもののほうが機械的強度が低くなってしまうのは明らかです。

最近の3Dプリンタに関するニュースを見ていると、3Dプリンタはあたかも「魔法の装置」のように語られています。上記で紹介した書籍なども流し読みをすると、3Dプリンタに過大な期待を寄せてしまうことになりそうです。

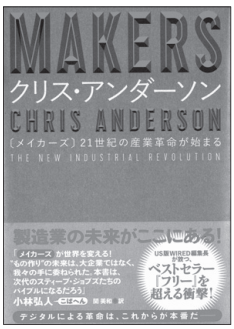
筆者が大きな違和感を抱いている誤解の1つに、「3Dプリンタがあれば製品が作れる」というものがあります。確かに3Dプリンタは大企業の製品開発プロセスでも使われていますが、3Dプリンタで直接コンシューマ向けの量産品を作っ

▼写真1



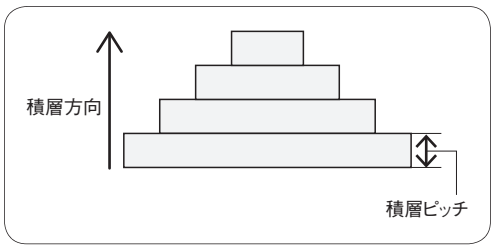
ISBN978-4-87311-588-7

▼写真2



ISBN978-4-14-081576-2

▼図1　積層ピッチ



注1) 同著は絶版で入手困難ですので、オライリー・ジャパンから新装再版されている『Fab — パーソナルコンピュータからパーソナルファブリケーションへ』(**写真1**)をお勧めします。

ている会社は1つもないはずです。

金型も3Dプリンタで作れる?

ちょうど本稿執筆時に、パナソニックが樹脂製品の大量生産に使われる金型と呼ばれる金属製の型を3Dプリンタで作るという報道がありました。記事にもあるとおり、従来はこの金型は工作機械で金属を切削加工して作られていました。こういった切削加工にはのちほど紹介するCNCミリングマシンなどを使っていたのですが、同社では金属粉を融かして積層することが可能な3Dプリンタを使って金型を作るそうです。

出力にかかる時間の問題

量産が3Dプリンタで行われていない理由には、上記の積層ピッチの問題や強度の問題も挙げられます。しかし、3Dプリンタには物を出力をするのにとても長い時間がかかるという問題点が存在します。**P.15写真16**の部品は、アクリル板2枚を留めるための小さな部品ですが、これを比較的速いとされているUltimakerというホビー用3Dプリンタで出力するためには1個あたり約15分を要します。アクリル板を留めるためには、この部品が2つ必要なのですが、3Dプリンタが動いている時間だけでも15分ずつ2個、つまり30分間3Dプリンタが動き続ける必要があるのです。この部品を100セット作ろうと思えば、プリントのセットアップをする時間や、製造不良(失敗)が発生しないという前提で、50時間3Dプリンタが動き続ける必要があるということになってしまうのです。

コストの問題

一方で、この部品はいくらで売れるでしょうか。おそらくペアで数百円がいいところでしょう。つまり、3Dプリンタは1週間以上(1日8時間労働で、週5営業日で40時間)動かしても、数万円の製品しか生産できません。

従来使われている樹脂成形で、中国の工場に製造を委託すれば、数万個を100万円くらいで製造できます。つまり、1個あたりの製造コストは数十円になります。3Dプリンタが製品の量産には向かない、ということを理解いただけたでしょうか。

小ロット作るのには適している

とはいえ、DIYの楽しみは量産品にはない自分のほしいニッチな物を作ることにあるはずです。中国の工場に樹脂成形品を頼むようなコストが出ない1品物、あるいはせいぜい数十個の物を作るのもDIYの楽しいところです。先ほど例に挙げたような小さな樹脂部品にしても、生産中止になっているけれども愛着のある装置の小さな樹脂部品となれば、数千円、数万円のコストや労力を出してもほしいということがあるはずです。

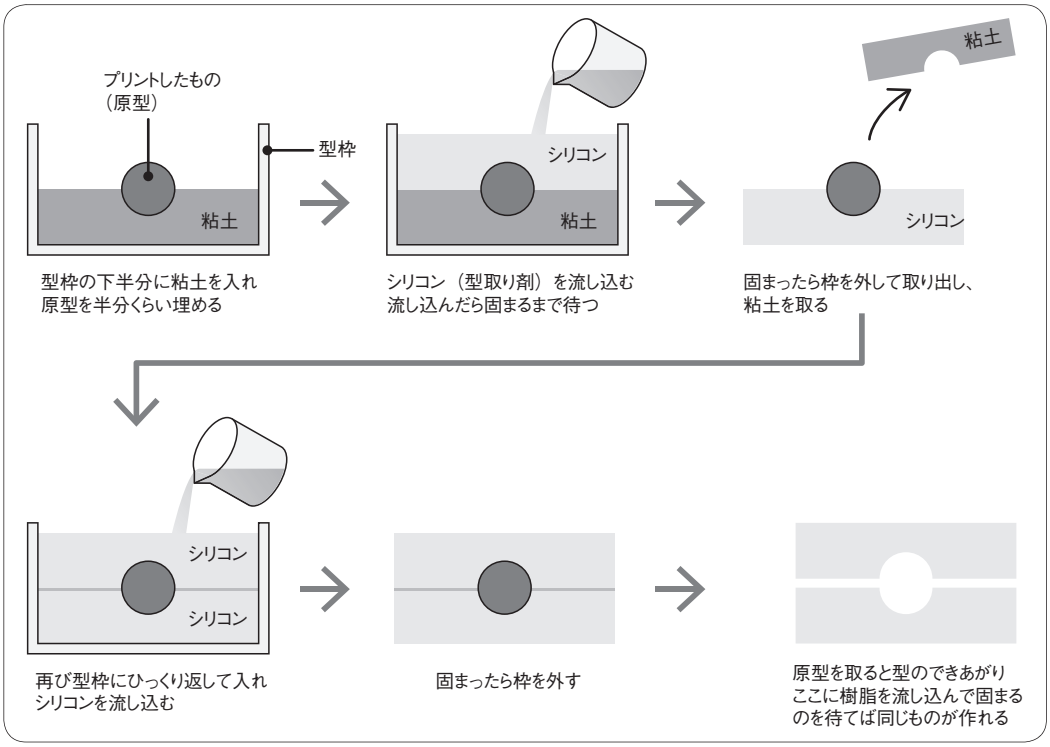
ようは、3Dプリンタというのは、出力するのに時間がかかる現状では、使いどころを選ぶけれども、使いどころによっては革新的な製造方法になり得る装置と筆者は考えています。

3Dプリンタによる元型作り

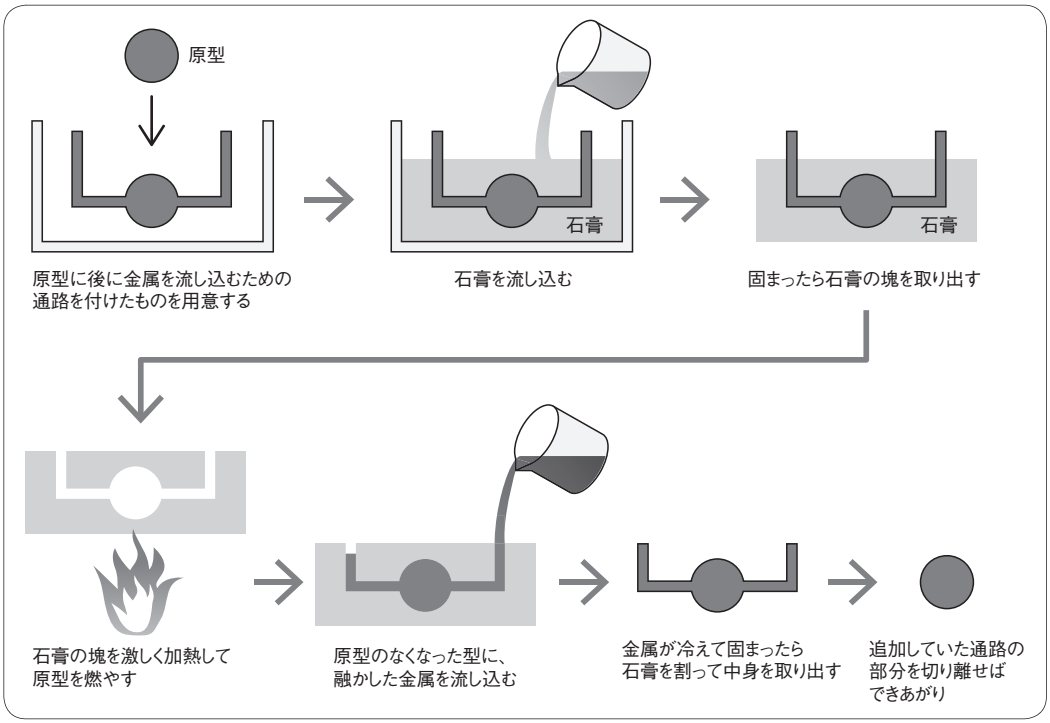
先ほどは、成果物を1個1個3Dプリンタで出力する例を挙げましたが、3Dプリンタで出力したものを原型(げんがた)として使い、レジンキャストなどの方法で少量の量産をするという方法も考えられます。レジンキャストというのは歯医者などでも使われている方法なのですが、原型をシリコーンゴムなどで型取りして作った型(原型と凹凸が逆になり、雌型とも呼ばれます)に樹脂を流し込んで固めて樹脂成形するという方法です(**図2**)。

型を作るというのが少し手間ですが、小規模な工業生産にも使われている方法ですし、ガレージキットなどを作るホビーユーザにも使われている方法です。レジンキャストをするために必要な材料は模型店やDIYショップでも売っていますので、同じ物を数十〜数百個作りたいときにはこういう方法もあります。

▼図2　レジンキャスト

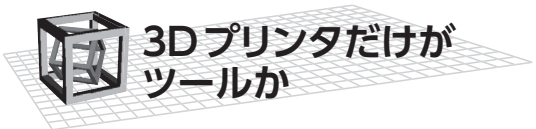


▼図3　ロストワックス法



ほかにも、ロストワックス法という手法を使えば、ホビー用3Dプリンタで金属部品を作ることもできます。最近、ブログの記事[注2]で知ったのですが、あまりにも印象的だったので紹介します。レジンキャストではシリコーンゴムで型取りをしましたが、ロストワックス法では鋳物砂や石膏を使って型取りします。石膏が固まったら、3Dプリントが埋まったまま型を高温で焼きます。すると中のPLA(ポリ乳酸；3章で詳しく解説)で作った原型は燃えてしまい、雌型ができあがります(**図3**)。ロストワックスは、ワックスをなくすことからこの名前が付いているのでしょうが、この方法はPLAをなくすためにロストPLAとして記事では紹介されていました。この型に融かした金属を流し込んで固め、型を割って中身を取り出すと金属部品が作れます。

ロストワックス法は、非常に古典的な鋳造技術なのですが、1つ作るたびに型を壊さなければならないのが難点です。



3Dプリンタだけがツールか

ところで、パソコンで作ったデータを物に出力する方法は3Dプリンタだけなのでしょうか。3Dプリンタと同様に、我々に比較的身近な加工機は、レーザー加工機とCNC(Computer Numerical Control；コンピュータ数値制御)ミリングマシンでしょう。

レーザー加工機

レーザー加工機は、高出力のレーザー光線で素材を燃やしたり融かしたりして切断する装置です(**写真3**)。最近のハンコ屋さんはレーザー加工機で印鑑を彫っていますが、基本的にレーザー加工機は3Dプリンタと違いZ軸方向(高さ方向)の調整が難しいです。また、よくある誤解なのですが、私たちが触る機会があるような、一般家庭に置けるくらいのサイズのレーザー加工機では、金属を切ることはできません。しかし、木や樹脂の板を手軽に、正確に切ることは得意です。レンズでレーザー光線を絞っている都合で焦点距離などがあり、あまり分厚い素材を切ろうとすると切断面が斜めになってしまうというのも注意が必要です。

レーザー加工機でできることは基本的に板状のものを切ることですので、平面を組み合わせて立体物を作ったりするために使われています。最近では、比較的手の届く30～50万円くらいの価格帯のレーザー加工機も出てきていますので、レーザー加工機を持つということも現実的になってきつつあります。

レーザー加工機を家に置くときの問題点は、素材を燃やしたりするために出る煙です。とくに樹脂板を加工したりすると相当な匂いが出ますので、集煙脱臭装置が必須です。レーザー加工機を購入する場合には、集煙脱臭装置のことも忘れずに検討するようにしましょう。

CNCミリングマシン

CNCミリングマシンは、コンピューター数値制御(CNC)された回転刃で素材を削って加工します。回転刃というのは、歯医者さんが歯を削るときに使うようなものです。

大型のCNCミリングマシンであれば、金属の加工も楽にこなすのですが、一般家庭に置けるくらいのサイズのミリングマシンでは、アルミの加工くらいが限度になります。細かい加工を行おうとすると細い回転刃を使う必要が出てくるので、少しずつしか削ることができず、加工時間(加工に要する時間)が長くなります。もち

▼写真3　レーザー加工機

注2) http://3dtopo.com/lostPLA/

ろん、細い刃は強度が下がりますので、少しずつ素材を削らなければ刃が折れてしまうという都合もあります。大きく削ろうとすると太い回転刃を使うことになり、加工時間が短くなりますが微細な加工ができなくなります。そのため、太い刃でだいたいの形を作ってから、細い刃で細かく削っていく、といったテクニックを使ったりします。CNCミリングマシンには音がうるさいという欠点があります。集合住宅で使うにはいろいろと気を遣うことになるでしょう。

ホビー用として有名なCNCミリングマシンにはローランドディー.ジー.のiModela(**写真4**)があります。これは8万円弱と比較的手軽に買えるマシンで、筆者のまわりにも個人で購入した人が数人います。

3Dプリンタのデータ

ここまで加工機の話を中心にしてきましたが、3Dプリンタを動かすには、3次元CADで3次元データを用意する必要があることは言うまでもないでしょう。ですが、この3D CADを使うにはそれなりに習熟が必要です。最近では無償で使え、かつ操作が比較的容易なものも出てきています。第3章では、TinkerCADという3D CADを使ってモデリングを解説しています。

STLフォーマットとGコード

3Dプリンタで出力するための3次元データですが、STLという形式が一般的です。STLは、最初に紹介したスリーディー・システムズ社によって開発されたファイルフォーマットです。STLファイルは、三角形のポリゴンの集合体でオブジェクトを表現するようになっており、カーブ形状などを表現することはできません。多くの3DプリンタではSTLファイルを読み込んで、3Dプリンタをどう動かすかという情報(Gコード)への変換を行います。このSTLファイルからGコードへの変換を行うソフトウェアはスライサーと呼ばれています。

たいていの3Dプリンタのコントロールソフトでは、複数種のスライサーから、使用するものを選ぶことができるようになっています。スライサーのアルゴリズムによって、得意不得意というか癖があるようで、同じSTLファイルを処理させても3Dプリンタで出力される結果が異なってきます。

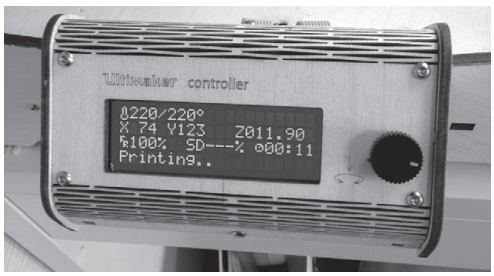
STLファイルからスライサーによって変換されたGコードですが、これが3Dプリンタに搭載されているマイコンに逐次送られ、3Dプリンタが出力をします。このため、たいていの3Dプリンタでは出力している間はパソコンを立ち上げ続けておかなければいけません。プリント中にパソコンがスリープしてしまうと印刷が止まってしまいますし、パソコンの負荷が高かったりすると処理落ちして出力の品質が落ちたりします。こういった問題を避けるため、いくつかの3DプリンタではSDカードなどのFlashメモリにGコードを保存しておき、プリンタがSDカードから直接データを読み込んで印刷するという、スタンドアローンな運用ができる工夫がなされているものもあります(**写真5**)。

3Dモデルの公開サイト

さて、3Dプリンタでちょっとしたものを出し

▼写真4　iModela

▼写真5　Ultimakerのコントローラ



てみたい、でも3D CADを習得してモデリングをするのは道のりが長そうだという方はどうすれば良いでしょう。実は、自分が作ったモデルを公開できるWebサイトがあります。Thingiverse[注3]です。

Thingiverseは、Makerbot社が運営しているコミュニティサイトで、数多くの3DデータがSTLや3D CADのデータ、場合によってはGコードといったデータ形式でアップロードされています。モデルデータのライセンスは、クリエイティブコモンズで表記されており、利用条件がわかりやすいのも好感が持てます。アップロードされているデータも3Dプリンタのパーツを始めとして、日常役に立ちそうなもの、iPhoneのケースなどガジェット関連、おもしろいオブジェなど幅広く、Thingiverseを見ているだけでもけっこう楽しめます。

3Dスキャナ

ところで、3Dデータを作る方法は3D CADだけではありません。3Dプリンタは話題になっていますが、3D CADでつまずく人も多いようで、データを作らずに直接廉価な3Dスキャナで複製物をキャプチャするという手段も徐々に提供されはじめています。3Dスキャナも従来は高価な装置だったのですが、Makerbot社がDigitizerという製品を開発していたり、3D CAD大手のオートデスク社がiPhone、iPadやパソコンで写真をたくさん撮ると3Dモデルにしてくれる“123D Catch”というアプリケーションの無償配布を開始したりと、徐々にさまざまな動きが出てきています。

3Dプリントサービス

3D CADを使いこなせるようになったけれども、手元の3Dプリンタで出力したモノでは品質に不満がある、といった場合にはshapeways[注4]というサービスがあります。shapewaysはデータを送ると3Dプリントして送ってくれるサービスを提供しています。彼らが持っている3Dプリンタはフルカラー出力できたり、金属などホビー用の3Dプリンタでは扱えない素材で出力できたりと、かなりの優れものです。また、自分の作ったデータをshapewaysにアップロードしておき、ほしいと思った人に売ることができるマーケット機能も提供されています(3Dプリントと送付はshapewaysがやってくれる)。また国内にも3Dプリントサービスが増えてきています。

GitHubも3Dプリンタに夢中

本誌読者にはおなじみのGitHubの人達も3Dプリンタに夢中のようです。彼らのオフィスにはMakerbotのReplicator 2があるそうなのですが、この3Dプリンタは彼らのHubot[注5]と呼ばれる社内システムに接続されているそうです。前述のThingiverseなどのURLを指定すると、3Dデータをダウンロードし、スライサでスライス、そしてReplicator 2での出力を始めるといった機能があるようで[注6]、このスクリプトもGitHubで公開されています[注7]。

GitHubにも訪れている3Dプリンタブームの影響はこれにとどまりません。2013年4月にはGitHubにアップロードされているSTLファイルが、ブラウザから3Dで見られるようになっています[注8]。

いつの日か、筆者も離れたところにある3Dプリンタをこんな方法で効率良く動かせるようにしたいものです。SD

注3) http://www.thingiverse.com/
注4) http://www.shapeways.com/
注5) http://hubot.github.com/
注6) このようにリモートから3Dプリンタの操作ができるようにしたいと思っている人は多いようで、OctoPrintという3DプリンタのWebインターフェースも存在します。http://octoprint.org/
注7) https://github.com/github/hubot-scripts/blob/master/src/scripts/make_me.coffee と https://github.com/make-me/make-me/
注8) https://github.com/lorennorman/octocat-3d/blob/master/stl/octocat_head.stl

理論から実践まで
3Dプリンタが未来を拓く理由

3Dプリンタのしくみ

第2章

坪井 義浩（つぼい よしひろ）　ytsuboi@gmail.com
Twitter:@ytsuboi

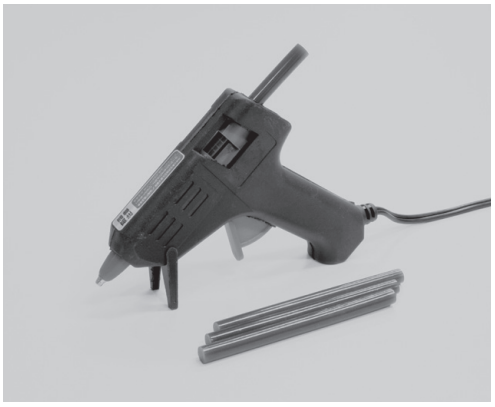
前章では3Dプリンタを取り巻く状況を紹介してきましたが、本章では3Dプリンタについて説明をします。3Dプリンタと一口に言っても、立体物を出力するにはさまざまな方法があります。ここでは、代表的ないくつかの方法を紹介します。

熱溶解積層法

熱溶解積層法（Fused Deposition Modeling method；FDM法）という方法は、デスクトップ3Dプリンタで最も採用されている方法です。FDM法は、熱可塑性樹脂（加熱すると柔らかくなり、冷却すると固まる性質の樹脂）を加熱し、細いノズルの先端から押し出して積み重ねていくことで造形します。

ホームセンターなどで、ホットボンド（またはグルーガン）という工具が売っているのをご存じでしょうか（**写真1**）。ホットボンドは、熱可塑性の樹脂の棒をヒーターに押し込み、柔らかくなって工具の先から出てきた樹脂を使って接着を行うという工具です。

▼写真1　ホットボンド



実際の3Dプリンタの構造

ここではホビー向けプリンタで使われている構造や用語を交えつつ、Ultimaker[注1]を例に、もう少し詳しくしくみを説明します。

ステッピングモーター

3Dプリンタではプリンタヘッドの代わりに、小さなホットボンドの先端をコンピュータ制御で位置合わせし、ボンドの棒をモーターで押し出しているにすぎません。

3Dプリンタのモーターにはステッピングモーターと呼ばれるものがよく使われます。ステッピングモーターは、モーターの軸を電気信号に合わせて一定角度ずつ回すことができるモーターです。たとえばありふれた200ステップのステッピングモーターですと、1周が200ステップになっているので、1ステップ動かすと1.8°ずつモーターが回ります。簡単な制御回路で位置を割と正確に出すことができるので、こういったメカには多用されています。

ステッピングモーターの回転運動を直線方向の運動に変換する方法にも定石があります。長いネジをステッピングモーターの軸に繋いで回し、ネジを通したナットを回らないように持つ

注1）http://www.ultimaker.com/

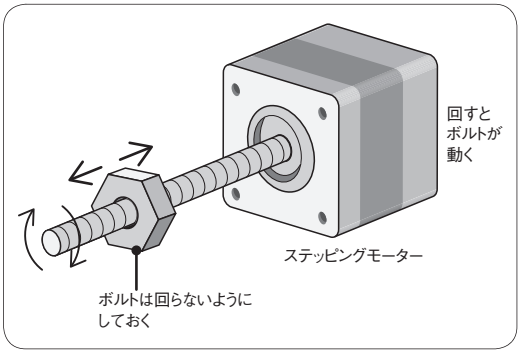
とネジの上でのナットの位置は変化します(**図1**)。これは送りねじ機構などと呼ばれる方法で、古くはフロッピーディスクドライブのヘッドの移動に使われていました。

それ以外の方法としては、ベルト駆動方式があります。タイミングベルト(歯付きベルト)と呼ばれるものを、ステッピングモーターに取り付けたタイミングプーリーと呼ばれる滑車で動かして、直線上の位置に変換するものです(**図2**)。ベルト駆動方式は家庭用のインクジェットプリンタのヘッドを左右に動かす機構などでも使われています。

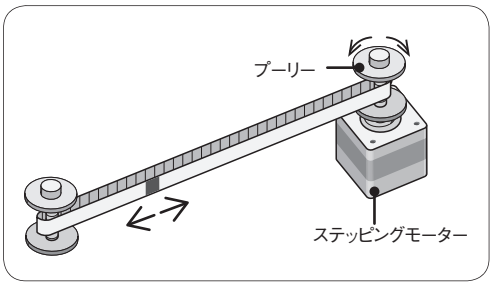
フィラメント

まず材料に使う樹脂は、フィラメントと呼ばれる直径が1.75mmや3mmの細長い棒状になっています。フィラメントの樹脂にもいくつかの種類があります。よく使われるのはABS(Acrylonitrile Butadiene Styrene;共重合合成)樹脂とPLA(PolyLactic Acid;ポリ乳酸)樹脂です(3章でも詳しく説明しています)。ABSはエンジニアリングプラスチックと呼ばれる樹脂の1つで、強度や耐熱性に優れているという特徴があります。一方、PLAはポリ乳酸という、生分解性プラスチックとかバイオプラスチックと言われている樹脂です。PLAのフィラメントのほうがABSのフィラメントよりも低い温度で融けるのですが、融けているときに粘り気があります。**写真2**のものはPLAの直径が3mmでリールに巻き付けて売っているタイプです。フィラメントには色も豊富にあります。

▼図1　送りネジ機構



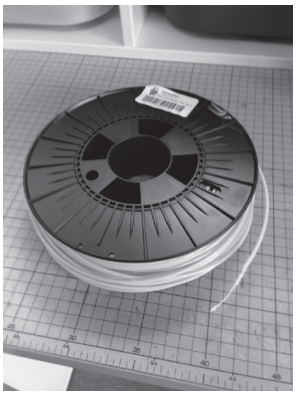
▼図2　ベルト駆動



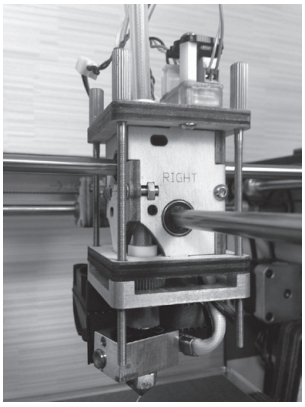
エクストルーダ

このフィラメントの先端を、プリンタのエクストルーダ(**写真3**;プリンタで言うところのヘッド)のフィラメント挿入口に入れると、3Dプリ

▼写真2　フィラメント



▼写真3　エクストルーダ



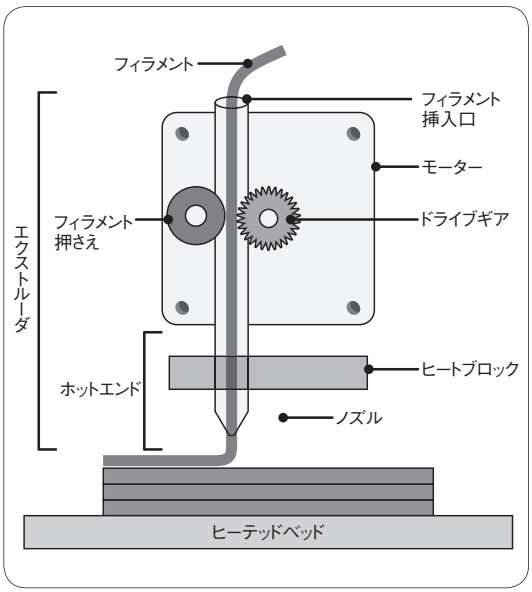
ント時にモーターのドライブギアが回転し、フィラメントはヒーターとノズルの方向に押し出されます。押し出されたフィラメントはヒーターで熱され、柔らかくなったところを細いノズルから押し出されてテーブルの上に出てきます。

Ultimakerでは、フィラメントの送り機構がホットエンドと別の場所にあるBorden式を採用していますが、RepRapやReplicatorなどエクストルーダと送り機構が一体のタイプも多く見かけられます。**図3**は、エクストルーダと送り機構が一体のタイプです。

フィラメントの温度管理

先ほど、3Dプリンタはコンピュータでコントロールされているホットボンドのようなものと書きましたが、普通のホットボンドとは異なり、3Dプリンタのホットエンドは温度もコンピュータによってコントロールされています。ホームセンターで売っているようなホットボンドのヒーターは温度管理されていないので、長くヒーターに通電をしていると熱くなり過ぎてボンド(樹脂)がノズルから垂れてきてしまいます。3Dプリンタでそのようなことが起きると正常な出力は得られなくなりますので、使用するフィラメントに応じた温度になるよう、ホットエンドの温度はコンピュータで管理されます。面倒なのは、ABSのフィラメントなら何でもこの温度というのが決まっているわけでもないということです。ノズルから垂れてきたりホットエンドに詰まらず、造形するのに十分な柔らかさになるような温度を見いだす必要があります。

▼図3　エクストルーダの構造



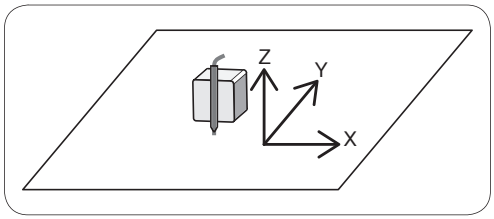
ノズルと積層ピッチ

ノズルの先端の穴の大きさ(ノズル径)には、0.5mmや0.35mmといったように種類があります。Ultimakerの場合、ノズル径は0.4mmになっています。ノズル径と積層ピッチは直接一致しません。たとえば、0.35mmのノズルで積層ピッチを0.3mmとすることは十分に可能です。ただし、たとえば直径0.5mmで出てきた樹脂を積層ピッチ0.25mmにしたりすると、押しつぶし気味になりますのでXY方向の解像度が下がることになります。ノズル径は大きいほうが出力したものが頑丈になりますが、出力の結果がノズル径の細いものより粗くなります。

積層方式の軸

積層方式の3Dプリンタは、X、Yの2軸で構成される平面に1層ずつ樹脂フィラメントをプリントし、Z軸方向に全体を下げていきます(**図4**)。この動かし方は各3Dプリンタごとに特色がありますので、調べてみるとおもしろいと思います。UltimakerはX、Y軸に取り付けたエクストルーダをベルトとプーリーで動かし、Z軸をネジ送り機構でテーブルの上げ下げを行いま

▼図4　積層方式のX、Y、Z軸



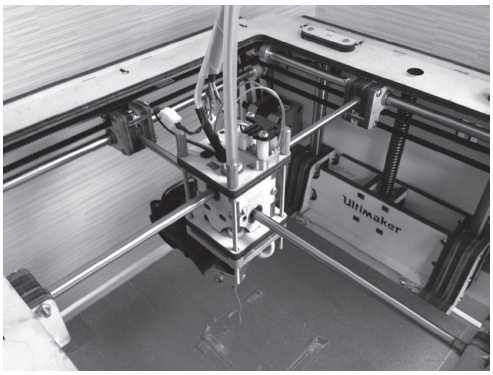
す(**写真4**)。

ヒーテッドベッド

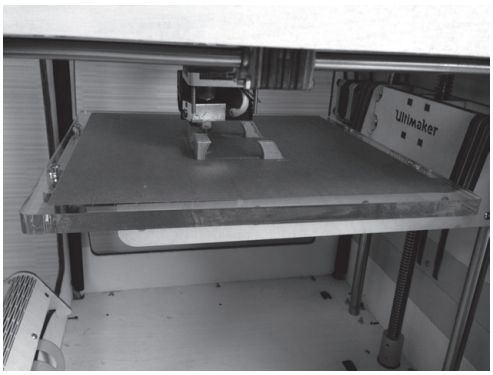
ヒーテッドベッドはヒーターで熱されているテーブルです。普通の平板ですと出力した樹脂が冷えて収縮することで板から剥がれてしまうことがあります。とくにABSでは冷えたときの収縮が顕著です。出力したものが剥がれないよう、ヒーターで熱しているわけです。

Ultimakerのテーブルにはヒーターが付いていません。**写真5**のようにアクリル板がテーブルになっており、テーブルには青いテープを貼っています。冷えたときに収縮の少ないPLAを使用していることもあり、出力したものがテーブルに貼り付きやすく、テープを貼って保護してやらないと完成品を剥がすときにテーブルが傷んでしまうためです。

▼写真4　X、Y、Z軸



▼写真5　テーブル

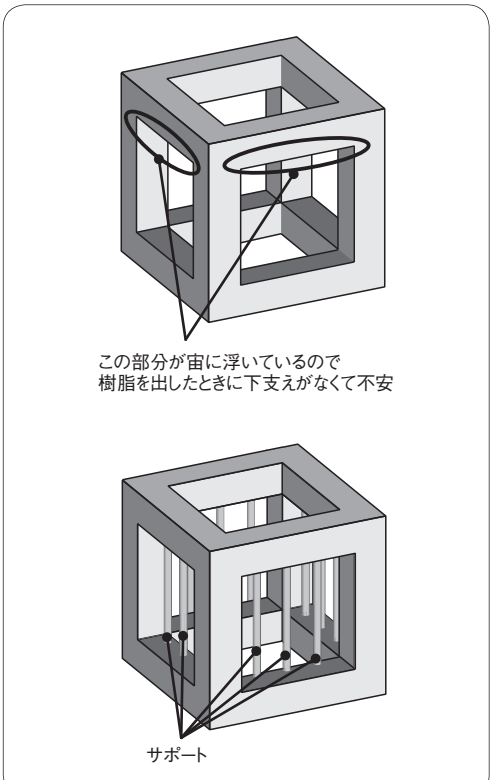


サポート

ところで、**図5**のような形状のものをプリントしようと思ったときにはどうすれば良いでしょうか。このような形状では、宙に浮いたところに樹脂をプリントしなければならず、垂れてこないかが心配です。実は多少であればあまり垂れてこないのですが、こういう部分をきっちり造形するにはサポートと呼ばれる下支えを作っておきます。このサポートですが、プリントしたあとに簡単に取り除けなければ困ります。エクストルーダが1つしかない装置の場合、出力するものと同じフィラメントでサポートを作ってやります。出力し終えたあとに、切って捨ててしまう支えを作っておきます。一般的にサポートは後で切り離しやすいような形状で作っておきます。

エクストルーダが複数あればもっと手軽な方

▼図5　3Dプリントしにくい形状



法があります。あとで取り除きやすい(たとえば水溶性)材料のフィラメントで宙に浮く部分の支えを印刷してしまうのです。こういったサポート材に使われるフィラメントにはPVA(ポリビニルアルコール)などがあります。たとえばMakerbot社のReplicatorはエクストルーダが2つあるため、同社のストアで販売されているPVAのフィラメントを使ってサポートを作成すれば、出力後に水に浸けるだけで除去できます。

◆◆◆

この熱溶解積層法は、米Stratasys社を創業したS. Scott Crump氏によって1988年に特許が取得された方法です。Stratasys社は世界最大の3Dプリンタメーカーで、昨年末、イスラエルのObjet社というインクジェット型の3Dプリンタメーカーを買収したというのがニュースになりました。

ホビー向けの3Dプリンタのほとんどが、この熱溶解積層法を採用しています。オープンソースであるRepRapもそうですし、Makerbot社の製品もこの方式です。また、3D Systems社もホビー向けにCubeや後継のCube Xという熱溶解積層法のプリンタを販売しています。熱溶解積層法の3DプリンタでABS樹脂を使って出力するときには、プリントする部分を覆って熱して保温するのが良いと知られています。しかし、このような機能のついたホビー向け3Dプリンタがほかにないのは、Stratasys社の米国特許[注2]が現在も有効であるためです。

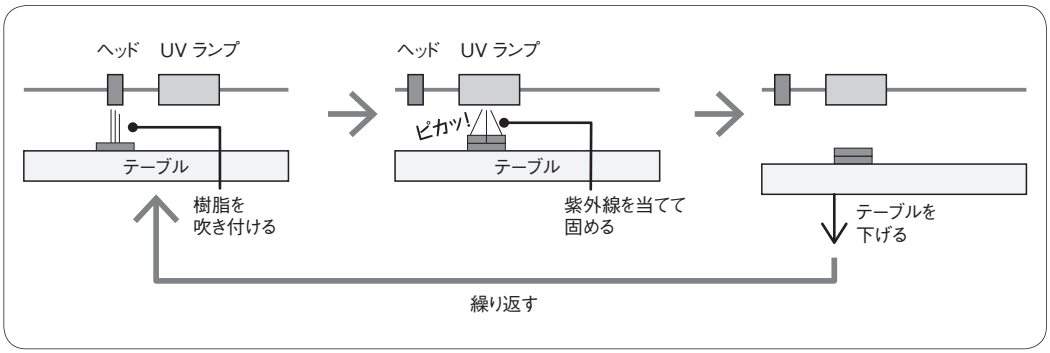
インクジェット法

インクジェット法(**図6**)は、先述のObjet社が採用している方式です。従来の紙に印刷するインクジェットプリンタと同様に、ヘッドから素材の細かい粒子や接着剤を吹き付けてそれを積み重ねて造形します。Objet社の3Dプリンタの場合、紫外線を当ててUV硬化性樹脂(紫外線で固まる樹脂)で固めるようです。

インクジェット法の優れている点は、インクジェットプリンタと同様に細かい粒を吹き付けるため、積層ピッチが16 μm(0.016mm)といった具合に熱溶解積層法と比べると相当に精細な造形ができることです。また、Objet社の3Dプリンタは複数のヘッドを搭載していて、それぞれ異なる樹脂を吹き付けることができます。この機能を使うと、たとえば車輪を出力するときに、ホイールとタイヤを異なる素材にし、さらにはタイヤをゴムのような弾力のある樹脂で出力するといったことができます。あるいは、透明の樹脂の中に色付きの樹脂の部品が入っているといった出力も可能で、展示会などでの同社の展示物には人体模型があります。

注2) High temperature modeling apparatus(USPTO #6,722,872)

▼図6　インクジェット法



粉末積層法

粉末積層法には大きく分けて2種類の方法があります。

粉末固着積層法

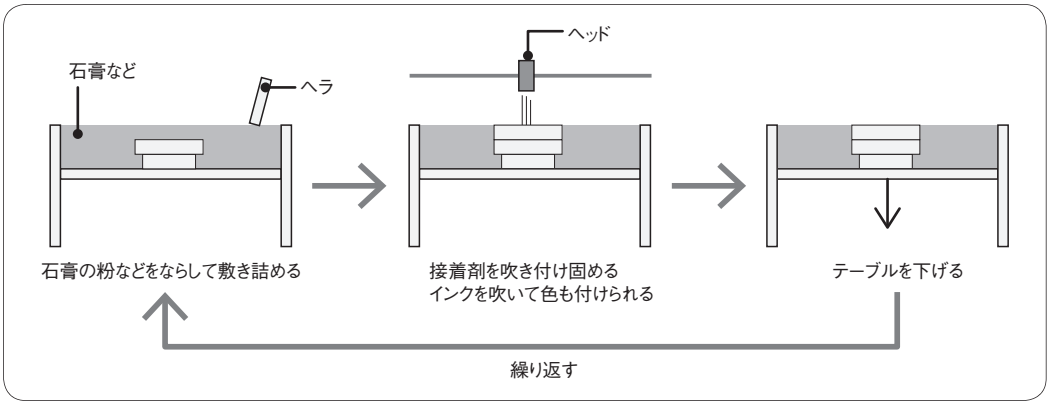
1つは粉末固着積層法と呼ばれており、石膏や樹脂などの素材の粉を敷き詰め、上から接着剤を吹き付けて断面を印刷するということを繰り返して造形する方法です(**図7**)。この方法を採っている3Dプリンタは米Z CorporationのZプリンタが有名で、Zプリンタには着色機能が備わっているため、カラーの3D出力が得られます。

粉末固着積層法は、最終的に粉に埋まっている出力を掘り出すようにして取り出すのですが、そのためにサポートが必要ありません。捨てるサポートを造形しなくて済むので、無駄な材料費が発生しづらいことからコストパフォーマンスが良いとされています。ただし、材料の粒がそんなに細かくないため、出力結果は少しザラザラした、粒を感じるような仕上がりになります。

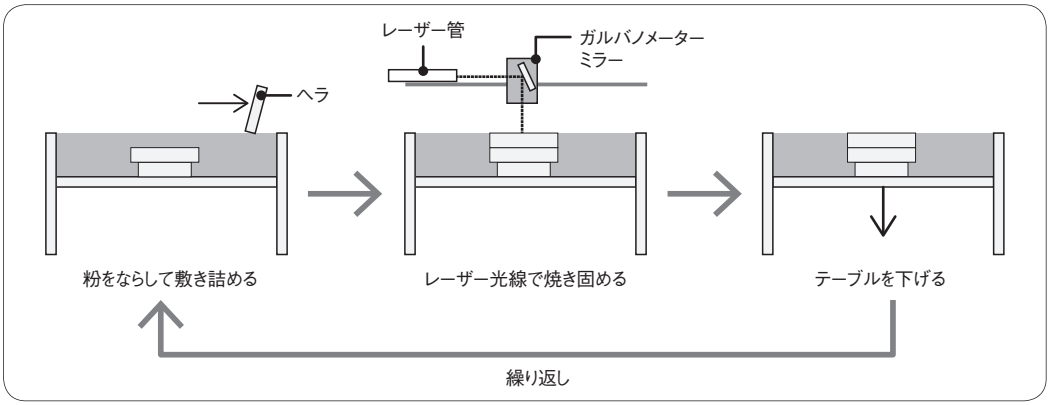
粉末焼結積層法

もう1つは、粉末焼結積層法と呼ばれるものです。この方法は、先ほどの石膏の粉を接着する代わりに、金属や樹脂の粉をレーザー光線で焼き固めるものです(**図8**)。レーザー光線を任意の場所に当てるには、ガルバノスキャナという機構を使うのが一般的です。ガルバノスキャナは、先端に鏡がついたガルバノモーターと呼

▼図7　粉末固着積層法



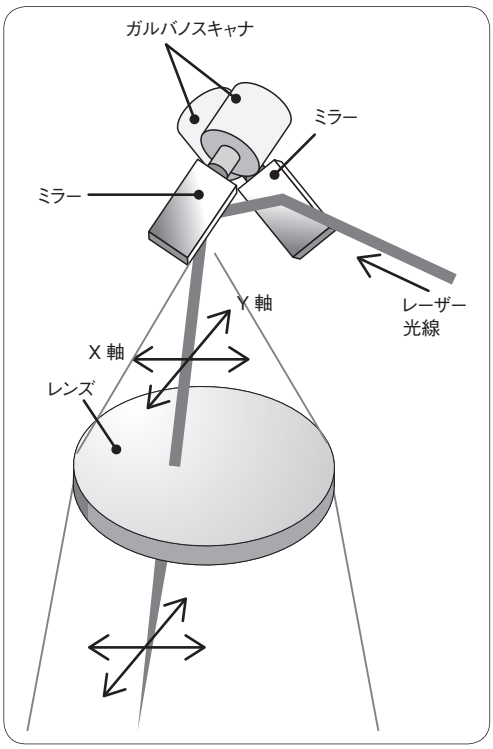
▼図8　粉末焼結積層法



ばれる反射角を変えられるモーターを複数組み合わせて構成されています。このようなケースでは2次元(XY)方向だけで良いので、ガルバノモーターは2つで済みます(**図9**)。

ほかの方法と異なり、金属粉が使えるため、金属の3Dプリントができるという特徴があります。この方法を採用しているメーカーは、独EOS社や米3D Systems社です。なお、3D Systems社は昨年初頭にZ Corp.の買収をしています。

▼図9　ガルバノスキャナ



光造形法

光造形法(Stereolithography。SLやSLAなどと略される)は、液体の光硬化性樹脂に紫外線などの光を当てて硬化させるということを繰り返して造形する方法です。この方法が最も初期に開発された3Dプリントの手法と言えるでしょう。光造形法は微細な出力をできますが、素材が光硬化性樹脂に限られます。

図10はレジン(樹脂でできた台)から出力を少しずつ持ち上げる方法の例ですが、逆に上からレーザー光線を照射して出力を少しずつ沈めていくという方式もあります。また、光造形法でも任意の場所にレーザー光線を照射するためにガルバノスキャナが使われているケースもあります。

光造形法の特許出願を初めて行ったのは日本人です。1980年に名古屋市工業研究所に勤めていた小玉秀男氏が特許出願をしています。小玉氏は現在弁理士なのですが、自身の審査請求を行っておらず、審査請求期限が切れたあとになって3D SystemsのCharles W. Hull氏による特許を知ったそうです。

ホビー向けの光造形法を採用した3Dプリンタとしては、FormlabsのForm 1という3Dプリンタが昨年注目を集めました。このForm 1は光硬化性樹脂にレーザー光線を当てて硬化させ、それを積層して造形する3Dプリンタです。Form 1はKickstarterというクラウドファンディング

Column ガルバノスキャナ

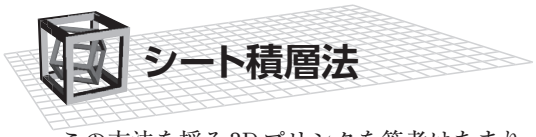
余談ですが、このガルバノスキャナという方法は意外と多くの場所で使われています。壁や煙などにレーザー光線を当てて絵を表示するレーザープロジェクタという装置があります。こういった装置の中にはガルバノスキャナが入っており、レーザー光線を照射する場所を動かして絵を描いています。基本的にレーザー光線は点なのですが、人間の目の残像効果を利用して絵を見せているわけです。

また、半導体などのパッケージの上などに、型番などが小さい文字で彫刻がされているのを見かけることもあります。あのような彫刻も、ガルバノスキャナの入ったレーザー加工機を使って彫刻しています。

(一般の人々から資金を募集すること)を行うWebサイトで、1ヵ月間で10万ドルを目標に資金調達を試みたのですが、3時間もかからずに目標額の調達に成功して注目を集めました。最終的にFormlabsは30日の間に294万ドルほどの調達に成功しています。

その後、Form 1は残念なことに訴えられるという形で再び注目を集めました。Formlabsと先述のKickstarterを特許侵害で訴えたのは3D Systemsです。3D SystemsによるとForm 1は同社の米国特許[注3]の請求項1および23を侵害しているとのことです。昨年11月のことですのでいまだ結果は出ていませんが、ホビー向け3Dプリンタの最大の障害は従来の3Dプリンタメーカーが持っている数々の特許という懸念が顕在化した事件であり、多くの人が行方を見守っています。

また、レーザー光線を使う方法のほかに、プロジェクタを使って樹脂を硬化させるという方法を採用している機種もあります。こちらもKickstarterで資金調達をしていたB9Creatorという3Dプリンタですが、レーザー光線の代わりにオフィスや家庭で使うような一般的な1,024×768ピクセルのプロジェクタの光を下から当てるそうです。プロジェクタはそんなに高速に点滅させられないからか、レジンのプールの下にはシャッターが設けられているというおもしろいしくみです。こちらも当初の目標である5万ドルを上回る51万ドル強の資金が集まりました。



シート積層法

この方法を採る3Dプリンタを筆者はあまりみかけませんが、シート積層法もおもしろい方法です。この方法では、紙や樹脂のシートを貼り合わせて立体を作ります。インクジェットプリンタなどで造形物の断面の色を刷り、断面の輪郭に合わせてシートを切り抜きます。切り抜いたシートに接着剤を塗って貼り合わせていけば立体物が作れるというしくみです。

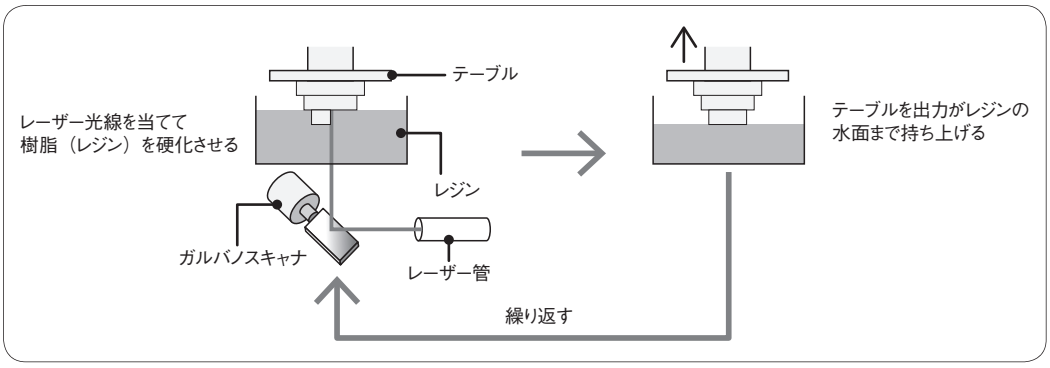
Mcor Technologies社のMcor IRIS 3D color printerなどがこの方法を採用しており、このプリンタに内蔵されているインクジェットプリンタはエプソンの市販品とのことです。

◆◆◆

ここまで、さまざまな方法を紹介してきましたが、ホビー向けの3Dプリンタで採用されている方法のほとんどは熱溶解積層法です。近年、光造形法を採用したものも登場しつつあり、注目を集めていますが、まだまだ熱溶解積層法を採用したプリンタが主流といった状況がしばらく続きそうです。SD

注3) Simultaneous multiple layer curing in stereolithography (USPTO #5,597,520)

▼図10　光造形法



理論から実践まで
3Dプリンタが未来を拓く理由

3Dモデルを作ってみよう

第3章

工房Emerge＋　山田 斉（やまだ ひとし）
http://www.emergeplus.jp　Twitter:@emergeplus

3Dプリントをするには、まずは3Dモデルのデータを作らなくてはなりません。もちろんネット上でダウンロードできる3Dデータをそのままプリントするのもありでしょう。しかし、オリジナルな3Dモデルを作れれば、本当の3Dプリントを楽しむことができます。本章ではその足がかりとして、無料で利用できる3D-CADソフトをいくつか紹介します。また実際に、チュートリアル形式で3D-CADを使ってiPhone 5スタンドをモデリングしてみます。

3D-CADとは

ほとんどのみなさんは、3D-CADソフト（以後3D-CAD）には馴染みがないでしょう。しかし、最近マスコミがさかんに取り上げて過熱気味になっている3Dプリンタは、この3D-CADがないと始まらないのです。何しろ3Dモデルのデータが作れないと3Dプリントできないのですから。

3D-CADというのは、コンピュータ上で自由に3次元形状のモデルを作るためのソフトです。たとえばパソコンを使ってペーパークラフトの型紙を印刷するには、Adobe Illustratorなどのドローイングソフトあるいは2D-CADを使って、あたかも2次元の紙に線を引くように図形を描いてプリンタで印刷します。これを3次元に拡張して、パソコン上で立体形状を作成できるのが3D-CADソフトです。

無料で利用できる3D-CAD

最近流行している廉価な3Dプリンタのサポートする3Dデータフォーマットは、STL（Standard Triangulated Language）ファイル形式がほとんどです。このSTL出力をサポートし、無料で使える高機能な3D-CADソフトはたくさん出始めてきました。とくにマルチプラットフォーム対応で初心者が始めやすく、比較的高度なスキルを必要としないものを選んでみました。本稿ではそのうちのいくつかを紹介します。ただし、このリストの中では、Linuxから利用する場合はブラウザ版を利用することになるでしょう。

なお、スタンドアロン版／ブラウザ版ともに、作成した3Dモデルをネット上に公開できる機能を持っています。ですからネットで公開されているほかの人の作った3Dモデルをそのまま、あるいは改変して使うこともできます。

スタンドアロン版

・Autodesk 123D Design[注1]

3D-CADの大御所、オートデスク社が提供している無料3D-CADです。リリースされて間もないのですが、基本的な3Dパーツを組み合わせたり、2次元図形からも3Dパーツを作り出せたりと、柔軟で直感的にデザインしやすいUIを持っています。Windows、Macだけでなく、簡易版としてiPad版もあります。STLファイル形式のインポートはできませんがエクスポートができます。残念ながら日本語版はなく、英語版を使うことになります。

注1）http://www.123dapp.com/design

・SketchUp Make[注2]

もともとはGoogle社が米Last Softwareから買収した、Google SketchUpという建築用3D-CADです。現在はTrimble社が買収して開発しています。操作法は、2次元画面を使って鉛筆でデッサンをするようなイメージのUIを持ち、3Dデザインで悩みがちな「3次元を描く」という難しさを独自の操作で容易にモデリングできるように工夫されています。

これまではSketchUpという名前でしたが、今年5月にアナウンスされたSketchUp8からSketchUp2013へのバージョンアップにともない(3DプリンタなどMaker市場の広がりを意識したのだと思いますが)、SketchUp Makeという名称に変わりました。日本語版もあります。STLは標準でサポートされていませんが、Trimble社が運営するExtention WarehouseからSTLプラグインをインストールすれば出力できるようになります[注3]。

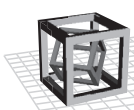
ブラウザ版

・Autodesk 123D Design[注4]

ブラウザでも123D Designを利用できます。こちらはスタンドアロン版ほど強力ではありませんが、それでもちょっとしたものであれば作成できる十分な機能があります。

・TinkerCAD[注5]

名前からして[注6]もおわかりいただけるように、子供でも扱えるように意識したUIのブラウザ3D-CADです。基本的な立体部品を、積み木のように組み合わせてデザインするという、かなり割り切った仕様になっているのが特徴です。それでも工夫すればかなり複雑なデザインができるパフォーマンスを持っています。実は今年3月末に運営を止めるとアナウンスされていたのですが、その後5月にオートデスク社が買収し、あらためて安定した運営が再開されました。



TinkerCADによる3Dモデリング体験

今回は前述の3D-CADの中から、TinkerCADを使ってチュートリアル形式で3Dモデリングを体験してみましょう。TinkerCADはUI的にもとてもシンプルですし、積み木を組み立てていくような操作でモデリングが完成しますから、初めて3D-CADにトライする方でもすぐに基本的な操作はマスターできるでしょう。

TinkerCADの使い方

動作環境

TinkerCADはブラウザアプリケーションですので、パソコンにインストールする必要はありません。WebGL対応のブラウザがあれば大丈夫です。現時点での推奨環境は次のとおりです。

- OS：Windows Vista以降、Mac OS X 10.6以降
- ブラウザ：Chrome10以降、Firefox4以降

今回はMacBook Airで、Mac OS X Mountain Lion(10.8.3)に入っているSafari(6.0.4)で実行しました。Safariは推奨ブラウザではないのですが、[環境設定]→[メニューバーに"開発"メニューを表示]をチェックし、メニューに新しく加わった[開発]→[WebGLを有効にする]をチェックすることで動作します[注7]。

アカウントの登録

TinkerCADを使うには、まずはブラウザで"https://tinkercad.com/"を開き(**図1**)、よくあ

注2) http://www.sketchup.com/products/sketchup-make
注3) http://extensions.sketchup.com/en/content/sketchup-stl
注4) http://apps.123dapp.com/design/
注5) https://tinkercad.com/
注6) Tinkerは下手な職人やいたずらっ子の意味。

注7) 筆者は普段Chromeを使っているのですが、マウスジェスチャープラグインが悪さをしてうまくマウス操作ができないので、しかたなくSafariを使っています。もし、うまく操作ができないときはマウスジェスチャーを無効にするのが良いと思います。

るWebサービスと同じように無料のアカウント登録をします。画面右上にある“Sign up for free account”ボタンをクリックしてください。登録内容は、名前・メールアドレス・パスワード・生年月日です(**図2**)。

レッスンを受けてみる

登録が終わると、はじめにオンラインレッスン画面になります(**図3**)。多くのレッスンがありますので、これらを一通りやれば、かなりTinkerCADの操作に慣れると思います。1つのレッスンを終えるのにそんなに時間はかからないので、手早く操作をマスターしたければレッスンはお勧めです。

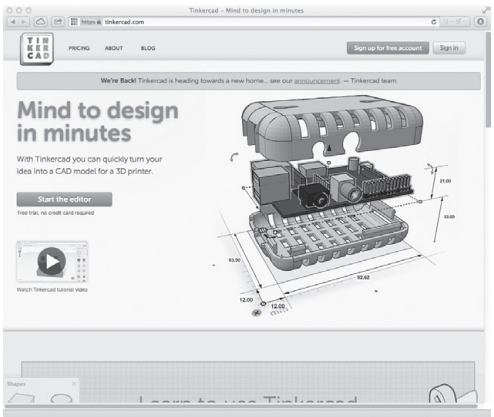
ダッシュボード(管理画面)

はじめのレッスンを試したあと、ダッシュボードの画面になります[注8](**図4**)。ダッシュボードでは自分の過去に作ったモデルやレッスンのサムネールが表示され、モデルの新規追加・修正・削除・公開／非公開、モデル名の変更などが自由にできます(サムネールにマウスを合わせると歯車アイコンが現れるのでこれをクリック)。また、自分のプロフィールも変更することができます。画面上部にメニューが並んでいます。ここでは説明を省きますが、このメニューからほかの人のデータにアクセスしたり、ビデオチュートリアルなども閲覧できます。

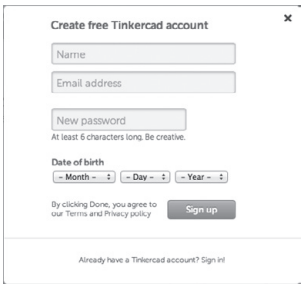
モデリング画面を見てみる

ダッシュボードの“Create new design”ボタンをクリックすると、いよいよデザイン画面になります(**図5**)。もし、前に作ったデータを修正

▼図1　スタート画面

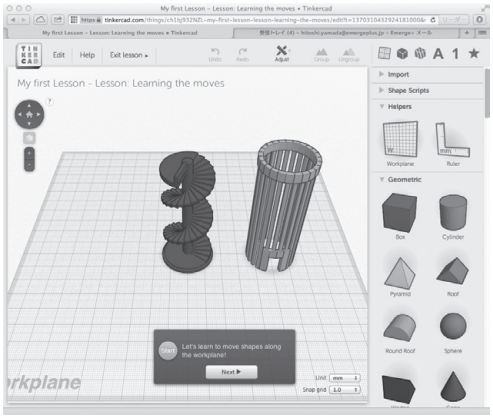


▼図2　アカウント登録画面

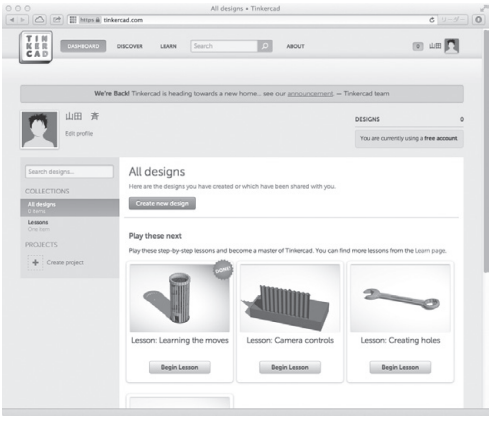


注8) レッスンになるのは初めて触ったときだけです。

▼図3　オンラインレッスン画面



▼図4　ダッシュボード画面



する場合は該当サムネールにマウスを合わせると“Tinker this”ボタンが現れますので、これをクリックします。

メニューの説明

では、操作を理解するために簡単に全体を見回してみましょう。デザイン画面は、メニュー、サイドバー、メインの大きく3つのレイアウトで構成されています。まずは1つめの上部のメニューをざっと見てみます(**図6**)。

一番左の“Design”はよくあるパソコンアプリの「ファイル」メニューに相当するもので、新規デザインを作ったり、保存したり、設定を変えたりというメニューです。3Dプリンタに出力させるには、“Download for 3D Printing”から“STL”ボタンをクリックします(**図7**)。また“Order 3D Print”では、クラウド3Dプリントサービスの PonokoやShapewaysなどに直接3Dプリントを発注できます。“Close”でダッシュボードに戻ることができます。ちなみにTinkerCADではデータ保存は自動ですのでとても便利です。

“Edit”、“Help”、“Undo”、“Redo”はお馴染みの機能ですので説明の必要はないでしょう。

“Adjust”は複数のパーツのレイアウトを整列してくれる“Align”とパーツ反転の“Mirror”があります。

“Group”はとても重要なメニューで、複数のパーツを1つのパーツにまとめてくれるだけでなく、あるパーツ形状でカットや穴あけするときにも使います(**図8〜10**)。つまり、3Dモデリングにおけるブーリアン演算の論理和と論理差を実現する機能です。“Ungroup”はそれを再び分解し、元に戻します。

一番右側に並んでいる5つのアイコンですが、これらはサイドバーの図形作成アイテムにクイックアクセスするためのショートカットアイコンです。実際にクリックしてたしかめてみてください。

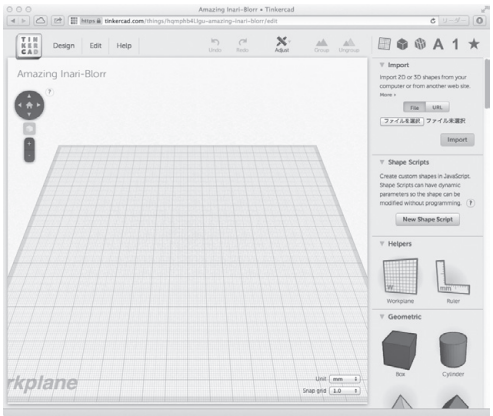
サイドバーの説明

続いて2つめのサイドバーに移ります。

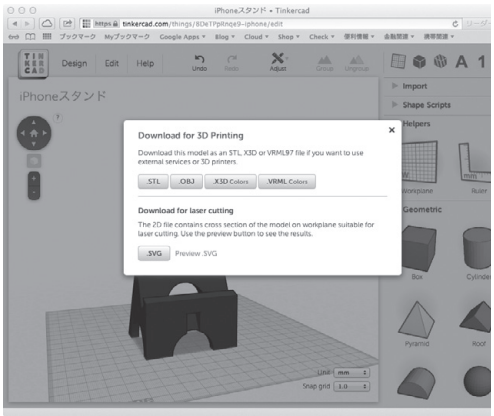
Importメニュー

一番上にある“Import”は、別のSTLなどのモデルをインポートする機能です。ローカルファイルだけでなく、インターネット上のモデルもインポートできます。

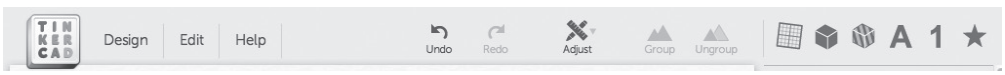
▼図5　デザイン画面



▼図7　STLダウンロード画面



▼図6　メニュー



Shape Scriptメニュー

次の“Shape Scripts”は、TinkerCADのモデリングを強力に拡張する機能で、オートデスク社による買収後、無料アカウントにも解放されたものです。これはJavaScriptを使って自由に図形を定義／登録し、それをモデリングに利用できるようにする高機能なツールです。ここではスクリーンショットと詳細の説明は省きますが、一度トライしてみるとおもしろいと思います。

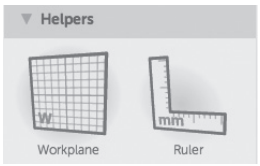
Helpersメニュー

“Helpers”は3Dモデリングをするにあたって、とても重要なツールです(**図11**)。ですので、少し詳細に説明します。

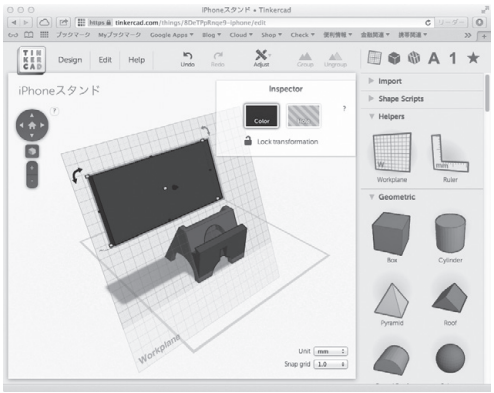
・Workplane

“Workplane”は2次元のディスプレイでの3次元のモデリング作業をとても容易にしてくれる強力なツールで、実際に何度も利用することになるでしょう(**図12**)。モデルのある面を基準に設定したWorkplane平面に3Dパーツを拘束す

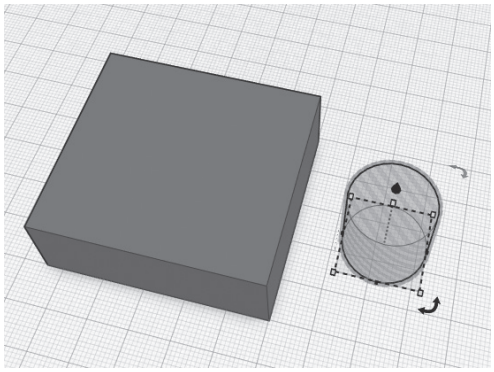
▼図11　Helpers機能



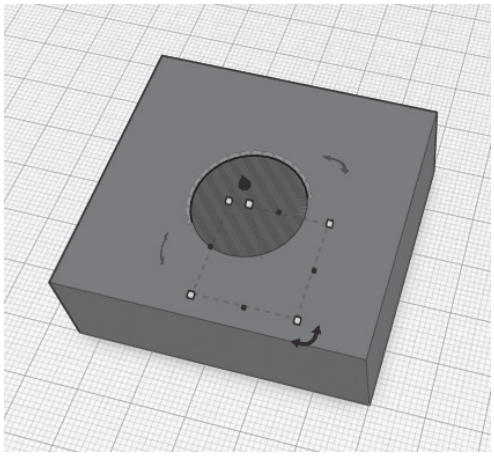
▼図12　Workplaneをパーツ斜面に適用



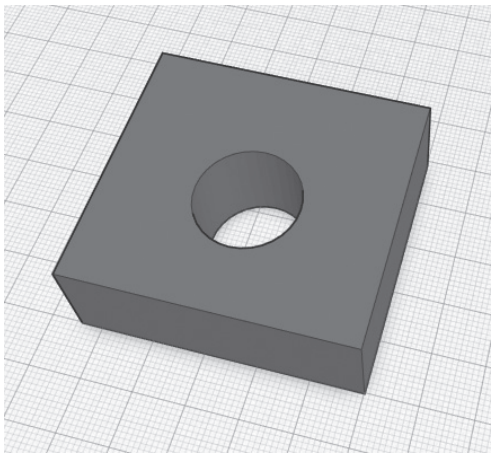
▼図8　Box(立方体)と穴あけ用のCylinder(円柱)



▼図9　穴あけしたい位置にCylinderを配置



▼図10　“Group”で穴あけ



ることで、あたかも積み木を組み上げるように高さ方向のズレをまったく気にせずレイアウトできる機能です[注9]。

また、mm/inch単位を基本とするグリッドスナップ機能を備えているので、マウスである程度の位置調整が簡単にできます。スナップする寸法はメイン画面右下の“Unit”でmmかinchかを選び、“Snap grid”でスナップ寸法を選びます。

実際の使い方は、“Helpers”に表示されているWorkplaneサムネールをクリックし、メイン画面の任意のパーツの任意の面にクリックして貼り付けるだけです。これでその面の無限面に設定したスナップでしか、ほかのパーツをレイアウトできないようになります。

デフォルトのWorkplaneに戻すには、何もないところにWorkplaneを配置すればOKです。

・Ruler

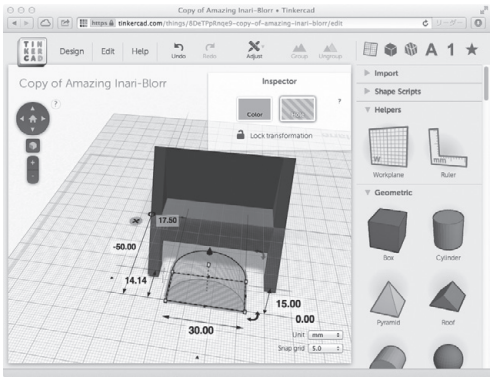
これも強力なモデリングサポートツールです。一時的に原点を自由に移動／設定することができます。Rulerサムネールをクリックし、基準とする位置にルーラーを配置します(**図13**)。

たとえばあるパーツの角を基準に、「X方向に何mm、Y方向に何mmのところに別のパーツをレイアウトしているか？」をリアルタイムに確認できます。さらにこのRulerの便利なところは、レイアウト後に各寸法をクリックして数値を編集すれば、正確にそのパーツの相対位置とそのパーツ寸法を指定できることです。不要になったら“×”アイコンをクリックしてルーラーを消去します。

マウスだけの操作でレイアウトしたときに、「見た目はくっついているように見えても、実際は離れたりずれたりして、3Dプリントしたときにバラバラになってしまう」などの設計ミスを防ぐことができるので、これも何度も利用することになります。本当は、角度も表示／設定できるようになるともっと強力なツールになるのですが、これは今後のアップデートに期待したいと思います。

注9) 例外的にパーツのZ軸移動ハンドルを使えば、パーツをWorkplane面からオフセットレイアウトできます。

▼図13　Rulerの基準を上部パーツに適用



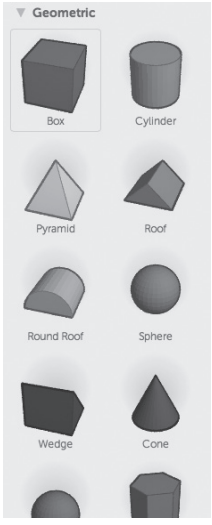
Geometric～Extrasメニュー

これらがTinkerCAD標準で用意されている基本パーツです。おおよそのモデリングは「これらのパーツを自在に変形させつつ組み合わせる」ということになります(**図14、15**)。子供が使うことを意識して、かなり可愛いアイテムがたくさんあります。

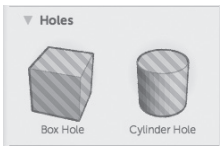
メイン画面

さて、最後はモデリングのためのメイン画面

▼図14　Geometric



▼図15　Holes



を見てみましょう。基本的な操作は、サイドバーのアイテムをクリック選択後、Workplane上にマウスドラッグで位置決めして、各ハンドルを使って変形／回転／移動することになります(**図16**)。

カメラビュー

メイン画面の右上にモデル名と視点を自在に変えることができるアイコンが表示されています(**図17**)。モデル名は初めはランダムな名前が付けられてしまいますが、もちろんダッシュボードから任意の名前に変更できます。

カメラビューは3Dモデリングには欠かせない機能です。アイコンごとに上から見ていきます。

一番上のアイコンは、家アイコンをクリックするとデフォルトの全体表示、矢印アイコンをクリックすると全体を前後上下左右から見ることができるRotate機能になります。真ん中のアイコンは該当するパーツを選んでからこのアイコンをクリックすると、そのパーツを全体表示することができます。一番下の＋－アイコンは文字どおりズーム機能です。また、右上の“？”をクリックすると、マウス操作のショートカットが表示されます(**図18**)。

アイコンでは画面全体を上下左右に移動するPan機能がありませんが、マウスではShift＋右クリックで操作することができます。このマウスショートカットは操作がとても簡単になるので、覚えておいて損はありません。

グリッド設定

すでにWorkplaneの項目で説明しましたが、メイン画面右下にはグリッドスナップの設定メニューが表示されています。必要に応じて設定を使い分けるととても便利です。

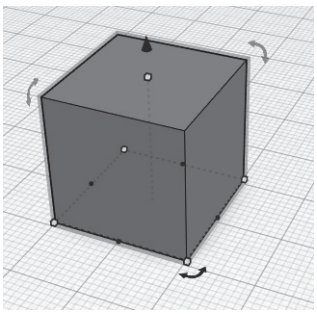
インスペクタ(カラー／穴あけ／変形ロック)

パーツを選択した状態では、右上にそのパーツのプロパティを変更できるダイアログが現れます(**図19**)。好きな色に変えたり、穴あけ属性を持たせたり、不用意な変形をロックできます。

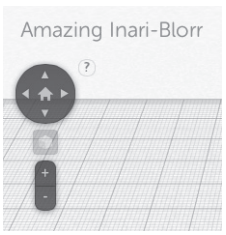
iPhone 5スタンドをモデリングしてみよう

さて、ここからいよいよ実際に3Dモデリングしていきましょう。3Dプリンタを使う理由の1つは、「自分がちょっとほしいものを自分で作

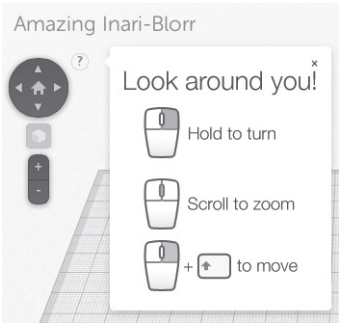
▼図16　パーツ変形／回転／移動のための各ハンドル



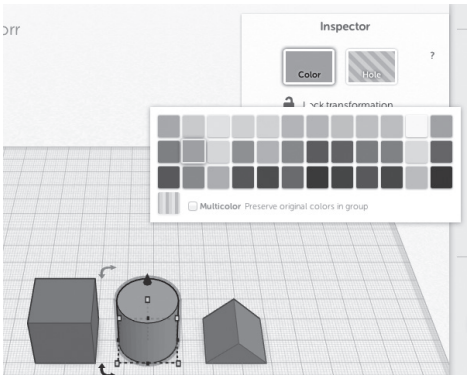
▼図17　モデル名とカメラビューアイコン



▼図18　カメラビュー機能のマウスショートカット



▼図19　パーツ変更ダイアログ



る」ことにあると思います。筆者の場合、「自分の所有するiPhone 5でムービーを手に持たないで見たい」と思っていました。そんなに大きくないですし、ちょうどTinkerCADで簡単に作れそうですので、iPhoneスタンドを作ってみることにしました。

最初にiPhone 5の寸法を調べる

iPhoneスタンドということは、iPhoneの寸法を知らないことには始まりません。Appleのサイトに寸法が出ていました[注10]。

・高さ：123.8mm
・幅：　58.6mm
・厚さ：7.6mm

どんな仕様(使い方・形状など)にすべきか?を考える

さて、どんなスタンドにしましょうか？　いわゆる仕様の決定です。今回筆者が考えたiPhoneスタンドの仕様は、次のようなものです。

iPhoneスタンド仕様

・iPhoneでムービーを見るので、見やすい角度で横に立てかけられること
・できれば縦にも立てかけられること
・Lightningコネクタが挿せること
・さっと置くだけにしたい(ネジか何かで固定することはしたくない)
・机においても邪魔にならない大きさ(せいぜいiPhoneの幅＋α)
・TinkerCADと3Dプリンタで無理なく作れるシンプルな形状

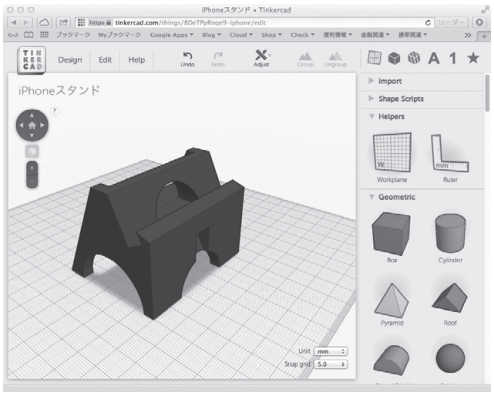
完成イメージが図20〜22です。

何か便利なものを作るときは、それがちゃんと機能するものでないといけません。iPhoneスタンドで重要なことは何でしょう？

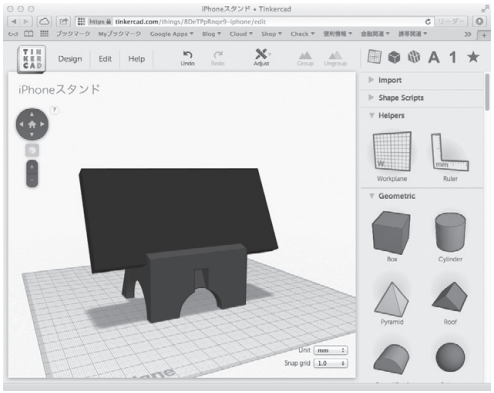
まずは、最低限iPhoneがきちんと立てかけられないといけません。

さっと置くだけにしたいので、背面の壁に立てかけたときに後にひっくり返らないように、iPhone下部に引っ掛けを作ります。スタンドが

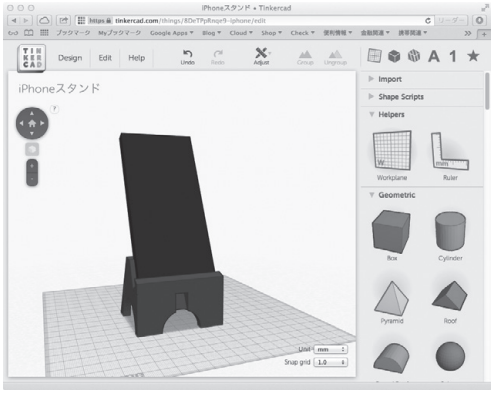
▼図20　最終形のiPhoneスタンド



▼図21　iPhoneを横に置いたイメージ



▼図22　iPhoneを縦に置いたイメージ

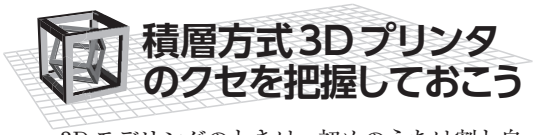


注10) http://www.apple.com/jp/iphone/specs.html

倒れないためにはiPhoneの重心より後ろのほうまで十分に足が伸びていないといけません。

次に、画面がきちんと見えなければいけません。スタンドが画面に張りだしてしまったり、タッチパネルの操作の邪魔にならないようにしたいです。なので、引っ掛け部分の張り出しは最小限にしたいと思います。

iPhoneを縦にも置けるように、Lightningケーブルを逃がす部分も配置しようと思います。



積層方式3Dプリンタのクセを把握しておこう

3Dモデリングのときは、初めのうちは割と自分の都合で設計をしてしまうのですが、ある程度実際のモノを作って失敗すると、モデリング時から前もって予防線を張ったデザインができるようになります。筆者もまだ積層方式3Dプリンタを使い始めて数ヵ月ですが、それでもその経験の中で「ここだけは抑えておかないとな」と思うノウハウがいくつかありますので紹介します。

積層方式に気を付けたデータの作り方

とくに寸法とオーバーハング(後述)に着目します。

寸法

積層方式の特徴は、ヒーターで溶かした樹脂を、先を絞ったノズルから糸のようにニュルニュルと押し出し、押し付けながら成形します。その糸の幅の分だけ寸法の誤差が出ます。今回使用するUltimakerは、ノズル径が約0.4mmで、実際に測ってみてもそのくらいですので、0.4mmの幅の線を引くことになります。すると、線の中心から左右に約0.2mm寸法が変わります。

たとえば直径10mmの軸と穴を組み合わせるようなデザインをしたと仮定すると、素直に軸を作ると直径10.2mm、穴は直径9.8mmとなり、軸を穴に通すことはできくなります。「軸を細くするか、穴を大きくするか？」ということを考えないといけません[注11]。

オーバーハング

積層方式3Dプリンタの最大の課題ですが、空中には樹脂を成形できないので、「あらかじめサポート材という捨て材料をプリンタテーブルから延ばしておき、それを橋渡しにしてせり出した(オーバーハング)部分を作る」というものがあります。もちろん、デザイン上必然であればこの対処をしなければなりません。

ただし、サポート材を使うデメリットがあります。まず、無駄な材料を使ってしまうこと。これは当然材料コストに跳ね返ってきます。また、たとえばプリンタテーブル対してに下向きにとても深い穴を作る場合、サポート材を取り除くのがとても難しくなるなど、あとでサポート材を取り除きやすいかどうかも意識しておいたほうがいいでしょう。

オーバーハングを避けるために、あえてモデルを横倒ししたデータを作り、サポート材を不要にできないか考えることも対策の1つになります。

プリント時間(と材料コスト)を減らすデータの作り方

プリント時間は、結局は「どれくらいの体積の樹脂を押し出すか」に比例しますので、体積を減らす＝プリント時間と使う樹脂の量を減らすのが効果的です。そのアプローチは大きく2つ考えられます。1つは3Dプリンタの出力時に使うスライサーアプリで設定を変更してなるべく無駄な樹脂を使わない方法です[注12]。もう1つはデザイン時に不要な部分を省くように、穴をくり

注11) そもそも直径10mmぴったりの軸と穴同士でもキツ過ぎて、壊さずにはめるのはとてもたいへんです。もしスムーズに穴を回転する軸にしたいときはもっとお互いに緩くなるように寸法を決めます。これを機械工学的には「はめ合い」と言います。もちろん、3Dプリント後に、ドリルやリーマーなどの工具で穴を広げる後加工をする方法もあります。

注12) 次章で説明します。

抜いたり、凹ませたりさせることです。今回のiPhoneスタンドでもそうしてみます。

材料の特徴を意識したデータの作り方

積層方式3Dプリンタの材料で最もメジャーなのは、ABSとPLAです。

ABS(Acrylonitrile Butadiene Styrene；共重合合成樹脂)

ABSは樹脂製品で最も多く使われている樹脂の1つで、パソコンや家電、車などでも多用されているエンジニアリングプラスチックです。ABSはいろいろな製品に使われているだけあって信頼性は高いのですが、積層方式で成形するためのポイントの1つであるテーブル密着性の悪さに苦労している話がネット上にはたくさん出ています。場合によっては成形中にテーブルからパーツが剥がれてしまって、グチャグチャになってしまいます。筆者は残念ながらABSの経験があまりないので突っ込んだ話ができないのですが、初めになるべくテーブルに接する面積を作ることがポイントかもしれません。

PLA(PolyLactic Acid；ポリ乳酸)

筆者が普段使っているのはこのPLAです。これは生分解性プラスチックですので、地球環境にやさしいと言われている樹脂です。この樹脂はABSとは性質が逆で、テーブルに貼り付きやすく成形はしやすいのですが、完成後にテーブルから剥がすときに力がいるので、薄いものは壊れることがあります。テーブルも傷みやすいので紙テープを貼って予防しますが、この紙テープも破れてくっついてきてしまうほどです。ですから、PLAの場合はなるべくテーブルに接する面積は少なくするのがポイントです。ただし設置面積を減らすということは、必然的にオーバーハング部分ができやすいデザインになるということです。なかなか悩ましいところです。

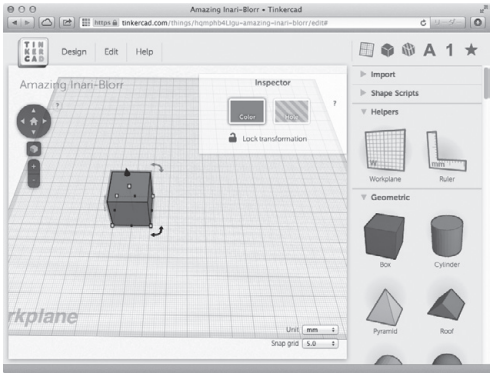
TinkerCADで実際にデータを作る

それではTinkerCADで、仕様を意識してデザインを進めていきましょう。今回はなるべくTinkerCADのシンプルさを最大限利用したいので、デザインにはあまりこだわらず、ざっくりと機能することだけを目指して作ってみようと思います。

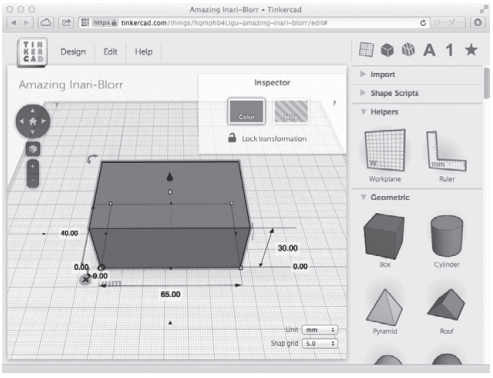
まず、大雑把にスタンド下部のベースとなる“Box”を適当な場所に配置します(**図23**)。

外形を仕様にマッチするようにハンドルを操作して変形します。今回は幅65mm、奥行き40mm、高さ35mmにしています(**図24**)。いずれも5mmで割り切れるので、“Snap grid”を5mmに設定すると変更しやすいです。各寸法の理由は、次のとおりです。

▼図23　ベースのBoxを配置



▼図24　寸法の調整



・幅65mm：iPhone 5を縦においた幅よりやや広めに設定
・奥行き40mm：画面を見やすくするためにiPhone 5を60度傾けたときの長さ
・高さ35mm：Lightningケーブルを挿したまま立てかけられるための逃げ分を考慮

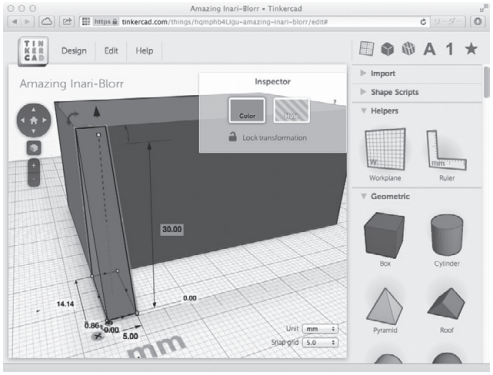
充電ケーブルの逃げを考慮した結果、高さがやや高く、このままではiPhoneを縦においたときに後にひっくり返ってしまいます。そうならないように十分な長さの足を後ろに取り付けます。“Roof”を90度回転し幅5mmにした足を両脇に2つ配置することにします(**図25**)。足の長さは倒れない程度に14.14mm(角度60°の斜面)くらいにします。真面目に重心位置を考慮すれば確実な寸法を得られますが、気軽に、ダメなら何回かトライすることにします。

反対側の足をメニューの“Edit”→“Duplicate”で同じパーツを複製して横にスライドさせて配置します(**図26**)。

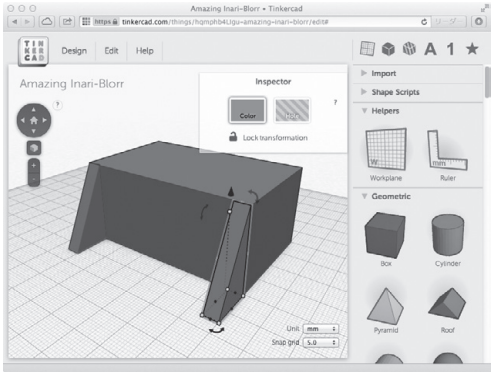
よく見ると少し隙間が開いてしまいました。3Dモデリングではよくある話です。これに気づかないとプリントしたときにバラバラになってしまいます(**図27**)。左の足に右の足を合わせてみましょう。“Adjust”→“Align”を使うと位置合わせができます。合わせたい部分の灰色丸表示をクリックすればOKです(**図28**)。

とりあえずここまででいったん3つのパーツを1つにグループ化してしまいます。“Group”をクリックします(**図29**)。グループ化しているかどうかは、全部が同じ色になることでわかります。もちろん、修正したくなったら“Ungroup”

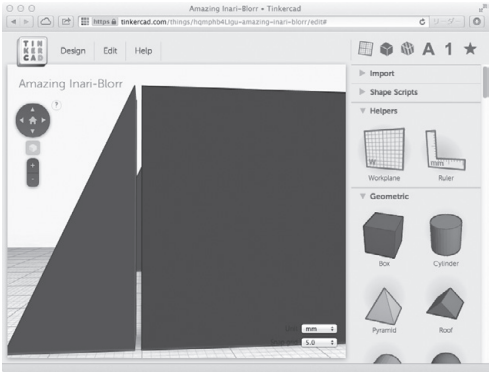
▼図25　“Roof”を配置



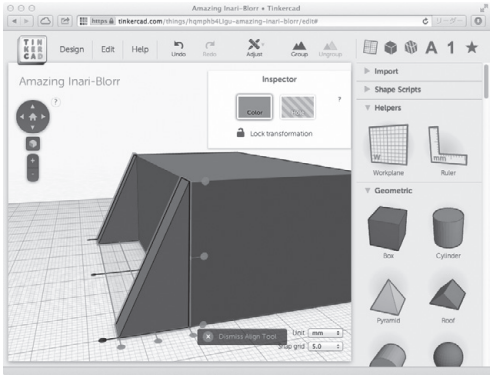
▼図26　パーツを複製する



▼図27　隙間が開いている



▼図28　位置合わせを行う



で解除して変更することもできます。

これだけでもけっこうな体積になるのと、テーブルに接する面積が非常に大きいので、材料を減らしつつテーブルへの貼り付きを緩和するために、デザイン的に間引くくことにします。“Round Roof”を使って穴を開けます。間引く大きさに適当に変形し、右上に出てくる“Inspector”で“Hole”の指定をします。すると灰色透明のハッチング表示になります(**図30**)。

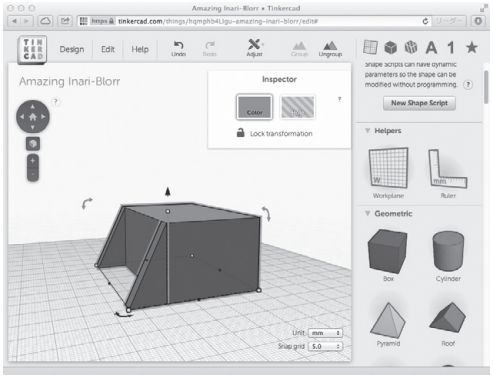
ベースの側面にピッタリになるように突っ込みます。2つのパーツを選択し、“Group”をクリックすると狙いどおりの穴があきました(**図31**)。

さて、ところでなぜ“Box”ではなく“Roof Round”で間引いたのでしょうか？　これは先ほど解説した、オーバーハングの問題を回避するためです。“Box”だと空中に急に真横反対側の柱まで長い距離の張り出しを作らなければならなくなるので、樹脂の糸が下に垂れてしまってきれいな形状が維持できなくなる可能性があり、場合によってはサポート材が必要になります。それに対して円弧でオーバーハングにすると、少しずつ横に伸びていく形になるので、確実にサポート材が不要になるのです。

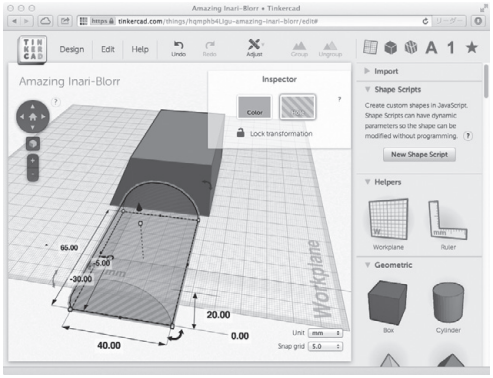
次にiPhoneを置くための部分をモデリングしていきます。Workplaneをベースの天面に設定します。そしてiPhoneを支える斜面“Wedge”を変形＆組み合わせで作ります。ベースの足部に滑らかにつながるように後部の支え部分を配置します(**図32**)。逆側にも同じ形状を“Edit”→“Duplicate”後、移動＆配置します(**図33**)。

iPhoneの背面を寄りかからせる斜面を作ります。先ほどの“Wedge”をコピーし、それを

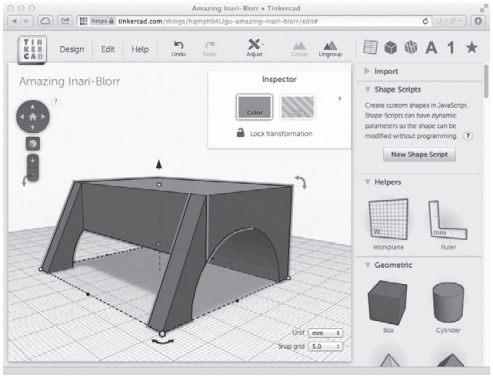
▼図29　グループ化する



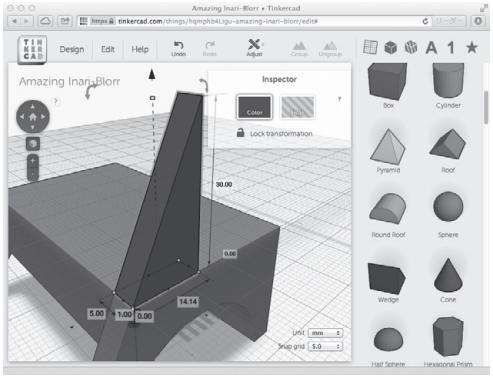
▼図30　間引く図形がハッチング表示される



▼図31　間引きを実行したところ



▼図32　iPhoneを支える部分を作る

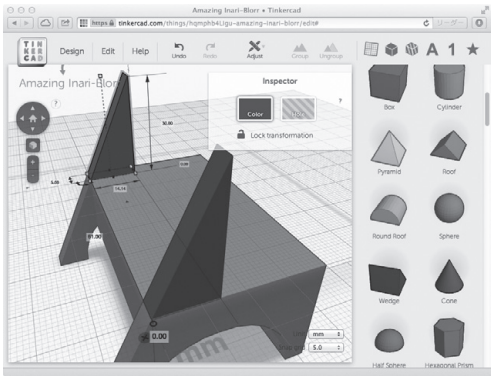


“Adjust”→“Mirror”で反転させ、背中同士をくっつけます。これで60°の斜面になりました。見えにくい角度なら後で作りなおそうと思います(**図34**)。それをベースの幅いっぱいに延ばして斜面ができました(**図35**)。

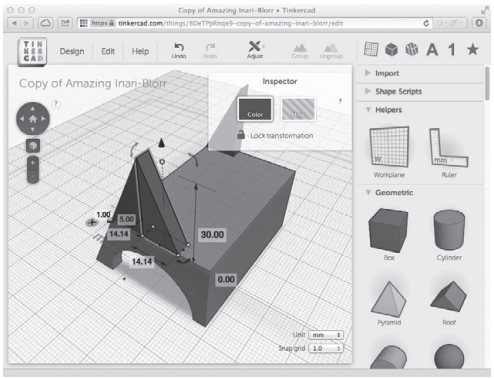
次にiPhone下部を支える斜面です。これも“Wedge”を変形させて背面斜面に垂直になるように配置します(**図36**)。背面斜面も真ん中に穴を開けてしまいましょう。**図37**はグループ化したところです。

続いてiPhoneが後にひっくり返らないように引っ掛けを作ります(**図38**)。Workplaneを下部

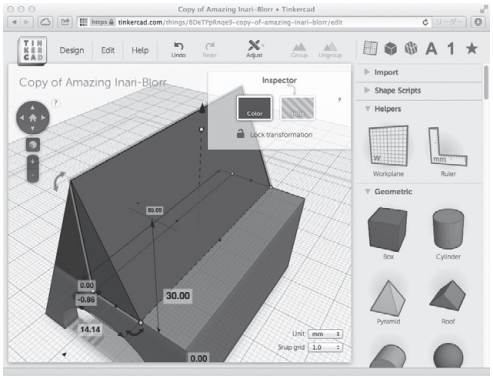
▼図33　支えを複製する



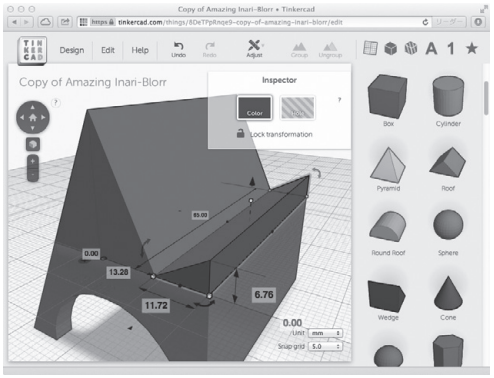
▼図34　iPhoneを寄りかからせる斜面を作る



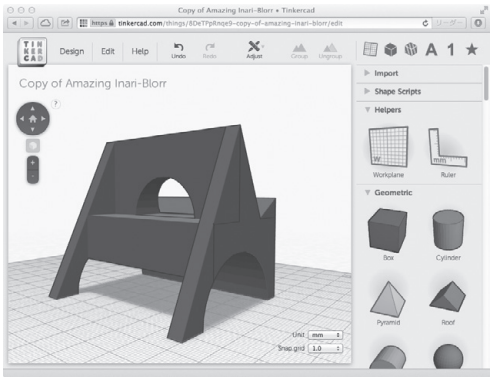
▼図35　横幅を延ばす



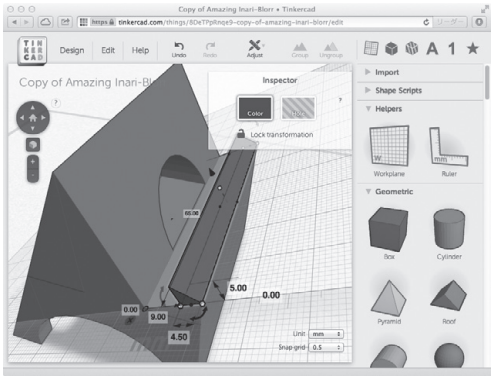
▼図36　下部を支える部分を作る



▼図37　穴をあけてグループ化する



▼図38　引っ掛けを作る



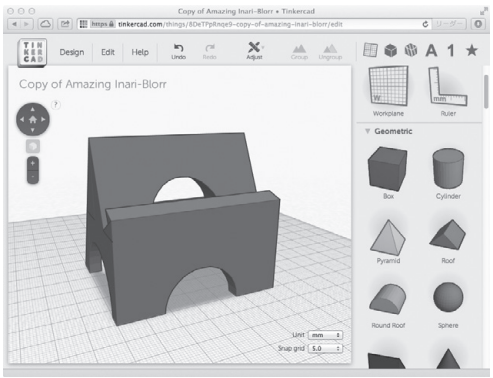
斜面にセットします。やはり“Wedge”で引っ掛けを作るのですが、iPhone 5の厚みは7.6mmですので、余裕を見て下部斜面幅が9mmになるように変形を調整します。引っ掛け高さはiPhone 5を横置きしたときに画面にかからない程度の5mmにしました。

ここまで作ってみて、前面部分もちょっとボッテリした感じがしたので、コスト削減のために“Round Roof”で削ってしまうことにします(**図39**)。

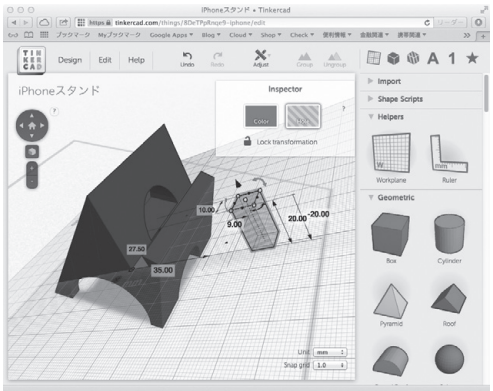
縦置きしたときのLightningケーブルの逃げ穴を“Box”を“Hole”にして作ります(**図40**)。Workplaneは下部斜面にセットしてから“Box”を作らないと、角度がうまく出せないので注意が必要です。ルーラーを下部斜面の角にセットすると、ルーラー原点からの相対位置を直接数値を入れられるので、逃げ部分の位置決めが正確にできます。

最後にもう1個所、一番上のとんがりはいらなさそうですので、これも上から10mmほどカットすることにしました(**図41**)。Workplaneがどの面上でもない位置にセットされていますが、これはスタンドを垂直に50mm下げる(Z軸の黒い三角錐ハンドルをドラッグする)ことで実現しています。十分に大きな“Box”穴でカットしてしまいます。

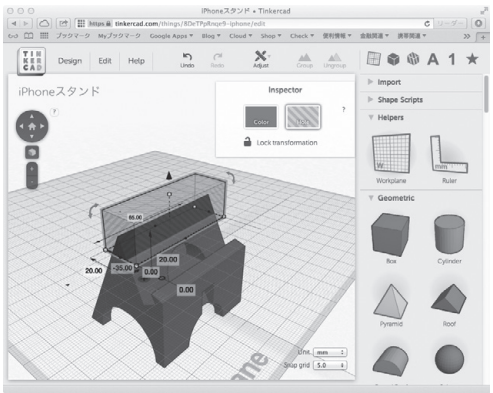
▼図39　前面部分を削る



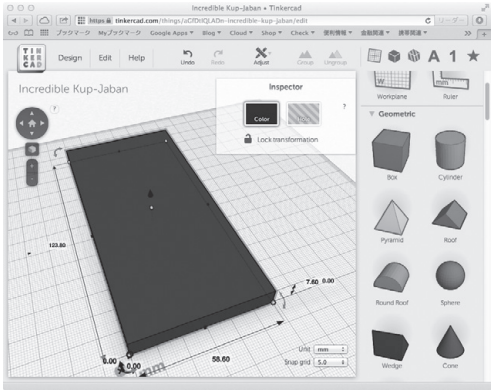
▼図40　Lightningケーブルの穴を開ける



▼図41　とんがりをカットする



▼図42　iPhoneのイメージを作る



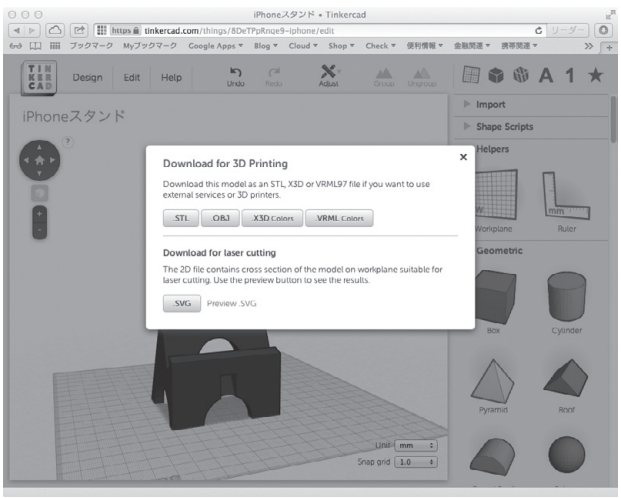
完成!

本当にうまくiPhone5が載せられるか、モデル上で検証してみようと思います。いったんスタンドのデザインを“Design”→“Close”でダッシュボードに戻り、iPhone 5を新規のデザインで起こします。角丸やボタンは今回は端折(はしょ)っているので、まるでモノリスのようですが、外寸はきちんとしています(**図42**)。これを“Edit”→“Copy”して再びiPhoneスタンドモデルのほうに戻ります。

背面斜面にWorkplaneをセットしたら、先ほどコピーしておいたiPhoneをペーストして、スタンドと組み合わせてみます。ここでもルーラーを使えばきちんと位置決めができます(**図21、22参照**)。お位牌にしか見えませんが(笑)、どうやら大丈夫なようです!

さて、一刻も早く3Dプリントしてみたいですよね。そのためにはメニューの“Design”→“Download for 3D Printing”で出てくるダイアログで、“.STL”を選べばローカルにダウンロードされます(**図43**)。実際の3Dプリントは次章で紹介します。

▼図43　ローカルにデータをダウンロードする



改良とバグフィックス

実はこのチュートリアルで紹介したiPhoneスタンドのモデルには不具合と改良すべき課題があります。実際に3Dプリントしてみるとなかなか思いどおりにはいかないものです。プログラミングでたとえれば、コンパイル&ビルドが終わっただけなのと同じで、機能的なバグは3Dプリント後に発見、修正するということです。

こんなふうにMaker界隈で言うところのAtomの世界は、モデリング修正と実物確認、つまり改良とバグフィックスを繰り返すことで完成に近づけていきます。今まではこの作業にかなり費用を費やさなければなりませんでしたが、この3Dプリンタを代表とするデジタル工作機械&ラピッドプロトタイピングにより、(完璧ではありませんが)かなり身近なものになったことはたしかだと思います。

ぜひ3D-CADにトライしよう

今までの3D-CADは、これで飯が喰えるプロの道具だったため、非常に高機能ですが高いスキルがないとモデリングができないものでした。また、価格も数百万～1千万円オーダーです。しかし、ReplicatorやRepRap、Ultimakerといった廉価な3Dプリンタの登場に合わせて、初心者でも取り組みやすいシンプルで無料の3D-CADが登場し、ちょっと練習すれば十分なモデリングができる環境が整ってきています。

ぜひ、一度実際にこれらの3D-CADをトライしてみてください。きっと新しい気持ちが芽生えてくるはずです。SD

理論から実践まで
3Dプリンタが未来を拓く理由

第4章 3Dプリンタで出力してみよう

工房Emerge＋　山田 斉（やまだ ひとし）
http://www.emergeplus.jp　Twitter:@emergeplus

第3章ではiPhoneスタンドを題材にして3Dモデルの解説をしました。本章では実際に積層方式の3Dプリンタを使ってiPhoneスタンドを3Dプリントしながら、作業の流れやポイントを解説していきます。なお、筆者の所有する3Dプリンタ「Ultimaker」[注1]（P.15写真17）を例にして説明します。

積層方式での3Dプリントの流れ

3Dプリントも、普通の2Dプリンタの印刷のように、できあがったモデルをそのままPCの3D-CADから3Dプリンタに出力できると話は簡単なのですが、実際はもう少し複雑です。3Dプリンタ専用のアプリケーションソフトにSTLファイルを読み込ませ、そこでいろいろな加工設定を施しつつ積層モデルに変換し、その後、3Dプリンタの制御コードを生成して出力します。

この専用アプリケーションソフトを一般的にはスライサーソフトと呼ぶことが多いようです。Ultimakerの標準スライサーはCura[注2]（**図1**）というアプリケーションですので、ここからはスライサーソフトについてはCuraをベースに解説をします。

一般に、積層方式3Dプリンタを使った3Dプリントの作業の流れは次のようになります。

①3Dモデルを3Dプリンタ用のファイル形式にして保存する
②スライサーソフトでモデルを積層モデルに変換する
③スライサーソフトで3Dプリンタ制御コード（Gコード）に変換する
④3Dプリンタで出力する

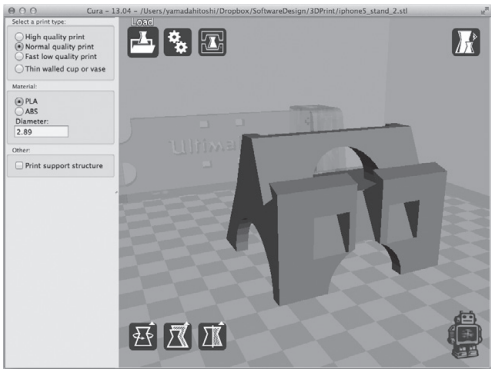
①3Dプリンタ用ファイル形式に保存

オープンソースハードウェア界隈で賑わっているほとんどの安価な3Dプリンタは、RepRapという3Dプリンタから派生しているものです。RepRap系列の3DプリンタではSTL（Standard Triangulated Language）が使われています。したがって、STL出力をサポートする（3D-CADなどの）3Dモデラを用意しなければなりません。第3章で紹介した3D-CADは、SketchUp Makeを除き、標準でこのSTL出力をサポートしています。SketchUp Makeの場合は、別途STL出力プラグインをExtension Warehouse[注3]からダ

注1）http://www.ultimaker.com/
注2）http://daid.github.io/Cura/

注3）http://extensions.sketchup.com/en/content/sketchup-stl

▼図1　スライサーソフト「Cura」



ウンロードしてインストールする必要があります[注4]。UltimakerもSTLファイルをもとに加工する仕様ですので、第3章で使ったTinkerCADからSTL出力してPCに保存して使います。

②モデルを積層モデルに変換する

積層方式の特徴は、あたかも病院のCTスキャンのように断面を積み重ねていくデータを生成し、それを一段一段樹脂でプロットしていくことです。

事前に3Dプリンタの加工条件(たとえば「どのくらいの密度で内部を樹脂で埋めるか？」とか、「1層の厚みをどのくらいにしておくか？」など)を設定しておきます。一度条件が決まって安定出力できるようになってしまえば、その後、変更することはあまりないと思います。この条件を織り込んでスライサーソフトは積層データを生成します。

③プリンタ制御コードに変換する

その後、3Dプリンタの制御コードであるGコード(**図2**)を生成します。しかし、多くのスライサーソフトは上記の積層モデル変換とGコード変換を、自動的に同時に生成すると思いますので、とくに作業が発生するわけではありません。Curaも一連の処理を一気に済ませてしまいます。Gコードというのは数値制御(Numerical Control：NC)工作機械の世界ではスタンダードの制御コードです。

④プリンタで出力する

ここまで準備が整えば、あとはそのGコードを3Dプリンタに送り込んで実際に3Dプリントするだけです。Curaの場合は途中でいったんプリントを中断することもできるので、色の違う樹脂フィラメントに差し替えて二色成形するという技も使えます。

プリントサイズしだいでは数時間かかることもザラです。プリントが最後まで何事もなく終わるか、ドキドキしながら待つことになります。

なお、ほとんどの3DプリンタはPCからUSB(あるいはWiFi)経由でGコードを送り出すことで3Dプリントします。中にはSDカードにGコードファイルを書き込み、これを3Dプリンタに挿入してスタンドアローンで動かせる機種もあります。

Ultimakerもオプションでスタンドアローンコントローラがありますので、筆者は普段は、これを使って3Dプリントをしています。完成して修正不要となったモデルをいくつか作るときは、いちいちPCとスライサーソフトを立ちあげなくてもすぐにプリントを開始できるメリットがあります。また、PCから制御するとうっかりスリープになって出力が止まってしまうことがありますし、何かのはずみでOSやスライサーソフトが落ちて制御不能になることもあります。停止したいときは、コントローラからすぐに停止できます。なによりPCの机と3Dプリンタが置いてある机が、かなり離れていてUSBケーブルが届かないせいもあります……(苦笑)。

▼図2　Gコードの例

iphone5_stand.gcode

```
;TYPE:CUSTOM
M92 E865.888000
M109 T0 S220.000000
T0
;Sliced /Users/yamadahitoshi/Dropbox/Software
Jun 2013 14:06:54
;Basic settings: Layer height: 0.2 Walls: 0.8
;Print time:     1:01
;Filament used:     4.07m    34.72g
;Filament cost:  Unknown
G21         ;metric values
G90         ;absolute positioning
M107        ;start with the fan off
G28 X0 Y0   ;move X/Y to min endstops
G28 Z0      ;move Z to min endstops
G1 Z15.0 F180 ;move the platform down 15mm
G92 E0                  ;zero the extruded le
G1 F200 E3              ;extrude 3mm of feed
G92 E0                  ;zero the extruded le
G1 F9000
M117 Printing...
;LAYER:0
;TYPE:SKIRT
G1 X66.8 Y83.668 Z0.3 F9000.0
G1 F2400.0
G1 E4.5
G1 F9000.0
G1 X66.8 Y113.668 Z0.3 F1200.0 E5.0488
G1 X66.8 Y117.216 E5.1137
```

注4) スライサーソフトにCuraを使うのであれば、メニューで「ファイル」→「エクスポート」→「3Dモデル」と選択して.DAEフォーマットで保存すれば、プラグインなしで読み込めます。

無料のスライサーソフト

さて、一通り作業の流れを把握したところで、スライサーソフトに着目してみようと思います。スライサーソフトは積層方式3Dプリントの肝になる部分です。3Dプリンタとの相性でできあがりの品質も大きく変わってきます。無料のスライサーソフトがたくさんあるので、いくつか紹介します。ほかにもたくさんあります。使いやすく狙いどおりのプリントができるスライサーソフトを探してみてください。

(1) Skeinforge+ReplicatorG[注5]

安価なオープンソースハードウェア3Dプリンタのパイオニアである CupCakeCNC のスライサーソフトとして登場しました。Skeinforge でGコードを生成後、ReplicatorGで3Dプリンタへ Gコードを送ってプリントします。初期に登場したスライサーであるだけに細かい設定ができ、根強い人気があるようです

(2) Kisslicer[注6]

これも RepRap では人気のあるスライサーソフトです。はじめに表示される画面にたくさんの設定項目が登場するので、それに圧倒されて慣れるまではたいへんかもしれません

(3) Cura

本章で使用している3Dプリンタの Ultimaker の標準スライサーソフトです。UIも直感的でとてもわかりやすいと思います。「quickprint」モードなら3クリックでプリントを始められます

注5) http://replicat.org/download
注6) http://kisslicer.com/

Curaの機能概要

3Dプリントを行う前に、今回使うスライサーソフト Cura について、簡単に機能を紹介したいと思います。本格的に説明し始めるときりがありませんので、必要最小限の解説にとどめておきます。

Cura には「quickprint」モードと「full settings」モードがあって、メニューの「Tools」から切り替えられます。プレビュー画面では3クリックでプリントできるアイコンや拡大／縮小、回転などを施してからのプリントなど、簡単で柔軟なUIを持っています。

quickprint モード

quickprint モードはその名のとおり、ほとんど Cura 任せで3Dプリントできます。プリント品質と ABS/PLA 選択、サポート材の有無を指定するのみで OK ですので、とても簡単です(**図3**)。筆者もおおよそのプリントはこれで済ませてしまうことが多いです。

full settings モード

かなり詳細に設定ができるモードです。プリンタヘッドの移動速度やプリント物の内部の樹脂充填密度、プリント品質などの詳細な設定が細かくできます。詳しい説明はしませんが、「Basic」タブのスクリーンショットを載せておきます(**図4**)。

プレビュー画面

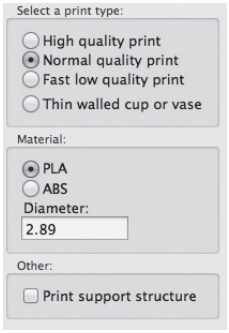
プレビュー画面の左上には、Load(ファイルロード)→Prepare(積層データ変換)→Print(プリント)の3クリックでプリントできるアイコンが並んでいます(**図5**)。Prepare では変換の進捗や予想プリント時間、使用する材料の量などが画面下のステータスバーに表示されます(**図6**)。また、Print では温度状況やプリント進捗状況がリアルタイムに表示され、プリントを一時中断

したり、手動でテーブルやエクストルーダの位置を動かすこともできます(**図7**)。

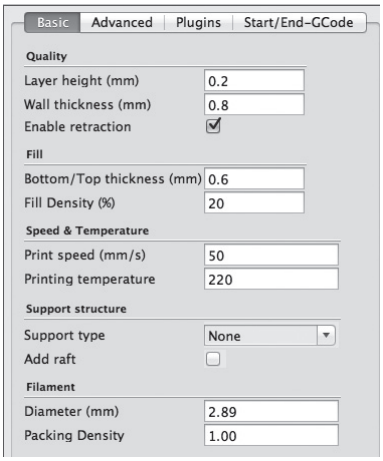
プレビュー画面にはモデルの拡大／縮小(Scale)、回転(Rotate)、ミラー配置(Mirror)をする機能もあります。元のモデルよりも大きく(または小さく)プリントしたり、オーバーハングの影響がないように横倒しにしてプリントしたりすることができる便利な機能です。これらは画面左下に配置されています(**図8**)。**図9**はモデルを回転させている様子です。

プレビュー画面にはもう1つすばらしい機能があります。右上のviewmodeアイコンがそれで

▼図3　quickprintモードのメニュー



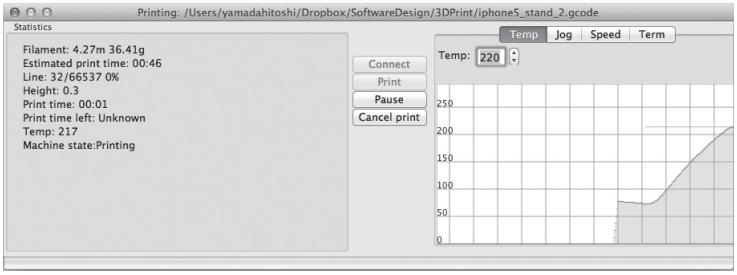
▼図4　full settingsモードのメニュー(Basicタブ)



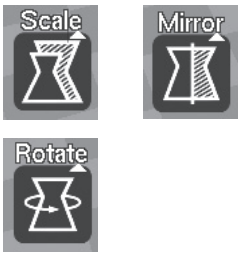
▼図5　Load/Prepare/Printの各アイコン

▼図6　Prepareステータスバー

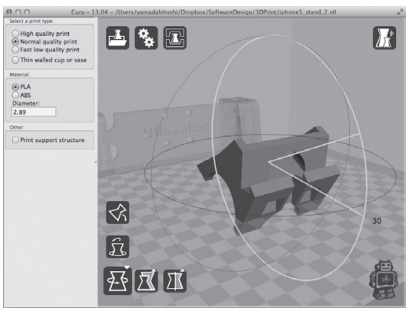
▼図7　Printダイヤログ



▼図8　Scale/Rotate/Mirrorの各アイコン

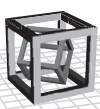


▼図9　モデルをRotateさせているところ



▼図10　viewmode各アイコン

す(**図10**)。「Normal」は通常のモデルイメージの表示、「Transparent」は透過表示、「X-Ray」は3Dモデル的に問題(不整合)がある個所を色つけ表示、「Overhang」はオーバーハング部分を色つけしてサポート材が必要かどうか確かめられる表示(**図11**)、「Layers」は積層モデルで見たい高さの断面がどうなっているか、を確認できる表示です。モデルをプリントする前に不具合を確認できますので、とても便利な機能だと思います。

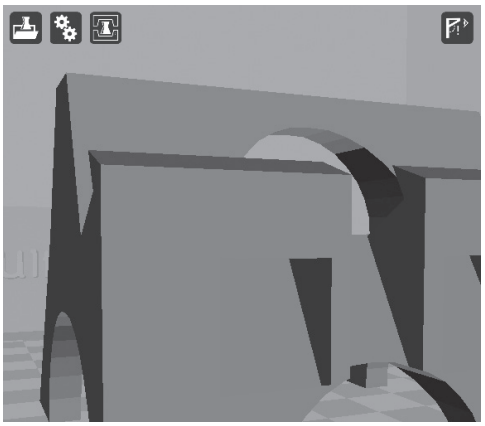


iPhoneスタンドを3Dプリント

前置きが長くなってしまいましたが、実際にUltimakerとCuraを使ってiPhoneスタンドを3Dプリントしてみます。はじめにTinkerCADで作成したモデルをSTLでPCにダウンロードしておきます。そして「Load」アイコンをクリックしてダウンロードしたiPhoneスタンドのSTLを選びます(**図12**)。

ロードが完了すると、次はプリント条件の設定です。CuraはUltimaker向けにチューンナップした設定を持っているので、「quickprint」モードを使えばプリント品質とABS/PLA選択、サポート材の有無を選択するだけでほとんど何もせずに3Dプリントができます。

▼図11　overhang表示(対象部分が赤い色で表示)



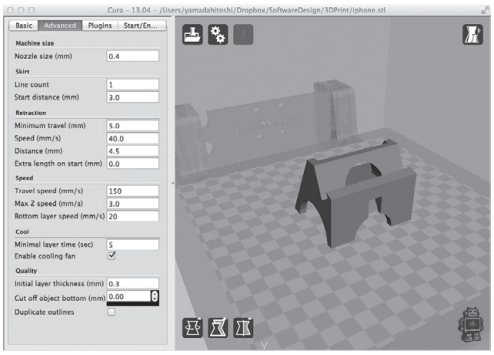
今回のプリントでは、「Normal品質」「PLA」「サポート材なし」で行います。設定が終了したら、積層モデル→Gコード自動変換のために「Prepare」アイコンをクリックすればウィンドウ下部にステータスバーが表示され、変換が始まります。変換が終わると、プレビュー画面に積層データが表示されます(**図13**)。

右のスライダーをドラッグすると途中の層が表示できるので、内部の充填密度や外壁の厚みなど、問題がないか確認します(**図14、15**)。

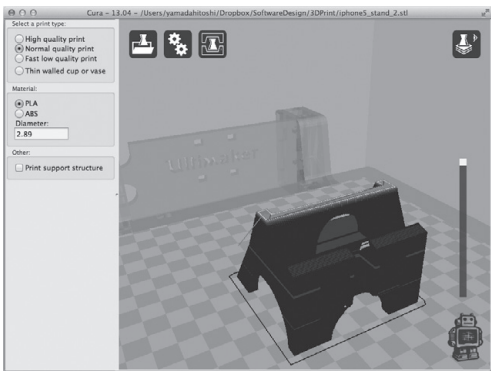
問題がなければ、いよいよ3Dプリントです。USB経由でプリントする場合は、「Print」アイコンをクリックしてPrintダイヤログで「Print」ボタンをクリックするだけです。

ただし、Curaの仕様なのか、バクなのかわからないのですが、ボタンをクリックしてもヒーターの温度が反映されず0℃のままで温度が上が

▼図12　iPhoneスタンドをロードしたところ



▼図13　積層データの表示



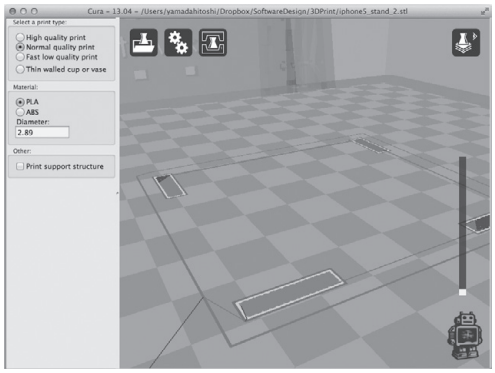
らないので、ボタンをクリックする前に「Temp」タブでPLAの場合の推奨ヒーター温度である220℃を入力しておきます。これでヒーターの温度が上がり始め、220℃に達したら自動的にプリントが始まります(**写真1**)。

今回のデータでは、Curaのデフォルトのヘッドスピード50mm/sだと2時間くらいかかりました(**写真2**)。

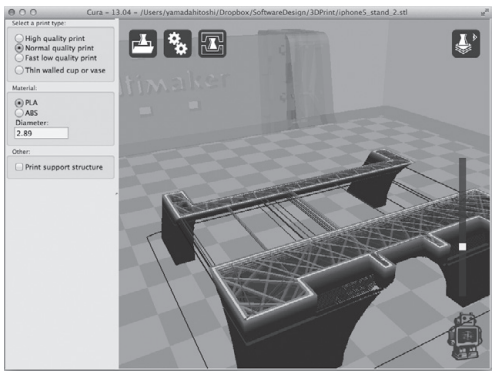
PLAで作成すると、テーブルにガッチリと張りついていますので、剥がすのに結構、力が必要です。テーブルに貼った紙テープもすぐに破けて使い物にならなくなります。薄い作品の場合は割れて手に怪我をすることがありますので注意が必要です。

ABSの場合は、逆にテーブルから剥がれやすく、プリント中に作品が動いてしまい、ゴミを作ってしまうことも多いようです。

▼図14　断面の最下層



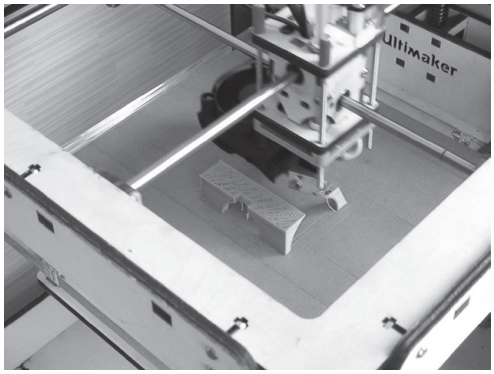
▼図15　途中の断面



できあがったiPhoneスタンドには、ヘッドの移動で空押し出しされた糸くずのようなフィラメントがあちこちにはみ出しています(**P.15写真18**)。これは現状の積層方式ではどうしても出てしまいます。ニッパーやカッター、ヤスリなどを使って仕上げることになります。

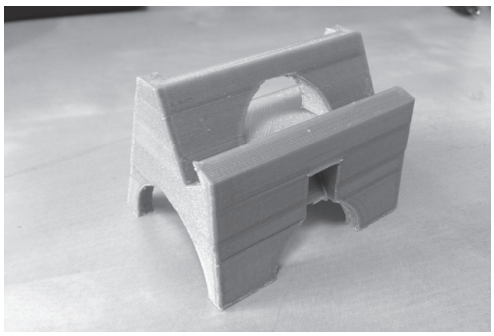
さて、オーバーハングの状態を確認してみま

▼写真1　プリント中のiPhoneスタンド



▼写真2　できあがったiPhoneスタンド

▼写真3　完成したiPhoneスタンド(試作)



す。やはり長い距離になにも支えがないところは少し垂れ下がっています(**P.15写真19**)。今回は機能的には問題にならないので、そのままで良しとします。

これで完成しました(**写真3**)。さっそく、iPhone 5を置いてみます。横に置いたときはまったく問題なしです。縦に置いてみると……。一応置くことはできますが、若干重心が後ろ過ぎて、少し押すと後ろに倒れてしまいました(**写真4**)。もう少し足を長くしないといけないようです。

また、Lightningケーブルは穴を通して挿すようにしたのですが、いちいち外さないと載せられないのが不便です。ケーブルを外さずに置けるように、ちょっと改善が必要です。

立てた状態で音楽を聞くとスピーカーが埋もれているので、こもった音になってしまって少し調子が悪いです。穴を開けたほうがよさそうです。

▼写真4　iPhone 5を縦においたところ(試作)

▼写真6　iPhone 5を縦においたところ(完成)

ラピッドプロトタイピングの魅力

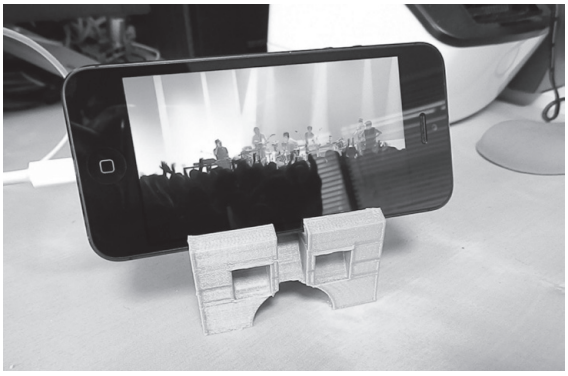
さて、試作でiPhoneスタンドの課題が浮き彫りになりました。TinkerCADでモデルを修正～3Dプリントして、再チャレンジです。この過程の詳細は記しませんが、**P.15写真20、21、写真5、6**のように仕上げてみました。

まず、足の長さを6mm長くして、後ろに倒れないようにしました。Lightningケーブルが通る部分は、溝にすることで取り外さずに縦置きできるようになりました。スピーカーの音の抜ける穴も用意したので、音がこもらなくなったと思います。iPhone 5の左の穴はマイク穴ですので音が入りやすいようにこちらも開けてあります。

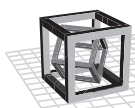
2回目の仕上がりはバッチリです。これでやっと自分の思うスタンドが完成です。この3DモデルはTinkerCADでCreative CommonsのBY-SAで公開してあります[注7]ので、よろしければご利用ください。

注7) https://tinkercad.com/things/hOY27ARjFzb-iphone5-stand-2

▼写真5　iPhone 5を横においたところ(完成)



3Dプリンタを代表とするラピッドプロトタイピングは、失敗してもすぐに作りなおしができて、しかもそれほどコストもかからないのが魅力です。逆に一度でうまくいくことはなかなかありません。このような失敗をすることで工作機械のクセをつかんだり、設計ノウハウを貯めたりできるので、実はメリットもたくさんあるのです。失敗しても諦めずに何度も試作を繰り返しているうちに、自分でモノを作る楽しさが心の中にできあがっていることに気づくことでしょう。



3Dプリンタのメンテナンス

最後に3Dプリンタを使用するにあたって、とても重要な「メンテナンス」について話したいと思います。

3Dプリンタも工作機械の一種です。正しく動かし続けるためには、日々メンテナンスを心がけないといけません。とくに安価な積層方式3Dプリンタはまだ登場して日も浅いうえに、キットで組み立てるものも多いため、ちょっとのことですぐに調子が悪くなります。いくつかメンテナンスのポイントを挙げておきます。

X、Y、Z各軸

3Dモデルが設計どおりにプリントされるためには、その位置決めをするX、Y、Zの各軸がスムーズに動かなければなりません。Ultimakerの場合はX、Y軸はプーリー・ベルト・シャフトで、Z軸はネジ・シャフトをステッピングモータを使って動かしています。

モデルが歪んでしまったりする場合は、ベルトの張りが適切か、シャフトがグリス切れになっていないか、筐体が歪んでシャフトが歪んでないか、ということをチェックするといいと思います。

テーブルの水平度

作品を作るためのテーブルは必ず水平にしておかなければ正しくプリントできません。傾いているとフィラメントが固定されなかったり、エクストルーダのヘッドがテーブルに当たって引っかかってしまい、脱調(ステッピングモータのトルクが負けて空回りすること)するなど、何もいいことがありません。

テーブルを水平に保つための調整機構、たとえば、ネジなどがあるはずですので、うまくプリントができないと思ったときは調整をしてみましょう。

エクストルーダ

エクストルーダは樹脂フィラメントをヒーターの熱で溶かして、それをノズルから細長く押し出す部分です。適切な温度の管理をするための温度センサーが付いていますが、これが正しく付いていないと、違う温度になってうまくフィラメントが押し出せなくなります。

フィラメントがうまく溶けていないと感じたら、センサーが壊れていたり外れていたりしていないかをチェックしてみましょう。

フィラメント送り機構

3Dプリンタによってフィラメント送り機構が付いている位置は違いますが、フィラメントを挟んで送り出すローラー部分とそれを歯車で回すステッピングモーターで構成されています。これが正しくスムーズに動くように組めていないとフィラメントが計算どおりに送れなくなります。挟みがキツ過ぎても緩過ぎてもいけません。フィラメントがうまく押し出せないときはここをチェックします。

また、時々フィラメント自体がリールの中で絡まって回らなくなることもありますので、そこも確認するとよいでしょう。SD

理論から実践まで
3Dプリンタが未来を拓く理由

第5章 3Dプリンタを選ぼう

坪井 義浩（つぼい よしひろ） ytsuboi@gmail.com Twitter: @ytsuboi
工房Emerge＋ 山田 斉（やまだ ひとし） http://www.emergeplus.jp Twitter: @emergeplus

ここまでは、3Dプリンタをとりまくさまざまな紹介をしてきましたが、結局3Dプリンタはどれを選べば良いのかと迷っている方もたくさんいらっしゃるでしょう。比較的購入して使っている人の多い3Dプリンタを例に、入手方法やそれぞれの特徴を少しずつですが紹介していきたいと思います。

Replicator 2

ホビーユーザ向けの3Dプリンタを検討するとき、Makerbot社の製品はたいてい候補に上がってくるでしょう。1章でも触れたようにMakerbot社のルーツは「RepRapプロジェクト」とも言える流れで、同社は2009年に発売した“Cupcake CNC”以来、ホビー用3Dプリンタの代表的なメーカーです。このCupcake CNCに続き、“Thing-O-Matic”という機種も出していたのですが、これらの2機種はキットで、プリントするまでには組み立てと調整の苦労が必要で、購入したけど使っていない、あるいは期待通りに動かせないといったユーザが数多くいました。このころは、3Dプリンタを使うには相応の努力が必要でした。

こういった事情もあってか、その後登場した“Replicator”からは同社で組み立て済みのプリンタとして販売されることになりました。さらに2012年に発表された“Replicator 2”（**P.15写真22**）からは組み立て済みであることに加え、従来のオープンソースからクローズドソースへの方針の変更が見受けられるようになり、多くの議論を呼ぶこととなりました。

現在の同社の製品には、Replicator 2と、最近発売された“Replicator 2X”（**写真1**）の2機種がありますが、ソースの問題はさておき、やはり人気のあるプリンタです。

Replicator 2の魅力

Replicator 2の魅力は、なんといってもパッケージを開けてプリンタを取り出せば、割にすぐに使い始められることです。スタンドアローンで動かすためのコントローラも標準で搭載されています。

プリンタを箱から取り出して、付属品（フィラメントのスプールをかけておくホルダ）と電源をセットして起動すると、初期設定のプログラムが動きます。このプログラムを使って、テーブルの位置（というか高さ）調整とフィラメント（これも付属です）をエクストルーダに挿入するということをします。これらのプロセスは、YouTubeにもアップロードされていますので[注1]、購入を

注1） 箱から取り出す：http://www.youtube.com/watch?v=ENtFOM-2I6o 初期設定：http://www.youtube.com/watch?v=KYGjICc-fV4

▼写真1 Replicator 2X

検討している人は見てみて、どんなものなのかあらかじめ感触を掴んでおくこともできます。

付属のSDカードにはいくつかの3Dプリント用のデータが入っています。セットアップを終えたら、これらのデータの中から出してみたいものを選んで出力できます。

トラブルに注意

こんなお手軽なReplicator 2ですが、ノートラブルというわけにはいきません。届いて少し使っただけでエラーが表示されたため、サポートに連絡を取り、ヒーターに初期不良があるということで交換部品を送ってもらった方がいらっしゃいます。また、筆者の身近にも、ノズルから樹脂が漏れてきてしまったためにサポートとのやりとりに非常に苦労しているケースも見受けられます。

また、Replicator 2のエクストルーダには使っているうちに緩んだりといったトラブルがあり、ユーザの方が作った改良部品がThingiverseにアップロードされ[注2]、さらにはMakerbot自身も$8でアップグレード用のパーツを販売[注3]しています。どちらの方法も部品は3Dプリントして作る必要があります。ほかにもテーブルに歪みがあるといった不満も耳にします。Replicator 2はPLA(ポリ乳酸；3章で解説)のフィラメントを使った出力のみに対応しており、エクストルーダも1つだけです。

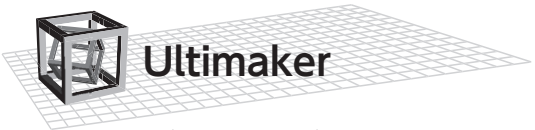
一方、最近発売されたReplicator 2XはPLAに加えてABS(共重合合成樹脂；3章で解説)のフィラメントを使用でき、フードがついてABSプリントを比較的快適に行うことができそうです。何よりもエクストルーダが2つに増えたうえに、前述のReplicator 2のエクストルーダやテーブルの歪みの問題が改良されているように見受けられ、なかなかに魅力的です。

このように、Replicator 2でも、まだ買ってきてノートラブルで使えるといった域には達していません。細々としたトラブルがありますし、それなりに試行錯誤をしなければうまく出力することはできないものだと覚悟の上で買う必要がありそうです。また、サポートは提供されていますが、Makerbot社のサポートとは英語でコミュニケーションを取る必要があります。

ユーザコミュニティを利用

ユーザコミュニティとしては、Japan Makerbot User Group[注4]といった集まりがあります。筆者が参考にさせていただいているブログの作者さんたちが始めたユーザグループで、いろいろと参考になる情報が載っています。

Replicatorは、Makerbot社のWebサイト[注5]からReplicator 2で$2,199、Replicator 2Xで$2,799といった価格で完成品が販売されています。



Ultimaker(P.15 写 真 23)は、オランダのFablab(ファブラボ)ユトレヒトが開発したRepRap系のオープンソースハードウェアの3Dプリンタです。それまでのMakerbotを始めとする各社の3Dプリンタの速度／品質に不満を持っていた彼らが、エクストルーダとフィラメント送り機構を分離したBorden方式を採用するなど、機構にいろいろと工夫をこらして開発したものです。

筐体はベニヤ板をレーザーカットして作られており、ガッチリとした作りになっています。ABSにも対応していますが、PLAを標準としています。フィラメント径は今の主流ではなくなりつつある感じですが、直径3mmです。

筆者はUltimakerを所有していますが、去年ニューヨークで開催されたMaker Faire 2012で

注2) http://www.thingiverse.com/thing:42250
注3) http://store.makerbot.com/drive-block-hardware-kit.html
注4) https://groups.google.com/forum/?hl=ja&fromgroups=#!forum/jmb-ug
注5) http://www.makerbot.com/

もらったUltimaker Robotサンプル(**写真2**)のできばえと、Make:誌が発行している"Make: Ultimate Guide to 3D Printing"[注6]という3Dプリンタ比較本を参考に購入を決めました。

Ultimakerは費用対効果という意味で、非常にバランスのとれた3Dプリンタだと思います。プリント速度、品質、加工サイズは実際に使っていて満足いく性能ですし、自分で組み立てるキットであれば、€1,194(オプション・配送料・関税は別)と比較的低価格で購入できます。筆者もUltimaker[注7]ショップからキットを購入して組み立てました(Maker Shed[注8]でも購入することができます)。

キットを組み立てるかどうか

RepRap系の3Dプリンタはまだまだキットで組み立てるものも多く、オンラインマニュアルも英語がほとんどですので、最後まで完成させられずに挫折してしまう人も多いと思います。Ultimakerも送られてきたキットは改良版で、オンラインマニュアルとはフィラメント送り機構の構造が全然違ったり、電気まわりの説明写真がリンク切れになっていたりと、ある程度組み立てに自信がある人でないと辛いかもしれません。組み立て済みUltimakerは€1,699ですし、スタンドアローンコントローラも付属してきますので、組立てが得意でない方はこちらがなかなかお買い得と言えるのではないでしょうか。

▼写真2　Ultimaker Robot

注6) http://makezine.com/volume/make-ultimate-guide-to-3d-printing/
注7) https://shop.ultimaker.com/
注8) http://www.makershed.com/Ultimaker_3D_Printer_p/mkum1.htm

筆者のUltimakerは半年ほど使っていますが、完成後の大きなトラブルはほとんどありません。たまにテーブルの水平出しをするくらいです。ネット上ではよく話に聞くエクストルーダノズル詰まりなどの機構的不具合は一度も発生していません。これはすばらしいことだと思います。

UltimakerはCuraが標準のスライサーソフトなのですが、これがまたできの良いソフトで、組み立て終わっていきなりのテストプリントで、なんの調整もなしに3クリックできれいにサンプルモデルのUltimaker Robotを出力することができました。こういうところも3Dプリンタを選ぶポイントの1つだと思います。

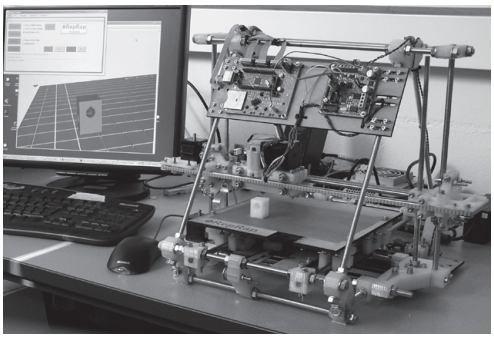
RepRap atom

この特集の記事でも、これまで何度か"RepRap"(**写真3**)に触れてきましたが、ホビー向けの3Dプリンタを語るうえでRepRapは外せません。RepRapの特徴の1つに、自己複製機械であるというものがあります。RepRapはフリー(自由)なプリンタで、GPLライセンスで設計情報が提供されています。このため、RepRapでRepRapの樹脂部品をプリントして、集めた部品と組み合わせることで新たなRepRapを作ることもできます。

RepRapや3Dプリンタを持っている友人がいなければ、RepRap Pro[注9]をはじめとする多く

注9) http://reprappro.com/

▼写真3　RepRap Version II "Mendel" machine



のサイトがキットを提供していますので、買ってきて組み立てるという選択もあります。

といっても、RepRapには相当数のバリエーションがあり、RepRapに手を出そうと思っても、どれを選んで良いのか悩ましいでしょう。日本人であれば、筆者は“RepRap atom”（**P.15写真24**；@arms22氏提供）をお勧めします。

自分で組み立てるには情報が必要

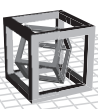
RepRap atomは、日本におけるRepRapのコミュニティであるRepRap Community Japan[注10]から発表された3Dプリンタです。atomに限らず、たいていのRepRapは自分で組み立てる必要があり、キットを買わなければ部品をそろえるところから自分でしなければなりません。キットでなく、RepRapのパーツを自分でそろえようと思うと、そこにはたいへんな苦労があります。パーツにはeBayで買わなければ手に入らないものがあったり、いつも売り切れなので入荷するのも監視していなければならないものもあったりします。

たとえばエクストルーダ1つをとっても、写真（**P.15写真25**；@arms22氏提供）のようにプリントした樹脂部品に買ってきたステッピングモーターにネジやナット、ワッシャー、スプリングなどを組み合わせて作る必要があります。こういった情報が交換されているのが、前述のRepRap Community Japanです。このコミュニティに頼らなければ、これから3Dプリントを始めてみようと思った人へのRepRapのハードルは相当に高いものになります。

出力品質は?

ここまで読んで、オープンソースでコミュニティで開発されているプリンタの出力できるものの品質なんて大したことはないだろうと思った方もいらっしゃるでしょう。しかし、Makerbotも、もともとはRepRap出身であることを忘れてはいけません。RepRapは使えるようになれば製品の3Dプリンタに遜色ないどころか、それ以上のパフォーマンスを発揮することもできます。また、オープンソースがゆえに、コントロールするソフトウェアの選択肢もとても幅広くなっています。たとえば、先ほどUltimakerのところに出てきたCuraはRepRapとともに使うこともできます。

atom[注11]は本稿執筆時点ではRC（リリース候補）版ながらも、このコミュニティの助けを借りて組み立てている人がいますし、また、組み立てワークショップつきのキット[注12]も147,700円で提供されるようになって、エントリへのハードルがぐっと下がってきました。ワークショップはいらないのでキットだけでも、という方にはキット[注13]が129,800円で販売されています。しかし、筆者としては、ワークショップつきキットのワークショップ分の価格は10,500円と内容と比べて格安であることもあり、RepRapではじめてみようと思う方にはワークショップつきのキットをお勧めしておきます。



まとめ

現時点では、ホビー向けの3Dプリンタでできることは似たり寄ったりなのが現状です。最近では光造形にも期待が高まりつつありますが、普及するにはもう少し時間を要するでしょう。そんな現状の中で3Dプリンタを購入するのであれば、やはりトラブルを相談できるユーザやコミュニティの存在が鍵になります。

そういった点では、Replicator 2は日本人ユーザも多く、比較的情報が収集しやすいのもメリットと言えます。RepRap atomはベータ版やRC版がリリースされたばかりですので、ユーザは多くありませんが、atomから3Dプリンタを始めた人も多くいます。Ultimakerについては、この3者の中では筆者の知る限り最もユーザの少

注10） https://sites.google.com/site/reprapcommunityjapan/
注11） https://github.com/hironaokato/atom_rc
注12） http://my.ebshop.info/
注13） http://genkei.thebase.in/

ないプリンタです。しかし、この3種の3Dプリンタを使う筆者の知人の中で、Ultimakerユーザが最もトラブルにあっている頻度が少ないということもあって、個人的には評価しています。

ここで取り上げたプリンタ以外にも、3Dプリンタメーカーの廉価版の機種、たとえば$3,588～と少し高価ですが3DシステムズのCubeX注14などは気になるところです。最近ではもっと低価格のCube(**写真4**)も国内代理店での取り扱いが始まったようです。

3Dプリンタを選ぶときには、カタログスペックや価格に目がいきがちですが、筆者なりの結論としては、まわりの人やコミュニティの具合、国内代理店の有無などを見て総合的に判断するのが良いだろうと考えています。また3Dプリンタは急速に発展する分野ですから、常に新しい情報を得るようにしてください。SD

注14) http://cubify.com/

▼写真4　Cube

Column ホビー向けフルカラー3Dプリンタ

フルカラーの3Dプリンタと言えば、ほぼ粉末固着積層法を採用しているZプリンタの独壇場で、ホビー向けプリンタでは見かけることがありませんでした。ですが、今年からは少しずつそんな状況が変わってきそうです。

たとえば、botObject社[注15]が開発しているProDesk3Dという3Dプリンタは、5色のPLAフィラメントを使って熱溶解積層法ベースの方法でフルカラーの出力を実現するそうです。さらにPVAでサポート材を出すこともできるようで、筆者は非常に注目しています。価格も$3,429と、ホビー向けの3Dプリンタの価格帯に収まっています。

注15) http://botobjects.com/

Column ランニングコストなどの話

ところで、3Dプリントをするにはどのくらいのランニングコストがかかるでしょう。3Dプリントをするのに必要な消耗品は、基本的にフィラメントとテープです。テープは必須というわけではないのですが、ヒートベッドのない3DプリンタでPLAのフィラメントを使っている場合などにテーブルを保護したり剥がせるように使います。

Ultimakerのオンラインストアで売っているフィラメントの値段を例にすると、PLAの白が0.75kgで$31.50(原稿執筆時で約4,120円)で売っています。3～4章で作るiPhoneスタンドの場合、出力の重量が30gほどですので、フィラメントの1/25、つまり165円程度の原材料コストで出力できたことになります。

皆、ランニングコストが気になるようで、Makerbot社は同社のReplicatorで実際に大量の3Dプリントをした結果を報告しています。同社のブログ記事[注16]によると、1kgのスプール(フィラメントのリール)で392個のチェスのコマを作ることができたそうです。使ったフィラメントはPLAのものだったようですが、Makerbot社では色によって価格が少し違うものの$43～48で売られています。約5,000円のフィラメントで392個のコマが作れたということは、コマ1つあたりの原材料単価は13円程度ということになります。

注16) http://www.makerbot.com/blog/2012/02/24/a-matter-of-scales-how-much-can-you-print-with-a-single-1kg-spool/

第2特集

システムを見通す力(眼力)でソフトウェア開発を楽にしませんか!

バグを狙い撃つ技術

社会統計・医学などの他産業の知見で進化するソフトウェアレビュー技術

「銀の弾丸は本当に必要ですか?」

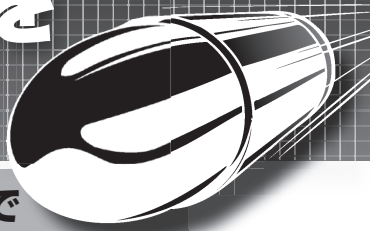
バグを見つけるために、ソースコードを読みます。そこに表現されているロジックをたどりながら、試行錯誤したり、ときにはデバッガの力を借りたりします。しかし、現在のソフトウェアはご存じのように莫大なファイルで成り立っています。一つ一つ探し出すのはたいへんな労力がかかるので、現実的ではなくなっています。

一方、そうした状況に対応するため、ソフトウェアの品質を管理するテストやレビュー技法が進化しています。本特集で紹介するQI法を適用すれば、コードを直接読まずにバグの原因を探ることもできます。まさに、バグを狙い撃つ技術です。たとえばソースコードをテキストエディタでソートしてみると、ELSE IFの数が合わない……、Try catch の例外処理ができてないなど、一目瞭然です。どこにバグがあるのか、すぐに見当がつけれらます。これはレビューテクニックのほんの一例です。本特集の内容を理解いただければ、システムを見通す力(眼力)を習得するヒントが得られるでしょう。そしてソフトウェア開発を加速し、品質も向上させてみませんか!

● 細川宣啓(ほそかわ のぶひろ)
日本アイ・ビー・エム株式会社 ソフトウェア事業部
Rational事業部ワールドワイドタイガーチーム　品質エンジニア

システムを見通す力(眼力)でソフトウェア開発を楽にしませんか!

バグを狙い撃つ技術



社会統計・医学など他産業の知見で進化するソフトウェアレビュー技術

Writer
細川宣啓(ほそかわ のぶひろ)
日本アイ・ビー・エム株式会社
ソフトウェア事業部
Rational事業部
ワールドワイドタイガーチーム
品質エンジニア

ソフトウェアの欠陥を見通す力とは

はじめまして。本特集を執筆する細川です。筆者はソフトウェアの欠陥・バグの研究をしています。主な経験領域としては、業務系システム開発、設計、要求分析を経験したあとに、品質保証部にも在籍しました。とくにプロダクト品質保証を担当後、品質管理技術や品質系ツールの技術者を経て現在に至ります。

本稿では、開発者が作り出したソフトウェアのソースコードや仕様書が、品質検証やメトリクス測定といったいろいろな分野の技術でどのように評価されるかを紹介します。

さまざまな品質評価技術の概観(ソフトウェアレビュー)を通じて、ソフトウェア開発に関わる技術者が、今後備えるべき「眼力」について紹介します。「眼力」とは「システムを見通す力」です。

ソフトウェアの成果物はどのように検査されるのか?

たとえば、レビューやソフトウェアテストの専門家は、開発担当者が作成する成果物をどうやって検査しているのでしょうか? あらゆる成果物を検査しますが、仕様書やコードを、単純に1ページ目から赤ペンを握り締めて「にらめっこ」しているのでしょうか? そんな悠長な品質担当者は今どきいません(もし実施しているとしたら開発そのものに相当な迷惑をかけているでしょうね……)。

昨今の開発エンジニアにとっては、「品質にスピードを持ち込む」という考え方が重要です。品質活動に時間と工数をかけずに低負荷で実践することが求められます。厳しいシステム開発のビジネスでは、旧来の負荷が大きい品質評価技術が市場で致命的なマイナス点となることも少なくないからです。

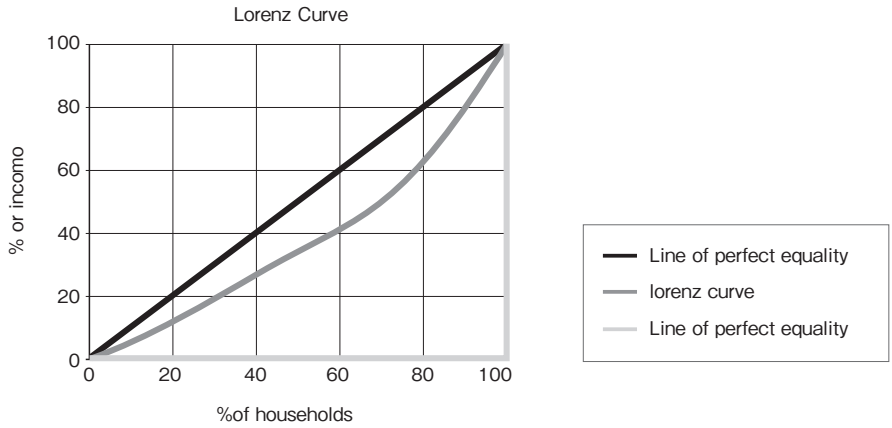
本稿で紹介するいろいろな品質エンジニアリングの技術を知り、開発者がどう観察され、どう評価されるのかを知れば、事前に開発側でも対応できます。それが実践できれば、もしかすると品質担当者を不要にすることさえできるかもしれません。さらには自信を持って製品やシステムをリリースできる、システム開発後半で混乱しないといったさまざまなメリットが期待できます。それが開発系のエンジニアにとって必要な「システムを見通す眼力」です。

眼力①「社会統計技術(ジニ係数)がIT分野に応用できる?

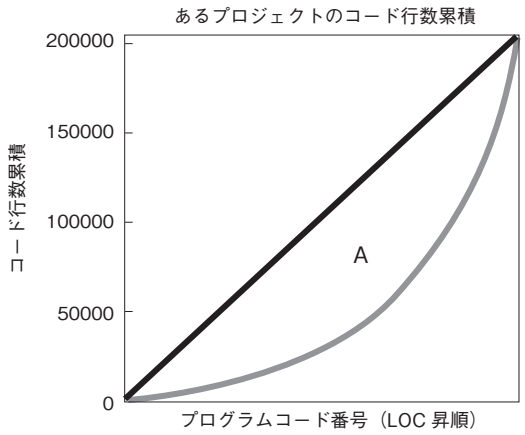
システム開発のプロジェクトが、社会統計技術で理解/説明できるとしたら、信じられますか。

社会統計などで用いられる所得分配の不平等さを測るメトリクスに「ジニ係数」[1]というのがあります。これはローレンツ曲線という「事

▼図1　所得分配の不平等さを示すローレンツ曲線[2]



▼図2　あるプロジェクトのコード行数の累積曲線



象の集中度合い」を把握するための曲線を基に、1936年にイタリアの統計学者コッラド・ジニによって考案されたメトリクスです。所得分配の不平等の測定に使うことで有名です。それ以外にも、ある社会の富の偏在性を俯瞰するといった使い方ができるそうです。

社会において1人1人の努力ではどうにも変化させることのできない統計データを収集して、社会全体の性質を俯瞰しようとする点、および定期的に測定してその変化を確認するという点において、わかりやすいメトリクスの好例です。

ローレンツ曲線による所得分配を示した例を**図1**に示します。

ところで、この社会統計とまったく同じことがIT業界でも観察できます。それはコード行数(以下LOC：Lines of Codeと呼びます)の累積グラフです(**図2**)です。まったく同じグラフのように見えませんか。

あるシステム開発プロジェクトで作成するソフトウェア規模をソースコード行数で表すことがあると思います。コード行数は工数見積もりや計画立案にも利用されるメトリクスです。**図2**を見てください。x軸にプログラムコード番号(LOC昇順)、y軸にコード行数の累積を描画

するとこのようなグラフになります。

(社会統計の用語を用いて説明すると)ジニ係数の線━と線▬で囲まれた面積Aが大きい場合、ある特定のプログラム1本に行数が偏っている、つまり「行数がある特定のプログラムに集中している」ことになります。

社会の中で所得不均衡が起きていると同じように、極端に大きな行数のコードが数本ある、ということになります。仮に全プログラムの行数が等しいとしたら、行数メトリクスに偏りがない、つまり「均衡している」状態になります。このとき、▬線が━線に重なり、ジニ係数はゼロになります。

プロジェクト現場の見積もりや規模把握の際、複数コードの行数のばらつきや偏りなどを考慮せずに単純総和や総和平均で行数を把握すると、本来の見積もりを大きく狂わせることがあります。また極端に品質が懸念される巨大な1～2本をプロジェクトの中から選び出す場合に、このジニ係数を計算しておくことですべてのプログラムを読む手間が省け、「おかしなサイズのコードが何本かあるだろうな」と推測できます。

カンの良い読者ならおわかりだと思いますが、前述のジニ係数というメトリクスはプログラム行数に限定しなくてもかまいません。コメントの含有率、もしくは単純にIFステートメント数に着目して当てはめることができます。

ITシステム開発のプロジェクトが、「同じ人間集団」の分析方法である社会統計の技術によって分析できるのです。全体を俯瞰して問題のある対象を絞り込む・抜き出す利用方法、さらには複数プロジェクトを相対比較する場合などでも、この技術は応用できます。

なぜ社会統計の手法がシステム開発に応用できるのか?

社会統計の手法がシステム開発の評価・測定に応用できる理由は、「人間の集団行動の特徴を観察する」ことが共通しているからです。

換言すれば、人間がプロジェクトという集団行動を行う中では必ず前述コード累積行数のような「数値に現れる何らかの兆候(＝サイン)」が現れます。しかもこれらの兆候はほぼ個人では回避することはできず、また測定・評価する専門家からは「品質の足がかり」になる恰好の材料になるのです。

所得の獲得にせよシステムの開発にせよ、集団が何か行動した際は、当該集団特有の「兆候」が顕在化します。集団特有の兆候をいかに適確に把握できるかが、開発者が備えるべき「眼力」として、昨今の開発エンジニアに必須の能力となってきました。人工物そのものの特性観察やそれを作り出す集団の兆候をデータから読み取る技術は、本稿のテーマである「眼力」として、必ず新しい世代の技術者となる読者の皆さんの武器になるでしょう。

眼力②「ソフトウェアのレビューを楽にするQI法」

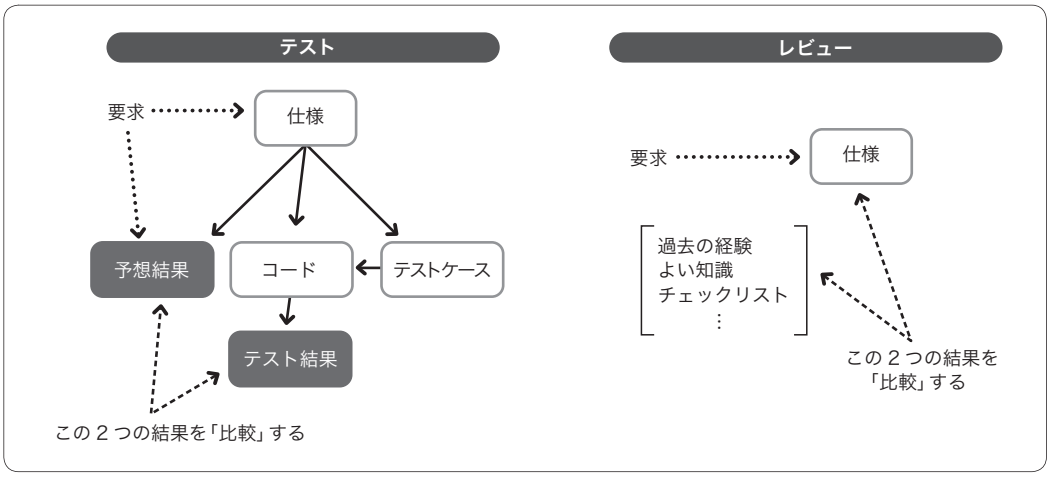
「システムを見通す眼力」の応用事例として、品質活動における「レビュー」を取り上げます。はじめにレビューはなぜ難しいのか？——というお話をします。

なぜレビューは難しいのか?

図3を見てください。一般的なレビュー手法は自由度の高さ、開催・実践工数負荷の高さなどから、プロジェクトでの実施が形骸化している品質活動といわれています。2つの結果(予想結果とテスト結果)を突き合せ・比較処理をするテストと違い、単位が存在しない品質に対する活動の有用性が定量分析しにくいとも言われています。これらの理由から、レビューは長い間「適用の難しい」技術とされていました。

さらに大規模プロジェクトにおいて上流仕様書の全ページをレビューすることや、作成コードの全行を目視で行うレビューは非現実的な労力と時間を要し、プロジェクト全体コストを大きく押し上げる原因になるとさえ言われています。

▼図3　テストとレビューの違い



QI法をご存じですか?

このレビューの実施負荷を軽減し、効果を最大化するためにはどうすればいいでしょうか?

その答えはレビュー実践以前に**メトリクスデータ測定を活用**することにあります。

つまり、全成果物の中に欠陥が混入している位置や欠陥種類を事前に把握できれば、重点的にレビューを行う対象を絞り込むことができ、欠陥の検出効率および品質活動の費用対効果が圧倒的に向上します。これがQI法(Quality Inspection)です。

さらにプロジェクトを数十から数百実施する企業・組織では、全プロジェクトにレビュープロセスを義務付けていることも少なくありません。仮に集中的にレビューを実施する必要があるプロジェクトがわかっていれば、限られた品質保証担当者の工数を集中投下できるようになります。

QI法を実践すれば、大量の仕様書を1ページ目からレビューする必要がなくなり「このあたりに欠陥がありそう」と欠陥混入確率の高い個所だけを拾い読みできるようになります。

またコードレビューの際は「このコードにあの種類の欠陥が入っている“はず”」ということがわかり、何十本もコードをエディタで開いてレビューをする労力が削減できます。換言すれば、仕様書を開ける前に欠陥の種類と場所が特定できると言えば良いでしょうか。QI法について概念を**図4**に示します。

この技法は開発側としてセルフチェックを行う場合でも比較的導入・学習コストは小さく容易に習得できるでしょう。また品質保証の担当者にとっては短時間で全体傾向の把握と欠陥混入確率の高い個所を特定でき、効率の良い品質検査が可能になります。さらにレビューに限らずテストやリリース判定においても同様の分析は有効です。

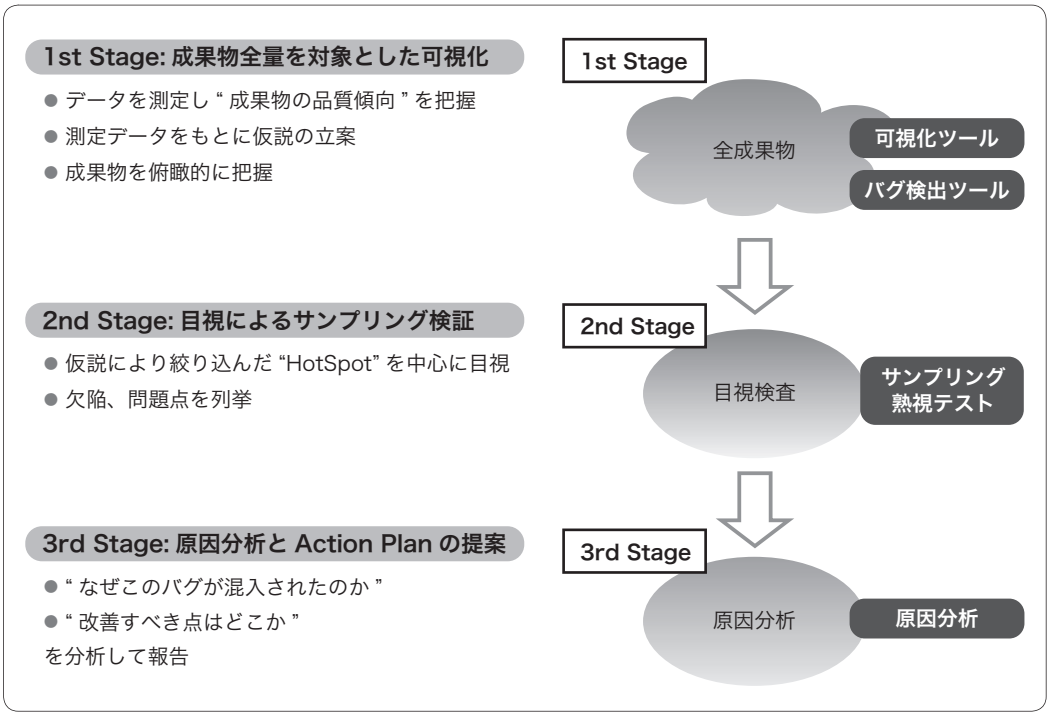
QI法が超高速レビューを可能にする魔法とは

では品質全般について「最低限の品質を最速で」手に入れることができる**魔法**を体得してみませんか。QI法では、とくにバグの位置と種類を効率的に特定するために、**図5**のようなプロセスを採用しています。

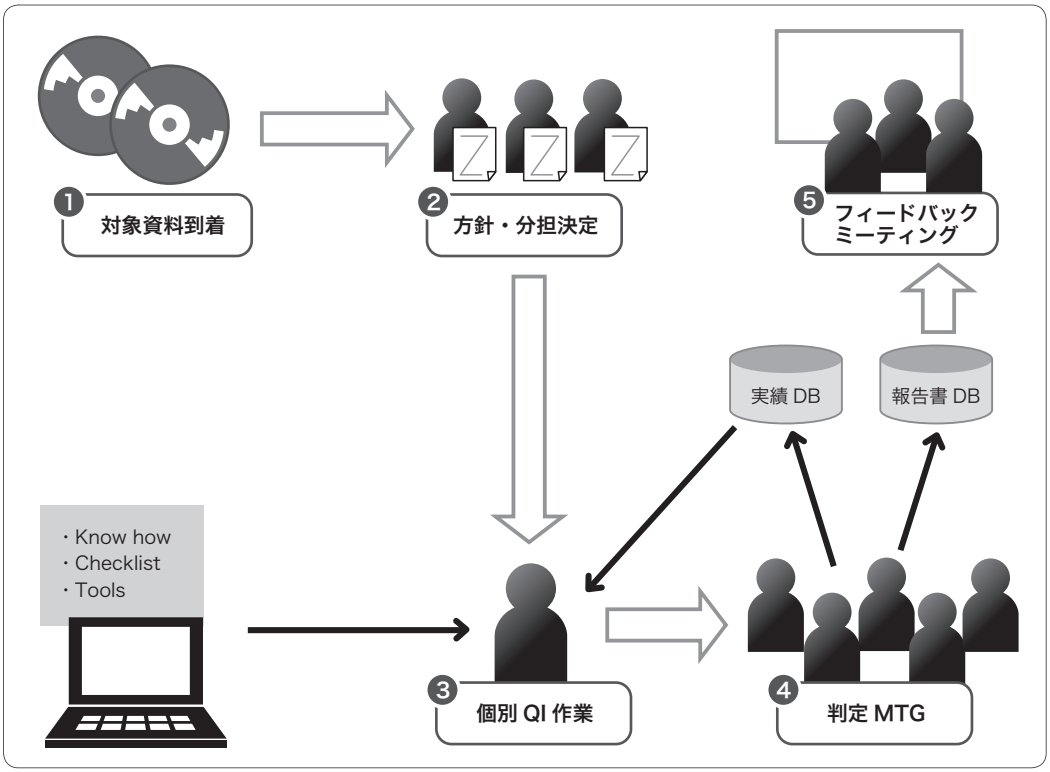
対象資料が届く前にQIは始まっている!

レビュー担当者は、システム開発に関わる資料を受け取ります。まず、①の対象成果物を「受け取る」収受作業からです。とくにQIは第三者品質検査機関として「品質のレビューを実施します。レビュー対象資料を検査組織に提出してください」とお願いしてから、対象資料が到着

▼図4　QI作業の概要



▼図5　QI実施プロセス



するまでの時間を測定しています。

サーバ上での成果物の管理、とくに作業者個人の最新版の把握などは提出までの時間に影響します。レビューの専門家は資料の到着までにかかる時間から、もうすでに次のような「仮説」を立てているのです(**表1**)。

資料到着時間とトラブルプロジェクト件数の不思議な一致

上記に示した「仮説」の一部または全部が該当するということは、おそらく「作成成果物のセルフチェックなし」「提出時の完了基準なし」「開発者間の整合性確認なし」である確率が高いのです。したがって対象資料には**表1**中の「混入欠陥(予想)」に示した欠陥が混入する可能性が高いと予測します。

無論、この到着時間は組織やプロジェクトなど企業ごとに基準値は異なります。しかし面白いことに、多くの場合正規分布(ポアソン分布)を示す点が特徴的です(**図6**)。

レビュー対象の「資料の平均到着時間」というメトリクスはプロジェクト管理の十分性と関連があるのか、チーム・コミュニケーション量に関連するのか正確な研究はまだ十分になされていませんが、前述の所得不均衡を示す例と同様に、集団の行動を把握する測定メトリクスとして興味深い関連を持っていそうです。

方針・分担決定

次に②の方針・分担の決定です。ここでは欠陥に関する仮説を立案します。

まず資料が到着しても、最初に対象資料の1ページ目からレビューをいきなり始めることはありません。目視で見始める前に、必ず対象のデータを測定します。測定メトリクスはプロジェクト全体の傾向を示すプロセスメトリクスやプロダクトメトリクスなどさまざまですが、まずは測定負荷のかからない簡便なメトリクスを数十種類取得して全体傾向を把握します。

表2、および**図7**は、プロジェクト全体で今回の検査対象資料を「どうやって作ったか？」という時間的経緯を把握するためのメトリクス測定結果を可視化したものです。

成果物の最終更新日付から見通せるヤバイポイント

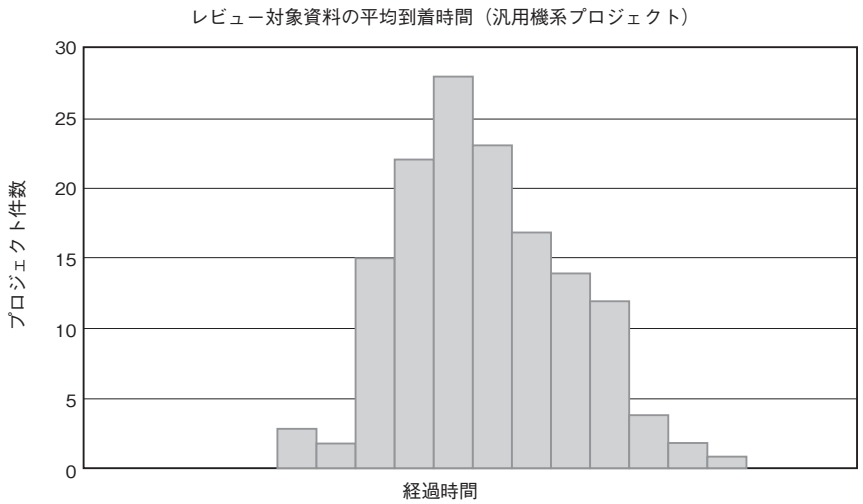
先の**図7**からわかることは、各フェーズ終盤ギリギリに成果物を更新している点です。そのため、品質レビューやセルフチェックを行っていない可能性が高いことが推測されます。欠陥の種類としては、未決事項が残存していないかレビュー時に目視確認すべきことです。

次に深夜作業の形跡が観察されます。よって「す

▼表1　メトリクス―兆候-仮説の定義例1

項目	内容
メトリクス	検査対象資料の到着時間
定義	レビュー実施を宣言してから実際の資料到着までの時間間隔
基準値[例]	3日以上(ただし、組織と体制、プロジェクト規模に依存する)
兆候	PM・リーダークラスが提出を宣言してから提出が遅れる／時間がかかる
仮説	・資料提出を宣言してから最新版を記述させる(プロセス管理不備) ・資料提出を宣言してから最新版をサーバに提出する(品質管理不備) ・全作成対象資料(ファイル枚数・コード本数)などを把握していない(予実管理の不備) ・品質担当者・責任者不在(組織・体制定義不備) ・サーバからの抜き出し・バックアップ手順定義なし(運用プロセス不備)
混入欠陥 (予想)：	・成果物間の論理不整合(作成者間不整合、作成時系列の論理不整合) ・誤字脱字、書式などの軽微欠陥(これは実際の検査では無視または警告に留める) ・バージョン間不整合・デグレード ・データ・処理などインターフェース不整合

▼図6　レビュー対象資料の平均到着時間(汎用機系プロジェクト)



▼表2　メトリクス―兆候-仮説の定義例2

項目	内容
メトリクス	成果物の最終更新日付＋最終更新時間
定義	成果物を最終更新した日付／時間
基準値	なし(全体傾向分析＋アンチパターン認識のため)
兆候	合計16のプロセスアンチパターン
仮説	プロセス欠陥混入のアンチパターンが存在しないか確認する
混入欠陥	・深夜作業によるによる欠陥＝成果物間の成果物不整合(コミュニケーション起因) ・要求仕様ー設計仕様間の追跡可能性欠如(進捗上間隔が開いている場合) ・フェーズ終盤の徹夜・深夜作業＝直前改定のケアレスミス(レビュー不足起因) ・フェーズ終盤に未決事項・保留事項残存(レビュー不足に起因)

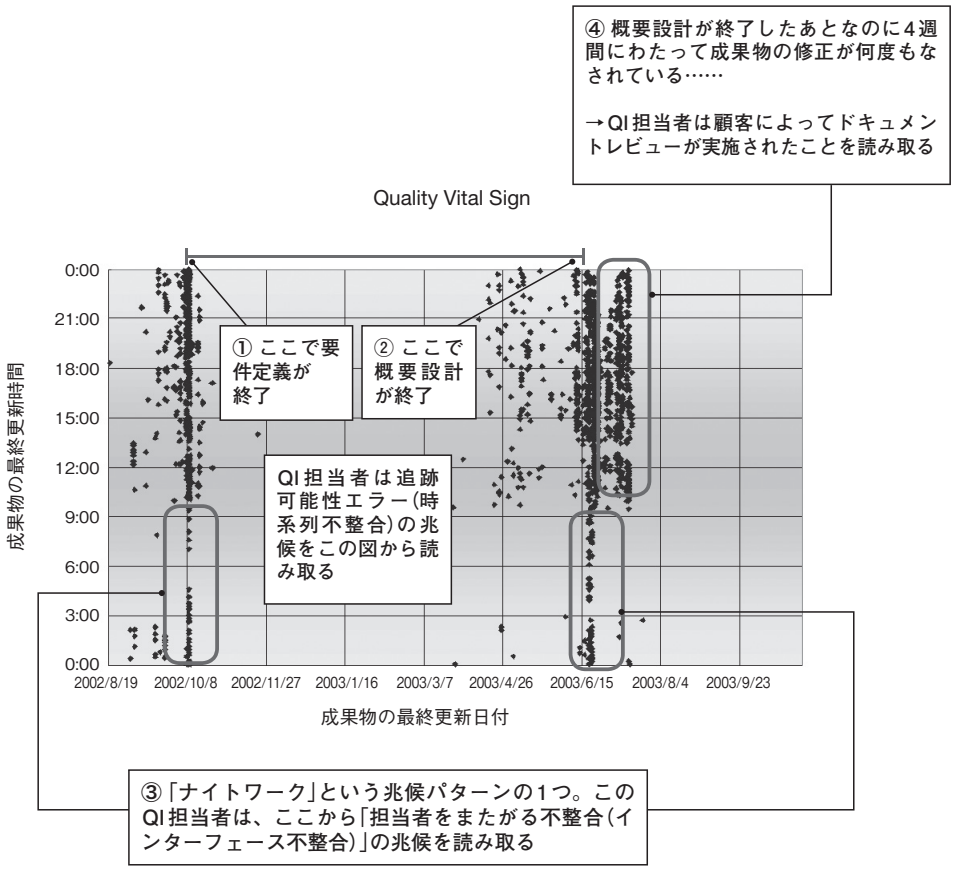
り合わせ」と呼ばれる仕様内容の確認が不足している可能性が読み取れます。なぜなら真夜中ですから確認チェックせずに各自仕様書作成作業に没頭してしまう場合が多く、他人との仕様の整合性について確認することを怠る場合があるからです。最後に、**図7**中で左右に大きな成果物の塊が観察され、両者の間が8ヵ月も仕様書が作成更新されていない点からは、追跡可能性の欠如が懸念されます。

これらグラフによる兆候把握の重要なポイントは、成果物の「最終更新日付」や「最終更新時間」という一般的には品質メトリクスではない点を活用していることです。

欠陥密度や試験密度といったクラシカルな品質メトリクスは測定負荷が大きく、多くは「欠陥が混入した痕跡」を示すもので、いわゆる「事後測定メトリクス」と呼ばれる作業完了後に測定するメトリクスです。欠陥が入り込んだ後にメトリクスを測っても「もうどうしようもない」のが実情です。

本来、レビューなどの品質検査の事前調査・傾向把握に用いる測定するメトリクスは、時系列変化の途中でも検出できる値でなくては大きな効果は望めません。**図7**の最終更新日付の例

▼図7　成果物の最終更新日付+最終更新時間



▼表3　キーワード数と集合図による仮説立案

項目	内容
メトリクス	コード上のWordカウント+基礎メトリクス
定義	コード上のキーワードカウントから混入欠陥予測
基準値	なし(対象コードの言語・プロジェクト人数・プロジェクト規模に依存)
兆候	平均値から残渣(=偏差)値が突出しているものが存在する
仮説	図8参照

は、プロジェクト進捗に伴って毎日のデータを測定でき、「パターン出現兆候」をとらえることができる点が注目に値します。

皆さんの組織・プロジェクトではどんなパターンが出てくるのでしょうか。ぜひ調べてください。

兆候から仮説立案までのどのような思考があったのか?

次に欠陥もしくは品質傾向全体の「仮説」を立案します。大量のソースコードが手元に到着した場合を例に、品質検査に先立ち仮説立案する思考方法を紹介しましょう。

▼図8　メトリクスから仮説を立案し、対象を絞り込む思考(集合図表現)

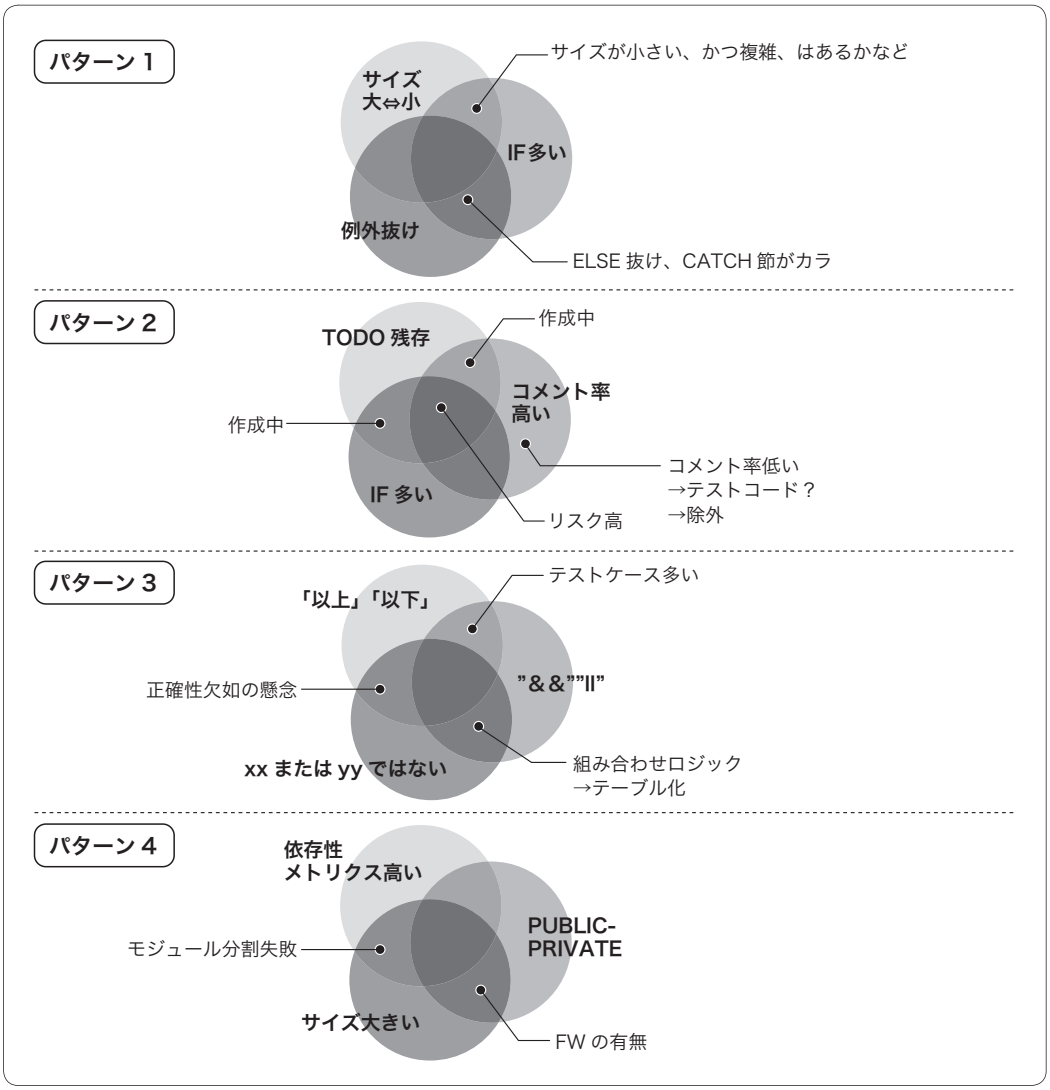


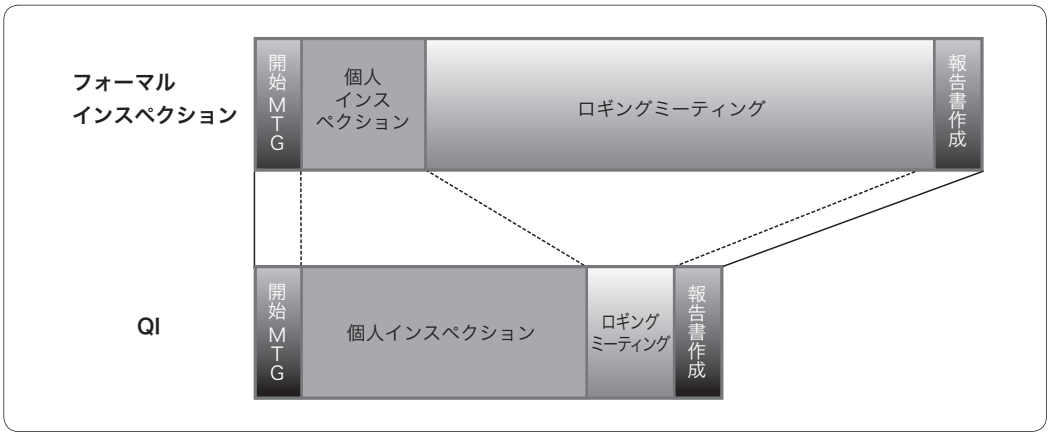
図8をご覧ください。**表3**測定したデータをどのように読み取り、仮説に結び付け、検査対象を絞り込むかの思考をベン図で示したものです。大量のコードの中から欠陥の混入確率の高い部分を選択抽出するには、これらの思考パターンをいくつも組み合わせて利用することで、より早く効果的なレビュー実施が可能になります。

詳細に見てみましょう。たとえば[**パターン1**]では、コード行数の大きさとIF分岐条件数、および例外処理(たとえばJavaのTryとCatch節の数)の3つの値を組み合わせて、コードサイズが小さく複雑なクラスは、単体テストケースが増加し、全般的に分岐処理に関するテストケース設計が難しくなるものを抽出する思考方法です。

[**パターン2**]では、複雑なクラスの作成途中であり、仕様が「十分に検討しきれていない」パターンや、テストケースが不十分になるリスクを内包している候補として注意が必要な対象を抽出できます(未確定や要検討の機能＋複雑度が高い＋コメントが改変のたびに追記されるため)。

さらに[**パターン3**]および[**パターン4**]では

▼図9　レビュー時間短縮・効率化は個人インスペクションの使い方



あいまいさの残存する設計や機能不備、標準違反や標準そのものの不備を示唆するプロセス不備起因の欠陥などを抽出する場合の絞り込み基準です。

実際は表計算ソフトのフィルタなどを利用してこの絞り込み・特定の作業を行うのですが、パターンとしてこのような集合図(ベン図)で保存している点が重要です。すでにあるベン図を用いて、

メトリクス→兆候→仮説→検査対象選択

の思考を記録する方法は、メトリクスデータ測定作業を自動化しておけば、パターン集の利用により未経験者でも高確率でバグにたどり着くことができる、ということが筆者の実験でもわかっています。

現実には、測定したメトリクスデータと兆候から、立案した仮説、実際の検査結果である欠陥含有率等を「セットで」記録して当該思考ノウハウを移転します。これは、医学のカルテの書き方を模倣しており、より複雑な思考パターンに対応できるという利点があります。

典型的なパターンを知っておくことで、検査対象範囲を狭めてレビューを実施できるようになるため品質活動の効率と効果が飛躍的に高まります。

このような集合図などを用いた品質観点・思考の保存は——あたかも医者がカルテを記載するように——思考過程／方法を他者に保存・共有・伝達可能となり極めて価値の高い企業資産となり得えます。

個別レビュー実施であぶり出される欠陥

仮説立案のあとは個人レビューを実施するわけですが、個人レビューに重点を置き、集中的にレビューを行うことで、重大欠陥だけ次のロギング・ミーティングに送り、負荷を軽くできます。

フォーマルインスペクションとQIの違いを**図9**に示します。

一般的なレビュー会議(これをロギングミーティングといいます)の負荷を極端に少なくしていくことで、レビュー全体の実施負荷を軽減できます。この時点で各自の検出欠陥のリストを照合します。レビューを分担した全員が指摘した個所(論理積)および1人だけが指摘した個所(論理差)を集計し、全体傾向の収集と、アクションプランリストの作成不足個所、および調査個所(担当個所と調査不足個所から追加検査個所)を特定します。

このあと、必要に応じて再検査／追加検査を行います。予定時間の残15%を上限目安として追加／再検査を実施するのですが、そのときもすでに検出した偏在する欠陥は、本当に偏在するのか？(全体の共通の欠陥ではないか？)、

すでに検出した重大な欠陥が、よそでも確認できるか？――などのクロスチェックを実施します。

注意すべき点は、再検査の指示は、既出欠陥に対してよそでも検出されるか？――という観点で追加を依頼します。検出した欠陥数が少ないからといって指示してはいけません。

ロギング・ミーティングの実施で突き合わせる欠陥の絞り込み

集合形式の検出欠陥を突き合わせる会議のことをロギング・ミーティングといいます。ここでも検出された欠陥を分類集計する際に定量データを用いて定性的な傾向分析を行います。たとえばレビュー担当者が4人いたとします。全員が気づいた欠陥は集合として「論理積(AND欠陥と呼ぶ)」、ある担当者1人だけが気づいた欠陥を「論理差(XOR欠陥)」と呼びます。このとき、AND欠陥は担当全員が気づいたため検出難易度の低いものである場合が多いのです。

仮に開発側メンバもレビュー(＝通称セルフレビュー)を実施したあとで、AND欠陥が残存しているとしたら、そのセルフレビューは「効果がなかった」ことになります。逆にXOR欠陥が多数検出された場合、検出難易度が高い欠陥がまだまだ残存している可能性を示唆します。そのためレビュー期間の延長や補足のためにより品質活動の充実を検討するわけです。

このようなロギング・ミーティング席上での検出欠陥の「突き合せ」を通じて、全体傾向のすりあわせを進めて最終の「フィードバック・ミーティング」に向けた準備を行います。

具体的には準備として検出欠陥(＝指摘欠陥)について**図10**のような確認を行います。

フィードバック・ミーティング実施

レビューは開発者に対して検出欠陥に関する知識移転を行う場であり、「改善を促す」活動です。大量の欠陥リストを開発者へ返送することよりも、実態に応じて現実的な指摘を返すべきです。

ところが、国内のさまざまな企業での品質活動を見ると、レビューの最終目標が欠陥リストを送り付けるだけになっているシーンが少なくありません。フィードバック会議が上手に開催されていないことに起因します。**図11**の4点はフィードバック会議の開催前に必ず確認すべきことです。

せっかく良い欠陥検出ができたとしても、開発者側にそれを除去してもらわなくては意味がありません。いわゆる「飲めない指摘」をたくさん出すことは工数の無駄になります。ここでもレビュー指摘欠陥リストのうち、いくつ(＝欠陥個数の場合と欠陥の種類数の場合があります)

▼図10　検出欠陥のチェックリスト

チェックリスト例

- ☐ 指摘の優先順位付けを行う
- ☐ 指摘実施にあたって障害となる制約事項がないか確認する
- ☐ 指摘実施にあたって障害となる対外契約条項はないか確認する
- ☐ 指摘事項の実施順序を検討する（依存関係／他欠陥誘発リスクはあるか確認等）
- ☐ カウンタークレームは想定できるか？
- ☐ 誤検出の可能性はあるか？
- ☐ フィードバック対象者は誰か？　フィードバック時期はいつか？

の欠陥が修正されたかを測定することで、そのレビュー活動そのものの価値を測定するとともに、仮説立案や前半でのメトリクスデータ測定がどの程度有効であったかセルフ・フィードバックの源泉として利用します。次回は今回より高精度の仮説立案を短時間で実施する目的で、このような自己最適化を継続して組織的に成長を目指します。

以上のように、最低限のメトリクスデータから兆候を読み取り仮説立案を行うことで、その仮説をもとに混入欠陥の位置と種類を特定します。この手法は比較的あらゆる分野(たとえばテストやプロジェクト管理レビューなど)に応用ができます。そしてその効果を損なわずに効率を飛躍的に向上させることができます。

ちなみに、一般的なレビュー技法と比較してQI法では1/16程度の工数でほぼ同等の効果得た調査結果もあります(**図12**)。

眼力③「個別・個人の行うレビューテクニック」

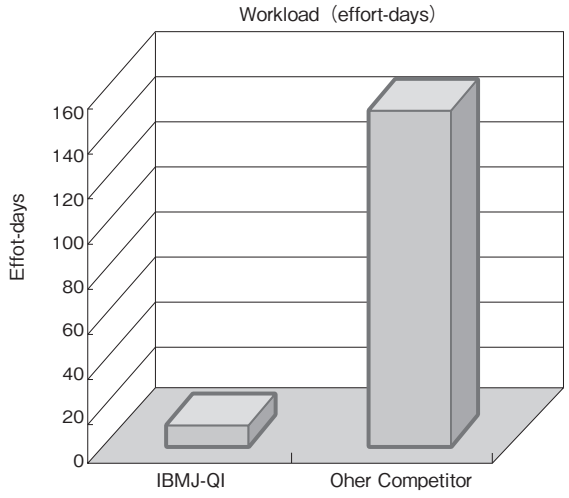
前述のQIのようなレビューテクニックでは

▼図11　フィードバック会議開催前のチェックリスト

チェックリスト例

- ☐ フィードバック会議を開催／招集する（べき）人を間違えていないか？
- ☐ フィードバック会議の目的は定義しているか？
- ☐ 検出欠陥を一言で言えるか？（長い欠陥の説明になっていないか？）
- ☐ 事実に基づく指摘を行っているか？（推測が含まれていないか？）

▼図12　QI法の生産性と効果



A) Workloads needed
- Other Competitor **150** effort-days.
- IBM-Japan QI team **9** effort-days.

B) #of defect type detected
- Fomal Inspection **64** type of defects detected.
- Quality Inspection **62** type of defects detected.

大量の対象資料から目視対象を抜き出す技術が中心でした。じつは単一の対象をレビューするテクニックも多数存在します。決して数値にだけ頼るわけではありませんが、よく観察する行為という意味では、効率的な品質活動時間の短縮につながる場合が少なくありません。

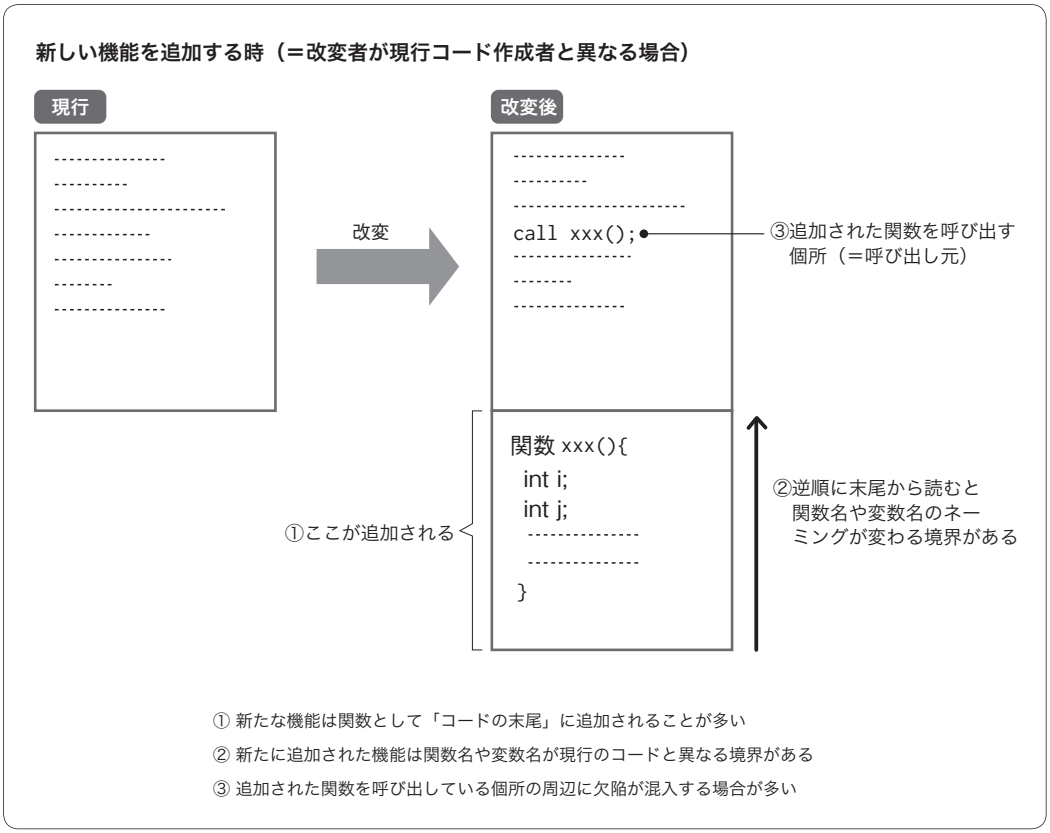
コードはお尻から読む

最も時間と工数を要するのがコードレビューです。レビューの際に有名な「拾い読み」をする方法です。一般的にコードファイルをエディタなどで開いて先頭1行目からていねいに記憶しながら読んでいる人がいますが、これでは時間がかかり過ぎます。

では、どうコードリーディングするのか？――実は拾い読みすらしません。**図13**に示すとおり、一般的に最終の改変個所はコードの末尾に追加される性質を利用します。よって、コードを最終行から逆順にリーディングします。そこで「既存コードと新規追加部分の変数名や関数名のネーミングルールが変わる「境目」や「境界」を探します。

この境界以降が新規追加された部分だとしたら、それを呼び出している個所を既存コードの中から探します。つまり新規追加部分の呼び出し元周辺に欠陥が入り込む可能性が高くなるわけです(機能挿入部分の前後・近接部分は既存コードを十分に理解していなければ挿入できませんので)。この**"追加がコードの末尾に来る性質"**を利用すると、全行を読み直すこと自体不要になることも多く、機能拡張や派生開発の際にはとくに有効です。

▼図13　眼力テクニック「コードは先頭行から読まない。改変・追加時は末尾から読む」



コードを絵画的に俯瞰する

Microsoft Wordなどにソースコードを貼り付け、編集画面を「ズームアウト」表示すると全体が見えるようになります。そうすると制御構造が俯瞰しやすく、どこからコードを見るべきかが把握しやすくなります。主な検査観点を**図14**に示します。

ズームアウトすることで、対象コードは内容が読めない単純な「図形」になります。マクロな視点では、図形として黒塗り個所が見えてきます。この黒塗りが多い個所は処理が多い場合が多く、インデント（＝入れ子）個所は分岐文やループ文が多く、テストしにくい個所が「どの程度」あるかが一目瞭然になります。さらに同様のパターンが「同じ形状」として何度も繰り返し出ているところも見られることです。これはコードをコピー＆ペーストしているのがわかります。

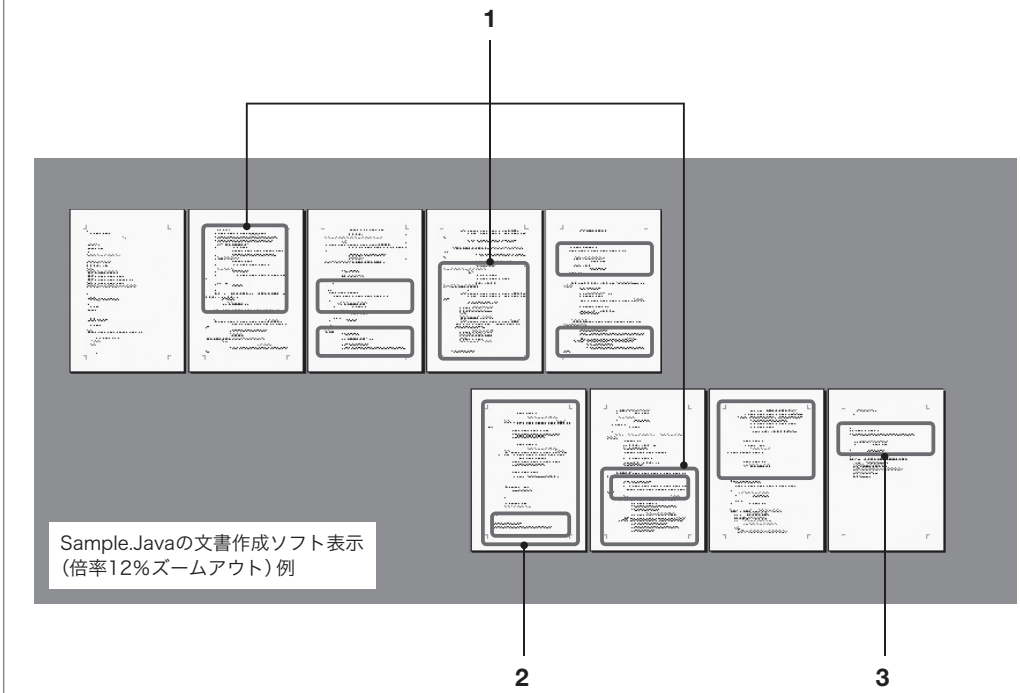
つまり、先頭行からコードを読まなくても、全体のコード構造を視覚的にとらえて重点を置いてレビューします。慣れてくると「一見して」おかしいコードの形というのも見えてくるようになります。

コードをソートする（プログラムコードの濃淡を観察する）

コードを各行の文字数で降順ソートすると、コピー＆ペースト状況や処理の抜けを検出できる場合があります。

ソースをソートした結果からは、①複数行に渡ってコピーした形跡、②コメントレベルで条件分岐が漏れている可能性、③ロールバリュー値が正しいか不明かといった観点でレビュー開

▼図14　眼力テクニック「コードを絵画的に俯瞰する」



検査観点：
1. 黒いところ=処理が複雑で重たい可能性が高い
2. 右にへこんだところ=ネストが深い構造・複雑な分岐やループの可能性
3. メソッドの塊は見えるか？=メソッドコメントの存在とその統一を観察

始時の候補が特定できることがあります。場合によっては仕様の抜け漏れを引き当てることもあります。

また、IF文やFor文といった主要な制御キーワードが塊として表示されることで、網羅性を把握したり、変数名のネーミングルールの違いやプライベート変数の違いなどから何人くらいが該当のコードを修正・変更したか？――なども直感的にとらえることが可能です。

◆　◆　◆

今回示したこれらの技術は、いずれも「ぼんやり」と対象を俯瞰したり、対象の「細部を注視」する技術です。これらは1つだけを行うのではなく、同一担当者が組み合わせて使うと良いでしょう。

いずれの手法もバグ票を起票する、または整理してデバッグする(またはデバッグを指示する)ための情報としては必ずしも十分とは言い難いものです。しかし、レビューの中で圧倒的に欠陥にたどり着く時間が短くなる効果が確認

▼図15　コードをソートする(ASCIIソート例)

```
public class childrenxxxx implements parentsxxxx {
  /** 工程機能カレンダー DAO */
  private xxxx_koutei_calendarDAO xxxDao;
  /**
   * 処理基準日取得
   * 実績管理画面、作業状況管理画面で呼び出すのメソッド
   * 入力パラメータ.集計単位フラグの値を元に後続処理の判断を行う。
   * 入力パラメータ.処理タイプの値を元に処理基準日を算出する。
   * @param 入力年月日Date
   *              入力年月日Date
   *              業務共通の共通部品を用いて取得したオンライン日付、
   *              もしくは、セッションから取得した前処理基準日
   * @param 集計単位Flag
   *              集計単位フラグString
   *              0:日次処理、1:週次処理、2:月次処理
   * @param 処理Type
   *              処理タイプString
   *              0:当日処理、1:前週(月／年)処理、2:次週(月／年)処理
   * @return Date
   *              処理基準日
   */
                :
                :
```

ASCII降順ソート

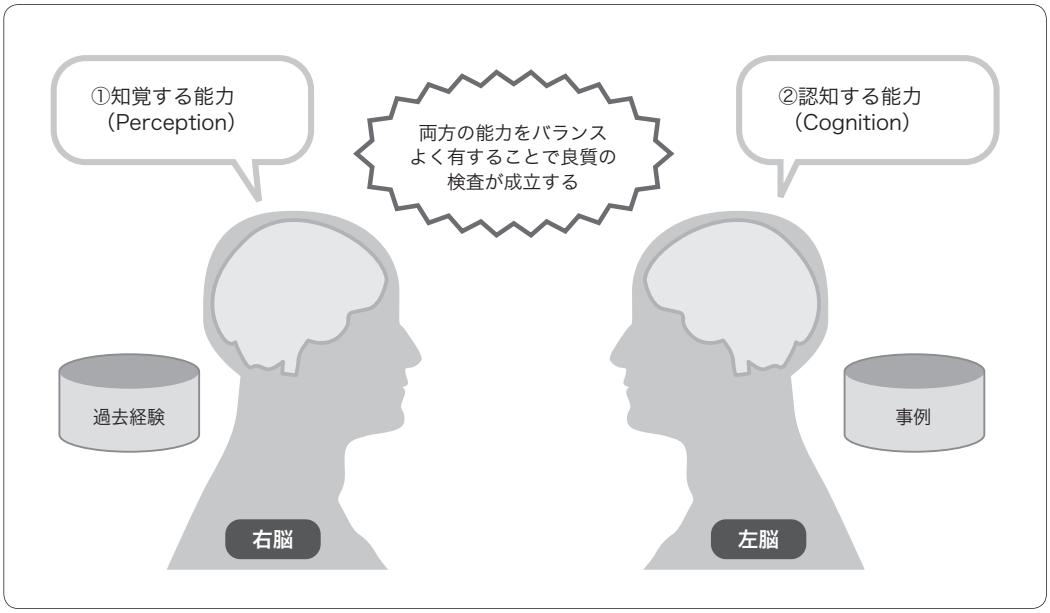
```
// 目的:工程機能カレンダーを元に次週(月／年)の
// 目的:工程機能カレンダーを元に次週(月／年)の
// 年度初め開始日を算出する。
// 日次の処理基準日
// 点検実施年月の開始日を算出する。
// 点検実施年月の開始日を算出する。
// 処理基準日を算出する。
// 処理基準日を算出する。
// 処理基準日=入力年月日の前週の月曜日
// 処理基準日=入力年月日の週の月曜日
                :
                :
// 2-3の処理へ 週次の処理基準日を取得
// 2-3-1-2. 入力パラメータ.処理タイプ = '1'(前週(月／年))の場合
// 2-3-1.【入力パラメータ.処理タイプの値によって以下の処理を分岐】
// 2-3. 週次の処理基準日を取得
// 2-2の処理へ 日次の処理基準日を取得
// 2-2-1-2. 入力パラメータ.処理タイプ = '1'(前週(月／年))の場合
// 2-2-1-1. 入力パラメータ.処理タイプ = '0'(当日処理)の場合
// 2-2-1.【入力パラメータ.処理タイプの値によって以下の処理を分岐】
// 2-2. 日次の処理基準日を取得
// 2-1-3. 入力パラメータ.集計単位フラグ = '2'(月次処理)の場合
// 2-1-2. 入力パラメータ.集計単位フラグ = '1'(週次処理)の場合
// 2-1-1. 入力パラメータ.集計単位フラグ = '0'(日次処理)の場合
```

されています。いわゆる「現場の知恵＝エンジニアリング」といったものでしょうか。

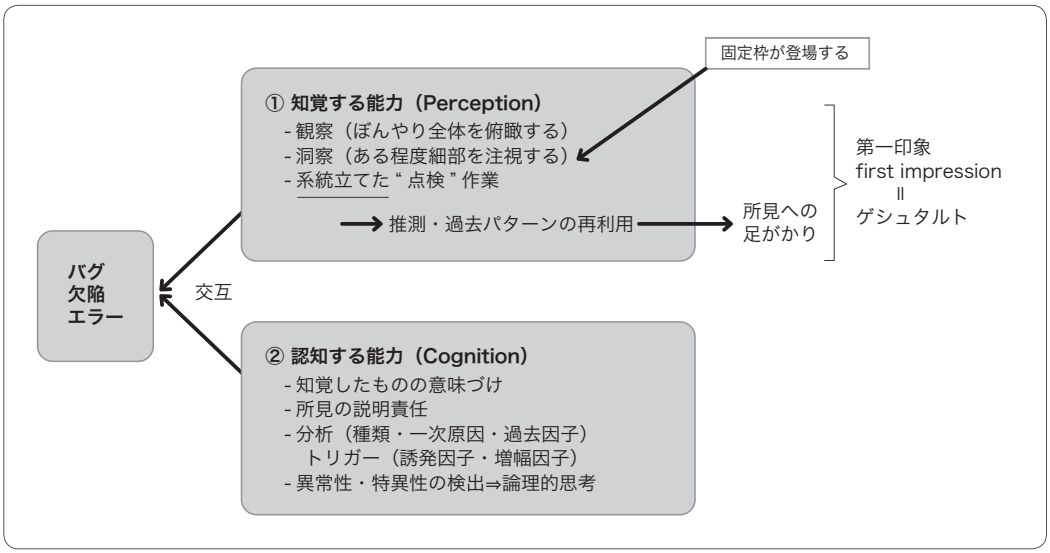
読者の中にはこれらのテクニックを読んで「なるほど」と共感された方もいらっしゃるかもしれません。これらの技法が理論として確立された方法でないとしても感覚的には効果が期待できそうです。

では、なぜこんな非論理的で現場的な技術が有効と感じられるのでしょうか？　そこには、実は品質活動、とくに欠陥検出に関する人間の能力獲得の科学が存在します。

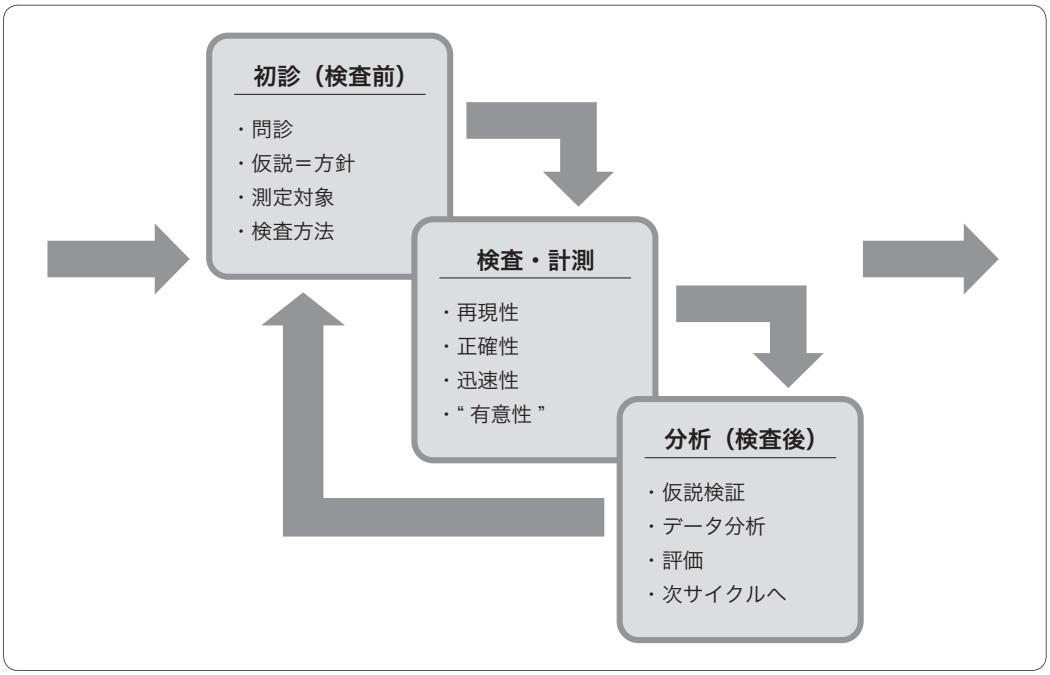
▼図16　欠陥を検出する2つの能力「知覚と認知」



▼図17　知覚と認知の定義



▼図18　医学の「診察」のサイクル



欠陥を検出する能力「知覚と認知」

一般的にテストやレビューでの欠陥検出能力は、単にバグを探し当てる能力と思われています。欠陥検出能力は技法をたくさん学習すれば向上し、優秀なテスターやレビュアになれる、と考える方もいます。

しかし、実際には違います。欠陥を検出する能力は、次の2つの能力で構成されます(**図16**、**図17**)。

●①知覚する能力(Perception)

観察(ぼんやりと全体を俯瞰すること)や洞察(ある程度細部を注視)すること、系統立てた点検作業を通じて「欠陥がそこに存在すること」を感覚的に対象をとらえること。経験的学習や推測・過去パターンとの照合によって実現

●②認知する能力(Cognition)

知覚した欠陥の意味づけ、所見を確立する作業などを含む分析作業(種類や位置)。とくに異常性や特異性を検出する際に論理的思考を通じて「存在した欠陥そのものを理解し説明できる」ようになること

②の理性的、論理的思考を強化することは、検出した欠陥を分類し、整理することであり、時間をかけて学習することで身に付けやすい能力と言えます(日々出会った欠陥を記録し「バグノート」を蓄積することで、他人と欠陥を交換することでもある程度学習ができます)。

ところが「バグに気づく力」すなわち①の知覚能力は特別な訓練でもしない限り急激に向上することはありません。本稿の主題である「眼力」は、実はこの①の能力のことです。

一般的に手にはいりにくい能力である①の知覚能力を補う方法として定量データを用いるのです。

兆候把握の重要性「定量データ測定を併用する医療現場」

医学の世界では、医師は症状の確定のために

問診と観察を行います。問診した情報を手がかりに無数の病気の中から2〜3の候補に絞り込みを行います。

このとき医師は考え方として「消去法」で思考するそうです。つまり「この情報からはこの兆候からxxという病気の可能性はない」と消去することで膨大な数の病気からピンポイントで特定の病気を見つけ出します(このような特徴的な兆候と実際の原因・病理を結び付けて医師の経験を補う知識体系のことを徴候学と言います)。2〜3の病気の候補を決定する行為、すなわち「仮説を立てる」わけですね(**図18**)。

このような仮説立案はいわゆる"見立て"と呼ばれ、医師の経験や過去のパターンからの推測を行います。この仮説を「確定」させるために検査データを取得し裏づけをとります。医師の眼力とは、知覚に相当する見立て＝兆候から仮説を立案してデータで裏付ける一連の思考であり、医師の能力そのものといっても過言ではありません。

では、ソフトウェア開発の世界ではどうでしょうか?

コンピュータに「どんな症状ですか?」と問診するわけにはいきません。品質を観察する手がかりになるデータを入手し、兆候観察します。そこから状況を推察することで、同一状況や過去に経験した症状をパターンと照合してゆくのです。さらにデータから観察できる兆候／推察(仮説立案)と実際の欠陥をセットで記憶しておきます。

兆候を示すデータと仮説およびデータをセットで記憶していく方法は、ソフトウェア品質の専門家だけでなく、多くの開発者の技術力を高め恒常的に品質を確保する有効な手段となります。

測定メトリクスは最初シンプルなものを

世の中にはいわゆるクラシカルメトリクスと呼ばれる古典的・定番的なメトリクスが存在します。読者の皆さんはどれくらいメトリクスをご存じでしょうか?

複雑度、大きさなどさまざまなメトリクスが存在しますが、いずれも測定負荷が大きいことや標本数が集められずあまり有効でないと感じていらっしゃる方も少なくないと思います。

品質検証に用いるメトリクスは——それこそメトリクスというほど格調高くないもので——簡単なものでかまいません。品質検査の出発点として、ヒントやきっかけ、欠陥特定に向けた「足がかり」になればどんなものでもかまいません。

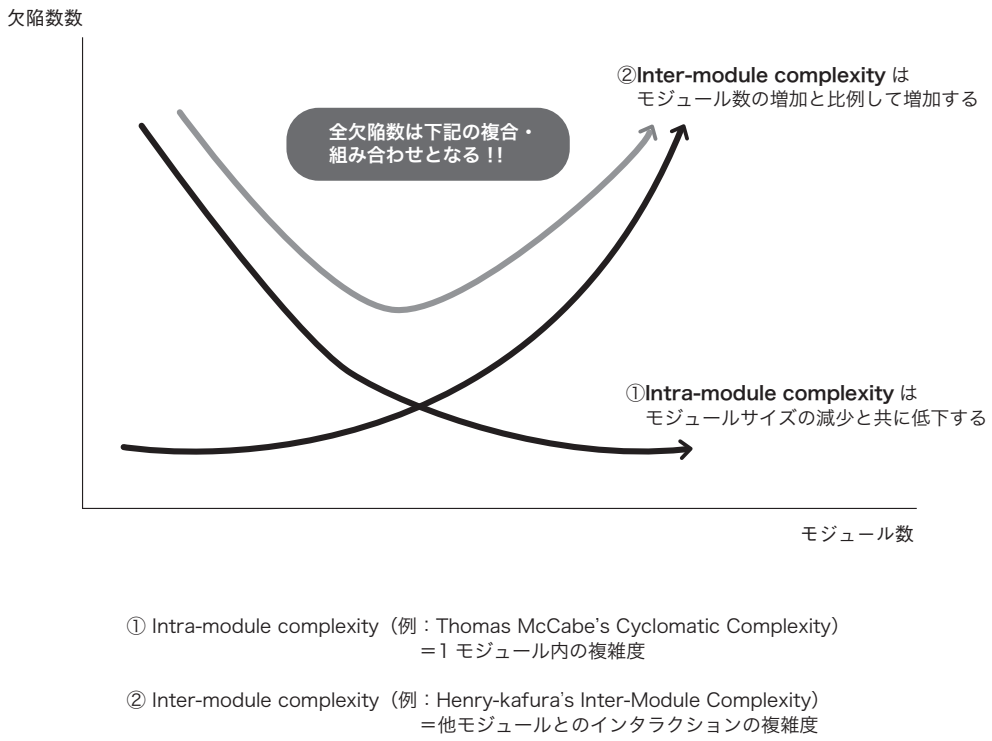
本稿の中で示した事例「ファイルの最終更新日付・時間」のような簡単なメトリクスでも、継続して測定し続けることで十分に役に立ちます。「十分に役に立つ」という感覚が重要です。昔から言われる用語ですが「Good Enough」な品質を目指すことを最優先に考えるべきです。その観点からは、本来の品質活動に注力できるようすばやく測定できるメトリクスであることが理想です。品質活動に完全性を求めるよりも生産性とスピードの概念を持ち込むことが重要なのです。

測定した単一のメトリクス、さらに絶対値で品質のよし悪しを判断しない

循環的複雑度(CC: Cyclomatic Complexity)というメトリクスをご存じでしょうか。Thomas McCabeという人が考案したプログラム内部の複雑度を分岐数や呼び出し数から算出するメトリクスです。このメトリクスは日本でも広く使われており、恒常的に測定される組織も少なくありません。「30を超えると構造に疑問、51を超えるとテストが困難に。そして75を超えると、いかなる変更も誤修正を生む原因となる」という説まであります。誤解を恐れず言えば、このような管理は「メトリクスの老害」とも言えるヒステリックなものであり、欠陥や品質とはほぼ無関係に測定されているのが実態でしょう。

この1つのメトリクスに注力するあまり、モジュール同士の複雑度(IMC:Inter Module

▼図19　2つの複雑度



Complexity：モジュール間相互複雑度。Henry-Kafra複雑度などが有名)は無視してしまう例も少なくありません。

モジュール間相互複雑度と循環的複雑度は**図19**のような相反を示すもので、利用者は両方のバランスを注目すべきなのです。ただし現実には、国内では両者を測定・活用している例はあまりありません。その理由は測定負荷が大きいからではないでしょうか。

実は測定負荷の軽いものの組み合わせで総合的にソフトウェア品質を評価したほうが、単一のメトリクスよりも有用であり、前述の「品質評価の足がかり」や「パターン蓄積」として重要な資産と成り得えます。

技法よりも大切なもの

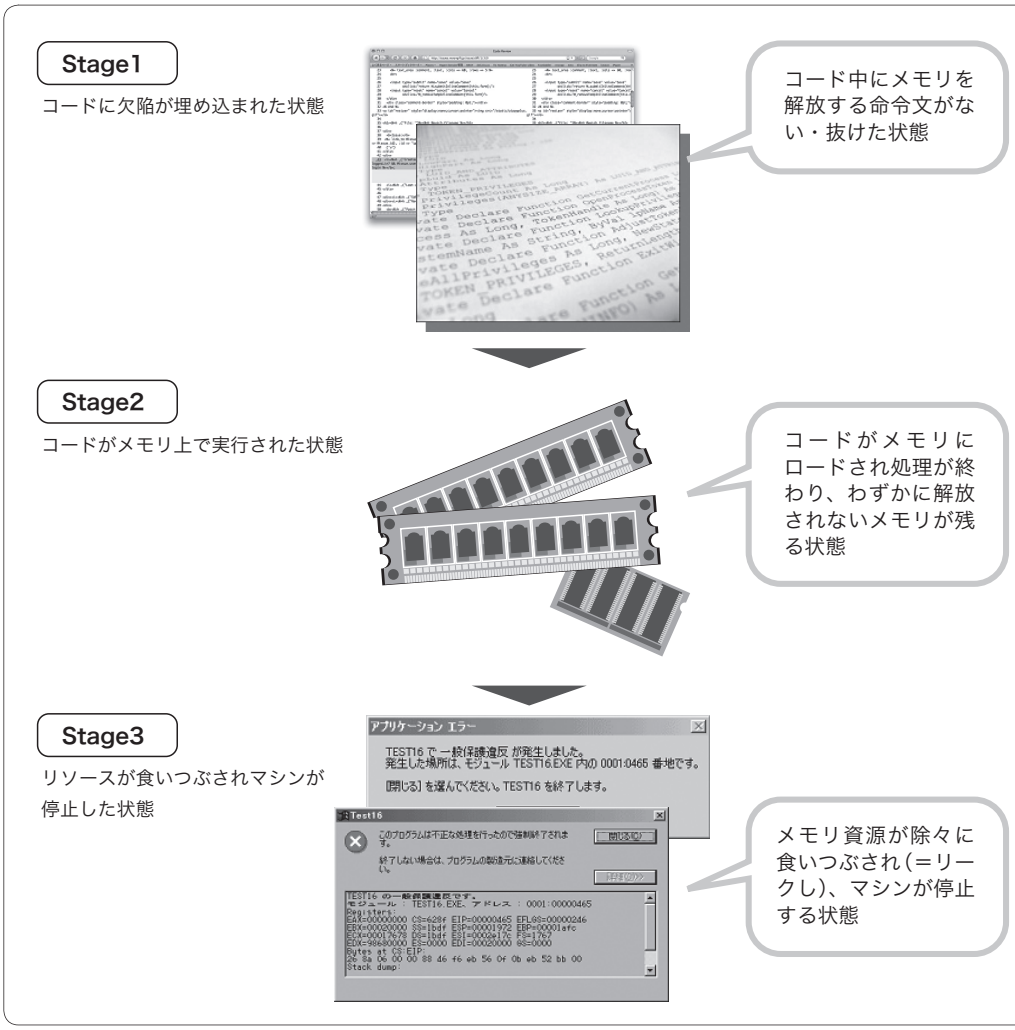
メトリクスと並んで「眼力」として重要な考え方は、「技法よりも対象に目を向けること」です。

かつてFrederick P.Brooksは1980年代には「狼人間を撃つ銀の弾丸はない」と表現して、万能のソフトウェア開発の技法はないとIT業界に警鐘を鳴らしました。

1つ想像してみてください。仮に皆さんの手に「狼人間を撃つ銀の弾丸」という魔法のような武器が渡されたとします。読者の皆さんはこの新しい武器を実際に使う前にどのような準備をしますか。どんな銃器を使うか選ぶ、至近距離で打つか、ワンハンドブロー(片手撃ち)がいいか――などの技術的な面を検討するかもしれません。果ては魔法の呪文でも用意するかもしれません。

しかし、肝心な倒す相手に関する情報を知らなくて良いのでしょうか。相手が狼人間であると識別・区別する方法を知らなければ、何の罪

▼図20　どの状態のことを「欠陥」と呼んでいるのか?



もない一般人に銀の弾丸を撃ち込んでしまうかもしれません。最初に行うべき準備は「狼人間を識別する」ことであり、**対象をよく観察する**ことが必要なのです。

では、開発エンジニアや品質エンジニア、テストエンジニアにとって「対象」とはなんでしょうか？　言うまでもなく「欠陥」です。欠陥に対する知識やノウハウは流通しておらず、たとえば「資源デッドロック(“すくみ”とも言う)」や「メモリリーク」を新入社員にどう教えていますか？

正確な定義や動作原理を正しく解説した書物や論文があるでしょうか？　もしあるというのでしたら、**図20**に示す3つの状態のうち、いずれを「欠陥」として表しているのでしょうか？

もう1つ重要な問題があります。この図中の3つの欠陥状態を、どうメトリクスで測定するのでしょうか？　欠陥は1つ(＝1種類)でしょうか？

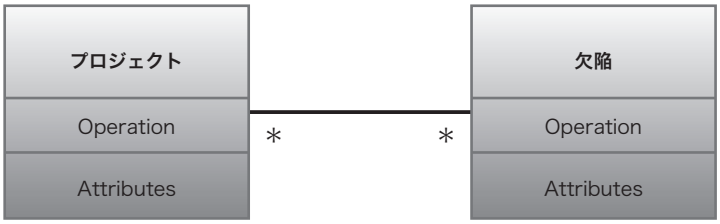
それとも3つと数えますか？　それとも20個所で発生していれば20個と数えるのでしょうか？

上記図中の3つの状態はいずれもテストやレビューで日常の開発プロジェクトで頻繁に目にするものであり、読者にとって馴染みのある欠陥だと思います。しかし、いかに欠陥に関する正確な定義や測定方法が存在せず、あいまいな

▼図21　プロジェクトと、欠陥、プロジェクト欠陥の関係

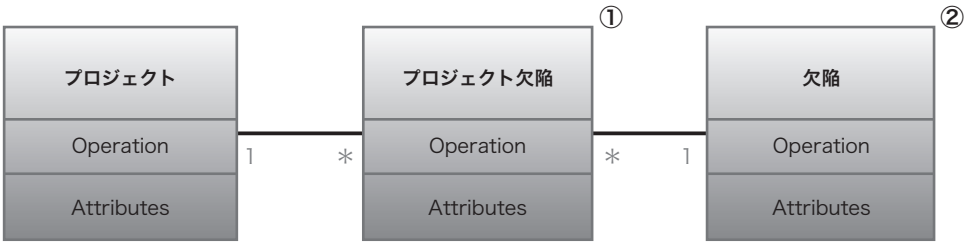
■ プロジェクトと欠陥の関係を整理すると次のような関係

1. プロジェクトには複数の欠陥が発生する
2. 一つの欠陥は複数のプロジェクトで検出される



■ 上記多対多の関係は、1 対 1 関係の「関連エンティティ」をおくと整理できる

1. プロジェクトと欠陥の間に「プロジェクト欠陥」エンティティを設定
2. 各 1 対多の関係でリレーションを整理



・プロジェクト欠陥（①）をどれほど多量に集めても、観察・参照・利用できない

・欲しいのは②の「欠陥マスター」情報。固定化・抽象化・整理されなくては参照できない

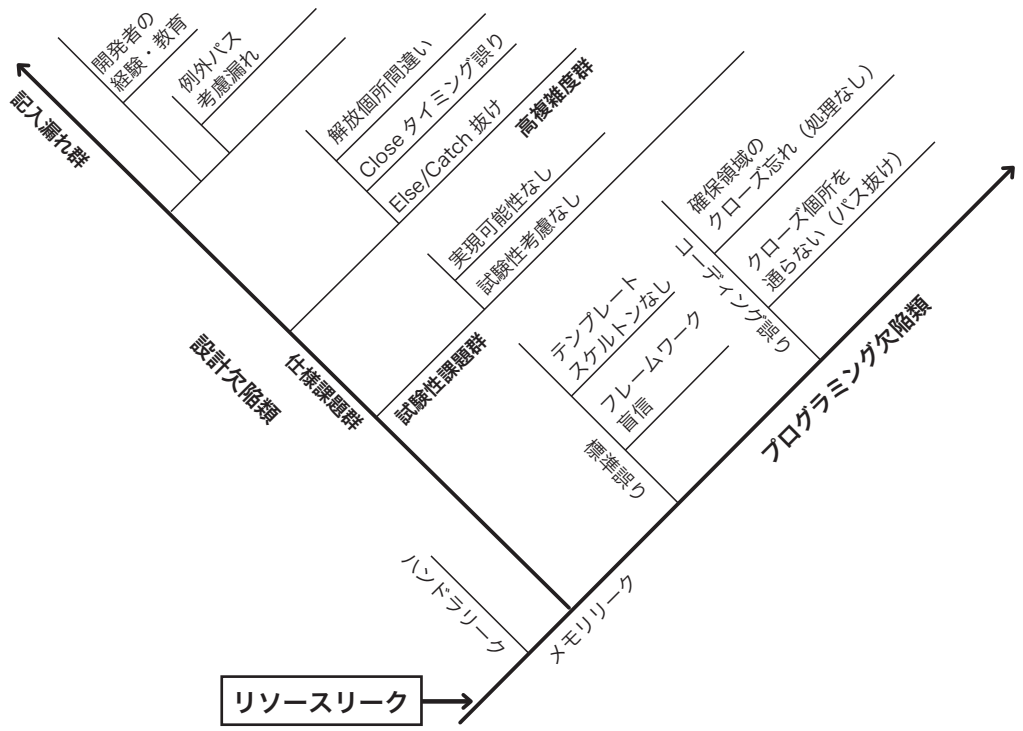
まま運用しているかが如実に現れていると思います。欠陥の研究や知識蓄積が公的なものとして行われず、欠陥を目にして「うん。知っている」と思った方は、ご自身がどこからその欠陥知識を獲得したか思い出してください。多くの方が先輩や上司から「口伝の暗黙知」として教わったものではないでしょうか。

欠陥のマスタデータベースがあれば眼力は向上する

IT業界にはソフトウェア欠陥のマスタデータベースが存在しません。これは本当に不思議なことで、産業が発展する際には必ず公共のデータベースが生み出されてくるのです。たとえば医学であれば病気を集めた病理学データベースが、料理やお酒にはスタンダードレシピが編纂され、今なお更新されています。

ではなぜソフトウェアの欠陥情報を蓄積し、次の世代のエンジニアに資産として継承しようとしないのでしょうか？ 筆者の知る限り「新人に過去のバグ票を大量に読ませる」「BTS(BugTrackingSystem)の過去ログを全部覚えさせる」といった企業もいくつか存在しますが、どうやってバグ票を起票するまでの思考を行ったかを見つけることができず、あまり効果的でない学習といってもいいでしょう。また**図21**に示すとおり、欠陥の「インスタンス」つまりプロジェクト依存の欠陥をいくつ覚えてもほかのプロジェクトでは流用・利用できないという事態に陥ります。

▼図22　メモリリークの原因樹形図



解説：メモリリーク

言語依存性：あり
検出方法：動的テスト
有効対策：仕様レベルでの予防（レビュー）
　　仕様・コードともに時系列不整合検査
注意点・併発症：過剰なクローズ処理（＝確保していない領域の解放）

図中のプロジェクト欠陥、すなわち日常開発者やテスターの方が起票する「障害管理・バグ票」は、先輩からの口伝として書き方を教わり、欠陥のマスタ情報（**図21の②**）を参照せずに記載される事態が多いのではないでしょうか？

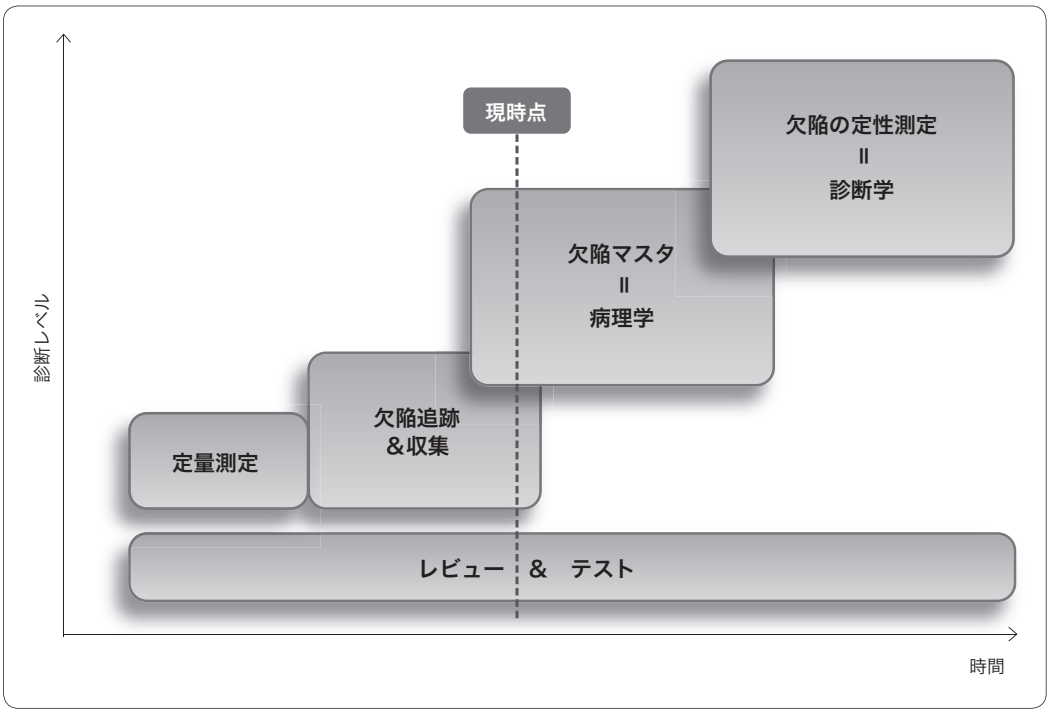
結果として「プロジェクト終了と共に削除してしまう」情報として自然消滅してしまうのです。

仮に十分に抽象化され、個々のプロジェクト状況やテクノロジへの依存性を除去した形で欠陥マスタが作成できたら、同じ欠陥の種類は再発が予防され、検出技法選択で悩むことも、副作用を意識しないで誤った除去方法を採択することもなくなります。

特定の欠陥に関する定義・特性や検出方法、先述の兆候や仮説立案方法、除去方法と副作用、予防方法などを定義したマスタを参照することで、属人性を排除できるかもしれません。

たとえばメモリリークであれば、**図22**のような「生物進化樹」の形式で混入原因をまとめることができ、関係する欠陥の幹をたどることで

▼図23　医学の進化とIT産業の技術進化



検出しやすくしたりすることもできます。

欠陥マスタが普及した世界では、開発者や品質担当者の仕事はどのように変わるのでしょうか？　医学が進化した経路に準えて説明すると、図23のような進化を遂げると筆者は考えます。

まず、現在はレビューやテストは一部の専門家のものであり、開発側と対立関係を構築する場合も少なくありません。開発側にとってのテストやレビューは、普段の生活で健康についてやかましく言われるのと同様、「しかたなく」やらされるものでした、これが現在までに一部で行われてきた「定量測定」や「欠陥追跡や収集」が欠陥マスタにより高い次元で実践されるようになります。いわゆる「欠陥知識＝悪さの知識流通」が始まるのです。そして、欠陥の定性測定、すなわちわずかな兆候が入手できれば対処方法や療法、さらに再発予防方法などが一意に決定される診断学の発展とともにハイレベルで高速な品質活動が行われていくのではないでしょうか。まさに先進的な品質エンジニアリングの時代が到来します。

つまり、未来の技術者たちには、技術者の「眼力」を強化するためのさまざまなしくみが提供され、あらゆる欠陥においてその内容が共有・移転・利用可能な状況となっていくのです。これは細かい技法や方法論を追い続ける教育を受けてきた時代から、次の時代「対象＝欠陥を学習し、悪さの知識流通」と欠陥を見抜く眼力を重視する教育がそう遠くない時代に来ることを意味します。

まとめ「銀の弾丸は本当に必要ですか?」

それでも読者の皆さんは個々の部分最適化技法やわずかな前進をもたらす銀の弾丸を追い求めますか？　差別化のための技法（＝銀の弾丸）を追い求めることは労力と時間を要するだけでなく、時として現場・現実を見る目を曇らせてしまうのではないかと筆者は考えます。そうであれば今後の開発者の差別化を目的とした新た

な考え方が必要となってくるでしょう。もはや鍛えるべきは腕だけでなく眼力、つまりものの真価を見極め、未来を予測・予防するためにメトリクスデータによる兆候捕捉、欠陥知識、検査技法などを駆使して「見る」技術であり、眼力の獲得なくしては生き残れない時期にすぐそこまで来ています。自らの成果物をこれらの新しい切り口で自律的に品質確保しなくては、誰もが「作るだけの開発者はもう要らない」といわれてしまう時代です。

言うまでもなくさまざまな開発技法や品質保証テストやレビューに代表される個々の技法は長い歴史上で堆積した課題の産物であり、知っておくに越したことはありません。けれど私たち技術者が最優先に身に付けるべきは、技法の知識ではなく欠陥知識とそれを見通し見極める眼力です。

本稿では、品質エンジニア、テストエンジニアが普段から使っているさまざまな品質評価技術の概観を通じ、今後開発系技術者が備えるべきシステムを見通す第三の力＝「眼力」について解説しました。本稿で示した「品質にスピードという概念を持ち込む」という思想と具体的な眼力が、さらに次の新しい潮流の礎となることを、筆者は願ってやみません。SD

●参考文献
[1] [2] ジニ係数: http://ja.wikipedia.org/wiki/ジニ係数
[3] 『人月の神話』フレデリック・P・ブルックス Jr.(著)、滝沢徹、牧野祐子、富澤 昇(訳)、ピアソン桐原、2010年12月、ISBN：978-4864010054

マニュアルどおりでホントに使える?

小規模プロジェクト現場から学ぶJenkins活用

第2回 管理は仕方ないけど、運用は楽しましょうか

第1回では筆者の会社でJenkinsをどのように活用しているかという話をしましたが、いきなりJOBの説明から入ってしまい筆者のJenkins環境を紹介していませんでした。今回は筆者の環境とどのように管理しているかについて紹介していきます。

嶋崎 聡(しまざき さとし)(株)XVI Twitter @sato_c

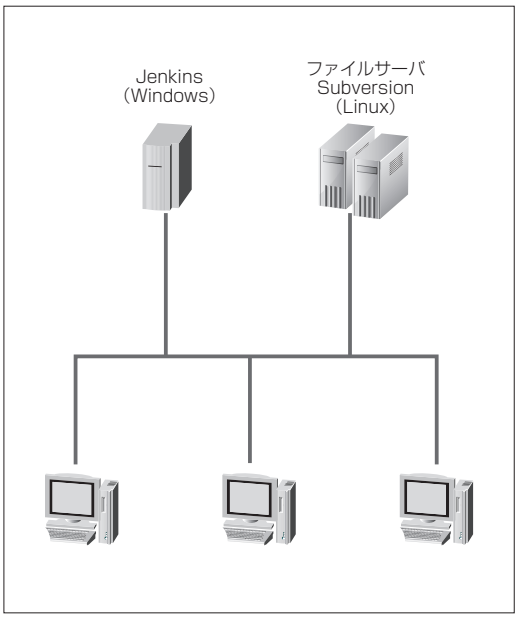
Jenkinsの運用環境

筆者の環境ではJenkinsをWindowsで動かしています(**図1**)。ファイルサーバやSubversionをLinux環境で動かしているので、当初はJenkinsもLinux環境で動かしたいと考えていました。しかし、画像の変換やファイルの取りまとめなど、やらせたいと思っている作業に密接にかかわっているツールがことごとくWindowsでしか動かないため、Windows上にJenkinsが必要でした。

Jenkinsでは、1台をマスターにして、ネットワーク経由で接続する複数のスレーブPCでJOBを実行できます。この形でマスターをLinux、スレーブをWindowsにして、そちらでツールを動かすことも考えました。しかし、この構成を構築する手間ほどのメリットを感じなかったので、今回はWindowsのマスター1台のみで運用することにしました。

マスター1台構成にするにあたって考えるべきポイントがあります。本稿の第1のテーマとして後ほど詳しく説明します。

●図1 プロジェクトで利用しているネットワーク構成



浮かび上がる運用の問題点

実際に動かし始めると次のような問題点が出てきました。

- JOBの記述方法
- JOBの実行結果を他のスタッフにどうやって伝えるか

などです。JOBの記述方法では、JOBの内容をブラウザから管理することができて手軽な反面、記述した内容の編集となるとブラウザからでは見通しが悪いという面が出てきました。これを改善するにはどうするかを考えましたので、本稿の第2のテーマとして紹介します。

実行結果の確認は「JenkinsのJOB一覧画面で確

認してください」で済ませたいところなのですが、「JOB一覧画面を見るのが面倒」とかJenkinsのJOB一覧のURLを伝えても「どこを見たらいいかわからない」という意見がありました。自分以外のスタッフがどうしたらJOBの結果を見てくれるかについて、どのように対策したかを第3のテーマとして紹介します。

①マスター1台のみの運用ポイント ――ワークスペースの管理

まずは、マスターサーバ1台の環境構築作業で出てきた問題点と解決策を紹介します。

Jenkinsでは、JOBが処理を行うために**ワークスペース**と呼ばれるディスク領域を必要とします。ワークスペースとはJOBの作業用領域であり、JOBが実行されるとリポジトリから最新の内容を取り込んで処理を行います。

1つのワークスペースに必要な容量は、取得する内容によりますので一概には言えませんが、プログラムをビルドするだけならば最新のソースコードを取り込めて、オブジェクトファイルなどの中間ファイルが保存できる量、多くても1GB程度で充分ではないでしょうか。

しかし、筆者の環境では映像データ（ターゲットのハードウェア用映像とAfterEffectsなどのオーサリングツールで利用する映像の2種類）の処理が必要で、それだけでも結構な容量になりました。データがどれほどになったとしてもJenkins側にたくさんのHDDを用意できれば問題ないのですが、現実問題としてはなかなかそういうわけにもいかず、今回は容量節約のため大容量データを複数のJOBそれぞれに持たせない方針としました。

JOBごとにワークスペースを用意するか

JOBのデフォルト設定は個々にワークスペースを持つようになっています。しかし、どうしても1つのワークスペースでやりたいと考えたので、ワークスペースを分けた場合と分けなかった場合のメリットとデメリットをディスク容量的な観点から考えてみました（**図2a**、**図2b**）。

●ワークスペースを分けたとき

各JOBが個別のワークスペースを持つ場合、各JOBが実行された時点でのリポジトリの内容を取得できます。このときに必要なHDD容量はソースコードやデータを取得する分と作業中にできるファイルが充分保存できる量になります。

他のJOBへの影響は、JOBが作成したオブジェクトファイルやデータがコミットされない限りはありません。また、各JOBは独立したHDD領域を使って動作しますので、複数のJOBが同時に動いたとしても問題は出ません。

他のJOBの成果物を利用する場合は、対象となるJOBの成果物を持ってくる必要があります。

●ワークスペースを分けなかったとき

ワークスペースを複数のJOBで共通領域として持つ場合、必要なHDD容量は1つ分で済みます。HDD容量はかなり節約できるでしょう。

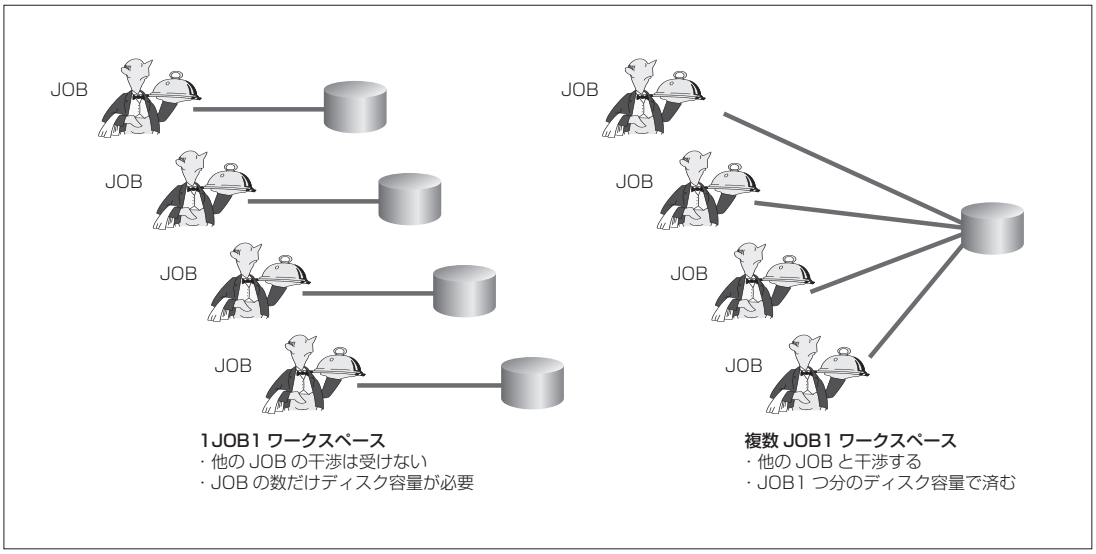
共通ワークスペースでは、JOBを同時に動かそうとすると同じHDD領域に対してリポジトリ更新を

Column 1 スレーブを用意する場合

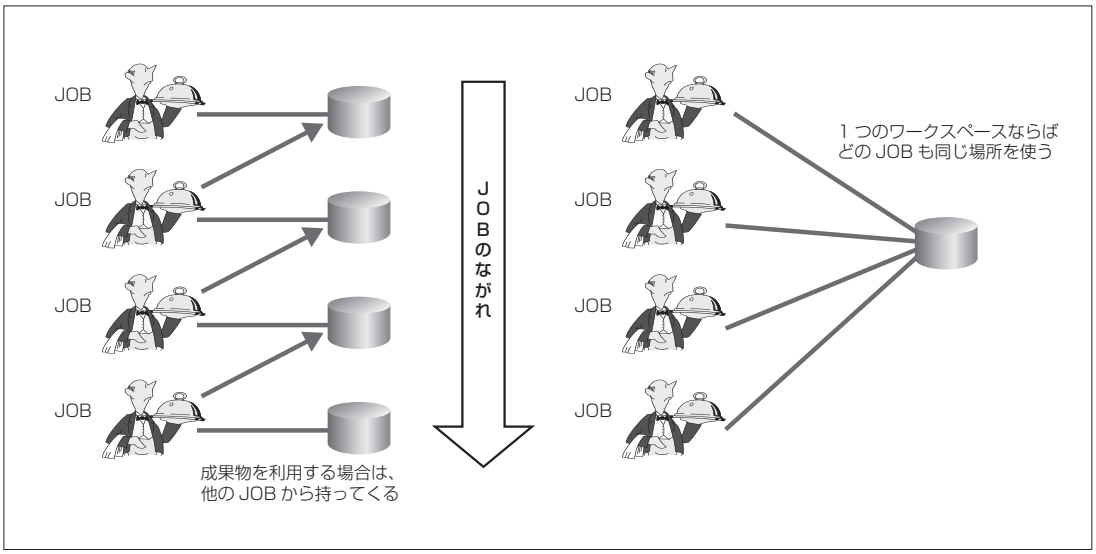
スレーブはマスター1台だけで作業が間に合う間は、それほど必要ないでしょう。マスターだけではJOBを実行する時間がかかりすぎるとか、並列ビルド条件を使って複数のバージョンのビルドが必要になるといった用途が明確になってから導入を考えるといいでしょう。スレーブでの作業を明確にしないでただ増設すると、スレーブがあるのにいつまでもマスターだけでJOBを構成するといった、もったいないことになってしまいます。

すぐにでもスレーブを使う構成にしたほうが良い例としては、英語版、日本語版といった言語違いのプログラムをビルドするとか、iOS版、Android版それぞれのアプリをビルドするといったものがあります。こうした環境が増えてきたのか、プラグインにもAndroid環境やiOS環境を対象にしたものがあります。

● 図2a　JOBごとにワークスペースを持つ場合（左）と持たない場合（右）の構成



● 図2b　JOBごとにワークスペースを持つ場合（左）と持たない場合（右）のデータ共有



行おうとします。このとき同時に更新を行ってしまうと更新作業の衝突が起きることになります。両方エラー終了してくれればいいのですが、片方はエラー終了して、もう片方はいつまで経っても終わらないという状態になることが大半でした。JOBが正常に終わっていないと、次に動かしたときにワークスペースのファイルアクセスができないといったエラーが出ますのでうまく動かなくなります。こうしたエラーが起きたときの手間はかかります。

JOBの結果を出力、保存する場所は同じですから、他のJOBの成果物はコピーしなくてもそのまま利用できます。

ディスク容量を節約する方向で

容量を節約しようと考えた場合のメリット／デメリットを検討した結果、今回はやはりディスク容量を優先することにしました。“同時に動かす必要のあるJOBではワークスペースが被らないこと”を基

本ルールとして、ワークスペースを共通にします。

JOB間でワークスペースを共通化するためにワークスペースを固定するには、カスタムワークスペースを利用します。通常のワークスペースは、Jenkinsの設定で指定するJenkinsのホームディレクトリ以下に作られます(**図3**)。カスタムワークスペースは影響する範囲がそのJOBに限られるため、JOBの設定画面から指定します。普段は表示されていませんが、「プロジェクトの高度なオプション」の項目にある「高度な設定」ボタンを押すと、**図4**のように設定項目が表示されます。

「カスタムワークスペースを使用」にチェックを入れてディレクトリを指定すれば、そのあとのJOB実行から新しい場所にワークスペースが作られて、そこでJOBが実行されます。この設定を必要なJOBの数だけ行います。

こうしてHDD容量と引き替えにJOBを複数同時に動かす際の制限はできてしまいましたが、割り切って共通ワークスペースを使うようにしました。実際に複数人で行う作業はムービーのコンバートであり、それ以外のJOBは単一作業だったり、定期的に自動で実行させても大丈夫なので、影響はでていません。

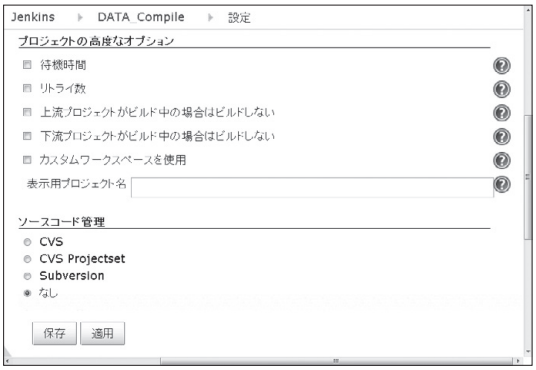
②JOB処理の記述

容量の問題が一段落し、運用を始めたところで次に出てきた問題はJOB処理の記載方法でした。JOBの処理は設定画面のビルド処理の部分に書けます。しかし、直接書けるからといって、そのまま書いていくのはあまり便利ではありませんでした。数行の処理であればいいのですが、それ以上のサイズになるとブラウザのフォームで編集していくのは結構大変な作業になります(**図5**)。

ちょっとテストという間はいいのですが、運用する段階になったら、できるだけバッチファイルやシェルスクリプトのバージョン管理をしたほうがいいでしょう。JOBはSubversionなどのバージョン管理システムを使っている前提で作ると思います。そこでJOB処理のファイルを保存するディレクトリ

●図3 JOBディレクトリはシステムの設定から指定する

●図4 カスタムワークスペースの設定

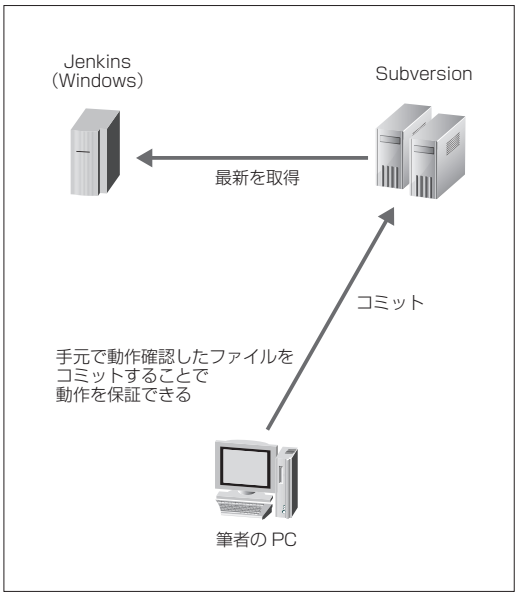


●図5 処理の編集フォーム

を作って、必要なファイルを入れて一元管理しましょう。こうすれば、Jenkinsのワークスペース側にも常に最新の処理ファイルを置けます。

Jenkinsで動かす前に手元で動くことを確認すれば動作保証もできますし、バージョン管理しておけば、変更が間違っていたときも動いていたバージョンに戻せばすぐに復旧できます(図6)。1人ですべてを管理している間はあまり意味を感じないかもしれません。しかし、忙しくなってきたときや複数人で管理を始めたときに重要になってきます。とくに複数人で管理を始めると、お互いに注意し合っていても何か失敗することがあります。JOBを同時に編集していて、保存するタイミングがずれてしまったときに編集した範囲が大規模だと時間が無駄になります。そんなときもバージョン管理していれば、先にコミットしたファイルと比較できたり、自分以外の人が入れた修正で動かなくなっても慌てなくて済みます。

●図6　手元で動作確認したファイルをJenkinsでも動かす



管理していく中で失敗することは当たり前にあります。問題は失敗することよりも、失敗したときにどれくらい素早く落ち着いて復旧対応できるかなので、手元にファイルを持つようにしておくといいでしょう。

JOBの記述にはバッチファイル?

JOBの記述にはバッチファイルとシェルスクリプトが使えますが、どちらを使うのがいいかという基準はありません。コマンドの呼び出しを繰り返すだけにするならば、どちらも変わりません。使い慣

Column 2
Windows環境での話
――コマンドラインの先頭から数文字がなくなる謎の現象

筆者はおもにCygwin環境を使っています。しかし、たまに「Windowsバッチコマンドの実行」(以下、バッチ実行)を使う必要がでてきます。これは、Windows上で使っている以上、避けられないと思ってあきらめています。なぜ、そこまでバッチ実行がイヤかというと漢字メッセージが入っているとJOBが失敗するからです。ほかにもバッチ実行の改行コードがWindows標準のCRLFではなくLFになっていることも関係しているようで、この2つが重なるとまず正しく実行されません。

実行中のログにどういうフェーズかのメッセージを表示したくなっても、わかりやすく日本語でメッセージを入れられないということになります。たとえば「コンバータを呼び出します」とか「ファイルをコミットします」といったメッセージを作業の区切りごとにechoで日本語メッセージを出力したいとします。処理にこういったことを追加してもとくに何事もなく保存できます。しかし実行すると、ログには「'(さっき入れたメッセージ)' は、内部コマンドまたは外部コマンド、操作可能なプログラムまたはバッチ ファイルとして認識されていません。」となってしまい、JOBが失敗します。

こうしたことで無駄に時間を使わないようにするためにも、バッチ実行に直接書かずに処理用バッチファイルを用意しています。

れているほうを使うのがいいでしょう。

筆者はシェルスクリプトを多めに使っています。Windowsのコマンドプロンプトでも環境変数を経由して文字処理をしたり、ifなどで処理の流れを制御できるのですが、Cygwin環境ではUNIXコマンドが使えたり、シェルスクリプトでif文などが使いやすいのでバッチファイルを使う機会が少なめになっています。プラグインを使うとJOBの記述にPythonやRubyが使えるようにもなっていますので、こうした環境を使うのもいいでしょう。

③JOBの処理結果を見てもらえない

次にJOB結果について、どうしたら自分以外の人にも見てもらえるかを考えました。

JenkinsのJOBはダッシュボードから一覧が見られます。名前と直近のJOB結果などが、どーっとリストアップされています。実行中のJOBは左側の「ビルド実行状態」にまとめられていて、JOB名の下のグラフをクリックすれば実行JOBのコンソール出力が確認できます。

エラーが出たらコンソールからエラー内容を確認してほしいところですが、Jenkinsをよく知らない人にはコンソール出力で何をどうすればいいのかわからない。なので、結局使ってもらえなかったり、ことあるごとにこちらに問い合わせが来たり……というのが実際の運用でわかってくるわけです。

JOBをタブ分けして少しでも見通しを良く

原因を考えていくうえで、まずJenkinsのJOB一覧を見直してみました。JOB一覧そのままでは「all（すべて）」タブがあり、すべてのJOBがまとめてあります。

これ1つでは、JOBが増えてくると表示が縦に長くなっていき、目的のJOBを探すまでに結構時間がかかります。まずは、JOBが多くなってきたら、all以外にタブを作って共通するJOBをまとめてしまう手があります。1つで全部を表示するより見やすくなります。データに関するJOBをまとめたタブ、プログラムに関するJOBをまとめたタブといった具合に複数のタブを作って、相手に見てほしいタブだけを知らせれば、探しているJOBをすぐに見つけてもらえるかもしれません……多分。全体のJOBの数が増えるにつれて、タブを作って管理するのも実際は手間になっていきますが（**図7**）……。

JOBの起動には必要ですが…

筆者自身はJOB一覧からJOBを起動していますが、同じように起動してもらうことが難しいのであれば、JenkinsのJOBを起動する方法をほかに考えなくてはいけません。

簡単にするということでまず思い浮かんだのは、Jenkinsは外部からJOBが起動できるので、起動用リンクを1ページにまとめることです。……しかし、冷静に考えるとJOB一覧と変わっていません。

もうブラウザを使ってもらおうと考えるのはやめて、普段全員が起動しているSkype経由にできないか考えました。JOBの外部起動方法はJenkinsのマニュアルに載っていましたから、これを応用して作ることにしました。SkypeにJenkinsコントロール用のチャットグループを作って、そこから送信されたコマンドを受信して処理します。JOBの開始／終了やエラー情報もここに流せば、確認する場所は1つで済みます。

●図7　タブに分けて表示するJOBをまとめられる

外部からJOBを起動する

JenkinsにはRemote access API[注1]が用意されています。前述したように、これを使えば外部からJOBを起動できます。最終的にはこのしくみを利用して、当社デザイナが使い慣れているSkypebot経由でJOBの起動と実行結果の告知ができるようにしたいと考えました。

想定している起動方法はAPIの説明ページにそのまま書いてありました。「JENKINS_URL/job/JOBNAME/build」という形式で、普段見ているJOB画面のURLの最後に/buildを付ければそのJOBが起動することもわかりました。これならばどこからでも、ブラウザでJOB画面のURLの後ろに/buildを付ければ、簡単に試せます。ただし、Jenkinsのユーザ認証設定を利用していなかった場合です。

ユーザ認証の設定

実際の運用では何が起こるかわかりませんし、人数が多かったり、外部からのアクセスも許可する必要があるならば、なるべく[Jenkinsの管理]－[グローバルセキュリティ設定]－[セキュリティを有効化]にチェックを入れて「アクセス制御」でユーザ認証設定をしましょう(**図8**)。

この設定で、ログインしていない人にはJenkinsやJOBの設定画面のリンクは表示されなくなるので、間違って消されるといった事故も防げます。外部に公開する場合は、これ以外にIPアドレスを制限するなどの対策もしないとダメかもしれませんので、そのときはネットワーク管理者の方などとも相談したほうがいいでしょう。

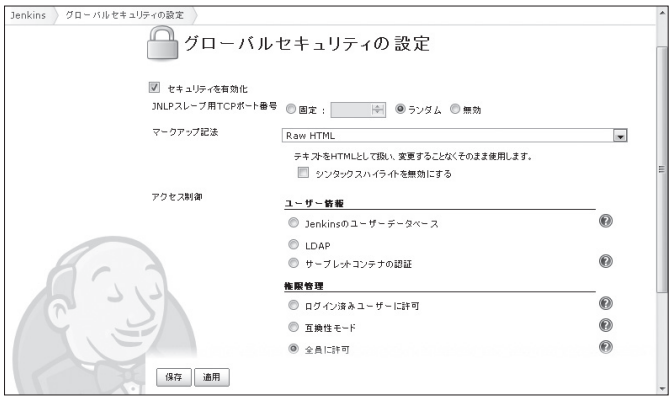
ユーザ認証設定をした環境では、JOBの外部起動のURLにアクセストークンを追加する必要があります(**図9**)。

トークンに設定する文字列にはとくに制限はありません。外部からの呼び出し用スクリプトに埋め込んであれば丸見えになってしまうのですが、一応このトークンを知っている人だけがJOBを起動できるようにするために使うものですから、誰にでもわかるような「aaa」といった単純すぎる文字列はやめたほうがいいと思います。

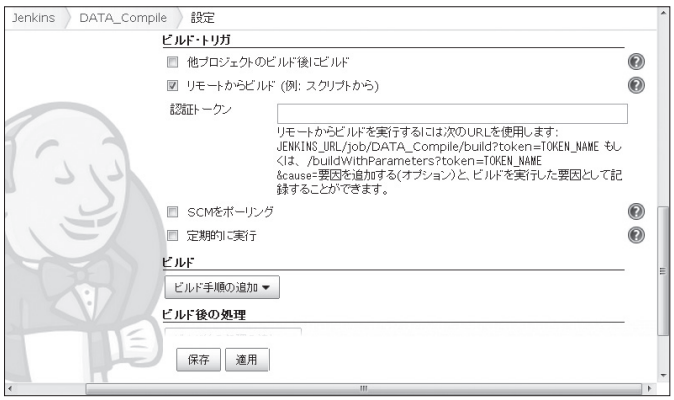
ブラウザ以外の方法で起動する

ブラウザからのJOB起動が確認できたので、実験として、JOBをCygwinのwgetコマンドを使って起動してみました。認証設定もした環境なので、JOBにアクセストークンを設定して実行しました。

●**図8　グローバルセキュリティ設定**



●**図9　アクセストークンをJOBに設定する**



注1) https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API

```
$ wget http://localhost:8080/job/DATA_Compile/build?token=compile  -o NUL
```

手元の環境でJOBを起動しているのでlocalhostになっていますが、実際の環境でJOBを起動する場合はURLとポート番号を変更する必要があります。

wgetはネットワーク上にあるファイルを取得するためのUNIXコマンドです。本来はURLとプロトコルを指定してファイルをダウンロードするものですが、今回はURLを指定してアクセスしてもらうだけで、ダウンロードしたファイルは保存しない設定です。これでJenkinsの画面を見てJOBが起動していれば成功です。テストが終わって実際に利用するときは、NULの後ろに--quietオプションを付ければ、wgetのメッセージを表示しなくなります。

Skypebotに応用する

こうしてJOBが起動できたので、第1回で紹介したSkypebotから使えるようにしました。Skypebotは、Skypeに常駐して受け取ったコマンドを実行するためのものです。すでにコマンドを受け取る部分と処理を行う部分はできていますから、処理にwgetを使ってJenkinsにアクセスする部分を追加します。最終的には、Skypeから特定のコマンド「@cnv」という@で始まる文字列があったら処理をするように作りました。動作イメージは**図10**のようになります。

Rubyスクリプトについて

Skypebot自体はRubyで作っています。RubyからSkypeの内容を読み書きするために、Ruby4Skype[注2]というライブラリを使っています。このライブラリはGemでインストールできます。

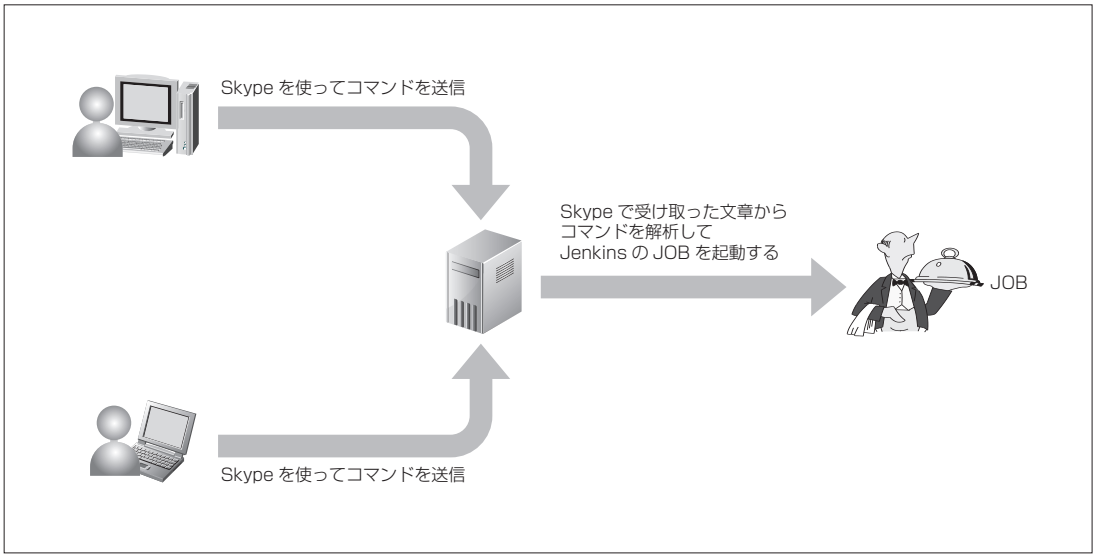
```
$ gem install Ruby4Skype
```

このライブラリはあまり新しいとは言えないものですが、後継のライブラリが見つからないためそのまま使っています。

swin.soというライブラリ[注3]が必要になるのですが、このライブラリを1.8系向けにビルドされたものを使っていますので、Jenkins上のRubyもそれに合わせて1.8系にしています。

実際のスクリプトを見てください（**リスト1**）。ま

●**図10　JenkinsとSkypeを連動させて動作する**



注2）https://rubygems.org/gems/Ruby4Skype
注3）http://www.osk.3web.ne.jp/~nyasu/vruby/vrproject-e.html

ず、Skypeからコマンドを受け取って、wgetでJOBを起動するスクリプトです。

7～9行目がライブラリの初期化、11～18行目でトピック名がJenkinsとなっているチャットの取得を行っています。このようにすることで、他のチャットに打ち込まれた文字列には反応しません。

20行目からは、コマンドとそれに対応したJOBの場所を配列に設定しています。25行目からがskypeからコマンドを受信して、JenkinsJOBを呼び出す本体となっています。

●スクリプトを実行

あとは、このスクリプトをJenkinsが動いているPCのスタートアップで起動して常駐させておけば、Skype経由でJOBを起動できるようになります。スクリプトが起動していれば、JOBを動かすのはチャットにコマンドを打ち込めばよくなります。これでJOB一覧を見てもらうようにお願いして回らなくて済みます。

また、これだけではJOBがどうなったのかまではわかりませんので、**リスト1**のスクリプトを少し改造してTCPポート経由でSkypeにメッセージを送れるようにしています。具体的には、TCPポート

● リスト1　Skypeからコマンドを受け取って、wgetでJOBを起動するスクリプト

```
001 : #!/usr/bin/ruby -Ku
002 : require 'rubygems'
003 : require 'Skype'
004 : require 'kconv'
005 : require "socket"
006 :
007 : Skype.init 'botbot'
008 : Skype.start_messageloop
009 : Skype.attach_wait
010 :
011 : chats = Skype.searchRecentChats
012 : chats.each do |n|
013 :   topic   = n.get_topic
014 :   members = n.get_members
015 :   if topic =~ /Jenkins/
016 :     $tcp = TCPServer.open(6001)
017 :   end
018 : end
019 :
020 : JOBS={
021 :   'cnv'   => 'DATA_Compile',
022 :   'color' => 'DATA_ReduceColor',
023 : }
024 :
025 : Skype::ChatMessage.set_notify do |chatmessage, property, value|
026 :   if (value == 'RECEIVED' || value == 'SENT') && property == :status
027 :     if /^ @/ =~ bd
028 :       bd = bd.slice!(1,bd.size)
029 :       bd.downcase!
030 :       if JOBS.key?(bd)
031 :         chatmessage.get_chat.send_message "#{bd} を開始 "
032 :         kick_job="#{WGET} #{JENKINS_HOST}/job/#{jobname}/build?"
033 :         system(kick_job)
034 :       end
035 :     end
036 :   end
037 : end
```

を6001から6601に変更して、20行目以降をリスト2のように変更します。

このスクリプトも常駐させておけば、TCPポート6601に書き込んだ内容がSkype側に表示されます。

これでデザイナさんたちに使ってもらうしかけは完成です。Skype経由でJenkinsに@cnvといったJOB起動コマンドを送ればJOBが開始します。JOBは処理の始まりにJOB名/ビルド番号/コンソール確認用URLを、処理の終わりにJOB名をチャットに送ります。これで開始、終了もわかりますし、コンソール確認用URLをクリックすれば、コンソール表示から実行中のログを確認してもらえます。

エラーが出たらチャットにエラー内容を送るようにコンバート用スクリプトを改造しました。エラーメッセージと原因をポート6601を開いて書き出す最低限の改造ですが、これだけでJOBを実行した人だけでなく、同じグループでチャットを見ている人がエラーの原因を探してくれました。1人で悩むことが少なくなりました。……よかった。

こうして、Jenkinsの画面を操作することなく、JOB状況の監視ができるようになりました。ちょっとした工夫ではあるのですが、JOB画面を見るのが面倒という声に対応しました。

まとめ

今回はJenkinsのワークスペース構築、JOBに関する話とSkypeとの連動環境について説明しました。

ワークスペースを共有するのは、あまり積極的にお勧めできることではないと思っていますが、どうしてもHDD容量を節約する必要がある場合もでてきます。今回は、大きなデータを何ヵ所にも持つのがイヤだというのが大きいので共通にしました。それ以外でも運用している最中にHDD容量を増やすことが難しい環境や、本格的に導入できる段階ではない環境などでは有効だと思います。あくまでもこうした方法もあるという一例なのでこれが解決策のすべてではありません。

JOBの処理はシンプルな構成にしたほうが作りやすさ、管理の両面でいいのは確かです。でも、

●リスト2 リスト1の改造コード

```
020 : while true
021 :   Thread.start($tcp.accept) do |s|
022 :     while s.gets
023 :       $chat.send_message($_)
024 :     end
025 :     s.close
026 :   end
027 : end
```

使っていくうちにつぎはぎの内容になってしまうことがあります。そんなときにブラウザのフォームにスクリプトを書いていくよりも、使い慣れたエディタで書いたほうが見やすいのではないでしょうか。1行消し間違えて気づかず保存、その後気づいても後戻りできず、元に戻すまで半日かかった……なんてことが起きるのはあまりいいことではないと思いますので対策しておくことをお勧めします(もちろん、そういったことを過去やったことがあります)。

JOBの結果の確認は作業に影響が出てくることも多いので、作業する人同士で確認してエラーが出ていたら原因を探してほしいところです。しかし残念ながら、そういうことをやってくれる人とくれない人はどちらもいるわけです。処理の過程は軽視されているのか、データができてこないのはプログラムが悪い！……ってな話もあったりして苦労するところです。

JenkinsにはJOBが失敗したときにメールで通知する設定が最初から用意されていますが、メールは数が多くなってくると読まなくなる人がいますし、見てくれてもただエラーと書いてあるだけだと訳がわからないので、結局放置されがちです。こうしたことを回避するためにチャットで確認できるようにしたら、エラーを出した当人以外にもエラー対策を手伝ってもらえるようになりました。このあたりは期待どおりとも言える効果がでました。

これ以外の対策ではプラグインも使いましたので、そのあたりについては次回いくつか例を紹介していこうと思います。SD

分散データベース「未来工房」

第2回 高可用性とは何か？
――Riakの機能を知る・試す

Writer 上西 康太（うえにし こうた） Bashoジャパン株式会社 kota@basho.com

先月号では、Riakの変わった使い方について述べた。もともとの設計思想やAPI、運用などソフトウェアとしての特徴も必要最低限に絞ったため、最終的にRiakが何なのかわからず、狐につままれたような読者も多いことと思う。もともとRiakは、Bashoのエンジニアが“自分たちのサービスで使うために”作ったものだ。その基本的なコンセプトは、

・拡張性……サーバを追加すればそれだけキャパシティや性能が向上する
・可用性……過負荷状況下でも安定した動作、レスポンス
・耐障害性…多重故障でデータを失わない、サービスが停止しない

であり、安定したシステムの運用を第一に目指していることがわかる。

本稿では、Riakの原点とも言える高可用性に注目し、どのようなことを目指して設計されているかを解説したい。

注）本稿は、筆者の意向により常体で表記している。

Riakの死活監視のしくみ

まず、基本的なRiakのしくみから説明しよう。分散データベースは、基本的に「どのサーバにデータが入っているか、または入れるべきか知る」ステップと「見つけたサーバに実際にデータを保存する」2つのステップから構成される。まず、前者の「どのサーバに入っているか、入れるべきか」の方法を簡単に説明しよう。

Riakはバケット名とキーでデータを保存すべきサーバを決める。リングサイズは定数だ。リング(ring)は、パーティション番号とサーバのアドレスの対応の一覧表だ。パーティション番号とは、2^{160}の整数空間をリングサイズで定数個で分割したそれぞれの空間に順番につけられた番号を指す（※注：この定数をリングサイズという）。それぞれ分割された空間のことをパーティションという。実際にデータを格納する実体についてはvnodeという言葉が使われる。Riakに格納されるデータは、基本的にはこのvnode単位で管理される。実際に保存すべきサーバは、バケット名bとキーkから160ビットの整数を求めるハッシュ関数 Hash(b, k)、リングサイズをN_rとした場合に、

$$\frac{2^{160}n}{N_R} < \mathrm{Hash}(b, k) < \frac{2^{160}(n+1)}{N_R}$$

b…… バケット
k…… キー
N_R … リングサイズ
n ……パーティションID
Hash(b, k) …ハッシュ関数

となるnをまず計算する。Riakでは、データを複製すべきvnodeをn、n+1、n+2とする。あとは、このパーティション番号をもとにringを参照しデータを保存すべきサーバのアドレスを取得する。これはringから、時計回りに3個のパーティションを選んでくることに相当する。

実際にリクエストを送信する際には、このサーバが生きているかどうかも考慮される。Riakはクラスタを構成するサーバ同士で常に死活監視を行なっており、もしもリング中のサーバが一時的にアクセスできない状態になっている場

合は、時計回りに順にアクセスすべきサーバをリング上から選びだす(**図1**)。

Erlang/OTPの死活監視

それでは、実際にどうやってクラスタ上のサーバをお互いに監視しているのだろうか？　他の分散データベースのように監視を専任とするサーバはおらず、基本的にはN対Nで互いに監視している。実はこれは、Riakの機能ではなくErlang/OTPの機能(Distributed Erlang)を利用している。

Distributed Erlang[注1]は、デフォルトでは15秒おきにKEEPALIVEの用の通信をやりとりしあう。最後のやりとりから60秒を超えていると、そのノードはクラスタから脱退したものとすることになる。それでは、実際にDistributed Erlangでの死活監視の挙動を確認しよう。1台のマシンでプロセスを分けたり、OSを分けたりするのではカーネル経由でリンクダウンなどが検出されてしまうので、たとえば片方のプロセスをkillした瞬間に故障を検出してしまう。

実際にサイレント故障を再現するために、有線LANでつながった物理マシンでの接続と切断を検証する。**図2**のような構成で、左右のマシンがDistributed Erlangで接続しているときに＊の部分のEthernetケーブルを抜くとどうなるかという実験だ。

まずは、Aのマシンで Erlangを起動する(**画面1**)。**-name**でクラスタ上での識別子とIPアドレスを指定する。**-setcookie**はErlangプロセス同士が接続してよいか確認するためのもので、これが一致していないと接続することはできない。あらかじめBでErlangを起動しておく。

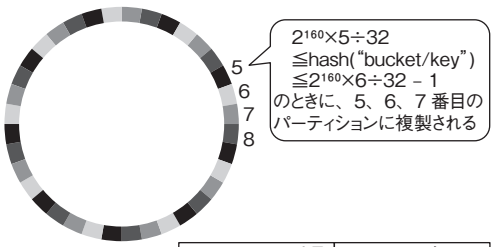
次に、Aで起動してBにDistributed Erlangの**ping**を送信する(**画面2**)。

画面2の、この**net_adm:ping/1**は、pingという名前ではあるが、Bとの接続がなければBと接続する動作も含まれている。無事にBと接続できれば、**pong**が返る。接続できていない場合はすぐに**pang**が返る(※注：これはpingではクラスタメンバに入っているどうかをチェックするためで、このたびに相手にパケットを送信して接続性を確認しているわけではない。クラスタメンバについては後述する)。

Distributed Erlangは非常に優れていて、RPC(Remote Procedure Call)呼び出しも通常の関数とほぼ同様にできる。たとえばA側で**画**

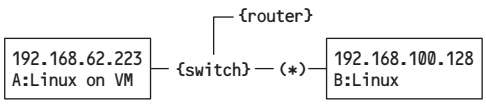
注1) http://www.erlang.org/doc/reference_manual/distributed.html

▼図1　キーの位置を発見するしくみ。ringとよばれるデータは、パーティション番号からノードIDを知ることができるマップになっている



パーティション番号	ノード名
…	…
5	riak@10.0.0.31
6	riak@10.0.0.27
7	riak@10.0.0.25
…	

▼図2　実験の構成



▼画面1　Aのマシンで Erlangの起動

```
$ erl -name dev@192.168.100.128 -setcookie foo
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:8:8] [async-threads:0] [kernel-
poll:false]

Eshell V5.9.1  (abort with ^G)
(dev@192.168.100.128)1>
```

画面3のように実行すると、B側では実際にコンソールに、

```
(dev@192.168.100.128)1> ["hogehoge~n"]
```

と表示される。

このように、Distributed Erlangを使えば、リモートマシンでも簡単にプログラムを起動できる。ほかにも、リモートマシンにも透過的にメッセージを送るなど優れた機能が多くあり、RiakはこのDistributed Erlangの機能を多用している。

net_kernelというモジュールのmanpageを見るとおおまかなデザインが見えてくるだろう。中でも重要なのはTicktimeという概念で、要はタイムアウトを検出するタイマーなのだが、この間隔は、**net_kernel:set_net_ticktime/2**で設定できる。

実際には、15秒間隔でErlang/OTPがKeepaliveをやりとりしている。さらに、最後にKeepaliveが成功したときから60秒経っても応答が1つもない場合、そのノードとの通信が途絶えたものとする。

この動作を実際に検証してみよう。まずは**erlang:monitor_node/2**を使ってAにBを監視させ、次にLANケーブルを抜いてどうなるかを観察する。期待される動作としては、ケーブルを抜いてほぼ60秒後に**{nodedown, 'dev@192.160.100.128'}**のメッセージが**net_kernel**から送られてくる、というものだ(**画面4**)。

実際に21:09:30という時間が表示された瞬間にLANケーブルを抜くと、1分ほど待って**nodedown**のメッセージが到着する。これとタイミングはある程度前後するが、B側のマシンでも

```
=ERROR REPORT==== 19-Jun-2013::21:10:22 ===
** Node 'dev@192.168.62.223' not responding **
** Removing (timedout) connection **
```

というメッセージが表示される。メッセージ中の時間が秒単位で一致しているのは偶然なので、実際の大規模環境ではこうならないこともある。

もし可能であれば、同じマシン内で2プロセ

▼画面2　Bにping送信

```
$ erl -name dev@192.168.62.223 -setcookie foo
Erlang R15B01 (erts-5.9.1) [source] [64-bit] [smp:2:2] [async-threads:0] [kernel-poll:false]

Eshell V5.9.1  (abort with ^G)
(dev@192.168.62.223)1> net_adm:ping('dev@192.168.100.128').
pong
```

▼画面3　A側からRPC呼び出し

```
(dev@192.168.62.223)2> rpc:call('dev@192.168.100.128', erlang, display, [["hogehoge~n"]]).
true
```

▼画面4　LANケーブルを引き抜くと……

```
(dev@192.168.62.223)3> erlang:monitor_node('dev@192.168.100.128', true).
true
(dev@192.168.62.223)4>  erlang:display(erlang:localtime()), receive R -> {R,
erlang:localtime()} end.
{{2013,6,19},{21,9,30}}
 ここで192.168.100.128の方のLANケーブルを抜く
{{nodedown,'dev@192.168.100.128'},{{2013,6,19},{21,10,22}}}
(dev@192.168.62.223)5>
=ERROR REPORT==== 19-Jun-2013::21:10:22 ===
** Node 'dev@192.168.100.128' not responding **
** Removing (timedout) connection **
```

スを立ちあげてBのUNIXプロセスをkillしたときの動作と比較していただきたい。この場合はkillを受けたカーネルがA側のソケットに通知してしまい、A側でもすぐにBが落ちたことを知ることができる。

これがDistributed Erlangの死活監視の挙動である。L2以上で切断を検知できる場合はたいてい、すぐに通知がアプリケーションまで得られるが、L1のレベルで切断が起きた場合には1分程度時間がかかる。

通知を受けたRiakの動作

このようにして、Distributed Erlangから**nodedown**を知ったRiakは、PUTなどのリクエストの転送先を調整し、リング上の次のノードへハンドオフ(※注:Handoffとはvonodeへのリクエストを代わりに処理させること)を行う。このようなしくみによって、ネットワーク上のサイレント故障を除いてほとんどのケースでRiakは正しく動作しているノードにアクセスできることが保証されている。サイレント故障の場合であっても、クライアントが待たされる時間は60秒程度である。

ほかにとくにブロックする処理はないように設計されているため、クライアントからのリクエストを待たせることはほとんどない。これがRiakの高可用性を可能にするしくみである。

障害時の挙動と対処

ノード障害時

ノード障害も、いくつか場合がある。HDDが故障してデータが消えた場合と、そうでない場合だ。

HDDが故障していない場合(Riakがデータを保存していたディレクトリが無事な場合)は非常に簡単である。マザーボード、電源、メモリなど壊れたと思われる部品を交換し、OSをインストールしなおして、故障前と同じIPアドレスを付与する。そしてRiakを起動すれば、以前と同様にクラスタに参加するだろう。落ちていたノードが復活すれば、故障中に他のマシンがHandoffで受け取ったデータを転送し始める。その様子は**riak-admin transfers**で観察できる。

HDDが故障し、そのマシン上のデータが失われた場合は、レプリカからデータを復元する必要がある。この状態では、他のマシンは一時的な障害だと認識し、また同じIPアドレスで起動してくることを待っている(一般的には、一次障害と恒久的障害の区別は自動的にはつきにくい。そこでオペレータがそれをクラスタに指示する必要がある)。代替マシンをすぐに準備できない場合には、**画面5**のように故障したマシンを、クラスタから強制的に外す指示を出す。

これで、クラスタ内でvnodeを再配置し、2に縮退していた複製数を3に戻す操作が行われる。その後、新しいマシンを追加するのは前回も説明した通常の**join**の操作で実行できる。

もしもすぐに準備できるなら、故障したマシンをクラスタから外し、新しいマシンをクラスタに追加するという処理を同時にやるとよいだろう(**画面6**)。

これによって、vnodeの再配置の処理が一度に行われるようになり、ネットワーク上の

▼画面5　クラスタの強制排除

```
$ riak-admin down riak@10.1.2.3
$ riak-admin cluster force-remove ↵
riak@10.1.2.3
$ riak-admin cluster plan
$ riak-admin cluster commit
```

▼画面6　新しいクラスタの追加

```
$ riak-admin down riak@10.1.2.3
$ riak-admin cluster force-remove ↵
riak@10.1.2.3
$ riak-admin cluster join riak@10.1.2.4
$ riak-admin cluster plan
$ riak-admin cluster commit
```

vnodeの移動を節約できる。

復旧するまでの間、できる操作とできない操作がある。PUT/GET/DELETEなどのキーをもとに探索する操作はノード障害時も問題なく実行できるが、MapReduceやlistkeysなど、クラスタ内の全マシンにアクセスする操作の場合は、複数ノードに障害が起きて特定のvnodeのすべての複製が見えなくなっている場合は初めから実行できないようになっている。

ネットワーク障害時

稀(まれ)にスイッチやルータが故障することがある。このときはそのスイッチやルータ配下のマシンが停止する。ネットワークの構成によっては、クラスタの一部が故障によって見えなくなってしまう。たとえば、50台で動作しているクラスタのうち25台がスイッチの故障により見えなくなったとすると、アクセス可能な残りの25台で、これまでと同様にWriteやReadのリクエストを処理しなければならない。負荷が2倍になってしまわないように、なるべく流入するトラフィックをスロットリングできるとよいだろう。

また、ノード故障と同様に利用できなくなるAPIも多いだろう。たとえば、前述の例のとおり半数が故障していた場合、半分のデータは読み取れなくなる。Consistent Hashingで分散しているので、どのデータが読めなくなるかはわからない。また、全キーを取得したり、MapReduceを実行するといったリクエストは、アクセスできないvnodeが1つでも存在すると実行できない。

しかし、データが消えたり復旧時にリセットされるようなことはないので、流入するトラフィックを抑えつつ、落ち着いてネットワークを復旧させることを目指すしかない。もしも落ちているRiakプロセスがいた場合は`riak start`として起動すればよい。

ディスクフルのときにどうすべきか

ほかの多くのシステムがそうであるように、残りのディスク容量が0になった時点でシステム設計や運用の失敗である。そうならないために慎重にシステムを設計する。

Riakも同様で、ディスクフルになると潔くプロセスが終了するようになっている。

したがって、ノードのディスクが満杯になりプロセスが落ちると、Handoffによって他のノードへ書き込みのリクエストが行くようになる。するとHandoffを受けたノードもディスクフルになり……というドミノ倒しの現象になるだろう。結果的に、最初のノードがディスクフルになる前に、PUTやPOSTなどのデータが増えるリクエストをすべて止めるべきだ。前段にロードバランサがあるなら、そこでPUTリクエストだけをフィルタするようにするとよい。

それでも、マシン台数が少ないときにvnodeの配置が偏ったり、どうしてもノード追加が追いつかない場合にディスク容量がどんどん切迫していくことはあるだろう。そういうときには、まずはノード追加をすればよい。再配置やコンパクションに倍の容量を必要とするようなことはない。基本的にはディスク容量は2割程度の余裕を持っておくのがよい。

ただし、キーを削除してもすぐに容量が空くわけではなく、削除のリクエストを処理した時点ではBitcaskやLevelDBはデータを削除するための削除フラグを追記するだけなので、わずかだがディスク使用量は増える。次にコンパクションが起きるタイミングでディスクからは削除される。したがって、コンパクションが実行できるだけの容量は最低限必要だ。

一般にコンパクションはヒープGC(Garbage Collector)のようなアルゴリズムで行われるため、コンパクションを実行するためには同量の空き容量が必要だ。しかし、Riakではvnodeの単位で1つのDBインスタンスになっており、さらに、BitcaskもLevelDBも非常に小さな単位でコンパクションを行うため実際にはわずかな空き容量で十分だ。それでも空き容量が足りない場合は、最後の手段だが、vnodeのディレクトリを削除してしまうという方法がないわけ

ではない。

これはレプリカが3個あるから、ノードの再配置の際にRiakのプロセスを安定して起動させるためにレプリカを2個に減らして容量を空け、プロセスが起動している間にノード追加なりをして容量が空くことを前提にしなければならない。非常にリスクが高い方法なので、ほかに選択肢がない場合の最後の手段だ。

可用性とは何か?

ここまで、Riakの障害時の挙動や、死活監視のしくみ、対処方法などを説明してきた。これらはすべて現実の問題で、システムを開発・運用するときに必ず考えなければならないことだ。他の分散データベースと比較すると、マスターやスレーブなどの役割分担がなく、そのために障害時の対処のパターンが比較的少ないことがおわかりいただけるかと思う。

CAP定理とRiakの可用性

CAP定理は**整合性**、**可用性**、**分断耐性**の3つの特性を同時に実現するのは不可能だとされる命題だ。これを証明したという論文も存在するが、一般的には分散システムにおける経験則である。Riakはこのなかでも可用性にフォーカスしており、どんなときでも書き込める、読み出せることが設計思想の根本にある。ほかの2つの性質もまったくないということではなく、ある程度はユーザに委ねている。

整合性は古くて新しい問題であり、基本的には複数のレプリカの整合を保証するためにロックをしたり、2相コミットのように複数のリクエストを複数のフェーズで待ち合わせるなどの処理が必要になる。そのために待ち時間が発生し、安定したレイテンシを期待できない場合がある。また、分断耐性を守るためには、システム全体が分断していない(たとえば、複数のマスターがいない、同じデータの更新を複数ヵ所で許さない)ことを保証するために、HBaseやBigTableのように中央管理型のシステム構成にしなければならない。

逆に、これらの性質についてある程度妥協すると、分散システムの設計は非常にシンプルにできる。ロックを作らず、複雑な待ち合わせをしなければ処理がブロックされることはほとんどないから、レイテンシは安定する。また、マスターを用意しなければ、マスターの故障やフェイルオーバーのために全体が停止することもないし、メタデータのトランザクションを設計する必要もない。こういった点について不要な複雑さをそぎ落として可用性を追い求めた結果が、今のRiakの設計である。

まとめ「高可用システムを実現するために」

ここまで、Riakの死活監視のしくみと設計の考え方、それぞれの障害への対応方針について書いた。

何よりも大切なのは、事前に備えることである。最初のクラスタは、きっと予算が小さいだろうから、小さなマシンの5台で構成していてもよいだろう。大きな負荷やデータの増加が来る**前に**、システムのキャパシティを十分に引き上げておくことが重要だ。そのためのベストプラクティス[注2]も公式のドキュメントに用意されている[注3]。

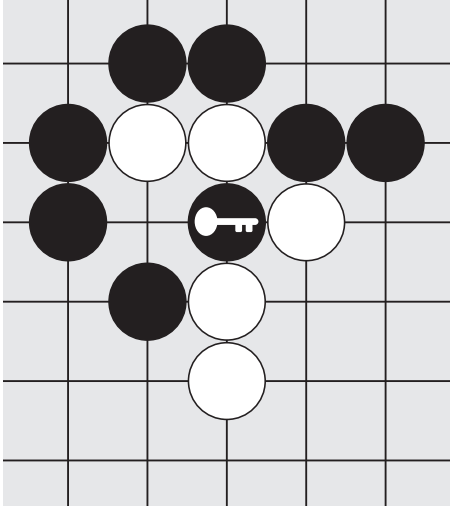
Riakなら、クラスタを止めずにさまざまなアップグレードを行うことができる。サーバは当然のことながら、うまくやればネットワークの更新もシステムの部分停止だけでできるだろう。SD

注2) http://docs.basho.com/riak/latest/cookbooks/best-practices/

注3) 参考URL
http://www.cse.psu.edu/~gcao/teach/513-00/c7.pdf
http://www.cs.duke.edu/courses/fall07/cps212/consensus.pdf
http://rodin.cs.ncl.ac.uk/Publications/avizienis.pdf
http://www.erlang.org/doc/man/net_kernel.html#set_net_ticktime-2
http://docs.basho.com/riak/latest/cookbooks/Linux-Performance-Tuning/

セキュリティ実践の基本定石

すずきひろのぶ
suzuki.hironobu@gmail.com



みんなでもう一度見つめなおそう

【第二回】今、流行の標的型攻撃、APT攻撃とは

今回は、世間で「標的型攻撃」や「APT（Advanced Persistent Threat）攻撃」、あるいは新しい攻撃などと呼ばれている攻撃について取り上げ、考えてみます。

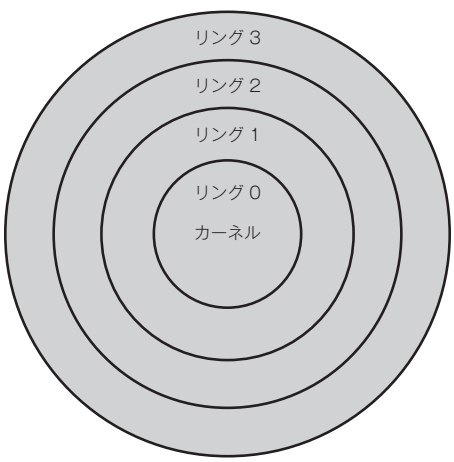
コンピュータ侵入の歴史

これまで、コンピュータへの侵入とその対策はどのような経緯をたどってきたのでしょうか。まずはその歴史を振り返ってみます。

Sneakers

1950年代から70年代のコンピュータが生まれて初期のころは、コンピュータは高価な機材であり、アクセスできるのはオペレータと極めて少ない特権的なユーザだけ、という特殊な環境でした。ゆえにコンピュータはその価値に見合う重要な情報が入っていました。それは軍事システムであり、銀行の帳簿管理であり、政府の扱う国家レベルのデータでした。重要な情報を扱うため、当然ながら情報の保護機能がありました（**図1**）。このころのコンピュータは汎用機と呼ばれるもので、今日的なPCの出現は

◆**図1　階層型保護ドメイン（hierarchical protection domains）**



利用権限は階層的に与えるというもので、これには最小権限しか与えないという考え方がベースになっている。
この考え方は 1964 年の Multics にはすでに取り入れられていた。
リングの考え方は「計算資源のアクセス権限を段階的に持つ」というものである。図のリング０が何にでもアクセスできる最も権限が強い状態で、外周にしたがって弱くなる。利用するレベルに対して適切な権限（Privilege）が与えられており、リング０がカーネルに対応し、リング１と２がデバイスドライバーに対応し、リング３がアプリケーションに対応する。
たとえばUNIX だと典型的なのがリング０がカーネルモード、リング３がユーザモードにあたる。

まだまだ先でした。

そのような時代には、コンピュータに侵入するという行為は、特定のターゲットに特定の意図を持って行われていました。もちろん端末にアクセスする物理的な制約もありました。

良い資料として、みなさんに映画『Sneakers』(1992)を鑑賞するのをお勧めします[注1]。この映画の中には、情報を盗む数々のテクニックが出てきています。90年代に入り濫造される、ただのクラッカーのことをわざわざハッカーと呼ぶ何もわかっていない映画からみると格段の出来です。

80年代の中期に入り、今日的なPCが現れました。やがてPCはビジネスや家庭だけではなく、あらゆるところで、あらゆる場面で重要なツールとして使われるようになります。

重要な情報がこれまでの汎用機の中ではなくPCの中に分散していきます。人々はそれをダウンサイジングと呼びました。しかし、セキュリティに関してはPCは汎用機のダウンサイジングではなく、何もないところからのボトムアップでした。汎用機に備わっていた貴重な情報を守る機能は、考慮されておらず、電気をつければパスワードもなく、いつでも誰でも使える「便利」なものでした。

PCは最初、1人で使うことしか想定されていませんでした。十分なアクセス権限能力を備えておらず、ソフトウェアはどのようなシステムへもアクセスできることを前提に作られていました。また、PCのリソースに対して何の規制もかけることなく何でもできることが、「便利で使いやすい」という言葉と同等の意味を持っていました。

しばらくするとPCがローカルエリアネットワークに接続されていきます。このころになると、ネットワーク経由で誰でもアクセスできるのは困るので、アクセス制御が強化されていきます。しかし、根本的な問題として、アクセス制御の一貫したポリシーはなく、脆弱なシステムでした。PCの基本ソフトウェアは何世代かの時間を経て、やっと一貫したポリシーやアクセス制御を持つようになります。

しかし、そのポリシーも初期は浸透しませんでした。せっかくアカウント権限として管理者権限と一般ユーザ権限を分離し利用することが技術的に可能になったにもかかわらず、多くのユーザはどんな作業も管理者権限のアカウントで行っていました。システムが機能的に要件を満たしても、一度、染みついてしまった利用スタイルを変えるのはなかなか難しいようです。

近年ではPCの基本ソフトウェアもずいぶんと安全性を高めています。しかし、PCにはサードパーティーのソフトウェアも多く搭載されており、サードパーティーのソフトウェアは基本ソフトウェアほど品質が高くなく、そこが突破口になり、システムへの侵入を許すことが往々にして発生しています。

また、いろいろな理由から、過去の安全ではない基本ソフトウェアを現在も使わなければならない、といった問題を抱えている組織などもあります。

レインボーシリーズとSELinux

最近ではSNS記録や通話記録を集め、国民を監視するPRISMプログラムでニュースを賑わしたNSA(アメリカ国家安全保障局)ですが、NSAには、国家安全保障上の要請からコンピュータを保護するための役割も与えられています。

米国国防総省のコンピュータセキュリティセンターやNSAのナショナルコンピュータセキュリティセンターといった組織が、1983年から1993年までネットワークやコンピュータが持つべきセキュリティの仕様を決めてきたドキュメント類があります。その冊子の色によって、オレンジブック(DoDTrusted Computer System Evaluation Criteria : 5200.28-STD)や、レッドブック(Trusted Network Interpretation : NCSC-TG-005)と呼ばれています[注2]。

それらをベースにNational Security Agency's

注1) Youtubeでは"Sneakers (1/9) Movie CLIP - Professional Bank Robber (1992) HD"というタイトルでビデオクリップが紹介されています。ロバート・レッドフォード、シドニー・ポアチエ、ダン・エイクロイド、リヴァー・フェニックス、ベン・キングズレーという名優らが出演しています。

注2) 本の色がカラフルなので、これらの本を総称してRainbow Book(レインボーブック)と呼んでいます。

Trusted UNIX（TRUSIX）Working Groupが作った仕様を実装したのがSELinuxです。軍事あるいは政府の機密を扱うのに必要と考えられているレベルのアクセス制御などの仕様が入っています。この仕様が搭載されていなければ、軍や重要な情報を扱う政府関連機関に導入することができません。

これで問題はある程度解決したのかというと、なかなかそういうことにはなりません。SELinuxは少しでも設定に矛盾があれば動かないので、オープンソースのサーバアプリケーションをインストールする方法を書いているブログなどを見回してみると、まずSELinuxの機能を無効にするところから始まるものが目につきます。

SELinuxのようなシステムで自由に動き回れるマルウェアを作るのは、かなり困難になります。安全ではありますが、その代償として正常なソフトウェアをインストールするのも同じ理由から、ひと苦労します。いくつもの独立したライブラリやフレームワークなどが相互に関係性を持つ場合に、必要なファイルに適切なセキュリティコンテキストの設定をほどこし、矛盾を起こさずに動くようにするのはたいへんです。

これは各種アプリケーションやシステムが、SELinuxを前提とせず古典的なUNIXのアクセスモデル（Discretionary Access Control）に基づいて作られているためです。それをアドホックにSELinuxに適用するわけですが、少しでも設定にミスや見落としがあると動きません。筆者はSELinuxを有効にしたままのCentOS 6にRedmine 1.3をインストールした経験をブログに書き留めていますが[注3]、やはりインストールには相応の時間がかかっています。

これらの経験からいえば、まずSELinuxを無効にするというのもしかたがないことかな、と思う部分があります。もちろん、SELinuxや同じ目的を果たすための別のアプローチであるAppArmorのようなメカニズムを広く使ってほしいとは思っています。

しかしながら、やはりセキュリティは、そのコスト（手間）と利便性とのトレードオフですので、それがどこで釣り合うのか難しい問題です。現状では、ただ単純に「やればいい」と一概には言えないのがセキュリティの難しいところです。

PCクライアントがサーバの脅威となる

たとえサーバ側が鉄壁の守りをしていても、そのサーバにログインするPCクライアント側がずさんな管理をしていれば、サーバ側は無防備になります。この弱点を利用したのが、2010年に現れたGumblarです。これの主目的はサーバを狙うことであり、PCクライアントを攻撃するのはその前段階にすぎません。

まず、PC側クライアントがGumblarに感染します。Gumblarは平文で接続先サーバ名とパスワードを記録しているアプリケーションのレジストリ情報や設定ファイルを探し、その情報を窃取します。

Gumblarはサードパーティのアプリケーションがセキュリティを考慮せず、ずさんな情報管理をしているという盲点をつきました。2013年にはBHEK2のような、Java、Adobe Flash、Adobe Readerなど主要なサードパーティ製アプリケーションをターゲットにしたものまで現れています[注4]。

サーバ側では、SSHの公開鍵でのみ受け付けるように対処するなど、できる対応は限られています。しかし、なんであれクライアント側が脆弱であれば結果としてサーバ側はそれに引きずられることになるでしょう。

中学生のサイバー戦争

今年の春ごろ、北朝鮮の対外宣伝サイトが「アノニマス」を名のるグループにクラックされ、情報窃取により会員名簿などが流出しました。マスコミがサイバー戦争だなどと大騒ぎしていたのを、みなさんも記憶しているかと思います。でも、ふたをあけてみれば韓国の中学生の仕業でした。

注3）Redmine 1.3 をCentOS6にインストールする　http://h2np.net/docs/redmine-selinux.html
注4）IIJ-SECT Security Diaryのブログ「BHEK2を悪用した国内改ざん事件の続報」に詳しいのでそちらを参考にしてください。
BHEK2を悪用した国内改ざん事件の続報　https://sect.iij.ad.jp/d/2013/03/225209.html

「なぜ中学生にそんなことが可能なのか？」と思うかもしれませんが、理由は簡単です。たんにサーバに侵入するだけなら、ツールも手法も幅広くインターネット上で流通しており、コンピュータの専門的知識がなくても、どこかのブログに書いてある懇切丁寧な説明をそのまま行えばできるからです。

また、攻撃を受けるサーバ側も知識も経験もない者が突然サーバ管理者になり、見よう見まねでソフトウェアを導入し、管理しているようでは、恰好のターゲットとなるでしょう。

その程度の行為を軍事にたとえてサイバー戦争と呼び、特別視し、意味のない危機を煽るのは、「いかに今のシステムが現実的に多くの脆弱性を抱えているか」という問題の本質をぼかしてしまいます。むしろ、そのように問題の本質をあやふやにすることこそリスクだと筆者は強く思います。

カオスな集団 カオスな攻撃

自己顕示欲に動かされネットを我が者顔にうろつき暴れる輩は昔からいますし、将来もずっと存在するでしょう。一方で、昔と今とで一番違う点は、窃取した情報をマネタライズ、つまりお金にかえるしくみがどんどん進化しているのでないかと思われる部分です。別の言い方をすれば、この手のシステムへの侵入を職業にしている人たちが増えているのではないかと考えられる部分です。

昔はお金になるものといえば、spam業者、クレジットカード情報窃盗、国際電話や有料電話の無断利用でした。どちらかといえば直接的で、付加価値というものもあまりないようなものです。

現在では、（政治的な目的のために）アノニマスが北朝鮮サイトから情報を流出させたり、WikiLeaksのように組織内部の情報を引き出し暴露するということが行われている現実があります。これと同様の手法で窃取した情報をお金に替えてしまおうという集団もいると考えるのが妥当です。もし情報バイヤーがいるならば、クレジットカード番号やspam名簿などよりもはるかに付加価値がついた商品になるでしょう。

軍事的な意味で相手の情報を窃取する行為を行っていてもさほど不思議ではありません。それも一種の付加価値商品の窃取だと言えます。それだけではなく、2011年当時のドイツでは独司法省が捜査のために、マルウェアを作成し使用することを許可し、実行しています[注5]。

何を言いたいかというと、この状況を見た場合、あまりにも潜在的参加者が多く、誰が何をやっているのか極めて見通しの悪い状況、つまり相当なカオスだということです。守る側にしてみれば、これほどどの方角から攻撃が来るか予想がつかないのは、かなりやっかいな状況です。

Advanced Persistent Threat

APTというのは直訳すると「高度な継続性を持つ脅威」となります。NIST（アメリカ国立標準技術研究所）のドキュメント[注6]などにその性質は説明されていますが、まとめると次のようなことになります。

NISTの定義するAPTの特徴

- 攻撃者は十分な技術力およびリソースを持っている
- 攻撃者はネットワーク上だけではなく、物理的な方法やデセプション（詐欺／欺瞞）など多様な攻撃をとることができる
- ターゲットとなる組織内へ侵入し、活動のための足場を確保する
- 内部から必要な情報を窃取する、あるいはコントロールするなど目的を実行できる
- または、目的を果たすときが来るまで潜んでいる

これらの条件は、先ほどの中学生のやっていた目的や行為とは明らかに違うレベルのものを指しています。ただし、APTの説明にはいくつもの要素が入り込んでいて、論点がわかりづらくなっています。

注5）Germany spyware: Minister calls for probe of state use　http://www.bbc.co.uk/news/world-europe-15253259
注6）Managing Information Security Risk　http://csrc.nist.gov/publications/nistpubs/800-39/SP800-39-final.pdf

そもそも「攻撃者は十分な技術力およびリソースを持っている」というのは、最初の攻撃時点でどうやってわかるのでしょうか。

一方で、Microsoft社がAPTに対する興味深い批判をしています。この考え方は極めて妥当だと思います。

APTのような漠然としたな(ママ)用語を使うことは、この現実があいまいになり、そして、すべてのこのような攻撃は、技術的に高度でマルウェアを利用する、そして、効果的な対策が難しいという印象を与えることで、効果的な防御対策を難しくします[注7]

ケビン・ミトニック

1990年代中頃、ケビン・ミトニックという人がコンピュータに侵入し逮捕されました。特定のコンピュータに侵入するために、技術的な方法では難しい場合、彼とその仲間は、デセプションを使っていました。つまり、詐欺的な方法で人を騙して情報を取り、その情報を元に侵入していました。

彼らは、それをソーシャルエンジニアリングと仰々しい名前をつけていましたが、それは単なるデセプション(詐欺/欺瞞)です。侵入技術といっても、オリジナルなものを開発したわけではなく、すでに知られている方法を用いただけでした。しかし、それでも十分に侵入には成功するのです。最後は、彼より技量が優れていたシステム管理者には歯がたたず逮捕されてしまいましたが。

ケビン・ミトニックのやっていたことは、NISTの定義するAPTそのものです。つまり、APTと呼んでいるものは、今に始まったことではないのです。

標的型攻撃

マルウェア侵入ケース

標的型攻撃の中で、次のようなマルウェア侵入のケースを考えてみます。マルウェアが侵入した後を考えるのはここではなく、別の機会に譲りたいと思います。ここでは内部に足がかりを作るまでを考えてみます。

メール添付ケース

①マルウェア添付のメールが特定の人物に送られる
②ウィルス対策ソフトでは対応されていないマルウェアなので対処できない
③PC内部にマルウェアが感染

誘導URLケース

①特定の人物にメールで、マルウェアを仕込んだサイトへ誘導するURLを送りつける
②メールは「申し込みがエラーとなりました」というようなうっかりURLを確認してしまうような内容
③メール自体は文章(テキスト)として認識されるので一般的なウィルス対策ソフトの対象にはならない
④誘導サイトにはSSLで接続
⑤誘導サイトにはAdobe Flash Player、Adobe Reader、Oracle Javaの脆弱性などを使った感染を引き起こすコードが用意されている
⑥PC内部にマルウェアインストーラが侵入
⑦マルウェアインストーラがマルウェア本体をダウンロード

これまでユーザが実施するべきウィルス対策の基本とされていた「怪しいサイトには近づかない」「怪しい添付ファイルは開かない」「ウィルス対策ソフトは常に最新にしておく」といった極めてプリミティブな心がけはもう意味をなしていません。

Google Chromeなどはアクセス先サイトに不審なコードがあると警告を出しますが、サードパーティーのアプリケーションやプラグイン(たとえばpdfファイルなど)に関しては、必ずしも有効に動作するというわけではありません。

さて、メール添付ケースと誘導URLケースのそれぞれの対策を考えてみましょう。前者の場合、

注7) Microsoft社「標的型攻撃および決意を持った敵対者—高度な技術と豊富な資源を備えた攻撃者からの脅威」 http://www.microsoft.com/ja-jp/download/details.aspx?id=34793 オリジナルの英語版は"DETERMINED ADVERSARIES AND TARGETED ATTACKS"となっています。日本語版のタイトルはもともとの主張と矛盾しているような気が……。

ウィルス対策ソフトが知らないマルウェアだった場合は対応できません。マルウェアの侵入を考えて、メールを読む環境と情報を扱う環境をはっきりと分けるべきでしょう。筆者はメールの添付ファイルは直接自分のコンピュータ環境で扱うことはありません。アプリケーション特有のフォーマットを持つタイプの添付ファイルはGmail上で扱い、Googleの上で確認して、問題がなく手元に保存したいものだけ、別途ダウンロードします。

後者の誘導URLケースの場合、メール自体はテキストですので、添付ファイルをウィルス対策チェッカで検査しても意味がありません。通信を監視しようと思ってもSSLですので防御されてしまいます。いったんマルウェアが侵入して、外部に通信しようとするとき、検知できる可能性はあるかもしれませんが、そのチャンスを逃せばおしまいです。

「不用意にURLをクリックしない」「ソフトウェアやウィルス対策ソフトを最新のものにする」というのは最低限の防御で、本来の積極的な安全策としては、ブラウザをサンドボックス環境で用意するようなことが必要です。

しかしながら、このようなサンドボックス機能が自動的にできる環境は、まだありません。筆者は手元のメールはEmacs環境で読むのですが、不用意にURLをブラウザで開かないように、リンク機能は無効にしてあります。そしてHTMLメールは何かをする前にHTMLコードそのものを見ています。

PC的な「ユーザに便利」という言葉は、内部の情報を隠蔽しユーザに特別な知識を求めないということと同等な意味を持つのですが、それはマルウェアにとって絶好の隠れ蓑なのです。

SELinuxやAppsArmorのような環境を組み入れることで効果は期待できるかもしれませんが、大量のサードパーティーのソフトウェアが使えなくなるなどの副作用も予想されます。

ブラウザや任意のアプリケーションが自動的に隔離された実行環境で、限られた権限で動作し、万が一脆弱性があったとしても他に与える影響を最小限にするといった機能がシームレスに提供されていると便利で安全なのですが……。このような環境が提供されるのはまだ先のようです。今はいちいち手作業で対応するしかないのは、実に残念なことです。

最後に

標的型攻撃、APT攻撃、あるいは新しい攻撃と呼ばれるものについては、攻撃者の強いモチベーションも含めて、技術的には過去のものと変わりはありません。

今回はとくに言及していませんが、最近、軍事的な意味でのサイバー攻撃の話題がそこかしこで聞かれます。しかし、軍事的なサイバー攻撃以前の問題として、前述したとおり、とくにPCを中心としたコンピュータは十分な安全性を確保する利用方法がなされていません。中学生でもアングラサイトに転がっている情報を組み合わせれば、マルウェアを作れるレベルなのです。コンセプトレベルで新しい、本当の意味で新種のマルウェアは年に何度も出てくるものではありません[注8]。しかし一方で、莫大な数の既存の脆弱性を狙うマルウェアのバリエーションが生まれてくるのは、なぜなのでしょう？　当たり前ですが、脆弱性が残されたままだからです。

現状の脆弱な構造と運用をそのままにして、極めて狭い範囲でのコンピュータの防御などを議論するのは、短期的には意味があるように見えますが、根本的な問題解決にはほど遠いでしょう。

ベンダーが提供し続ける脆弱なシステムに振り回されながら、ダマシダマシ使うような後ろ向きな方法論や運用で当面の問題を回避をするのではなく、安全な新しいアプローチから安全なシステムを開発し、積極的に利用する前向きな方法論で解決していくことを願わずにはいられません。時間がかかるように思えますが、急がば回れ、有効かつ根本的な解決策をとるべきだと筆者は強く思います。SD

注8）感染と攻撃を分離したコンセプト実証型のRamenワームや先ほどのクライアント側からサーバ側を狙うGumblarのような新しいコンセプトで作られるマルウェアの出現は全体の総数からいえば極めて稀です。またゼロデイアタックと呼ばれる未知の脆弱性を突くものも頻度は多くありません。

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4回

画像を機種別に切り替える処理を書こう

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/
イラスト●中川 悠京

プログラミングをしたことのない方にもアプリを作る楽しさを味わってもらいたい本連載。今回は、前回で用意した画面サイズと解像度別の画像を、機種に応じて切り替える方法について解説します。

おさらい

前回はさまざまな画像の種類とその用途についてお話ししましたが、今回はそれらの画像を機種に応じて切り替えるためのコードの書き方を解説します。最初からコードの1行1行の意味を理解することは簡単ではありませんが、まずは“こう書いたらこうなる”とだけでもわかるようになれば十分です。

深い理解は少しづつ、徐々に基礎を固めていきながら培っていくものです。たとえまったく意味がわからず難しいと感じても、諦めずにまずは書かれたとおりに真似してみてください。エンジンの構造を理解しなくても車を運転することができるように、プログラムも最初から完全に理解しようとするのではなく、とりあえず動かせるようになることを目指しましょう！

画面サイズに合わせて画像を切り替える方法

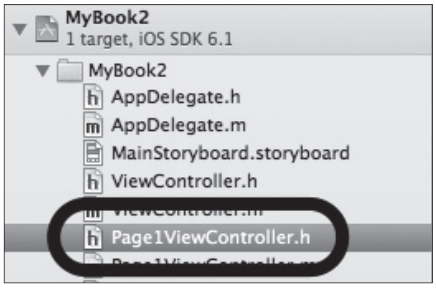
それではいよいよコードを書きはじめますが、より理解度を高めるために、それぞれのコードの意味を「演劇」に例えてみることにします。プログラムは言わば“台本”です。そして画像やボタンなどは表舞台で活躍する“役者”になります。役者は台本どおりに役を演じなければいけませんが、現状（StoryBoard上に画像をドラッグ＆ドロップしただけの段階）ではまだ“役名”が与えられていません。“ただそこに立ったまま何もしなくていい”と言ってるような状況です。画像にはもともと「Page1.png」のようにそれぞれ固有の名前が付いていますが、それはあくまで“実名”でしかなく、実際に劇の中で演じる“役名”はまだ与えられていないのです。自分がどの役を演じるかも知らされていない状況ですから、台本を読んだところで自分がいつ何をしていいかもわかりません。

ステップ1

Xcodeの左カラムのフォルダー一覧「Project Navigator」から「Page1ViewController.h」を選択しましょう（図step1-1）。「.h」の拡張子で終わるファイルは“ヘッダーファイル”と言って、演劇で例えるとシーンごとに登場する役を記載（定義）するページだと考えてください。現段階では、

```
#import <UIKit/UIKit.h>
@interface Page1ViewController : UIViewController
```

step1-1



```
@end
```

の3行しか書かれていないはずですが、@interface Page1ViewController : UIViewControllerの直後に波括弧（なみかっこ）「{」を付け足してreturnを押しましょう。自動的に1段空いて「}」が追加されるはずです。この2つの波括弧の間に、

```
IBOutlet UIImageView *page1Image;
```

を書き加えてください。この行の最後の部分の“page1Image”が役名を表します。第1シーン（1ページ目）に登場するのがこの“page1Image”という役になります。

さらに、閉じた波括弧の後に、

```
@property (nonatomic, retain) IBOutlet UIImageView *page1Image;
```

を加えてください。変更後は図step1-2のようになるはずです。

step1-2

```
#import <UIKit/UIKit.h>

@interface Page1ViewController : UIViewController{

    IBOutlet UIImageView *page1Image;
}

@property (nonatomic, retain) IBOutlet UIImageView *page1Image;

@end
```

ステップ2

次はProject Navigatorから「Page1ViewController.m」を選択しましょう。「.m」で終わるファイルは“実装ファイル”と言い、ここは役ごとの細かい動きを書き込む場所です。どの役が何をして、シチュエーションに応じてどう行動を取るか、具体的に指示が出せます。

まずは図step2-1のように、@implementation Page1ViewControllerのあとに、

```
@synthesize page1Image;
```

を入力してください。これでヘッダーファイルで定義した役が実装ファイルに結びつきます。

次に- (void)viewDidLoadという部分を見つけてください。これはview（場面）が読み込まれたとき、つまりシーンの幕開けを表します。このシーンで何が行われるかの説明をこの中に加えることで、シーンが始まると同時に行ってほしいことを指示できます。

```
- (void)viewDidLoad
{
    [super viewDidLoad];
```

のあとに次のスクリプトを入力してください（図step2-1参照）。

```
if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPhone) {

        CGRect frame = [[UIScreen mainScreen] bounds];
        if (frame.size.height==568.0) {
            [page1Image setImage:[UIImage imageNamed:@"Page1-568.png"]];
        } else {
        }
}
```

ifというのは「もしも」、elseは「それ以外」のときに何をすればいいかを状況に応じて切り替えるための手法です。ここでは使っている機種の画面の高さを確認し、もしも高さが568pxある場合は、page1Imageの役を「Page1.png」ではなく「ß」に代わってもらうようにと指示しています。

さて、ここでなぜ「Page1-568h@2x.png」ではなく、「Page1-568h.png」なのかと疑問を持つ方もいるかもしれません。そもそも@2xが付いていない標準解像度の「Page1-568h.png」は最初から（前回）用意していませんし、プロジェクトそのものにも存在すらしないファイルを呼び出す意味があるのか？と。

前回でも少し触れましたが、プログラムはすべて標準解像度の「ピクセル数」と比例した「ポイント」を基準に動いています。@2x画像はあくまでもRetinaディスプレイ端末が使用されている場合に、プログラム側が優先的に選ぶ画像です。なので、今回のケースの場合、4インチ画面の端末はそもそもRetinaディスプレイしかないので、「Page1-568h.png」を指定することで自動的に「Page1-568h@2x.png」が使われます。仮にここでプログラム上「Page1-568h@2x.png」を指定した場合、プログラムはそのファイル名にさらに@2xが付いた「Page1-568h@2x@2x.png」を探そうとします。しかし同名のファイルが見つからない場合は自動的に「Page1-568h@2x.png」の画像を2倍に拡大した状態で表示してしまいます。つまり本来の4倍の大きさで表示されてしまうということです。

step2-1（※22行目：page1-568.png→Page1-568.png）

```
#import "Page1ViewController.h"

@interface Page1ViewController ()

@end

@implementation Page1ViewController

@synthesize page1Image;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.

    if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPhone) {

        CGRect frame = [[UIScreen mainScreen] bounds];
        if (frame.size.height==568.0) {
            [page1Image setImage:[UIImage imageNamed:@"page1-568h.png"]];
        } else {
        }
    }
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)page1ReturnSegue:(UIStoryboardSegue *)segue
{
}

@end
```

文章を読んでもイメージがつかみづらい場合は、実際に@2xを付けた状態で試してみましょう。プログラムはトライ＆エラーを繰り返すほど理解度が深まるものです。

次に、elseの中に何も書かれていないのは、もともとPage1Imageの役を「Page1.png」に割り振っているので、高さが568px以外のときはとくに変更がないまま進めて良いという意味です。

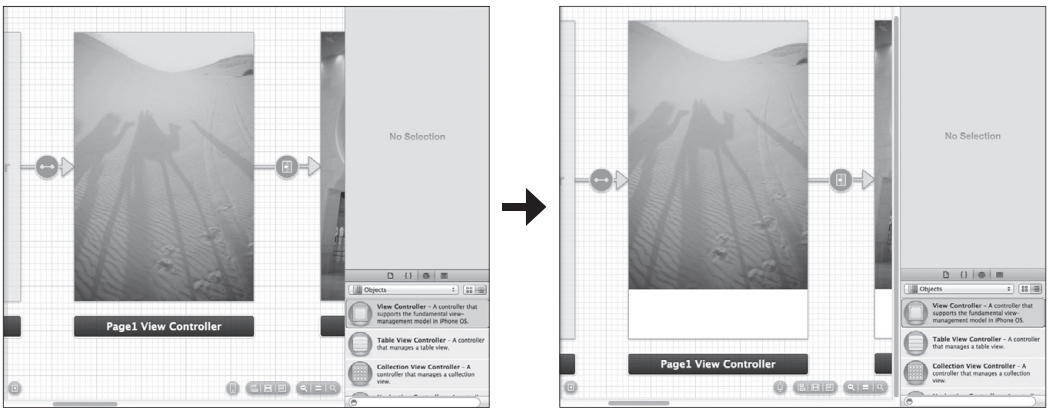
ここまで終えたら、1ページ目以降のView Controllerの「.h」、「.m」ファイルにもステップ1とステップ2をそれぞれ書き加えてください。その際、“役名”はView Controllerごとのページ数と一致するように変更してください。たとえばPage2ViewControllerの場合は「page2Image」、Page3ViewControllerであれば「page3Image」……という感じにです。コードの入力は以上です。

ステップ3

Project Navigatorから「MainStoryboard.storyboard」をクリックします。ストーリーボード内の右下に半透明のボタンが並んでいますが、その中の一番左の丸いボタンを押しましょう（図step3-1）。するとView Controllerの縦幅が伸びることが確認できるはずです（図step3-2）。これで4インチ画面の機種でどのように表示されるかを視覚的に確認できるようになります。

step3-1

step3-2



BEFORE　　AFTER

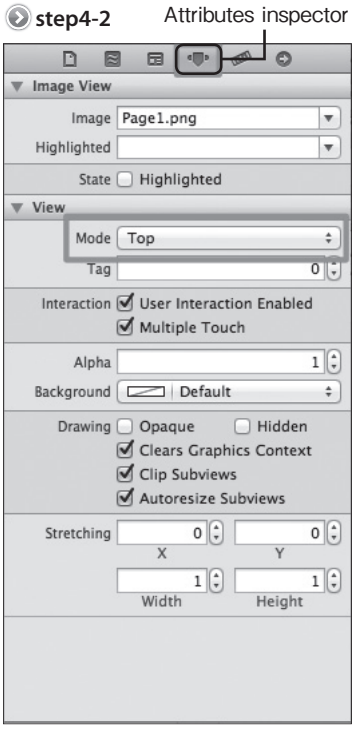
ステップ4

次に図step4-1のように「View Controller Scene」の中の「Image View - Page1.png」をクリックし、Xcodeの右カラムの「Size inspector」でHeightの数値を「480」から「568」に変更してください。

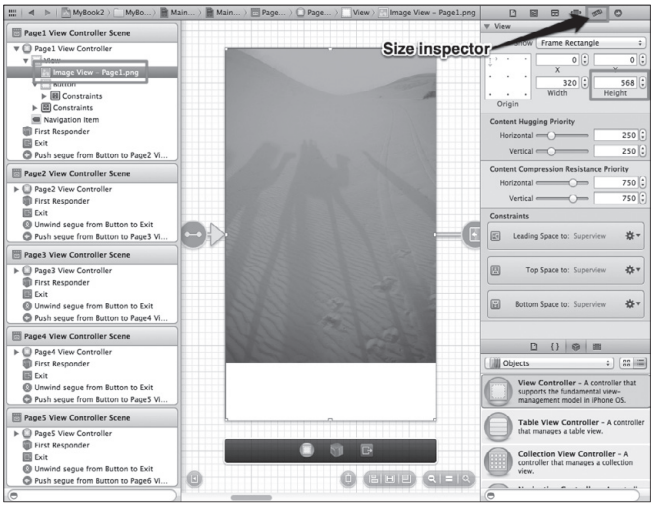
これで画像を表示する範囲を4インチ画面に調整しました。ここを568にしておかないと、画像の中身が4インチ用のものに切り替わっても表示範囲が高さ480なので、その先の部分は途切れてしまいます。

しかし今度は画像が中央寄せになったので、3.5インチのときは上に余白ができてしまいます。これをなおすために「Attribute inspector」をクリックし、Modeが「Center」になっているのを「Top」に変更しましょう（図step4-2）。これで画像が上に寄せられ、3.5インチのときも4インチのときも正常に表示されます。

step4-2



step4-1



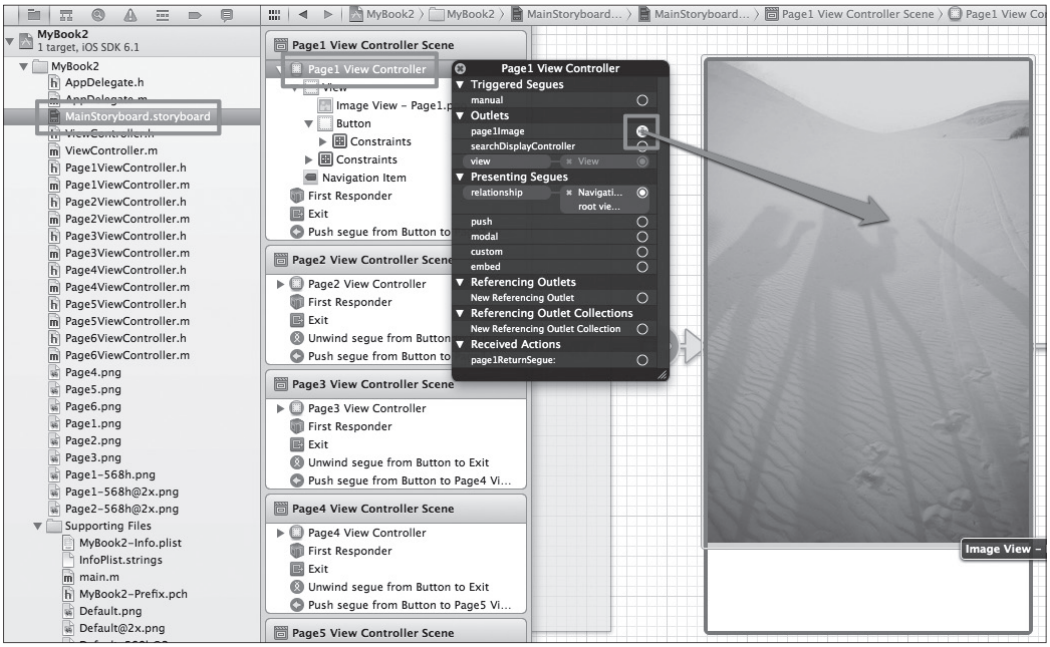
ステップ5

あとは役の振り分け作業になります。台本はできあがりましたが、まだそれぞれの役者に対して役名を教えていないので、誰がどのシーンでどの役を演じていいのかがわからない状態です。役者に役名を伝えれば、あとは役者たちが台本どおりに演じてくれます。

図step5-1のように、ストーリーボード内の左カラムの「Page1 View Controller」をcontrolキーを押しながらクリックすると、ポップアップメニューが出現します。メニュー内にある「Outlets」の中に、先ほどのコードで追加した「page1Image」が含まれています。「page1Image」の名前の右にある丸をクリックし、そのまま1ページ目の画像の上までドラッグして離しましょう。これで「page1Image」と1ページ目の画像がリンクされます。この段階ではじめてこの画像は役名を指定してもらったことになります。

あとは同じことを「Page2 View Controller」、「Page3 View Controller」……とページの数だけ繰り返していき、すべての画像に役名を振り分けていきましょう。これが終われば役者と役名がリンクし、台本どおりにプログラムが動きます。

step5-1



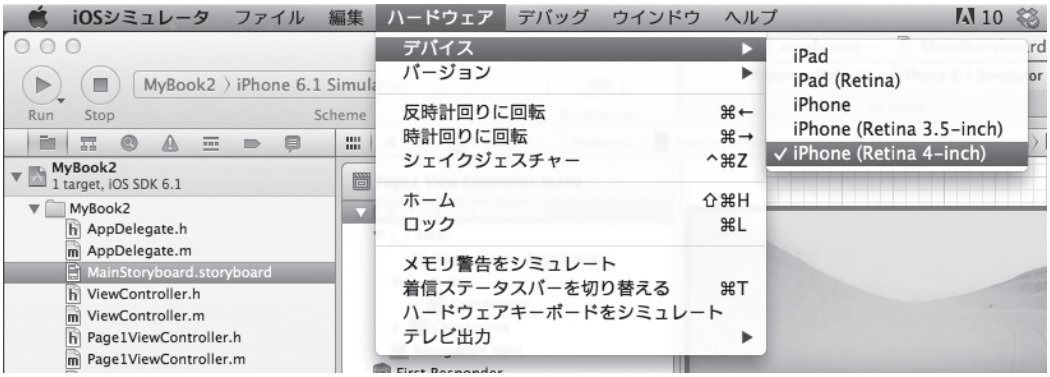
実行して確認

実際に画像が切り替わるかテストを行うため、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。iPhoneシミュレータが起動します。シミュレータが起動したら、図1のようにメニューバーの[ハードウェア]－[デバイス]から機種の変更ができるので、「iPhone Retina(3.5-inch)」と「iPhone Retina(4-inch)」とで切り替えて、両デバイスでどう表示されるかを確認しましょう。

以上で機種別の画像の切り替え方の説明を終えます。今回のような要領でいろいろな画像やボタンなどを配置していき、役名と役割を振り分けていくことがiOSプログラミングの基本になります。今後、徐々にコードを書く量も増えていくので、この基本をしっかり理解しておく

とプログラムの構造がわかりやすくなるはずです。SD

▼図1　シミュレータデバイスの切り替え



Column

コメントの活用

プログラムを書くうえで「コメント」を残すことが大切になってくることがあります。コメントとはコード内に自由にメモを残すことです。部分部分の役割などをメモしておくことで、あとから自分やほかの誰かがコードを見たときにわかりやすくなります。

コメントを書くにはプログラム処理側がコードと区別できるようにする必要があります。Objective-Cの場合はコメントの先頭に「**//**」を加えることで、それ以降の文字から行末までは「コメントアウト」され、プログラムの処理からは除外されます。

何行もまとめてコメントアウトしたい場合は、コメントアウトしたい部分の頭に「**/*** 」を入れ、終わりの部分に「***/**」を入れることで、その間のコードをまとめてコメントアウトできます。ほかにもショートカットとして、選択範囲を command ＋ / でまとめてコメントアウトしたり、コメントアウトを外したりする方法があります。

本連載のサポートページで記事の補足説明をしています。あわせてご活用ください!
› http://www.gimmiq.net/p/sd.html

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(Fusion Calc)」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。

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列車の車窓からーそうだ! 京都行こう!ー」(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。

第39回 設計しやすい高速FPGAが組込みAndroidを変える

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！

小山 忠昭　KOYAMA Tadaaki
FPGAインフォメーション
koyama@fpga.co.jp

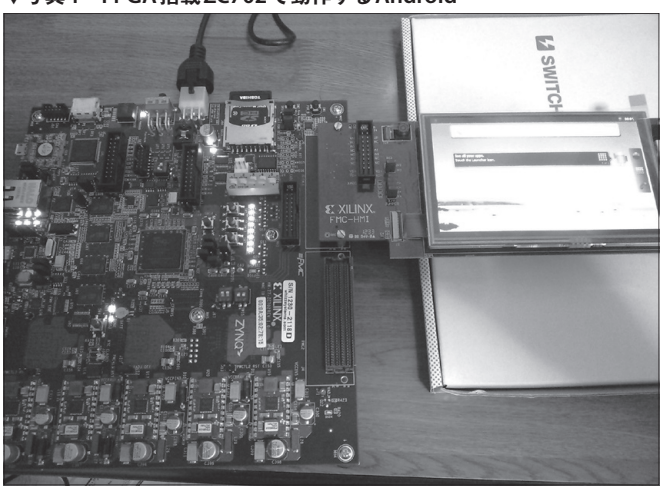
FPGAにAndroidを載せてみる

いきなりですが、FPGA（詳しくは後述します）にAndroidを載せてみました。**写真1**を見てください。Androidのバージョンは少し古いのですが、動作している基板はXilinx（ザイリンクス）社から出ている、「ZC702」という製品です。この基板にXilinx社のFPGAが載っています。

このFPGA、種も仕掛けもございます。FPGAのトップメーカーであるXilinx社が2011年に発表、2012年に発売した「Zynq」（ジンクと読む。詳細は後述）というシリーズのFPGAです。

今までFPGAというと、搭載できるCPUもFPGAメーカー独特で一般的なOSも走らなかったのですが、このZynqではLinuxはもちろん、Androidなどいろいろなオペレーティングシステムが走ります。**写真1**でFPGAにAndroidを載せているのは、操作やI/Oまわりのインターフェースに理由があります。Androidを始めとするスマートフォンが普及したので、タッチパネルといったユーザインターフェースの部材もそれらに対応したものが多く出回るようになりました。また、LANやUSB、SDカードなどのインターフェースはパソコンやスマートフォンだけではなく、あらゆる電子機器に搭載され始めています。つまり、AndroidをFPGAに載せたのは、ユーザインターフェースや各種インターフェースがより早

▼写真1　FPGA搭載ZC702で動作するAndroid



く搭載できるからです。

Zynq自身がまだ新しいFPGAということもあり、Androidを載せたら何ができるかはこれから探していくところです。現在のところ、FPGAの得意分野である、画像処理のデモが多いです。またデータ処理の高速化というのも期待されています。まあ、誤解を恐れずにおおざっぱに言えば、KinectがAndroidに載ったり、ビックデータの処理がAndroidでできるというイメージでしょうか？

FPGAとは

まず、FPGAを知らない人のために、簡単にFPGAの紹介をしましょう。FPGAはField Programmable Gate Arrayの略で、デジタル回路をプログラムできるIC（集積回路）です。1985年にXilinx社から発表・発売され、性能を上げながら今に至っています。

通常、プログラムというのはCPUに対して行うものです。JavaやC言語といったソフトウェア言語からコンパイルを行い、マシン語を生成、それをCPUで逐次実行します。それに対し、FPGAはハードウェア記述言語というハードウェア向けのプログラム言語を使い、それをコンパイルしてFPGA内部データに変換します。内部データはデジタル回路そのものを表します。

FPGAは電源を入れるとFPGA用内部データを読み込んでデジタル回路を構成し、内部データに沿った形でデータを処理します。そのため、必要な処理は複数同時に行うことができ、動作スピード上ではCPUの100倍程度、最適化によっては10,000倍程度の処理速度が可能になっています。

実際に使われている分野は幅広く、現在の電子回路技術はFPGA抜きでは考えられない状況です。製品の特性上、少量多品種に向いています。産業向けが主ですが、一部民生品にも使われており、おもに通信関係や画像処理に強いため、携帯電話の基地局や製造工場の検査装置にはよく使われています。そのほか、高価なテレビやオーディオ機器、鉄道車両や人工衛星などにも入っています。ゲーム機向けではKinect（キネクト）にも搭載されています。変わったところでは、金融機関にも使用されています。市場の変化のスピードにCPUだけでは間に合わない処理が増えてきており、高速化のためにFPGAに演算をさせてその結果を使うというしくみが用いられているようです。最近のトレンドは自動車向けでしょうか？　ぶつからない車など、映像処理が必要な機能が搭載されてきたために使われ始めています。

FPGA設計の現状

処理速度が速いFPGAですが、まだまだ知名度は低く、なかなか気軽に使えてもらえていないのが現状です。その理由にはいくつか心当たりがあります。

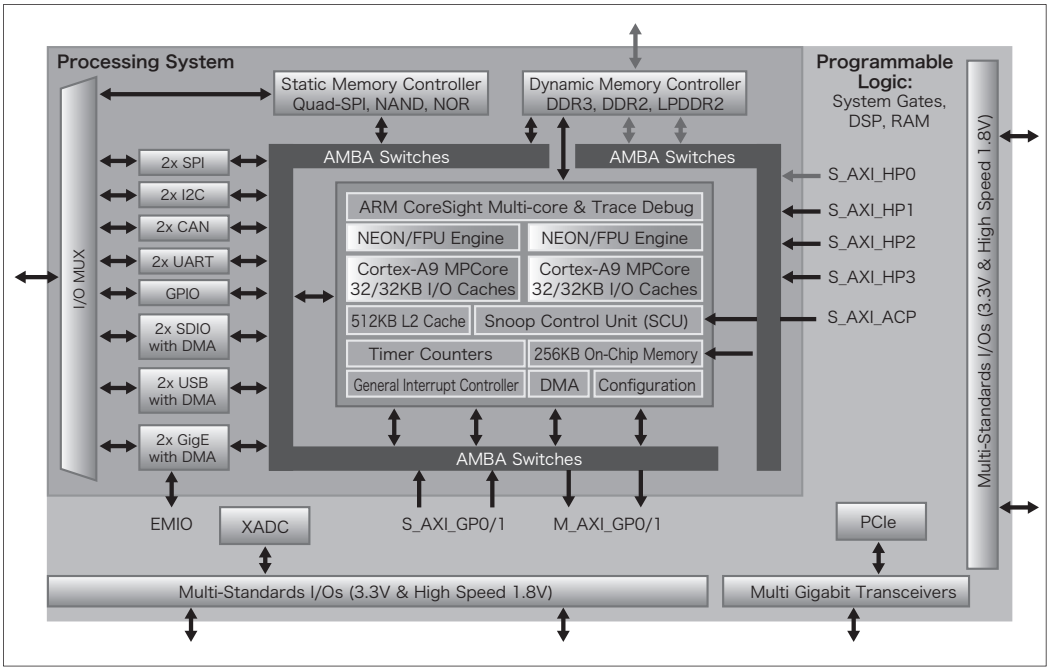
[理由1]もともとはデジタル回路、ハードウェアから発達した

ハードウェアの知識、とくにデジタル回路の設計技術が必要になります。設計ツールもハードウェアの知識を前提として作成されています。最近はハードウェア記述言語（VHDLまたはVerilog）がありますので、C言語などに似た文法でFPGAをプログラムできるようになりました。しかし、文法が似ていてもハードウェア（デジタル回路）を意識した記述にしていないとうまく動作してくれません。設計するうえで隠れた前提条件もあり、まったく同じ記述でもコンパイルの条件で動作しなかったりすることもあります。

[理由2]デバイス単価が高い

最近はFPGAのデバイス単価が安くなったとはいえ、それでもマイコンに比較されることがあります。仕様上マイコンで間に合うのであれば、そのほうが安価にできます。

▼図1　Zynqのアーキテクチャ図



[理由3]FPGAで何ができるかが知られていない

FPGAを仕事にしていると、FPGAで何ができるのかを知らない方が多い気もします。確かにわかりにくい、ということもあるのですが、それが使ってもらえない理由の1つにもなっています。

新FPGA、Zynqシリーズについて

Zynqは Xilinx 社から出た新型のFPGAです。特徴は、1つのデバイスにARM Cortex-A9がデュアルコアで入っているのと、そのための周辺I/Oが搭載されていること、そして最新のXilinxの7シリーズのFPGAが入っていることです。さらにそれらが太いバスでつながっています(**図1**)。

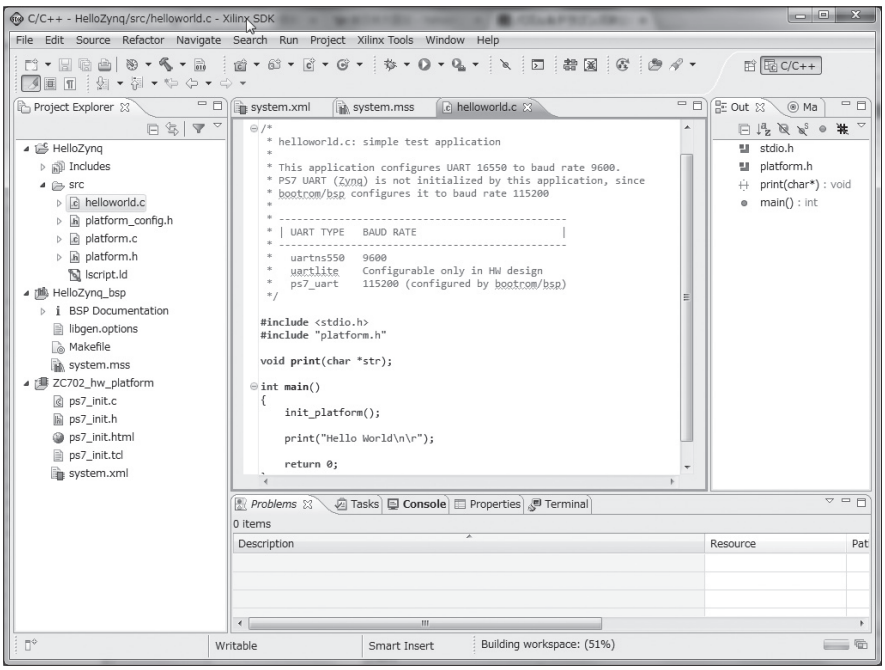
Cortex-A9が入っているので、Androidが動くのも当然です。CPU側にはメモリインターフェースと各種インターフェースが搭載されています。Cortex-A9がデュアルコアで最大1GHzで動作し、キャッシュも内蔵されています。通常のマイコンに必要なものはすべてありますので、すぐ使えると思います。それと、搭載されている同社最新の7シリーズのFPGAは低消費電力とA/Dコンバータの内蔵が特徴です。

Zynqは今までのFPGAの常識をくつがえす(そしてCPUの常識をくつがえす)、数々の機能を搭載したデバイスと言えます。これはハードウェアから見た面とソフトウェアから見た面があります。

ハードウェア面の特徴

CPU + FPGAという構成なら、今まではFPGAに内部でソフトコアを内蔵する、または、CPU + FPGAの2チップ構成にしてしまうという方法がありました。しかし、Zynqはそれらと異なっています。大きな特徴はCPU部とFPGA部分がAXIバスという、太いバスでつながっていること、そしてメモリアクセスがCPU側とFPGA側で一貫性があり、どちらからでも自由自在にアクセスできることです。

▼図2　アプリケーション開発ツール



今までCPUからFPGA側にデータを受け渡すときには、受け渡し用のプログラムの追加、およびハードウェアの追加が必要になり、追加部分の作成と検証が別途必要でした。Zynqでは、CPUで必要な処理を行いメモリに保存し、FPGA側にアクセスしたいメモリポインタを渡してしまえば、追加のハードウェア不要でFPGA側からそのメモリにアクセスが可能になっています。

ソフトウェア面の特徴

今までFPGAのプログラムというと、ハードウェア記述言語を始めとしたデジタル回路の知識が必須でした。Zynqでは、FPGAの部分の設計もC言語で行うことができるようになりました(まだ開発ツールが洗練されていないので、筆者が試したところでは素直に行かない部分もありましたが)。この技術の採用により、どこをソフトウェアにし、どこをハードウェアにするかを区分けせずに、アプリケーションの作成やデバッグが行え、あとで高速化したいところをハードウェアに変更するといったことができます。

Zynqの新しい設計方法

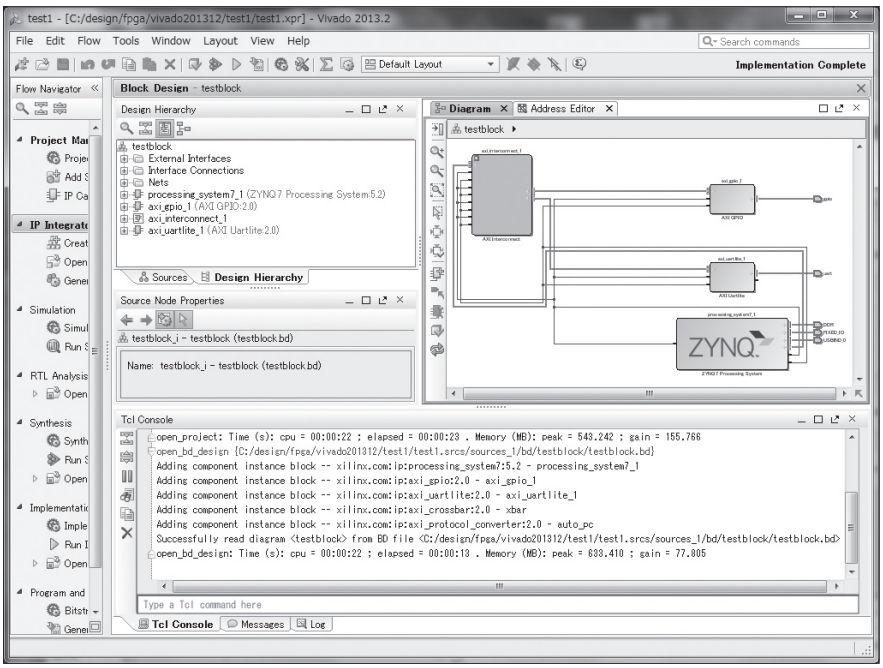
Zynqの設計方法はいろいろあります。従来どおりのハードウェア記述言語をベースに、Cortex-A9やプログラムの設定などする方法はもちろんあります。こちらはたいていのFPGAの入門書などに書いてある方法なので省略します。ここではせっかくなので、これから普及するであろう設計方法を紹介したいと思います。

現在、Xilinx社が提供する開発ツールはいくつかあります。そのうち、大きく分けて3つのツールを駆使して設計することが多いです。

❶アプリケーション開発ツール(SDK)

SDKと言われていることからわかるとおり、ソフトウェア開発ツールです(**図2**)。Cortex-A9のソフトウェア開発に使用できます。Eclipseをベースにした統合開発環境で、CおよびC++で開発できます。Eclipseでの開発経験がある方

▼図3　Vivado 2013.2



は、とくに違和感なく開発できるはずです。

❷FPGA開発ツール　Vivado 2013.2

FPGAの総合ツールで、今後Xilinx社ではメインになるツールです(**図3**)。本来はFPGAにかかわる細かい調整ができる多機能ツールなのですが、C言語でFPGAを作ってしまうと接続情報だけを管理するだけのツールになりそうです。新たに覚えることが多少ありますが、ハードウェア特有の操作がなくなるのが特徴になります。

❸C言語論理合成ツール Vivado HLS 2013.2

図4を見てください。これもEclipseベースのツールです。このツールでFPGAのロジック部分を設計することになります。現在のFPGA設計は、C言語から必要に応じてハードウェア化することができます。C言語で作ったプログラムのうち、高速化したいものをHLS側にコピーし、一部をハードウェアに適した記述にします。多少のプログラムの追加で、CPUでの処理に比べて100倍程度高速に動くプログラムが完成します。

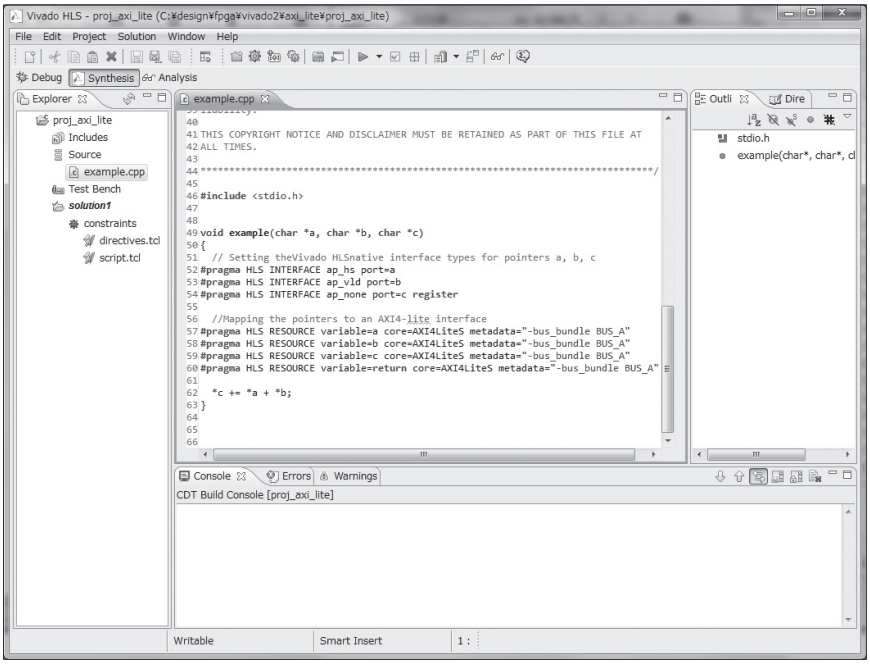
設計の流れは、SDK(❶)を使ってソフトウェアによるプログラムを作成。その中で高速化したい部分をHLS(❸)を使ってハードウェア化。Vivado(❷)を使って、FPGAが動くように調整を行う形になります。

設計ツールが成長段階にあり、まだ一貫性がなく、多少の調整が必要になりますが、ハードウェア記述言語に頼った現在の主流に比べれば、楽に、早く設計ができるようになるでしょう。

AndroidにZynqを使うメリット

ソフトウェアによる処理の数十倍～100倍程度の動作ができる、これがZynqの特徴です。今のスマートフォンにZynqが入ったらいいのですが、さすがにすぐには難しそうです。現状はスマートフォンでの利用ばかりが目立ちますが、

▼図4　Vivado HLS 2013.2



Android自体はスマートフォンだけではなくていろんな機器に使うことができます。スマートフォンの普及でタッチパネルなどのインターフェースが普及したことにより、他の電子機器もタッチパネルありき、他のインターフェースありき、ということが多くなってきます。冒頭での話につながりますが、そのような状況では、Androidが搭載されていると各種インターフェースの制御面で有利になってきます。

逆にAndroidから見ると、もっと処理を高速化したいときにFPGAは有利になります。最近は画像を扱う処理やビックデータの処理などが増えてきたこともあり、CPUだけではなかなか処理が間に合わないこともあります。そのプログラムの一部をFPGA化するだけでも、高速化がやりやすくなります。

スマートフォンに載るのはまだまだ先だと思いますが、もし、ZynqがAndroidスマートフォンに載ることになったら、時代とともに変化する電波方法(LTE→LTE Advanceなど)もプログラムによる変更だけで対応できるので楽しみは増えそうです。

Zynqのような“CPU + FPGA”というデバイスへの注目が集まっており、いくつかのメーカーが製造を計画をしています。プログラムの高速化という永遠の課題がいつまでも続きそうですが、Zynqのようなデバイスが、高速化の手助けになれれば幸いです。SD

小山 忠昭（こやま ただあき）　有限会社FPGAインフォメーション　代表

1985年に初めて世に出たFPGAに、2年後に運良く出会う。それ以降、FPGAにかかわり、今に至る。現在、おもにFPGAの設計、開発を行っている。Xilinx社のトレーニング講師も行っている。日本アンドロイドの会 神戸支部（おもにハードウェアの分野）に参加している。

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11回 virtioによる準仮想化デバイス その1「virtioの概要とVirtio PCI」

浅田 拓也（ASADA Takuya）Twitter @syuu1228

はじめに

前回までに、ハイパーバイザでのI/O仮想化の実装を、BHyVeのソースコードを例に挙げ解説してきました。今回は、ゲストOSのI/Oパフォーマンスを大きく改善する「virtio」準仮想化ドライバの概要と、virtioのコンポーネントの1つである「Virtio PCI」について解説します。

完全仮想化デバイスと準仮想化デバイス

x86アーキテクチャを仮想化する手法として、「準仮想化」と呼ばれる方式があります。これは、Xenによって実装された方式です（※第1回連載参照）。

準仮想化では、仮想化に適した改変をゲストOSに加えます。これにより、改変を加えずゲストOSを仮想化する完全仮想化と比較し、高いパフォーマンスが得られるようになります。この準仮想化では、ハードウェア仮想化支援も必要としていませんでした。

現在では、先に説明した準仮想化よりも完全仮想化が広く使われるようになっています。これは、ハードウェア仮想化支援機能を持つCPUが普及し、多くの環境で高速なハードウェア支援機能が利用できるようになったためです。

しかし、完全仮想化を採用したハイパーバイザにおいても、部分的に準仮想化の概念を取り入れています。その部分としては、システム全体のパフォーマンスに大きく影響を及ぼす仮想デバイスが挙げられます。

このような仮想デバイスのことを「準仮想化デバイス」と呼びます。これに対して、実ハードウェアと同じデバイスをエミュレーションしている仮想デバイスのことを「完全仮想化デバイス」と呼びます。

完全仮想化デバイスでは、実ハードウェア向けのOSに付属しているデバイスドライバをそのまま使用できま。しかし、準仮想化デバイスではゲスト環境向けに、準仮想化デバイス用のデバイスドライバをインストールする必要があります。

準仮想化デバイスのフレームワークとして「virtio」があります。これは、特定のハイパーバイザやゲストOSに依存しないフレームワークです。その仕様やソースコードは公開されているため、ハイパーバイザ側ではKVM・VirtualBox・lguest・BHyVeなど、ゲストOS側ではLinux・FreeBSD・NetBSD・OpenBSD・Windows・MonaOSなど多くの実装が存在しています。

また、XenやVMware、Hyper-Vなどのハイパーバイザでも、同様の考え方を採用した準仮想化ドライバが採用されています[注1]。

完全仮想化デバイスが遅い理由

実機上でネットワークインターフェースやブロッ

注1）virtioとは異なる独自方式のドライバが用いられています。

クデバイスなどに対してI/Oを行う場合、OSはデバイスドライバを介して各デバイスのハードウェアレジスタに対して読み書きを行います。

前回までの記事で解説したとおり、Intel VT-xではこのハードウェアレジスタに対するアクセスを検知するたびにVMExitを行います。ハイパーバイザはVMExitを受けてデバイスエミュレーション処理を行います。

この一連の処理は仮想化を行うときだけに発生するオーバーヘッドであり、この部分の処理の重さが実機と比較した時のI/O性能の差に現れてきます。

より詳細には次のようなコストが発生し、実機上のI/Oと比較してレイテンシが大きくなる可能性があります。

VMX non-root mode・VMX root mode間のモード遷移にかかるコスト

ハードウェアレジスタアクセス時のVMExitとゲスト再開時のVMEntryでは、それぞれVMX non-root modeとVMX root modeの間でモード遷移が発生します。このコストはCPUの進化に伴い小さくなってきているものの、VMExit・VMEntryにそれぞれ1000サイクルほど消費します。

デバイスエミュレータの呼び出しにかかるコスト

多くの場合、ハイパーバイザのデバイスエミュレータはユーザプロセス上で動作しています。このため、ハードウェアレジスタアクセスをエミュレートするにはカーネルモードからユーザモードへ遷移し、エミュレーションを行ってからカーネルモードへ戻ってくる必要があります。

また、ユーザプロセスはプロセススケジューラが適切と判断したタイミングで実行されるため、VMExit直後にデバイスエミュレータのプロセスが実行される保証はありません。

同様に、ゲスト再開のVMEntryについてもデバイスエミュレーション終了直後に行われる保証はなく、スケジューリング待ちになる可能性もあります。

また、たいていの完全仮想化デバイスでは一度のI/Oに複数回レジスタアクセスを行う必要があります(たとえば、あるNICの受信処理では5〜6回のレジスタアクセスが必要になります)。レジスタアクセスを行うたびに、上述の処理が発生し、大きなコストがかかります。高速なI/Oが求められるデバイスの場合には、ここが性能上のボトルネックになります。

virtioの概要

virtioは前述のようなデバイスの完全仮想化にかかるコストを減らすため、ホスト・ゲスト間で共有されたメモリ領域上に置いたキューを通じてデータの入出力を行います。

VMExitはキューへデータを送り出したときに、ハイパーバイザへ通知を行う目的でのみ行われ、なおかつハイパーバイザ側がキュー上のデータを処理中であれば通知を抑制することも可能す。このため、完全仮想化デバイスと比較して大幅にモード遷移回数が削減されています。

virtioは、大きく分けてVirtio PCIとVirtqueueの2つのコンポーネントからなります。

Virtio PCIはゲストマシンに対してPCIデバイスとして振る舞い、次のような機能を提供します。

- デバイス初期化時のホスト⇔ゲスト間ネゴシエーションや設定情報通知に使うコンフィギュレーションレジスタ
- 割り込み(ホスト→ゲスト)、I/Oポートアクセス(ゲスト→ホスト)によるホスト⇔ゲスト間イベント通知機構
- 標準的なPCIデバイスのDMA機構を用いたデータ転送機能

Virtqueueはデータ転送に使われるゲストメモリ空間上のキュー構造です。デバイスごとに1つまたは複数のキューを持つことができます。たとえば、virtio-netは送信用キュー・受信用キュー・コントロール用キューの3つを必要とします。

ゲストOSは、PCIデバイスとしてvirtioデバイスを検出して初期化し、Virtqueueをデータの入出力に、

割り込みとI/Oポートアクセスをイベント通知に用いてホストに対してI/Oを依頼します。

今回の記事では、このうちVirtio PCIについてより詳しく見ていきましょう。

PCIのおさらい

Virtio PCIの解説を行う前に、まずは簡単にPCIについておさらいしましょう。

PCIデバイスはBus Number・Device Numberで一意に識別され、1つのデバイスが複数の機能を有する場合はFunction Numberで個々の機能が一意に識別されます。

これらのデバイスはPCI Configuration Space、PCI I/O Space、PCI Memory Spaceの3つのメモリ空間を持ちます。

PCI Configuration Spaceはデバイスがどこのメーカーのどの機種であるかを示すVendor ID・Device IDや、PCI I/O Space・PCI Memory Spaceのマップ先アドレスを示すBase Address Register、MSI割り込みの設定情報など、デバイスの初期化とドライバのロードに必要な情報を多数含んでいます。

PCI Configuration Spaceにアクセスするには、次のような手順を実施する必要があります。

① デバイスのBus Number・Device Number・Function Numberとアクセスしたい領域のオフセット値をEnable BitとともにCONFIG_ADDRESSレジスタ[注2]にセットする。CONFIG_ADDRESSレジスタのビット配置は**表1**のとおり

② CONFIG_DATAレジスタ[注3]に対して読み込みまたは書き込みを行う

OSはPCIデバイス初期化時に、Bus Number・Device Numberをイテレートして順にPCI Configuration Spaceを参照することで、コンピュータに接続されているPCIデバイスを検出できます。

PCI I/O SpaceはI/O空間にマップされており、おもにデバイスのハードウェアレジスタをマップするなどの用途に使われているようです[注4]。

図1のようにPCI Memory Spaceは物理アドレス空間にマップされており、ビデオメモリなど大きなメモリ領域を必要とする用途に使われているようです。どちらの領域もマップ先はPCI Configuration SpaceのBase Address Registerを参照して取得する必要があります。

Virtio PCIデバイスの検出方法

virtioデバイスはゲストマシンに接続されているPCIデバイスとしてゲストOSから認識されます。

この際、PCI Configuration SpaceのVendor IDは

注2) PCではCONFIG_ADDRESSレジスタはI/O空間の0xCF8にマップされています。

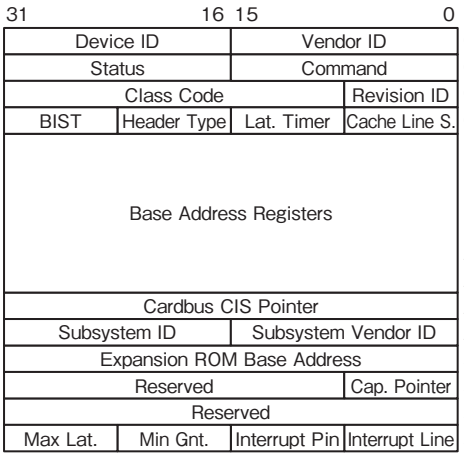
注3) PCではCONFIG_DATAレジスタはI/O空間の0xCFCにマップされています。

注4) PCではI/O空間にマップされますが、他のアーキテクチャではメモリマップされる場合もあります。

▼表1　CONFIG_ADDRESS register

bit	name
31	Enable Bit
30-24	Reserved
23-18	Bus Number
15-11	Device Number
10-8	Function Number
7-2	Register offset
1-0	0

▼図1　PCI Configuration Space



0x1AF4、Device IDは0x1000 - 0x1040の値が渡されます。さらに、virtioデバイスの種類を判別するための追加情報として、**表2**のようなSubsystem Device IDが渡されます。

ゲストOSはこれらのIDを見て適切なvirtio用ドライバをロードします。

Virtio Header

Virtio HeaderはPCI I/O Spaceの先頭に置かれたvirtioデバイスの設定用のフィールドで、ゲストOSがvirtioデバイスドライバを初期化するときに利用されます(**表3**)。

Virtio Headerの終端部分(MSIが無効の場合は20byte、有効の場合は24byte)からはdevice specific headerが続きます。

表4にvirtio-netのdevice specific headerを示します。

▼表2 Subsystem Device ID

Subsystem Device ID	device type
1	network card
2	block device
3	console
4	entropy source
5	memory balooning
8	SCSI host
9	9P transport

Virtio PCI デバイスの初期化処理

ゲストOSにおけるVirtio PCIを用いたvirtioデバイスの初期化処理は次のようになります。

Step 1

通常のPCIデバイスの初期化ルーチンを実行し、Vendor ID・Device IDがvirtioのものを発見します。

Step 2

デバイスのPCI I/O Spaceのマップ先アドレスを取得し、Virtio HeaderのSTATUSフィールドにACKNOWLEDGEビットをセットします(STATUSフィールドが用いるビット値は**表5**に記載)。

Step 3

Subsystem Device IDに一致するドライバをロードします。たとえばIDが1の場合はvirtio-netをロー

▼表3 Virtio header

offset	field name	bytes	direction	description
0	HOST_FEATURES	4	RO	ホストが提供する機能のビットフィールド
4	GUEST_FEATURES	4	RW	ゲストが有効にしたい機能のビットフィールド
8	QUEUE_PFN	4	RW	QUEUE_SELで指定されたキューに割り当てるメモリ領域の物理ページ番号(PFN)
12	QUEUE_NUM	2	RO	QUEUE_SELで指定されたキューのサイズ
14	QUEUE_SEL	2	RW	キュー番号
16	QUEUE_NOTIFY	2	RW	QUEUE_SELで指定されたキューに処理すべきデータがあることを通知
18	STATUS	1	RW	デバイスのステータス
19	ISR	1	RO	割り込みステータス
20	MSI_CONFIG_VECTOR	2	RW	コントロール用キューのMSIベクタ番号(MSI有効時のみ存在)
22	MSI_QUEUE_VECTOR	2	RW	QUEUE_SELで指定されたキューのMSIベクタ番号(MSI有効時のみ存在)

▼表4 virtio-net specific header

offset	field name	bytes	direction	description
0	MAC	6	RW	MACアドレス
6	STATUS	2	RO	リンクアップ状態などのvirtio-net固有ステータス
8	MAX_VIRTQUEUE_PAIRS	2	RO	最大RX/TXキュー数(マルチキュー用)

ドします。

Step 4

ドライバがロードできたらVirtio HeaderのSTATUSフィールドにDRIVERビットをセットします。

Step 5

デバイス固有の初期化処理を実行します。virtio-netの場合、virtio-net specific headerからMACアドレスをコピーする、NICとしてネットワークサブシステムに登録するなどの処理が行われます。この時、Virtio HeaderのHOST_FEATURESフィールドで示されているデバイスで使える機能のうち、ドライバで使用したい機能のビットをGUEST_FEATURESへ書き込みます。FEATURESの全ビットの紹介は省略しますが、たとえばvirtio-netでChecksum Offloadingを使いたい場合はビット0、TSOv4を使いたい時はビット11を有効にする必要があります。

Step 6

デバイスに必要な数のキューをアロケート、Virtio Headerを通じてホストへアドレスを通知します(「キューのアロケート処理」に詳述)。

Step 7

ドライバの初期化処理がすべて成功したらVirtio HeaderのSTATUSフィールドにDRIVER_OKビット、途中で失敗したらFAILEDビットをセットします。

キューのアロケート処理

「Virtio PCIデバイスの初期化処理」のStep 6で言及したキューのアロケート処理は、次のような手順をキューごとに実施する必要があります。たとえばvirtio-netの場合は3つのキューが必要なので、3回繰り返します。

Step 1

設定を行うキューの番号をVirtio HeaderのQUEUE_SELフィールドに書き込みます。

Step 2

Virtio HeaderのQUEUE_NUMフィールドを読み込みます。この値がこれから設定を行うキューのキュー長になります。値が0だった場合、ホスト側はこの番号のキューを使うことを認めていないため使用できません。

Step 3

キューに使うメモリ領域をアロケートします。アロケートするサイズはキュー長に合わせたサイズで、先頭アドレスはページサイズにアラインされている必要があります。

Step 4

メモリ領域の先頭アドレスの物理ページ番号をVirtio HeaderのQUEUE_PFNにセットします。

Step 5

MSI割り込みが有効な場合、Virtio HeaderのMSI_CONFIG_VECTORフィールドまたはMSI_QUEUE_VECTORフィールドに割り込みベクタ番号を書き込みます。どちらのフィールドに書き込むかはキューがコントロール用キューか否かによって異なります。

▼表5　device status

bit	name	description
1	ACKNOWLEDGE	ゲストOSはデバイスを発見
2	DRIVER	ゲストOSはデバイス向けのドライバを保有
3	DRIVER_OK	ゲストOSはドライバ初期化を完了
7	FAILED	初期化失敗

まとめ

virtioの概要とVirtio PCIの実装について解説しました。次回はいよいよVirtqueueとこれを用いたNIC(virtio-net)の実現方法について見ていきます。SD

BOOK no.1 自宅ではじめるモノづくり超入門
3D プリンタと Autodesk 123D Design による、新しい自宅製造業のはじめ方

水野 操 【著】
B5判、 288ページ／価格＝2,940円 （税込） ／発行＝ソフトバンク クリエイティブ
ISBN＝978-4-7973-7354-7

今号の特集でも取り上げた、最近手が届く感が増してきた3Dプリンタだが、個人で購入するのにはまだ躊躇している方も多いのではないだろうか。本書は実際に一歩踏み出して何かを作ってみたい人用の道しるべである。3Dプリンタの基礎はもちろん、3Dデータを作るためのモデリングに紙幅を割いている。Autodesk 123D Designを使った3D CADを用い、特徴的な造形を作る解説も念入りだ。何でも作れそうだが、実際はこういった手順を踏まなくてはならないという苦労も実感できるだろう。3Dプリンタを購入せずに出力サービスを利用する方法についても解説されている。自宅製造業を目指す人だけでなく入門的に試したい人にも役立つ1冊だ。

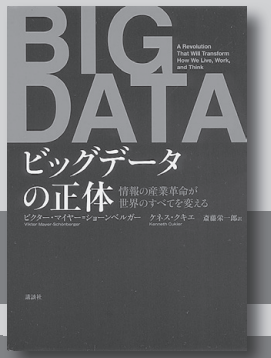
BOOK no.2 入門 Androidアプリケーションテスト

瀬戸 直喜、 株式会社ブリリアントサービス 【著】
B5変形判、 336ページ／価格＝3,360円 （税込） ／発行＝オライリー・ジャパン
ISBN＝978-4-87311-578-8

JUnitに関する基本的な知識は求められるが、テスト工程の負担を軽減する自動化についての基本がまとめられた1冊。1章のテストの基礎知識は、テスターの心得として読んでおきたい。入門書らしく開発ツールのインストールから設定までがきっちりと解説されており、とくに正確なテストに不可欠なエミュレータの詳解が役立つ。テストコードについてはAndroid Testing Frameworkを使ったUI／ライフサイクル／画面遷移のテスト手法が、サンプルを使ってていねいに解説されている。さらに、最終章でのJenkinsへの統合は、より広範に渡る自動化による継続的インテグレーションの導入としても参考になるだろう。

BOOK no.3 ビッグデータの正体
情報の産業革命が世界のすべてを変える

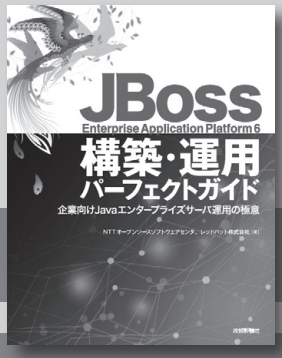
ビクター・マイヤー＝ショーンベルガー、 ケネス・クキエ 【著】 ／斎藤 栄一郎 【訳】
四六判、 319ページ／価格＝1,890円 （税込） ／発行＝講談社
ISBN＝978-4-06-218061-0



本書の第一印象は「ビッグデータの事例集」。海外の多くの企業、研究機関、政府などのビッグデータ活用／失敗事例が満載だ。とくに自動翻訳の事例がおもしろい。「人間が翻訳ルール（アルゴリズム）を考えるよりも、膨大な翻訳データを基にコンピュータに学習させたほうが高精度で翻訳できる」という結果が出ている。ここから「もはやアルゴリズムを考える必要はないのでは」という新たな仮説が生まれる。本書ではその考察も行っている。結論は読んで確かめてほしい。本書の事例からビッグデータが想像以上に高度な使われ方をしていることがわかる。自分たちから収集されたデータがどう使われているのか？　そんな視点で読んでもおもしろい。

BOOK no.4 JBoss Enterprise Application Platform 6 構築・運用パーフェクトガイド

NTTオープンソースソフトウェアセンタ、 レッドハット株式会社 【著】
B5変形版、 448ページ／価格＝3,990円 （税込） ／発行＝技術評論社
ISBN＝978-4-7741-5794-8



Javaをベースにここ四半世紀のITビジネスは発展してきた。当初はWebブラウザで動くオモチャのようなJavaアプレットから、銀行などの基幹業務での利用へ大きく進歩したJava。本書はオープンソースソフトウェアでありながら、企業で使われる重要なミドルウェアに成長したJBossの構築・運用をまとめた解説書だ。まさにJBossはオープンソースの成功モデルと言えよう。それは企業がソースコードを公開し、いろいろな個人や企業がソースコードをコミットし、さらにみんなで自社の製品にフィードバックするというエコシステムだ。この生態系を持ったOSSは強い。本書は長年に渡りJBossの開発・運用、システム構築に関わってきたNTT OSSセンタと開発元のレッドハットによる共著である。極めて実践的な内容であることはもちろんのことだ。

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20回 CGIスクリプトを作る(2)——GETで文字列を取得する

はじめに

毎度お馴染み流浪の連載、開眼シェルスクリプトですが、前回[注1]からCGIスクリプトをbashで記述するというお題を扱っています。この禁断の技は、かつては特別珍しいものではなかったようです。UNIXの古い方[注2]に話をすると、「ああ、懐かしい」という言葉が返ってきます。

しかし、懐かしいと言われて喜んでもいられません。別に古いことを懐古するためにこの連載があるのではありません。bashでCGIスクリプトを書く技を身につけると、端末の操作の延長線上でCGIスクリプトを書けるのですから、実はなかなか便利です。この伝統芸能が消えないように、コソコソとここに方法を書いておくことにします。

今回は、GETを使ってブラウザからCGIスクリプトに文字を送り込み、CGIスクリプト側でそれに応じて動的に表示を変えるというお題を扱います。

ゲッツ！——ダンディー坂野

注1）本誌2013年7月号。
注2）言い方がよくないか。

環境

前回に引き続き、筆者の手元のMacでApacheを起動し、そこでCGIスクリプトを動かします。MacでのApacheの設定方法は前回を参照してください。Linux、BSDなどの場合はWebに説明をゆずります。

前回紹介したように、筆者のMacではOS Xに標準でついてくるApacheを起動してあります。/Library/WebServer/CGIExecutables/にCGIスクリプトを置いてブラウザからhttp://localhost/cgi-bin/hoge.cgiなどとURLを指定すると、CGIスクリプトを起動できます。前回も設定しましたが、/Library/WebServer/CGI-Executables/にいちいち移動するのは面倒ですので、**図1**のように筆者のアカウントのホーム下にcgi-binという名前でシンボリックリンクをはり、そこにスクリプトを置くようにしました。

今回もバージョンが気になるような特別なことはしませんが、念のためMacのソフトウェア環境を**図2**に示します。

GETの使用方法

まず最初に、一番簡単なGETから説明します。GET(GETメソッド)というのは、HTTP

▼図1　ホームから簡単にアクセスできるようにする(再掲)

```
$ ln -s /Library/WebServer/CGI-Executables/ ./cgi-bin
$ cd cgi-bin
$ sudo chown ueda:wheel ./
```

でCGIスクリプトに文字列を渡すための方法の1つです。ブラウザなどでURLを指定するときに、後ろに文字列をくっつけてCGIスクリプトにその文字列を送り込む方法です。

リスト1にシェルスクリプトで実例を示します。これを/cgi-bin/に置いて、**chmod +x**で任意のユーザに実行権限を付与したあと、ブラウザから**図3**のように実行します。

これを解説すると、まずブラウザに打った文字列http://localhost/cgi-bin/echo.cgi?gets!!ですが、これは「echo.cgiにgets!!という文字列をGETで渡す」という意味になります。

文字列を送りつけられたCGIスクリプトのほうは、なんらかの方法でその文字列を受け取らなければなりません。が、案外簡単で、QUERY_STRINGという変数に入っているのでそれを使うだけです。ですから、**リスト1**のようにHTTPヘッダを付けて、ただechoするだけで、ブラウザに向けてGETで受け取った文字列を出力できます。

インジェクションに注意

変数QUERY_STRINGを使うときは、よほど特殊な事情がない限り、5行目のようにダブルクォートで囲みます。囲まないと、**図4**のようになってしまいます。これは端末上でのルールと同じです。**図5**のように端末で実験すると理解できるはずです。

シェルスクリプトでCGIスクリプトを書くときは、良くも悪くもシステムと密着していることを忘れてはいけません。ただ、コマンドをインジェクションされることに、あまりビビってもいけません。たとえ、$QUERY_STRINGのクォートがなくても、echoの後ろの変数はただ文字列に変換されるだけで、実行はされません(**図6**)。

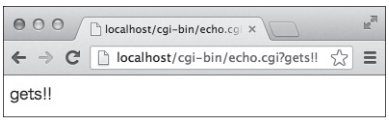
▼リスト1　GETで文字列を受け取り表示するCGIスクリプト(echo.cgi)

```
01 #!/bin/bash -xv
02
03 echo "Content-type: text/html"
04 echo
05 echo "$QUERY_STRING"
```

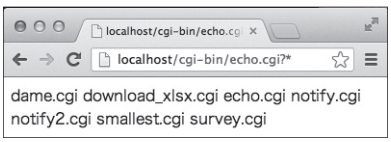
▼図2　環境

```
$ bash --version
GNU bash, version 3.2.48(1)-release (x86_64-apple-darwin12)
Copyright (C) 2007 Free Software Foundation, Inc.
$ uname -a
Darwin uedamac.local 12.3.0 Darwin Kernel Version 12.3.0: Sun Jan  6 22:37:10 PST 2013;
root:xnu-2050.22.13~1/RELEASE_X86_64 x86_64
$ apachectl -v
Server version: Apache/2.2.22 (Unix)
Server built:   Dec  9 2012 18:57:18
```

▼図3　GETで送った文字列を表示



▼図4　QUERY_STRINGのダブルクォートを除いて*を送り込む



▼図6　セミコロンの後ろにコマンドをインジェクション

▼図5　端末上でのクォート有無の実験

```
「*」を変数Aにセット
$ A="*"
クォートしない
$ echo $A
dame.cgi download_xlsx.cgi （略）
クォート
$ echo "$A"
*
```

逆にまずいパターンを**図7**に挙げておきます。まずいというより、問題外ですが……。この例のように、クォートしたからと言って絶対に安全というわけではありません。

ほかにもいろいろまずい書き方はありますが、今回の内容では、これくらい知っておいて予防しておけば大丈夫です。もちろん閉じた環境で実験するときには何も気にする必要はありません。筆者は、セキュリティレベルはWebサイトの用途しだいで変えるべきだという立場ですので、これくらいにして次にいきます。

▼図7　GETで受けた文字列を実行してしまう例

```
その1 変数が行頭にきている
$ cat yabai1.cgi
#!/bin/bash -xv

echo "Content-type: text/html"
echo
"$QUERY_STRING"

コマンドが実行できる
$ curl http://localhost/cgi-bin/yabai1.cgi?ls
dame.cgi
download_xlsx.cgi
echo.cgi
（以下略）

その2 evalを使う
$ cat yabai2.cgi
#!/bin/bash -xv

echo "Content-type: text/html"
echo
eval "$QUERY_STRING"
コマンドが実行できる
$ curl http://localhost/cgi-bin/yabai2.cgi?ls
dame.cgi
download_xlsx.cgi
echo.cgi
（以下略）
```

コマンドを選んで実行結果を表示

では、ここからはGETを使って作り物をしてみましょう。ここで作るのはサーバ管理用のWebページです。ページからコマンドを呼び出すことができるCGIスクリプトを作ります。以前（本連載の第4回、第5回[注3]）も、HTMLを出力するシェルスクリプトを作ったことはありました。しかし、今回はHTMLを作り置きするのではなく、CGIシェルスクリプトに動的にHTMLを生成させる点が違います。

まず、エディタで**リスト2**のhtmlファイルを作ります。これをCGIスクリプトで読み込み、sedなどで加工することで動的にHTMLを出力します。

次に**リスト3**のCGIスクリプトを用意し、このhtmlファイルを表示します。デバッグ用に、後ろのほうで`echo "$QUERY_STRING"`してお

注3） 本誌2012年4月号、5月号。

▼リスト2　com.html

```
01 <!DOCTYPE html>
02 <html>
03     <head>
04         <meta charset="UTF-8" />
05         <title>オレオレマシーン情報</title>
06     </head>
07     <body>
08         <form name="FORM" method="GET" action="./com.cgi">
09             コマンド：
10             <select name="COM">
11                 <option value="0">cat /etc/hosts</option>
12                 <option value="1">top -l 1</option>
13             </select>
14             <input type="submit" value="ポチ" />
15         </form>
16         <pre>
17 <!--RESULT-->
18         </pre>
19     </body>
20 </html>
```

きます。htmlが終わったあとの出力になるので邪道ですが、ブラウザには表示されます。

これでブラウザからcom.cgiを呼び出し、セレクトボックスから項目を選び、ボタンを押してみてください。**図8**のように左下にGETで送った文字列が表示されるはずです。

リストをHTMLにはめ込む

さて、**図8**の「COM=1」ですが、COMというのはセレクトボックスについた名前(com.htmlの**name="COM"**の部分)、「=」より右側は選んだ項目のvalueの値です。

GETの文字列を受け取るCGIスクリプト側では、valueの値からブラウザでどの項目が選ばれたかわかるので、セレクトボックスに書かれたコマンドをそのまま実行すればよいということになります。番号とコマンドの対応表のファイルをどこかに置いておけばよいでしょう。また、今のところ、com.htmlに直接コマンドを書いていますが、対応表のファイルの内容を動的に反映させたほうが良いでしょう。

このとき、open usp Tukubai(https://uec.usp-lab.com)のmojihameというコマンドを使います。まずcom.htmlを**リスト4**のように書き換えます。

▼リスト3　セレクトボックスで選んだ項目を表示するCGIスクリプト(com.cgi)

```
01 #!/bin/bash -xv
02
03 htmlfile=/Users/ueda/cgi-bin/com.html
04
05 ###表示
06 echo "Content-type: text/html"
07 echo
08 cat $htmlfile
09
10 #デバッグ用
11 echo "$QUERY_STRING"
```

▼リスト4　mojihameに対応したcom.html

```
01 <select name="COM">
02 <!--COMLIST-->
03         <option value="%1">%2</option>
04 <!--COMLIST-->
05 </select>
```

次に、com.cgiを**リスト5**のように書き換えます。これでブラウザには$tmp-listに書かれたコマンドが番号(行番号)を付けられてセレクトボックスにセットされます。コマンドのリストは外部のファイルでもよいのですが、説明のためにヒアドキュメントで作っています。

先に**リスト5**について、本題と関係ない細かい部分を説明しておくと、1行目の**-xv**はシェルスクリプトの実行ログの出力を行うためのオプションです。3行目の**exec 2>**は、このスクリプトのエラー出力をファイルにリダイレクトするためのコマンドです。10行目から14行目のヒアドキュメントは、**FIN**と**FIN**の間に書い

▼図8　フォームで送信される文字列

▼リスト5　コマンドのリストをcom.htmlにはめ込むようにしたcom.cgi

```
01 #!/bin/bash -xv
02
03 exec 2> /tmp/log
04
05 PATH=/usr/local/bin:$PATH
06
07 htmlfile=/Users/ueda/cgi-bin/com.html
08 tmp=/tmp/$$
09
10 cat << FIN > $tmp-list
11 cat /etc/hosts
12 top -l 1
13 echo test_test _
14 FIN
15
16 ###表示
17 echo "Content-type: text/html"
18 echo
19 sed 's/_/¥¥_/g' $tmp-list       |
20 tr ' ' '_'                      |
21 awk '{print NR,$1}'             |
22 mojihame -lCOMLIST $htmlfile -
23
24 #デバッグ用
25 echo "$QUERY_STRING"
26
27 rm -f $tmp-*
28 exit 0
```

たものを標準出力に出力するという動きをします。**FIN**は、始めと終わりで対になっていれば、別に**EOF**とか**HOGE**とかでも動きます。

mojihameの部分だけ抜き出すと、まず、21行目のawkのあとのパイプには**リスト6**のようなデータが流れます。行番号がついて、スペースは_、_は¥_にエスケープされています。open usp Tukubaiのコマンドは空白区切りのデータを受け付けるので、それに合わせてデータを変換してやらなくてはいけません。

これで2列のデータ(1列目がセレクトボックスのvalue、2列目がセレクトボックスの各コマンド)になります。これをmojihameに入力すると、COMLISTで挟まれた部分がレコードの数だけ複製され、1列目が**リスト4**の**%1**、2列目が**リスト4**の**%2**、にはめ込まれます。エスケープされた文字はもとに戻ります。mojihameが出力するHTMLのうち、セレクトボックスの部分を**リスト7**に示します。

mojihameは慣れると便利です。が、頑張る人はawkでもHTMLの部品は作れます。

再度、インジェクションに注意

ここでもう1回注意があります。com.cgiはセレクトボックスから数字を受け取りますが、数字だけしか受け取れないわけではありません。**図9**のようにcurlなどを使っても、ブラウザでURLの後ろを細工しても邪悪な文字列を送ることができます。

この対策もなかなか面倒なのですが、今回の例だと単に数字だけを受け付ければよいので、GETされた文字を受け取ったらすぐにtrで数字以外の文字を削除します。**tmp=/tmp/$$**の行の下あたりに、

```
NUM=$(echo "$QUERY_STRING" | tr -dc '0-9')
```

と付け足します。trのオプション**-d**は文字を消すという意味ですが、**-c**を付けると意味が反転します。ですので、**図10**のような挙動を示します(UTF-8なら日本語が混ざっても問題ありません)。このように12だけ残ります。改行すら消えます(プロンプトの$が行末にきているのはそのためです)。これで行番号が変数NUMに入るので、あとはリストのコマンドを実行するだけです。

完成

完成したcom.cgiを**リスト8**に示します。変数に文字列が入っていなかったり、中間ファイルができなかったりというところでバグが出るの

▼図9　com.cgiに直接GETでデータを渡す

```
$ curl "http://localhost/cgi-bin/com.cgi?reboot"
 (略)
</html>
reboot
```

▼リスト6　エスケープ後のコマンドのリスト

```
1 cat_/etc/hosts
2 top_-l_1
3 echo_test¥_test_¥_
```

▼図10　trで指定の文字以外を削除

```
$ echo 'COM=1aewagああ2' | tr -dc '0-9'
12$
  ↑ 1と2だけ残る
```

▼リスト7　com.cgiが出力するHTMLの一部(セレクトボックス部分)

```
<select name="COM">
        <option value="1">cat /etc/hosts</option>
        <option value="2">top -l 1</option>
        <option value="3">echo test_test _</option>
</select>
```

で、多少慣れが必要です。たとえば、COMにコマンドが入らないと23行目でエラーが出るので、21行目でCOMに:(なにもしないコマンド)を入れるなど、細かい芸が必要です。しかし、行数は短くなりますので、慣れるとさっさと何か試作したり、USP友の会のサイト(http://www.usptomo.com)のように見栄えのよいものも早く作れるようになります。

完成品では、もう1つfilehameというコマンドを使いました。これは、あるファイルの間に別のファイルの中身を差し込むコマンドで、図11のように使います。これもがんばってsedを使えば同様の処理はできます。

最後に実行結果を図12に示します。どのコマンドを選んでボタンを押しても、セレクトボックスの選択結果がデフォルト値の「cat /etc/hosts」に戻ってしまいますが、これもコマンドで対応できます。open usp Tukubaiのformhameというコマンドを使いますが、その説明はチャンスがあれば次回以降ということで。

終わりに

今回はGETを使ってCGIスクリプトに字を送り込む方法を説明し、ブラウザからコマンドを実行するアプリケーションを作りました。com.htmlとcom.cgiを合わせても60行程度ですので、いつも端末を叩いたりシェルスクリプトを書いたりしている人が覚えておくと、とくに何か試作するときに威力を発揮することでしょう。SD

▼リスト8　コマンドを呼び出すCGIスクリプト(com.cgi完成品)

```
01 #!/bin/bash -xv
02 exec 2> /tmp/log
03
04 PATH=/usr/local/bin:$PATH
05 htmlfile=/Users/ueda/cgi-bin/com.html
06 tmp=/tmp/$$
07
08 ######実行可能コマンドリスト######
09 cat << FIN > $tmp-list
10 cat /etc/hosts
11 top -l 1
12 echo test_test _
13 FIN
14
15 ######コマンドの実行######
16 #番号受け取り
17 NUM=$(echo "$QUERY_STRING" | tr -dc '0-9')
18 #指定された行を取得
19 COM=$(awk -v n="$NUM" 'NR==n' $tmp-list)
20 #COMが空なら : を入れておく
21 [ -z "$COM" ] && COM=":"
22 #実行
23 $COM > $tmp-result
24
25 ######HTML出力######
26 echo "Content-type: text/html"
27 echo
28 #エスケープ処理
29 sed 's/_/¥¥_/g' $tmp-list       |
30 tr ' ' '_'                      |
31 #行番号をつける
32 awk '{print NR,$1}'             |
33 #出力 >>> 1:行番号 2:コマンド
34 mojihame -lCOMLIST $htmlfile -  |
35 #コマンド実行結果をはめ込み
36 filehame -lRESULT - $tmp-result
37
38 rm -f $tmp-*
39 exit 0
```

▼図11　filehameの使い方

```
$ cat file1
===参加者===
ATT
===以上===
$ cat meibo
山田
里中
殿間
$ filehame -lATT file1 meibo
===参加者===
山田
里中
殿間
===以上===
```

▼図12　実行結果

短期集中連載

SDNは使えるのか、使うべきなのか？

仮想ネットワークの落とし穴

第3回

OpenFlowの検証

株式会社データホテル
伊勢幸一（いせこういち）
Twitter @ ibucho

今回が本連載の最終回ですが、最後にOpenFlowについて検証してみましょう。前回述べたように、仮想ネットワークの目的が物理ネットワークに依存しないオーバーレイセグメントの形成とIEEE802.1QにおけるVLAN数上限の拡大であるすると、OpenFlowというスイッチングプロトコルだけで仮想ネットワークは実現できません。したがって厳密に言うと本連載が対象とする仮想ネットワークではありませんが、フロー（flow）によってネットワークをドライブするというアーキテクチャが、仮想ネットワークの形成と非常に近い状態を実現するため、OpenFlowも議論の対象としてみることにしました。

また、いまだにOpenFlowにはフワっとした理解しかなく、いったいどのような問題や環境に有効なのかを明らかにできないまま導入を検討している人も多いと思います。ベンダが連呼するポジティブイメージだけが先行して正体がわからなくなってしまった革新的な技術の本質を理解するには、その利点や優位性がアピールされている部分ではなく、欠点や欠陥、問題点や課題を明らかにするほうが効果的です。その技術や対象はいったいどのような問題に向かないのかを知ることで、逆に効果的に活用できるポイントが浮き彫りになってきます。対象技術の優位性や先進性を刷り込まれ、その技術を使うことを前提として用途、用法を検討すると歪んだ設計や運用を生み出しやすく、無駄な労力と投資を浪費するだけの結果に陥る可能性が高くなります[注1]。

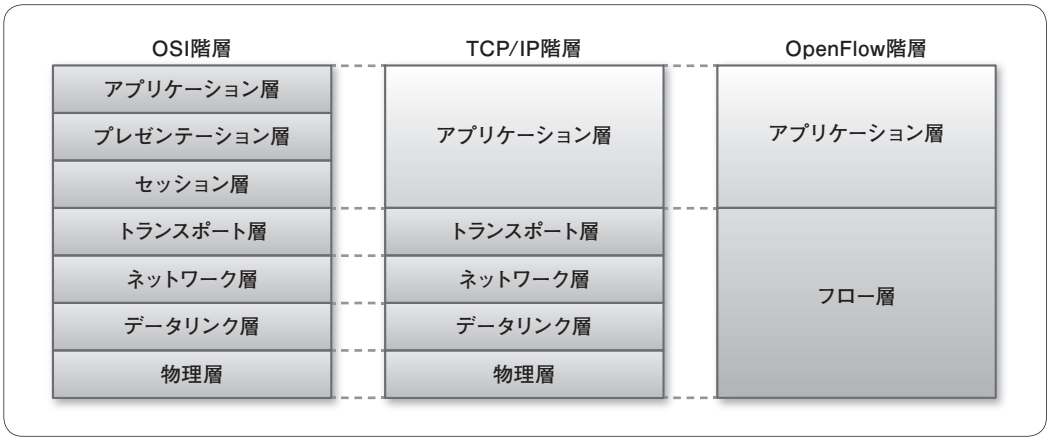
OpenFlowプロトコルスタック

OpenFlowについての一般的な技術的内容は、すでにさまざまな記事や書籍で紹介されているので本稿では割愛します。今回はOpenFlowプロトコルとスイッチング、そしてネットワークのしくみについて基本的な技術や規格、内容を理解しているという前提で話を進めます。

OpenFlowはまったく新しく革新的なプロトコルであるという評価のされ方をしていますが、それは当然のことです。従来、代表的なオープンプロトコルスタックとしては、OSIプロトコルとTCP/IPプロトコルという2種類のスタックがあります。そして、これらとは別にOpenFlowプロトコルスタックという階層モデルがあるととらえることができます。ただし、まったく新しいプロトコル階層モデルというわけでもなく、OpenFlow階層はOSI階層の下位4層を1つのフロー層として実装しています。したがってOpenFlowプロトコルスタックにおいてフォワーディングや更新、廃棄などのアクションを

注1）経験者は語る。

▼図1　オープンプロトコルスタック



決定するフロー識別子がIPアドレスやVLAN-IDのような単一の識別子ではなく、TCP/IPスタックにおける下位4層の組み合わせであるということです。それも下位4層すべてからなる組み合わせではなく、下位4層の中の任意の識別子の組み合わせであることがOpenFlowによる自由度の高いネットワーク・トポロジーの形成を可能としていますが、逆にネットワーク設計に混乱を招く要因にもなっています(**図1**)。

OpenFlowに対する誤解とは

OpenFlowはOSIやTCP/IPに対してレイヤーバイオレーションをしているという批判も聞きますが、それは筋違いです。もともとOSIやTCP/IPとは異なるスタックですので、ほかのスタックの階層モデルと定義や処理が異なるのは当然であり、もしこれがレイヤーバイオレーションであるというのであれば、TCP/IPのアプリケーション層もOSIの上位3層をバイオレーションしていることになります。

このフローによるフォワーディングを行うことで、見かけ上、既存の物理ネットワーク上にフローネットワークをオーバーレイしているかのように利用できます。また、このフロー識別子の要素にブロードキャストアドレスを適用すると仮想セグメントを形成でき、また運用負担を考慮しないのであれば、VLAN-IDと仮想マシンのMACアドレスやIPアドレスとの組み合わせによって仮想的に4094以上のテナントセグメントを作成することもできます。

このようにOpenFlowをTCP/IPの拡張プロトコル、もしくは革新的進化系というとらえ方をする限り、その本質を理解することはできません。従来とはまったく異なる新しいプロトコルスタックであり、そのスタックを用いたネットワークの設計や運用はまったく別物であるという認識が必要です。

エッジエンドポイントとホップバイホップ

OpenFlowにはエッジエンドポイントとホップバイホップというデプロイメント方式がありますが、これも混乱を招く要因となっています。単純にとらえるとエッジエンドポイント方式は物理サーバ内のネットワークをフローでドライブする方式であり、ホップバイホップは物理サーバ間のネットワークをフローでドライブする方式です。

しかし、すでに数百台から数千台の物理サーバによって構成されるクラウド基盤やデータセンターにとって物理サーバ内に限定してネットワークをドライブするエッジエンドポイントはほとんど必要性がありません。しかしサーバ間ネットワークに従来型の物理ネットワークを使っ

仮想ネットワークの落とし穴

ている限り、サーバ間の通信をフロードライブすることはできないため、OpenFlowエッジエンドポイントはサーバ内フローネットワークと既存のTCP/IPネットワークのゲートウェイでしかなく、ネットワーク全体をフロー化することは不可能です。

エッジポイント方式

そこでエッジエンドポイント方式ではサーバ間の通信をトンネル化することで既存の物理ネットワーク上にフローネットワークをオーバーレイすることで対応しています。すると、自動的に前回で述べたようなエンドポイントモデルによる仮想化の欠点や欠陥をそのまま引き継ぐこととなり、高速な通信よりも柔軟なセグメント形成が優先されるネットワークに適したプロトコルであると言えます(**図2**)。

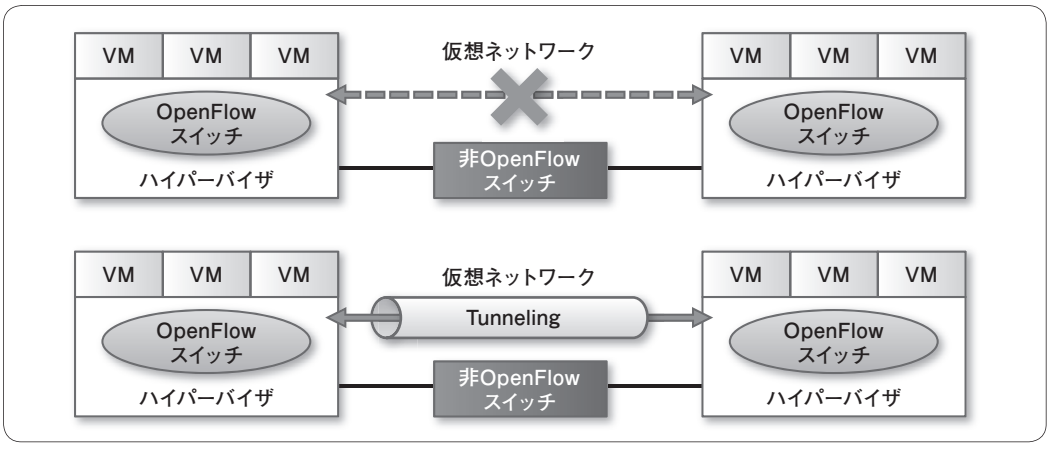
ホップバイホップ方式

ホップバイホップ方式はドメイン内にあるすべての物理スイッチがフロードライブできることを前提としたネットワークです。サーバ間の途中経路がすべてフローによってドライブされるので、経路をトンネル化する必要がなく、ネットワークオーバーレイのパフォーマンスオーバーヘッドはありません。ドメイン内すべてのスイッチをOpenFlow対応にするには設備投資という経済的問題がありますが、OpenFlowによるフロードライブ方式がコモディティ化することによっていずれ解決されるでしょう。ホップバイホップ方式の場合、サーバ間の通信経路を完全にフロードライブできるため、従来の階層別フォワーディングポリシーをいっさい考慮する必要はなく、フローだけによる通信チャンネルトポロジーが形成できます。これはフローによるネットワークトポロジーの形成であり、MPLSなどの従来型オーバーレイや、前回議論した仮想ネットワークとはまったく別の新しいネットワークアーキテクチュアです。しかし、実際にはフローがTCP/IPプロトコルの下位4層から構成されている事実により、ある意味既存ネットワークの仮想化、もしくはオーバーレイと言えなくもありません。

実装の違いと誤解

このようにエッジエンドポイント方式とホップバイホップ方式は実装や機能がまったく異なるため、同じアーキテクチュアとして考えないほうが良いでしょう。つまり、OpenFlowには2つのデプロイメントモデルがあるというのが混乱を招く原因であり、エッジエンドポイントとホップバイホップという2つのネットワークアーキテクチュアがあり、それぞれが同じOpenFlowというプロトコルスタックを採用し、利用して

▼図2　エッジエンドポイントモデル



いると理解するほうが素直です。ただし、それぞれOpenFlowという同じプロトコルを使用していることから、両者を相互接続し、サーバ内VM間とサーバ間の通信をすべてフローでドライブするネットワークを構成できます。これは、ネットワークを構成するサーバ、スイッチ、ルータがすべてTCP/IPをサポートしているならば、それぞれ相互接続して1つのネットワークを構成することと同じです。

それではOpenFlowの致命的欠陥とは何でしょう。それにはいくつか考えられます。

コントロールプレーンの分離の問題

ネットワークが不安定になる恐れ

まず1つはデータプレーンとコントロールプレーンが分離されていることです。OpenFlowスイッチの挙動を指示するコントローラインスタンスはスイッチインスタンスとは別に存在します。データプレーンとコントロールプレーンを分離する目的は複数のデータプレーンを1つのコントローラで集中管理することであり、理想的には1つのコントローラがドメイン内すべてのOpenFlowスイッチを制御することでしょう。

しかし、逆に言うとコントローラとスイッチの通信チャンネルに問題があるとネットワークの制御が不安定になります。スイッチのフローテーブル更新方式には2種類あり、起動時にコントローラ側にあるフロー制御プログラムをすべてダウンロードしてフローテーブルを作成するプロアクティブ方式と、スイッチが受け取ったパケットに対するフローエントリが自身のフローテーブルにない場合、随時コントローラに処理を問い合わせるリアクティブ方式があります。リアクティブ方式はスイッチがパケットを受け取るたびにフローテーブルにフローが登録されていくことになりますが、1つのコントローラで一極集中管理されている場合、そのドメインを管理するコントローラに対してすべてのスイッチが問い合わせを行うと、コントローラの負荷が高くなってしまいます。

コントローラの負荷

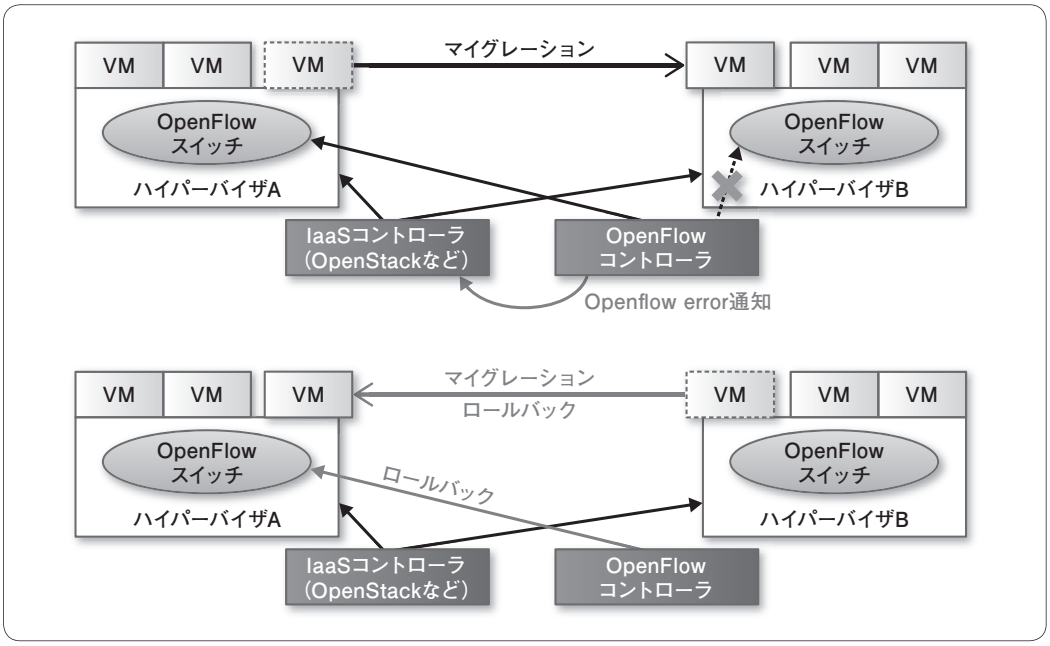
プロアクティブの場合であっても、すべてのフローについてあらかじめ処理内容を記述することは現実的ではなく、フローテーブルにないパケットを受け取った場合、そのパケットへの処理方法をコントローラに問い合わせることになります。完全なリアクティブ方式に比べるとコントローラへの問い合わせ件数は激減しますが、事前に指示するフロー制御のエントリが少ないほど、コントローラへ問い合わせる件数が増えることになります。したがって、実際の運用では制御するフローエントリが少ない場合、それ以外のパケットに対してはデフォルトアクションとして従来どおりのプロトコル階層別フォワーディング処理を行うことを指示することになるか、ワイルドカードを使って条件にある範囲を指定し、なるべくそのワイルドカード条件を充実させておくことになります。すると、コントローラ側でフローを追加したり、あるフローに対する処理を変更した場合にのみ、スイッチとコントローラ間の通信が発生するだけで済むため、コントローラの負荷は非常に低い状態で運用を行うことができます。

フローの処理のしくみ

ここで、問題なのは1つのコントローラですべてのスイッチを制御している場合、フローの変更に対して矛盾なくパケットをフローに基づいてスイッチがバケツリレーできるように、通信経路にあるすべてのスイッチにそのフローに関する処理を一斉指示する必要があるということです。この際、経路途中のどれかのスイッチとコントローラとの通信が遮断されていると、そのフローによるパケット転送に矛盾が生じます。OpenFlowでは、関連するスイッチのすべてと通信が可能であるかどうかを死活監視によって定期的に確認しています。しかし、(実装しだ

仮想ネットワークの落とし穴

▼図3　マイグレーションロールバック



いですが)死活監視はフロー更新前に必ず行われるという保証はなく、死活監視間隔の途中でフローの更新が行われ、フローの更新が正常に行われなかったスイッチが存在したならば、フローの更新をいったんロールバックしなければなりません。

この段階でフロー更新をロールバックすることに大きな問題は生じません。なぜなら、フローに関する処理はスイッチとコントローラとの間で閉じているからです。しかし、IaaSのようなクラウド基盤に対してOpenFlowネットワークを適用している場合、フローの変更は仮想マシンインスタンスの起動、停止、移動をトリガとして実行されるはずです。したがって、ネットワーク側でのフロー変更が失敗し、OpenFlow側でのロールバック処理は、最終的にIaaSコントローラ側まで通知され、IaaS側においても仮想マシンインスタンス処理をロールバックする必要があります。IaaS基盤も含め、ネットワークトポロジーの形成を矛盾なく整合性を維持しつつ運用するというたいへん負担の大きいネットワークになる可能性があります(**図3**)。

フロードライブネットワークの問題

ネットワークプロトコルの見地から

OpenFlowにおけるもう1つの致命的欠陥はプロトコル階層別にフォワーディングを行うのではなく、フローによって行うという部分です。最初にOpenFlowはTCP/IPとは別のプロトコルスタックであり、TCP/IPの下位4層を1つのフロー層として定義付けられていると述べました。フロー層は1つの独立したプロトコル階層ですが、その実態はTCP/IPの4層からなる階層です。つまり、スイッチのポート、フレームのMACアドレスとVLAN-ID、IPアドレスもしくはMPLSタグ、そしてTCP/UDPポートという複数の識別子による組み合わせでフォワーディングを判断しますが、これにも問題があります。

通常ネットワークのパフォーマンスはスイッチやルータ、ロードバランサ、そしてサーバなど各階層単位での個別チューニングによってトー

タルなパフォーマンスが次第に向上していきます。もちろん、常に全階層が同じペースでチューニングされるわけではなく、ある時点ではサーバCPUとメモリによる処理能力がNICのワイヤースピードを超えるデータのI/Oを引き出すこともあります。また、ある時点ではサーバCPUやNICでは出しきれないほどの広帯域なポートがスイッチ側でサポートされたりすることもあるでしょう。

OpenFlowでは階層別ではなく、フロー層全体でのチューニングが必要になります。TCP/IPでは各層単位でのチューニングが可能であるため、各層は独立して性能向上させることができ、各層間のインターフェースを維持しておけば、同時並行的に開発ができます。しかし、フロー層は4階層のハンドリングを1つのインスタンスで実装しなければなりません。そのため、NIC、スイッチ、ルータ、サーバOSなど階層別に別々のベンダによる製品開発、評価検証などができません。フロー層の開発やチューニングは単一の開発団体がすべて行うため、階層別での開発よりも不利になる可能性があります。

現段階ではネットワークプロトコルにおいて、垂直分割型と水平分割型のどちらが開発工程や性能アップにおいて有利であるかという結論は出ていませんが、今言えることはそれぞれ異なる開発工程であるということです。

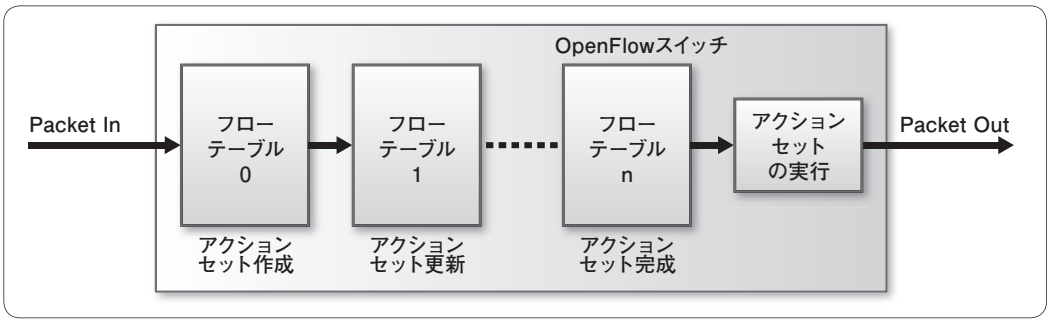
運用面での過負荷の恐れ

もう1つは運用面での欠陥です。OpenFlowはフロー層によりパケットフォワーディングが処理されますが、このフローを構成するフィールド(MACアドレス、IPアドレス、VLAN-IDなど)は非常に多様であり、任意のフィールドの組み合わせによって定義されるフローの数は膨大になります。巨大なテーブルからスイッチが受信したパケットにマッチするフローを検索する負荷は大きくなり、フォワーディングパフォーマンスに少なからず影響を与えるでしょう。OpenFlowソフトウェアスイッチであるOpen vSwitchではこのフローエントリをハッシュ化し、テーブルの検索負荷を低減させています。また、OpenFlowスイッチ仕様1.3ではフローテーブルを分割し、それらをパイプライン化することによってフローマッチング処理の性能向上とアクションセットの多様化を図っています(**図4**)。

ところが、Open vSwitchのようにフローエントリをハッシュ化するにはパケットとのマッチングを完全一致で実施する必要があります。したがって、コントローラ側のプログラムでIPアドレスや物理ポート、論理ポートが範囲で指定されていたとしても、それらをひとつひとつ分割し、個別のフローエントリをテーブルに登録する必要があるため、テーブルサイズは非常に大きくなります。前述のようにエントリ数が増えるとテーブル検索やマッチング処理の負荷が上がり、同じようにフォワーディング性能を劣化させます。

また、フローテーブルのパイプライン化にも

▼図4　フローテーブルパイプライン



仮想ネットワークの落とし穴

問題があり、フローテーブルに該当するエントリがなくとも、受け取ったパケットを一通りすべてのフローテーブルでマッチング処理を行わなければならないということです。テーブルサイズを縮小することが逆にテーブルのフローエントリとのマッチング処理を増やすという結果を招いています。

フローエントリマッチング負荷を低減するため必要最小限のパケットに対してフロー処理を施し、そのほかのパケットに対してはデフォルトフォワーディングを適用する方法も考えられますが、そうするとあらゆるパケットをフローでドライブするというOpenFlowの基本的コンセプトからどんどん遠ざかっていきます。結局のところ、ネットワークをすべてフローでドライブすると著しいネットワークの性能劣化を生じ、ネットワーク性能を維持しようとすると、極力フロードライブを利用しないほうが良いという矛盾が発生してしまいます。

プログラマブルネットワークの問題

もしものときに対応できない

OpenFlowの欠陥の1つにパケットに対する処理がプログラマブルであるということがあります。ネットワーク通信チャンネルの確立やパケットに対する処理をif-else分岐などで詳細に定義できるということ自体に不都合はありません。それ自体は歓迎すべき機能でありアーキテクチュアでしょう。問題はプログラマブルであるがゆえにネットワークを稼働させるためにはプログラミングをしなければならないということです。それもネットワーク全体をOpenFlowによってフロードライブするには、すべてのOpenFlowスイッチに対し、すべてのパケットに対するすべてのフロー処理をプログラミングしなければなりません。ネットワーク全体のフロー定義数はスイッチとパケットとの積になるため天文学的数字になる可能性があります。現実的にはすべてのスイッチ、パケットではなく、部分的なパケットに対して局所的なスイッチにのみフロー定義を与えることになると思いますが、これもフローを活用するにはプログラムによる定義作業が増加し、逆に運用負担を軽減するためプログラム定義を低減するとフロードライブネットワークから遠ざかるという矛盾を抱えています。

さらに、障害や誤動作により正常にパケットが搬送されなかった場合、原因を特定するためには各スイッチの状態やスイッチ間の状態に加え、コントローラ側にあるプログラムコードををチェックする必要があります。実際、NOXやTremaといったコントローラやフレームワーク上でプログラミングされたコードはそれほど複雑な構成をしてはいませんが、それでも、各モジュールやクラス、メソッドなどを個別にデバッグしていく作業が発生します。

原因の探求が困難

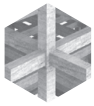
一般的にネットワークを構築する場合には、トポロジーを設計し、スイッチやルータなどの機器を価格と性能面から考慮して選定し、導入後は各スイッチをコンフィギュレーションして順次疎通確認を行い、そして運用に入ります。トラブルがあった場合には該当個所の機器の状態や隣接システムとの疎通を個々に調べていくうえで、ある特定の機器設定や1組の機器間の設定などを修正することで対処します。

しかし、OpenFlowによって構成されるネットワークはスイッチを導入してポートのIPアドレスやデフォルトのフォワーディング設定を個別に設定し、さらにそれらスイッチに対するフロー定義をプログラミングしなければなりません。トラブルに対しては個別機器をチェックすることはもちろんですが、スイッチ上にすべてのフローエントリがあるわけではなく、参照されなかったフローが定期的にテーブルから削除され、またフロー制御対象のパケットを受け取らなかった場合にはそのフローエントリがコントローラから取得されていないこともあります。

したがって、フロー全体をチェックするには、各スイッチの物理的状態と基本設定、フローテーブルのチェックに加え、コントローラ側のフロー定義、そしてプログラムコードそのものをチェックする必要があります。対処する作業が増えています。自動的に障害原因を特定するまでの手間が増えることから復旧までの時間が大幅に増える可能性もあります。

このようにパケットフォワーディングをプロトコル階層別ではなく、フロードライブによってネットワーク全体をプログラマブルに制御するというシステムは大きな潜在的欠陥を持っています。

OpenFlowの効果的利用術



それではOpenFlowはダメなパラダイムなのでしょうか？　筆者はそうは思いません。逆にここまで致命的な欠陥を浮き彫りにすることで非常に有効な利用方法が考えられます。

マイナスからプラスへ転換するには

OpenFlowの致命的欠陥をまとめると次のようになります。

1. コントロールとデータプレーンが分離されている
2. 4階層を1つのフロー層にしてフォワーディングする
3. プログラマブルである

つまり、これらの欠陥を解消し、かつOpenFlow特有の機能を活用すると良いのです。したがって

1. コントロールとデータプレーンを分離しない
2. フローによるフォワーディングをしない
3. プログラマブルにしない

この状態でOpenFlowの機能を活用します。OpenFlowの機能として特筆すべきことはフローによるフォワーディングではなく、フローによるパケットの書き換えです。つまり、ある特定のフローにマッチするパケットを受け取った場合、そのパケットの任意のヘッダフィールドを書き換えたり、削除、追加ができます。もしくは、新たにパケットを生成して、それを転送したり返信したりすることもできます。

また、「3. プログラマブルにしない」という条項はフローフォワーディングをプログラマブルにしないという意味で、プログラミングをフィールドの書き換え、もしくはパケット生成に限定する限り、大きな導入運用負担はないでしょう。

上記1～3の条件に対して、パケットのフィールドを書き換えるという処理を加えると、自動的に活用シーンが見えてきます。現在のネットワークドメインはドメイン内に統一されたポリシーが適用され運用されています。それはAS、IPアドレスブロック、VLAN、オーバーレイなどです。クラウドシステムのように複数のドメインを連結連携させて1つの仮想的なコンピューティングシステムを構成する場合、このドメインごとのポリシー不整合が問題になります。

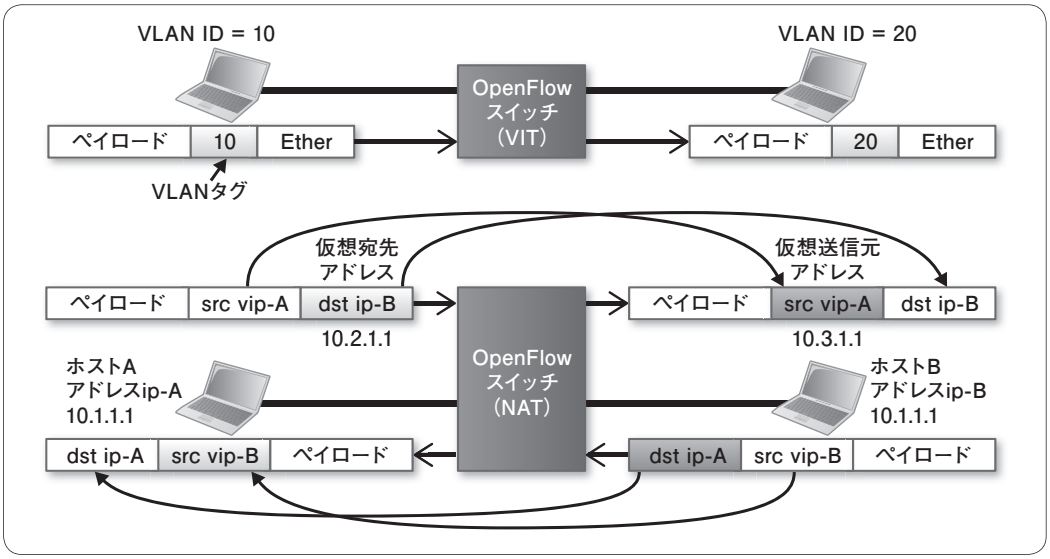
たとえば、別々のVLAN-ID割り当てポリシーを持つドメイン間をシームレスに連結するにはVIT(Vlan Id Translation)が必要になります。仮想化されていない物理ネットワークと前回解説したVXLANやNVGREで仮想化されたオーバーレイ・ネットワークとの連結には物理ネットワーク側のVLAN-IDとVXLANのVNI、NVGREのVSIDとのマッピングが必要です。同じIPプライベートアドレスを割り当てられたドメイン間で相互通信するにはそれぞれ相手側のアドレスを仮想的なIPアドレスにマッピングするNATのような機能も必要です(**図5**)。

OpenFlowを活かすには

このように複数のポリシーで構築されたネットワークドメイン間を連結するゲートウェイとしてOpenFlowスイッチは非常に有効なアーキテクチュアです。基本的にヘテロジニアスドメイン間を接続するゲートウェイはドメインあた

仮想ネットワークの落とし穴

▼図5　インタードメインゲートウェイとしての活用



り1台か数台で間に合います。したがって、データプレーンであるスイッチとコントロールプレーンであるコントローラは1対1で運用できるので、物理的に1台の機器やサーバ上に共存させることができます。すると、インスタンスとしては別々に存在しますが、運用者からみると従来型のスタンドアローン型ネットワーク機器と同じになります。結果的に、スイッチとコントローラとの接続性やコントローラ側で認識すべきスイッチの区別などを気にする必要がなくなり、プレーンが分離されているという欠陥が解消します。

また、ゲートウェイはフローをもとにパケットをマッチングして必要なヘッダ修正をするだけなので、ネットワーク全体に対してすべてのパケットにフロードライブを適用する必要はなく、部分的なドメイン間のトラフィックのみが処理対象となり、フォワーディング性能の劣化は必要最低限に抑えることができます。

VXLAN網との相性はいかに

また、パケットによって条件分岐や複雑な処理は必要なく、プログラミングはマッチするフローの定義とヘッダフィールドの更新処理だけになるため、プログラミングによる実装工数や障害に対する原因究明もそれほど困難ではありません。たとえば、非仮想ネットワークとVXLAN網を連結するには、パケットのアプリケーションデータユニットにあるVXLANヘッダのVNIを書き換える必要があります。OpenFlowスイッチはアプリケーションデータへの書き換えをサポートしていませんが、OpenFlowスイッチのデータパスにVXLANがサポートされているならば、そのVNIはVXLANのインスタンスに設定されているはずであり、OpenFlowスイッチはパケットのVLAN-IDをチェックし、そのVLAN-IDに相当するVNIへの転送を担うポートにフォワードすることで非仮想ネットワークとVXLANドメインとを連結することはできます。基本的にOpenFlowスイッチはプログラマブルなのですから。

データセンターの悩みが解消される

現在、おもにデータセンター内やクラウド基盤が抱えている大きな課題はパケットフォワーディングにおける自由度ではなく、ヘテロジニアスドメインをいかにシームレスに、そして運用ワークフローやフレームワークを大幅に変更

することなく、障害対応のリスクを上げずに連結連携することです。この課題に最も適した技術、プロトコルがOpenFlowであると言えるでしょう。

最後に

筆者がこの連載の企画をいただいたとき、OpenFlowのプロトコルやパラダイムについてこれといって使い道がないということ以外にとくに致命的な欠陥が思い浮かびませんでした。しかし、OpenFlowに関してさまざまな識者から意見を伺い、文献、論文、仕様書、ソースコードなどを調査し分析することで、OpenFlowの優れた点としてアピールされている特徴がすべて致命的欠陥であり、OpenFlowはそのアーキテクチュアに根本的な問題を抱えているということに気づきます。すると、ごく自然にOpenFlowの有効な活用方法が思い浮かびました。たまたま筆者はOCDET(オープンクラウド実証実験タスクフォース注2)というコミュニティ活動の中でヘテロジニアスなクラウド基盤間を連結するトンネルゲートウェイの実装をしていたこともあり、筆者が実装していたVITトンネリングは、すでにOpenFlowスイッチで可能であることに気づいたのです。そこからOpenFlowはLAN全体を構成する技術ではなく、ヘテロジニアスなドメイン間をシームレスに接続する技術として非常に有効であるという結論を得ました。それぞれの技術やプロトコル、実装に対し、ポジティブ、ネガティブのどちらかだけで判断するのではなく、ポジティブであるならばネガティブな部分に着目し、ネガティブであれば逆にポジティブな部分に着目して検討することで、それぞれの技術の能力をうまく活用できるでしょう。SD

注2) http://www.ocdet.org/

6

Debian Developer　やまねひでき　henrich@debian.org

Debian 7.0 デスクトップ周りの変更点

Debian Hot Topics

Debian 7.0リリース!（続き）

先月号に引き続き、Debian 7.0“Wheezy”の特徴を取り上げていきます。今回はデスクトップ周りです。

デスクトップテーマは「joy」

Squeezeのデスクトップはポップでオモチャをイメージする「spacefun」でしたが、その反動か、Wheezyのそれは一転してシックでモノクロな「joy」に変わりました。この連載の誌面デザインも合わせて変更……とするとデザイナーから悲鳴があがってきそうですし、筆者も気に入っていますのでそれは行わないでおきましょう :-)

Wheezyにアップグレードすればテーマ画像は自動的にjoyに切り替わりますが、以前のspacefunが恋しいという方も、いらっしゃるかもしれません。これらの画像を収録しているdesktop-baseパッケージには、まだspacefunテーマ画像が含まれています。以前の画像を使いたい方は、GNOME 3であればシステム設定から「背景」でウィンドウ左下の「＋」ボタンをクリックして直接ファイルを選択し、/usr/share/images/desktop-base/以下にあるspacefun-wallpaper.svg、あるいはspacefun-wallpaper-widescreen.svgを指定しましょう。

標準デスクトップはGNOME 3

デスクトップ環境はGNOME 3.4が標準で、そのほかにKDE 4.8.4、Xfce 4.8、LXDE 0.5.0が提供されます。一時期、「GNOMEではなくXfceが標準デスクトップになる」とまことしやかに吹聴されたことがありますが、実際のところは「CD1枚目にGNOMEだと収まらないからGit上でいったん変更してみた」だけにすぎません。その後、Debianのパッケージアーカイブを管理する「ftpmaster」の1人であるAnsgar Burchardtさんが、精力的にパッケージの圧縮形式をxzに変更

▼図1　Squeezeのデスクトップ（spacefun）

▼図2　Wheezyのデスクトップ（joy）

することで容量削減を行い、再びGNOMEが標準に戻りました[注1]。しかし、GNOME 3.x(のGNOME Shell)はGNOME 2.xと大幅に最初の見た目が異なるため、拒否反応を示す方も少なくないと思います。このような場合は、

①とにかく慣れる[注2] (^^;;;
②GNOME 3の「classic」モード[注3]を利用して、GNOME 2の操作感に近づける
③Xfce4などの6.0(Squeeze)のころと変わらないデスクトップ環境に変更する

などの方策があるでしょう。標準デスクトップ環境が何であれ、「使いたいものを選んで使える」のがDebianのいいところです。GNOME 3に囚われることなく、お好きなものをお選びください。

なお、仮想マシン環境で顕著ですが、利用しているグラフィックチップの3Dアクセラレーション機能が効かないため、**図3**[注4]のようにメッセージが出て強制的にclassicモードが選択されることがあります[注5]。

注1) 標準はGNOMEであるとはいえ、インストール時に他のデスクトップ環境を指定することでGNOME以外のデスクトップ環境も簡単に利用できるようになります……が、この事実はあまり知られていません。

注2) デスクトップの操作には慣れの部分が大きいので、最初のうちは意図した操作ができなくて不満が出るでしょう。しかし、慣れてくると「それなりに」使えるというのが筆者のGNOME 3に対する感想です。もっとも、GNOME側がラディカルに変更を強い過ぎの感もします……。

注3) 少し前までは「fallbackモード」と言われたもので、ログイン時にウィンドウの左下の選択肢で選べます。これを選ぶとGNOME 3のGNOME Shellの操作感から一転、過去のデスクトップに近いメニュー表示などが出現します。

▼図3　3Dアクセラレーション機能が使えない場合

日本語入力はmozcがデフォルトに

日本語入力(IM：Input Method)の標準は、これまでのAnthy(アンシー)から入力精度が高いmozc(モズク)に変更されました(Anthyも同時にインストールされています。こちらがお好みの方は選択いただければと思います[注6])。なお、IMのフレームワークは昨今はibusを採用しているディストリビューションも多いですが、Debianでは変わらず安定のuim(ユーアイエム)です。ibusはGNOMEとの統合を行ったためか、バージョンアップにもかかわらず既存の機能をドロップするなど傍(はた)から見ると迷走気味になってきているので[注7]、過去にuimを選択したのは、意図せずして正解だったかもしれません。

LibreOfficeへの移行

オフィススイートのOpenOffice.orgはLibreOffice 3.5へ置き換えられています。これはOpenOffice.orgパッケージとLibreOfficeパッケージのメンテナであるRene Engelhardさんが LibreOfficeの開発母体であるThe Document Foundationの設立当初からのプロジェクトメンバーである[注8]ことからわかるように、既定事項であると言えるでしょう。また、KDEのオフィススイートKOfficeが、upstreamでの改名に伴い「Calligra」という名前

注4) 右横が切れているのはテスト用の仮想環境でバルーンが正しい位置で出なかったためです。

注5) ちなみに、最新のGNOME 3では、グラフィックチップの3Dアクセラレーション機能が使えない場合、llvmpipeを利用したソフトウェア処理を行うことでGNOME Shellがどのような環境でも起動するようになっています……が、llvmpipeの処理は非常に重く苦痛ですので、ハードウェア／ビデオドライバでの3Dアクセラレーション機能のサポートがない環境ではclassicモードの利用を強くお勧めします。

注6) Anthyパッケージはここ2、3年ほど筆者がアップデートを実施していますが、些細な修正のみで改善などはあまりありません。

注7) ibus以外にも「Fcitx(https://fcitx-im.org/wiki/Fcitx)」というものも出てきているようです。今後、要注目かもしれません。

注8) URL https://www.documentfoundation.org/foundation/members/

に変更されています。

ブラウザ

主要ブラウザIceweasel(Firefoxの商標非使用版)はバージョン10を採用しました。ですが、かなり古く見劣りしているというのが、事実です。Debianでは、通常の6週間ごとのバージョンアップではなく、法人向けにサポート期間を約1年、マイナーアップデートを6週間ごととした「延長サポート版」[注9]のバージョン10をベースに採用したのですが、すでにupstreamであるMozilla側ではバージョン10のサポートが終わっており、現在の延長サポート版はバージョン17となっています。この辺は筆者もリリース前に問題提起したのですが、結論が出ずにリリースに至りました。

その後、セキュリティアップデートの際に17.0ESRに更新がされました[注10]。Firefoxのような巨大なソフトウェアに対して、古い10.0にbackportし続けるよりも17.0をベースにリリースを行ったほうが良いという判断がされたようです[注11]。ただ、これによりextensionのパッケージをどのようにハンドリングしていくかについては、今のところ結論がでていません。さらに新しいバージョンを使いたい場合には、Debian Mozilla teamが提供するmozilla.debian.netのリポジトリを追加して利用することが考えられます。こちらのほうは随時更新されており、さらにBeta/Aurora版も使えますので新しいもの好きの方にはお勧めです。リポジトリの追加は、サイトでapt lineを生成できますので、そちらを/etc/apt/sources.listに貼り付けて利用しましょう。

他方、Chromium(Google Chromeのオープンソース版)では現時点でバージョン27がリポジトリに入っており、upstreamに追随しています。WindowsではChromeを使っているという方はこちらをインストールしても良いでしょう。

注9) URL http://www.mozilla.jp/business/downloads/all/ESR(Extended Support Release)といいます。

注10) URL http://www.debian.org/security/2013/dsa-2699

注11) URL http://lists.debian.org/debian-devel/2013/05/msg01572.html

マルチメディアサポート

過去、Debianではmp3のエンコードに使われるlameやmplayerなどのソフトウェアはコーデック周りの特許/ライセンスの問題から、パッケージが入ることがありませんでした。しかし、昨今、方針が変わったらしくWheezyではさまざまなコーデックが利用できるようになりました。音楽CDリッピング用ソフトSound-Juicerでも標準で出力ファイル形式としてogg/mp3/mp4/FLACが選択できます(標準はoggですが、携帯音楽プレイヤーなどでサポートしていない場合はmp3などに変更するのが良いでしょう。音質にこだわる方はFLACをどうぞ)。動画についてもH.264/MPEG-4などの再生がx264パッケージによって可能となっています。これまではdeb-multimedia.orgという外部サイトのリポジトリ[注12]が、この辺を担ってきましたが、今後は、どうしても必要という場合以外はDebian公式パッケージのみで済ませられます。

7.1リリース

7.0のリリース後に見つかった問題点の修正とセキュリティ修正の集積として、6月15日にDebian7.1がポイントリリースという形でアップデートされています。Squeezeまでは6.0.1と小数点第2位のリリースだったのに何が変わったのか……というと、とくに理由はないようです。以前の記事(2013年4月号)でも取り上げたように、ポイントリリースについては事前にproposed-updatesリポジトリで状況が確認できますので、システムなどを運用される方は検証環境を整えておき、必要に応じてフィードバックしていただけると助かります。次はまた3、4ヵ

注12) 昔はdebian-multimedia.orgという名前でした。しかし、公式サイトのような名前であったため、ユーザがこのサイトのリポジトリにあるパッケージについてDebian側にバグレポートを投げる事態が頻発しました。そのため名前が変更された、という経緯があります。また、debian-multimedia.orgドメインは現在第三者の手に渡っていますので、利用しないようにご注意ください。

月後にアップデートがかかるはずです。

次のリリース「Jessie」の開発へ

Wheezyがリリースされたということは、アップデートも再開されるということです。フリーズ期間中は、unstableへのアップロードは原則禁止で、代わりにexperimentalへのアップロードが推奨されていました[注13]。experimentalは、stable/testing/unstableとは完全に分離されており、

```
# apt-get install -t experimental <package>
```

のように明示的にexperimentalをターゲットとしてインストールしないとパッケージは導入されません。unstable→testingのような自動的なパッケージの移行は実施されない隔離空間です(そのため、experimental(実験)という名前が付いています)。

unstableのアップロードが解禁されることで、パッケージメンテナらがexperimentalにアップロードしていたパッケージを次々とunstableへ放り込むことになります。この号が発売されるころには落ち着いているとは思いますが、しばらくの間はunstableには激しい変化が繰り返し訪れることになるでしょう。

宇宙やクラウドでも活躍するユニバーサルOS

ISS(国際宇宙ステーション)のラップトップがWindowsからDebian 6.0に置き換えられることがメディアで報じられました[注14]。これはSqueezeのテーマであるspacefunの壁紙が気に入ったから……ではなく、それまでのWindowsにおけるワーム被害などの教訓を踏まえて、最初からLinuxへの移行を視野に入れていたようです。

日本でもこの話題を取り上げている人を見かけました。その中で「サポートは大丈夫なのか」という見当違いの論評をされる方がいらっしゃいました。しかし、NASAのシステム担当者(ISSのシステムを担当する企業United Space Allianceのマネージャー)は、

We migrated key functions from Windows to Linux because we needed an operating system that was stable and reliable—one that would give us in-house control. So if we needed to patch,adjust,or adapt,we could.

(WindowsからLinuxへの移行を行うのは、我々には安定して信頼できるOSが必要だからだ——その要素の1つは我々の手でコントロールが可能である、ということだ。パッチが必要であれば当てられる、調整が必要なら調整できる、変更が必要なら変更できるようになる)

というように、「企業から与えられるサポート」ではなく「自分たちで手当てできること」に価値を見出しています。Debianのような「コミュニティ」ディストリビューションには非常に相性がよさそうです。

そして、Debian 7がリリースされたのと同じ週、Googleが提供するクラウドコンピュータサービス「Google Compute Engine」の主要サポートOSにDebian 6と7が選ばれたことが発表されました[注15]。これで、Amazon Web Services、Windows Azureと有名どころのクラウドサービスでは、Debianがすぐに利用可能、あるいはサポートが提供されている状態となりました。SD

注13) リリースマネジメント上、testingに問題が発生した場合、unstableで直っていればフリーズを「unblock」してunstableのパッケージを取ってくるようにします。しかし、unstableに新しいバージョンが入っていたりするとtestingとunstableの差分が必要以上に大き過ぎて、リリースチームが困ります。そのため、unstableへのアップロードは原則禁止としています。ですが、設定上はunstableへのアップロードを妨げるものはとくにありません。あくまでもアップロードする開発者にその辺は任せられています。なお、testingにあるパッケージに問題があっても、「unstableにアップロードされたバージョンへのアップグレードは難しい」と判断された場合には、testing-proposed-updatesに最小限のパッチを加えたバージョンをアップロードして、ポイントリリースで問題を修正する、というアプローチが採られます。

注14) URL http://www.zdnet.com/to-the-space-station-and-beyond-with-linux-7000014958/

注15) URL http://googleappengine.blogspot.fr/2013/05/bringing-debian-to-google-compute-engine_9.html

ウチの子症候群の変遷など

大村 芳樹
Yoshiki OOMURA
レッドハット(株)グローバルサポートサービス
シニアテクニカルサポートエンジニア

自己紹介とか

はじめまして。レッドハットGSS(グローバルサポートサービス)の大村です。部門名から想像がつくとおり、製品サポートを担当する部門に所属しており、Red Hat Enterprise Linux(以降RHEL)を担当しています。ユーザを全力で「サポート」することがミッションです(これ、前フリです)。

本連載も、コンサル、SAなど、弊社の名だたるファンタジスタによってここまで回を重ねてきたわけですが、今回は当社と技術評論社の懐の深さから、決してファンタジスタでもない一般人にも何か書かせてみようという試みのようで、筆者に白羽の矢が刺さることと相成りました。

職種上あまり仕事の具体的なことは書きにくいという、身悶えするような制約にあとから気がつきましたが、今回は恵比寿の某所で囁かれがちな「ウチの子症候群」にしばしおつきあいください。諸々別な考えのある方もいらっしゃると思いますので、単なる一般人の私見だ、と先にお断りしておきます。

「ウチの子」談義の前提

本誌読者の職種やシステムへのかかわり方はそれぞれですが、「お客様・ユーザから問い合わせを受けて応答する」ということを経験している方も多いと思います。筆者が初めてLinuxに仕事として触れてから15年は経っています(どこから仕事だったか正直あまり覚えていないのは内緒です)。そして当社に参加してからもう7年目になります。当初はラーニングサービスに所属し、研修担当でしたので、読者の皆さんの中にはお目にかかった人もいるかもしれません。そのあとサポートサービスに移り、現在に至ります。入社以前から、コンサルテーション・研修・製品サポートをしてきたこともあり、形こそ違えどユーザに直接「説明」し「質問を受ける」という日々を過ごしてきました。それを振り返ると、Linuxに関与している人の層が随分と様変わりしたんだな、と感じることがよくあります。ユーザの層が広がったという歓迎すべきこととも、恐らく関連しているのだと思います。

「ウチの子」の親は誰?

ファンタジスタ達からはお叱りを受けそうな気もしますが、15年前のLinuxはユーザにとって「ペット」のような印象がありました。何だか言うことを聞いてくれないときには、自分なりになだめすかして、「うん。これが全面的に正しいかはともかく、ご機嫌がなおったようだな」と楽しんだり、やり方が悪くて余計にご機嫌をそこねてみたり。そして、その情報をお互いに共有してみたり。同じペットを飼う親(?)同士のコミュニケーションありきで「ウチの子は、最近ちょっと反抗期でね」「正しい餌は与えてみたのかい? ウチでは……」的なやりとりを通じて見識を深めていたような気がします。間違いなく「ウチの子」の親はユーザである前提が多分あって、何となくそういうモノだと思って過ごして

いました。

時は経ち、弊社製品(まだ当時は部外者でしたが)にも「Enterprise」なるミドルネームが付くようになって、徐々にペットブームが訪れます。Linuxで動くシステムに関与する人数や階層が増えてくることによる変化を本当に感じるようになったのは、Red Hatに入ったあとのことだったと記憶しています。

「ウチの子がこんな事を」事例

本題の「ウチの子症候群」に戻ります。よく見かけるサポート事例に、次のようなものがあります。あくまで日常的なお問い合わせの例として書いているもので、具体的なログの引用(実際にいただいた問い合わせの「対象」など)は割愛します。ログ自体を見るのが好きな人は、自分の子のログを見て我慢していただく必要があります。

Q-1:
以下のメッセージが出力されました。影響と対策を教えてください。
<……1行程度のログメッセージ引用>

同様の問い合わせのバリエーションは多数ありますが、ここでは問い合わせのゴールを「影響と対策」に置いている例にしてみました。冒頭で述べたとおり、GSSのミッションは「製品ユーザを全力でサポート」することです(「サポート範囲」とか、堅苦しい話は今回は省きます)。それでも、ログの引用だけで影響と対策まで回答するはが難しい場合も多々あります。

そこで、たとえばログからわかる情報を可能な範囲で回答すると共に「該当のメッセージはINFOレベルのログであり、XXXの挙動において問題がなければ、通常は気にする必要のあるログではありません」といった回答を返すことになります(話が長くなるのでここはINFOやDEBUGレベルのログの際の物語に絞らせてください)。

ユーザによっては、次のような追加質問を送る必要がある、と感じる方もいるようです。

Q-2:
なぜ、通常影響がないという回答となるのかを説明してほしい。影響がないログがなぜ出るのか。

このやりとりを収束するには、どうしても避けては通れない難題があります。「INFOレベルのログも、DEBUGレベルのログも、挙動を追ううえでは有用ですが、ログからシステムの状態をはかることはかなわない」ということを「説明する」という行為です(えっと……。付いて来られますか?)。

「ウチの子がこんな事を」の考察

上のQ-1〜2の流れは次のようになります。ある日の朝、ある家の子が日記にこう書いていたとします。

「昨日の夜はウニ丼を食べた:丼2杯分」

これを両親が、読んでいいのか、という話は置いておいて、これを見た親御さんが本人をつれずに医者を訪れ、日記の文言を伝え、「ウチの子、どうしたんでしょう? どうすれば?」と聞く。医者は「お子さんの様子はどうなんですか?」と尋ねたうえで「ウニ丼を食べたんですね。体調に異変がなければ気にする必要はないと思います」と答えるしかありません。

それに対して、親御さんが「なぜ、体調に異変がなければ気にする必要はないのですか?」と聞いてしまうと、この時点ですでに興味は子供ではなく日記の解釈に移ってしまっていることになってしまいます。

「腹が痛い」と日記に書いてあったならともかく、問題は日記に書かれている内容ではなく、たとえ話とはいえ両親が子供の日記を盗み見たことでもなく、「お子さんの様子」だということです。

日記の例と違い、自分の管理するサーバのログを確認することは、必要な行為です。普段の

ログを知らなければ、ログから兆候をはかることはできないので、ログを通じて「ウチの子＝管理対象のサーバ」の状態を見守ることはとても良い取り組みです。あるいは、その取り組みによって「何か腹の具合が……」というメッセージを読み取り、顕著化していない問題の兆候を見いだすこともできるかもしれないからです。顔色や朝の挨拶の元気さなど、すべてを見守ったうえで、日記や発言に気を配るのは、「それぞれ」必要で、どれか1つだけで成立するものではないので、ログも気にする習慣があることも重要です。

「ウチの子。オタクの子」

ウニ丼を例にするかどうかはともかく（カレーの場合もあります）、実際にこうした説明を行っているケースはいくつもあります。また、ここでは、ログ出力に関するやりとりを例に取りましたが、こういう傾向のやり取りは実際に増えているという気がします。ここで言うのは、管理対象のサーバを全面的に「オタクの子（Red Hatの子だったり、その他ベンダーの子だったり）」として扱う傾向のことを指しています。

これは必ずしも間違いとは言えません。確かに見方によっては、人手に渡ろうとも「各コンポーネントとディストリビューション＝ウチの子」と言えなくはありません。

なので、「お腹が痛い」と言っていれば「この薬を与えて見てください」という助言を与えたり、全力で今の親御さんを「サポート」するのが我々の部門のミッションです。

ただ、我々は「ウチを出たウチの子」に会いに行き、「どうした？」と直接なだめることはできません。日々顔色や発言を見守ってほくそ笑むこともかないません。

私見を重ねてしまいますが、当社は人手に渡った「Red Hatから来た子」を育て、見守っているそれぞれの親御さんを見守る祖父や祖母（家系図がおかしなことになりますが）、あるいはペットショップの店員や、獣医のような、子育てを支援できる存在なのかな……、と思っています。

そのためにレッドハットでは各種の「親御さん支援」のためのサービスを提供しています。ときには親御さんに「Red Hat育ちの子供ってのはな……」と基本的なしつけから教えることができる研修サービス。ケガの様子を見たり、どう接したらいいかわからない人に、子育てのサポートを行うグローバルサポートサービス。もっと踏み込んで「ウチの子をもっとキレイに」「ウチの子をもっと優秀に育てるには」という要望だって、トップブリーダーにあたるファンタジスタ層が相談に乗ることができます。

次回予告など

今回、機会をもらったにもかかわらず連載のタイトルを忘れて、1ミリも恵比寿でない駄文に明け暮れてしまいました。そのうえ一部恵比寿以外で作文してしまったことをお詫びいたします。次回は連載のタイトルを知り尽くした男が、恵比寿ならではの恵比寿通信を上げてくれることと思います。ハードルを上げ合って高め合う、素敵な職場です。

今後ともよろしくお願い申し上げます。SD

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

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

2013年7月号

第1特集
データの価値を見直しませんか?
ここからはじめるデータ分析学習

第2特集
自分の定規を持っていますか?
ボトルネックを探れ!「ベンチマーク」活用テクニック

一般記事
・小規模プロジェクト現場から学ぶJenkins活用

1,280円

2013年6月号

第1特集
わかった人だけメキメキ上達
ちゃんとオブジェクト指向できていますか?

第2特集
研修じゃ教えてもらえない!?
あなたの知らないUNIXコマンドの使い方

一般記事
・リアルタイム分散処理「Storm」ほか

1,280円

2013年5月号

第1特集
IT業界ビギナーのための お勧め書籍55+α
新しい季節に君へ

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?

一般記事
・バーチャルネットワークコントローラ2.0開発の実際

1,280円

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

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。

第40回 Ubuntu Monthly Report

LibreLogoとその日本語化

おがさわらなるひこ　OGASAWARA Naruhiko
ogasawara.naruhiko@miraitsystems.jp

今回はUbuntu 13.04で標準になったLibreOffice 4.0の、LibreOffice Writer上で動くLogo的プログラム環境「LibreLogo」でのプログラミングとその日本語化について解説します。

LibreLogoとは?

本誌2013年6月号p.116からの「Ubuntu 13.04 "Raring Ringtail"」紹介記事でも取り上げられていましたが、Ubuntu 13.04ではLibreOfficeのバージョンが4.0に上がりました。本誌2013年3月号の本連載でのLibreOffice 4.0紹介にもあったとおり、機能的に区切りとなるリリースというわけではないのですが、それでもいくつかおもしろい機能が搭載されました。その1つが今回紹介するLibreLogoです。

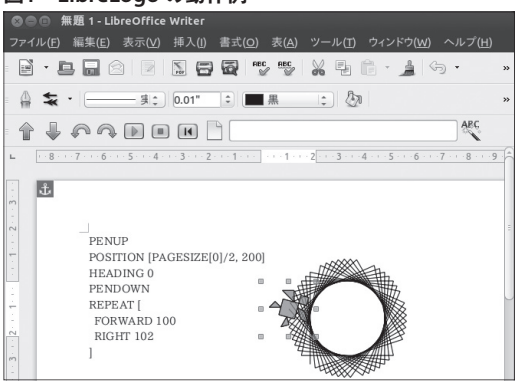
LibreLogoは「LibreOffice Writerで動くLogo的なプログラミング環境」です(**図1**)。ところで「Logo的」というからにはオリジナルのLogoというものがあるのですが、みなさんご存じでしょうか?

Logoは、MITの人工知能研究所(AIラボ)のシーモア・パパート氏のグループが中心となって作られたプログラミング環境です。パパート氏はジュネーブ大学所属の認知学者、ジャン・ピアジェ氏の研究に感銘して彼のもとで5年ほど研究し、「子どもは誰からも教わらなくても多数の物事を学ぶものである」「子どもの成長について研究するには、成長の対象について深く理解しなければならない」という理論に強い影響を受けました。

パパート氏はピアジェ氏の研究から一歩踏み込み、子どもたちの「自発的な」学びを引き起こすための方法として、コンピュータの活用に着目しました。その取り組みがLogoです。

Logoの代名詞とも言える「タートル」は、「位置」だけでなく「向き」を持つことが大きな特徴です。位置と向きを持つ存在ということは、たとえば子どもたち自身が校庭で、木の枝を持って、線を描きながら動くということと対応付けられます。このように「体の動きと対応付けできる」(身体同調)道具を与え、おもしろい、興味深いと思える(自我同調)課題を与えることで、さまざまな気づきを起こさせ、子どもたちの自発的な学びのきっかけとなるのがLogoの主な目的でした[注1]。

図1　LibreLogoの動作例



注1) ピアジェおよびパパートの考えと、それがLogoでどこまで実現したかはここでは語り尽くせないため、『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』(シーモア・パパート著、未來社)をご一読いただくと良いでしょう。「マインドストーム」といえば「LEGO Mindstorms」を思い浮かべた方もいるかもしれませんが、あのプロジェクトを手がけたミッチェル・レズニック氏はパパート氏の指導を受けており、そのレズニック氏が中心となっているプロジェクトが、昨今何かと話題のビジュアルプログラミング環境「Scratch」です。

さて話をLibreLogoに戻しましょう。LibreLogoはハンガリー人のLászló Németh氏によって開発された、Logoとほぼ互換のプログラミング環境です。最初はLibreOffice Writerの機能拡張として作成されました。

昨年のLibreOffice Conference 2012 BerlinでのNémeth氏による発表[注2]によると、LogoをLibre Office Writer上で実装することは、WriterをIDEとして用いることができるだけでなく、完全なUNICODE対応、高品位なベクターグラフィックスが得られ、教育現場でもオフィスソフトの教育とプログラミングを同時に教えることができるなど利点が多い、とのことでした。さらに教育言語としてだけではなく、ベクターグラフィックス記述言語としてもビジネス用途でも利用でき、またLibreOfficeのPythonによる拡張の良いサンプル（たった1000行程度だそうです）でもあります。

そして晴れてLibreLogoは、LibreOffice 4.0から正式に本体に取り込まれました[注3]。では、LibreLogoの実力を見てみましょう！

LibreLogoで遊んでみる

実は、UbuntuのパッケージングではLibreLogoはLibreOfficeとは別になっています。**図2**のようにしてインストールしてください[注4]。

インストールしたらWriterを起動し、[表示(V)]-[ツールバー(T)]-[Logo]にチェックを入れましょう。新たにツールバーが表示されます。試しに左側の4つのボタンを適当に押してみましょう。タートル（緑色のウミガメみたいなもの）が前に進んだり、後ろに戻ったり、左右に向いたりするはずです（**図3**）。

次は、コマンドでタートルを操作してみましょう。LibreLogoのツールバーの右側に大きな入力窓があります。これは「コマンドライン」といって、LibreLogoのちょっとしたコマンドを試すのに便利です。その隣にある2つのアイコン「画面を消す」で先ほどタートルが動いた軌跡を消し、「もとの場所へ」でタートルが最初にいた場所に戻しましょう。そして「コマンドライン」に「fd 100 rt 178」とタイプして、Enterキーを長押ししてください。どうです？ちょっときれいじゃありませんか？

タートルを操作するコマンドは一部ですが**表1**に示しました[注5]。LibreLogoではほとんどのコマンドに省略形が用意されています。コマンドラインでいろいろ試してみましょう。

LibreLogoでプログラミング ①タートルグラフィックス

ではプログラミングに挑戦です。プログラミングを行う場合は、コマンドラインではなくそのままWriterのテキストエリアに書くことになります。

たとえば、正方形を描くプログラムは**リスト1**のようになります。「まっすぐ進んで右に直角に曲がる」を4回繰り返せば正方形が描けることは当然です。これをWriterのテキストエリアにタイプし、ツールバーの左から5つめの「実行」ボタンを押してください（プログラムを止めたいときにはその隣りの「停止」ボタンを押します）。またコマンドラインの隣

図3　LibreLogo ツールバー

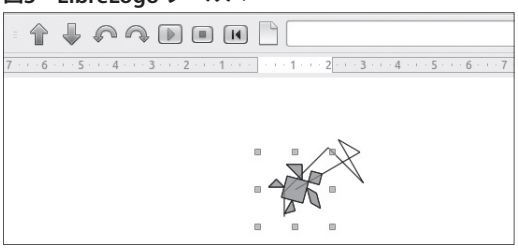


図2　LibreLogo のインストール

```
sudo apt-get install libreoffice-librelogo libreoffice-script-provider-python
```

注2） http://www.numbertext.org/logo/librelogo.pdf

注3） LibreOfficeに限った話ではありませんが、教育現場に採用されると「学校で慣れたものを使いたい」ということで家庭や職場に入り込んでいくという効果を狙っているのではないかと考えます。

注4） そもそも13.04より前のバージョン（12.04LTSなど）を使っておりLibreOffice4.0を利用したい場合は、LibreOfficeパッケージングチームのPPA (https://launchpad.net/~libreoffice/+archive/libreoffice-4-0) を使いましょう。

注5） LibreLogo作者のNémeth氏によるマニュアル https://wiki.documentfoundation.org/User:Nemeth も参照してください。

の「マジックアイコン」は、LibreLogoプログラムの文法をチェックし、正しく解析できたコマンドは正式名称の大文字にしてくれます。たとえば「fd」と書いてマジックアイコンを押すと「FORWARD」にしてくれるわけです。

ところで同じことを何度も書かずにループしたいですよね。LibreLogoにもいくつかのループのコマンドがあります。一番よく使われるREPEATコマンドで書きなおしてみました(**リスト2**)。かなりスッキリしましたね。

なお、REPEATのほかによく使う制御構文を**表2**に挙げておきました。

さて同じように正三角形を描く……のですが、いったん算数の知識は忘れ、ここではちょっと「タートル幾何学」的な考え方を試してみましょう。

先ほど、正方形を描くときに、「まっすぐ行って90度」を4回繰り返すと元の位置に戻ってくることを見ました。「ぐるりと1回りして元の向きに戻る」とき、曲がった角度の合計は360度になる、ということを「タートル全回転量の定理」と呼びます。この定理はタートルになったつもりで自分で歩いてみると体得しやすいです。

これがわかってしまえば、三角形は3回曲がって戻ってきたいわけですから、360 ÷ 3 = 120という数字が出てきます。正三角形を描くためには**リスト3**のようになることがわかりました。各自試してみてください。

表1　タートル操作関係(描画関係)のコマンド

コマンド	省略形	意味
FORWARD n	fd	n歩進む
BACK n	bk	n歩戻る
RIGHT n	rt	n度右に曲がる
LEFT n	lt	n度左に曲がる
PENUP	pu	ペンを上げる
PENDOWN	pd	ペンを下ろす
HOME	-	元の位置(ページ中央)に戻る
CLEARSCREEN	cs	画面をクリアする
HIDETURTLE	ht	タートルを非表示にする
SHOWTURTLE	st	タートルを表示する
HEADING n	-	上を0度としてn度方向を向く
POSITION [x, y]	pos	原点を用紙左上として座標(x,y)に移動する ※ "["の直後、"]"の直前は詰めること
LABEL "str"	-	文字列strをタートルの位置に書く

表2　制御構文

コマンド	備考
REPEAT [n] [...]	nを省略すると無限ループ
FOR c IN str [...]	文字列・配列でのイテレータ
WHILE expr [...]	普通のWHILE文
REPCOUNT	擬似変数。 ループ内でのループ回数を示す
IF expr [trueblock] [[falseblock]]	falseblockは省略可

LibreLogoでプログラミング ②手続きを作る

さてこうやって作った正方形と正三角形ですからいろいろ使いまわしたいですよね。そんなわけで(Libre)Logoにも、こういったひとかたまりの処理に名前を付ける機能があります。プログラミング用語の手続きとか関数といったものの作成です。

そのための構文がTO...ENDです。さっそく、正方形と長方形に名前を付けてみましょう。**リスト4**を見てください。MYSQUAREとMYTRIANGLEという手続きを作っています。「MYSQUAREするには……終わり」と読み下すことができます。

TO文の後ろの手続き名の、さらに後ろに書いたものは引数になります。**リスト4**の場合、大きさを自由に指定できるようにしています。引数は不要なら省略できます。

一点注意ですが、LibreLogoは比較的改行に寛容な言語です(一番最初の例で「fd 100 lt 179」とつなげ

リスト1　正方形を書くプログラム(素朴版)

```
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
```

リスト2　正方形を書くプログラム(REPEAT使用)

```
REPEAT 4 [
  FORWARD 100
  RIGHT 90
]
```

リスト3　正三角形を書くプログラム

```
REPEAT 3 [
  FORWARD 100
  RIGHT 120
]
```

て書いていたことを思い出してください)。しかし、TO文については引数の直後で改行、という決まりがありますので気を付けてください。

例が**図4**です。「PENUP」で始まる行の意味を読み解いてみると良いでしょう。

では読者のみなさんにクイズです。このMYSQUAREとMYTRIANGLEを使って、**図5**の右のような絵を書くHOUSEという手続きを作ってみてください。何も考えずにMYSQUAREとMYTRIANGLEを並べると左のようになってしまいます。これをデバッグするという経験も子どもたちには重要だと、パパート氏は考えていました。そして「タートル幾何学」ならば、子どもたちが正解にたどり着くやり方がいろいろ体験できる、というのがパパート氏の主張です[注6]。ぜひ、みなさん自身が解くだけでなく、「全然プログラミングの経験がない人ならどう考えるだろう」「そのときどんなアドバイスをすればいいだろう」などと考えてみてください。

LibreLogoでプログラミング ③応用編をいくつか

Logoを使ったおもしろい例として再帰が挙げられます。再帰的な考えは、学校教育では数学的帰納法としてかなり後に導入されますが、タートル幾何学によって子どもたちに受け入れやすく、なおかつ多くの驚きを与えてくれます。紙幅の関係で例を出すに留めます(**図6**)が、いろいろ応用を考えてみてください。

注6) このHOUSEの例自体もパパート氏の著書に出てくるものです。

リスト4　TO ... END 構文の使用例

```
TO MYSQUARE size
REPEAT 4 [
  FORWARD size
  RIGHT 90
]
END

TO MYTRIANGLE size
REPEAT 4 [
  FORWARD size
  RIGHT 120
]
END
```

LibreLogoはLogoに加えて、さまざまなカラフルな図形描画、線の種類や塗りつぶしの種類、文字列操作などなど充実しています。LibreLogoの公式

図4　「手続き」の例

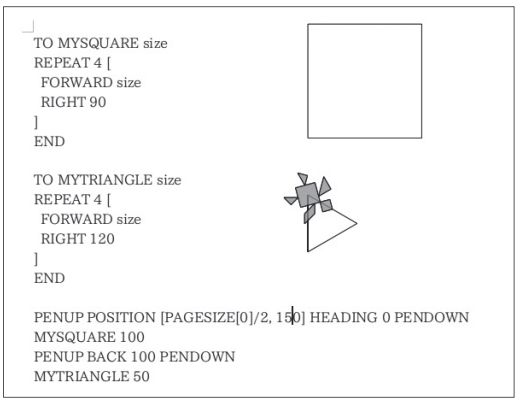


図5　例題「HOUSE」

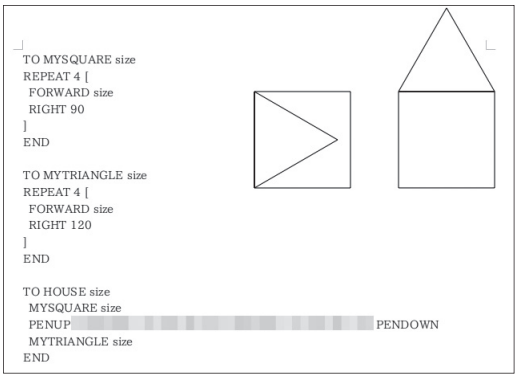
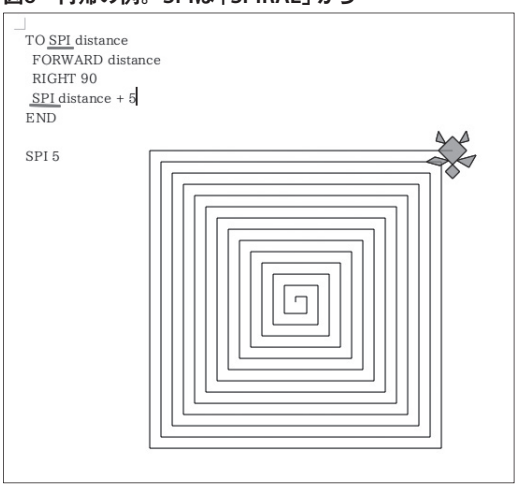


図6　再帰の例。SPIは「SPIRAL」から



ページ[注7]にさまざまなサンプルが落ちていますので、ぜひ自分で試してみましょう。

またLibreLogoで作成された図形はWriterの図形扱いになるので、そのままコピーしてプレゼン資料などに貼りこむことができます。繰り返しパターンなどを描画する場合はDrawなどよりずっと簡単ですので、プレゼン背景などにカッコイイ幾何学図形を入れたりするのもいいですね。

日本語LibreLogo

LibreLogo作者のNémeth氏によると「10歳の子どもに使ってもらうには、その母語でコマンドが使えることが大事だ」とのことで、LibreLogoのコマンドは翻訳可能になっています。4.0.1からは日本語のコマンドもサポートされました。

ただし、日本においては母語を利用できるようにすることについていくつかの問題があります。一番大きいのは日本語入力についてです。漢字を中途半端に開くと日本語入力ソフトウェアで素直に入力できなくなりますし、といってあまり難しい日本語を使うこともできません。

LibreLogoのコマンド翻訳については筆者が中心で行ってきましたが、正直筆者は教育の専門家でもなんでもないので、さしあたってのポリシーは「コマンドは英語で入力し、マジックアイコンをクリックすると日本語になって読みやすい」を目標としています。まだ試行錯誤を続けている段階で、ぜひいろいろな方からのご意見をいただければと思います。**表3**に一部の例を示しました[注8]。

LibreLogoの日本語コマンドを有効にするには、[ツール]-[オプション]-[言語設定]-[言語]の[ドキュメントの標準言語]で[西欧諸言語]を“なし”にして[アジア諸言語]を“標準 - 日本語”にしてください。なお、常にこうすると和欧混在の文書で問題になるため、LibreLogoで遊ぶドキュメントだけこの設定にするため「現在のドキュメントのみに適用」をオンにしてください(**図7**)。

日本語LibreLogoについて詳しくはWikiページ[注9]を参照いただくとして、例だけ示しましょう。**リスト5**を入力して[注10]マジックアイコンを押して、実行した結果が**図8**になります。

日本語LibreLogoのコマンドを直してみよう

LibreLogoはPythonで書かれているため、通常のLibreOfficeのローカライズのしくみと異なり、手元で日本語コマンドのカスタマイズができます。

LibreLogoのローカライズされたコマンドは/usr/lib/libreoffice/share/Scripts/python/LibreLogo/LibreLogo_lang(_COUNTRY).

注7) LibreLogoがまだ機能拡張だったころのページをポータルに流用しています。http://extensions.libreoffice.org/extension-center/librelogo

注8) 完全なリストはhttps://wiki.documentfoundation.org/User:Naruoga/LibreLogo/command_jp にあります。

注9) https://wiki.documentfoundation.org/User:Naruoga/LibreLogo

注10) “❄”(雪結晶マーク)の文字はU+2744で、[挿入]-[記号と特殊文字(P)...]でフォント「DejaVu Sans」などを選ぶとサブセットDingbats内にあります。全角の「＊」などで代用すると、LibreLogoのバグでフォントの拡大縮小がうまくいきません。

表3　英語コマンドと日本語コマンドの対応表

英語	日本語
FORWARD	すすむ
BACK	もどる
RIGHT	右
LEFT	左
REPEAT	くりかえす
IF	もし
TO ... END	動きを作る ... おわり
FOR c IN str	ひとつずつcを次から取り出してstr
NOT a	次が正しくないa

リスト5　日本語LibreLogoのテスト用リスト

```
CLEARSCREEN HOME
HIDETURTLE
FILLCOLOR “SKYBLUE”
RECTANGLE PAGESIZE
FONTFAMILY “DejaVu Sans”
FONTCOLOR “WHITE”
FILLCOLOR “SKYBLUE”
PENUP
REPEAT 50 [
        POSITION ANY
        FONTSIZE 10 + RANDOM 80
        LABEL “❄”
]
```

propertiesというファイルに格納されています。日本語の場合はLibreLogo_ja.propertiesです。先頭をのぞいてみると「\u」でエスケープされたUCS2文字列であることがわかるので、「echo -e」で中身を確認できます（**リスト6**）。

では試しに、LEFTコマンドの日本語候補に「ひだり」を追加してみましょう。「ひだり」のUCS表現を確認する方法はたくさんあるでしょうが、ここは単にechoとiconvとodという組み合わせを使ってみました。これで「ひだり」=3072 3060 308aだということがわかるので、TURNLEFTの行を編集してみます（**リスト7**）。日本語LibreLogoを有効にして「left」とタイプしてマジックアイコンをクリックしてみましょう。「ひだり」と出れば成功です[注11]。

終わりに

LibreOffice Writerの機能の1つLibreLogoを紹介しました。LibreOfficeはMS Officeクローンを目指しているのではないという象徴的な機能だと思います。もし小学生ぐらいのお子さんがいれば一緒に遊んでみてはいかがでしょう？ SD

注11）まれにマジックアイコンがうまく動かないときがあります。そのときは別のコマンド（たとえば「fd」）を変換させてやるとそれ以降はうまくいったりするので、試してみてください。

図7　日本語LibreLogoを有効にする設定。枠で囲った部分を変更する

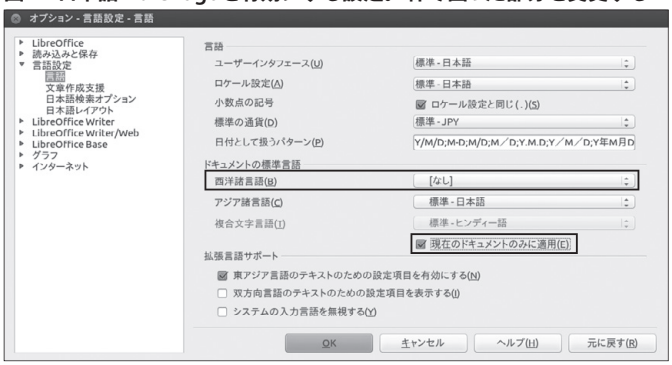
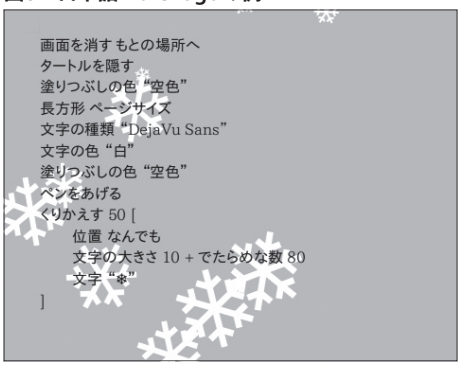


図8　日本語LibreLogoの例



リスト6　今のコマンド文字列のチェック方法

```
$ head /usr/lib/libreoffice/share/Scripts/python/LibreLogo/LibreLogo_ja.properties
# turtle graphics

FORWARD=\u3059\u3059\u3080|forward|fd
BACKWARD=\u3082\u3069\u308B|back|bk
TURNLEFT=\u5DE6|\u5DE6\u306B\u66F2\u304C\u308B|left|turnleft|lt
TURNRIGHT=\u53F3|\u53F3\u306B\u66F2\u304C\u308B|right|turnright|rt
PENUP=\u30DA\u30F3\u3092\u3042\u3052\u308B|penup|pu
PENDOWN=\u30DA\u30F3\u3092\u304A\u308D\u3059|pendown|pd
HOME=\u3082\u3068\u306E\u5834\u6240\u3078|home
POINT=\u70B9|point
head /usr/lib
$ echo -e "TURNLEFT=\u5DE6|\u5DE6\u306B\u66F2\u304C\u308B|left|turnleft|lt"
TURNLEFT=左|左に曲がる|left|turnleft|lt
```

リスト7　コマンド文字列に変更を加えてみる

```
$ echo -n "ひだり" | iconv -f UTF8 -t UCS2 | od -x
0000000 3072 3060 308a
0000006

/usr/lib/libreoffice/share/Scripts/python/LibreLogo/LibreLogo_ja.properties を編集
TURNLEFT=\u5DE6|\u5DE6\u306B\u66F2\u304C\u308B|left|turnleft|lt
↓
TURNLEFT=\u3072\u3060\u308a|\u5DE6|\u5DE6\u306B\u66F2\u304C\u308B|left|turnleft|lt
```

第17回

Linux 3.10の新機能
～タイマ割り込みを減らすNoHZ～

Text：青田 直大　AOTA Naohiro

NoHZに向けて

マルチタスクを実現するために、カーネルはタイマ割り込みをたとえば1,000分の1秒ごとに受け取って、そのタイミングでタスクのスケジュール処理を行っています。しかし、今ではマルチコアが一般的になってきていますので、1つのCPUにただ一つのプロセスを実行するように設定することもできます。そういった状況では、ほかに動き得るプロセスがないわけなので、このCPUにタイマ割り込みを行うのは無駄のように思えます。ということで、このタイマ割り込みを完全になくしてしまう（あるいは割り込みを減らす）ことを目標としてNoHZという機能の開発が進められており、Linux 3.10のRC版にはその第一歩となるpatchがマージされています。今回はこのNoHZについて解説します。

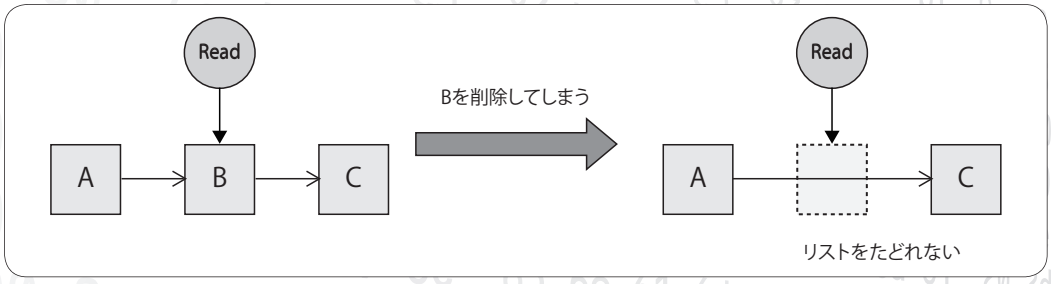
RCU

RCUのコールバックがNoHZに絡んできますので、NoHZの解説に入る前にまず、RCUについて触れておきます。

RCUとはRead-Copy-Updateの略で、Linuxカーネル内でReaders-Writer lockの代替としてよく使われる同期機構です。Readers-Writer lockとの違いを念頭におきながらRCUがどういったものか見ていきましょう。

Readers-Writer lockはある共通のデータを読み書きするためのロックです。**図1**のようなリストを考えてみましょう。あるスレッドがリストを読んでいる最中に、ほかのスレッドがリストの要素を削除してしまうと、リストを正しくたどれなくなってしまいます。そこでReaders-Writer lockでリストへの読み書きを制御します。

▼図1　スレッドがリストをたどれない例



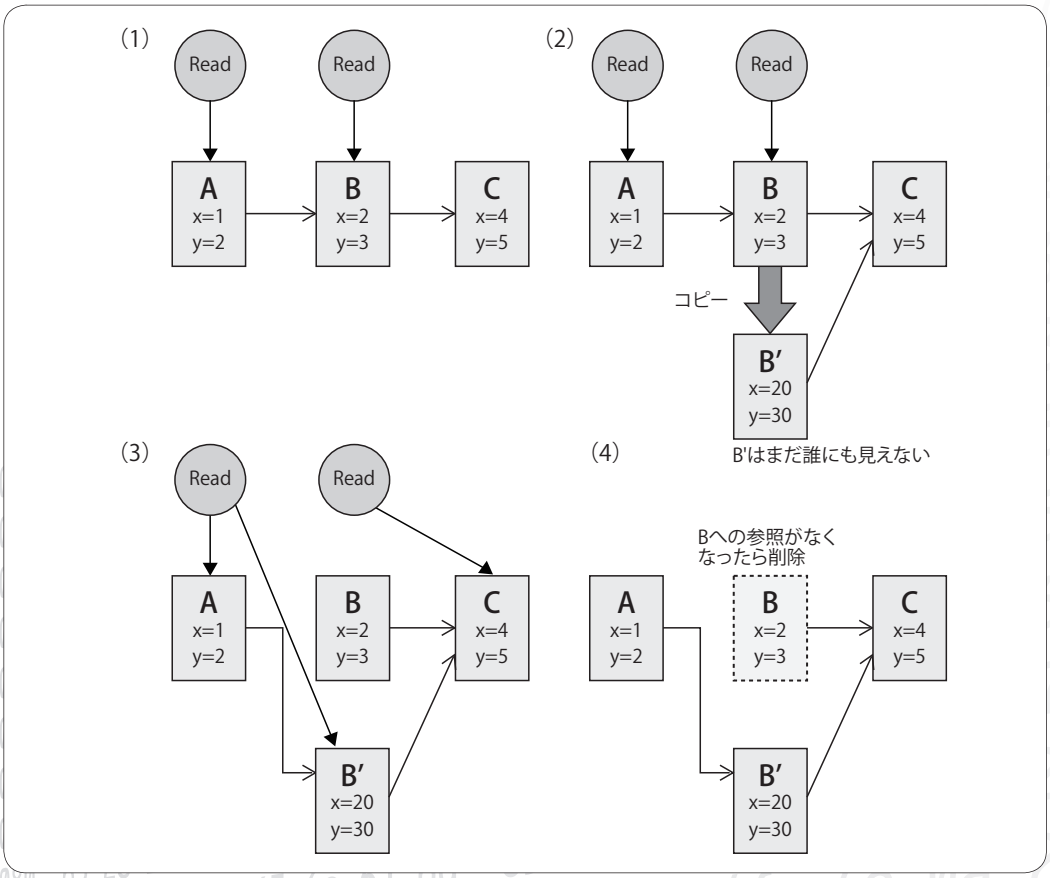
このロックでは読む側はリストを変更せずに読むだけなのでいくらでも同時に実行できます。

一方で書き手がいる途中では、ほかの書き手／読み手が同時に存在すると、今挙げた問題が起きてしまうので書き手は1つしか許されませんし、読み手と同時に実行することもできません。「Readers-Writer lockでは読み手と書き手が同時に実行できない」ということに注意してください。つまり、読む側が大量に、常にいるようなケースでは、書く側がいる間、読む側の作業は完全にストップしてしまうことになります。

これに対してRCUでは読み手と書き手を同時に実行できることが特徴的です。それではどうやって最初の読み書き問題を解決しているのでしょうか。**図2**をもとに、リストの要素を更新する様子を紹介していきます。初めのリストの状態は(1)です。このリストの全要素が読み手から参照されている可能性があります。(2)では、BをB'にコピーし、B'のデータを更新します。この時点までは新しいデータは読み手には見えません。そして(3)でAからBへのポインタをB'に切り替えます。この操作は「ポインタの更新がアトミックに行える」ことによって壊れないことが保証されています。さて、ここでB自体はまだ削除されていないので、Bを参照していた読み手は、とくに問題なく動作を継続できます。(4)では読み手がいなくなるのを待ってから、元のBを削除しています。こうやって読み手と書き手とが同時に実行できるようになっています。

さて、このようにRCUの書き換え部分自体はシンプルな処理になっています。大きな問題になるのは「どうやって元のデータを参照している

▼図2　リストの要素を更新する様子



読み手がいなくなったことを知るのか」ということです。これを実現するためにLinuxカーネルでは2個所に手を入れています。

まず、読み手が読んでいる部分(RCU参照側クリティカルセクション)を“rcu_read_lock()”と“rcu_read_unlock()”といった2つの関数で保護します。このRCU参照側クリティカルセクションの間では、プリエンプションを禁止しておきます。つまり、この間ではコンテキストスイッチが起こりません。そしてCPUの個数幅のbit配列を用意しておいて、“synchronize_rcu()”でGracePeriod(Grace Period開始時点に存在する読み手がいなくなるのを待つ期間)を開始します(**図3**)。

この関数はbit配列のすべてのCPUに対応するbitをすべて立てておきます。さらに、コンテキストスイッチ時にそのCPUに相当するbitをクリアします。こうするとすべてのbitがクリアされた時点で、すべてのCPUがコンテキストスイッチを行ったことがわかります[注1]。RCU参照側クリティカルセクションの間で、コンテキストスイッチが起こらないことを考えると、仮にCPUがGrace Periodの開始時にRCU参照側クリティカルセクションにいたとしてもそのセクションがすでに完了し、元のデータを参照している読み手がすべていなくなったことがわかります。

ところで、**図3**では書き手が“synchronize_rcu()”を使って読み手がいなくなるのを待って削除処理を行っていますが、その代わりに“call_rcu()”を使い削除処理を行うコールバック関数を登録しておいて、あとでほかのスレッドから読み手がいなくなるのを待って削除処理を実行するようにもできます。こうやってGrace Periodの処理をほかのスレッド(“rcu_bh”や“rcu_sched”といったカーネルスレッド)に移しておけば、書き手は待つ必要がなくなりレイテンシを改善できます。Linux 3.6以降ではこれらのカーネルスレッドに多くのGrace Periodが移動されています。

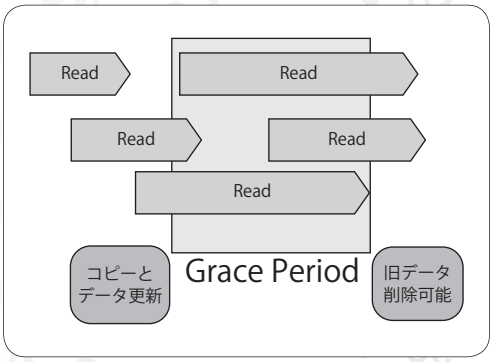
さらにLinux 3.8からはcallback-free CPUsというpatchが入っています。この機能を有効にすると、bootパラメータ“rcu_nocb=”でRCUのコールバックを「実行しない」CPUを指定できます。このCPUで実行されるはずだったコールバックはほかのCPUで動く“rcuox/N”(xはRCUの種類によってbかpかsで、Nはは CPU番号)というカーネルスレッドが担当することになります。このカーネルスレッドを実行するCPUを制限したり、優先度を変更できるようになります。このcallback-free CPUsの機能を使うことで、あるCPUでカーネルがするべき仕事が最小になるということがNoHZでのポイントになります。

ここまでのRCUの話をまとめておきましょう。

- RCUは同期機構の1つ
  読み手のロックコストが低い
  読み書きが同時に実行できる
  ただし削除処理をどこかで実行しなければならない
- 削除処理はコールバックとして後で実行できる
- RCUコールバックを実行しないCPUを設定できる

注1) このbit配列を使う方法は昔の実装で、1つのbit配列のロックをすべてのCPUがとりあうというスケールしない実装になっています。今はTree RCUというよりスケールする方法に書き換えられています。

▼図3 GracePeriod



dyntick-idle mode

では、NoHZの解説に入っていきましょう。タ

イマ間隔を減らすことができる状況には大きく分けて2種類があります。1つはCPUが何もすることがなくアイドル状態にあるとき、もう1つが冒頭で挙げたCPUにただ1つの実行可能なタスクしかないときです。アイドル時にタイマ間隔を減らすことをdyntick-idle modeといいます。これはLinux 3.8以前にもすでに入っていた機能ですが、簡単にこちらについても見ていきましょう。

タイマ割り込みというものは結局のところ、CPUを複数の実行すべきタスクのそれぞれに切り替えさせるために使われています。なのでCPUにやるべきことがないアイドル状態ではタイマ割り込みを送る必要はありません。

dyntick-idleのメリットとデメリットについて見てみましょう。メリットはもちろんタイマ割り込みが減ることによってCPUが起こされて無駄な仕事をする必要がなくなることです。こうしてCPUが電力消費の少ない深いアイドル状態により長く入ることができるようになります(**図4**)。

dyntick-idleでタイマ割り込みを減らしていないカーネルは、割り込みを減らしているカーネルよりも2～3倍速くバッテリを消費すると言われています。また、多くの仮想マシンを動かしている状況でもdyntick-idleは有用になります。dyntick-idleがない場合、1,000秒に1回の割り込みによる処理が仮想マシンの数だけ実行されるわけで、ホストのCPUの多くの時間が実際にはとくに何もしていないアイドル時の割り込み処理に割かれてしまいます。

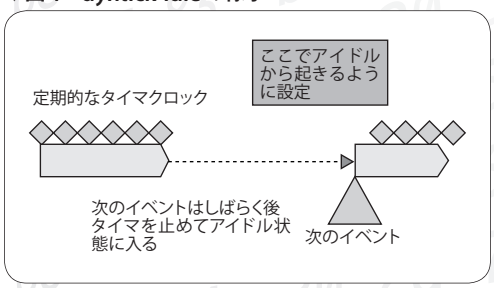
デメリットのほうはどうでしょうか。dyntick-idle modeはアイドル状態に入るときに、次にスケジュールされているタイマイベントをチェックすることで動作しています。つまり、そのイベントがしばらく先であれば、CPUの定期的なタイマ割り込みを停止し、そのイベントを処理する時刻にタイマを設定してアイドルから復帰できるようにしています。そして、復帰した後はまた定期的なタイマも再開させます。ということは、アイドル状態に入るとき／出るときで、余分に処理が必要になってしまうというデメリットがあります。リアルタイムな応答が重要なシステムではこのレイテンシが問題になる場合もあります。

adaptive-ticks mode

先に述べたとおり、実行可能タスクが1つしかなければタイマ割り込みの意味はありません。実行可能タスクが1つしかないときに、タイマ割り込みをなくしてしまう(adaptive-ticks modeに入る)CPUをadaptive-ticks CPUといいます。CONFIG_NO_HZ_FULL_ALLのカーネルオプションを設定していない場合は、デフォルトではadaptive-ticks CPUはなく、カーネル起動パラメータnohz_fullで、adaptive-ticks CPUを指定します。逆にCONFIG_NO_HZ_FULL_ALLが設定されている場合、デフォルトでCPU0以外のすべてのCPUが、adaptive-ticks CPUになります。nohz_fullを使うと、これも上書きされます。まとめると**表1**のようになります。

ここでCONFIG_NO_HZ_FULL_ALLのときでもCPU0は、adaptive-ticks CPUとはならないことに注目してください。これは、もしもすべてのCPUのタイマ割り込みを止めてしまう

▼図4　dyntick-idleの様子



▼表1　カーネルオプションとnohz_fullの関係

	nohz_full=なし	nohz_full=による指定
CONFIG_NO_HZ_FULL_ALLなし	まったくなし	指定されたCPUのみ
CONFIG_NO_HZ_FULL_ALLあり	CPU0以外すべて	指定されたCPUのみ

と、現在時刻を知るgettimeofday()といった関数が正確な値を返せなくなってしまうためです。ここはdyntick-idleとは違うところです。dyntick-idleの方であれば、すべてのCPUのタイマを止めてしまっても、動いているプロセスが存在しないので問題とはなりません。

さて次に、adaptive-ticks modeのデメリットについても見ていきましょう。簡単に列挙すると次のようになります。

- POSIX CPUタイマが不正確になる
- プロセス実行時間などの統計情報が不正確になる
- スケジューラのLB_BIAS機能が無効になる
- 多くのperfイベントを同時に取得できなくなる

動作しているCPUのタイマ割り込みを止めてしまうので、時間管理ができなくなってしまいタイマプロセス実行時間の統計が正確ではなくなってしまいます。また、そのためにCPUの動作時間を取得できなくなるので、LB_BIASの機能がうまく動作しなくなります。なので、この機能を無効にしてしまいます。

最後に挙げられているものは複数のperfイベントについてです。以前の記事でも書きましたが、CPUで監視できるイベントの数には(CPUの種類ごとに固有の)限界があります。この限界数を超えるイベントを監視しようとすると、perfはCPUに登録するイベントを一定間隔でラウンドロビンに変更していきます。もちろんここで「一定間隔でラウンドロビンに変更する」にはタイマが必要なのでadaptive-ticks modeではこれが動作しなくなります。

NoHZとRCU

ではNoHZとRCUの話に入りましょう。最初に触れたようにRCUにはコールバックを処理する部分があります。このRCUコールバックがある場合には、そのCPUは基本的にNoHZに入ることができません。この問題を解決するには2つの方法があります。

1つめの方法は、カーネルオプションのCONFIG_RCU_FAST_NO_HZを設定しておくことです。そうするとRCUコールバックを持つCPUであってもNoHZに入るようになります。ただし、RCUの処理を行うために通常のタイマ間隔の4倍(たとえば通常1,000分の1秒ごとにタイマを設定していれば、250分の1秒ごとになる)でCPUを起こしています。

もう1つの方法がRCUの項で解説したcallback-free CPUです。これを使えば、そのCPUは完全にRCUコールバックの実行から解放されるわけなので、RCUのせいでNoHZに入れなくなるといったことがなくなります。

Linux 3.9でRCU callbck-free CPUが導入されたので、NoHZが十分な効果を発揮できるようになったというわけです。

NoHZの将来

Linux 3.10でadaptive-ticksが入ったように、NoHZの機能は大きく進歩しています。しかし、まだまだ改善の余地は残されています。1つは32bitへ対応することです。このpatchはもうあるようなので、そう長くかかることはないかと思われます。もう1つはadaptive-ticks modeにおいて、よりタイマ間隔を小さくすることです。今の実装ではadaptive-ticks modeに入るとタイマ間隔が1秒に1回にされます。これは従来の1,000分の1秒に1回に比べるとかなりの進歩[注2]ですが、完全にタイマを切ってしまえれば、それにこしたことはありません。それにはスケジューラでの実行時間統計などカーネルのコア部分に修正する部分がいくつもあります。

perf user-space return probe

もう1つ、今回はperfに追加されたuser-

注2) 1,000分の1秒に1回の割り込みの処理でCPUの1%の時間が使われているようです。

spaceでのreturn probe機能について見てみましょう。しばらく前からperfでuser-spaceのプログラムもprobeできるようになりましたが、Linux 3.10からはreturn probeもできるようになりました。これは指定した関数が、その関数を呼び出した元の関数に戻ったときをチェックできるprobeです。

perf probeに"-x"を使うと、probeしたいuser-landのプログラムを指定できます(**図5**)。そこでkernel内の関数と同様に関数名の後ろに"%return"を付けることでuser-spaceのprobeを追加できます。perf recordのコマンドで10秒間にmalloc()から返っている部分を記録します。

perf reportすると、malloc()を呼び出していたプロセスが記録されているのを見ることができます。一番多く呼び出していたのは、プロセス名はconsolelog.shでプログラムはbashのgmatch_wc()という関数の中からだというのがわかります(**図6**)。

それぞれのエントリで"Annotate ..."するとその関数の中でのイベント発生位置をさらに詳しくみることもできます。このreportはjump先を矢印で表示してくれたりして、なかなか楽しいのでぜひ見てみてください。

userlandのprobeについても開発がどんどん進んでいますね。こうしてお手軽にシステム全体でのperformance情報をとることができてとても便利です。

まとめ

今回はCPUが動いているときにも定期的なタイマを減らし、電力消費や無駄なカーネルの処理をなくす機能であるNoHZについて解説しました。SD

▼図5　probeしたいuser-landのプログラムを指定する

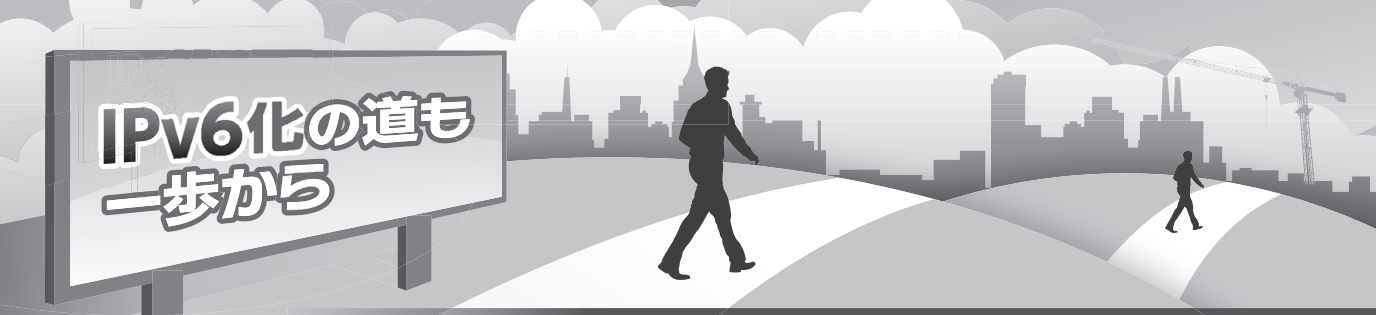
```
# perf probe -x /lib/libc-2.17.so -F (probeできる関数の確認)
…
lsetxattr
lutimes
malloc
malloc@plt
malloc_info
mblen
…
# perf probe -x /lib/libc-2.17.so malloc%return (probeの設定)
Added new event:
  probe_libc:malloc    (on 0x7ed80%return)

You can now use it in all perf tools, such as:

        perf record -e probe_libc:malloc -aR sleep 1
# perf probe -l (probeが入っていることを確認)
  probe_libc:malloc    (on 0x000000000007ed80%return)
# perf record probe_libc:malloc -aR sleep 10 (10秒間probeを監視する)
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.541 MB perf.data (~23626 samples) ]
```

▼図6　perf reportの実行結果

```
Samples: 2K of event 'probe_libc:malloc', Event count (approx.): 2588
 51.85%    consolelog.sh  bash                    [.] gmatch_wc
 12.36%            uname  bash                    [.] gmatch_wc
 10.82%            uname  libc-2.17.so            [.] __strdup
  9.27%            uname  libc-2.17.so            [.] _nl_intern_locale_data
  1.55%            uname  libc-2.17.so            [.] _nl_normalize_codeset
  1.00%       notify-osd  libxcb.so.1.1.0         [.] _xcb_in_read
  0.93%  at-spi2-registr  libxcb.so.1.1.0         [.] _xcb_in_read
  0.85%    consolelog.sh  libc-2.17.so            [.] set_binding_values.part.0
  0.85%    consolelog.sh  libc-2.17.so            [.] qsort_r
  0.85%    consolelog.sh  libc-2.17.so            [.] __fopen_internal
...
```



第8回 ネットワーク構築時の注意点と落とし穴（アプリケーション編）

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

前回は試験環境にてDHCPv6サーバの設定を行い、Linux、DNSサーバ、Webサーバ、MTA、プロキシのIPv6設定を行いました。これでIPv6に対応したサーバができあがりました。

今回は、アプリケーションのIPv6対応について取り上げます。

アプリケーション開発におけるIPv6対応とは?

サーバ構築まで完了すればIPv6対応は完了、と思われるかもしれませんが、サーバ上で動くアプリケーションまでを含めて「システム」です。そのため、アプリケーションのIPv6対応を行わないと、システムとしてのIPv6対応とは言えません。

アプリケーションにおけるIPv6対応とは、そのアプリケーションがIPv6とIPv4の両方で動作することです。この先、数年あるいは十数年という長い期間、IPv4とIPv6が共存することが予想されます。システムのライフサイクルを見据え、アプリケーションの保守性を考慮すると、単一のソースプログラムで、IPv4とIPv6の両方での動作を実現することが望ましい姿です。

アプリケーションにおけるIPv6対応の主なポイントは次の3点です。

- IPv6対応のプログラミング言語と実行環境を使う
- 通信部分をIPv6に対応させる
- データとしてIPアドレスを扱う個所をIPv6に対応させる

IPv6対応のプログラミング言語と実行環境を使う

アプリケーションをIPv6に対応させるには、その前提として、使用するプログラミング言語および、フレームワークやライブラリ、ミドルウェア、データベース（以下、DB）などの実行環境がIPv6に対応している必要があります。

そのなかでもプログラミング言語と実行環境に求められるのは、「IPv6で通信できる」ことです。この「IPv6で通信できる」とは、名前解決でIPv6アドレスが扱えることと、IPv6で接続できることが、両立して成り立たなければいけません。

前者は、「通信相手のホスト名からIPv6アドレスを名前解決できること」、「通信相手のIPv6アドレスからホスト名を名前解決できること」を指します。これらはプログラミング言語もしくは関連するライブラリがサポートすべきものです。後者は決められた通信方法で通信相手とIPv6で接続できることを指します。通信方法については、SMTPやHTTPなどの既存L4プロトコルを利用するものと、ソケット通信との2つに大別できます。これらはプログラミング言語もしくは関連するライブラリでサポートされるべきものであり、ミドルウェア、DBにおいてもそれぞれサポートされるべきものです。そ

のため、通信を行うプロトコルによって対応状況に差異があります。開発するアプリケーションが提供する機能を考え、それをIPv6に対応するうえでの過不足を判断しましょう。

各言語、実行環境ごとの対応状況に関する情報は、株式会社インテックが公開している「IPv4アドレス枯渇対応アプリケーションチェックリスト α 4 版」[注1]やIntenet Week 2012講演資料「スクリプト言語とIPv6 2012」[注2]に記載されています。各言語、実行環境はおおむね対応しています。

通信部分をIPv6に対応させる

通信処理のIPv6対応では、IPv4とIPv6の両方で通信できることを目指しますので、IPv4/IPv6のいずれかが通信できない状況も想定する必要があります。そのため、1つのIPアドレスを決め打ちして接続するのではなく、ホスト名を引いて返ってきたアドレスリストを上から順番に試していくロジックが望まれます。従来のIPv4のみの場合と比較すると、アルゴリズムの変更が必要となります。また、実装においては、IPv4/IPv6の双方に対応するオブジェクト、構造体、関数を使用しますが、これらはIPv4で使用されていた従来のものとは別に用意されていることが多いです。

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が作成した「アプリケーションのIPv6対応ガイドライン 基礎編」[注3]では、BSDソケットでのプログラミングを例に詳細に説明しています。今回はその概要を説明します。さらに詳細を知りたい方は、ぜひ、ガイドライン基礎編もご覧ください。

フォールバック

IPv6とIPv4の両方での動作を考えると、IPv6で接続できない場合にはIPv4へ切り替えて接続する、もしくはIPv4で接続できない場合にIPv6へ切り替えて接続する、という場面が生

注1）http://www.intec.co.jp/technology/technology/paperpatent/pdf/ipv4_app_check_list_20110831.pdf

注2）https://www.nic.ad.jp/ja/materials/iw/2012/proceedings/t7/t7-sekine.pdf

注3）http://www.v6pc.jp/jp/entry/wg/2012/12/ipv610.phtml

COLUMN IPアドレスをハードコーディングしていませんか?

たまにIPアドレスをハードコーディングしているプログラムを見かけます。IPv6対応の話以前に、IPアドレスのハードコーディングは厳禁です。IPアドレスをハードコーディングしていると、IPアドレスが変更された場合に、ソースコードの修正が必要となります。ソースコードを修正すると再度、ビルド、テスト、デプロイが必要となります。かなりの手間ですよね？　このアプリケーションを保守する技術者の身になれば、IPアドレス変更の話が挙がろうものなら、猛烈に反対するでしょう。

でも、ちょっと待ってください。IPアドレスはネットワーク管理者が管理する資源です。ネットワーク最適化を重視してアドレスを割り振っていくものです。ネットワーク構成の変更に伴い、IPアドレスが変更されるケースは少なくありません。アプリケーション開発者は、IPアドレスが変更されうる資源であることに留意しましょう。

では、IPアドレスをハードコーディング厳禁と言うのであれば、どうすればいいのでしょうか？　その答えは、外部のファイルにFQDNで記述して、そのファイルを起動時に読み込むことです。FQDNを用いることでIPアドレスの決定をOSの名前解決機構に委ねることができ、アプリケーションでIPアドレスの変更を気にする必要がなくなります。また、FQDNが変更となる場合にも、ソースコードの修正が不要です。外部ファイルを書き換えて再起動し、疎通確認を取るだけで済みます。

じます。これをフォールバックと言います。このフォールバック処理を実装する場合、単純に順次処理していくと切り替えが遅くなり問題が生じる危険性があります。フォールバックが気になる場合には、Happy Eyeballs(RFC6555)のような処理も検討しましょう。

サーバプログラム

サーバプログラムのIPv6対応は大きく分けて2つの方法があります(**図1**)。1つは、IPv4用とIPv6用の2つのプロセスを起動して接続を受け付ける方法です。もう1つは、1つのプロセスでIPv4用とIPv6用に複数のソケットを生成して接続を受け付ける方法です。

2プロセス起動によるIPv6対応サーバプログラム

IPv4用、IPv6用それぞれにプロセスを起動して対応する方法です。vsftpdなどでこの方法が用いられています。プログラム本体は、従来IPv4用に実装されていたアルゴリズムをそのままに、プロセス起動時にIPv4かIPv6かいずれかを指定して動作させます。既存のIPv4用プログラムがある場合には、対応が容易です。しかし、同一プログラムを複数起動するため、複数のプロセスで同一のリソースにアクセスする場合には、アクセスが競合しないよう排他制御を考慮する必要があります。

プログラムのアルゴリズムとしては、従来IPv4用に実装されていたアルゴリズムをそのままに、アドレスを指定してソケットを生成(socket)、bind、listenして接続を待ちます。この一連の流れをIPv4とIPv6とで別のプロセスとして起動し、実行します(**図2**)。

1プロセス複数ソケットによるIPv6対応サーバプログラム

単一のプロセスでIPv4とIPv6のそれぞれのソケットを生成して接続を受け付ける方法です。前回、サーバ構築で紹介したApache HTTP ServerやPostfixなどでこの方法が用いられています。

シングルスタックの場合とはアルゴリズムが異なり、接続待受を並列化する必要があります。

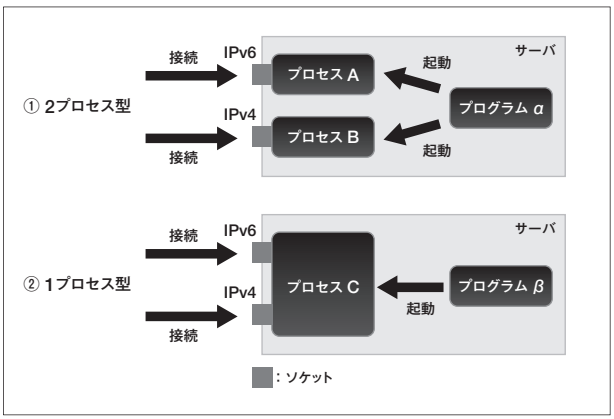
プログラムのアルゴリズムを**図3**に示します。まず、接続を受け付ける複数のアドレスをリスト化したアドレスリストを指定します。ソケット生成(socket)、bind、listenの一連の接続待ち開始処理を、アドレスリストの各アドレスに対して実施します。接続待ち状態の複数のソケットについて、selectで待ち合わせし、それらのソケットへの接続が検出されしだい、該当ソケットにacceptしてアプリケーションデータを読み込み、応答を返します。

クライアントプログラム

クライアントプログラムでは、接続先がIPv4とIPv6の複数のアドレスを持つことを考慮する必要があります。従来のIPv4のみのクライアントプログラムでは、接続先アドレスが1つの前提で、そのアドレスに接続し、クエリー送信／応答を行っていたかもしれませんが、これからはそうはいきません。

接続先アドレスを複数取得してアドレスリストに格納し、アドレスリストの順に接続を試み、接続が確立したものとクエリー送信／応答を行います。

▼図1　サーバプログラムのIPv6対応



データとしてIPアドレスを扱う個所をIPv6に対応させる

アプリケーションによっては、IPアドレスをデータとして扱うことがあります。たとえば、接続履歴として残す、アクセスの制御のために用いるなどがあります。連載3回目[注4]でも触れましたが、IPv6アドレスはIPv4アドレスと大きく異なります。そのためIPアドレスを扱う際には注意が必要です。これまで4つの3桁テキストボックスを並べていた入力フォームは使えなくなります。

それではあらためてアドレスの違いを見ていきましょう。**表1**にIPv4アドレスとIPv6アドレスの比較を示します。IPv6アドレスの文字列表記は、16進数を用いること、区切り文字に「**:**」(コロン)を用いること、文字列長39文字以内となることがIPv4アドレスと大きく異なります。この違いが、IPアドレスをデータとして扱うDBへの格納やログ出力に影響します。

DBへの格納

では、DBへ格納するケースを考えてみましょう。IPv4アドレスを格納する場合は、どうしていたでしょうか？　おそらく、1つの文字列として扱いVARCHAR(15)などとしていたでしょうか。もしくは、1オクテットずつINTEGER型としていたでしょうか。いずれの方法もIPv6に対応する場合には変更が必要です。このままではIPv6アドレスは格納できずエラーとなります。IPv6アドレスの最大文字数である39文字の文字列が格納できるようVARCHAR(39)に格納しましょう。

PostgreSQLにはネットワークアドレス型とネットワークアドレス関数が提供されています。入力値のエラー検査と専用の演算子が提供されるため、PostgreSQLを使用する際はネットワークアドレス型を使用するほうが良いでしょう[注5]。

ログ出力への影響

IPアドレスをログに出力する場合も影響があります。単純にアドレス部分の文字列長が長くなります。**図4**にApache HTTP Serverのログを

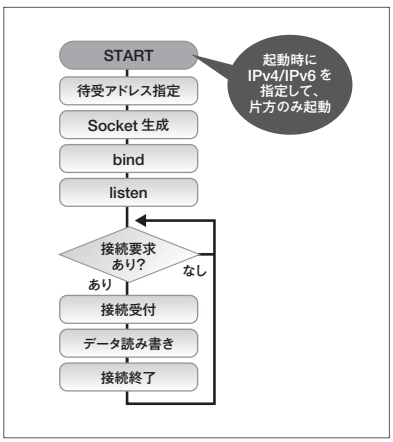
注4) 本誌2013年2月号。

注5) PostgreSQL ネットワークアドレス型(http://www.postgresql.jp/document/9.2/html/datatype-net-types.html)

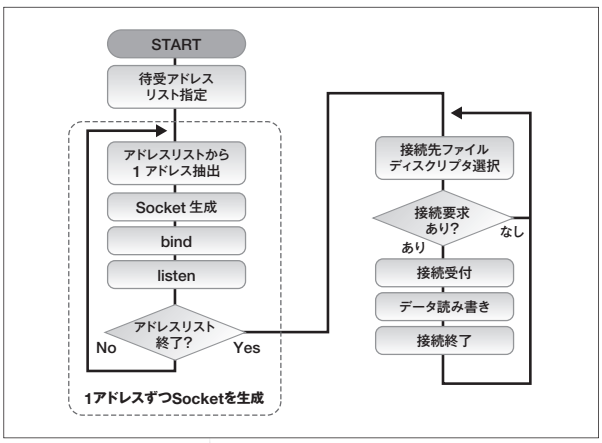
▼表1　IPv4アドレスとIPv6アドレスの比較

		IPv4アドレス	IPv6アドレス
アドレス長		32bit	128bit
文字列表記	表記法	8bitずつ区切って10進数で表記	16bitずつ区切って16進数で表記
	区切り文字	**.**(ドット)	**:**(コロン)
	文字列長	15文字以内	39文字以内

▼図2　2プロセス型のアルゴリズム



▼図3　1プロセス型のアルゴリズム



示します。①がIPv4のアクセス、②がIPv6のアクセスです。行頭のIPアドレス部分が変わっています。Apache HTTP Serverの標準ログやAWStatsなどのOSSログ解析プログラムは、たいてい問題なく処理できますが、ログ解析プログラムを自作していたり、ログ出力を自作している場合には影響の有無を確認してください。

Webフォームへの入力

Webのフォームにてアドレスを入力させる際にはとくに注意が必要です。脆弱性対策のため、一般に公開するWebサイトのフォーム入力については、入力値のバリデーション処理は不可欠と言えるでしょう。入力する値に含まれる文字が明らかになっている場合、入力値チェックの段階で入力が許される文字だけを通すことが望ましいです。そのため、アドレスとして取りうる値かどうかをチェックします。IPv4では整数と「.」だけを通せば良かったものが、IPv6ではこれに加えて「a」から「f」までの英字と「:」を通す必要があります。

プログラミング言語によっては、IPv6用のアドレス処理ライブラリがあり、IPv6アドレスの検証が可能なものもあります。たとえばPHPでは、PEARで提供されているNet_IPv6パッケージに含まれるNet_IPv6::checkIPv6()がこれにあたります。ライブラリの関数でアドレスの検証が行える場合にはライブラリを活用しましょう。

また、WebフォームへのIPv6アドレス手入力は入力ミスが起こりやすくなります。コピー&ペーストで入力できるようにするなど、入力の省力化を心がけましょう。

IPv6アドレスの検索やソート

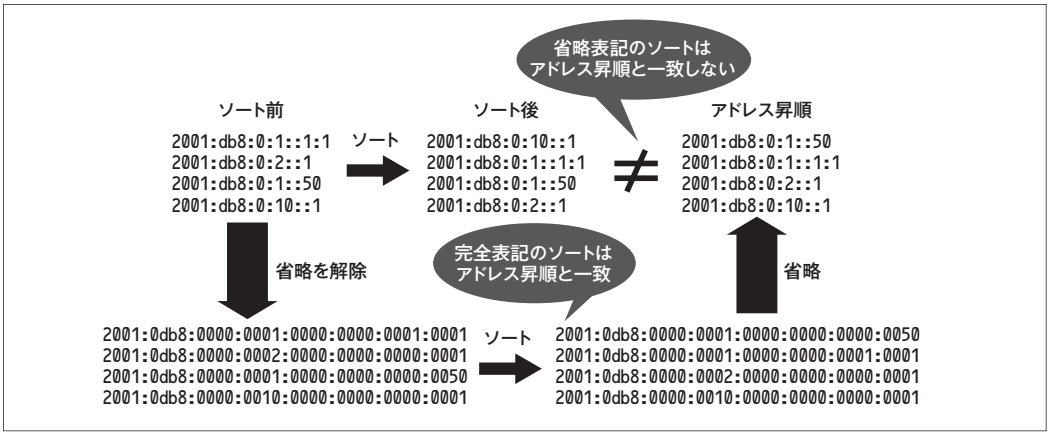
IPアドレスの検索やソートを行う場合には、省略表記に注意が必要です。完全表記で格納されたアドレスデータに対して、省略表記で検索を行っても、何もマッチしません。また省略表記されたアドレスのソートも注意が必要です。アドレス型が定義されているプログラミング言語やDBではアドレス型に変換して検索／ソートし、アドレス型が定義されていない実行環境では省略表記から完全表記に戻して検索／ソートすることが必要となります。

▼図4　access_log

```
192.0.2.10 - - [05/Jun/2013:22:11:46 +0900] "GET / HTTP/1.1" 200 123 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:17.0) Gecko/20130514 Firefox/17.0" ←①

2001:db8:0:1::10 - - [05/Jun/2013:22:12:35 +0900] "GET / HTTP/1.1" 200 143 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:17.0) Gecko/20130514 Firefox/17.0" ←②
```

▼図5　IPv6アドレスにおけるソートの注意点



この例を**図5**に示します。2001:db8:0:1::1:1、2001:db8:0:2::1、2001:db8:0:1::50、2001:db8:0:10::1の4つのアドレスをアドレス昇順に並べようとして省略表記のまま文字列でソートすると、期待に反した結果になります。省略表記を解除してソートすることにより、アドレス昇順にすることができます。この例のサンプルプログラム(PHP)を**リスト1**に示します。

終わりに

今回はアプリケーション開発におけるIPv6対応について説明しました。次回は、IPv6におけるセキュリティについて取り上げます。SD

▼リスト1　IPv6アドレスのソートの例

```
<?php
    $tmp_arr = array("2001:db8:0:1::1:1", "2001:db8:0:2::1", "2001:db8:0:1::50",
"2001:db8:0:10::1");

    require_once "Net/IPv6.php";    // Net_IPv6パッケージを呼び出し
    foreach ($tmp_arr as $short_addr){
        if (Net_IPv6::checkIPv6($short_addr)){    //IPv6アドレスであることをチェック
            $full_arr[] = Net_IPv6::uncompress($short_addr, true);    //省略表記を解除
        }
    }
    echo "before: ";
    print_r($tmp_arr);    //ソート前配列の出力

    sort($full_arr);    //配列をソート

    foreach ($full_arr as $full_addr){
        if (Net_IPv6::checkIPv6($full_addr)){    //IPv6アドレスであることをチェック
            $sorted_arr[] = Net_IPv6::compress($full_addr);    //省略表記化
        }
    }
    echo "<BR>after: ";
    print_r($sorted_arr);    //ソート後配列の出力
?>
```

COLUMN　IPアドレスでユーザを識別していませんか?

接続元IPアドレスでユーザやセッションを識別しているという噂を聞くことがあります。これも避けるべきです。

本質的にIPアドレスはネットワーク資源であり、ネットワークに接続するネットワークインターフェースに対して割り当てるものです。割り当てられたIPアドレスは、ネットワーク管理の都合で変更されることがあり、恒久的なものではありません。さらに、ユーザとネットワークインターフェースは一意に紐付かないため、IPアドレスではユーザを一意に識別することはできません。Webでは、ユーザの一連のアクセスを一貫させるものがセッションであり、ブラウザのアクセスに対してセッションを発行します。そのため、セッションにはユーザが紐付きます。それゆえに、同様にIPアドレスではセッションを一意に識別することはできません。

Wi-Fiの普及により1台の端末で複数のインターフェースを持ち、複数のIPアドレスを使用することが当たり前になっています。また、それとは別に、IPv4ではIPマスカレードを用いて複数の端末に同一のIPアドレスを割り振ることが当たり前になっています。今後、IPv4アドレス枯渇、IPv6化により、この流れが加速します。

接続元IPアドレスでユーザやセッションを識別してはいけません。本質を理解して使い分けましょう。

August 2013

NO.22

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp

夏だ! まつりだ! LLまつり!

今回は、8月24日に行われるLightweight Languageイベント（LLイベント）「LLまつり」の内容などをお伝えします。

LLイベントとは

LLイベントはおもにプログラマ向けのカンファレンスです。初回は2003年で、このときはPerl、Ruby、PHP、Pythonのコミュニティが参加しましたが、その後はほかの言語コミュニティやプログラミング好きの集うイベントとして定着し、今年で11回目となります。セッション内容も、初期は各言語を比較するようなものが多かったのですが、もともとLLを「プログラミングをより手軽にし、高い生産性でソフトウェアを創り出せる言語」という意味合いでとらえていることもあり、プログラマに対する有用な情報提供やプログラミングの楽しみを分かち合うような内容にシフトしています。LLイベントのおもな特徴として次のようなものが挙げられます。

- **複数コミュニティによる運営**
  LLイベントの実行委員会には多くのコニュニティから人が集まっています。このような運営形態を採る場合、特定の言語コミュニティが主導権を握ると不公平感が出る恐れがありますが、LLイベントの場合はjusという中立的な組織が入っているおかげでほど良いバランス感覚が保たれています
- **毎年イベントタイトルを付ける**
  各年のテーマやキーワードとなる単語をタイトルに含めることでイベント全体に統一感を持たせる効果を生んでいます
- **チケット制の導入**
  初年度に約150人の参加費の授受を会場で行ったところ受付が大きく遅延したので、翌年からはチケット会社への委託を行いました。これはIT系イベントではパイオニア的な試みで、ほかのイベントにも普及していきました
- **会期は1日だが長時間**
  かつては会期を2日間にしたこともありますが、そうすると週末をすべてイベントに費やしてしまって休めません。だからと言って1日の日中だけでは物足りない感じがします。そこで会期を1日とする代わりに朝から夜8時ぐらいまでセッションを行い、参加者に満腹感（?）をもたせています
- **見た目の楽しさを追求**
  プログラムはホットな技術テーマを取り扱い、発表者も著名な方が数多く登場しますが、それだけではなくセッションを見て楽しんでもらうための工夫を凝らしています。これまでに、ライブハウスの利用、プロレスのリング上でのセッション、映像を使ったオープニングやエンディング、ライトニングトークの対抗トーナメントなどを実施してきました
- **プログラミング関連書籍の紹介**
  IT系出版社の協力を得て書籍の販売や展示を行っています。展示に使用した書籍は参加者にプレゼントしており、イベントのエンディングで行われるボール投げによる抽選会は名物の1つです
- **参加者への無線LAN提供**
  ネットワーク構築チームを組織し、機器ベンダの協力を得て参加者への無線LAN提供を行ってい

ます。ほかのカンファレンスではなかなか成し得ないこととして高い評価を得ています

- **金銭スポンサーなし**
  開催にかかる経費は参加者が支払う参加費だけで賄われています。企業からの協賛金は景気や業績などに左右されやすいのに対して、参加費は参加者がいる限り集まるので、イベントを長く続けるためにはこのほうが良いという考えによるものです

LLまつりの見どころ

■プログラム構成

LLイベントはこれまで常に1トラックでプログラムを組んできました。しかし、もっと幅広い層に参加してもらいたいとの願いから、今年はタイプの異なる3つのトラックを用意します。

- **プレゼンテーション**
  今年初の試みとして、約20分のプレゼンテーションを十数人の方に行っていただきます
- **パネルディスカッション**
  例年行っている形態のセッションで、注目の技術を中心に数本程度を企画しています
- **チュートリアル**
  これも今年初開催で、おもにこれからLLの世界に入る人たちを対象に、各言語コミュニティによる教育的なセッションを何本か組む予定です
- **ライトニングトーク**
  毎年恒例のセッションで、上記3トラックとは別にイベントの締めくくりとして実施します

■チケット販売方法の改革

今年はチケット販売にもいくつかの新アイデアを導入します。

- **ソーシャルIDで買えるチケットシステム**
  昨年まではおもにローソンチケットを利用していましたが、今のITエンジニアにはソーシャルIDで購入できるシステムのほうが馴染み深いので、今年はPeaTiXでチケットを販売します
- **ペアチケット**
  初の試みとして、1枚で2人入場できるチケットを発売します。カップルだけでなく友人同士や、先輩が後輩を連れて行くのにも使ってください
- **Tシャツ券**
  LLイベントでは毎年Tシャツを製作しており、参加者もTシャツ付きチケットを買えば入手できますが、今年はTシャツを分離してチケット化しました。しかもサイズ別にチケットを売ります

■平面型ホールの使用

今年は例年利用してきた劇場型のホールが空いていなかったため、展示会でよく使われる平面型のホールを利用することになりました。我々はこれも新たなチャレンジととらえ、自由にレイアウトできる空間の良さを活かした演出を構想しています。

「LLまつり」に込めた想い

今年のイベント名称を考えるにあたり、スタッフの間で浮かんだイメージは、平面型のホールでさまざまな企画が同時進行する様子でした。そこから浮かび上がるカオス感のようなものを「祭」にたとえ、今年のイベント名を「LLまつり」としました。原稿執筆時点で決まっている概要を**表1**に掲載します。

LLイベントも新たな10年をスタートさせるにあたり、今年は例年以上に多くの挑戦を行っています。これらの中から今後のLLイベントの柱になるようなものが見つかればと考えていますし、読者のみなさんにもぜひ我々の挑戦を見てもらいたいです。8月24日、すみだ産業会館でお会いしましょう! SD

▼表1　開催概要

イベント名	Lightweight Language Matsuri (通称：LLまつり)
日時	2013年8月24日(土) 10：30～20：00(予定)
場所	すみだ産業会館
Webサイト	http://ll.jus.or.jp/2013/
公式タグ	llmatsuri
Twitter	@lljapan
facebook	lljapan
問い合わせ先	info2013@ll.jus.or.jp

Hack For Japan
エンジニアだからこそできる復興への一歩

● Hack For Japanスタッフ
関 治之 Hal Seki
Twitter @hal_sk

第20回 世界最大規模のハッカソン、International Space Apps Challenge 2013

社会的課題をテクノロジで解決するためのコミュニティ、Hack For Japanの活動をレポートする本連載ですが、今回は、NASAやJAXAなどのAPIやデータを使った国際的なハッカソン、International Space Apps Challengeの模様をレポートさせていただきます。

44ヵ国、83都市から9,000人以上が参加

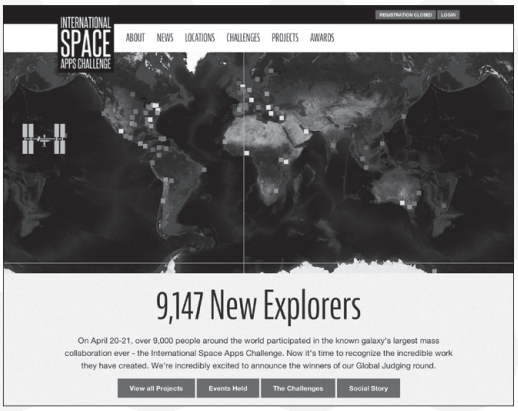
このハッカソンは去年も開催されており、本誌の2012年7月号でも報告レポートを掲載させていただいたのですが、昨年のイベントが世界的に評価が高かったため今年も開催することになりました。昨年の2,083人の参加者に対して、今年はなんと9,000人以上が参加し、世界最大規模のハッカソンとなりました。筆者は昨年に引き続き今年も東京地域のオーガナイザをさせていただいたのですが、昨年同様、東京大学空間情報科学研究センター(CSIS)やJAXAなどの協力を得て、4月20日、21日の2日間+アイディアソンを実施することができました。日本でも昨年を上回る110名以上の参加者が素晴らしいソリューションを生み出しました。

オープンデータ、オープンテクノロジ

International Space Apps Challenge(以下、ISAC)は、オープンな宇宙観測データと市民のニーズを元に、オープンなテクノロジを使って課題解決を行おうというイベントです(**図1**)。NASAおよびJAXAを始めとした、世界各地の政府機関、市民団体の協力により、全大陸および宇宙空間の開発者コミュニティを結び、航空・宇宙データを活用したオープン・ソリューションを開発することを目指しました。

NASAが事前にWebサイトで公開している50のチャレンジに対し、ハッカソンで集まったメンバーが独自のアイディアを元にソリューションを考え、

◆図1 International Space Apps Challengeのオフィシャルページ(http://spaceappschallenge.org/)



プロジェクト化します。そして、それに賛同したメンバーが集まってアプリケーションやサービスを開発します。ソリューションはソフトウェア開発、市民科学、ハードウェア開発、データ可視化の4つの分野に分かれており、それらの中からローカルウィナーを選出します。選ばれたウィナーがグローバルコンペティションに参加し、そこから各分野ごとのウィナーが決まるしくみです。

科学未来館でアイディアソン

今年は、ハッカソンに先立つアイディアソンを、お台場にある科学未来館で行うことができました。しかも、未来館のシンボルであるジオ・コスモスの下という絶好のロケーション(**写真1**)。70名以上が参加し、たくさんのユニークアイディアが生まれました。視覚会議というプロセスを提供してくれたアイデア創発コミュニティ推進機構の矢吹博一さんのファシリテーションのもと、100以上生まれたアイ

ディアの中から参加者が投票した上位、および「どうしてもこれをやりたい！」というアイディアについてチーム分けを行い、そのチームでハッカソン当日に望みました。

東京地域は18個のソリューションが完成

昨年同様、駒場東大リサーチキャンパスで行った東京地域のハッカソンでは、当日生まれたアイディアなども含めて計18チームが誕生し、各チームは2日間寝る間も惜しんで開発を続けました。1日目の深夜1時には、残っていたメンバーでAstronaut Hangoutに参加し、宇宙飛行士の方々と各会場をGoogle Hangout でつないで宇宙飛行士へ直接いろいろな質問をするといったことも行いました。

2日目の開発時間が終了した後は、デンソーアイティーラボラトリさんの会場にて発表および審査会です。今回は審査員も、前回を上回る12名＋特別審査員3名と前回の5名から大幅に増えています。審査員となったのは東京大学の柴崎亮介教授、法政大学の岡部佳世氏、慶應義塾大学の神武直彦准教授、JAXA広報部長の寺田弘慈氏、LINE株式会社の谷口正人氏、デイリーポータルZの林雄司氏、日本科学未来館の今泉真緒氏、デンソーアイティーラボラトリ代表取締役の平林裕司氏、ASTRAXの山崎大地氏、無重力スペシャリストのくさまひろゆき氏ら。

◆写真1　アイディアソンでの記念写真

各チームのプレゼンも実際のソリューションもどれも素晴らしい作品ばかりで、審査員も入賞作品を決めるのにかなり悩んでおられました。今回はその中から入賞作品をご紹介します。ここで紹介できなかったプロジェクトも含めた全プロジェクトのリストは、公式サイト[注1]から確認いただけますので、ご興味のある方はぜひご確認ください。

1位▶Personal Cosmos

アイディアソンのときに見た科学未来館のジオ・コスモスに着想を得て、さまざまなデータを投影できる地球型のパーソナルディスプレイを作りたいという思いでスタートしたプロジェクトです（**写真2**）。前回のハッカソンでもJAXA賞を取った、クウジット株式会社の湯村翼さんがリーダーとなり進められ（**写真3**）、2日間で、身近で手に入る材料のみを使っ

◆写真2　Personal Cosmosの写真

◆写真3　開発チーム（プロジェクトサイト：http://spaceappschallenge.org/project/personal-geo-cosmos/）

注1）http://tokyo.spaceappschallenge.org/results

て実際に動作するものを作りあげてしまいました。

プロジェクションマッピング用のソフトウェアなども作成し、着想、アプローチ、時間配分、プレゼンテーションすべてにおいてハイクオリティな作品だったと思います。昨年の1位もハードウェア系のプロジェクトでしたが、今年もハードウェアが1位を取ったことはとても興味深かったです。

2位▶Cloudless Spots for Solar Power Generation

東京大学空間情報科学研究センター特任研究員の、大平亘さんによるプロジェクト(**写真4**)。膨大な衛星データを解析して得られた日本上空の雲の割合から、ソーラーパネルの設置に優位な場所を把握できるというものです(**図2**)。

NASAが公開している地球観測衛星データ(MODIS)にある日本の過去12年分の雲の情報を元に、だいたい1キロ四方ごとの晴天率をヒートマップ状に表示する機能と、太陽光発電システムの初期費用回収期間のシミュレーション機能を備えています。学者やハイスペックなエンジニアのチームで、技術的にもかなり高度なアプリケーションでした。

3位▶Dear my SPACE DEBRISおよびMarsface Project(宇宙文明発見チーム)

3位は同点で2つのチームが選ばれました。Dear my Space DEBRISプロジェクトは「気に入ったデブリ(宇宙ゴミ)にタグ付けし、モニタリング。デブリの一生を通して宇宙のライフサイクルを感じる」というもの。宇宙上には50万を超えるデブリが存在しており、宇宙研究上も大変深刻な問題となっています。普段の生活をしていると忘れてしまいがちで実感がわかない問題を、タグ付けすることで身近に感じることを目指した点が評価されました。

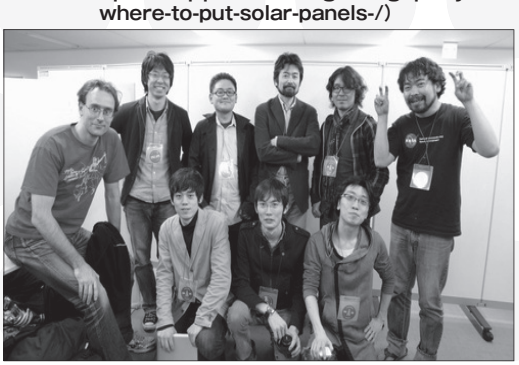
Marsface Projectは月および火星表面の画像から、顔認識アルゴリズムを使って宇宙文明の痕跡(人の顔)を見つけようというプロジェクトです。産業技術総合研究所の栗原一貴氏による、Google MapsとOpenCVを使って人の顔を見つけるGeofaceプロジェクトの宇宙版といった位置づけです。栗原さんの支援も取り付け、なんと120億ピクセルに対し、3種類の認識手法を使っていくつかの顔画像を見つけ出しました。チームラボの高須正和さんによるプレゼンの破壊力により、会場が笑いの渦となっていました。

特別賞も

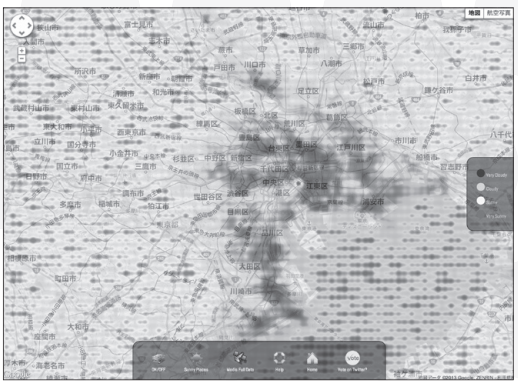
このほか特別賞として用意されたカパル賞として「Making the New Constellations Sets：新しい星座を作る」、Samurai Fabヨコハマものづくり工房賞として「火星猫じゃらし」と「Linking Space and Health」、AWSアーキテクト賞として「Linking Space and Health」、「Lunar Travel Agency」および「VOY∀GER」がASTRAX賞、「Star Music」がYahoo!賞、「Dear my SPACE DEBRIS」がJAXA賞を獲得しました。

審査員の方々のうち、昨年も参加された東大の柴

◆写真4　開発チーム(プロジェクトサイト：http://spaceappschallenge.org/project/where-to-put-solar-panels-/)



◆図2　アプリケーションのスクリーンショット



◆写真5　ハッカソン後の集合写真

◆写真6　今年も1日目の夕食で登場したスペースシャトル寿司

崎先生からは、多くのチームが昨年度よりもプレゼンテーションやソリューションの質が格段にアップしていたという評価をいただいています。発表を見に来られた方々も、たった2日間（実質的には1日半）で作られたとは思えないほどの成果物の品質の高さに驚かれていました。前回に比べるとアイディアソン、ハッカソン共に女性の姿も多く、デザイナの参加も多かったため、アウトプットも洗練されていたように思います。

グローバルアワードの結果

日本からは、1位のPersonal CosmosとCloudless Spotがグローバルアワードに進みました。5つあるグローバルアワードの特別賞は逃したものの、Personal Cosmosは「Best Use of Hardware」部門、Cloudless Spotsは「Galactic Impact」部門でそれぞれ「Honorable Mentions（佳作）」として選出されました。この「Honorable Mentions」に選ばれたのはグローバルアワードに参加した134プロジェクト中29プロジェクトのみで、かなり高い評価だったと言えるでしょう。昨年は日本から選出された2チームはグローバルアワードでは選外でしたので、今年はかなり東京地域は成長したのではないかと思っています。

何より感動的だったのは、ローカルアワードの決定後1週間ほど用意されていた、グローバルアワードに向けての調整期間において、チームメンバーではない他の参加メンバーが上位選出2チームをサポートしていたことでした。ビデオ撮影、英訳、サイトデザイン、写真撮影など、まさにオールスターなチームができあがり、日本代表チームの後押しをしたのです。

NASAより、今回のミッションレポートが公開されているので[注2]、世界中でどのようなことが行われていたか興味がある方は、ぜひご一読ください。

オープンデータ成功の鍵はコミュニティにあり

昨年、今年とこのハッカソンにかかわってきて思うのは、オープンデータ活動の本質は、データを使ったソリューションを生むためのコミュニティ作りなのではないかということです。日本でも政府のデータをオープンデータ化する動きが出てきていますが、データをただ公開していくだけでなく、エンジニアのコミュニティを作り、そこでさまざまなソリューションの案やプロトタイプが生まれていくような体制をいかに作るかが大事なのではないかと思っています。データだけでは便利なアプリケーションにはならず、課題解決のためにどのようにデータを使うか、どのようなソリューションを提供するために使うのか。そういった部分をドライブするようなしくみとしてのハッカソン実施というのは、コミュニティ形成上も良い効果が得られるのではないかと思っています。

Hack For Japanでは、引き続き社会的課題を解決するためのハッカソンを実施していますので、ご興味ある方はぜひ「http://hack4.jp/」までアクセスいただければと思います。SD

注2）https://speakerdeck.com/nasa/2013-international-space-apps-challenge-mission-report

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。

Event オープンソースカンファレンス2013 Kansai@Kyoto開催

8月2日、3日の2日間、京都リサーチパーク（京都市下京区）において、「オープンソースカンファレンス2013 Kansai@Kyoto」が開催される。

オープンソースカンファレンス（OSC）は、全国各地で月一度の頻度で開かれているオープンソース関連技術の開発者や企業が一堂に会する見本市。

京都での開催は今回で7回目。約70の企業／団体による展示があるほか、Webサービスなどのインフラ技術から学生やコミュニティの発表まで約70種の幅広いテーマの技術セミナーを無料で受けられる。

今回は、広い展示スペースを実行委員会のメンバーが一緒にまわりながら案内する「展示ツアー」を1時間に一度ずつと積極的に実施し、初心者にもオープンソースの魅力を紹介する。

さらに、京都地区のOSC実行委員がローカル企画として、次の3つを柱とした「オープンソース入門塾」のブースを出す。

①オープンソースの楽しさを紹介するための、LEGOロボット、Arduinoマイコン、Raspberry Piなどのデモ
②学生による、学生のための「プログラミング教育の推進活動」
③オープンソースの底力を知ることができる、ビジネス活用例のデモ

▼開催概要

日　程	・8月2日（金） 11：00～17：00（セミナー、展示ともに17：00まで） ・8月3日（土） 10：00～17：00（展示は16：00まで）
会　場	京都リサーチパーク（KRP） 京都市下京区五条七本松通
費　用	無料
内　容	オープンソースに関する最新情報の提供 ・展示：オープンソースコミュニティ、企業、団体による展示 ・セミナー：オープンソースの最新情報を提供
セミナー申込方法	下記のWebサイトより事前登録

CONTACT OSC2013 Kansai@Kyoto公式Web
URL http://www.ospn.jp/osc2013-kyoto

Event ユニバーサル・シェル・プログラミング研究所、トークライブイベント「TechLION vol.13」を開催

㈲ユニバーサル・シェル・プログラミング研究所は、UNIX OSやWeb系の技術をベースに活動するITエンジニアをターゲットにしたライブ、および交流の場となるコミュニティイベント「TechLION vol.13」を、7月23日にSuperDeluxe（東京・西麻布）にて開催する。

TechLIONは、IT文化の振興と、IT文化の楽しさを広く伝えエンジニア同士の連帯を図ることを目的とするトークイベント。登壇者と参加者がともにお酒を酌み交わしながら、IT業界の歴史、舞台裏、問題などさまざまな話題を語り合う。今回のプログラムは次のとおり。

- 第一部：獅子王たちの夕べ
  ゲストは、さくらインターネット社長の田中邦裕氏。インターネットのインフラを支える立場から、エンジニアとしての生き方と、起業家としての想いを伝える
- 第二部：ジャングルバス.com
  ゲストは、サイバーエージェントの紫竹佑騎氏と、ミクシィの田中和紀氏。これからの日本のWeb/IT業界を引っ張っていくであろう若手エンジニア2名をゲストに迎えて、エンジニアとして目指すこと、やりたいこと、過去のWebと未来のWebなどについて、若者視点で語る

▼開催概要

イベント名	TechLION vol.13
日　時	2013年7月23日（火）19:00開場、19:30開演、22:30終了予定
会　場	SuperDeluxe 東京都港区西麻布3-1-25 B1F　Tel:03-5412-0515 https://www.super-deluxe.com/map/
出　演	田中邦裕（さくらインターネット）、田中和紀（ミクシィ）、紫竹佑騎（サイバーエージェント）
M　C	法林浩之（日本UNIXユーザ会）、馮富久（技術評論社）
料　金	前売2,700円、当日3,200円（1ドリンク付き）
申込方法	次のWebサイトにて参加登録 http://techlion.doorkeeper.jp/events/3701
問合せ先	TechLION事務局（E-mail：staff@techlion.jp）

CONTACT TechLION公式サイト
URL http://techlion.jp/vol13

Software

Citrix Systems、XenServer 6.2をフルオープンソースで提供

米Citrix Systems社は6月25日、無償で利用できる全機能装備のフルオープンソースソフトウェア「XenServer 6.2」の提供と、新たなコミュニティポータルサイト「XenServer.org」（英語）の公開を発表した。

XenServerは、仮想化技術Xenをベースにしたサーバ仮想化プラットフォーム。これまでは一部プロプライエタリな技術が含まれていたが、6.2よりすべてがオープンソースソフトウェアとして提供される。

6.2では、プライベートクラウドとパブリッククラウドに対応するパフォーマンス、スケーラビリティ、集約率が改善されているほか、デスクトップおよびサーバの仮想化向けの拡張機能も追加されている。

無償のオープンソースXenServer 6.2はXenServer.org（http://www.xenserver.org/）から入手できる。

また、Citrixではサポート完備の商用版XenServer 6.2も提供しており、こちらはCitrixWebサイト（http://www.citrix.com/downloads/xenserver.html）からダウンロードできる。商用版ではパッチの適用、24時間365日のワールドワイドサポートなどのサービスを受けられる。

CONTACT シトリックス・システムズ・ジャパン㈱
URL http://www.citrix.co.jp

Software

ミラクル・リナックス、Zabbix/Nagiosの監視統合ビューア「Hatohol」を開発、OSSとして公開

ミラクル・リナックス㈱は、オープンソース統合監視ソフトウェア「Zabbix」および「Nagios」を利用した複数の監視サーバを一括監視する、表示専用のソフトウェア「Hatohol」を独自に開発し、6月27日よりオープンソースソフトウェアとして公開した。これに合わせて、コミュニティ「Project Hatohol」を設立し、Hatoholの開発および運営を推進することを発表した。

Hatoholは、複数のZabbix/Nagiosサーバからデータを取得し、監視サーバをまたいだ環境の一括監視を可能にする。監視対象の規模に応じて柔軟にZabbix/Nagiosサーバを拡張できるため、最初から想定される最大監視規模の監視サーバを用意せず、スモールスタートで監視を始められる。Hatoholは次のWebページからダウンロードできる。

https://github.com/project-hatohol/hatohol

今後、商用ソフトウェアを含むZabbix/Nagios以外の監視サーバの統合などの機能強化も計画されている。

CONTACT ミラクル・リナックス㈱
URL http://www.miraclelinux.com/jp

Service

アシスト、LibreOfficeのFAQや技術資料を利用できる「FAQ検索サービス」を提供開始

㈱アシストはオープンソースのオフィスソフト「LibreOffice」のFAQ（よくある質問と回答）および技術資料（ガイドブック）を利用できる「FAQ検索サービス」の提供を7月22日より開始する。

同社ではこれまで、LibreOfficeのマイグレーション支援サービスや各種研修サービスに加え、LibreOfficeに関する問い合わせに対応するヘルプデスクサービスを提供してきた。ユーザからは「同社のサポートセンターが持っているFAQや技術資料を使って、操作やトラブルに関する問い合わせに対応したい」といった要望が多いことから、FAQおよび技術資料を整備し、使いやすい検索システムを構築した。

サポートセンターの検索システムに、新たにサジェスト機能（検索窓に入力中の検索ワードから候補ワードを表示）を導入し、FAQのヒット率の向上を実現しているほか、利用者がより直感的に理解できるように、回答内容や技術資料に画像や動画を使用している。

価格は、1年間の利用で125,000円～（PC250台まで）となっている。

CONTACT ㈱アシスト
URL http://www.ashisuto.co.jp

Hardware

ぷらっとホーム、Syslog/DHCP/DNSなど用途を絞った小型アプライアンスを発表

簡単にログが活用できるSyslogアプライアンス

ぷらっとホーム㈱は、ログの収集や確認を手間なく導入でき、ログ活用のコストを大幅に削減する小型アプライアンス「EasyBlocks Syslogモデル」を発売した。

本アプライアンスは、従来のEasyBlocksネットワークコア統合型モデルにて選択によって利用可能であったSyslogサーバ機能に特化した製品。機能を小規模向けへと限定したことで、統合型のEasyBlocks Enterpriseに比べて導入しやすい価格体系での利用が可能となった。また一方で、ユーザからの要望が多かったログ領域の大容量化や、閲覧、抽出などを行うためのインターフェースおよびツールの強化を行っている。

▼ラインナップ

品名	価格（税込）
EasyBlocks Syslogモデル 120GB	オープン
EasyBlocks Syslogモデル 240GB	オープン
EasyBlocks Syslogモデル 480GB	オープン

DHCPやDNSなど小型アプライアンス6モデル

また同社は、EasyBlocksシリーズにDHCPやDNSなどのネットワークコアサービスを手間なく低コストで開始できる小型アプライアンス6モデルを新たに追加した。Syslogアプライアンスと同じく、EasyBlocksネットワークコア統合型モデルにて、選択によって利用可能であった各機能に特化した。

今回発表の製品（Syslogモデル含む）は、いずれも基本的な設定は実施済みであるため、エンジニアのいない拠点でも、ケーブル接続だけで導入できる。

▼ラインナップ

品名	価格（税込）
EasyBlocks DHCPモデル	オープン
EasyBlocks DNSモデル	オープン
EasyBlocks NTPモデル	オープン
EasyBlocks RADIUSモデル	オープン
EasyBlocks 監視管理モデル	オープン
EasyBlocks Webキャッシング向けProxyモデル 小規模版	オープン

▲Syslogモデル

▲Webキャッシング向け Proxyモデル 小規模版

CONTACT ぷらっとホーム㈱
URL http://www.plathome.co.jp

Service

at+link、専用サーバサービスにおいて、ハードウェア情報の監視サービスを提供開始

㈱リンクと㈱エーティーワークスが共同で展開しているホスティングサービスat+linkは、「at+link専用サーバサービス」において、ハードウェアの情報を定期的に記録し、その情報の変化を監視することで、異常な状態を事前に検知し障害発生を未然に防ぐサービス「ハードウェアモニタリングシステム」を、6月10日運用開始分のサーバから、標準サービスとして提供を開始した。

当サービスは、運用開始時からCPUの温度、システム（M/B）温度、CPUのファン回転数、電源電圧といった情報を定期的に記録し、設定した閾値に基づいて監視を実施する。5分間隔で監視を行い、あらかじめ設定した閾値を超えた場合にアラートが通知される。当サービスのサポートがそのアラート内容を検討し、通達が必要と判断した場合にユーザへ通知される。

ハードウェアモニタリングシステムを搭載したサーバを利用することで、通常時と異なる事象が起きた場合には、当サービスサポートから対策に関する提案を受けられ、障害を未然に防ぐことが期待できる。

CONTACT at+link
URL http://www.at-link.ad.jp

Interview

オープン・ネットワーキング・ファンデーション（ONF）、事務局長 ダン・ピット氏に聞く SDN/OpenFlowの今後

2013年6月12日～14日のInteropの開催に合わせ、ONF事務局長ダン・ピット氏が来日した。

本誌昨年11月号のSDN/OpenFlow特集はおかげさまで完売し、日本でも多くのネットワークエンジニアがネットワークの仮想化に注目していることがわかった。当日開催されるセミナーの合間にインタビュー取材の時間をいただく機会を得た。

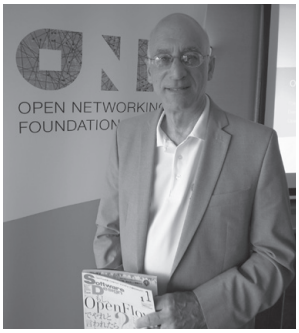
――昨年からの大きな変化は何でしょうか。

仮想マシンがサーバにインストールされている数や密度が上がってきています。ソフトウェアベースでのクラウドサービスの成長が目覚ましくなっています。それに伴いSDNへの関心も増えています。

――ONFを中心としたエコシステムが他のオープンソースプロダクトと比べてユーザへのアプローチが少し消極的な印象を受けるのはなぜでしょうか。

オープンソースのエコシステムの性格を決めるものは、大きく分けて2グループあります。ベンダー、もしくはユーザのどちらかが強くなるというものです。多くのオープンソースプロダクトは、ユーザのほうが強い場合が多いです。ONFの性格上どちらかに力を入れるという組織ではありませんし、中立的なものです。しかし、ハードウェアの実装が必要なので実際のところベンダーの協力が必要です。そのためベンダー寄りに見えるのでしょう。実際のところONF会員社数は昨年よりも増え、99社になりました。ゴールドマン・サックス社もボードメンバーです。

■ダン・ピット（Dan Pitt）
オープン・ネットワーキング・ファンデーション事務局長。2011年からONFの事務局長を務める。IBM Networking Sytems社で20年間、ネットワークアーキテクチャと製品開発を行う。その後IBM Research Zurich、Hewlett Packard Labsを経てBay Networksの副社長に就任し現在に至る。



――SDN/OpenFlowは日本のエンジニアにとって怖い存在です。というのはネットワーク技術に加え、プログラミング能力が新たに必要だと皆さん考えているからです。これからのネットワークエンジニアやサーバエンジニア達に必要とされるスキルは何でしょうか。

アメリカのネットワークエンジニア（ネットワーキングスタッフとダン氏は表現しました）もスキルレベルは日本と同じです。実は変わりません。だれもが十分なプログラミングスキル持っているというわけではありません。普段の仕事もやっている内容も日本のエンジニアと同じなのです。ですからSDN/OpenFlowが推進されることで、根こそぎ仕事のスタイルが変化することはないと考えています。これからSDN/OpenFlowの技術全般を学んでいくのもまったく遅くありません。

CONTACT オープン・ネットワーキング・ファンデーション
URL https://www.opennetworking.org

Software

キヤノンITソリューションズ、ESET法人向けライセンス製品がAndroidに対応

キヤノンITソリューションズ㈱は、Android対応のプログラムを追加し、マルチプラットフォームを強化した総合セキュリティ製品「ESET Endpoint Protection Advanced」と、ウィルス／スパイウェア対策製品「ESET Endpoint Protection Standard」を7月1日より販売を開始した。

ESET Endpoint Protectionシリーズは、従来のWindows、Mac、Linux対応のプログラムに加え、Android対応プログラム「ESET Endpoint Security for Android」を追加したマルチプラットフォーム対応の製品。提供価格は従来のESET法人向けライセンス製品と同一。

Android対応プログラムにより、ウィルス対策、スパム対策、盗難対策、セキュリティ監査、パスワード保護などの機能が利用できる。

標準で付属するクライアント管理用プログラム「ESET Remote Administrator」を導入することで、各プラットフォームの端末を一元管理できる。クライアント端末から収集したログや設定情報の管理やレポート作成ができるほか、各種タスクの配布が可能。

CONTACT キヤノンITソリューションズ㈱
URL http://www.canon-its.co.jp

Letters from Readers

3Dプリンタで家を建てる!?

今月号の第1特集は3Dプリンタでした。ここ最近、にわかに注目を集める3Dプリンタですが、海外では3Dプリンタで本物の家を建てようという試みもあるようです。「もしや一度のプリントで家全体を建てるの?」と期待しましたが、どうやら複数の部品に分けてプリントし、あとから組み立てるようです。プリンタの大きさから考えれば、当たり前か……。

2013年6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　ちゃんとオブジェクト指向できていますか?

オブジェクト指向はわかっているようでわからない、いつまで経ってもマスターできない、ということがよくあります。そんな苦手意識を克服するために、習得の手がかりを紹介しました。

オブジェクト指向について知ってはいても、たまにこういったものを読み返すのは良いと思った。

岐阜県／辻さん

オブジェクト指向で大事なのは、ネーミングセンスです。短過ぎず、長過ぎずです。長い名前は大嫌いです。スペルミスをしていると気分が悪くなります。

東京都／オブジェクト脳192さん

「オブジェクト指向の言語をオブジェクト仕様らしく使っていない」という指摘があって、はっとしました。まさに自分のことのように思えてなりません。やはり開発に追われて基礎を疎かにしてはいけないですね。

東京都／tomato360さん


オブジェクト指向をちゃんと習得できている人はまだ少ないように見受けられます。本特集がご自身の理解度を確認するきっかけになれば幸いです。

第2特集　あなたの知らないUNIXコマンドの使い方

新人さん向けに、サーバやネットワークの管理／監視、ファイル一括処理など、実務ですぐに役立つであろうUNIX/Linux系コマンドを紹介しました。

普段は自己流なので、こういう特集はためになる。

宮城県／あかぴさん

CentOSを個人的に使い始めたので、第2章は興味深い内容でした。

埼玉県／大柴さん

他誌ではなかなか読めない内容なのかなと思いました。

神奈川県／眞　泰志さん


実務で遭遇しそうなシチュエーションを想定しながらコマンドを紹介しました。仕事でも使ってみてください。

一般記事　リアルタイム分散処理Storm

日々発生する大量のデータを即時に処理する並列分散処理のフレームワーク「Storm」について取り上げました。

まずStormを聞いたことがなかった。この記事はもう少し経ってから読むことにします。

千葉県／夏樹さん

こういった書籍で出版されていないような技術について記事にしていただけると、非常に参考になるので助かります。ただ、内容としては概要となっていたので、インストール方法も含めた特集を今後期待します。

千葉県／今井さん


Stormはビッグデータ処理の新しい技術としてこれから注目です。

一般記事　HiveでHadoopを活用してみませんか?

SQLライクにHadoopを扱うことができるHive。その導入方法と使い方を紹介しました。

興味ある分野だが、まだ難しくて手を出せていない。

埼玉県／高崎さん

Hadoopはなかなか敷居が高い。手軽に試せるようになると、ビッグデータの活用も盛んになるのではないか。

神奈川県／to4さん


GUIでHiveを活用できるWeb Hiveもありますので、Hadoopに

難しい印象を持っている方でも、今後は手を出しやすくなりそうです。

一般記事　Ubuntu 13.04 “Raring Ringtail”

4月25日にリリースされたUbuntu 13.04の新機能と今後の方向性について説明しました。

Ubuntu Touchの動向を継続的に取り上げて下さい。

京都府／藤井さん

Nexus 7やUbuntu Touchなどの情報もあって良かった。

神奈川県／Alexさん

13.04の機能よりUbuntu Touchなど今後の方向性に興味を持たれた読者が多かったのが印象的でした。

連載

「enchant～創造力を刺激する魔法～」が楽しみです。20～30年くらい前、MSX-BASICで簡単なゲームをプログラミングして雑誌に投稿していたころを思い出します。

東京都／山下さん

BASICがenchant.jsを発想するヒントになったというのは、意外な事実でしたね。

「温故知新 ITむかしばなし」のMP/Mの話がよかったです。さっそくダウンロードしていじってみたいと思います。

神奈川県／drepoxyさん

限られたリソースの中で、マルチユーザを実現するためにさまざまな工夫がなされていました。昔のOSだからといって、侮れないですね。

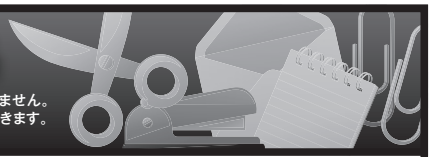
フリートーク

パソコンの電源がおかしくなったので、新しく電源を買い、ケーブルを挿し間違えたら、バックアップも含め、ハードウェアが全部壊滅しました。いい機会ですのでWindows 8のマシンを組みます。

東京都／ゴンベさん

担当なら、保存してあったデータのことで頭がいっぱいになりそうです。悲惨な状況にへこたれず新たなOSへのチャレンジ精神に燃える前向きな姿勢に頭が下がります。

エンジニアの能率を高める一品

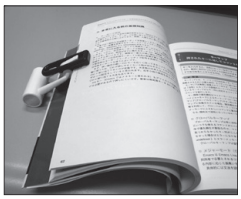


仕事の能率を高める道具は、ソフトウェアやスマートフォンのようなデジタルなものだけではありません。
このコーナーではエンジニアの能率アップ心をくすぐるアナログな文具、雑貨、小物を紹介していきます。

ブックストッパー

840円（税込）／トモエそろばん　http://www.soroban.com/

書籍のページ開いたままにする方法として2012年5月号の本コーナーで、ブックスタンドを紹介しましたが、今回は机に置いたまま使えるブックストッパーを紹介します。形状は洗濯バサミに重りが付いたような形をしています。使い方は簡単。ハサミをページの端に挟むだけ。これで重りの重みでページが閉じなくなります（**写真1**）。分厚い本の場合は、**写真2**のように挟める分量だけを挟めばOK。担当の場合、若干本文が隠れますが、ページの上に重りを載せてしまいます。このほうがしっかり開いた状態で安定します。2つ入手して左右のページに使えば、より安定します。ブックスタンドは机上で場所をとるので嫌だという人は、ブックストッパーを試してはいかがでしょう?



▲写真1

▲写真2

6月号のプレゼント当選者は、次の皆さまです

①AtermWG1800HP ..... 千葉県　近井大道様
②リュックの中身 Ver.3.0 ..... 東京都　伊藤由貴様

★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

2013年9月号
定価1,280円　176ページ

8月17日発売

［第1特集］UNIXエンジニアのたしなみ

今からはじめるsed/AWK再入門

ワンライナーVS.スクリプティング

「1行で済ませますか、それともエクセレントなスクリプト書きますか?」
とてもシンプルな処理系としてsed/AWKがあります。UNIX遣いならば誰しも使いこなせるものでしょう(?)。perlもいいけど、sed/AWKにはミニマムだけどパワフルな処理能力という魅力があります。久々にsed/AWKを使う方も、これから学んでみる人も、ぜひ習得してみてください。コンピュータのパワーを身近に使ってみましょう。

［第2特集］iOS7で変わるMac開発環境

開発するならやっぱりMacですよね?

Mac使いの開発者の手の内「デスクトップ拝見!」

［一般記事］

徹底解説 Fedora19

こなれたOSでありながら貪欲に新機能を実装し、進化するFedora19を解説します。OpenShift/OpenStack対応、MariaDB完全対応などなどディープに紹介!

※特集・記事内容は、予告なく変更される場合があります。あらかじめご容赦ください。

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7月号　第1特集 CHAPTER1
●P.23　左段9、10行目
［誤］http://ibisforest.org/index.php?%E6%A9%96%A2%B0%E5%AD%A6%E7%BF%92
［正］http://ibisforest.org/index.php?%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92

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)、「温故知新　ITむかしばなし」(第25回)は都合によりお休みいたします。

SD Staff Room

●今年のInteropはSDNブースを中心に研究したかったのだけど、ダン・ピットさんのインタビューやうち合わせで願いかなわず。それにしてもSDN/OpenFlowはもう一度特集を組んでみようと思う盛況ぶり。ネットワークがこれほど面白いものに進化したのはエンジニアにとって幸せだ。(本)

●「グリシン」が効くと聞いていろいろ試している。主に寝起きのスッキリ感と昼間の集中力を得るためだが、思ったより効果がある。アミノ酸の一種でコラーゲンに1/3ほど含まれているものらしい。製品がいろいろな会社から出ていて値段がかなり違う。人によっても効果のバラツキがある。研究の価値あり。(幕)

●所有デバイスにタブレットが多くなってきた。iPad、Nexus 7、そしてenchantMOON(この号が出るころには!)。こいつらはWi-Fi接続だからモバイルルータやテザリングの出番が多くなりそうだ。通信環境の見直しかな。いやそのまえに、どれを持ち歩くべきかで毎日悩まされそうだ。(キ)

●小さな畑を借りて家庭菜園を始めました。キュウリ、ナス、トマトの苗木がポツポツと植えてあるだけで畝もない素人丸出しの畑です。あまりの素人ぶりを哀れに思ったのか、畑に行くと先に誰かが水をやってくれていることがあります。家庭菜園の醍醐味(?)の水やりを先にやられて少し悔しい気持ちです。(よし)

●最近週末に少しだけ料理をするようになった。必要な材料はわかるのだけれども、問題はいまいち適量がわからず作り過ぎてしまうこと。なので料理本をみて作ることにしたが、ちょっと材料が残るからって入れてしまい、結局作り過ぎに。余った材料でさらにもう一品ささっと作れればかっこいいけどなぁ。(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8月号

発行日
2013年8月18日

●発行人
片岡　巌

●編集人
池本公平

●編集
金田富士男
菊池　猛
吉岡高弘

●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。