Special Feature 01
プログラミングの学び方
Special Feature 02
大きいオブジェクトから卒業
2013 04 April
Software Design
[ソフトウェア デザイン]
OSとネットワーク、
IT環境を支える
エンジニアの総合誌
2013年4月18日発行
毎月1回18日発行
通巻336号
（発刊270号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
1,280円

Special Feature 01

Why I became a programmer?

Introduction programming through the back door.

裏口からのプログラミング入門

僕（私）の言語の学び方

Special Feature 02

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？

Extra Feature

ゲーム開発環境にP2P技術を導入
スクウェア・エニックス
+Skeed
「ゲーム開発の舞台裏」

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。

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52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。

グラフ検索

Facebook が始めた「グラフ検索」とは

大手ソーシャルネットワーキングサービス（SNS）の Facebook が今年1月にスタートさせた「グラフ検索」は、SNS 上での“つながり”に注目した新しいスタイルの検索機能です。ここで言う「グラフ」とは「人や物のつながり」といった意味を表す用語で、ソーシャルネットワークの世界では「ソーシャルグラフ」などの用語とともによく知られています。

グラフと言えば、Facebook が従来より提唱してきた概念に「オープングラフ」があります。これはソーシャルグラフの考え方を拡張し、「人と人」や「人と物」、そして「人と行為」などの関係、およびそれを活用するための機能群を総称する用語です。Facebook ではオープングラフ提唱の一環として、ユーザが投稿した写真やコメント、使用した Facebook アプリ、シェアした内容、そして「いいね！」した内容などをユーザ間で簡単に共有するためのしくみを提供しています。聴いている音楽や見ている動画などを Facebook の他のユーザと共有できるサービスがありますが、これもオープングラフを利用することで実現しているものです。

グラフ検索も、このオープングラフの活用によって実現した機能です。オープングラフには Facebook ユーザのさまざまな行動の情報が膨大に蓄積されています。たとえば過去に「いいね！」した内容や、コメントで言及した内容などを調べれば、その人がどんなことに興味を持っているかがわかるでしょう。それらの情報を元にして、ユーザ同士のつながりやユーザと物のつながりなどを分析し、検索結果を導き出すというしくみなのです。

グラフ検索で探せるもの

グラフ検索と通常の Web 検索は何が違うのでしょうか。一般的な Web 検索エンジンは、キーワードの組み合わせやサイト同士のリンク情報などを頼りにして、目的の情報が掲載されている Web サイトを探し出します。それに対してグラフ検索エンジンが探すのは、「人」や「物」、「場所」などの対象そのものです。

たとえば、「友人が興味を持っていること」や「自分と同じ学校出身のユーザ」、「自分と同じ趣味や興味を持っているユーザ」などといった条件で情報を簡単に探すことができます。「自分と同じ映画を見たことがあるユーザがよく訪れる場所」のような条件の検索も考えられます。このような、通常の Web 検索エンジンとは異なる切り口での検索を行える点がグラフ検索の魅力です。

Facebook によれば、グラフ検索はオープングラフとあわせて今後も成長を続け、将来的にはオープングラフでシェアしたものはすべて検索できるようになるとのことです。

プライバシー保護の問題

ユーザに新しい体験をもたらしてくれるグラフ検索ですが、これについてプライバシー上の問題を指摘する声もあります。Facebook によれば、グラフ検索はあくまでも Facebook 上で公開状態にある情報だけを頼りに検索を行うので、公開されていないプライバシーが曝されることはないといいます。しかし問題はもう少し複雑です。

前述のように、過去に「いいね！」した対象を統計的に調べていけば、その人がどんなことに興味を持っているのかは簡単にわかってしまいます。同様の分析をもっと詳細に行えば、趣味や性格に加えて、おおよその年齢や住んでいる地域、宗教、政治的な立場など、日頃目に付く「いいね！」だけでは把握できないような情報が構築できてしまう可能性があります。グラフ検索の検索結果はこのような分析情報に基づいて導き出されるため、日々の何気ない行動に隠された趣向が公にされてしまう危険も生じるわけです。

もちろん、Facebook 上での行動情報の公開範囲をプライバシー設定で適切にコントロールすれば、このような事態は防ぐことができます。しかし一方で、厳密すぎるプライバシー保護はオープングラフの利便性を損ねることにもつながります。どの情報を守り、どの情報を公開するのか、そしてその結果としてどのようなメリットとデメリットが生じるのか。オープングラフの特性をよく理解し、自身のプライバシー情報をこれまで以上に慎重にコントロールすることが求められるでしょう。SD

グラフ検索
http://www.facebook.com/about/graphsearch

※日本でのグラフ検索の提供はまだスタートしていませんが、ベータ版に登録しておけばサービスが利用できるようになった段階で通知が送られるそうです。

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。

Software Design Contents

僕（私）の言語の学び方

裏口からのプログラミング入門

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。

Software Design Contents

第2特集 Special Feature 02

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?

著:トム・エンゲルバーグ
訳:長谷川 裕一

特別企画 Extra feature

巻頭Editorial PR Editorial PR

アラカルト A La Carte

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

［目次］

Software Design April 2013［CONTENTS］#03

Software Design Contents

Column

Development

OS/Network

Inside View

広告索引

AD INDEX

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 > Plush Studios/Getty Images
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

スマートフォンアプリ開発者とデザイナのための総合情報誌

Smartphone Design

Software Design 編集部　編
B5 判　168 ページ　定価 1,659 円（本体 1,580 円＋税）
ISBN 978-4-7741-5335-3

ユーザから支持されるスマートフォンアプリ開発のために「デザインする力」がより一層求められている開発現場。悩めるエンジニアとデザイナがともに力を発揮するためのヒントが満載です！

第1特集　どうしてデザインと開発は両立できないのか？
第2特集　スマホ開発者がUnityを理解しておくべき理由
その他　Web＋ネイティブでスピード・コスト・メンテに効くアプリ開発／Windows Phoneアプリ開発入門／PlayStation Mobileアプリ開発／仮想化技術でスマホ＆タブレットを業務に／既存のPCサイトをスマホ用に変換！／Webブラウザでクロス開発できるappMobi　ほか

現場で使える [逆引き+実践] Androidプログラミングテクニック

石原正樹、松尾源、磯村禎孝、森靖晃、奥谷修治　著
A5 判　464 ページ　定価 2,919 円（本体 2,780 円＋税）
ISBN 978-4-7741-5187-8

普及が進むスマートフォンで注目されるAndroid OSですが、組込みシステムの宿命とも言うべき「リソースの制限、バッテリー駆動」といった、プログラミングに関わる制限事項が多数存在します。また、「新しい情報をリアルタイムで追随しにくい」といった問題、さらには「従来型のC/C++の組込み開発をしてきた人や会社はJavaに不慣れ」「Javaに慣れた人や会社は低レベルの理解が足りず参入に苦労」といったノウハウ不足の問題に対し、本書は徹底的にチューニングの方法を解説することでも寄与します。

iPhoneアプリ開発塾

iOS 5.1 & Xcode 4.3 対応

カワサキタカシ　著
B5 変形判　320 ページ　定価 2,919 円（本体 2,780 円＋税）
ISBN 978-4-7741-5105-2

iPhoneアプリの作り方について書かれた本は数多くありますが、「基本はわかっても応用がきかない」「やっぱりiOSプログラミングは難しい」といった声が多いようです。本書は、そんな悩めるiPhoneアプリ開発エンジニア達に人気のポータルサイト『サルでき.jp』（旧ブログ：サルにもできるiPhoneアプリの作り方）の管理人が、Xcodeの読み方、プログラミングの基本はもちろん、サポートページの作り方までを、“どこよりも敷居の低い”書き方で丁寧に解説しています。

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

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

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

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

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

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

Software Design plus

Software Design plusシリーズは、OSとネットワーク、IT環境を支えるエンジニアの総合誌『Software Design』編集部が自信を持ってお届けする書籍シリーズです。

プロになるための データベース技術入門
木村 明治 著
定価 3,180円+税　ISBN 978-4-7741-5026-0

データベース技術[実践]入門
松信 嘉範 著
定価 2,580円+税　ISBN 978-4-7741-5020-8

2週間でできる! スクリプト言語の作り方
千葉 滋 著
定価 2,580円+税　ISBN 978-4-7741-4974-5

PCのウイルスを根こそぎ削除する方法
本城 信輔 著
定価 1,980円+税　ISBN 978-4-7741-4867-0

Androidエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4859-5

Vyatta入門 実践ルーティングから仮想化まで
近藤 邦昭、松本 直人、浅間 正和、大久保 修一(日本Vyattaユーザー会) 著
定価 3,200円+税　ISBN 978-4-7741-4711-6

プロのためのLinuxシステム・ネットワーク管理技術
中井 悦司 著
定価 2,880円+税　ISBN 978-4-7741-4675-1

サーバ／インフラエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4600-3

Linuxエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4601-0

Nagios統合監視[実践]リファレンス
㈱エクストランス 佐藤 省吾、Team-Nagios 著
定価 3,200円+税　ISBN 978-4-7741-4582-2

プロのためのLinuxシステム構築・運用技術
中井 悦司 著
定価 2,880円+税　ISBN978-4-7741-4501-3

サーバ構築の実例がわかるSamba[実践]入門
髙橋 基信 著
定価 2,480円+税　ISBN 978-4-7741-4405-4

間違いだらけのソフトウェア・アーキテクチャ
トム・エンゲルバーグ 著 長谷川 裕一、土岐 孝平 訳
定価 1,980円+税　ISBN 978-4-7741-4343-9

よくわかるAmazon EC2/S3入門
五十嵐 学、深海 寛信、馬場 俊彰、藤崎 正範 著
定価 2,580円+税　ISBN 978-4-7741-4284-5

Zabbix統合監視[実践]入門
寺島 広大 著
定価 3,500円+税　ISBN 978-4-7741-4213-5

最新刊！

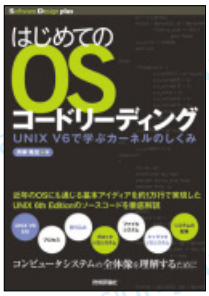
WINGSプロジェクト 著
B5判・352ページ
定価 2,580円(本体)+税
ISBN 978-4-7741-5611-8

最新刊！

菅野 裕、今田 忠博、近藤 正裕、杉本 琢磨 著
B5変形判・336ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5567-8

青木 直史 著
A5判・288ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5522-7

高宮 安仁、鈴木 一哉 著
A5判・336ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5465-7



青柳 隆宏 著
A5判・448ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5464-0

河村 嘉之、川尻 剛 著
B5変形判・480ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5438-1

和田 裕介 著
A5判・208ページ
定価 2,180円(本体)+税
ISBN 978-4-7741-5407-7

㈱マピオン、山岸 靖典、谷内 栄樹、本城 博昭、長谷川 行雄、中村 和也、松浦 慎平、佐藤 亜矢子 著
B5変形判・256ページ
定価 2,580円(本体)+税
ISBN 978-4-7741-5325-4

三苫 健太 著
B5判・400ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5189-2

中井 悦司 著
B5変形判・352ページ
定価 3,400円(本体)+税
ISBN 978-4-7741-5143-4

Software Design編集部 編
B5判・200ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-5037-6

Software Design編集部 編
B5判・200ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-5038-3

木本 裕紀 著
A5判・352ページ
定価 2,780円(本体)+税
ISBN 978-4-7741-5025-3

鶴長 鎮一 著
A5判・344ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5036-9

技術評論社　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

Software Design 総集編 2001→2012

2013年3月27日発売

12年分のバックナンバーがこの1冊に！

Software Design 2001年1月号～2012年12月号の特集、一般記事、連載、特別企画の記事PDFを収録。
総ページ数2万8,000ページ超。

- 1冊1ファイル形式でiPadなどでも使いやすい
- PCより全号一括検索が可能
- 書き下ろし特集「〈OSS全盛期を生き抜くために〉技術の進化をたどりながらLinuxを完全理解」を収録

さらに……
『Software Design総集編【1990～2000】』2013年5月末に発売予定！
【2001～2012】と合わせて購入すれば、約23年間のITの技術／ノウハウが手元にそろう！

B5判　100ページ
2,079円（本体1,980円＋税）
ISBN 978-4-7741-5593-7

技術評論社　全国の書店、またはオンライン書店でお買い求めください。
〒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

リンクアップ　著／224ページ
定価 1,344円（1,280円＋税）
ISBN 978-4-7741-5449-7

技術評論社編集部　著／240ページ
定価 1,239円（1,180円＋税）
ISBN 978-4-7741-5426-8

リンクアップ　著／192ページ
定価 1,029円（980円＋税）
ISBN 978-4-7741-5525-8

リブロワークス　著／144ページ
定価 1,029円（980円＋税）
ISBN 978-4-7741-5571-5

手のひら情報を使いこなせ!!
タブレット・スマートフォン活用ガイド

リブロワークス　著／192ページ
定価 1,029円（980円＋税）
ISBN 978-4-7741-5487-9

技術評論社編集部　著／224ページ
定価 1,239円（1,180円＋税）
ISBN 978-4-7741-5413-8

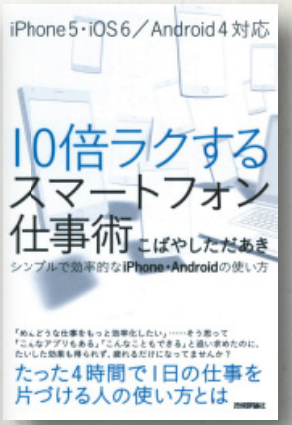
リンクアップ　著／128ページ
定価 1,029円（980円＋税）
ISBN 978-4-7741-5444-2

池田冬彦ほか　著／160ページ
定価 1,554円（1,480円＋税）
ISBN 978-4-7741-5556-2

伊藤朝輝ほか　著／480ページ
定価 1,344円（1,280円＋税）
ISBN 978-4-7741-5496-1

芝田隆広ほか　著／336ページ
定価 1,449円（1,380円＋税）
ISBN 978-4-7741-5298-1

鈴木友博ほか　著／336ページ
定価 1,449円（1,380円＋税）
ISBN 978-4-7741-5524-1



こばやしただあき　著／232ページ
定価 1,659円（1,580円＋税）
ISBN 978-4-7741-5440-4

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

定平誠、須藤智　著
A5 判／ 544 ページ
定価 1,764 円（1,680 円+税）
ISBN 978-4-7741-5386-5

栢木厚　著
A5 判／ 456 ページ
定価 1,869 円（1,780 円+税）
ISBN 978-4-7741-5398-8

きたみりゅうじ　著
A5 判／ 656 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5437-4

山本三雄　著
B5 判／ 544 ページ
定価 1,974 円（1,880 円+税）
ISBN 978-4-7741-5446-6

あなたを合格へと導く一冊があります！

大滝みや子、岡嶋裕史　著
A5 判／ 728 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5351-3

加藤昭、芦屋広太、矢野龍王　著
B5 判／ 464 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5352-0

岡嶋裕史　著
A5 判／ 656 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5355-1

エディフィストラーニング株式会社　著
B5 判／ 392 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5356-8

山平耕作　著
A5 判／ 704 ページ
定価 3,465 円（3,300 円+税）
ISBN 978-4-7741-5353-7

エディフィストラーニング株式会社　著
B5 判／ 484 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-5321-6

村松倫明　著
A5 判／ 544 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-5354-4

内田保男 他　著
A5 判／ 504 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-4944-8

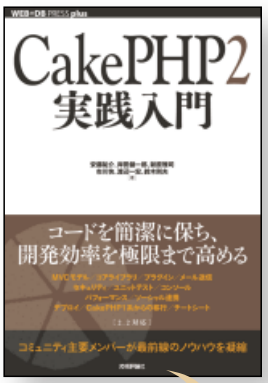
技術評論社
当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150　FAX.03-3513-6151

WEB+DB PRESS 太鼓判。

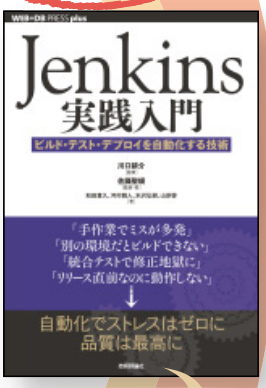
WEB＋DB PRESS plusシリーズは、
Webアプリケーション開発のための
プログラミング技術情報誌
『WEB＋DB PRESS』編集部が
自信を持ってお届けするシリーズです。

外村 和仁 著
A5判・272ページ
定価 本体2,380円＋税
ISBN978-4-7741-5376-6

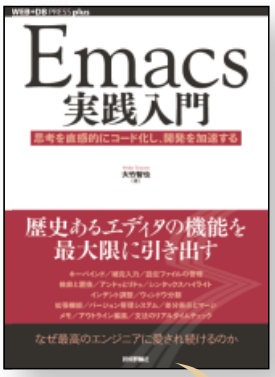
渡辺 修司 著
A5判・480ページ
定価 本体3,300円＋税
ISBN978-4-7741-5377-3



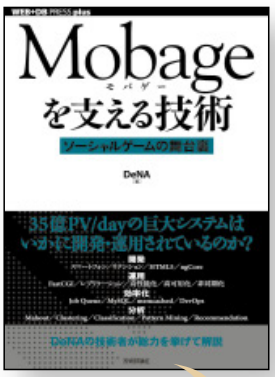
安藤 祐介、岸田 健一郎、新原 雅司
市川 悦、渡辺 一宏、鈴木 則夫 著
A5判・416ページ
定価 本体2,880円＋税
ISBN978-4-7741-5324-7



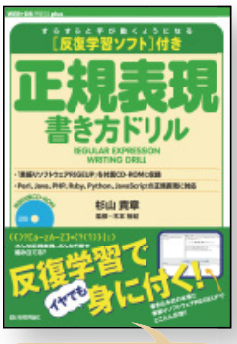
川口耕介 監修、佐藤聖規 監修・著
和田貴久、河村雅人、米沢弘樹、山岸啓 著
A5判・336ページ
定価 本体2,780円＋税
ISBN978-4-7741-4891-5



大竹 智也 著
A5判・272ページ
定価 本体2,480円＋税
ISBN978-4-7741-5002-4

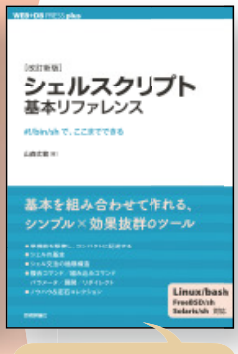


DeNA 著
A5判・384ページ
定価 本体2,780円＋税
ISBN978-4-7741-5111-3



杉山 貴章 著　木本 裕紀 監修
B5判・296ページ
定価 本体2,780円＋税
ISBN978-4-7741-4509-9

羽生 章洋、和田 省二 著
B5判・320ページ
定価 本体2,460円＋税
ISBN978-4-7741-3085-9



山森 丈範 著
A5判・336ページ
定価 本体2,480円＋税
ISBN 978-4-7741-4643-0

杵渕 聡 著
A5判・256ページ
定価 本体2,180円＋税
ISBN978-4-7741-4595-2

縣 俊貴 著
A5判・240ページ
定価 本体2,280円＋税
ISBN978-4-7741-4596-9

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

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

Volume 172

2013 Interaction Awardに見る、インタラクション潮流

技術で人やモノとのかかわりにイノベーションをもたらす

Interaction Award（インタラクションアワード）は、米国IxDA（Interaction Design Association）が主催するアワードで、昨年から始まった新しいものです（昨年のアワードは本誌2012年12月号の本連載で取り上げました）。昨年に比べ今年は作品数も多く、多種多様な応募がありました。アワードの表彰はカナダのトロントで開催されたInteraction13で行われました。

Interaction Award >>> http://www.ixda.org/awards

Interaction13 >>> http://interaction13.ixda.org

Interaction Awardでは、その名のとおり人とモノの相互関係、動作や操作に視点が置かれています。人の行動や行為をどれだけ誘発することができ、新しい体験を提供できたのかどうかが受賞の観点になっています。数多くの応募作品の中から最終候補に残ったのは、世界17ヵ国から寄せられた75のプロジェクトです。その中から珍しいもの、興味深いものをいくつかピックアップして紹介します。

各賞は次の6つのカテゴリに分けられ、複数部門で受賞した作品もあります。

- ●Optimizing（最適化）／日々の行動をより効率的に行う方法
- ●Engaging（魅了）／日々の出来事に喜びや注目を集め、その事柄に意味をあたえる
- ●Empowering（助力）／人々が限界を超え、今までできなかったような事柄をできるように仕向ける
- ●Expressing（表現）／人々の自己表現や創造性を手助けするしくみ
- ●Connecting（つながり）／人と人、人とコミュニティ間のコミュニケーションを手助けするしくみ
- ●Disrupting（再構築）／当たり前となっている既存のサービスや事柄を壊して再構築し、新しい価値を生み出す

人の身体性を生かした遊具のようなインタラクションから、デジタルテクノロジーによってスポーツや身体性の価値を向上させるものなど、インタラクションをキーにさまざまな切り口をもった作品とも製品ともサービスとも呼べるものが受賞作として並んでいます。さまざまなデジタルテクノロジーが、その技術単体では成り立たず、人や人の行動との関係性が高まることで、より良い価値をもたらしていることがわかるものばかりです。

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

gadget 1 Optimizing部門

MyFord Mobile

http://www.myfordmobile.com

車とスマートフォンをつなぐプロジェクト

MyFord MobileはIDEOがFordのために開発した、電気自動車のエネルギー消費量を監視するためのスマートフォンアプリです。現在のバッテリー残量と、次に充電可能な充電スタンドの場所などを的確に把握しながらドライブが楽しめます。また、車がそばにないときでも、充電量や前回いつ充電したのかといった情報をリモートで確認できます。そのほか電気料金の安価な時間に車への充電を開始するように設定できるなど、車とスマートフォンをつなぐツールとしての最適化が図られています。受賞部門がConnecting部門ではなくOptimizing部門であることからも、プロジェクトの方向性が見えてきます。

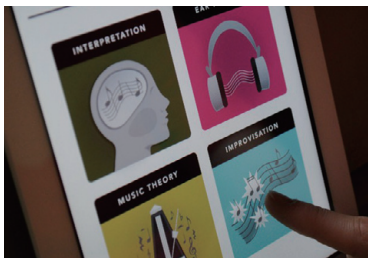
gadget 2 Optimizing部門

Rehearsal

http://www.kmsouthwell.com/?page_id=503

楽器練習アプリ

RehearsalはKirsten Southwellとノースカロライナ州立大学との共同プロジェクトです。楽譜を読む人も、そうでない人も、いろいろなアプローチで楽器を練習できるようにと考えられたタブレットアプリです。バーチャルミュージシャンと共演することで練習したり、演奏の録音／再生、演奏の比較、音を聴く耳をクイズによって鍛えたりと、さまざまな要素が詰め込まれています。



gadget 3 Engaging部門

21 Balançoires

http://www.dailytouslesjours.com/project/21-balancoires/

皆のゆれで音楽を奏でるブランコ

21 Balançoiresはカナダモントリオールの科学センターに設置された巨大な遊具です。ブランコ1揺れ分で、ブランコごとに違う音楽が流れます。1人でも楽しめますが、周りにいる人とブランコのこぎ方を合わせることによって、より複雑な音楽を奏でることができます。このブランコの最大のポイントは、ブランコという誰でもできる遊具によって、見ず知らずの人同士でも人と人のつながりや「協力」といった事柄を生み出す点です。夜になると、ブランコの椅子裏の部分が白色LEDで輝き、ブランコに乗っている人だけでなく、周りで見ている人も楽しめる工夫がなされています。

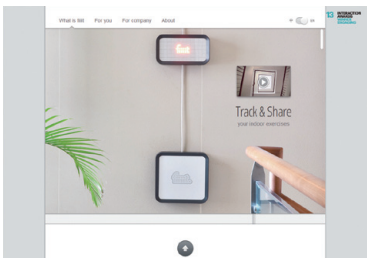
gadget 4 Engaging部門

Fiiiit

http://fiiiit.com

運動促進の非接触カード

Fiiiitは非接触のカードを活用した、おもにオフィスなどの建物内での運動を促進するしくみです。階段の踊り場に設置したカードリーダーにFiiiitカードをかざすことで、運動量や昇降スピード、消費カロリーなどが記録されます。記録された情報はグラフなどで確認でき、個人やコミュニティ内で管理できます。中国からのエントリーで、Interaction Awardの広がりが実感されます。



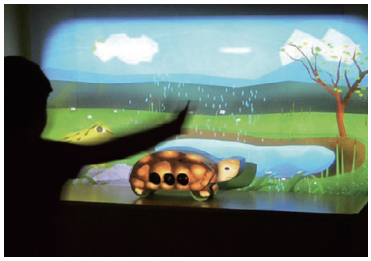
gadget 5 Engaging部門

Gundulas Stories

http://www.fabiangampp.de/GUNDULAS-STORIES

インタラクティブ教育環境

Gundulas Storiesはオールドタイプの木製のおもちゃと、タブレット端末から出力される最新鋭のデジタル映像の組み合わせで、子どもたちが自身の感覚や運動能力を向上させることのできる教育環境です。絵本の読み聞かせが、インタラクティブなストーリーと映像でさらに表現力豊かになったものとも言えます。



gadget 6 Empowering部門

ZocDoc

http://www.zocdoc.com

お医者さん予約サービス

ZocDocは急成長している医療系のサービスです。利用者は無料で活用でき、場所や診療科などによって適切な病院を探し出し、予約することができます。24時間利用可能なことや、その病院で医療を受けた人による医者のレーティングやレビューを読めることなどが人気の秘密のようです。運営費用は医者や病院の登録料によってまかなわれており、登録している医者や病院はそれによって患者数が増え、収益も伸びているそうです。もともとはZocDocのCEOが旅行先で鼓膜が破れ、病院を探すのに苦労した経験からできたサービス。Interaction Awardの受賞作の中では異色のサービスですが、利用者とのインタラクションで問題を解決し、生活を便利に豊かにするサービスとして評価されています。

gadget 7 Expressing部門

Paper by FiftyThree

http://www.fiftythree.com/paper

超絶手書きアプリ

シンプルかつ美しい、手書きノートのようなiPad用のアプリです。スケッチや水彩画のキャンバス、手軽なメモ帳などとして使え、描いたページは美しい表紙のノートとして保存できるといった楽しくなる要素が満載。手書きアプリは数多くあれど、ボタンやメニュー、ファイル、フォルダといった要素は皆無で、ほとんどすべての操作をジェスチャで行えるお手軽かつ深みのあるアプリです。開発したのはMicrosoftの未完のタブレット端末、Microsoft Courierの開発メンバーだそう。

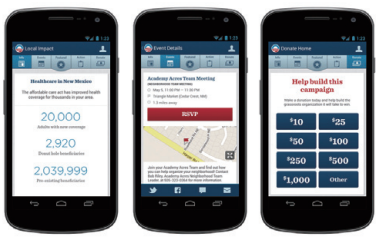
gadget 8 Connecting部門

Obama for America Mobile Campaign

http://thirteen23.com/projects/obama/

選挙戦専用のモバイルアプリ

Obama for America Mobile Campaignは、thirteen23社が作成した大統領選の寄付を集めるためのスマートフォンアプリです。選挙活動におけるさまざまな情報を有権者、応援者に向けて広報するために考えられています。きちんとブランディングがなされているうえ、選挙戦関連の会合やイベントの場所を知ることができたり、少額からオンラインで寄付できるしくみなどが用意されています。選挙戦やキャンペーン活動とうまく連動した形で、モバイルアプリが活用された事例です。



gadget 9 Disrupting部門

NOVA

http://www.nova.lunar-europe.com/Nova.html

デジタルロッククライミングウォール

NOVAは人工的なフリークライミング練習用ボルダリングの壁面と、クライミングコースをアドバイスするスマートフォンアプリの組み合わせで実現されます。コース選択による難易度の変化を教えてくれたり、壁を登っている最中に次に手がかりとする場所が青く光ってサポートします。本物のフリークライミングとは求められる技能が異なるかもしれませんが、テクノロジーを活用した新しいタイプのスポーツとして評価されそうです。

gadget 10 Disrupting部門

Nike+ FuelBand

http:/nikeplus.nike.com/plus/

運動促進デジタル測定バンド

FuelBandは腕時計のように身体に身につけて、運動量を記録するためのデバイスです。個人個人がそれぞれ自分の目標を定めて、その目標に向かって運動量を増やす心がけができます。カロリー消費量や歩数という一般的な数値ではなく、“Nike Fuel”という新しい概念を提唱したところも評価されています。Disrupting部門のほかにOptimizing部門、Empowering部門でも受賞しており、その影響力や安定性、新規性が大変評価されました。スマートフォンやWebとの連携も充実しています。

Interaction Awardは審査員による審査のほかにも、一般の人々による投票の得票数も加味したうえで、受賞作が決まります。今回の2013 Interaction Awardは米国大手家電メーカー GEが大口スポンサーになっており、業界内に新しいアイデアや体験を巻き起こしてくれることを期待しているようです。

全25本の受賞作品は、1月に開催されたカンファレンス、Interaction13の中で発表されました。全体として受賞作は、物事や出来事、人の行動の方向性が「ポジティブ」になるものがとくに評価されているようです。また木製のおもちゃとスマートフォンの組み合わせのように、最新テクノロジーだけではなく、従来手法の新しい焼き直しも効果的に使われている印象があります。作品なのか、製品なのか、広告なのか、オモチャなのか、サービスなのか、簡単にはカテゴリ分けできないものが世界中で考えられ、作られ、広がっていっていることが実感されます。

すでに来年の2014 Interaction Awardも予定されています。Interaction Awardは世界中の最先端のインタラクション技術、インタラクティブアート作品が評価され、世界に知ってもらえる良い機会となります。デザインに特化したコンペティションは数多くあれど、インタラクティブデザイン、インタラクティブ技術に特化したコンペティションは珍しく、来年の作品群もおおいに期待されています。分野としても、アワードとしてもまだまだ歴史の浅いインタラクションデザインですが、世の中にあふれるデジタルデバイスと人との親密な関係を構築していくには欠かせない要素ですね。SD

はんだづけカフェなう

進化したmbed

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

はじめに

今回は“Raspberry PiでI/Oしてみよう（後編）”となる予定だったのですが、筆者が用意していた焦電型赤外線センサモジュールが期待していたような動作ではなく、〆切に間に合わなくなってしまいました。次回までに他のモジュールを入手して、問題解決をしたいと考えています。そんなところにタイミングよくmbedに大きな動きがあったので、今回はmbedについてあらためて紹介をさせてください。

mbedとは

mbedについては本連載の第17回でも紹介しましたが、簡単に触れておきます。mbedは、ARMのCortex-M3というCPUを搭載した高速プロトタイピングを行うことのできるマイコンです。高速プロトタイピングとされているのは、mbedが組み込み開発の煩雑さを省いて、ちょっとしたマイコンを使ったプロトタイプをさっと作りやすいことからでしょう。

たとえば、一般的にはマイコン（組み込み）の開発にはIDE（統合開発環境）やコンパイラ、ライブラリなどを用意しなければなりませんし、使用するマイコンの仕様に依存したコードを書かなければならないためにデータシートを読んで仕様の比較的深い部分を調査する必要があります。こういった手間を軽減するためにmbedはWebベースの開発環境を提供し、またマイコンに依存した部分を相当に抽象化することによって何か動くコードを書くことが非常に楽になっています。

やはり何かを作ってみた人のブログエントリや、サンプルコードの類の情報量はArduinoがダントツに多いのですが、mbedに関する情報も相当にそろってきており、また情報がmbedのWebサイトに集まっているために比較的手軽に始めることができます。また、8bit CPUであるArduinoと比較すると、32bitであるARM Cortex-M3の性能は比べものにならないほど高くなっています。とくにTCP/IPなどのネットワークまわり、USB Hostなどをmbedは標準で搭載しているため、ネットワークやUSBを使おうとするとArduinoよりもmbedのほうが相当に楽に開発できます。

mbed祭り

そんなmbedのユーザや、興味を持っている

▼写真1　NXP玄関

人々の集まりである“mbed祭り”が東京でほぼ1年に1度行われており、今回はmbedのリードパートナーであるNXPのオフィス（**写真1**）で開催されました。これに、筆者もプレゼンをしませんかとお声がけをいただいたので参加してきました。当日の様子はtogetter[注1]にもまとめられていますし、Ustreamの録画もありますので興味がある方はのぞいてみてください。

次回の東京でのmbed祭りは2014年になってしまうと思いますが、有志の間では、近々大阪でもやろうという話になっています。大阪では今年の前半にも開催される見込みです。

mbedの価格

筆者がmbed祭り2013で喋らせていただいた内容とカブるのですが、少しmbedの価格の話をさせてください。mbedは、最近妙に安価になっているマイコン評価基板の中では比較的高価なものになっています。たとえば秋月電子通商で5,200円、スイッチサイエンスで5,250円といった具合に5,000円以上します。同じCPUの評価基板であるLPCXpresso NXP LPC1769ですと、秋月電子通商で2,500円だったりします。

あくまで筆者の推測ですが、この価格差はmbedではクラウドコンパイラが提供されているということが大きいからではないでしょうか。ちなみにmbedのクラウドコンパイラにはRVDS（RealView Development Suite）4.1というARM社の純正コンパイラが提供されており、このRVDSはgccと比較してとても効率の良いバイナリを出力することが可能なものです。なお、このRVDSのライセンスを購入するには100万円近い費用が必要です。

この優れたコンパイラや豊富なライブラリ群へのアクセス権もセットで、この値付けなのだと筆者は推測しています。

注1） http://togetter.com/li/457764

mbed 2.0

そんなmbedですが、最近大きな変化がありました。まず、Webサイト[注2]（**図1**）のデザインが一新されています。まだ公開されてから日が経っていないので、そんなに詳細を見ることができていないのですが、デザインがすっきりとしてずいぶんと読みやすくなりました。

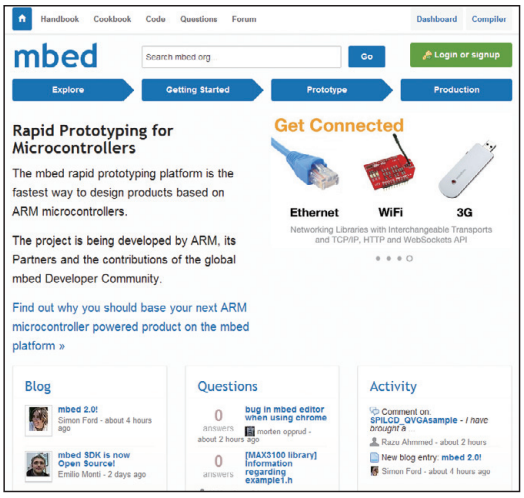
これは今回増えた機能ではないのですが、mbedのWebサイトにはstackoverflow[注3]的なQ&A機能が追加されています。今回のデザイン刷新で、このQ&A機能へのアクセスがとてもよくなったように筆者は感じています。

なお、mbedのサイトは大半が英語ですが、2011年の夏からForumに日本語フォーラム[注4]というコーナーが開設されており、こちらでは日本語での質問ができるようになっています。

そして何よりも待望していたライブラリが、ついにオープンソースになりました。Apache 2.0ライセンスです。コードはGitHub[注5]でも公開されています。もともとかなりオープンな開発環境だったのですが、今回のオープンソース

注2） http://mbed.org/
注3） http://stackoverflow.com/
注4） https://mbed.org/forum/ja
注5） https://github.com/mbedmicro/mbed

▼図1　mbedのWebサイト



化によって、ライブラリの問題を見つけ出し、修正したものをpull requestできるようになったのは大きいと思います。筆者もいくつかライブラリの問題を見つけてはいるので、良い解決策を見いだして、いつかはパッチを提出してみたいと考えています。

mbed HDK

従来よりmbedは互換機が作られることに後ろ向きではありませんでした。中の人がmbedでプロトタイピングしたものを基板に起こすための情報を公開していたり[注6]しますし、回路図や基板製図CAD用のファイルなども配布されています。

今回、HDK(Hardware Development Kit)ということでページが追加された[注7]のですが、今のところとくに新しい情報は見受けられないのが残念です。

余談ですが、mbedのオンラインコンパイラでコンパイルしたバイナリは商用・非商用を問わず、自由に使って良いとされています[注8]。

このように、ハードウェアもソフトウェアも流用ができますので、いろいろな遊び方ができるのではないでしょうか。

CMSIS-DAP

このソースコード公開からは少し前になるのですが、mbedにβ版ながらもデバッガ機能が追加されました[注9]。

デバッグ機能にはCMSIS-DAPというプロトコルが使われています。CMSISというのは、Cortex Microcontroller Software Interface Standardの略で、Cortex-Mプロセッサシリーズ向けのベンダに依存しないハードウェア抽象化レイヤです。この、CMSIS-DAPは、同じく標準化されたプロトコルで、mbedなどのターゲットとなるボードのデバッグポートとホスト(PC)をUSBで接続してソフトウェアの実行を制御したり、メモリのリードライトなどのデバッグを行うといった、ソースコードレベルのデバッグができるものです。

残念ながらクラウドコンパイラでは、このデバッグ機能をホストできません。CMSIS-DAPをホストするには、MDK-ARMといった有償のARM開発環境や、OpenOCDやgdbといったフリーの環境を使う必要があります。先述のmbed祭り2013では、CMSIS-DAPとMDK-ARMを使用して、ステップ実行やブレークポイントの設定、レジスタへのアクセスといったデバッグ機能のデモが行われました(**図2**)。

以前、本連載の第19回で、Netduinoのデバッグ機能を紹介させていただきましたが、mbedのデバッガも逆アセンブルが表示されたりとたいへん便利そうです。難点はNetduinoの場合はVisual C# Expressで費用負担なく開発環境を用意できるのですが、mbedの場合、無償で使うことができるMDK-ARMのLite版では扱えるバイナリのサイズが32KBまでと制限されていることです。しかし、これもgdbとOpenOCDを使ったCMSIS-DAPによるデバッグがサポートされたり、情報が多くなれば解決することでしょう。

mbedの新機種

これまで発売されていたmbedは、すべてNXPセミコンダクターズのCPUを搭載していましたが、ついにNXP以外のmbedが登場しました。

フリースケール・セミコンダクタはテキサス州オースティンに本社があり、KindleやKobo、Android端末などにも採用されているi.MXファミリのプロセッサなどを製造している会社です。

同社のFRDM-KL25Z(**写真2**)という評価基

注6) http://mbed.org/users/chris/notebook/prototype-to-hardware/
注7) http://mbed.org/handbook/mbed-HDK
注8) http://mbed.org/handbook/Nontechnical-FAQs
https://mbed.org/handbook/mbed-Compiler#compilation
注9) http://mbed.org/handbook/CMSIS-DAP-MDK

板向けにmbedの“interface upgrade”というソフトウェアの提供が開始されたので、入手した評価基板にこれをインストールすることでmbedとして利用することができるようです[注10]。

このFRDM-KL25Zですが、一個千数百円で販売されています。これをmbedとして使うために追加コストは必要ないようですので、KL25Zを使うのが費用負担を最も少なくしてmbedを始めることができる方法になりました[注11]。従来のmbedと比べるとブレッドボードに挿して手軽に使ってみることができませんが、基板上にはRGBのLEDに加えて、タッチセンサと加速度センサが搭載されており、mbedの開発環境がどういうものか覗いてみるには良い選択肢だと思います。

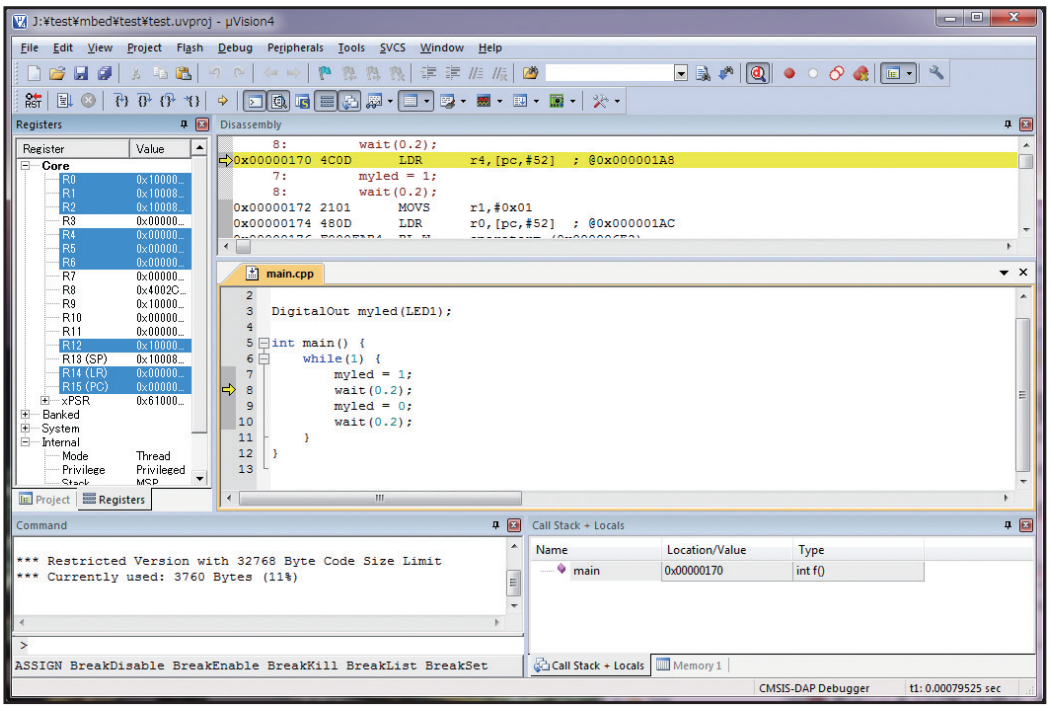
最後に

「執筆活動中にどんなときも私を支え、励ましてくれ、側にいてくれるような妻がほしいです。」これはTwitterで見かけた@siritori氏の呟きなのですが、筆者のツボにハマりました。

本連載をはじめてから2年半になりますが、連載についての感想やお問い合わせや叱咤激励などございましたら気楽に、筆者宛に直接お便りください。SD

▼写真2　FRDM-KL25Z

▼図2　mbedをMDK-ARMでデバッグ



注10）https://mbed.org/handbook/mbed-FRDM-KL25Z-Getting-Started

注11）筆者の手元に評価基板が届いていないため、まだ実際には試していません。

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4月17日です。プレゼントの発送まで日数がかかる場合がありますが、ご了承ください。

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 ロジクール タッチマウス t620

1名

Windows 8のタッチジェスチャを表面全体で操作できるマウス。水平／垂直スクロール、ページの進む／戻る、スタート画面の呼び出し、アプリケーション切り替えなど、ジェスチャで快適に操作可能。Windows 7でも使えます。

提供元 ロジクール
URL http://www.logicool.co.jp

02 Boombero Wireless Speaker

1名

iPhone用のワイヤレススピーカー。特別な設定をしなくてもiPhoneをスピーカーの上に置くだけでパワフルなサウンドを再生できます。iPhone 5/4S/4/3GS/3G/iPod touch（第4世代以降）に対応。

提供元 リンクスインターナショナル
URL http://www.links.co.jp

03 ほこりトリ

10名

鳥の形をしたシリコーンゴム製掃除マット。静電気を帯びやすく、ほこりが吸い付きやすいシリコーンゴムの特性を利用して、PCやテレビの周辺、棚の上などのほこりをしっかりキャッチします。

提供元 おおかわ
URL http://www.ookawa1976.com

04 CloudStack 徹底入門

3名

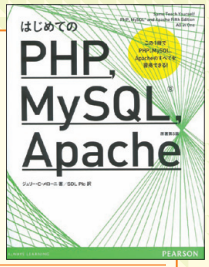
日本CloudStackユーザー会 ほか　著／
B5変型判、392ページ／
ISBN = 978-4-7981-3058-3

サーバ仮想化環境を基盤にして、IaaS型のクラウドを提供するOSS「CloudStack」。その基本的概念、導入の流れと操作、応用的な利用方法、開発プロジェクトへの貢献のしかたまでを紹介します。

提供元 翔泳社　**URL** http://www.shoeisha.co.jp

05 はじめての PHP,MySQL,Apache

2名



ジュリー・C・メローニ　著、SDL Plc　訳／
B5変型判、624ページ／
ISBN = 978-4-86401-111-2

PHP、MySQL、Apacheの使い方を短時間でまとめて習得できる1冊。初心者でもダイナミックでインタラクティブなWebサイトを作成できます。PHP 5.3、MySQL 5.1、最新版Apacheに対応。

提供元 ピアソン桐原　**URL** http://www.pearsonkirihara.jp

06 ストレージの原則と技術

2名

EMC Education Services　著、㈱クイープ　訳／
B5変型判、464ページ／
ISBN = 978-4-8443-3351-7

ストレージ技術の概念や原理から、実装の技術的考察までを網羅。新技術として重複排除やユニファイドストレージ、仮想プロビジョニング、FCoE、フラッシュデバイス、ストレージ階層化を解説。

提供元 インプレスジャパン　**URL** http://www.impressjapan.jp

07 Windows ストアアプリ開発スタートガイド

2名

田中 賢一郎　著／
B5変型判、340ページ／
ISBN = 978-4-8443-3352-4

はじめてWindowsストアアプリを開発する初心者向けの解説書。画面デザイン、機能、UIから完成品のストア登録まで、公式ガイドラインを熟知した著者がやさしく丁寧に解説します。

提供元 インプレスジャパン　**URL** http://www.impressjapan.jp

第1特集

裏口からのプログラミング入門
――僕（私）の言語の学び方

CONTENTS

Part 1

リビドー（ドリブン）駆動プログラミング学習

株式会社ワディット 代表取締役社長 和田裕介
ゆーすけべー / yusukebe
yusuke@kamawada.com　http://yusukebe.com/

はじめに

本記事では「裏口からのプログラミング入門」の事例およびメソッドとして「リビドー駆動」なプログラミング学習を紹介します。これは筆者自身が自然と体験した「学び方」そのものであり、「裏口」と言えど、多くのハッカー達が歩んで来た道でもあるかと思われます。「表口」からのプログラミング学習が技術本を読み、ステップバイステップで体系的に学んでいくことと今回の記事内では仮定し、ところどころ「表と裏」を比較しつつリビドー駆動プログラミング学習について紹介していきます。

リビドーで何が悪い!

さて、とりあえず「リビドー」という言葉について理解してみましょう。そもそもは哲学用語で、精神分析学者ジークムント・フロイトが「性的衝動を発動させる力」という意味合いで使っていたらしいです[注1]。またもう少し突っ込んで調べてみると、性的なものが対象になるばかりではなく、科学研究や芸術活動に対しての強い意志とそれを何らかの形で爆発させることもリビドーと言って良いようです。

こうした言葉の意味を踏まえつつ、さらに広くとらえるとリビドーはこのように定義できます。

「自らが叶えたい欲望を爆発させ実現に向かうその力」

何やらドロドロしつつも力強いイメージですね（笑）。最もわかりやすい例が「性欲」なんですが、そのほかにも「俺はコレがやりたいんだー！」「私はこんなものを作りたいんだ！！」といった普段から強く個人的にやりたいことがリビドーの対象となるでしょう。

このリビドーの力をプログラミングの学習に応用してしまおうというのが、今回紹介する「リビドー（ドリブン）駆動プログラミング学習」です。プログラミングを学習する段階において、どのように学んでいるか？――の答えがエロいことでも、それを人様に教える必要はもちろんありません。プログラミングをするための目的はけっして制限されているわけでもないのです。

YAPC::Asia 2010 というイベントのキーノートで宮川達彦さんがスピーチをしました[注2]が、

注1）Wikipediaのリビドー項目（http://ja.wikipedia.org/wiki/リビドー）

注2）「そんな言語で大丈夫か？　YAPC::Asia Tokyo 2010フォトレポート－@IT自分戦略研究所」（http://jibun.atmarkit.co.jp/ljibun01/rensai/photo/12/01.html）

その中で筆者が強烈に覚えているフレーズがあります。

「コードを書くのに許可はいらない」

宮川さん本人が喋っていたときのコンテキストとは多少ずれるため、彼が表現したかった意味からずれるかもしれませんが、とにかく僕らは誰かの許しを乞う必要もなくプログラミングをできるのです。だから、人様に言うと恥ずかしいような目的だとしても、そのためのコードを書いてもいいのです！　ちなみにネタとして「自分はこんな（恥ずかしい）目的で学んでる」ということをブログ等の記事にして注目を集めるのもエンジニアとしてのセルフブランディングとしてはアリかとは思います。

リビドーを実現したい、そのためにコードを書く、わからなければ調べる、習得する。端的に言えばこうした行程がプログラミング学習を加速させるというのが筆者の結論です。

目的まずありき

人それぞれですが、プログラミングを、あくまでやりたいことを実現するための「手段」と考えると最初のとっかかりとして有効です。たとえば、筆者はWebサービスの開発および運用を日々の業務としていますが、それはあくまでサービスを提供するためにコードを書いているととらえています。

するとリビドー駆動でやる意味が出てきます。表口からの学習はどうしてもプログラミングそのものに焦点があたりがちですが、リビドー駆動だと目的がはっきりと明確なため、とにかく動かすことになります。コードが汚くても、コピペで書いたコードでも、何をやっているのか理解してなくともリビドーを昇華できればいい。すると「動いた！」という小さな成功体験を得ることができるので、プログラミングにのめり込みやすく、継続的な学習につながるのです。

学んだ結果の実力は必ず残る

リビドー駆動で勉強をするとなると、欲求を満たす何かをプログラミングで「必ず」実現しなくてはいけないのか?——と思われるかもしれません。おっぱい画像が欲しかったらそれを大量に集める。もちろんうまくいくにこしたことはないのです。ただ、もし実現できなくとも、1つだけ確実なことが言えます。それは学んだ結果、スキルや知識は確実に向上するってことです。難しいことに対してリビドーの強い力で食らいつくことが、学習の効率化につながるでしょう。

概念的な解説が済んだところで……

以上、リビドー駆動プログラミング学習についての概念を解説したところで、筆者は「どのようにリビドーを昇華しつつプログラミングを学んでいったか？」を以降紹介していきます。

性欲を昇華させることで学んだ

筆者がWebプログラミングを本格的に始めたのは大学院卒業後でした。学生のときには授業でJavaやCなどを、研究室のプロジェクトでFlashのActionScriptによるプログラミングを行っていたのですが、Web方面のそれとなると経験がそこまでなかったのです。院の卒業後すぐに父親とともに起業をして、「稼げるのはWebだろう」と始めたWebプログラミング。これこそリビドー駆動で学習していきました。

Plaggerから入ったWebプログラミング言語

その当時2006年頃、ちょうど「Plagger」というアプリケーションがエンジニア界隈で大流行りをしていて、それをたまたま触ったことから筆者はPerlという、Webプログラミングで使える言語を習得していくことになります。

Plaggerとは一言で言えば「プラガブルなフィードアグリゲータ」。ブログなどで出力さ

れる通常のRSSやAtomを取り扱うだけではなく、Web上の情報を集約、まとめ、希望の場所に配信できます。これを触るのがすごくおもしろかった。

まず、手始めにやったこと。それは、僕が愛してやまない横浜ベイスターズ(現在では横浜DeNAベイスターズ)の試合情報が、当時RSS/Atomフィードを吐いていなかったので、フィードリーダーなどで読めるようにフィード化する作業です。そのためには「YAML形式という設定ファイル上でWebページ上のどの情報を取得していくか?」を「正規表現」なるルールで書かなくてはいけません。今まで馴染みがなかった正規表現をGoogleで検索しながら学びつつ、試行錯誤してやっとうまく動いたときにはうれしかったです。

スクレイピングをエロで学ぶ

Plaggerで球団情報を扱うだけだと少しもの足りないので、リビドー駆動を発動。大手エロサイトからの更新情報を取得して一元的に閲覧できるものも作りました。そうこうしているうちに正規表現の基本を知り、Plaggerの挙動を調べるために内部のソースをあさり、結果的にPerlプログラムを「コードリーディング」するようになるのです。

そこで気づいたのは上記のようなWebページの一部の情報を拾ってくるプログラムはとくにPlaggerに限らず「スクレイピング」と呼ばれる手法であること。それを「今度はPerlプログラムで一から書いてみたい!」と思うようになります。もちろんエロなコンテンツを集めるために……。

スクレイピングをするためにはいくつかの段階を踏まなくてはいけません。

①対象となるWebページのHTMLを取得する
②HTMLから正規表現で必要な情報を抜き出す
③標準出力に表示するなど応用する

具体的な例を挙げましょう。エロサイトをここで紹介すると筆者の性癖がばれるので(笑)今回は筆者のブログ記事のタイトルのテキストをぶっこ抜いてみます。Perlで書いてみると、まずはおまじない的な記述とWebページをダウンロードするためのユーザエージェントのライブラリを使用する宣言をします。

```
use strict;
use warnings;
use LWP::Simple qw/get/;
binmode STDOUT, ":utf8";
```

次にHTMLをダウンロード。「$content」という変数に入ります。

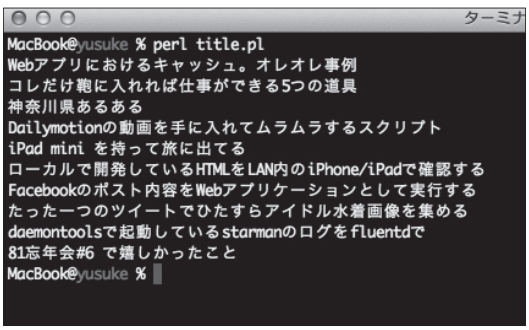
```
my $url = "http://yusukebe.com/";
my $content = get($url);
```

$content 変数の値に対して正規表現でタイトルを取得し、標準出力へ表示しています。

```
my @titles =
  $content =~ m!<h2.*?entry-title.*?><a.*?>(.+?)</a></h2>!g;
print "$_\n" for @titles;
```

実行結果は**図1**のようになります。10行にも満たない短いスクリプトですが、スクレイピングの要素が詰まっていると思います。また、今回はPerlでの実装例ですが、ほかの言語でもスクレイピングはできますし、コード内の過程はほぼ同じでしょう。

▼図1　実行結果



エロ動画自動キュレーションアプリケーション

さて、スクレイピングという技術を習得する

と、ワールドワイドウェブに乗っているWebページのすべてがリソースになり得る可能性を感じます。そこで「性欲」という強烈なリビドーを昇華するために次の命題に思いを馳せることになります。

いかにして無料エロ動画を集め整理をするか

さんざん考えたあげく一部スクレイピングも利用しながら、この命題に答えるWebアプリケーションをつくるしくみを思いつきました。名前はあえて伏せておきますが、現在も稼働中の某アダルトサイトで使われています。このしくみ、発見したときには「俺ってば天才!?」と思い込んでしまう興味深いもので、事実当時ではおそらく誰も試していなかったものです。紹介しましょう。

世の中には無料エロ動画サイト（おもに海外でホスティングされているサイトが多い）のリンクを張って記事にしているエログと呼ばれる類のブログサービスが多数あります。ただ、こうしたサイトでは広告ページやランキングサイトなどへのリンクが多数張られ非常に見づらいのが特徴です。いわゆる「騙しリンク」と呼ばれるものも多いです。これではせっかく紹介されているエロ動画にもなかなかたどり着きにくいですし、さらに言えばこのようなエログを横断的に検索、カテゴライズして動画を探すことができれば、うれしいと思いました。リビドー駆動でもあり、このような問題を解決したいという欲求にかられたのです。

では、どうしたか？　いきなりこのしくみのフローを紹介します。

①Googleなどのブログ検索で海外エロ動画共有サイトの名前で検索
②ヒットするブログ記事はたいていエログである
③その記事に対して正規表現でエロ動画共有サイトへのリンクを見つけ出す
④記事のタイトルにAV女優の名前が入っていれば見つかった動画と結び付ける
⑤これで大量の動画へのリンクが集まり、一部は女優と組み付けがされる

うーん、権利関係等問題がありますが、我ながらすばらしいしくみですね（笑）。

こうして集まった動画の情報をデータベースに入れ、Webアプリケーションでキレイに配置したらエロ動画の「自動キュレーションサービス」と呼べるものができました。ちなみに2007年の年末にこのしくみをWebサービス化し公開したのですが、発表当初のインパクトは大きく、はてなブックマークも異様について、はてブトップページに掲載されてしまうほどでした（笑）。今でこそはてブにはアダルトコンテンツが載りにくくなっていますが、その当時は自分が作ったものが多くの人に見られるってことで興奮しましたね。

自分の欲望、みんなの欲望

こうして、PlaggerからPerlとスクレイピングを学び、可能性を感じ、最終的には自動更新のエロサイトを作るまでになりました。これは自らのリビドーからスタートしているのですが、エロサイトを公開することでほかの多数の人の欲望をも叶えるサービスになりました。リビドー駆動プログラミング学習は冒頭でも申したとおり「オレオレ」な個人的範疇の欲望達成を目的とすることですが、もしかして、その結果、ほかの人の欲望も達成していることになるかもしれません。ただし、あくまでも著作権侵害や公然わいせつ罪に抵触しないか法律関係のリスクは検証して公開をしたほうがいいでしょう。もし自信がなければ、手元のローカルマシンでWebアプリケーションを立ち上げるのもオツですね。

みんな大好きダウンロード

「ダウンロード」その言葉はプログラマだけでなく、一般のPCユーザにとっても興味を示す

ものです。近年、違法コンテンツをダウンロードすることが犯罪となり、この件に関してなかなか言い難い状況ではありますが、ネット上の気になる動画や音楽をファイルとして手元に落とすことはいかにもリビドーが働いた結果でしょう。

YouTube、ニコニコの動画が欲しい!

エロサイトを作っていた同時期にこうしたダウンロードをプログラムを使って自動化する、もしくは、手動では難しいダウンロードをプログラムを使い容易に行うというスクリプトを書いてました。とくに興味を持ったのがYouTubeやニコニコ動画などの動画共有サイト上の動画ファイルです。ちなみに、このケースは違法ダウンロード禁止には引っかからないとはいえ、サイト運営者に迷惑がかかる行為ではあるので、あくまで私的利用で行います。

この動画共有サイトからファイルをダウンロードするプログラミングは、得られるものが多い分、楽しいです。各動画共有サイト側は年々こうした機械的にコンテンツをダウンロードする件について対策を行っており、徐々にプログラムを書くのが難しくなっています。それでもコードを書く大きな流れは変わらないのでコードの中でやるべき手順を紹介しましょう。基本的に以前説明したスクレイピングを一部で利用します。

▼リスト1　動画共有サイトのファイルをダウンロードするスクリプト

```
use strict;
use warnings;
use LWP::Simple qw/get getstore $ua/;
use URI::Escape;
use JSON;

# ダウンロード状況をモニタリングするために設定
$ua->show_progress(1);

# Dailymotionの個別リンクのURLを引数に持つ
my $url = $ARGV[0] or die "URL argument is required!";
die "URL is not dailymotion link"
  unless $url =~ m!^http://www\.dailymotion\.com/video/.+$!;

# URLから動画のIDを切り出す、後ほど使う
my ($vid) = $url =~ m!/video/([^\?#]+)!;

# 個別リンクのHTML取得、変数に入れる
my $content = get($url);

# Dailymotionはビデオの情報をJSON形式の文字列で保持しているので
# その文字列を取得し解析
my ($text) = $content =~ m!"sequence":"(.+)"!m;
my $json = uri_unescape($text);
my $ref = decode_json($json);
my $param
  = $ref->{sequence}[0]{layerList}[0]{sequenceList}[2]{layerList}[2]{param};

# JSONの中に記載してある動画ファイルへのURLを取得
# 高画質版と通常版があるので高画質があればそちらを優先
my $video_url = $param->{hdURL};
$video_url = $param->{sdURL} unless $video_url;
warn "GET: " . $video_url . "\n";

# 動画のIDを元にファイル名をつくり動画をダウンロード
getstore($video_url, $vid . '.mp4');</code></pre>
```

①欲しい動画の固定リンクにアクセスする
②HTMLを解析
③動画ファイルへのパスやパスを取得するためのAPIへのURLがたいてい含まれている
④正規表現で上記を取得、動画のパスを取得
⑤APIのURLだけ記載の場合はそこへアクセスし動画へパスを取得
⑥動画自体へのパスへアクセスし、それ自体をユーザエージェントでダウンロード

先日Dailymotionという動画共有サイトのファイルをダウンロードするスクリプトを書いたので、事例として**リスト1**に掲載しましょう。

ちなみに、YouTubeの動画を取得するにはもう少し工夫が必要になっています。とはいえ、Perlのライブラリ集であるCPANにはその名もWWW::YouTube::Download[注3]というモジュールが配布されています。インストールすると「youtube-download」なるコマンドがシステムにインストールされるのでそれを使うと簡単にダウンロードができてしまいます。このモジュールはxaicronさん[注4]作なのですが、僕が一部アドバイスをさせてもらったことから「貢献者」としてコード内に「yusukebe」の文字が掲載されています。

iPodに動画を入れて楽しむハック

ダウンロードのしくみを理解しプログラムを書けるようになるとさまざまな応用が効きます。今まで試したもので効果的だったのが、動画の一覧をPodcastとして出力、iPodなどへファイルを転送するというしかけです。それを実現するために、Plaggerを使ったり、自作のアプリケーションを作ってみたりしました。そのころ契約していたとある常駐先へ通勤する時間にiPod nanoで映像を見てた記憶がありますね（**写真1**）。

画像、音楽、動画とダウンロード対象はさまざまあるものの、スクレイピングがやはり基本となるプログラムを組むことになります。リビドーにかられ夢中になってやっているとプログラミングスキルが向上していることでしょう。ただ、繰り返しになりますが、法律や倫理の関係であくまで私的利用にとどめたほうがいいでしょう。

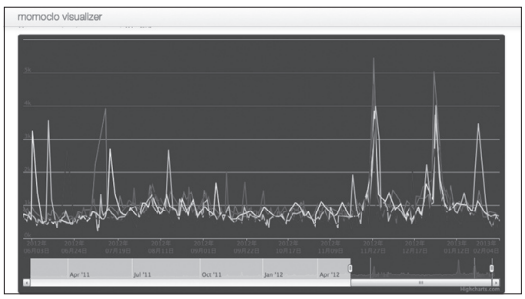
アイドル、グラビアというリビドー駆動も……

sugyan[注5]という冬でも雪駄を履いてるエンジニアの友達がいます。彼はいつの日かアイドルグループ「ももいろクローバー（ももクロ）」に目覚め、世界を含めた各地に出向いてライブを見に行く感じになっちゃいました。おそらく、彼にとってのリビドーとはももクロへの熱意であ

注3） Yuji Shimada / WWW-YouTube-Download - search.cpan.org （ http://search.cpan.org/dist/WWW-YouTube-Download/）
注4） にひりずむ:: しんぷる（http://blog.livedoor.jp/xaicron/）
注5） すぎゃーんメモ（http://d.hatena.ne.jp/sugyan/）

▼写真1　Plaggerのデータを Podcast化

▼図2　momoclo visualizer



りましょう。そして、それを表現するため、エンジニアであるsugyanはももクロ関係のWebアプリケーションを作っています。たとえばももクロメンバーのブログへのコメント数を可視化する「momoclo visualizer[注6]」がその一例です。

時系列でコメント数が可視化されるので、ブログへのコメントがいつ盛り上がっているのかをメンバー別で追うことができます。たとえば誕生日を迎えた子がいたらそのグラフが一気に上昇するなどの現象も観察されているようです[注7]。

また、ひょんなことから参加させていただいている勉強会がありまして、それを主催しているやたら貫禄のある大学院生の飯塚君はどうやらアイドル写真を眺めるのが好きなようです。「Imagerous*[注8]」という壁紙を探せるサービスを作ったのですが、彼がそのサービスを紹介したブログ記事ではアイドル画像の一覧が例として推されています[注9]。

2人ともアイドル、グラビアとフィールドは違えども、それぞれ苦労しながらリビドー駆動で開発をしていることが雰囲気として伝わってきます。

おもしろさで仕事もする

筆者にとってリビドー駆動はプログラミングを学ぶときに重要な要素なわけですが、仕事においても「やっていて楽しい！！」という感覚を持てるように努力しています。現在、筆者がコミットしている「ボケて[注10]」ではいくつかの要素がうまく機能して楽しく開発、運用をさせてもらっています。

まずは信頼できて「ノリ」が合う仲間とフラットな関係で仕事をすること。奇遇にも一緒にやっているパートナー達が筆者と同じくらいの「81世代」であり、育ってきたバックグラウンドと価値観が比較的近いのです。また、それぞれ、その道のスキルを十分に持っていて、向上心も高いことなども楽しく仕事ができている要因でしょう。

次に重要なのは、作っているものが自分たちで使えておもしろいこと。これはリビドー駆動にも通じることで自らのアプリケーションを1ユーザとして毎日チェックしています。ボケての場合は「笑い」がテーマになっているので、日々笑わせてもらっています。

そして、最後はビジネス的な側面で充実すること。これはもちろん仕事としてやっているので避けて通ることはできません。

一例であるボケてのケースではヒットするまでローンチから4年という長い年月が必要だったのですが、このような「おもしろさ駆動」の仕事ができるかどうかは時間がかかります。周りの仲間、作るもののアイデア、そしてそれをビジネスとして成り立たせる努力、この3つをじっくりと固めていくとおもしろさ駆動で仕事として開発ができるかもしれませんね。

まとめ

今回はリビドー駆動プログラミング学習と題し、著者が体験してきたプログラミング学習の行程や具体的な実装方法などを紹介してきました。「これがやりたい！！」「こんなの作りたい！！」というリビドーはたぶんみなさんの心の中に潜在的にでも存在していると思います。それに目を向けて自分なりのリビドー駆動で楽しくプログラミングを学んでいってください。SD

注6）momoclo visualizer（ http://momoclo-visualizer.herokuapp.com/blog_comments/）

注7）ももクロの人気上昇ぶりをグラフで可視化する ― すぎゃーん メ モ（http://d.hatena.ne.jp/sugyan/20120417/1334623422）

注8）Imagerous（http://imagero.us/）

注9）Imagerous つくったよ！ ― tulog つろぐ（http://d.hatena.ne.jp/tushuhei/20120506/1336295385）

注10）ボケて（bokete）：写真で一言ボケるウェブサービス（http://bokete.jp/）

美容師の世界からプログラマへ

株式会社ステラート 横山 彰子
YOKOYAMA Akiko　TwitterID：@acotie

はじめに

自己紹介

はじめまして。20才からプログラマとして仕事をしている横山 彰子です。それまでは美容専門学校に通い、美容師アシスタントを2年半ほどしていました。現在は、複数の開発会社と2年間のフリーランスを経て、おもにスマートフォンアプリケーションの開発を行う株式会社ステラートという会社を設立し、代表兼チーフエンジニアとして活動しています。今回はプログラマとして今までの行動を振り返り、どのようにすれば効果的に学習できるかを真剣に考えてみました。少しでも皆さんの活動のヒントになれば幸いです。

美容室で働くきっかけ

もともと筆者は高校在学中に、年長の良い先輩の紹介で美容室でカットモデル(練習台)をしていました。その美容室の美容師さんも素敵な人で、1年近く通っている内に「私も美容師になりたい」と思うようになりました。そうした縁で美容室に内定をもらい、高校3年生から大阪の通信制美容専門学校にも通いました。早く社会で自立したいという考えがありました。弟が私立中学に在校していたので、経済的に親に迷惑をかけまいと勝手に思い込んだ結果、美容室の女子寮で一人暮らしを始めることにしました。

美容専門学校と美容師アシスタント時代

美容師時代は、朝7時から朝レッスンをして、8時～20時までシャンプーや仕事をし、20時から24時まで夜レッスンをしていました。その結果、深刻な手荒れに悩まされ、自分の顔すら素手で洗えなくなってしまいました。加えて狭い美容の世界にどっぷり浸かっていて一般社会の常識がない人間になってしまうのではないか、と怖くなってしまったのです。ついに、志していた道を諦めることにしました。個人として大きな挫折でした。

プログラマへの道

その当時、志半ばで心にぽっかりと穴が開いてしまった状態で、何も考えられませんでした。実家に帰り、地元の友人に偶然会うと近所の工場で働いているといいます。なんとなく興味本位で勤務先を紹介してもらいました。自動車で使うゴムの部品を検品をする単純作業です。1日目から本当に退屈に思えて3ヵ月くらいで辞めました。その仕事は頭を使うことなく、ひたすら同じ作業をすることがたいへん苦痛だったのです。次の仕事は頭を使う仕事がいいな、と

漠然と考えていました。

父が銀行系のコンピュータの営業をしていたこともあって、当時のPC(PC-98)は物心ついたときからありました。Windows 95や98、2000、XPと触れていました。それでPCを使った仕事が面白いかもしれないと思いました。たまたまWebサイトで、大阪で未経験のプログラマの求人を見つけ、応募してみたのです。「今までと違ったことをしてみたら？」と知人から後押ししてもらったこともありますし、美容師と同じように手に職を付けたかったのです。

面接では、PC-98やファミコンの話、高校時代に携帯で勝手に学校のホームページを作ってばらまいて炎上して警察沙汰になったこと、何回もパケ死して親に怒られたこと、趣味で簡単なHTMLタグを覚えていたこと、学習意欲があることをアピールしました。どうしても受かりたかったので「Excelやパワポ使えます！タッチタイピングできます！」と全然できないのにできると言っていました。1ヵ月以上経って採用の連絡がきました。実は60人以上応募が来て、採用されたのは筆者ともう1人だけで年上の経験者だったということを後から聞きました。当然ながら入社後すぐに「全然タッチタイピングできないやん！」とツッコまれました。

現在まで

最初の会社から転職してPHPやPerlを始めるのですが、最初は全然できなくて本当にダメダメな感じでした。エンジニア3年目のころ、Perlをはじめて半年ほど経ってからパッと視界が開けたような感覚がありました。憧れる人が目の前に現れて、具体的なイメージが明確になり、家に帰ってもプログラミングに夢中になったからだと思います。そしてありがたいことに、ずっと憧れている会社の人達と一緒にお仕事をさせてもらう機会があったり、仲良くなったり、いろいろと感慨深いです。

それまでWebプログラミングをずっとしていたのですが、現在は2011年から始めたゲームプログラミングに夢中です。Objective-CやC++を使ったネイティブアプリケーションと呼ばれるゲームをみなさんと同じように、いろんな企業と一緒に企画を含め1から開発し勉強しています。ゲーム作りはこれまでと違って映画を作るような感覚で、アニメーションやパーティクルなどの演出をはじめ、ユーザビリティ、ゲームレベルなど非常に奥が深くて楽しいです。いろんな人達と一緒に1つの作品を作るということを経験し、辛いときもありますが、その分の喜びも大きいです。

プログラミングをどうやって学んだか

先輩とのマンツーマン開始

プログラマを始めて約2年が経ったころに「オープンソースの言語をしたい！」と意気込んで転職したら、他の人の10倍くらい仕事が早くて凄腕の先輩プログラマに教えてもらうことになりました。その先輩は「自分がたどった道を全部教えるから、集中して短期間で覚えることができる」と自信満々に言っているので必死で付いて行くことにしました。それまでは完全にWindows環境で、UNIXにすら触れることはありませんでした。個人でPerlの本は何冊か買っていたり、会社のブログを始めるためにMovableTypeの設定で軽く触る程度のレベルで、まったくわからないところからのスタートです。

■先輩からのアドバイス

まず読むことと言われたドキュメントは、株式会社リズムファクトリーさんが運営されている「Smart - Web Magazine」内のPerl講座[注1]というWebのコンテンツです。とくに2部のPerl言語仕様の各ページに関しては、初心者にも優しい例文と説明で素晴らしいです。このページをプリントアウトして移動中にも教科書

注1) http://rfs.jp/sb/perl

のように何回も読んだり、わからない部分はメモし、調べたり質問していました。

きれいな書き方を覚えたいときは、オライリー・ジャパンの『Perlベストプラクティス[注2]』という本を読みなさい。と言われて、まずはこれを読みました。また、同じく通称ラクダ本[注3]はよりPerlを深く知るために必要ですが、先輩いわく「ラクダ本は最初に読んだらめげるから後で読んだほうがいいよ」との話で、ある程度基本が理解できるようになってから読み進めるようにしました。

それからCPANサーチ[注4]で検索して、FileやDBの処理など実現したいモジュールを探すこと、そしてモジュールのソースを読むこと、と教えてもらいました。

■先輩と一緒にOJT開始

さらに実践的に学ぶため、学習と並行しながら社内製品の一部を開発させてもらうことになりました。先輩が作ったCatalyst製のアクセス解析のサービスの追加機能の部分です。お客様ごとにアクセスログの解析をし、特定のIPを年月日ごとに抽出し、それぞれ日付ごとにフォルダ分けして書き出しなおす、バッチ処理と呼ばれるメンテナンス用のスクリプトです。このログ解析というテーマには、Perlの基本的な動作がすべて網羅されています。たとえば、ファイルの読み書き・正規表現・XML/JSON/YAMLファイルのパース(構文解析)などです。まずは先輩のコードリーディングを始めました。それから、お手本となりそうな先輩のスクリプトのコピーに細かくコメントを入れることを始めました。コードがまったくわからなくて泣きそうになりましたが、1つ1つ調べていき、わからなければ先輩にまとめて質問していました。

■毎朝の5分間プログラミング

毎日5分以内で、PerlでFizzBuzz問題や、JSONファイルを読み込んで特定の項目を取り出すスクリプトなど、先輩からテーマを与えられタイムアタックのようなプログラミングの練習をしました。ルールはリファレンスを何も見ないで書くということが条件で、最初は全然できませんでした。できない部分は調べて何度もやり直ししました。標準関数を使うパターンと、CPANモジュールを使うパターン、平均や合計を計算するパターンなど、いろんなバリエーションを付けることによって、常に新しい感覚で取り組むことができます。この5分間プログラミング練習を数週間続けたころには、徐々に簡単なテーマをクリアできるようになってきて、非常にうれしかったのを覚えています。これなら1日何回でもできますし、楽しくゲーム感覚で学べるのでお勧めです。

リスト1はPerl特訓中に作ったスクリプトで、コマンドラインから実行をしてread、writeなどコマンドを入力すればhoge.txtにファイルを書き出したり、読み込んだりできるものです。

■CPANを使って応用プログラミングを開始

先輩にCPANモジュールを使えばこんなことができるんだよ、といろいろと面白そうなモジュールをいくつか教えてもらいました。たとえばコマンドラインから電卓のように計算するスクリプトやLWP::USerAgentを使ったGoogleの検索をするスクリプトなど、応用としてできそうなテーマでペアプログラミングを行います。

少しずつ言語に対する理解ができていく中で、教えてもらった内容を基に自分でも構造を少し変えて動かしてみたり、工夫ができるようになってきます。ゼロからのよちよち歩きから数ヵ月が経ち、気づけば完全にPerlの世界に夢中になっていました。Perlと並行してざっくりとVim、GNU Screen、UNIXシェルも使えるよ

注2) Perlベストプラクティス、Damian Conway(著)、クイープ(翻訳)、オライリー・ジャパン、2006年8月

注3) プログラミングPerl〈VOLUME1〉、〈VOLUME2〉、ラリー・ウォール、ジョン・オーワント、トム・クリスチャンセン(著)、近藤 嘉雪(翻訳)、オライリー・ジャパン、2002年9月(第3版)

注4) http://search.cpan.org

▼リスト1　fileio.pl

```
#####################################################
#fileio.pl
#####################################################
use strict;

use warnings;
use utf8;
use English;

use Encode;
use Smart::Comments '###';

use Readonly;
use Cwd;

binmode(STDERR, ":utf8");

my $COMMANDS = {
  'write' => \&file_write,
  'read'  => \&file_read,
  'match' => \&file_read_with_match,
  '?'    => \&print_help,
  'help'=> \&print_help,
  'anon'=> sub{ print "aon\n"; },
};


print STDERR "コマンドを入力してください\n";
while(my $str = <>){
  # $str
  chomp $str;
  print $str, "\n";

  #入力した文字列(コマンド)が$COMMANDSに定義されているものかチェックする
  if($COMMANDS->{$str}){
    #コマンド名に応じた関数を実行する
    $COMMANDS->{$str}();
  } else {
    last  if ($str eq 'exit' or $str eq 'EXIT');
  }
  print STDERR "コマンドを入力してください\n";
}

sub print_help{
  print "write, read, match\n";
}

##
# ファイルを開いて、ファイルから一行ずつ読み込んで、標準出力(STDOUT)へ出力する
##
sub file_read{
  ##read001
  open(my $file, "<", "hoge.txt") or die $OS_ERROR;
  #ファイルから一行ずつ読み込んで
  while(my $str = <$file>){
    chomp $str;
    print $str, "\n";
  }
  return 1;
}
##
# 標準入力から一行受取って、ファイルを開いて、ファイルの最後の行に追記する
#
sub file_write{
  ##write001
  print STDERR "書き込みたい内容を入力してください\n";
  my $str = <>;
  chomp $str;
  ##write002
  open(my $file, ">>","hoge.txt") or die $OS_ERROR;
  print $file $str, "\n" ;
  return 1;
}
##
# 標準入力から一行受取る
# ファイルを開く。
# ファイルから一行ずつ読み込む
# 行頭が標準入力の文字列と一致する行のみ、ファイルから、標準出力へ出力する
##
sub file_read_with_match{

  #標準入力から一行受取る
  print STDERR "検索したい内容を入力してください\n";
  my $stdin_line = <>;
  chomp $stdin_line;

  #ファイルを開く
  open(my $file, "<", "hoge.txt") or die $OS_ERROR;

  #ファイルから1行ずつ読み込んで
  while(my $file_line = <$file>){
    chomp $file_line;

    #行頭が標準入力の文字列と一致する行のみ
    if ($file_line =~ m{^$stdin_line} ){
      print $file_line, "\n";
    }
  }
  return 1;
}
```

うになっていました。最初はなかなか覚えられず何回も不安になりそうですが、「一見遠回りのように見えて習得の近道です。けもの道のような最短コースを進んでいるから大丈夫」と励ましてもらいながらできるようになりました。

転職してからも独学でPerlを勉強する

その後Twitter経由で当時の上司に誘われて転職することになります。PHPで書かれている社内向けのシステムを開発する部署に配属になりました。せっかく教えてもらったPerlの学習をここで辞めてしまうのはもったいないと思い、仕事が終わってから1人でPerlの勉強をしてブログを書いていました。もしかしたら日々の業務の不満のようなものを発散するかのように、家でコードを書いたりブログにぶつけていたのかもしれません。

その頃から上司の理解もあり、社内や社外の方を集めて自主的に集まってプログラミングをする、スマイリーハッカソンという勉強会を始めました。有志の人を集めたかったので社内の人達に説明してみたところ、同じチームの人には全然興味を持ってもらえなかったのですが、他の部署やネットを通して参加してくれる人がいてくださったお陰で何年も続けることができました。

■ブログがきっかけでPerlで新しい社内サービスを作る

Perlのことばかり書いてるブログを上司が読んでいて配慮してもらったのか、Perlのプロジェクトをやろうと言ってもらい、1人で新規のWeb情報管理システムを作ることになりました。一からシステムを作ったことがなかったので、要件定義から設計、実装、テスト、ユーザトレーニングまで一貫して行うことができ、本当に勉強になりました。当時のイケてるフレームワークは、Catalystがスタンダードな流れになっていましたが、いろいろと制約がありCGI::Applicationを使うことに決めました。CGI::Applicationは、もう何年も前に流行が終わっていて日本語では古い情報しかないことや、最新情報やマニアックな拡張部分は英語のドキュメントしかないということがたいへんでした。

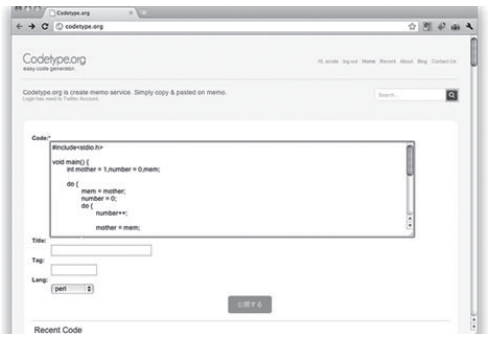
今まで体験したことがないことばかりで、結構時間がかかったりつまづくことが多かったです。そして社内にPerlの質問できる人が少なく、ほとんど孤独な状態です。たまにIRCで社外のできる先輩エンジニアにいろいろと質問したりしていました。

ほぼ一晩で作ったサービスがMashup Awards 6で受賞するまで

きっかけはプレゼン発表のネタとして作成

フリーランス時代にお世話になっていたMashup Awardsの運営の方から声をかけてもらって、あるイベントで発表することになりました。とくに個人として何も代表的なサービスがなかったので、せめて何か新しいことを勉強してその共有でもしようと思い、Pythonの軽量フレームワークFlask[注5]のドキュメントを読みはじめました。するとその実装するコードのシンプルさ、ドキュメントの美しさとわかりやすさに、こんなに簡単なしくみでできるのかと感動しました。さらにフレームワーク自体のコードを読んでいくと、非常にきれいなコードでいっそう感動しました。

▼図1 http://codepad.org/ （※現在はサービスを停止中）



■コードに感動し、高いテンションのまますぐ行動

すぐにレンタルサーバにインストールを始め、すぐにサンプルも動かすことができたら作りたいテーマを決めました。なんとなくcodepad.orgのようなコードをコピー&ペーストできるサイトを作ってみようと思いました(**図1**)。これは当時Twitterでログインできる OAuthの機能を使ったWebサービスがメジャーになってきており、OAuthを使ってみたいということと、機能を絞ったシンプルなものが作りたいという考えがあったからです。

そしてジャストアイデアなので、あまり長期に渡って作ってしまうとモチベーションが下がってしまうかもしれないという思いもありました。反省として、長い時間をかけすぎて最終的な形まで落とし込めないことが過去に何度もあり、荒削りでもいいので早く形にすることを心に決めました。

Pythonは以前からDjangoやmoinmoin wikiやTracをサーバにインストールしたり、好きで軽くコード書いたりしていたくらいのレベルです。FlaskのJinja2というテンプレートの書き方も、Perlでいう Template Toolkit(通称

注5) http://flask.pocoo.org/

TT)とさほど変わらないもので簡単でしたが、独自の機能は調べながら実装しました。

■ライバルが少ない中で選ばれる

金曜日の夜にドキュメントを読み始めて夢中になり、エクステンションのインストールや、テンプレートも決めて、データベースの設計をして、朝方にはざっくりとできていました。

Mashup Awards 6の公式APIではなかった小飼弾氏(@dankogai)のLLEvalという、ブラウザでLLを実行ができるAPIを使わせてもらって、簡単なデモとまとめを発表しました。それから数ヵ月後に発表がありましたが、直前で小飼弾氏が特別審査員となり、LLEvalも公式APIになっていました。運営の方と提出する約束をしていたので、約束どおり登録したら特別審査員賞に選ばれてビックリしました。

他の参加者のかなり気合が入った作品を見ていると、ほぼ一晩で作ったような雑なものがこんなことになるなんて……と少し申しわけない気持ちになりました。「プログラマの三大美徳の第一、laziness(怠慢)に最も忠実な本サイトを推す次第」というコメントをもらいました。

プログラミングのブレイクスルー

プログラミングをしていくと「一気に理解ができた！」という瞬間がやってきます。ふと振り返ってみると、なんとなくできていたのかなぁというパターンが多い気がします。よくよく考えると実際はそこまで理解していなかったりするのですが、理解が深まっていることはたしかでうれしい気持ちになります。

今でも覚えているのは、プログラミングを始めて3ヵ月くらい経ったころのことです。個人で掲示板やいくつかのお問い合せフォームのようなものを作ったりするくらいになっていました。それまでは読んでいてもまったく理解できなかった本が少し理解できるようになったのです。それでも当時在籍していた会社の上司に「横山さん、(開発のスピードが)遅い」と厳しく指導されたり、できない自分が悔しくてたまらない時期でもありました。

効果的な学習方法について

学習と書いていますが、周りの技術者は独学でプログラミングを始められたという人が多い気がします。筆者はある程度のラインまでは、確実にほかの人よりも多く教えてもらってプログラミングを覚えました。たくさんの本を読む、コードリーディングはしている前提で話を進めます。振り返ってみて効果があったのではないかという方法をいくつかピックアップしてみます。

■ペアプログラミング

教育として2人1組でするペアプログラミング[注6]は教える側のパワーや教材が必要ですが、非常に効果があります。とくに上司や先輩とペアプログラミングをすると、シェルスクリプトからエディタの使い方まで目で覚えることができます。historyコマンドで先輩が打ったコマンドを後でメモをするといったことができます。

■ブログでのアウトプット

ブログでのアウトプットも効果的だと考えています。備忘録として整理できて、ちゃんと調べてからまとめるという作業をするので、さらに理解が深まります。たまに知りたいキーワードで検索したら昔の自分のブログのエントリがでてくる、ということもあるでしょう。

もし万が一間違っていても、誰かが教えてくれたりツッコミが入ります。地道に続けていくとネットだけでなく実際に会う方や同僚も応援してくれたりします。本人はこんなことは当たり前だろうと思っても、他の人にとっては有益で役に立つ情報ということだってあり得る話です。まずは自分のために気づいた内容をまとめ

注6) 参照:Wikipedia「ペアプログラミング」http://ja.wikipedia.org/wiki/ペアプログラミング

てみてください。アウトプットの内容によって、その時の自分の理解度を確認できます。

■人に教えるということ

ブログ同様に、人に教えるということは効果があります。人に何かを教えようと思ったら、ある程度まで知識が必要なので細かいところまで調べたりします。そして一連の学習ステップを考えたり、人によって教え方を変えたりする必要が出てきます。

筆者もなるべく率先して人に教えるということをしてきました。なるべく最初から楽しさを感じてもらえるように工夫することが多いです。これは最初にプログラミングの本当の楽しさを教えてくれた先輩の影響が強いかもしれません。事前に興味のあることをヒアリングして、ちょっと興味を持ってもらえるような内容を挟んだり、こういうこともできるんだよという提案もできます。

実際に社内で勉強会を提案して始めたり、プログラミングに興味がある人にプログラミングを教えたり、社外の勉強会を主催して発表したりしました。論理的に説明することを繰り返していくと、より思考が洗練されてさらに理解が深まります。また想定外の質問をされたり、わからない部分を見つけたりすると新しい発見があります。

Twitterからのオフ会デビュー

ちょうど大阪から東京に転勤してきた2007年にTwitterを始めました。当時筆者の知っている範囲で、Twitterを使っている人はWeb系のプログラマやデザイナ、ディレクター、ライターといった職業が多かったです。もともとオフ会という存在は気持ち悪いと思っていたし、ネットがきっかけで知らない人と会ったことがないので、なんとなく怖いイメージばかり。なぜか最初にTwitterで仲良くなった人は、たまたま深夜のテレビ番組に出演するというので観てみました。軽いドキュメンタリーのような内容で、顔も名前も仕事内容もわかったので後日実際に会ったら、すぐ意気投合しました。それからいろんな方を紹介してもらったりしている内に、勉強会に参加も始め、ブログも始めました。都内でWeb系の人達がよく集まって、週に何回もTwitterやはてな経由のオフ会という名の飲み会がありました。インターネットのおかげで一気に世界が広がり、気づけばオフ会というものに抵抗がなくなっていました。

■とある天才プログラマの1日

一部の界隈でオフ会が頻繁に行われていたころ、Twitterがきっかけで天才プログラマと呼ばれている方と知り合いました。その人ともいつの間にか仲良くなって、飲み仲間みたいな感じで、いろんな技術者を紹介してもらったりもしました。朝早く起き、夜遅くまでネット上の情報をチェックしたり、プログラミングをしているストイックなタイプの方でした。

どのようなプロセスで成長されたのかが知りたくて、過去の話や、仕事に対する考え方、1日の過ごし方、読んだほうがいい本を聞いたり、アドバイスをいろいろもらったりしました。ポール・グレアムの著書「ハッカーと画家」を教えてもらったものの、「なんか文字がビッシリで小さいですね！　難しそう！」という斜め上のリアクションを当時はしていました。天才と呼ばれる人でもオライリー・ジャパンをはじめとしてたくさんの本を読み、1日のいろんな時間を削り、見えない努力によって結果を出されていることを知りました。そんな生活を知り、その方のプログラミングの思想に多少影響を受けているのかもしれません。

Perlコミュニティとの出会い

会社で働きながら1年間独学で勉強し続けていた頃、そして自然とPerlの勉強会というものに参加するようになりました。勉強会に参加すれば女性が少ないお陰で、顔と名前をすぐ覚えてもらえました。その前後で尊敬するエンジ

ニアの方々とも知り合い、さらに強い憧れを抱きます。YAPC::Asiaではゲストでいらしていた海外の方とも仲良くなり、日本を案内するという異文化交流ができました。

■Perlコミュニティでの発表

ブログを書いていると、Yokohama.pmを主催されている方に声を掛けていただいて、Yokohama.pm Tech Talk #2で発表したのが始まりです。プレゼンの内容はTwitter APIを使わずにWWW::MechanizeでTwitterにログインし、MeCabを使って名詞の利用頻度を抽出するという謎なものです。発表直前にPerl製のプレゼンツールごとMacBookが動かなくなって、マシンが不安定で頭が真っ白になり、声はずっと震えて時間もオーバーする悲惨なものになりました。誰も本当にフォローができないくらいひどく、残念なLT(Lightning Talk)デビューとなりました。そのあと別のブログカンファレンスでリベンジを果たし、少し褒められて調子に乗り、Shibuya.pmやYAPC::Asiaでも発表させていただきました。Shibuya.pmは当時の上司の業務命令です。よくよく考えれば受け身な感じなのですが、声を掛けて頂いたら基本は断らないというスタンスで発表していました。

ちょっとずるい学習方法

■積極的に質問して教えてもらう

持論なのですが、会社・学校で一番できる人に教えてもらうのが手っ取り早いです。技術職の人は教えたがりな人が多いのはご存じでしょうか？　懐に飛び込んで、可愛がってもらうのも1つの手です。もちろんずっと教えてもらうばかりだとウンザリされるので、なるべく自分で調べられることは調べて、当たり前のことは聞かないようにします。当然のように一度聞いたことは、必ずメモをして同じことを質問しないようにします。逆の立場になって「あなたはどのような人に思わず教えたくなりますか？」と質問すれば自ずと答えが出てくるかと思います。言い方は良くないかもしれませんが、相手の時間を奪ってでも成長したいという気持ちで挑むと丁度いいかもしれません。そして自分が成長し続ける姿を見せて、まだまだ成長できる伸びしろがあることをアピールすることも重要でしょう。

■いろんな人に応援してもらう

ブログをがんばって更新していると応援してくれる人や、勉強会を主催した際にいつも来てくれる人が増えてきます。メールで問い合わせをいただいたり、知らない方からTwitterやFacebookで温かいメッセージをもらうこともあります。そういったメッセージや声をかけてもらうと非常にうれしいですし、いろんなときに励みになります。

■思い切って転職する

いろんな価値観を持った人と人が一緒に仕事をしていると、わかりあえなくなったり、愚痴を言ったりやる気がなくなったりするときもあるかもしれません。もし、より良い業務内容や環境(オフィス・人・待遇)や、より成長できそうな会社で働くチャンスあれば、思い切って転職してみるのも1つの手です。個人的には仕事を通して学んだり得られることが多いので、環境というのは本当に大事な要素だと考えています。より情熱を持って取り組むことができる職場や、仕事内容を探すことは決して悪いことではないと思います。持てるリソースを最大限に活かして、会社という組織を最大限に利用して成長するんだという気持ちで挑めばきっとうまくゆくでしょう。

モチベーションの維持

■ライバルや目標を見つける

ネット上で知り合う人達や、会社の同期や先輩でもいいので、心の中で良きライバルや目標となる人を探してみてください。それぞれ立場が違いますから、他人と比較して落ち込む必要はありません。ライバルが頑張っている姿を見

て負けないように、そして目標とする人に近づけるよう、イメージを具体的に描いてみましょう。

■ごほうびの設定

さらに筆者の場合は、物理的なごほうびを設定することをしています。これが終わったら○○を買う、といった自分へのプレゼント的なものから、○○にゴハンを食べに行く……など、好きな食べ物でもいいでしょうし、温泉や旅行、マッサージなどいろんなごほうびを考えてみるのも1つの方法です。

継続して学ぶために

筆者が以前働いていた美容室のトップディレクターが言っていたことがあります。

「器用な人はすぐ上達するので(すぐ満足して)辞めてしまう。不器用な人は何回してもできないから長く続けるんだ。なぜなら僕もすごく不器用で、一番できなくて怒られていた人間だからだ」

世の中には最終的に才能がある人よりも、不器用でも続けた人のほうが上手になることが実際にあります。

■今の先に何があるのか

いったい今の仕事の先に何があるのかというイメージを膨らますことが重要だと考えています。たとえば、共通のモジュールやメソッドを作っておけば次回から楽ができる、Webサービスを1つ作ったら工夫した点をまとめて発表できる、など。目の前の仕事だけを見ると悲観的になりがちですが、長期的な目で見るとメリットがいろいろあると考えられます。1つできると、仕事の中に新しい発見や副産物ができます。その小さな成功体験を積み重ねていくことが自信につながるはずです。

■定期的な振り返り

たとえば半年や1年が経って、今進んでいる道が正しいのか不安になる時もあるかもしれません。そういう時にブレたりしないように、月1回など定期的に振り返り、目標を確認することをお勧めします。大事なことは、ダメだったことを繰り返さないことです。

まとめ

今回は基本的な考え方、行動をまとめてみました。いろんなアイデアを実現できるプログラミングは、本当に楽しい仕事です。プログラミングを学ぶためには必ず学校に行かなくても、いくらでも成長できます。「Learning by fire(火災から学ぶ)」という言葉があります。火事に飛び込んで学ぶ、すなわち実践から学ぶという意味です。できないことも実践すればできるようになります。そして今結果を出している人は、過去に見えないところで努力をしている人ばかりです。ですが必ず先輩方が通った同じ道を通る必要はなく、効率よく学習できる部分はたくさんあります。

ここに書いていることは特別なことではなく、誰にでも同じようにチャンスがあり、手を挙げるか挙げないかの差でしかないです。情熱を持って取り組むことができる技術や言語やライブラリ等、好きなことを見つけて実際に触れてみましょう。それをWeb上や仕事でアウトプットしてみてください。無駄なことはなに1つありません。そして、いつかこの記事が役に立ったと言ってくださる人が1人でもいたら幸いです。

SD

Part 3

タワレコ店員からエンジニアへ

小林 徹　KOBYASHI Toru　株式会社モバイルファクトリー
Mail:koba0004@gmail.com　Twitter:@koba04

未経験からプロの世界へ

典型的な文系人間

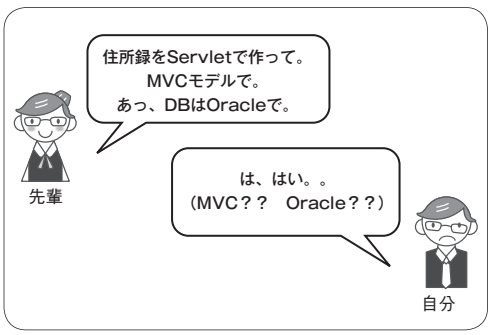
筆者は大学を卒業しタワーレコードで2年ほど働いたあと、未経験のエンジニアとして20名程度の受託開発を行っている会社で働き始めました。そこで4年ほど働き今の会社に転職したのですが、エンジニアになろうと決めたときは数学は苦手、プログラミングはもちろん未経験の典型的な文系人間でした。

独学でスタート

タワーレコードで働いていたときにも独学で勉強はしていましたが、初級システムアドミニストレーター（現ITパスポート）の資格取得と、Javaの初心者本を読んで動物クラスや犬クラスを作っていた程度です。これらを使ってどう仕事を進めていくのかまったくイメージできない状態からのスタートでした。ですが、エンジニアになるために京都から東京に出てきていたので退路はない覚悟でした。

そんな中で考えたことを書かせていただきます。

▼図1　わからない



とにかくわからない

自分で勉強していたとはいえ、入社して最初に与えられた課題は「～がわからない」というのも説明できないくらい理解できませんでした。

図1みたいな状況でしたのでこの先やっていけるのかという不安がよぎりましたが、課題に関係することが書かれていそうな初心者本を買ってきて写経してみたり、先輩に聞いたりインターネットで調べたりととにかく試行錯誤を繰り返しました。

その過程で大切なこと学びました。

何かを試すときは1つずつ

わからないときというのは、焦りや不安からとにかく試せることをなんでも試してしまい、気がつくとどんな対策を入れたのかもわからなくなり、その結果多くの時間を無駄にしてしまうことがよくありました。その経験から、何か

を試すときは1つ試しては元に戻して、また別のことを1つずつ試していくほうが最終的には早く解決できるということを学びました。

■当時の失敗

EclipseやApache/Tomcatの環境設定周りでうまく動作させることができず、闇雲にいろいろと試してよくわからなくなり、再インストールすることがありました。

そのときに、試したことをログに取っておくことの必要性を学びました。

聞くことの重要さ

聞く準備をする

とくに最初は、時間が許す限り悩み試行錯誤することが自分のためになるのでオススメなのですが、仕事の場合はお金をもらっていますし、周りに迷惑をかけてしまう場合は早めに質問をして解決することも重要です。

また、質問するということは自分の時間だけでなく相手の時間も使うことですので、質問する前には自分で試したことや知りたいポイントなどをまとめ、相手にこう答えてほしいというイメージを持って質問をすることが大事だと学びました。

■当時の失敗

聞きたいことを説明しているうちに自分の勘違いや答えがわかり、聞いた相手の時間を無駄にしてしまうことがありました。

考えることを放棄しない

考えることを放棄してわからないことがあったら反射的に質問するというのは、解決までの時間は早くなるかもしれませんが、答えだけ得られて理解ができない状態になってしまうことも多いので、自分のためにも避けたほうがいいと考えています。

空気読む

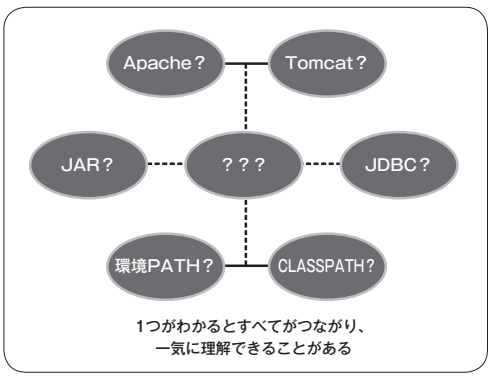
あとこれが一番難しいのですが「空気読め」というのも重要です。今このタイミングは質問するタイミングなのか、質問する前に一度考えてみるといいのではないでしょうか。

すべてをわかろうとしない

未経験で基礎的知識や経験がない状態ですべてを理解しようとすると破綻してしまうので、今の時点ではこういうものだと割り切って進めていくことも重要であると考えています。

勉強を続けていればふとしたときにいろいろつながって、もやもやしていたことが一気に理解できることがあるものです(**図2**)。それは勉強をしていてとても達成感を感じる瞬間でもあります。

▼図2　すべてがつながり理解できる



勉強する

3つの軸

勉強する内容について筆者は、次の3点を軸として考えています。

・業務に必要なこと
・基礎的知識
・業務に関係ないけど興味のあること

基礎的知識

その中でも基礎的知識は、アルゴリズムなどのほか、Web系でいうとTCP/IP、HTTP、Linuxなどシステムを構成する基盤的な技術を指しています。これがあるかどうかにより、新しいことを学ぶときに理解できるスピードが大きく変わってきますのでコツコツ続けていくことが大事だと考えています。ただ、ついつい後回しになってしまうのですが。

■英語と数学

また筆者は最近、英語と数学という学生時代に苦手だった分野の必要性を痛感しています。足りないことで可能性を制限してしまうことになるのでこちらもコツコツ続けていくことが大事だと考えています。

踊らされない

この業界は学ぶことが膨大なため、何でもかんでも手を出さないことの大事さも身を持って学びました。とくに「知っておくべき～」みたいな記事に踊らされないようにすることも重要であると考えています(笑)。

本で勉強する

■初心者本で学ぶ

本に投資することは無駄にはならないのでどんどん購入するといいと思っています。

どういった本を買うかということですが、筆者は新しいことを学ぶときには初心者本から取り組むようにしています。初心者本にあるような「変数とは？」といった説明は今さらで無駄な部分はありますが、一気に中級者向けの本に取り組んでたくさんの「？」を増やすより、初心者本で最初から学んでいったほうが結局理解が早いのではと考えています。

■Webの記事だけでOK？

また最近はWebにもたくさんの「～入門」といった記事はありますが、筆者の場合はそれだけだと結局しっかりと理解できないことが多いので、Webでそういった記事を読んで興味を持ったらすぐに本を買ってみることにしています。

また、本を買うことで早く読んで学びたいという気持ちになれる効果もあります。

資格を使う

資格については議論のあるところですが、とくにIPAの試験については、筆者のような基礎的な学習をまったくしていない人間にとっては効果的であると考えています。仕事に直接的に役に立つといったことはあまりありませんが、前述した基礎的知識と同様にちょっとしたときにヒントとなることがあります。

■目的ではない

ですが、資格取得は目的ではなく勉強する手段として使うものだと考えています。

筆者自身セキュリティに関する仕事に携わっていたときに、セキュリティについて体系的に勉強したいと思い「情報セキュリティスペシャリスト」の資格取得を通じて勉強しました。

きっかけは勉強会

最初の会社で何年か経つとコードを書く機会が減り、メールやExcelを開いている時間が増えていきました。そのときにこれまで書いてきたコード量の少なさに危機感を感じ、業務でもコードを書けるよう調整しました。そこで使った言語はPerlでした。

そして偶然知った「Shibuya.pm #12」に参加してみました。話されている内容はほとんどわかりませんでしたが、miyagawaさんのTalkで画面に映し出されるコードの美しさに衝撃を受け、もっとプログラミングをやりたいと考えるようになりました。

これをきっかけに、もっとコードを書いたりブログを書いたりコミュニティに参加していこ

うと考えるようになり、その結果いろんな方と出会うことができ、さらに刺激を受けるようになりました。

まずは参加してみる

勉強会はモチベーションと知識を得られるので、興味のある勉強会を見つけたら積極的に参加してみるといいのではと考えています。勉強会という場は意識の高い人たちがいっぱいだからと恐れる必要はないですし、最初は知識を吸収するためだけに気楽に参加してもいいと思います。

▼図3　タワレコ店員からエンジニアになって思ったこと

（http://koba04.com/slide/become_a_programmer/）

そこからつながりが

また、懇親会で話しかけられなくても、ブログで感想を書くことで発表者の方からリアクションをもらえ、つながりができることもあります。筆者も参加した勉強会の感想などをブログで書いていたところ、それを見ていたyusukebeさんに声を掛けていただき、YAPC::ASIAの前夜祭でLTをさせていただくことができました。

ブログを書く

書かない理由はない

自分のブログを作りそこに日々学んだことを書くというのは誰でも簡単にできて効果も大きく、やらない理由はないのでオススメです。

学んだことをそのままブログに書いてみるだけでも自分が理解できているのかわかりますし、アウトプットする過程で知識の整理もできるのでオススメです。

誰かのためと自分のため

何かハマったときに誰かのブログに答えをもらったことはありませんか？　自分がハマったことや学んだことを書いておくと他の人の役に立つことができます。

また筆者は一度学んだことでもよく忘れてしまうので、未来の自分のためにも学んだことはできるだけ書くようにしています。

チャンスが

ずっと書いていることで自分という存在をアピールできますし、思わぬチャンスを得られることがあります。

今回執筆させていただいているのも「タワレコ店員からエンジニアになって思ったこと」というスライドを書いてブログに公開したことがきっかけです（**図3**）。

サービスを公開してみる

まずは仮想環境

最初は自分のPCに仮想環境を作りUbuntuやCentOSをインストールして、好きに試しては壊してを繰り返すことからはじめてみるといいと考えています。perlbrewやrbenv、MySQL::Sandboxなど、プログラミング言語やDBの環境を気軽に作ることができるライブラリを利用するのも簡単に試せるのでオススメです。

VPSを借りる

そして、慣れてきたらぜひVPSを借りてみることをオススメします。公開されている環境があるというのは何かと便利でもありますし、何か作ろうというモチベーションにもなります。ドメインも取得するとさらにやる気が出ると思います。

公開しない理由はない

botでもなんでもサービスを公開し運用してみるというのは思った以上にいろいろな作業が必要で面倒なものです。

ですが、実際にやってみるとその面倒な部分で勉強になることはとても多いので、何か思いついたらセキュリティや社会的に問題のないものであればぜひ公開してみることをオススメします。それが、自分のためだけのサービスでもかまわないと思います。

リリースしてみる

筆者も1年に1つは個人でサービスをリリースしたいと考えており、これまで「CountDown LastFMRanking」というものや「PetaTube」などをリリースしました。これらは自分のあるといいなという思いから作った自分のためのサービスです。

ソースはGitHubにあるので見てもらうとわかりますが、どちらも小規模なものです。

▼図4　CountDownLastFMRanking

（http://cdlm.koba04.com/）

▼図5　PetaTube

（http://petatube.koba04.com/）

・CountDownLastFMRanking

日本やアメリカのLastFMのランキングの曲をYouTubeから取得してカウントダウンで連続再生するサービス（図4）。

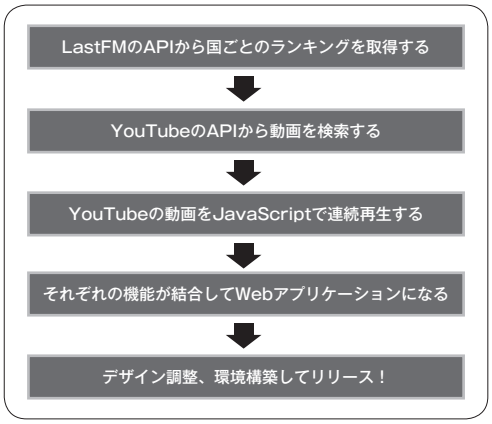
・PetaTube

YouTube動画があるページのURLを指定すると、そのページにあるYouTube動画すべてを連続再生してくれるサービス（図5）。

途中に達成感を

とはいえ、仕事後に家で開発してサービスを公開するというのはたいへんに感じることもあります。筆者は何か開発するときには、モチベーションを保つため小さな達成感が途中で得られるようにと考えています。

▼図6 「CountDownLastFMRanking」の完成まで



■「CountDownLastFMRanking」の場合

「CountDownLastFMRanking」の場合はまず、LastFMのランキングをAPIから取得しその結果を発言するTwitterBotを作りました。そのあと、HTML/CSS/JavaScriptでYouTubeを検索してその結果を連続再生するページを作りました。そして、それらを組み合わせて「CountDownLastFMRanking」としてリリースしました。このように、途中で動くものを作ることで達成感が得られ、続きを作ろうという気持ちになれました(**図6**)。

好きな技術を使えるおもしろさ

また、個人でサービスを作ることの醍醐味として、好きな技術を使いたいからという理由で使えるということもあります。業務だと当然検証したうえで導入することが求められますが、個人で作る場合は誰にも遠慮することなく導入できます。

■「PetaTube」の場合

「PetaTube」の場合は、そのとき使いたいと思っていた「Backbone.js」というJavaScriptのフレームワークと、「Compass」というCSSのフレームワークを使ってみたいという理由だけで使いました(**表1**)。そうすることで作るときのモチベーションも高くなるので使ってよかったと思っています。

▼表1 PetaTubeを構成する要素

Web Server	Sakura VPS Ubuntu
Web Server	Nginx
Application Server	Starlet
DB	MySQL
Cache	Memcached
Process Control System	Supervisor
Programming Language	Perl
Web Application Framework	Amon2::Lite
HTTP	Furl,Coro
CSS	Compass
JavaScript	jQuery,Backbone.js
Build Tool	Grunt.js

無理しない

実は一番言いたいことでもあるのですが、無理しないことはとても大事であると考えています。勉強しなければいけないという思いから無理にやってしまうと苦しいという印象が残ってしまいます。

筆者の場合、プログラム書くのに疲れたときは本を読み、本を読むのに疲れたときはプログラムを書き、どちらも疲れたときは飲んだり寝たり好きなことをして過ごすようにしています。

やる気がでないとき

また、モチベーションが下がってしまうこともあるものです。そういったとき、筆者は勉強会に行ったり、すごい人のインタビュー記事なんかを読んだりしてモチベーションを上げています。

エンジニアってたいへん?

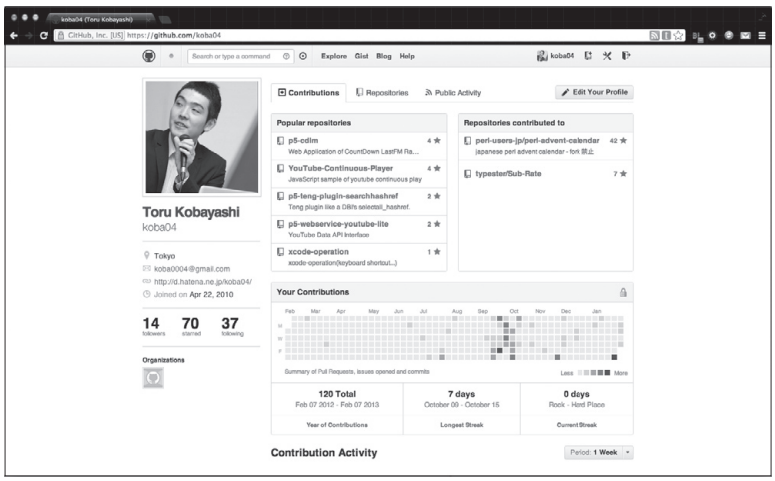
息をするように学ぶ

エンジニアだけに当てはまることではありませんが、一生学び続ける覚悟が必要であると感

じています。ですのでいかに学ぶことを楽しむかが重要だと考えています。楽しかったり目的がはっきりしている勉強は苦にならないと思います。

理想を言えば、まるで呼吸をするように学ぶ、つまり自然に学べて、吸うこと(インプット)と吐くこと(アウトプット)をバランスよくできればと考えています。

▼図7　GitHub



アウトプットする

今は会社の力を借りなくてもたいていのサービスなら個人で作れますし、個人でコード書いているかはGitHubを見ればわかります。そんな言い訳のできない状況ですので、しっかりと学びアウトプットしていくことが重要になってくるのではないかと考えています。

慣れたときが危険?

仕事に関しては、これは自分への戒めですがリスクを冒さないとチャンスはないと考えています。つまり、仕事に慣れて持っている知識である程度仕事をこなせるようになってきたときが一番大事であると考えています。そこで満足せずに新しいことにチャレンジできるかどうかが大きく成長できるかどうかの分かれ目になると考えています(筆者自身もこの点は甘えがあるので日々反省です)。

スタートの差?

また、周りのエンジニアの技術力が高くて焦りを感じることもありますが、技術力を身に付けることに近道はないので、手を動かして学んでいくしかないと考えています。自分はエンジニアになるのが遅かったからと思っていたこともありましたが、すごいなと思う人たちを見ていると、普段のコード量も勉強量も筆者より多くてどんどん差が広がっています。つまりスタートの差ではなく日々のコード量や勉強の差だと痛感しています。

「何ができるか」ではなく「何をやったか」

ですがたいへんなばかりではなく、未経験からエンジニアになってみてこの仕事や業界って本当に面白いなぁと感じています。自分の力でサービスを作り世の中にリリースできることってカッコよくないですか?　筆者自身は未経験からこの業界に入ってきて、技術的にもまだまだ足りない部分ばかりですが、

「自分しかできないことはほとんどないけど自分しかやらないことはある」

と考えています。「何ができるか」ではなく、「何をやったか」が大事だと思いますので、できることは積極的に表現していきたいなと考えています。

とはいえ、無理をせずに楽しむこと、楽しみためには積極的にアウトプットしていくこと、これが大事なことだと筆者は考えています。SD

Part 4

文系書店員→スタートアップエンジニアへの180°転回

Tunnel株式会社
高橋 弘　TAKAHASHI Hiroshi

はじめに

みなさん始めまして。高橋と申します。私はもともとは典型的な文系で、大学の文学部を卒業後、某書店の外商部門に新卒入社。大学や研究所に本や電子ジャーナル、学術系データベースなどを売る仕事に従事しておりました。

その中でプログラミングに興味を持ち、現在はベンチャーでスマートフォンアプリの開発などをやっております。今回は「裏口からのプログラミング入門」ということで、その過程について書かせていただければと思います。

前職時代

初期状態

周囲のエンジニアの方々は、高校や大学時代からプログラミングをやっており、実際にサービスを作ったりもしていたという人が多いのですが、私の場合は大学の実習で簡単なHTMLを習った程度で、プログラミングの知識はなしといった状態でした。ネットはよく見ていたというくらいでしょうか。とにかく大学を卒業した時点で、下積みとなるスキルはほぼゼロでした。

もともとそんな状態であり、さらにITとは真逆の業界に身を置いていましたが、入社後しばらくして買ったiPhoneを通じてWebサービスやアプリに興味を持ち、その延長としてプログラミングや技術系の話題にも興味を持つようになりました。EvernoteやDropboxの使いやすさに素直に感動したのを覚えています。実際そのころはよくわかってはいませんでしたが、ゼロから何か生活を変えるようなものを作り出せるプログラミングというものに大きな可能性を感じたのです。

勉強したこと

とりあえずC言語だろうという根拠のない先入観から「猫でもわかるC言語プログラミング[注1]」(**図1**)を買い、当時使っていた古いPCにVisual C++を入れて「printf("Hello World");」と打ったのがプログラミングに触れた最初の瞬間でした。

次に触れたJavaに関しても同様に、入門書を読み、抽象クラスやインターフェースなどよくわからないままにサンプルコードを一通り書いて動かしてみたり、ハードへの興味からPCを自作してみたり、IPAの情報技術者試験を受

注1) 粂井康孝著、ソフトバンククリエイティブ　ISBN978-4797345650

けたりと、そのころはまだ純粋に、それまで知らなかった分野への興味でいろいろやっていたと記憶しています。

その後はサンプルコードではなく、実際に動いてWeb上でほかの人が見ることのできるものが作りたいと思い、ロリポップの一番小さい容量のプランを契約して自作の掲示板サイトを作ったりしていました。そこで初めてPHPやJavaScriptを学んだのですが、最初に買った、たにぐちまことさんの「よくわかるPHPの教科書[注2]」(**図2**)が非常にわかりやすく、MySQLなどのデータベースの扱いを含めたWebプログラミングの世界にすっと入っていけたように思います。逆にJavaScriptについては「独習JavaScript[注3]」の分厚さの前に半ば挫折。JavaScriptは難しい、JQueryは(よくわからないけど)便利という観念を持ちました。

また一方ではAndroidの簡単な電子書籍アプリを作ってAndroidマーケット(当時)に公開してみたりと、自分で打ち込んだものが意図したとおりに動く気持ち良さに、次第に業務時間以外のほとんどの時間をプログラミングに当てるようになっていきました。終業後はもっぱら近場のマクドナルドで作業していたため、このころのことはポテトの油の匂いとともに思い出されます(笑)。

葛藤

会社を辞めてエンジニアとしての方向に進むことを意識し始めてはいましたが、とはいえ文系で未経験分野。ここで辞めてこの先やっていけるのか、そもそも転職できるのか、しかしこのまま今の仕事を続けていていいのか、という葛藤に悶々としていました。

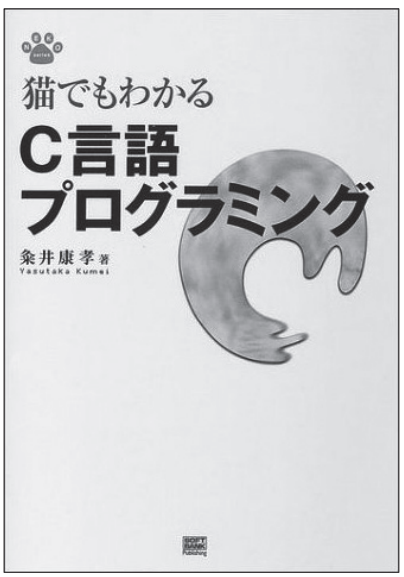
そんなとき、「高学歴文系&職歴なし&30歳でWebクリエイターになる方法」というブログ記事[注4]を読み、同じような文系未経験からWeb系のエンジニアになったということで影響を受けました。「身に付けた技術のすべては自習によるもの」、(ハンデを負った状況から始

注2) たにぐちまこと著、毎日コミュニケーションズ ISBN978-4839933142

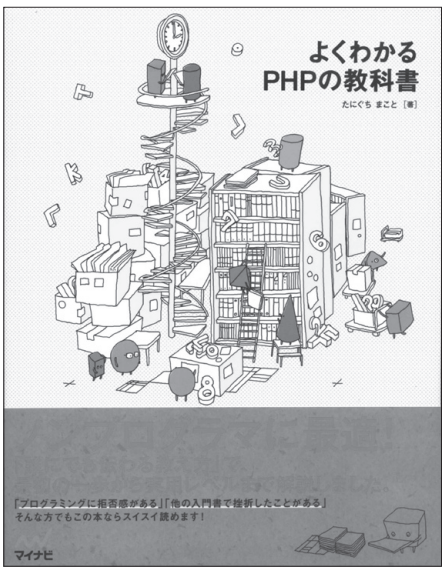
注3) 高橋和也・竹添直樹・里見 知宏著、翔泳社 ISBN978-4798116754

注4) http://takahashifumiki.com/others/480/

▼図1 猫でもわかるC言語プログラミング



▼図2 よくわかるPHPの教科書



めるのであれば)「周囲が引くぐらい勉強する」という言葉がとくに印象に残っています。死ぬ気でやれば何とかなるんじゃないかと思い始めたのはこのころでした。

辞めたのち起業→RoomClipの開発

いろいろ悩みもしましたが、最終的にすでに別の会社を辞めていた仲間の誘いもあって会社を辞め、RoomClip[注5]というiPhoneアプリで起業(**図3**)、スタートアップの世界に飛び込んでいくこととなりました。

体当たり型開発

RoomClipの前にも起業前の段階で比較的簡単なiPhoneアプリは作っていましたが、今回は業務として作る初の本格的なサービス。当初はチーム(といっても最初は2人でした)にほかにエンジニアもいなかったため、ほぼ1人で手探りで開発を進めていきました。リリース時期のこともあり、勉強してから作るというよりも作りながら勉強するという、まさに「体当たり型」のスタイルで、当初はさまざまなハードルにぶつかりました。

オブジェクト指向?

最初はとにかく作ってみるという思いのままに、ネイティブ言語(Objective-C)でコードを書いていきましたが、クラス設計もあまり考えないまま書き始めてしまったこともあり、1つのクラスにコードを長々書くという非オブジェクト指向な状態に陥りました。各ViewControllerで共通して使うメソッドを別クラスにせず、そのまま1つのViewControllerクラス内に書いてしまったり、継承を使わずほとんど同じようなクラスをコピー&ペーストに近い形で2つ作って、変更があるたびに修正に二度手間をかけたりと、非効率的かつ遠回りなやり方をしておりました。

最初の段階でどういう構成にするかを一度立ち止まってイメージしたり、紙に書いたりしてから実際のコードを書き始めるようにすればよかったと今では思います(さらに言えばデザインパターンをもっと勉強しておけばよかったと思います。これは今でも自戒を込めて)。

注5) http://roomclip.jp/

▼図3 iPhoneでのRoomClipの画面

※部屋のインテリアや家具、レイアウトの写真を共有するSNSです。

自力解決の是非

こちらも最近反省していることですが、開発の途中で何かどうしても解決できないことやバグなどがあった場合、ほかの人に聞かずにドキュメントやStack Overflowなどの Web上に落ちている情報を見て、自力で解決することにこだわっていました。自分で調べて解決したほうが理解が定着すると思っていたのかもしれません。それで解決できる場合もあったのですが、一方でその分必要以上に時間がかかってしまうこともありました。

その最たる例がAWS(Amazon Web Services)の設定でした。RoomClipもSNSのアプリということで、当然クライアントサイドだけでなくサーバサイドも構築しなければならないのですが、それまでサーバについての知識はなし。この手のサービスの場合、どういったサーバ構成が一般的なのかということもわからないまま、ひたすら関連の開発ブログやドキュメントを見ては悩んでいました。Amazon EC2とS3とRDSはそれぞれどういうものでどう機能分担するのかという初歩的なところから、実際どれくらいのスペックのインスタンスをいくつ持っていれば想定のトラフィックに耐え得るのか、その場合の金額はいくらになるのか……等々。

結局はAmazon主催のセミナーなどに何度か出席して、Amazon側の方の話や他社の利用事例など、実際の温度感のわかる話を聞くことで理解を深め、何とかサービスインできました(**図4**)。しかしこのとき自力ですべてやろうとしていたら、より多くの時間がかかっていたか、もしくは適切でないやり方を取ってしまっていたかもしれません。

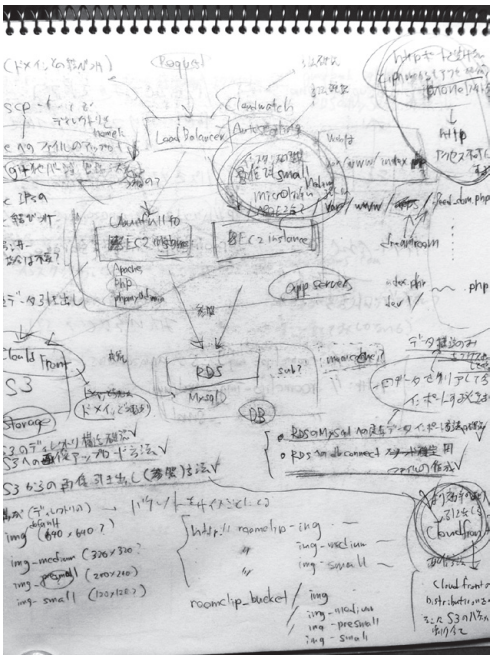
自力で解決することももちろん意味はありますが、わからないところはすでに知っている人に素直に聞いてしまったほうが早いこともある、ということを学んだしだいです。

◆

これ以外にも、Linuxの環境構築やプッシュ通知の実装、FacebookやTwitterといったほかのSNSと連携させる際のOAuthの設定など、苦戦を強いられる場面は多々ありましたが、「今日の自分には無理でも明日の自分にはできるようになってる(かもしれない)」と前向きに考えるようにしつつ、ひとつひとつ積み上げられた課題を崩していきました。よさそうなライブラリやコードは、何でもとりあえず組み込んで試してみる。人よりスタートが遅い分、より多くの時間勉強する、といったことを意識しながらの開発でした。

上で書いたような反省すべき点はほかにも山のようにありますが、1つのサービスを作るにあたって必要なことを一通りやったということで、結果として言語や実装について理解を深めることができたように思います。しかし一方で自分のレベルや未熟さも、よりクリアになった部分があり、今後もまた勉強していかねばという思いにも駆られています。

▼図4　一番最初のサーバ構成図草案



リリース後

アプリのリリース直後はとにかくバグやサーバの稼働状況に非常にやきもきしていました。とくに一番肝を冷やしたこととして、一番最初にAWSのRoute 53でELB(ロードバランサ)のDNS登録をした際、ELBのIPアドレスが時間が経つと変わることを認識せずにIPで指定してしまったため、最初にそれが変わるタイミングで一時サーバにアクセスできなくなる状況になりました。その後該当個所を修正して無事に復旧しましたが、そのときの焦りは、リリース後まもない時期だったこともあり、かなり鮮明に覚えています。リリースはゴールではなく、保守・運用・改善を考えたうえで、あくまでスタートだと思い知った出来事でした。

現在はリリースから約9ヵ月経ち、ユーザ数もまだ4万弱ではありますが、順調に増え続けています。チームのメンバも4人に増え、運営体制を強化するとともに次のステージに向かってサービスの拡大、改善を図る毎日です。

今後やりたいこと

勉強会、ハッカソンに出る

「情報交換できる人を見つける」「会社という枠をはずした自分の現在のレベルを相対化できる」という意味もあり、勉強会などには積極的に参加していきたいと思っています。最近はハッカソンにも参加してみましたが、普段接する機会のない方々の中に混じって制限時間内で何とか形になるものを作るというのは、ほど良いプレッシャーもあり、非常にいい経験になりました。ときには何も成果物ができなかったり、まわりのレベルの高さに気後れしたりすることもあるかもしれませんが、それも糧にするという気構えでまた参加していきたいと思っています。

インプット／アウトプットサイクル

今までやれていなかったこととして、技術書を多読してポイントとなる部分や、そこで実践した結果をブログなどに書いたり、作ったモジュールを公開したりと、インプットとアウトプットのサイクルを細かく回していくことの重要性を感じています。

個人プロジェクト的なこと

基本的には今のサービスを伸ばすことに注力するという前提はありますが、それ以外でも、何が作りたいか、どんなものが世の中で必要とされるかといったアイデアを常に持ち、ときにはそれを形にしてみるという姿勢を持っていきたいと思っています。それにより発想を柔軟にし、またそこで得られた成果をメインのサービスに還元するといった期待もあります。

終わりに

こうしてもともととはまったく異なる、そして昔は予想もしていなかった状況になりつつも何とか生き延びてはいますが、やはりプログラミングを始めるのが遅かったという後発としての焦りと、人と同じだけやっていては追い付けないという切迫感はあります。しかし一方で、自分の作ったサービスが世の中に与えるかもしれない影響の可能性や、書いたコードが自分の思い通りに動いたときの快感、その純粋な楽しさが今後も自分を引っ張っていくモチベーションになるとも思っています。

今はまだ出始めの段階で、この先どうなるかはわかりませんが、今の道に進んで後悔したこともほとんどありません。あとはもう覚悟を決めて遮二無二やるしかないと思っています。同じような状況、境遇の方ともぜひ情報交換したり、一緒に何かやったりできればと思います。SD

Part 5

今すぐはじめよう！脱超初心者からのベストプラクティス

株式会社アイスリーデザイン　http://www.i3design.co.jp
及川 智　OIKAWA Satoshi

イントロダクション

春からプログラミングを始める方もすでにプログラミングを始めている方もこんにちは。はじめまして、アイスリーデザインの及川といいます。

さっそくですが、これから始めるプログラミングにどういうイメージを持っていますか？

・未経験な自分にもできるのだろうか？
・何から勉強すれば良いのだろうか？

といった不安でいっぱいでしょうか？

筆者もそういった不安を抱えながらも、社会人になってから我流でプログラミングを学んできました。もともとプログラミングに興味はあったのですが、まさか自分にできるとは思っていなかったですし、プログラミングすることで何ができるのかわからなかったというのが正直なところです。

そんな初々しいころもありましたが、10年以上プログラミングをしてきた甲斐もあり、今ではバリバリのプログラマになったと自負できます。

万人向けではないかもしれませんが、これからプログラミングを始めたいと思っている方の後押しになれたらと思い、筆者がどのようにしてプログラミングを学び、習得してきたかをキャリアをふりかえりながら紹介していきます。

プログラミング怖くないよ！

プログラミング歴

「それではさっそく！」と行きたいところですが、ちょっとだけ筆者の経歴を紹介させてください。

プログラミングを始めたのは、高専で卒業後、新卒で入社し、システム部署に配属されたことから始まりました。入社当時のプログラミングの知識は、授業でパソコンに触った程度でプログラミングができると何ができるのかすらよくわからないレベルです。配属先も、新設されたばかりで、人員は自分1人！というトンデモ部署でした。ですから、誰かに教わるということはできず、自ら道を切り開かざるを得ない状況でした。

会社からは、「業務用のWindowsアプリケーションを“Visual C++ 6.0”で開発してね」とだけ言われ、そこから筆者のプログラマ人生が始まりました。参考までに筆者が今まで作ってきたソフトウェア、システムは次のようなものです。

・業務用アプリケーション（for Windows）
・業務用スタンドアロンシステム（for Windows）
・業務用クライアント／サーバシステム（for Windows）
・業務用Webシステム
・ASPサービス
・iPhoneアプリ

プログラミング言語でいうと、C++（Visual C++ 6.0）からはじめて、Java（JDK1.3～1.4）、PHP（4.3～5.x）、JavaScriptといった複数の言語でプログラミングしてきました。OSはWindows、Linux、Mac OS上で、開発してきたソフトウェアの対象もWindowsアプリだったり、Webだったりと多種に渡っています。

実は転職も数回しており、

・特定業務向システム開発（1～10人規模）
・Webパッケージ開発（50～100人規模）
・ASPサービス開発（ベンチャー）

といった組織で働いています。

これからプログラマ人生が始まるという時点でイメージするのは難しいかもしれませんが、筆者の境遇と、あなたの境遇に類似個所はありそうでしょうか？ マッチする個所がありましたら自分と重ねて、ない場合でも想像しながら読み進んでいただけたらと思います。

プログラミング入門

先ほども触れましたが、筆者がプログラミングを始めた言語は会社から勧められたC++（Visual C++ 6.0）でした。とりあえずC++という言語を使うんだということはわかったので、それに関する情報をWebなどからググったり（当時はGoogleはないのでYahoo!だったりgooだったり）して何とか情報を集めました。

しかし、何もわからない状態で自分に必要な情報だけ集めるというのは大変なので、書店に行き「Visual C++ 6.0」「入門」というキーワードで自分に合いそうな入門書を1冊買うことから始めました。

筆者の場合はC++言語と指定されたので、ある意味楽だったのですが、みなさんはどの言語を使うか決まっていますか？ もし決まっていないのならザクっと決めてしまいましょう。入門書を1冊やってから、違うプログラミング言語を学んでも良いのですから。ひとまずは、「下手な考え休むに似たり」ということで一歩目を踏み出してみてください。またちょっと遠い将来ですが、複数の言語を学ぶことでよりプログラミングの理解は進みます。

とはいえプログラミング知識ゼロで本を読み進めても頭には入って来ず「お前は何を言っているんだ……」という感じでした。しかし、始めたばかりでわからないのは当たり前ということで、深く考えず入門書を1冊終わらせることを目標に進めました。

書いてあるとおりにコードを入力し、トライ&エラーの繰り返し。知らない用語はたくさんあるし、自分の書いているコードの意味がわからない部分もあります。なぜか「このコードを書かないとたくさんエラー出てしまうから書いておく」といった「おまじない」も大量にあったりしましたが、そんな五里霧中でも自分で入力したサンプルが動くと、「おぉ！ これがソフトウェアってやつか、自分でも作れる！」と思って一気にテンションが上がったものです。

そんなわけで、プログラミングを始めるには、

・プログラミング言語を決める
・自分に合った入門書を探す
・ひたすら手を動かす

ということが遠いようで近道だと思います。

今はGoogleなどで簡単に調べられるので、何かわからないことがあってもだいたいのことは見つけられると思います。ですがそうやって調べた情報も、情報の海に溺れて目的を見失っ

てしまったり、読んだだけでわかった気になってしまったりで、身につかないことが少なくないです。むしろネットの情報を遮断して入門書のみで進んだ方が、より集中できて理解が進むと思います。

身銭を切ることでやらなきゃ感もアップするので、筆者は入門書を1冊、自分で買うことをお勧めします:-)

サンプルまたは実践で学ぶ

当時筆者が購入した入門書は『入門Visual C++6.0』[注1]でしたが、1冊終えてなんとなく雰囲気は掴めるものの、まだまだ自分から何をしていいのかはわかりません。ほかの入門書も読んだりしてみましたが、入門書ばかりでは入門の域を超えることはできませんでした。

筆者の場合は環境がよかったのか、入門を終えて割とすぐに実践でプログラミングをする機会に恵まれました(すぐに実践でプログラミングできるほうが珍しい気がしますので、実践とはいかない場合は、Webの連載記事や自分が作りたいものなどサンプルレベルでかまいません)。とはいっても、入門書なしで1人歩きできるほどではなかったので、入門書をリファレンス代わりにして、足りないものは書籍なり雑誌なりWebなりで情報を探しながらよちよち歩きです。

しかし入門書のプログラムではなく、自分で作りたいものを作るというのは全然異なった印象を受けました。入門書のサンプルプログラムを作ったときでもそれなりに満足感はあったのですが、自分が作りたいプログラムというのは思い入れも違うし、モチベーションが一気に上がるものです。

「実践よりも基礎が大切では？」と思われるかもしれませんが、何が必要な基礎なのかもわからない状態で「鶏が先か、卵が先か」で悩むくらいなら、入門書を終えた今はひたすら手を動かし、実践で学んだ方が、何が必要な知識かがわかってきます。基礎は、何が必要な知識かわかってきてから学ぶでもまったく遅くありません:-)

ベストプラクティスを学ぶ

Visual C++ 6.0でプログラミングをしていたころ、『CODE PROJECT』[注2]というサイトを見つけたときは歓喜したものでした。これから自分が作ろうとしていて、しかし自分が作れるかどうかわからないというものを、どこかの誰かがすでに作っていて公開してくれている！

このころから入門用のコードではなく実践向けに作られたコードに直面して、人のコードを読むこと、有益な情報を得るためには英語がそれなりに必要なことを理解し始めました。人のコードを読むことで、なぜそう書くのか、どの書き方がベターなのかを気にするようになり、これがプログラミング上達につながったのかなと今では思います。

また、同時期に出会った本が『Effective C++改訂第2版』[注3]です。この本を知ったきっかけはメーリングリスト[注4]だったような気がするのですが、ちょっと記憶が定かではありません。とりあえず作り方はわかる。だけどこの作り方で正しいのかは自信がないという筆者にとって最良の良書でした。

本書を簡単に説明すると、「C++でこういうことをしたい場合は、こう書くべき」ということが理由を含めて数十パターン網羅されていて、C++におけるベストプラクティスが体系的にまとまっている本です。

ある程度自分でプログラミングできるようになってきたら視野を広げてみましょう。きっかけは人それぞれだと思いますが、良書と呼ばれ

注1) 山田昌良著、広文社、ISBN978-4877780067

注2) CODE PROJECT http://www.codeproject.com/

注3) Scott Meyers著、吉川邦夫訳、アスキー、ISBN978-4756118080

注4) Cppll ML http://www.freeml.com/cppll

ている本がたくさんあります。良書に出会い、ぜひ先人の知恵から学べる部分は学んでいってください。入門時にはわからなかったパズルのピースがはまり出しますよ！

反面教師、尊敬できるプログラマから学ぶ

以前、プログラミングができると紹介された30代の方と一緒に仕事をする機会がありました。はじめは作ったアプリケーションを見せられてスゲーと思ったのですが、いざソースコードを見るとコピー&ペーストしたような似たコードばかりだったり、グローバル変数だらけだったり、メモリリークだらけでした。

なぜここはこう書いたのかと聞いてもわからないと言われる始末……。何かおかしい……。

自分のイメージしていた“できるプログラマ”とは異なり、プログラミング言語に使われているようなプログラマでした。独学で学んできた筆者にとって良い機会になるはずだったのですが、反面教師としてこういうプログラマにはならないように気を付けようと思ったものでした。

- 自分のコードを他人に説明できない
- 自分のコードをキレイに書こうとしない
- 自分の手法にこだわり、新しい手法を学ぼうとしない

これはお勧めできる習得方法ではないので、気になる方は『アンチパターン ― ソフトウェア危篤患者の救出』[注5]を参考にしてくださいね。

次に、他社のプログラマと協業するプロジェクトに参加する機会もありました。当時真っ黒の画面(ターミナル)にviという得体のしれないエディタで作業することはまるで魔法を使っているように見え、「あぁこれがプログラマだ！」と思ったものです(笑)。

そこで、「プログラマとは最低限のインプットで最大限のアウトプットを作り出すクールな仕事だ！」と思えた瞬間でした。そういった人たちと一緒に仕事ができるということは筆者を大きく成長させてくれました(その後、私は転職するという道を選ぶことになります)。

もちろん、我流でもプログラミングはできますが、尊敬できるプログラマと一緒に仕事ができる環境というのはすばらしいものです。すごいプログラマというのはたくさん教えてくれます。なぜなら楽をしたいから。楽というのは自分だけではなくあなたも含めて全体という意味です。

尊敬できるプログラマを見つけて、教えてもらえるよう頑張りましょう。

複数のアーキテクチャ、プログラミング言語を学ぶ

数年の間、Visual C++ 6.0でWindowsアプリ開発というプログラミングしかしていなかったのですが、うってかわってWebシステムを開発するという機会に恵まれました。筆者にとってOSといえばWindows、プログラミング言語といえばC++でしたが、それが同時に変わってしまったのです。

- OS……Windows ⇒ Linux
- 言語……C++ ⇒ Java
- アーキテクチャ……Windowsネイティブ ⇒ Webシステム
- プロジェクト規模…1人 ⇒ 複数人

戸惑いも多かったのですが、新しい技術をたくさん学べることのほうが楽しく必死になって勉強したことを覚えています。筆者の場合はC++を学んでからJavaに進んだのですが、新しくJavaを学んだときは文法と、どうやってビルドしてどうやって動いているのかということに主眼を置きました。

JavaといえばEclipseというすばらしい統合開発環境がありますが、始めからこれを使って

注5) W.J. ブラウン、Hays W. McCormick、Raphael C. Malveau、Thomas J. Mowbray共著、岩谷宏訳、ソフトバンククリエイティブ、ISBN978-4797321388

しまうと、裏側がわからずどうやってアプリケーションができあがるのかがまったくわかりません。裏で何をやっているかを把握しておかないと、問題が発生した場合に調査する術を失ってしまいます。

Javaがどうやってビルドしているかを学ぶことができたおかげで、C++がどうやってアプリケーションをビルドしているのかを理解できるようになったりと、複数言語を学ぶことで良い相互作用が働きました。

技術には得手不得手があります。すべて同じ道具で作業するよりも、この作業にはこのプログラミング言語、このシステムにはこのOSというように選択できるようになれば、何倍もの効率を得ることができるようになります。最適なツールを選び、最小の作業で最高の効率を上げる、これこそプログラマの腕の見せ所でしょう。

1つの言語をある程度使えるようになってきたら、ほかの言語や技術、ツールにぜひチャレンジしてみてください。

自分のアプリ、サービスを開発する

仕事ではプログラミングさせてもらえない、または指示されたとおりの作業しかさせてもらえないということもあると思います。プログラミングは良い意味で1人でもできるものです。ある程度プログラミングできるレベルになって、作りたいものがあれば思い切って作ってしまいましょう。

自分で作るにあたって、筆者からアドバイスするとしたら次の2点でしょうか。

①自分がユーザで、自ら使いたいもの
②機能は盛り込み過ぎず、リリース優先

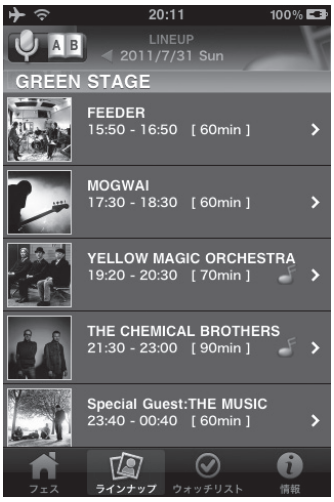
仕事ではないため自ら時間を見つけて開発しなければならず、こだわり過ぎた挙句リリース前に挫折してしまうということも少なくないからです。

また、誰が使うのかわからないものを作るというのは、なかなかモチベーションを維持することが難しいものです。

時間はかなりかかってしまうと思いますが、プログラミング以外の、サーバ、アーキテクチャ、インフラ、セキュリティなど意識が低くなってしまいがちなさまざまな必要技術／作業を意識することで、技術の幅を一気に拡げることができます。

筆者の場合は、Titanium Mobileを利用して

▼図1　fesmo



「fesmo」[注6]というiPhoneアプリを開発・リリースしました(**図1**)。

ちょっと脱線しますが、フジロックフェスティバルという音楽イベントがあり毎年のように参加しているのですが、「もっとフェスを楽しむためのアプリを！」というのが開発のキッカケでした。

新しい技術にチャレンジすることで、Mac OS、Titanium Mobile、JavaScript、iOS、AWS(Amazon Web Services)など新しいテクノロジに触れ視野が広がり、また、WindowsからMac OSに乗り換え、エディタをVimに乗り換えるなど毎日使う道具まで変える機会ともなりました。

アウトプットを出す

最後になってしまいますが、筆者が一番お勧めするのは「アウトプットを作る」です。

コードを書くこともアウトプットなのですが、学生の授業／テスト勉強でも、読んだだけでは理解できずノートに書かないと覚えることができませんでした。筆者の感覚的には、下記のような順序で理解が深まると思っています。

人に説明する ⇒ 書いて理解する ⇒ 読んで理解する

人に説明するというのは、相手(ぬいぐるみでも良いのですが)が必要なことなので場や機会がなく難しい面があります。それに比べ書いて理解するというのは簡単に実践できる方法だと思います。

具体的に何をすれば……という読者は、ブログを開設して今日理解できたこと、理解できなかったこと、調べたメモなどを記事にしていくのが1つの手です。Webニュースやブログ記事を流し読みしただけでわかった気にならず、未来の自分のために、何がわかって、何がわからないのかをドンドン明らかにしておきましょう。ちなみに筆者のブログの過去記事は恥ずかし過ぎて消してしまいました。ですが今の自分があると思えば、そんな恥ずかしさは一時のものです(笑)。

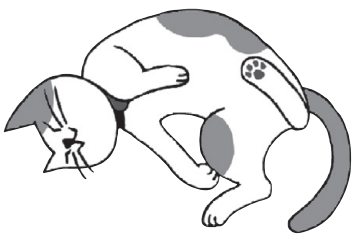
また、人に説明することで理解を深めたい場合には勉強会に参加してスピーカーになるという手もあります。探してみるとわかりますが、たくさんの技術勉強会が連日のように開催されています。スピーカーを募集していることが多いので思い切って申し込んでみるというのも1つの手です。

Webニュースやブログ記事を流し読みしただけでわかった気にならず、アウトプットすることでより理解を深めていってください。筆者の場合は社内に技術勉強会の場を作り、説明する機会を設けることでアウトプットしています。

まとめ

いかがだったでしょうか。過去をふりかえりながら、勉強になった点、こうやって勉強してきたという筆者の経験からくるノウハウを駆け足で紹介させてもらいました。そして筆者自身にとっても、あらためて日々の勉強が重要だということを実感することができました。何かを始めるのに遅過ぎることはありません。

この記事が、みなさんのプログラミングを始めるきっかけになってくれたら、これにまさる喜びはありません。みなさんの作ったプログラムに触れられる日を楽しみにしています！SD



注6) https://itunes.apple.com/jp/app/id452598711

Part 6

これからエンジニアになるあなたへ

コニカミノルタ ビジネステクノロジーズ株式会社
中原 慶　NAKAHARA Kei

プロローグ

やっと寒さが和らいできたかと思えば、次は花粉が襲ってきました。この時期、多くの方が新たな生活の準備をされていることでしょう。筆者は、十数年前にソフトウェアを開発する会社に就職しました。初めて出社するときのドキドキ感は、今でも忘れません。とくに「みんなの足をひっぱらないだろうか」「みんなとうまくやっていけるだろうか」といった心配が尽きませんでした。一方で「将来は独立して社長になってやるぞ」「業界を代表するスーパーエンジニアになってやるぞ」といった、夢と希望と野望に胸を躍らせる気持ちもありました。その後、数回の転職／職場移動を経験しましたが、その都度、期待と不安でいっぱいになります。そのたびに初心を思いだし、自身の目的を再確認しています。

さて、みなさんはなぜソフトウェア業界を選択されたのでしょうか？ 「プログラミングが好き」「ゲームやインターネットビジネスに関心がある」「お給料がよさそう」など、さまざまでしょう。

筆者の動機は、ほかの職種に比べて最も早く社長になれると思ったからです。当時の筆者のイメージは「社長＝お金持ち」でした。ちなみに筆者は文系学部の出身で、プログラミングどころかコンピュータの起動や停止もままならないまったくの素人でした。まったくの素人だったからこそ、エリートエンジニアに比べて苦労話は多いかもしれません。何せ、新人のころは先輩たちが話している内容が宇宙人の言葉のように思えましたから。

そういった背景もあり、新しい技術を効率よく学習するために、自分なりにいくつかの心構えや方法が身につきました。本記事では、これからエンジニアになる方をはじめ、エンジニアになって3年目ぐらいの方までを対象に、筆者の経験をもとにした技術力を向上させるための心構えや方法を紹介します。なお本記事では、できるだけ多くの読者に使っていただける汎用的な心構えや方法をお伝えします。そのため、詳細なプログラムの記述方法やフレームワークの使い方、方法論の説明などは解説しません。具体的には次に示す3点をお伝えします。

・文献の読み方
・プログラミング能力の向上方
・モチベーションの保ち方

本記事がみなさんの向上心を掻き立て、ソフトウェアを探求する際のお役にたてれば幸いです。

文献の読み方

まずは、筆者がエンジニア1年目のころから心がけている「文献の読み方」について、お伝えします。

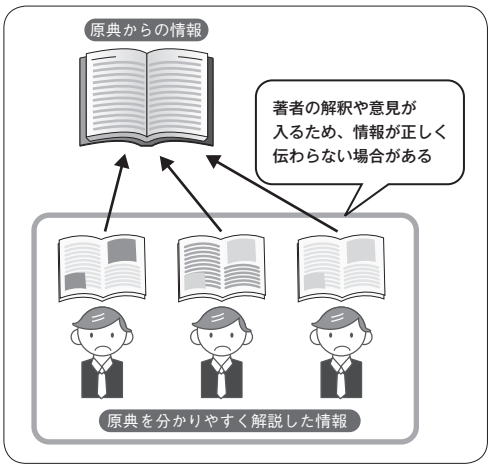
先述のとおり、筆者はまったくの素人でした。新人のころは、同僚についていくために入門者向けの書籍やWeb記事を読みあさりました。当時の筆者にはすべてが新鮮で、サンプルを動かすたびにワクワクしていました。そうこうしている中、同じ技術（とくにプログラミング言語）に関する文章の参考文献が、ほとんど同じであることに気づきました。そして、その参考文献を読んでみところ、筆者が読みあさった多くの入門者向けの記事の原典ともいえる内容が書かれていました。これまでに読んできた「○×入門」の文献は、原典の情報をわかりやすく解説したものだったのです。

原典の重要性

これらの入門書は、難解な技術をわかりやすく解説しているので、筆者のような初学者にはとてもとっつきやすい文献です。しかし、著者が原典からの情報をわかりやすく説明しているので、どうしても著者の解釈や意見が入ってしまいます。意地悪な言い方をすると、著者の解釈が間違えている可能性もあるのです。原典以外の情報に価値がないと主張しているわけではありません。ただ、原典となる仕様や論文だけが、唯一純粋で正しい情報だということを認識する必要があるのです（**図1**）。

新しい技術を学んでいる際に、理解が追い付かなくなったらまずは原典を読みましょう。筆者もそうだったように、最初はほとんど理解できないでしょう。それでも、水中深くに潜っていくように、一生懸命、あきらめずに理解を試みましょう。その後、わかりやすく解説された二次的、三次的な文献を読むようにしましょう。そうすることで、原典の情報の理解が促されるでしょう。できれば、複数の文献を読むことをお勧めします。いろいろな観点で問題をとらえることで、理解が促進されるでしょう。

▼図1　原典とそれ以外の情報



二次的、三次的な文献の扱い

さらに筆者が心がけていることは、二次的、三次的な文献を読む際は斜に構えて読むことです。そして、疑わしい個所があれば、本当に正しいのかをじっくり調べ、考えます。そのために、原典はもちろん、ほかの文献も調べます。その結果、理解が深まり、自身の意見も確立されるのです。それが学んだ結果であり、エンジニアとしての基礎になると筆者は考えます。

具体的な例を挙げると、たとえばJavaを新しく学ぶ際は、Javaの仕様（APIリファレンスだけではなく、Virtual Machineの仕様やBlue Printなども含めて）が原典であることを認識したうえで学習を進めます。そして、それら原典の理解を促すために、二次的、三次的な情報を補助的に用います。ただし、二次的、三次的な文献をすべて鵜呑みにするのではなく、間違えているかもしれないという心持ちで読みます。そうすることで「Javaにおける Threadのしくみ」や「Javaにおけるメモリの使い方」などについて深く理解し、自分なりの意見を持てるようになります。さらに、Javaを学んで得た知識が、

「C#」や「Ruby」など異なるプログラミング言語を学ぶ際に、1つの指針になります。

プログラミング言語以外にも、各種フレームワークやツール、ライブラリなど、エンジニアは多くのことを学ばなければなりません。その際も、原典を意識し、あらゆる情報を疑って読む方法は有効です。

プログラミング能力の向上法

筆者は、オブジェクト指向に関する教育コースの作成や、講師を務めていたことがあります。ちょうど今の季節は、新人教育の担当をしていました。受講される新人さんの多くは、不安そうな表情をされていました。そういった新人さんから「どうすればプログラミングが上手になりますか」とよく質問されました。そんなとき、筆者は決まって同じ答えを返しました。本節では、筆者がエンジニア2年目のころから始めた、プログラミング能力を向上させる方法についてお伝えします。

優秀なソースコードを多く読もう

筆者は、優秀なプログラマをたくさん知っています。そのため「オレのプログラムはエクセレントだぜ。世界最高にすばらしいぜ」なんて、口が裂けても言えません。しかし、まったくの素人だったころに比べると、少しは上達したと思います。それは、優れたプログラム(ソースコード)を多く読んだからです。ここで大切なのは、「優れた」プログラムを読むということです。ダメダメなプログラムを1億行読んでも、大きな進歩はありません。逆に、自身の中での妥協点が下がってしまい、マイナスになってしまうかもしれません。しかし、優れたプログラムを1万行読めば、必ず成長します。

その理由の1つとしては、プログラムの記述方法そのものが参考になるからです。たとえば、変数名やメソッド名、クラス名、インデントの付け方やアルゴリズムなどが参考になります。優れたプログラムはとても読みやすく、内容の理解が容易です。逆に、ダメダメなプログラムは分岐やループのネストが激しく、非常に理解しづらいです。ひどい場合は、理解を促すために1行1行にコメントが書かれています。このようなプログラムは、プログラムとコメントを同時にメンテナンスしなければなりません。そのため開発効率が悪く、もしいずれかの修正を忘れてしまうと、次に読む人を混乱させてしまいます。そもそも、コメントがないと意図を伝えられないほど複雑なプログラムは、メンテナンス効率が悪く、バグの温床となりやすいのです。

人に理解してもらうプログラム

プログラミングの原則は「シンプルに」です。プログラムは、コンピュータに命令を与えるためだけではなく、次に読む人(メンテナンスする人)にも理解してもらわなければなりません。次に読む人が理解に苦しむと、メンテナンス効率や品質の低下に直結してしまいます。優秀なプログラマが作成したプログラムは、非常にシンプルで読みやすいものです。優れたプログラムを多く読むことで、自身も読みやすいプログラムの作成を心がけるようになり、プログラミング能力が向上します。書道や絵画、音楽なども、初めは優秀な作品をお手本にするところから始まります。プログラミングにも同じことが言えるのです。

さらに、優れたプログラムを読むことで、優れた設計も学ぶことができます。先にも述べましたが、優れたプログラムはとてもシンプルです。それは、命名の明確さやアルゴリズムの完結さだけではなく、メソッドやクラスの責務がシンプルだからです。1つのメソッドが何百行もある場合、恐らくそのメソッドは複数の責務を負っています。

そのようなプログラムは、テストプログラムの作成や内容の理解が非常に困難で、バグが多く潜む可能性があります。また、たった少しの

変更が多くの個所に影響を与えるため、変更に弱いプログラムになってしまいます。

優れたプログラムを多く読むことで、クラスやメソッドの責務の分け方、さらには、設計のパターンやインターフェースの使い方などを実例で学ぶことができるのです。これらの理由から、優れたプログラムを読むことは、プログラミング能力の向上に欠かせないと言えます。ところで、どこに優秀なプログラムがあるのかは、みなさん自身で見つけてください。オープンソース、書籍のサンプル、または、みなさんの職場にあるかもしれません。

人が理解しやすいプログラムを書く

もう1つ、筆者がプログラミング能力を向上させるうえで実施した方法があります。それは、プログラムをたくさん作成することです。当たり前のことですが、実践されている方は少ないと思います。

たとえば、新しい技術を習得するために書籍を購入したとします。書籍の内容に納得しながら読み進めますが、自身でゼロからサンプルを作る方は少ないでしょう。実際に手を動かして経験しなければ、プログラミングの技術は上達しません。もちろん、ロジカルで構造的な考え方が得意な方は、プログラミングの上達が早いかもしれません。しかし、経験なくして優れたプログラムを書くことはできません。実際に手を動かして、失敗と成功を何度も繰り返しながら進めることで、初めて自分の力になるのです。これは、プログラミングに限ったことではありません。何かを学ぶ際は、実際に自分で手を動かして経験しないと身につきません。

ここで1つ重要な注意点があります。たくさんプログラムを作成するといっても、やみくもに作成するだけではいけません。少なくても、人に見せることを前提とした、読みやすくシンプルなプログラムを作成しなければなりません。逆に言うと、「動けばいい」を最優先に考えた場当たり的なプログラムをいくら作成しても、プログラミング能力は向上しないでしょう。シンプルで、読み手にやさしいプログラムを作成しなければなりません。

本節では、プログラミング能力を向上するための2つの心構えをお伝えしました。「優れた」プログラムを多く読み、「他者が理解できる」プログラムを作成することです。そうすることで、必ずプログラミング能力は向上します。

モチベーションをキープする方法

素人だった筆者は、カラカラに乾いたスポンジのような状態でした。そのため、いろんなことを吸収できました。そして、ソフトウェア開

Column

アジャイル型開発での設計

世間では、いまだに「アジャイル型開発では設計をしない」と思っている方が多くいらっしゃいます。筆者は前職でアジャイル型の開発を行っていました。少なくとも筆者が経験したアジャイル型開発では、必ず設計をしていました。設計ドキュメントを残すかどうかは別として。

ただ、ウォータフォール型の開発とは異なり「設計フェーズ」を設けることはありません。設計は、プランニングやペアプロ[注1]、リファクタリング、テストコードを作成している際に行われます。直感的には、プログラミングをしている時間よりも設計をしている時間のほうが長いと感じていました。そして、常に最適な設計であるために、改善を行います。その結果、設計もプログラムもシンプルで理解しやすい状態を保つのです。

注1） 2人で組になり、コーディングと観察の役割を持たせる開発手法。

発のおもしろさにどんどんのめりこんでいきました。通勤電車で読んでいた週刊誌は、いつしかソフトウェアの専門誌に変わっていました。個人でドメインを取得し、レンタルサーバを借りて実験サイトを運用していました。今思い返すと、よくモチベーションをキープできたと思います。

前節までは、学習の方法についてお伝えしてきました。しかし、やらされている学習は長続きしません。また、興味のある技術を探求する場合でも、時間の経過とともに徐々に情熱が冷めていくことが、しばしばあります。そこで本節では、筆者の体験をもとにモチベーションのキープに有効だと思われる方法を3つお伝えします。

・学んだことを伝える
・定期的にコミュニティに参加する
・仲間を作る

学んだことを伝える

1つめは「学んだことを伝える」ことです。人に伝えるためには、自身の意見を理路整然と説明できなければなりません。また、質問に対しても的確に答えなければなりません。そのために、自分なりの説明ロジックを組み立てなければなりません。そうすることで、自分自身の理解が深まります。ただし、勉強したばかりの情報を伝えますので、間違えていることも多々あります。そのため、聞く側もすべてを鵜呑みにせず、本当に正しいかどうかを見極めながら聞く必要があります。

筆者の場合は、新しく得た情報を積極的に業務に活用したり、仲間に伝えたりしました。このように、新しく得た情報を実際に使うことで、学んだ価値を実感できます。これも、モチベーションをキープするうえでとても良い方法だったと思います。

ただ、今思い返すと誤って伝えたこともありました。その際、聞いている方が指摘してくれることが多々ありました。指摘を受けることで、誤りを認識し正しい理解につながりました。自分の誤りに気づき理解を正しくするという意味でも、学んだことを伝えることには価値があるといえます。

定期的にコミュニティに参加する

2つめは、「コミュニティに参加する」ことです。筆者はエンジニアになって2年目あたりからいくつかのコミュニティに参加しました。具体的には、メーリングリストや読書会、要素技術のコミュニティ、ツール開発のコミュニティなどです。その中で、社外の仲間をたくさん得ることができました。社外の仲間は、担当している業務範囲や開発ドメイン、企業文化も異なります。そのため、得られる情報がとても新鮮で勉強になりました。また、外部のコミュニティに参加されている方は、業務外でもソフトウェアに関わっているわけですから、基本的にモチベーションが高いです。そういったモチベーションの高い方と定期的に接することも、自身のモチベーションをキープすることに役立ちました。

とはいうものの、コミュニティに参加し始めたころは自分に自信がなく、ほとんど発言できませんでした。しかし、何度か出席しているうちに他の参加者とも面識ができ、気軽にコミュニケーションがとれるようになりました。そして、自分の意見を発信するようになり、議論するようになりました。議論することで、新しい知識や考え方を得たり、自分の意見をみんなに認めてもらったりします。そうなってくると、コミュニティの定例会に参加することが楽しみになり、技術を探求するモチベーションがさらに上がりました。

仲間を作る

3つめは、「一緒に学ぶ仲間を作る」ことです。筆者がエンジニア2年目のとき、レンタルサーバを借り独自ドメインを取得しました。そして、

そこを実験用のスペースとし、新しく得た技術を仲間と一緒に試しました。雑誌で見た新しいアプリケーションサーバをインストールし、新しい言語を試したりしました。これらはもちろん業務外です。しかし、遊び感覚で新しい技術を試すことができましたので、モチベーションはもちろん、技術的な力も向上しました。もし1人だったら、途中で飽きてやめていたかもしれません。仲間がいたから楽しくできました。

また、筆者は仲間と一緒に一昨年の12月にリアルタイムシステムの形式的な表現に関する調査研究の成果を発表しました。この発表に至るまでに、約2年間の時間を要しました。調査対象のテーマは非常に難しかったのですが、大先輩から多大なアドバイスをいただき、仲間とともに励ましあい、試行錯誤を繰り返して資料をまとめました。その結果、成果を発表でき、とても心地良い達成感を得ることができました。もちろん、1人でも研究成果を発表することはできるでしょう。しかし、業務時間外に長期間に渡って1つのテーマを探求するのは、よほどのモチベーションがないと続きません。そんなとき、同じゴールに向かう仲間がいるととても励みになります。

◆ ◆ ◆

本節で紹介した「モチベーションをキープする方法」は、どれも1人で実施するには難しいものです。良い仲間を作り、仲間と切磋琢磨することが、ソフトウェアに対するモチベーションをキープすることにつながるのだと思います。そして、何よりも楽しくないと長続きしません。みなさんもぜひ多くの仲間を作り、ソフトウェアに対する活動を楽しく進め、モチベーションをキープしてください。

エピローグ

以上、本記事ではまったくの素人だった筆者が経験の中で身に着けた心構えや方法をお伝えしました。最後に、筆者が思うエンジニアにとってもっとも大切なことと、もっとも不要なことについてお伝えし、本記事を締めくくりたいと思います。

まず、筆者が思うもっとも大切なことは「楽しむ」ことです。先にも述べましたが、しかたなくやっていることは楽しくありませんし、長続きしません。一方、知的好奇心をくすぐられる研究活動は、時間を忘れてはまってしまいます。その結果、エンジニアとしての能力も向上していくのだと思います。

一方、筆者が思うエンジニアにとってもっとも不要なものは「プライド」です。もちろん、プロとしてのプライドは必要です。筆者が不要とするのは、知らないことを恥だと思い人に聞くことを阻害してしまう気持ちのことです。いくら調べても、1人ではどうしても理解できないことが多々あります。また誤って理解していることもあります。その際、もっとも効率的な方法は詳しい方に聞くことです。そこでプライドが邪魔すると、せっかくの知るチャンスを逃してしまいます。妙なプライドは捨ててしまい、馬鹿にされることを恐れず、ぜひ質問してください。逆に、もし質問された場合は「人に伝える機会を得た」と考え、丁寧に説明してあげてください。先に述べたとおり、人に説明するということは、自分自身の理解を深めることにつながりますので。

本記事を読まれた皆様は、ご自身のチームを誰でも気兼ねなく質問しあえる雰囲気にすることを心がけてください。メンバ間には、上限関係やパワーバランスがあり、わかりきった質問をすると叱られるかもしれません。それでも、勇気をもって質問してください。下調べは大切ですが、はじめはどうやって調べて良いのかさえも分からないのです。質問できない環境は、業務の遂行に支障をきたすだけでなく、自分のエンジニアとしての成長を阻害しかねません。みなさんがベテランエンジニアになっても、初心者の気持ちを忘れないでください。SD

Part 7

プログラミング言語への理解を深めよう――お勧めの学習ステップ

福原 毅　FUKUHARA Takeshi

はじめに

Webサイトから検索したサンプルプログラムをコピー&ペーストして、なんとなくWebサイトを構築するというのが、今のプログラミングを学ぶスタイルでしょう。誰でも簡単に動くものを作ることができるという点で、とても恵まれた環境です。将来Webデザイナになるのであれば、これで十分です。しかし、もしプログラム開発を生業とするならば、これではお話になりません。少なくとも筆者は、このような方と一緒にお仕事をしたくありません。

コンピュータの基本動作は、情報の加工と移動の繰り返し

極論すれば、コンピュータでの処理に求められるのは、情報に付加価値を付けて別の場所に移動することだけです。ですので、プログラミングを行ううえで覚えなければならないことは、極めて少ないのです。大雑把に言えば、演算、条件分岐などの制御、外部とのデータの入出力方法の3つだけで開発ができます。あとは、どうすれば効率を上げられるかという点が残るだけです。効率とは、プログラムの処理速度であったり、単位時間あたりのコードの記述量であったり、プログラムの変更や再利用の容易さなどのことです。

プログラミング言語や、それに付随する処理系は、効率を追求するために改良されてきました(近年は、セキュリティという効率以外の側面もありますが、今回は棚上げしておきます)。プログラミング言語を習得するということは、その言語が、何をどう効率化するために作られたのかを理解し、その特性を活かした効率的な開発ができるようになることです。

その言語が何をどう効率化しようとしたかを知るには、その言語が登場した歴史的背景を知り、プログラミング言語開発者たちがたどった苦悩の道筋を追体験することが一番の近道です。残念ながら、この苦悩の道筋を追体験するためのステップを効率化する方法を、筆者は知りません。

まずは机上でプログラミング言語が開発された歴史的背景を知る

言語の仕様には、その言語開発者からの「こう使って効率化してほしい」という思想が込められています。まずは、その思想を理解し、対象となる言語の全体像をとらえることが必要です。この思想を大まかに把握してから、細かい仕様の理解をしたほうが良いでしょう。

言語開発者の思想を知るには、先ほども述べたようにその言語が登場した歴史的背景を知るのが近道です。この目的には、Web上の分散

した情報ではなく、書籍を購入して読むべきです。紙、電子媒体かは問いません。サンプルコードの多さや、すぐに使えることを売りにしているような書籍は、この用途には適しません。それよりも、何を効率化するためにその言語仕様が作られたのかを丁寧に解説している書籍が良いでしょう。翻訳本はお勧めしません。翻訳の質が問題なのではなく、欧米人が物事を理解するプロセスと、日本人が物事を理解するプロセスが根本的に違うからです。日本人が、日本人の思考に沿って例示しながら解説している書籍をお勧めします。

言語開発者の苦悩を追体験する

人は抽象的な思考と、具体的な経験を繰り返すことで、物事を理解します。机上で歴史的背景を知る段階は、抽象的な理解の段階です。抽象的な理解だけでは実際のプログラミングはできません。逆に、具体的な経験だけでも、効率的なプログラムを書くことはできません。

具体的な経験を積むためには、自らプログラムを記述し、デバッグしながら、その動作を確認することが大切です。思いどおりに動かないプログラムを、動くように修正する過程を体験してください。

まず、学習開始当初のあいまいな知識を総動員して予測を立てましょう。そしてその予測通りにプログラムが動作しているかを1ステップずつ確認し、問題があれば修正するようにします。そして再度テストを行うというサイクルを、繰り返し繰り返しを行うのです。アセンブラや、C言語といった古くから存在するプログラミング言語で開発し、それが思うように動かないという苦い経験が、次に学ぶ最新の言語のすばらしさを理解する近道となります。

過ちに気づくことが最も難しい

人は、過ちに気づくことができれば、必ずそれを改善できます。同じ過ちを繰り返すのは、その過ちを認知できないからです。プログラミングにおける過ちの結果は、バグとなって現れます。バグは、見つかれば必ず修正できます。そして、同じようなバグを作り出さないように改善する方法を模索し、それによってプログラマとして大きく成長します。しかし、机上の学習では、バグは現れません。たとえ自分で実際にプログラミングを行ったとしても、1人でプログラミングを行っていると、バグに気づく機会が極端に少なくなってしまいます。

仕事の現場でプログラム開発を行うと、自分以外の多くの眼がバグを見つけてくれます。バグを見つける機会、すなわち、過ちを認知できる機会が多くなれば、成長する機会もそれだけ多くなります。顧客や先輩、同僚から多くのバグを指摘されると、自分を否定されたかのような錯覚に陥り、最初はつらく感じるでしょう。しかし、自分の仕事(プログラム)を多くの眼に晒すことのできる実務経験こそ、成長を加速させてくれることを忘れないでください。

とりあえず動く言語だけでプログラミングするな!

今日(こんにち)のほとんどの実務は、Java、C#、JavaScript、PHP、VBA(Visual Basic for Application)といった高級言語を使ったお仕事でしょう。これらの処理系は、コードを記述すればとりあえず動きます。しかし、初心者が記述したC/C++やアセンブラのプログラムは、ほとんどまともに動きません。知識があいまいなまま記述してもとりあえず動く処理系と、知識があいまいだと動かない処理系の2つに分類できるのです。

とりあえず動く処理系では、とりあえず動いてしまうため、過ちに気づくことができません。誰が使ってもとりあえず動く優れた処理系を使うと、確かに生産性は向上するけれど、それを使う技術者のスキルは逆に低下するのです。直近の業務では、アセンブラやC/C++を使うことはないかもしれません。しかし長く使えるス

キルを身につけたければ、独学でも良いのでC/C++やアセンブラに関するスキルを習得することをお勧めします。

こんな感じの学習計画はいかが?

ここまで、抽象的なお話をしてきましたが、もう少し具体的に、どのようなステップでプログラミングを学んでいけば良いか、20年ほど前の筆者自身の経験を基に記述するので参考にしてください。

1st Step:とりあえず動く言語と、全然動かない言語の両方を、並行して始める

まずは、Webブラウザを利用してJavaScript、もしくはVBAなどのマクロを使って"習うより慣れろ"の気持ちでプログラミングを行ってみましょう。とりあえず、思い通り動く経験は、学習意欲につながります。ただ、プログラミングのスキルという意味では、この経験はほとんど意味がありません。

ここで並行してアセンブラの学習を始めることをお勧めします。アーキテクチャは何でもかまいませんが、今なら、携帯端末に採用されているARMでしょうか。書籍が充実しており、学習効果を測ることができるという意味で、情報処理技術者試験のアセンブラ言語であるCASL IIも良いでしょう。PCで一般的なx86は敷居が高過ぎるので避けたほうが良いと思います。

ここではアセンブラで本格的なプログラミングができることを目的とはしません。四則演算程度ができれば十分です。コンピュータの基本動作を理解しつつ、Javaなどの高級言語がない場合のたいへんさと、めんどうくささを経験すればOKです。

2nd Step:C言語を味わう

アセンブラがたいへんなのだということを経験したうえで、C言語を学ぶことをお勧めします。もしC言語しか知らなければ、ポインタやメモリの扱いなどを難しく感じるかもしれません。しかし、1st Stepでアセンブラのたいへんさを経験していれば、C言語のすばらしさがわかります。C言語は歴史の長い言語ですので、日本人が執筆した定番と言われる教科書がいくつかあります。それらの教科書でしっかり学習してください。

先ほど過ちに気づくことができれば、人は改善し、成長できるとお話ししました。しかし、C言語で特徴的なメモリ関係のエラーだけは、机上の学習で気づくことができません。メモリ関係のエラーに実際に遭遇しない限り、C言語の何がわかっていないのかが、いつまでたっても認知できません。これが、C言語の習得を難しくする要因の1つです。

メモリ関係のエラーに遭遇する機会を増やすためには、メモリエラー自動検出ツールがお勧めです(コラム参照)。少しプログラミングができるつもりになった時点で作成したプログラムを、メモリエラー検出ツールでチェックすると、驚くほどさまざまなエラーや警告が出力され、その内容をほとんど理解できない自分に気づくはずです。なぜ、そのメモリエラーが検出されたのかを、再度、購入した書籍で確認し、デバッガで動作を追いかけることで、C言語についての本当の理解が得られます。

残念ながらこの手の製品は、学習のためだけに購入できるほど安くありません。少しプログラミングができるようになったと思う段階で、無償評価版をダウンロードして試してみるのも1つの方法です。また、UIや機能は劣りますが、同様のオープンソース製品もあり、"memory debugger"で検索すれば、いくつかのツールを見つけることができます。

Windowsであれば、"Visual Leak Detector for Visual C++"というオープンソース製品がよさそうです。言語の勉強以外にツールの勉強も必要となり、若干負担は増えますが、初期の学習の段階からこのようなツールに慣れておけば、高い学習効果と、将来の開発生産性の向上につながります。ぜひ、言語習得と並行して、生産

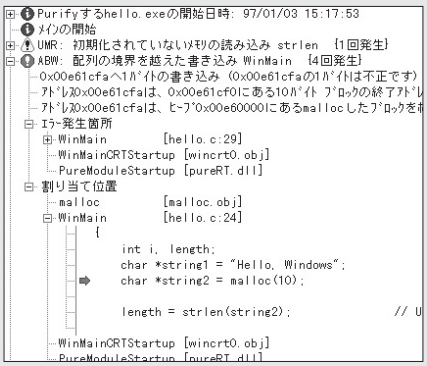
Column

メモリエラー自動検出ツール“Purify”

筆者は18年ほど前に、当時のPure Software社のメモリエラー自動検出ツール“Purify”（現在のIBM Rational PurifyPlus）という製品に出会い、これをC言語開発者の方々へ訴求、販売していました。

Purifyは、実行するだけでメモリに関するエラーを自動的に検出し、レポートしてくれます（**図1**）。筆者はこのツールで、自分が過去に作成したプログラムのエラーを確認して初めて、C言語について、自分が理解できていなかった領域に気づくことができました。本来この手のツールは、ソフトウェア製品の品質を向上させ、サポートコストを低減させる目的で使用します。しかし、筆者にとっては、C言語のプログラミングスキル向上に大きく貢献したツールでした。

▼図1　メモリエラー自動検出ツール“Purify”



※現行バージョンの製品画面ではなく、16年前の製品の画面です。

性を向上させてくれるツール、とくに手に馴染むデバッガを見つけておくことをお勧めします。

余談ですが、メモリ関係のエラーとは無縁と思われているJavaScriptのメモリリーク検出用の“leak-finder-for-javascript”というツールも、Google社から提供されています。

3rd Step: C++で、オブジェクト指向を学ぶ

C++は、C言語のダメさを補うために登場しました。C++言語を学ぶ場合は、C言語のダメさを十分に理解させてくれる書籍を読み、C言語が難しいのは自分の責任ではないことを理解してください。このような書籍は多数出版されています。筆者はC言語のダメさを理解した時点で初めて、C言語を使った比較的大きなプログラムを記述できるようになりました。そして、メンテナンスのしやすいアプリケーションをどうやって記述するかという概念を掴むことができました。

C++が提供する機能をフル活用したアプリケーションを書ける必要はありません。C言語のスキルをベースに、C++の便利な機能を少しだけ使う程度で良いと思います。逆に、C言語の仕様の範囲でC++言語ライクなオブジェクト指向プログラミングができるようになることのほうが、プログラマとしての成長に寄与すると考えています。

4th Step：本格的なプログラミングは、JavaやC#で

ここで初めて、JavaもしくはC#で本格的なプログラミングを行います。Javaは、C++の問題を克服するために、C#は、JavaとC++を超える言語として開発されました。3rd Stepが終わっていれば、JavaやC#の言語仕様を習得するための時間はかからないでしょう。

並行して、SQL(Structured Query Language)やXHTMLも学ぶ必要があります。JavaやC#から、RDB(Relational DataBase)へアクセスできる必要があります。SQLを使わないDBへのアクセス方法や、RelationalでないDBも存在しますが、まずは基本のSQLでRDBへアクセスすることを押さえてください。また、SOAP(Simple Object Access Protocol)やREST(Representational State Transfer)で他システムとの入出力も、使いながら習得してください。

5th Step：アーキテクトを目指して

これが、プログラマとしての最終ステップで

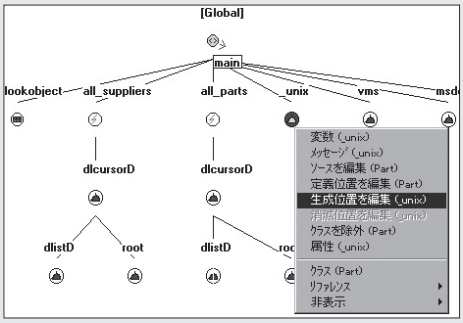
Column

C++のビジュアルデバッガ“LOOK!”

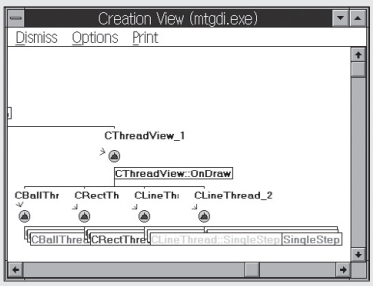
余談ですが、筆者はPurifyとともに、C++のビジュアルデバッガ“LOOK!”という商品も輸入し、日本語化して販売しておりました。

“LOOK!”は、C++で記述されたプログラムが、内部でどのように実行されるのか、つまり、クラスからどうインスタンスが生成・破棄され、どういったメソッドが呼び出されたのかといったことを、リアルタイムに、目で見て確認できるツールでした（**図2、3**）。C++の動作をビジュアルに理解するためには、すばらしい製品です。今、世の中に存在すれば、C++の理解を深めるためにお勧めするのですが、残念ながらもうどこにも存在しないようです。

▼図2　オブジェクトの状態を実行時に表示



▼図3　コールスタックの表示



す。ここまでくると他人が作ったクラス図を見ると、クラスの設計者が、何をどう効率化したくてそのクラスを設計したか、手に取るようにわかります。これが、設計者の開発思想です。逆に、設計者の思想が見えてこないうちは、まだまだプログラマとしては道半ばです。

また今後も、時代のニーズに応じて次々に新しい効率化の手法が登場します。好奇心に従い、新しいプログラミングのパラダイムを追いかけ続けてください。ここで紹介したステップを踏んだ方であれば、新しい言語が登場しても、「この言語には、こういう機能が実装されているはずだ」という予測ができ、言語の習得に時間はかかりません。

人材育成計画は7：2：1で

最後にプログラミングとは少々違うお話です。7：2：1は、某ソフトウェアベンダが実践している人材育成計画の配分です。70％を現場の仕事で経験し、20％を他の人から教えを乞い、10％を机上で学ぶ、という程度の配分で育成計画を立案すると、最も効率が高いという経験則です。

何かを習得するというと、新社会人は机上での学習を中心に考えてしまいがちです。しかし、机上での学習効果は限定的です。机上での学習の倍は、経験のある先輩に教えを乞い、さらにその3倍以上を現場での経験に費やすべきです。机上での学習のために、経験のある先輩から学ぶ機会を削るべきではありませんし、知識や経験が少ないという理由で、与えられた現場仕事を先延ばしにすべきではありません。

新人のころに現場で仕事をするということは、失敗する機会を与えられたと考えるべきです。失敗は課題を認知し、改善する機会になります。開発現場での1回の失敗は、机上で何十時間独学するよりも効果的です。新社会人が目を見張る業績を上げることを期待して仕事を与えるような馬鹿なマネージャはいないはずです。早目に現場で多くの失敗を経験し、すばらしい開発者になってください。SD

第2特集

エンゲルバーグ先生降臨
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?

Javaプログラマを目指したものの、長いコードを受け付けない体質に気づいてしまった。あきらめようとしたそのとき、偶然出会ったアパッチ顔のおじさんが、この体質を「オブジェクト指向向きだ」と肯定してくれた。——可読性、変更容易性、再利用性などを高めるために、クラスやメソッドを小さく作るSmall Object Programming(S-OP)とは。エンゲルバーグ先生による、すぐに実践してみたくなるオブジェクト指向再入門!

■ 著:トム・エンゲルバーグ
■ 訳:長谷川 裕一　HASEGAWA Yuichi　合同会社Starlight & Storm
■ イラスト:髙野 涼香

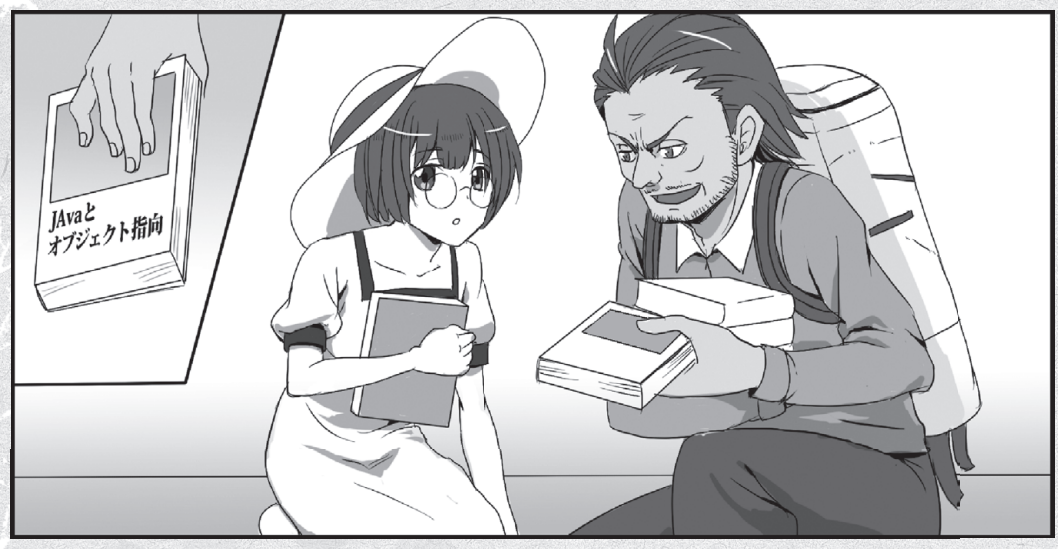
[PEAK 1]
最初の最初

うん、昨年に続いての登場だが、今回もまた、日本の古い友人からの頼みとあっては仕方ない。Javaによるオブジェクト指向プログラミングについて、いつものように好きなように小説風に書かせてもらった。とりあえず、暇つぶしに読んでもらって、Javaプログラミングとはこんなモノなんだとか、Javaプログラミングにはこんな見方もあるんだと楽しんでもらえたら嬉しいな。まぁ、ワシに正解は求めないことだ。そもそも、仕事や人生、オブジェクト指向やプログラミングにも絶対に正しいものなんてないのだよ。人生万事、道化芝居さ!(センバー・ファルシシムス!)。

[PEAK 2]
プロローグ
～オブジェクト指向プログラミングの寓話。いつか、どこかの街で～

この話は、私がまだ、駆け出しのJavaプログラマだった頃にさかのぼる。その日の私は、北アルプスが目前に迫るこの街を、夏の日差しを遮るためのつばの広い麦わらの女優帽を冠り、数冊の技術書を小脇に抱えて歩いていた。正確に言えば、その日の私はJavaプログラマを辞める決心をしていて、小脇に抱えている技術書とは、これからの私には必要のないJavaプログラミングの入門書で、それらの本をどこかに売ってしまうつもりだった。

あてはなかったが、この街は大学生も多いので、本はどこか買ってくれる所があるだろう、それに、今日はたっぷり時間がある。これまで仕事に忙しくて、この街をこうしてゆっくりと歩いたことはなかったので、いろいろなお店を覗きながら、あてどなく歩くのも悪くなかった。

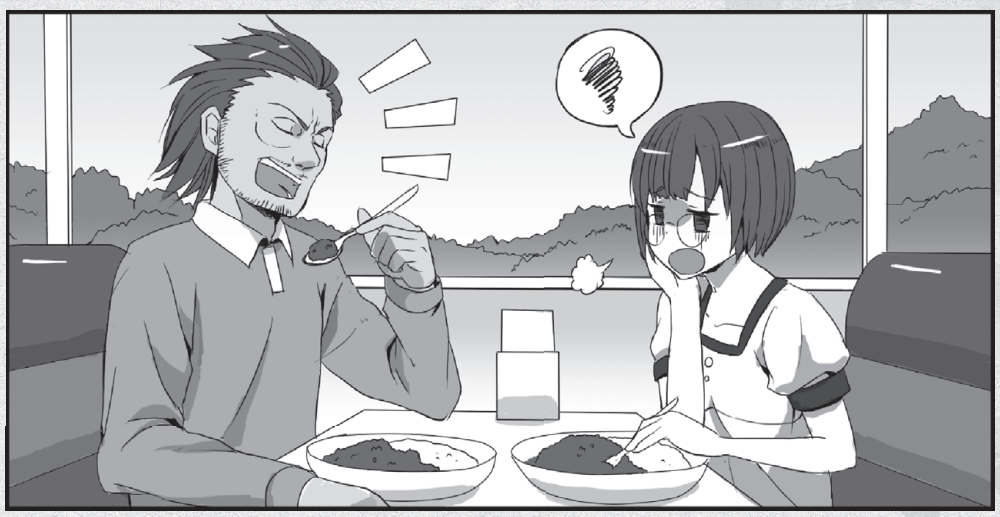


きょろきょろと余所見をしていたせいで、前から歩いてくる人に気がつかなかった。気がついたときには、すでにぶつかる寸前で、驚いた私は脇に抱えていた本を落としてしまった。ぶつかりそうになった人——この街で良く見かける登山者は「ワーオ！」と小声で言うなり、素早く屈むと本を拾おうとしてくれた。「すいません」私も慌てて本を拾おうと屈むと、その登山者は屈んだままの姿勢で1冊の本をすでに手に取り、表紙を怪訝な眼で眺めていた。

「あの……」私が声をかけると、その登山者は「ん？」という感じで私の方を向いた。もじゃもじゃの髪、深い皺が刻まれた赤銅色の肌、人を射るような鋭い目つき……その顔は、ハリウッドの西部劇に出てくるような、悪いアパッチのような顔をしていた。

一応、前置きをしておくと、この話は古いJavaプログラミングと古いオブジェクト指向の話だ。人の手によるJavaのプログラムがオブジェクト指向として正しいものとは限らない。眼を背けたくなるような酷いプログラムもあるかもしれない。でも、それは改めることだってできるという話だ。

[PEAK 3] 最悪な新人研修



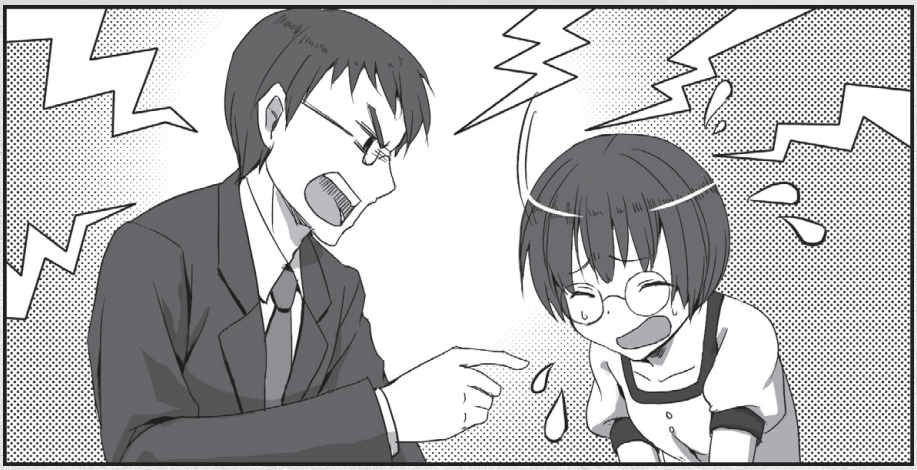
私と、そのアパッチのような顔をした年配の登山者は、よく冷房の効いた、ビブリアという名前の小さなカレー屋さんに来ていた。

「ここはね、ワシが留学していた頃に、山仲間と良く来ていた店なんだ」氷の入ったグラスを口に運びながらアパッチのおじさんは言った。

「やっぱり、外国の方なんですね。こちらに留学されていたんですか」

「そう、随分昔のはなしだがな。で、ここのお勧めはキーマカレーなんだが、それを食べる前に」テーブルに積まれた本の上に手をのせて言った「余計なお世話かもしれんが、なぜ、Javaのプログラマを辞めなきゃいかんのか訳を聞いても良いかな」

「私、長いコードの読み書きができない体質なんです。長いコードを見ると目眩や吐き気がし



て……」私は、これまでの経緯を話し始めた。

数年前、文学部の大学生だった私は、情報システムとかプログラムとか、よくわからずに安定しているだろうという理由だけで、いわゆる、中堅のSI企業に就職した。4月から普通に集合研修が始まり、コンピュータ入門とかネットワーク入門などを受講した。この辺りの研修は文系の私でも何となくこなしていけた。私にとって苦痛が始まったのは、Javaプログラミング研修からだった。そこで、私は長いプログラムが読み書きできないという自分の体質に初めて気がついた。

私にとって最悪だったのは、Javaプログラミング研修の大詰めであるJSP/ServletとJDBCを使った文房具のショッピングサイトの構築演習にはいってからだった。幾重にもネストするif文、RDBにアクセスするための決まりきったコード、そしてtry-catch。演習の解答を導きだすためには、ダラダラと続く50行にもなる長いメソッドや400行にもなる長いクラスを読み書きしなければならなかった。それは本当に気分の悪くなる目眩がしそうな作業だったし、最終日に配布された演習の解答は私のクラスやメソッドよりもっと長かった。これが正解で、これからも本当にこんな長いコードを読み書きしなくちゃいけないんだろうか。私は何気なく講師に質問してしまった。それまで柔和だった講師は、その質問を聞いたとたんに鬼のような形相になり「なに！　そんなことを言っていると、うちの社員じゃなくなっちゃうぞ！」。

それが講師の回答で、そのトラウマが私の長いコードの読み書きができない体質を決定付けてしまった。

「そりゃ無体な話じゃ」アパッチのおじさんは悲しそうな顔をして呟いた。

「新人研修って奴は多分に厄介で、そのショッピングサイトの構築演習って奴にも、受講生が自力でプログラミングして動かすことを通して、達成感を得られるようにするとか、グループ作業で進捗管理をすることが重要とか、もっともらしいことを言う人間がいるのじゃが、最初にそんな長いメソッドや長いクラスを書いたんじゃ、まともなプログラマにはならんのではないかと思うよ。お嬢ちゃんは何も悪くない」

「どういうことですか？」

「新人はHelloWorld程度の話しか理解できないという前提で、最初に最悪のプログラムを見せて新人研修を終わるのはナンセンスということじゃ。もし、そういう演習をやりたいのなら、小さなクラスやメソッドで作られたショッピングサイトを最初に配布して、それに機能変更や機能追加を加えるような研修をおこなうことで、まずは良いプログラムをたくさん読ませるよう

▼図1　小さなビデオカメラ

にすべきだな」

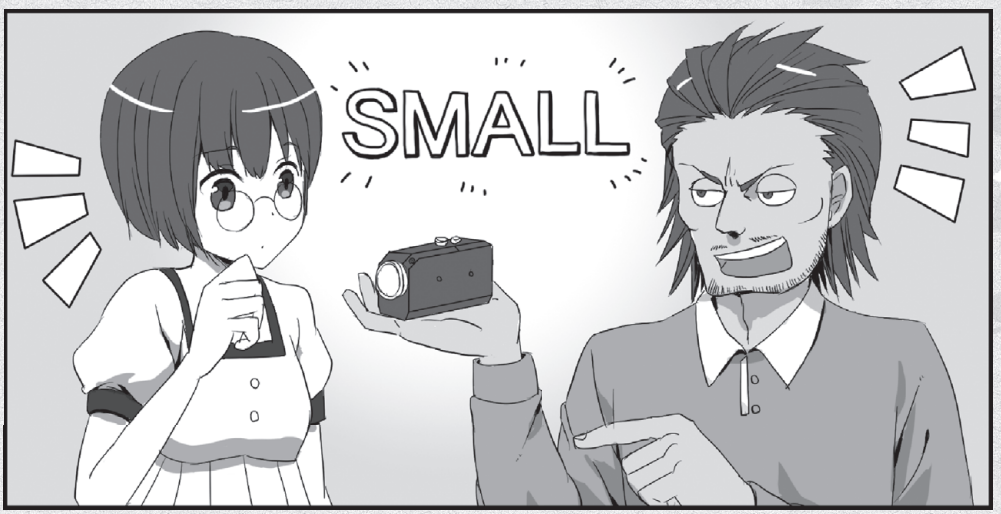
「それは、私の受けた新人研修の演習が良くないってことですか」

「良くないな。お嬢ちゃんが気分が悪くなるのはもっともだよ。オブジェクト指向には部品化を促進するという面があって、そうだな、言ってみれば電気製品のような部品化をしたいんだ。だからクラスやメソッドは、その名前が表す部品として、やるべきことだけをやるように、小さくしなければならん。新人には、そういうプログラムを読ませるべきなのだ」おじさんはポケットから小さなビデオカメラを取り出した(**図1**)。「これを見たことはあるか？」

「いえ、随分と小さな、変わったビデオカメラですね」

「うん、山でアクティブなことをするときに、頭に着けたりして撮影するためのモノなんだが、さて、このビデオカメラ、録画するのにどこを押すのかわかるか？(※編注：カラー写真だと中央部にある銀色の丸いボタンの真ん中が赤くなっています)　ズームをするとき、再生するときはどうだ？　日付を合わせたいときは最初にどこを押す？」私は質問に答えてそれぞれを指差した。

「ご名答。お嬢ちゃんは初めてコレを見たのにワシの質問にちゃんと正解できたろ。もし、今



の質問に答えられないようなビデオカメラだったらどうだ？　買って使いたいと思うか？　思わんだろ、つまりそういういうことだ。どこで何をしているかわかる小さな部品、オブジェクト指向で言えば小さなクラスやメソッドを作ることが重要だということなんだ。そうすることで初めてオブジェクト指向のメリットとも言える可読性とか変更容易性、再利用性などというメリットを享受することができるしな。そう考えると、お嬢ちゃんの受けたような新人研修の演習はダメダメだ。まぁ、新人研修がダメダメなのにはいろんな要因があるだろうが、大抵の教育担当者と教育担当者に教育を要請する現場のエンジニアがオブジェクト指向を理解していないのと、クラスやメソッドを小さく作ることを理解していないからなのじゃ。JavaによるS-OP(Small-Object Programming)をちゃんと理解しておれば、お嬢ちゃんのような体質でもJavaプログラマを続けることができるのじゃよ」

「Small-Object Programming!?ですか……」

Column

最初に教える言語はしっかりと選ぶべし

CMM/CMMIで有名なSEI(ソフトウエア工学研究所)があるCMU(カーネギー・メロン大学)ではコンピュータ・サイエンスのカリキュラムを大幅に更新し、わかりやすさや今後のITの進化の方向を考慮し、プログラミング言語としてJavaを教えるのをやめたらしいな。

その是非は兎も角として、日本の多くのIT関連企業では「弊社の開発現場で使うから」という理由でJavaを選択しているが、その選択理由はナンセンスだ。エンジニアの一生は長いのだ。その一生の中で、エンジニアはさまざまなプログラミング言語を習得していかなければならん。多くのエンジニアにとって、その最初の基礎を作るのが新人研修だということを忘れて、「弊社の開発現場」を理由にプログラミング言語を選択するべきではないのだ。

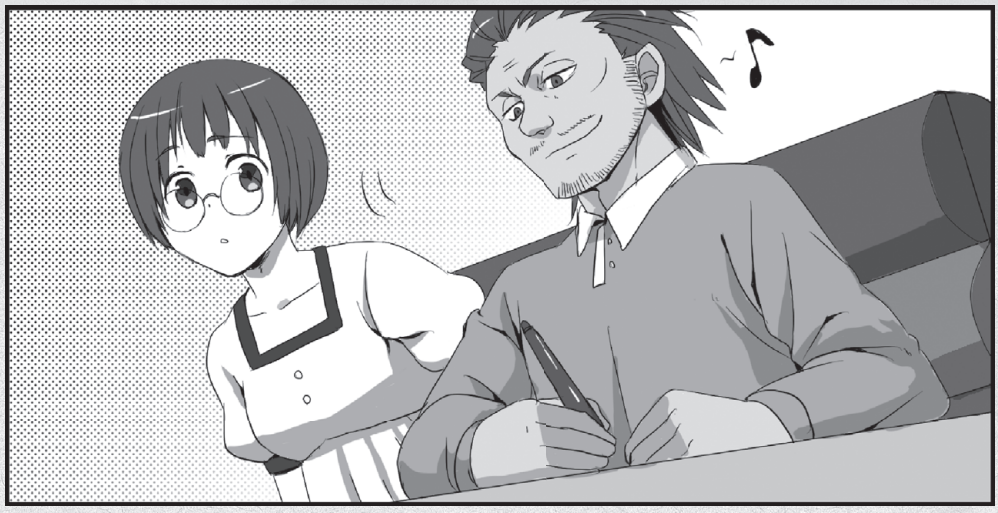
ワシが思うに、新人研修で最初に憶えるべきプログラミング言語というのは、スタートアップが簡単で(たとえばスクリプト的に使え)、新人研修後に、他のプログラミング言語を憶えるときの、基礎となるようなモノが良いのだ。

そうした意味で、Javaは新人研修で扱うプログラミング言語として最適とは言えんだろう。そうだな、今だったら新人研修で使うプログラミング言語はJavaScriptやScalaが面白いと思うな。

まぁ、最後に言っておくが、(マナー講習などを除いた)新人研修とは新人エンジニアの研修であって、新人会社員の研修なぞであってはならないのだ。どこぞの会社の立派な社員を育成するというツマラナイ目標ではなく、人類の財産となるエンジニアを育成することを目標とした公共の慈善事業みたいなモノだということを、どこのIT企業の教育担当者も忘れてはいかんな。もちろん、新人エンジニアもそのつもりで学ばねばならんだろうよ。

[PEAK 4]
Javaと9つのルール

「そうじゃ、古(いにしえ)からあるオブジェクト指向プログラミングのことじゃよ。ただ、Javaはオブジェクト指向言語と言われてはいるものの、処理の記述に関してはC言語から貰ってきた部分がと



ても多い。だから、気をつけてプログラミングしなければ、C言語のプログラムかJavaで書かれたプログラムか一見してわからないことになってしまう。だから、一工夫してJavaでオブジェクト指向的な小さなプログラムを作ろうってことなんだ。うん、そういうこと。最近ではプログラムを小さく作るための練習として9つのルールなんていうのもあるな。細かい説明は省略するから出典の出典にあたってくれ」アパッチのおじさんは、カフェナプキンに9つのルールを書いて(**図2**)、私に渡した。

「へぇ〜。クラスが50行ですか？　それなら、私みたいな体質のプログラマでも読めますね。でも、なんだか現実的ではない気がします」

「いや、そうでもないのだ。1回自分でやってみると良い。それに古からあるオブジェクト指向言語でもあるSmalltalkでは3行程度の小さなメソッドだって普通だしな。まぁ、確かにルールを適用することが不自然な場合も現実的にはある。でもな、これはあくまで練習ルールだ。

▼図2　9つのルール

1. ひとつのメソッドのインデントは1段階まで
2. else句を使わない
3. すべてのプリミティブ、文字型をラッピング
4. 1行につき、ドットはひとつ
5. 名前は省略しない
6. クラス50行、パッケージ10ファイルまで
7. ファーストクラスコレクションを使う
8. インスタンス変数は2つまで
9. getter/setterを使わない

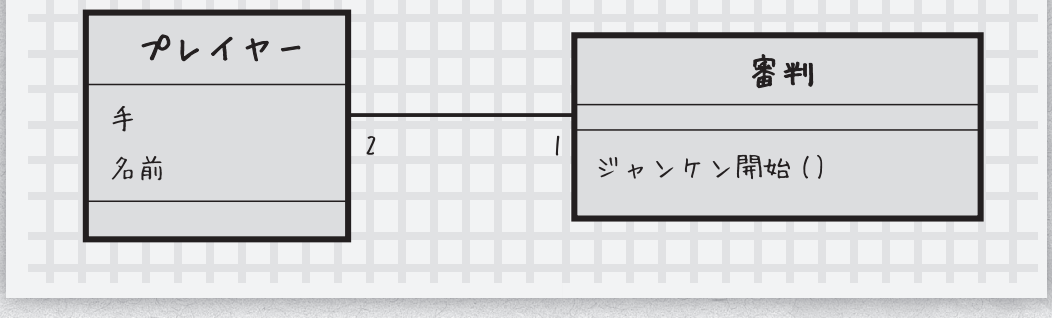
出典：豆ナイト資料「小さなオブジェクトでドメインモデルを組み立てる」有限会社システム設計 増田亨
出典の出典：ThoughtWorks アンソロジー 5章「オブジェクト指向アンソロジー」

まずはルールを守ることで、Javaプログラマを皆、あんたのような体質にする練習なんだ」
「え？　私のような体質ですか」
「そう、長いプログラムを読めないあんたの体質はオブジェクト指向を学ぶものにとっては、正しい体質なのだ」アパッチのおじさんは真摯な強い眼差しで私を見ていた。私の体質は正しい。そのことが、私の中にあった固い澱（おり）のようなものを溶かしていった。
「じゃあ、じゃあ、私はJavaプログラマを辞めなくても良いんですね。新人研修のようなプログラムは小さくしちゃって良かったんですね。9つのルールに従って、リファクタリングして小さく、小さく……」
「うん、まぁ、そのとおりだ」そう言うとアパッチのおじさんはちょっと考え込むような顔をした。「うん、しかし新人研修の解答とされた大きなプログラムを、単に9つのルールを適用して小さくリファクタリングすることに関しては、ちょっとした問題があるかもしれん」
「問題？」
「そう、最初から大きく作られ過ぎたプログラムを何も考えずに小さくしたとしても、問題が解決しなかったり、オブジェクト指向としては不自然な意味のないクラスが出来上がってしまうかもしれんということだよ。つまり小さくするにも、どういうオブジェクト構造になっているべきかを、最初に分析してから小さくしないと、結局、ただの小さなクラスやメソッドができあがるだけで、さっきのビデオカメラの質問の解答のように、何をしているかわかりやすいクラスやメソッドにはならないってこともあるってことだ。つまり、何をしているかわかりやすい小さなクラスやメソッドを作るためには、プログラミングの対象とする問題（その問題領域、大抵は業務の領域をドメインって呼ぶ）のオブジェクト構造をちゃんと分析（分析の過程で出てくる成果物がUMLで描かれたモデル）ができないといかんのだ。まぁ、分析と言うのは仰々しいかもしれんな。簡単に言えば“大きな問題をより分解して理解しよう、そのためには図を描くと良いぞ”ってことなんだ」

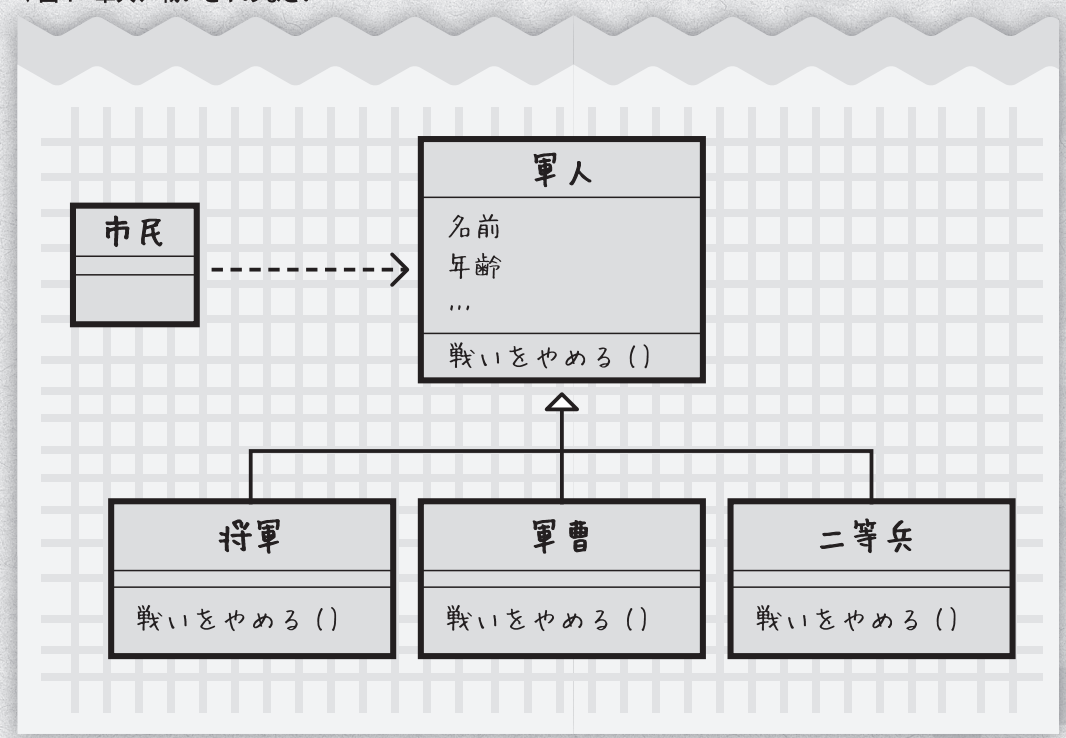
[PEAK 5] ジャンケンをオブジェクト指向する

「たとえば、よくあるオブジェクト指向の入門者向けの課題に“ジャンケン”があるな」
「えぇ、なんか良く目にするかもしれません」机の上に置かれた本に手を軽くのせた。
「ワシはアパッチ族の末裔だから、日本人のようにジャンケンというものを知らない。だから、ジャンケンを理解しなければプログラミングすることはできんし、そのせいでジャンケンをJavaでプログラミングしたものの多くに不自然さを感じるのかもしれんが、あぁ、そうだな、ジャンケンを扱ったプログラムは、世の中的にはこんな感じのモデルであることが多い」そういって、ジャンケンのモデルを描いた（**図3**）。
「普通のクラス図ですよね。ジャンケンのモデルとしてもとくにおかしいとは思わないですが……」
「うむ、オブジェクト指向を理解していると思われている人でも、そう言う人が多いのは知っとるよ。最近じゃUMLがわかればオブジェクト指向ができると思われているしな。だから、ジャンケンのモデルとして、こんなクラス図で納得できるのじゃ。そうそう、“軍人、戦いをやめなさい”なんてモデルでオブジェクト指向を納得することができたとか言う奴が出てくるのもそのせいだ」と“軍人、戦いをやめなさい”のモデルを描き始めた（**図4**）。

▼図3　良くあるジャンケンのクラス図



▼図4　軍人、戦いをやめなさい

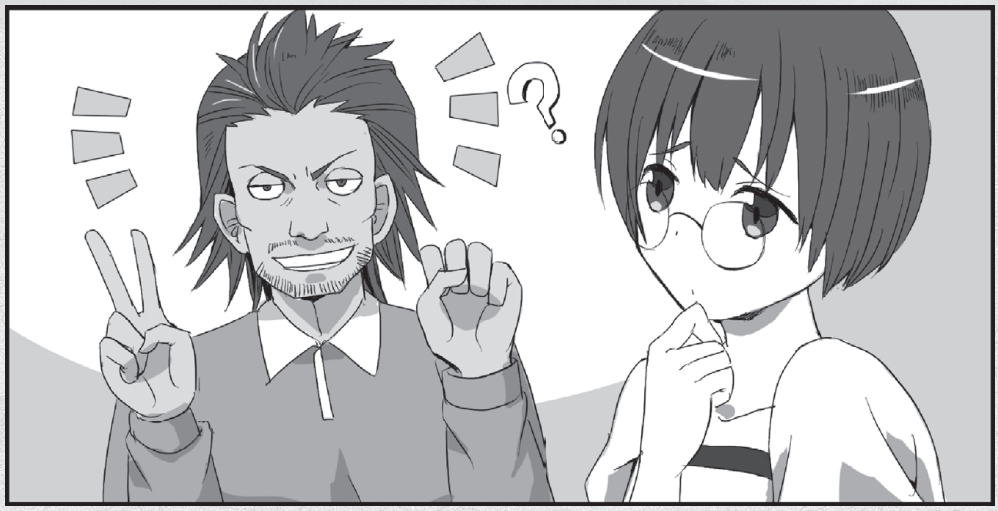


「え、私もこれで将軍も二等兵も同じように扱うことのできる“ポリモフィズム=オブジェクト指向”が納得できたと思ってたんですけど……」

「まぁ継承によるポリモフィズムを理解することが、オブジェクト指向を理解することとイコールだとはワシは全然思わんがね。世の中には継承によってオブジェクト指向がダメになったという人もいるくらいだ。まぁ、メッセージパッシングこそがオブジェクト指向であるとか、流派がいろいろとあるからな」

「メッセージに注目しないで継承を使っているから、このモデルはダメなんですか？」

「全然、違うな。このモデルのダメなところは、二等兵が軍曹に昇進することが困難な点にある。たとえば、サンダース二等兵がいたとして、サ



ンダース二等兵を昇進させたいときにどうするのだ？　いったん、サンダース二等兵を削除して、新しくサンダース軍曹をインスタンス化するのか？　そうだとして、サンダース二等兵のもっていた名前などのデータはどうする？　どこかに保存して、サンダース軍曹に移動するのか？」

「あ、そうですね。まったく気がつかなかった！」

「ポリモフィズムを理解させるにしても、根本的に間違ったモデルを使ってはならんだろ。まぁ、世の中で公開されているものすべてが正しいと思わないということだ。つねに疑うことじゃ。ワシの話も含めてな。

　さて、ジャンケンのモデルに戻ろう。このモデルはさっきのビデオカメラのように、何をやっているかがわかるようになるまでクラスやメソッドが小さくなっていない。つまり分析が足りていないのだ。だから、このモデルをプログラミングするとジャンケンのほぼすべての判定ロジックが審判クラスのジャンケン開始メソッドに集中して、肥大化してしまうだろう。それに比べて、プレイヤーの中身は、名前とかグー・チョキ・パーを数値に置き換えたデータだけになる。このモデルは簡単に言うとトランザクションスクリプトになっているのだ」

「えぇ、そうですね」

「そして、このプログラムを動かしてみたら、こうなったとする」　また、カフェナプキンをコンソールに見立てて、ジャンケンプログラムの動作を描き始めた(**図5**)。

「どう思う？」

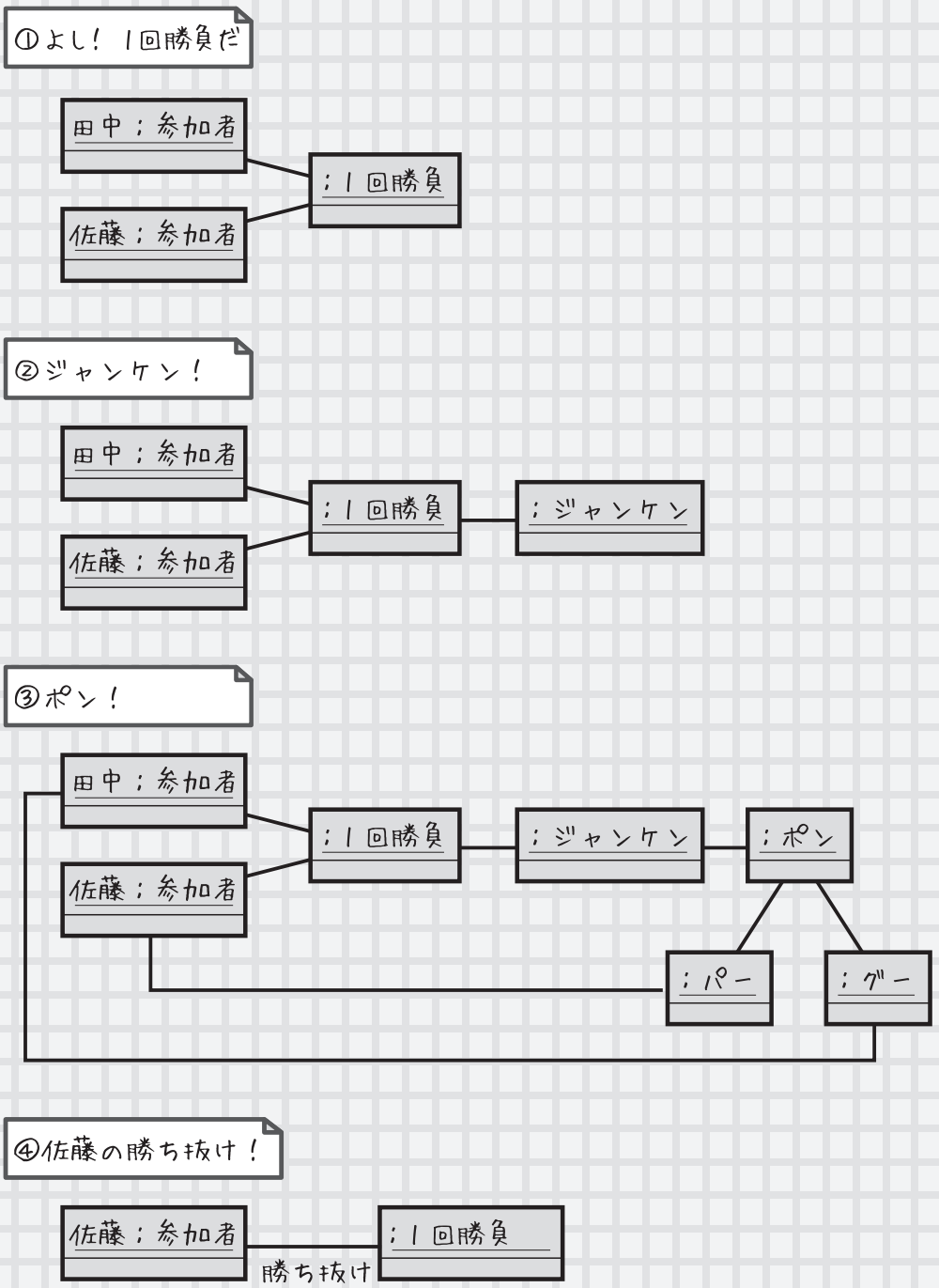
「えーと、普通のジャンケンの3回勝負だと

▼図5　ジャンケンの動作

```
$ java main.JankenSampleRun
【ジャンケン3回勝負】
【1回戦目】
佐藤さんの手はグー
鈴木さんの手はパー
鈴木さんの勝ち
…中略…
【3回戦目】
佐藤さんの手はチョキ
鈴木さんの手はチョキ
引き分け
【ジャンケン終了】
```

……」
「日本人はジャンケンが身近にある分、誤りにも寛容になるのかもしれんな。いいか、ジャンケンの勝負に引き分けなんかない」
「あ!」
「そもそも、ジャンケンとは、1回のジャンケンに複数のポンが含まれ、そのポンは1回のジャンケンの勝敗決着がつくまで続くのだよ」
「え、えぇ、ジャンケン・ポンでアイコだと、次はアイコでショですけど……」アパッチおじさんが険しい眼でこちらを睨んだ。「えぇ、そうですね、ジャンケン・ポン、ポン、ポンって続けることもありますね。えぇ、そうです」
「そうじゃろ。まず2人でおこなう1回勝負のジャンケンで考えてみるとこんな感じとなるのではないか?(**図6**)」
「そして、2人でおこなう3回勝負とは1回ごとのジャンケンで勝者が決まり、最終的にジャンケンに2回勝った者の勝ち抜けとなる」
「そのとおりです」
「うむ、ではジャンケンの分析はこれで完了か?それで良いか?」
「えぇ」
「違う! ジャンケンを思い出すのだ」
「あ、3回勝負のように1対1でやるジャンケンのほかに、大勢でやるジャンケンがあります」
「そう、それは何回勝負だ?」
「えーと、1回勝負です。え? ちょっとまって、5人の参加者でジャンケンの勝負をするとして、最初は3人が勝ちになって、次は3人でジャンケンして、1人が勝ち抜けして勝負は決着する……。そう、1回勝負だけど、1人が勝ち抜けするのね」
「そうじゃ、ジャンケンには"1対1のジャンケンをやって、勝敗数で勝ち抜け"と"大勢でジャンケンをやって、1人が勝ち抜け"という2つの勝負の判定があることがわかる。そう、ジャンケンをちゃんと分析して考えれば、ポンやジャンケン、そして勝負の判定、これだけいろいろと出てくる。日本人はジャンケンを知っているから適当なロジックを思いついて、審判クラスのジャンケン開始メソッドに思いつきの判定ロジックを押し込めて、オブジェクト指向風に値を持たせたプレイヤーを作ってお茶を濁すのでこんなモデル(図3)から適当なプログラムを作るのじゃ。もし、途中でさっきのような勝負の2つの判定に気がつき、審判クラスにそのロジックを書き加えたとすると、審判クラスは肥大化して、訳のわからないものになる。これでは、もし途中で審判クラスが大きすぎると気づいて、審判クラスを分解して小さいクラス群にリファクタリングしても、多分、ジャンケンを表現したモデルとしてはイマイチなモノにしかならないだろう。それに、さっきワシが書いたようなジャンケン・クラスやポン・クラスは、審判クラスを小さくするだけの行為からは出てこないだろうよ。もし、出てくるとすれば、それはジャンケンを再分析しているにほかならないのだよ」
「だから、プログラムを小さくすることを考える前に、プログラムの対象となるドメインを最初に分析、そのためにモデリングをして考え、プログラミングする必要があるんですね」
「そうじゃ」
「でも、もっと言えば大人数のジャンケンには負け抜けっていうのもありますよね。やりたくないことを誰かに押し付けるときとか、あと、順位をつけて勝った人から好きなものが取れるっていう順位付けとか……」
「う〜む、ワシは見てのとおりのアパッチだからそんなことまでは知らんよ。まぁ、ここで言いたかったのは、オブジェクトをきちんと捉えてモデルを描いた方が良いぞってことだ。しかし、動きもしないモデルの分析にこだわり過ぎてもいかん、いつまで経ってもモデルが完成しないからな。ある程度で見切りをつけて、実際にそれで動くのか試してみる、つまり、プログラミングしてモデルが正しいことを検証することも重要になるのだ」

▼図6　おじさんのジャンケンモデル



[PEAK 6]

注文書でS-OPする

「では、続いて新人研修でおこなったショッピングサイトの一部でもある注文書を作成するプログラムを、S-OPで考えてみようかな」と言いながらカフェナプキンに注文書を描き始めた(**図7**)。

「は、はい」

「そうだな、問題を簡単にするために在庫処理とかは考えないことにしよう。Excelで注文書を作るような感じだ。で、たぶん、こんなモデルになる」とモデルを描き始めた。「どうかな?」(**図8**)

「えぇ、そうですね。すごく普通なモデルだと思います」

「うん、そうだな。S-OPをおこなうにしても、分析のモデルとしてはこれで問題ないだろう。しかし、これを普通にプログラミングすると、イマイチなことが多いのだよ。つまり、プログラミングするときにはモデルそのままではなく、先ほど言ったように9つのルールを意識して、S-OPを心掛けないといかんのだ」

「たとえば、このままプログラミングすると、どんなところがイマイチなんですか?」

「そうだな、じゃあ、このモデルを参考に、お嬢ちゃんがJavaでプログラミングしたと考えて、ワシの質問に答えてくれないかな」

「は、はい」

「注文書で扱う通貨を"¥"から"$"に変えたいときには、どのクラスを修正すれば良いのかな?」

「え、え〜と。注文書クラスかしら……、あれ、でも変ね、品目クラスがもっている単価の通貨がそもそも何なのかわからないわ……」

「うん、そういうことだ。じゃあ、この注文書をもう一度見てほしいんだがな」と言って先ほど注文書を描いたカフェナプキン(図7)を指差した。

「これをExcelで作っているとしたらどうだ。単価や小計や合計の通貨を変えたかったらどうする?」

「え〜と、それぞれのセルの書式が通貨になっているはずだから、普通に右クリックして、単位を"¥"から"$"に変えるだけかなぁ、違いま

▼図7 注文書

注文書

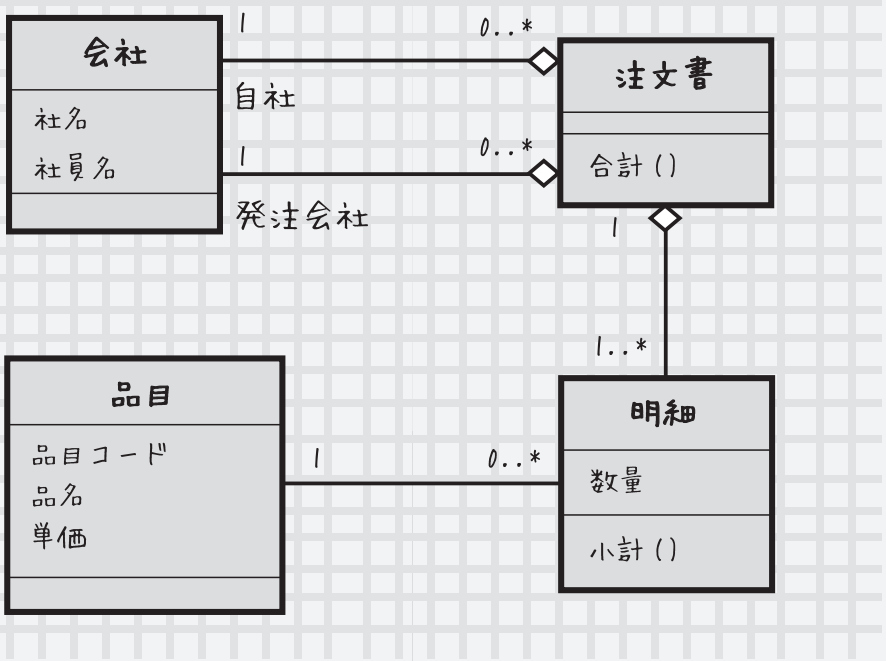
月光商会 斉藤

発注: 星文具 鈴木

合計: 3,000 円

品目コード	品目	単価	数量	小計
A01234c	鉛筆	50 円	40	2,000 円
B85931f	ノート	100 円	10	1,000 円

▼図8　注文書のクラス図



すか？」

「そうだ。9つのルールにある“すべてのプリミティブ、文字型をラッピング”とは、つまりはそういうことと同じなんだ。たとえば、お嬢ちゃんは頭の中でプログラミングしたときに、合計や小計の計算、単価を無意識にプリミティブ型を使ってコーディングしたのだろう、違うかね。しかし、それでは通貨はどのクラスが知っているかわからなくなってしまうのだ」

「えぇ、そうですね。Excelのセルの表現形式と同じように、プリミティブ型をラッピングしたお金クラスを作って、お金クラスでコーディングすればよかったのね(**図9**)」

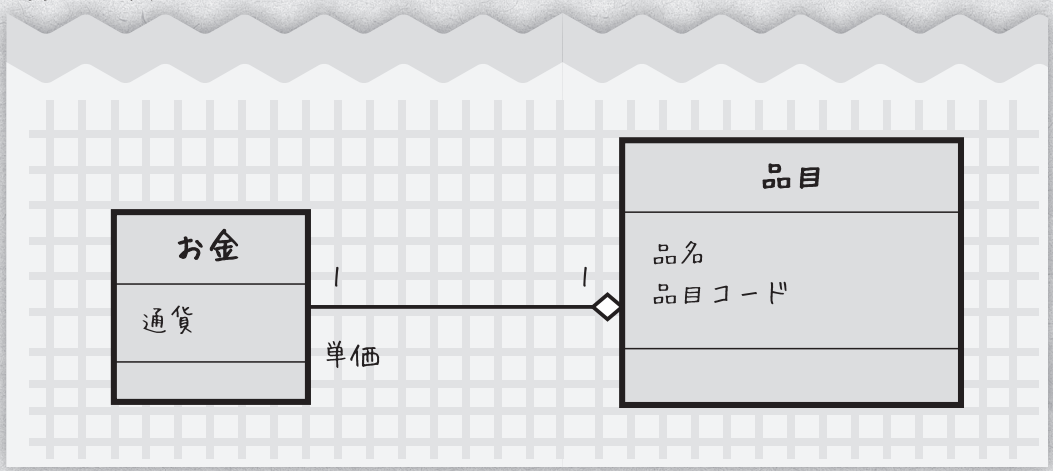
「そういうことだ。同様に、注文書の品目コードがあるだろ。品目コードが英数字6桁であるとすると、それを検証するロジックをもつのは、どのクラスかな？」

「品目コードクラスを作って、そこに検証ロジックをもたせるのね」

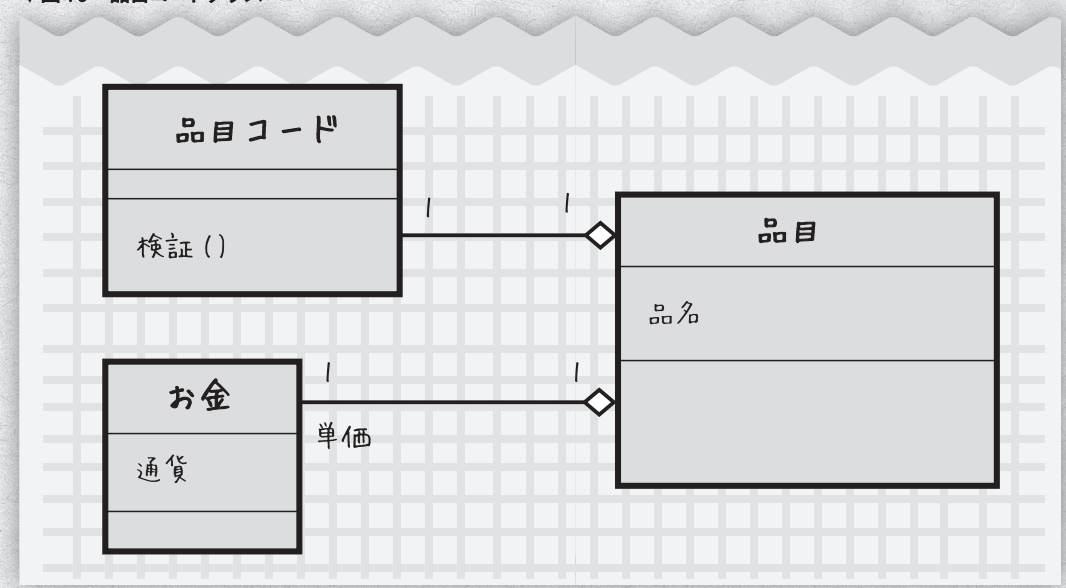
「そうじゃ、わかってきたようだな。もし、品目コードクラスを作らないと、品目コードは文字型で実現されてしまうだろうな。そうすると、実際の受発注システムでは、きっと検証ロジックはさまざまなクラスに散らばってしまう。品目コードは必ずしも品目クラスの中だけで使われるのでなく、画面から入力されたり、品目コードそれだけで使われることもあるしな。そうなると、品目コードの形式が変更されたときに、どこを直せばよいのかわからなくなってしまうんじゃ。さて、品目クラスに関しては、品名もクラスにしてしまえば終了じゃな(**図10**)」

「でも、でも、おじさん。品目コードに関わるロジックが散らばっていても、修正するときにプログラムを品目コードの変数名で検索すれば、

▼図9　お金クラス



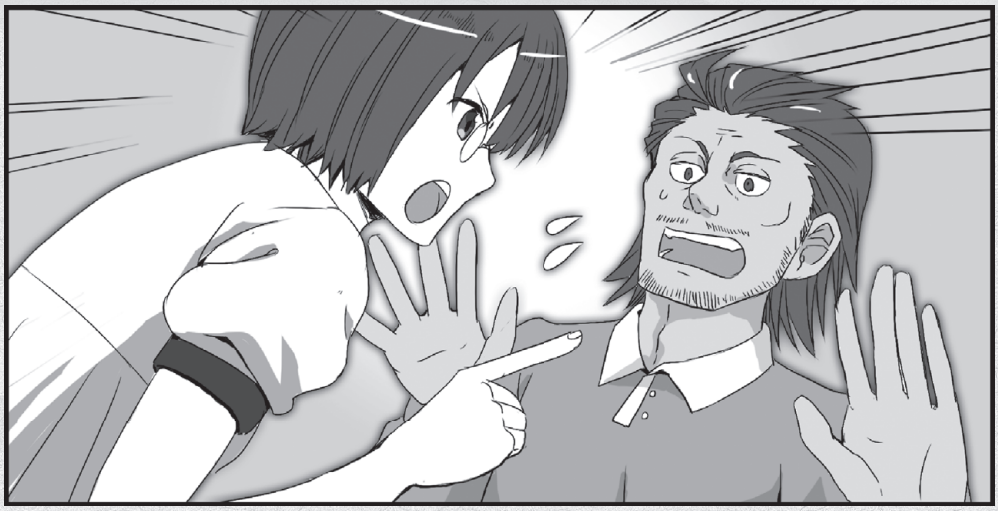
▼図10　品目コードクラス



それはそれで場所を特定して直すことはできるんじゃないですか？」

「検索？　文字型で実現されている品目コードの変数名は何じゃ？　hinmokuCodeかね、itemCodeかな、hinCodeかもしれんな。うむ、何を検索するのだ？　わからないだろ？　実際のシステムで起きている問題は、検索できないために変更時にデグレードするとか、そういうことなのだよ。すべてのプリミティブ、文字型を排除して、お金クラスや品目コードクラスを作ることで、どこでどんなことをしているかビデオカメラの操作のようにわかるし、処理も分散することがなくなるのだ。

　このようにすべてのプリミティブ、文字型をラッピングして直接使わないようにすれば、変更などがあるたびに場当たり的にあっちこっちと適当にロジックを追加／修正したり、わけのわからない変に長いロジックを作ることがなくなり、お嬢ちゃんもJavaのプログラマを辞めなくてすむわけだ」



「そうですね！　嬉しい！　じゃあ、9つのルールと、S-OPについてもっともっと教えてもらえますか？」
アパッチのおじさんの顔が困ったようで、ちょっと曇ったようだった。
「う〜ん、教えたいのはヤマヤマだが、そろそろ東京に帰る電車の時間なのだ。なにしろ、ワシの休みも残り少なくてな……正直なところ、ジャンケンの話で時間を使い過ぎてしまったようだ。まぁ、あとは、お嬢ちゃんが自分で勉強すればよいのだ。こんな本を読むと良い(**図11**)。中には絶版の本もあるが、古書店で手に入るだろう。とはいえ、本だろうが研修だろうがそれは学びの入り口で、本当の学びというのは自分で会得しなきゃならんがな。とにかく、お嬢ちゃんの長いコードが読めない体質というのは、問題ないということだ。長いコードを書く奴が悪いのだ」
「わかりました。9つのルールを意識して、S-OPを実現すれば良いんですね」
「そう、あと、9つのルールだけでなく、Javaでプログラミングするならフレームワークなどを上手く利用することも必要だ。
　たとえば、Javaのデファクトとも言えるSpringFrameworkを使えば、例外処理やLogの出力などを、ドメインを表現したプログラムからは排除することができるから、小さなクラスやメソッドが作りやすくなるはずだ。
　SpringFrameworkやHibernateのValidationフレームワークを使えば、アノテーションで検証することができるから、検証ロジックを減らすこともできるしな。
　ほかにもLombokのようなライブラリを使えば、Setter-Getterを書かないで済むし、toString()やequals()も書かないで良いから、非常にクラスがシンプルになる(**図12**)。
　もっとも、Setter-Getterを書かないためには、そのクラスの属性の扱いを良く考えねばならん。たとえば、今回は注文処理らしきものは省略したが、在庫数なんて属性をもつ倉庫クラスがあったとすると安易にset在庫数()なんてメソッドを書いてはいかん」
「え、どうしてですか？」
「うむ、倉庫クラスの在庫数っていうのは、実際の倉庫の在庫数に連動しているものだろ？　実際の倉庫の在庫数は増えたり、減ったりすることはあっても、突然、20とか30の在庫数にSETされることなんてない。そうだろ？　そういうことを考えずにLombokを使うと小さなクラスにはなってもダメなのだよ。
　まぁ、そうしたことを注意すればS-OPができてくるだろう。そして、クラスやメソッドは

小さな部品群となり、可読性が上がるし、変更容易性も再利用性なども容易になってオブジェクト指向のメリットと言えるモノが手に入るのだよ」

「はい。わかりました。これからもっと勉強して、もう1回Javaプログラマとして頑張ってみます!」

▼図11 おじさんの推奨本

- 『オブジェクト指向ソフトウェア工学 OOSE』
  イバー・ヤコブソン著/トッパン……絶版
- 『オブジェクト指向入門』
  バートランド・メイヤー著/アスキー……絶版の2版も良いが長いぞ
- 『ThoughtWorks アンソロジー』 ThoughtWorks Inc. 著/オライリー
- 『ドメイン駆動設計』 エリック・エバンス著/翔泳社

▼図12 Lombokの例

- @Data アノテーションを付加するだけで、
  Getter-Setter、equals()、toString()、hashCode()の
  メソッドを書いたのと同じ

```
@Data
public class Employee {
    private String name;
    private String code;
}
```

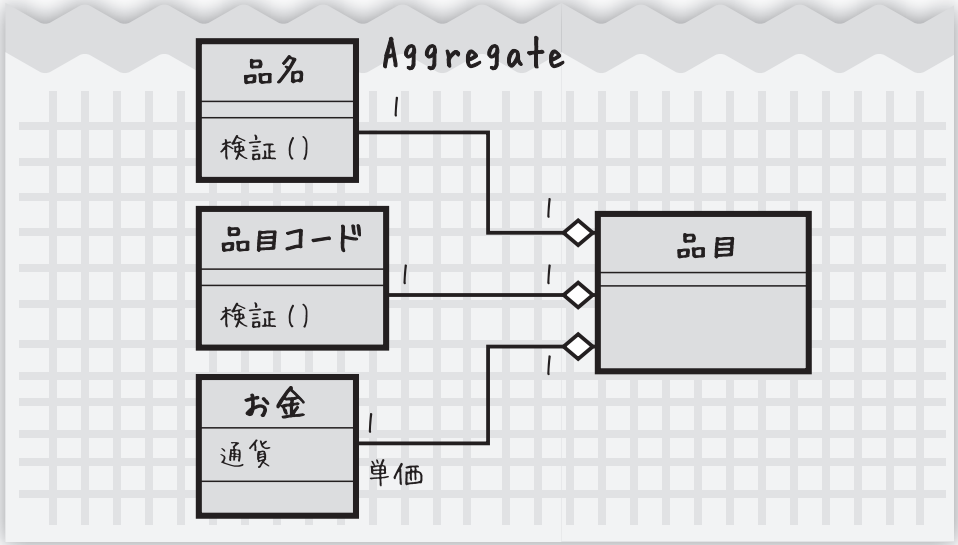
Column

DDDの考え方を取り入れるべし

S-OPを実践で活かすにはDDD(ドメイン駆動設計)の考え方を取り入れると良いだろう。たとえば、時間や人材といったリソースに限りのあるシステム開発の場合、巨大企業の業務領域全体のドメインモデルを作成するのは無理がある。そうした場合、DDDで言うところのCore Domain(コアドメイン)を見つけ出して、S-OPの対象にすると良いだろう。

そして、S-OPを進めてゆくと、**図13**のようなクラスを見つけることになる。そうした場合はDDDのAggregates(集約)となるだろうな。

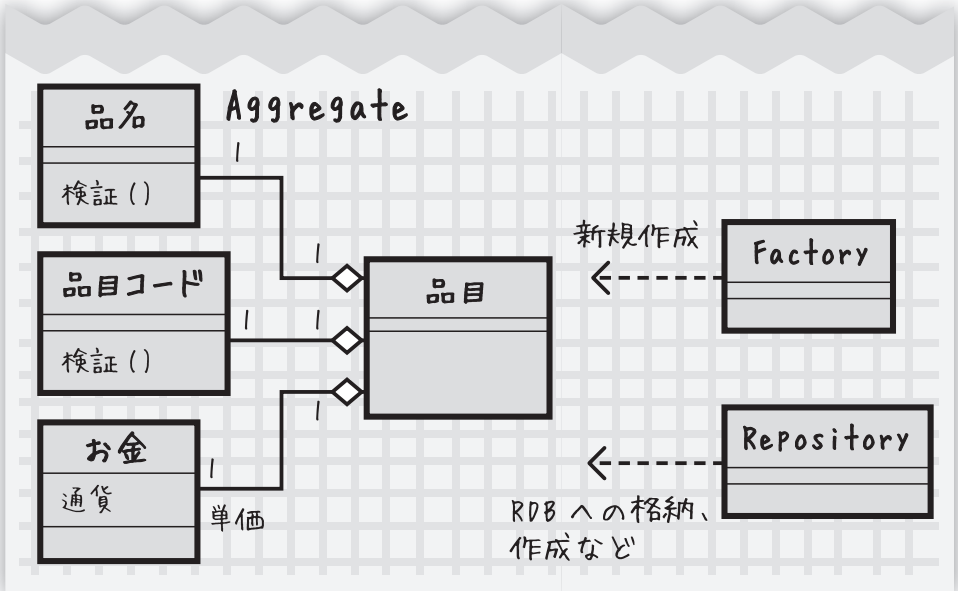
▼図13　Aggregateの例



そうすると、**図14**のようにFactories（ファクトリ）やRepositories（リポジトリ）が適用できるのじゃよ。

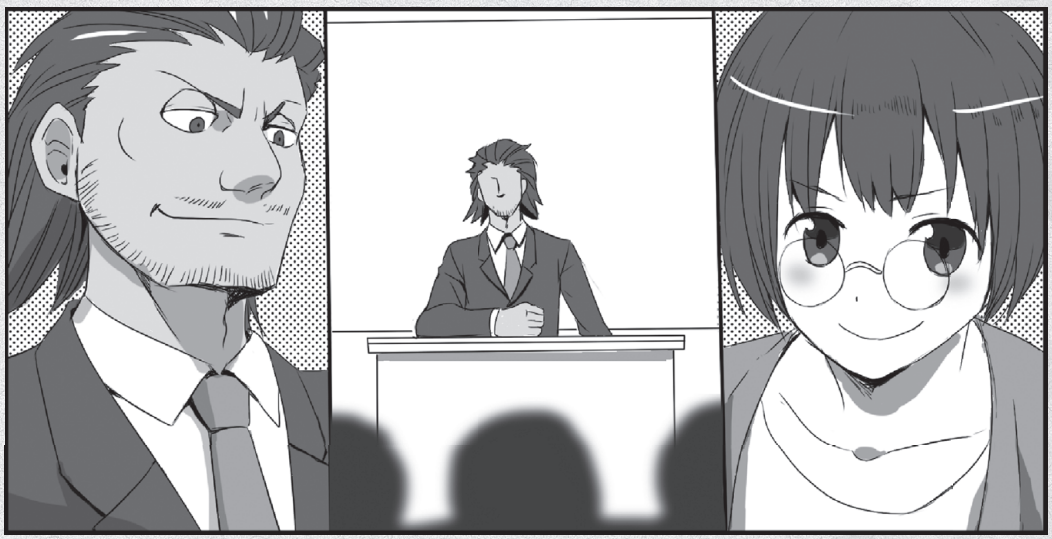
S-OPを実践しようとすると、ユーザと一緒にモデルを書いて、実装して確認するという小さな繰り返しをおこなう必要があることに気がつくじゃろう。このやり方をアジャイルと言っておくと便利だ。会議中にお菓子を食べられるし、嫌な上司にそのお菓子を買いに行かせることもできるしな。システム開発全体をアジャイルと言っておこなう必要はないが、ドメインモデルを構築するまでは、アジャイルと言っておくと良いぞ。

▼図14　FactoryやRepositoryの例



[PEAK 7]

エピローグ



あの、オブジェクト指向に詳しいアパッチのおじさんに会ってから、もう数年が過ぎた。アパッチのおじさんとは、あの日「君が必要だと思ったときに、私はまた来る」という言葉を残して駅に向かって歩く後ろ姿を見送って以来、会っていない。私といえば、あの日以来、S-OPを自分で試行錯誤しながら実践し、社内にも広めたお陰で、今ではアーキテクトなんていう肩書きを貰って仕事を続けていて、最近では、S-OPと相性の良いDDD(ドメイン駆動設計)とアジャイルの社内導入を任されている。

さて、今日は最近問題となっている軽減税率などの税率変更に対して、既存のプログラムのどこをどう修正すれば良いか、上司が探してきた実践派と噂の海外の講師を招いた講演会を聴く予定で、若手のエンジニアで熱気ムンムンの大会議室に来ている。

大会議室のドアが開き、上司が入ってきた。会場のざわつきが徐々に静まり、上司に続いて講師が入ってきた。もじゃもじゃの髪、深い皺が刻まれた赤銅色の肌、人を射るような鋭い目つき……西部劇に出てくるような悪いアパッチのおじさん。

アパッチのおじさんは講師卓につっ張るように両手をつくと、ギロリとこっちを睨み、ニヤリと口元を歪めて言った。

「さて、お嬢ちゃん。軽減税率の問題はS-OPを実践してれば解決しとるだろう?」

私は頷き、ニヤリと笑った。SD

Column

仕事道具にはこだわるべし

モデリングするにあたっても、プログラミングするにあたっても、その仕事の道具選びというのは重要じゃ。昔の大工さんが金槌や鉋(かんな)といったものにこだわったように、我々もパソコンやその中に入っているソフトウェアにはこだわりたいものじゃ。まぁ、何が良いとかそういうことは別として、なにしろ、そうしたものにカネを使うことを、あまり惜しんではいかん。それは、カタチあるモノだけではなく、カタチには残らない知識を得ることにも同様のことが言える。

どうにも、エンジニアの中には5,000円程度の技術書ですら高いとか、技術的なカンファレンスの参加費が何万円で高いとか、会社が出してくれないからとか言う馬鹿者がいるようだが、どんなに高くとも10万円もしないだろう。だったら、そのくらいのカネは自分で払えば良いのだ。それで得られる知識が、自分自身にとっての血や肉となり、自らの価値を高め、ユーザにとって良いモノが作れるようになるのであれば良いではないか。そもそも、エンジニアが得た知識というのはエンジニアという独立した個人のモノであって、会社で共有される所有物ではないのだから、知識を得ることに会社に全面的に依存するのは間違っとるよ。そうは思わないかね？

しかも、そういうことにカネを使わない馬鹿者に限って何十万とかいうスーツや腕時計をしていたりしていて腹立たしい。エンジニアは外観をいくら取り繕っても、ろくなモノが作れないようであれば価値はないのだよ。

OSとネットワーク、
IT環境を支えるエンジニアの総合誌

Software Design

毎月18日発売

5% OFF& 送料無料

年間定期購読のご案内

富士山マガジンサービス版

年間購読なら割引料金で購読できます!

全国どこでも直接お届けしています!

1年購読(12回)

14,580円(税込み，送料無料)1冊あたり1,215円(5%割引)

- インターネットから最新号、バックナンバーも1冊からお求めいただけます！
- 紙版のほかにデジタル版もご購入いただけます！
  デジタル版はPCのほかにiPad/iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく読むことができます。

[月額払い]スタートしました！

デジタル版Software Design

Webで購入

家でも外出先でも

※ご利用に際しては、／~＼Fujisan.co.jp(http://www.fujisan.co.jp/)に記載の利用規約に準じます。

お申込み方法

1 >> ／~＼Fujisan.co.jp クイックアクセス
http://www.fujisan.co.jp/sd/

2 >> 定期購読受付専用ダイヤル
0120-223-223(年中無休、24時間対応)

PART 1 特別企画

SQUARE ENIX + Skeed Presents

ゲームエンジン Luminous Studioが変える ゲーム開発の舞台裏

P2P技術によって大きく性能が向上したゲームエンジンによるアセット管理のしくみとは?

ゲーム開発は、高度なコンピュータ技術を使用し、新しいモノを作り上げる場になってきています。グラフィックスの華麗さや、キャラクターのスムースで美しい動き、臨場感ある音声……などもはや総合芸術と言えます。しかしライバルも多く、ハイクオリティなゲームを制作すべく鎬(しのぎ)を削って開発が進められています。スクウェア・エニックスが開発した「Luminous Studio(ルミナス スタジオ)」は、ゲーム開発を強力に支援する「ゲームエンジン」です。このツールの性能を高めるために、Skeedと協業し、さらに進化したゲームエンジンになりました。本稿では、Luminous Studioの基本機能を解説し、協業によりどのように機能が向上したのか紹介します。

㈱スクウェア・エニックス　**重国 和宏**　SHIGEKUNI Kazuhiro
㈱Skeed　**柳澤 建太郎**　YANAGISAWA Kentarou

ゲームエンジン「Luminous Studio」とは?

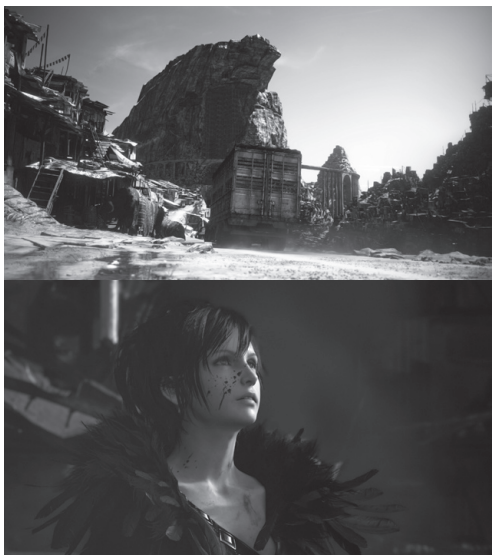
現在のゲーム開発は非常に複雑かつ大規模なものになってきています。古きよき時代には、ゲームタイトルの開発は、まずメモリ管理システムを作って、次にグラフィックスのシステムを作って……というように、その都度ゼロから作り始めることが普通でした。ですが、今現在、多くのアプリケーションで自前のメモリ管理システムを開発しないのと同じく、ゲーム開発においても、ツールやフレームワーク、ミドルウェアを導入することで開発を効率化することが求められています。その結果、おもしろいゲームを作ることにリソースを集中させることができるようになります。ゲーム開発のためのさまざまなツール、フレームワーク、ミドルウェアを統合した開発環境がゲームエンジンです。

現在、スクウェア・エニックスではLuminous Studioというゲームエンジンを開発しており、その最初の成果が昨年(2012年)に「Agni's Philosophy」というリアルタイム3Dデモとして公開されています(**図1**)。ぜひこのデモをご覧ください。

Luminous Studioのアセット管理コンポーネントContentServerとは?

Luminous Studioにはさまざまなコンポーネントが存在していますが、ここではゲーム開発でのアセット(ゲーム制作で使用するデータ:画像、音声、動画、3Dモデルなど)管理に使われるContentServerについて説明します。ハードウェアの進化に伴って、ゲーム開発で扱うアセットの量は膨大になる一方です。たとえば、先に

述べたLuminous Studioの技術デモンストレーションムービーであるAgni's Philosophyの総データ量は2TBになります。実際のゲームではさらに増えることが予想されます。数・量ともに増えていくアセットのバージョン管理・保存をLuminous Studioで担当するのが「Content Server」と呼ばれるコンポーネントです。Content Serverはゲーム開発の経験から得られた知見と、近年の著しい発展を見せている大規模データ処理技術を取り入れて設計・実装されています。このような独自開発の大規模ファイル管理システムを備えていることもLuminous Studioの大きな特徴の1つです。



▼図1　Agni's Philosophy
(http://www.AgnisPhilosophy.com/)

ContentServerの高速なアーキテクチャ

ゲーム開発におけるアセット管理システムにおいて最も重要なことは何でしょうか？　アセットは、普段みなさんがビジネスで使うような一般的なファイルと比較して、非常に大きなサイズになる傾向があります。ですので、大量の大きなサイズのアセットを効率的に扱えることが、ゲーム開発におけるアセット管理において重要な条件になります。

ContentServerは、まさにこの問題に対処するために設計・開発しました。次ページの**図2**にContentServerの概要を示します。巨大なアセットを高速に保存・取得することを第一目標としてContentServerは、たくさんのアセットをSkeedObjectStore(以降、SOSと省略)というSkeedが開発したファイルストアに保存します。SOSはP2Pの技術を利用したもので、分散されたノードにファイルを分割して保存することで、非常に高速にファイルを保存・取得できるシステムです。WEB+DB PRESS VOL.73において、このしくみを詳しく解説しました。掲載の記事『ゲーム開発の未来を支えるLuminous StudioとSkeedObjectStore』をぜひ参照してください。本稿の後半でも、その設計の詳細について説明します。

メタデータサーバ

ContentServerにおいてアセットの高速な取得・保存はSOSが担当しますが、アセットの管理はメタデータサーバが担当します(図2)。なぜSOSとは別にアセットを管理するサーバが必要になるのでしょうか？　SOSは純粋なファイルストアですので、SOSクラスタに投入したファイルに対してハッシュが生成され、そのハッシュをキーとしてファイルが取得できますが、ファイルにハッシュ以外の情報は結び付いていません。ゲーム開発を円滑に進めるには、アセットにさまざまな属性(メタデータ)を付与することが必要です。とくに、アセットにバージョン情報を付与することでアセットをバージョン管理できることが必要です。そこで、アセットにメタデータを付与するためのサーバとしてメタデータサーバが導入されたのです。

SOSのハッシュがたとえば、

```
0x1234cafe
```

であれば、

```
{key = 0x1234cafe, name="/foo/bar/test.bin"ア
, revision = 13, size = 1234 byte, ... }
```

のようにハッシュをキーとして、そのファイルのメタデータを管理します。

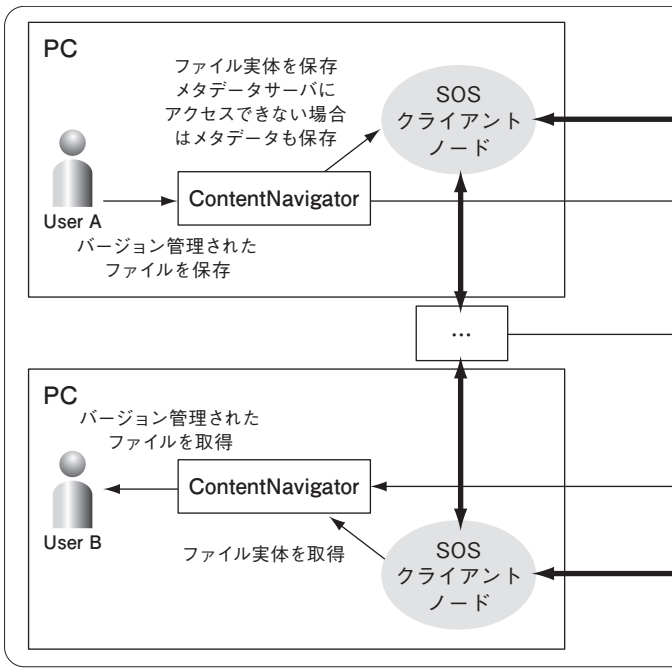
分散システムとしてのContentServer

■アセット管理コンポーネントとしてのContentServerに求められる要件

ここまでで、ContentServerの概要を説明しましたが、次にContentServerに求められる要件をより細かく整理してみます。

①ContentServerは、きわめて多数かつ大容量のファイルを、高い耐障害性をもって保管できなければならない
②ContentServerの容量は、アセット総量の累積的増加に応じて柔軟に拡張できなければならない
③ContentServerは、多数のアクセスが集中する状況下においても、アセットを高い性能によって読み書きできなければならない
④ContentServerは、多数ユーザからの同時アクセスを前提として、同一アセットに対する読み取りや書き込みの競合を調停しつつ、トランザクションを伴う更新を実現できなければならない
⑤ContentServerは、アセットのすべての更新履歴を直列化して世代管理できなければならない

ここには、「スケーラビリティ、パフォーマンス、耐障害性に関する要求(①②③)」と、「原子性(atomicity)や妥当なレベルの独立性(isolation)を備えたトランザクショナルな更新の実現に関する要求(④⑤)」との両方が含まれており、その両者にはある種の緊張が生じているのを見ることができます。

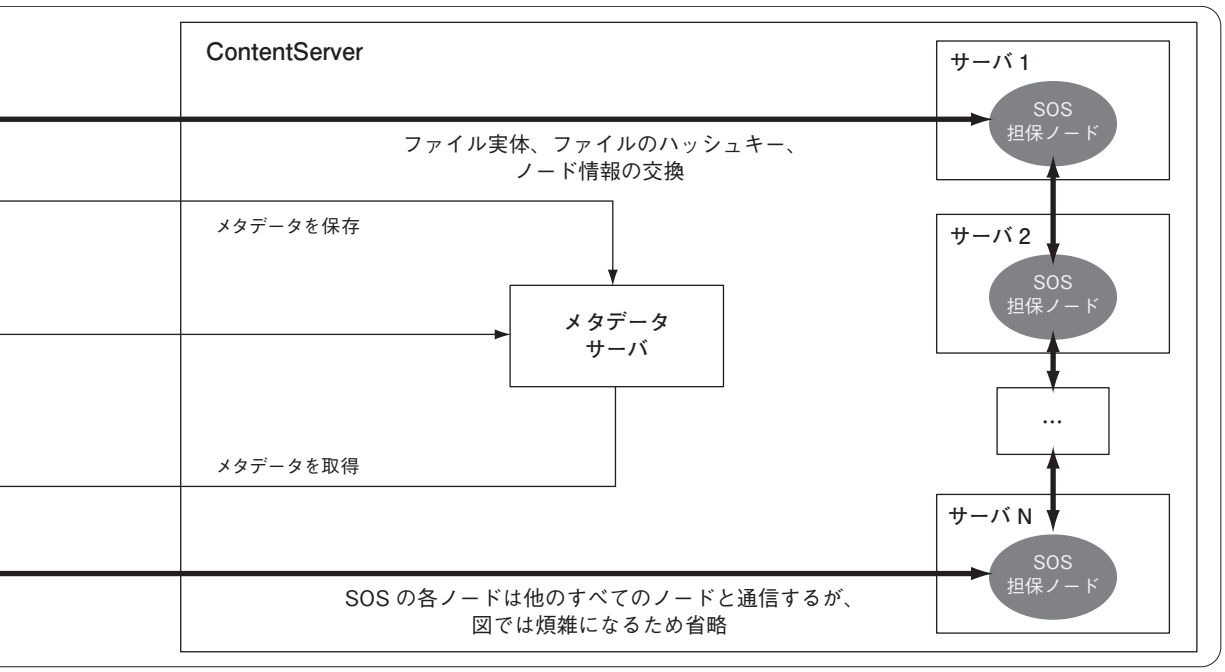


▼図2　ContentServerのアーキテクチャ

すなわち、前者のスケーラビリティ、パフォーマンス、耐障害性などに関する要求は、アセットの保管機能を提供するノードを多数用意してそれらを効率よく協調動作させる、という分散システムの手法によって最も自然に解決できるのに対し、後者のトランザクショナルな更新に関する要求には、そのような分散システムの本質と鋭く対立する面があるのです(いわゆる「CAP theorem」を根拠に、トレードオフが声高に叫ばれがちなところです)。

■中央集権型モジュールと分散型モジュール

このような緊張関係の存在をふまえたうえで、前述の要求をすべて満たすためContentServerに採用された手法は、アセットの名前(識別子)や属性を保存し管理するモジュールと、アセッ



SOSクラスタでファイルを保存するノードが担保ノード、ファイルを取得するノードがクライアントノードと呼ばれる。

トの実体を保管するモジュールを強く分離する、というものでした。そして前者については、中央集権的なアーキテクチャを採用した「メタデータサーバ」というモジュールを利用し、後者については、分散的なアーキテクチャに依拠した「SOS」というモジュールを利用することとしたのです。

このような、「重要な情報を一元管理するモジュール」と「データの実体を分散管理するモジュール」とからなる二重構造のアーキテクチャは、いくつかの有名な分散ファイルシステムにおいても採用されているものです(**表1**)。

このように、分散ストレージシステムを構成するノード群の内部に本質的な役割の区別を設け、場合によっては重要な情報を管理する唯一のノードの存在をも許容(これは単一障害点の存在に直結します)することで、データI/Oの性能と現実に必要とされる機能性とのバランスをとる、というのはある意味定石化された手法であるとも言えるでしょう。

■強化された役割の分離

ContentServerにおけるメタデータサーバとSOSとの役割分担は、このような定石をふまえたものではあるのですが、しかしそのうえで上記のいずれとも異なる特徴を備えてもいます。それは、ContentServerにおける中央集権的なモジュールであるメタデータサーバは、その管

▼表1　分散システムの比較

	管理モジュール	データ保管モジュール
Google File System[注1]	master	Chunkserver
HDFS(Apache Hadoop[注2])	name node	data node
Ceph[注3]	metadata server	object storage device

注1) Sanjay Ghemawat et al., 'The Google File System', 2003.
注2) http://hadoop.apache.org/
注3) http://ceph.com/

理対象とするアセットの実体について、SOS上での識別子を知っているだけで、その実体が具体的にSOSの内部でどのように管理されているかにはまったく関知しない、ということです。

たとえば、メタデータサーバは、あるアセットの実体がSOSにおいてどのノード(群)に保存されているかを、把握できるような設計にはなっていないのです。ゆえに、ContentServerにおけるメタデータサーバは、たとえば表1に示したGFSのmasterとは違い、SOSやそのノード群の管理者ではありません。

この設計によって、アセット実体の保管やI/OをSOSが独自に最適化する際の裁量の余地が非常に大きくなっており、後述するようにエンドユーザのPCもアセットI/Oの戦力として利用するような実装が可能になっています。

■メタデータサーバ障害時の動作

では、ContentServerの単一障害点であるメタデータサーバに障害が発生した場合、ContentServerはどのように動作するのでしょう？　以下、具体的に説明していきます。メタデータサーバに障害が発生した場合、メタデータをメタデータサーバに保存することはできません。ここで重要なのは、メタデータサーバ障害時であってもゲーム開発者、とくにアセットを作成するアーティストの手を止めないことです。メタデータサーバの障害を2つのケースに分けて考えてみましょう。

①ネットワークに障害が発生しておらず、メタデータサーバに障害が発生している場合

この場合はSOSのクラスタは全体としては動作していることが期待されます。ですので、メタデータサーバの障害発生中にアーティストが作成したアセットはSOSクラスタに残すことができます。その場合、どんなアセットをSOSクラスタに保存したかは、後述するContentNavigatorというコンポーネントが管理しています。ですので、メタデータサーバが復帰した時点で、ContentNavigatorが管理していた作業履歴をメタデータサーバに保存します。この段階で、多くの場合、あたかもメタデータサーバの障害がなかったかのように見えます。ただし、もし同一アセットを複数のアーティストが編集していた場合は、何らかのコンフリクト(競合)が発生しますので、そこは事後的に手動でコンフリクトを解消する必要があります。

②ネットワークに障害が発生している場合、またはスタンドアロンPCで作業する場合

この場合、SOSノードはネットワークに接続できませんので、アーティストが作成したアセットはローカルのSOSノードに保存されることになります。この場合はアセットはSOSノードに保存されますが、SOSクラスタには保存できません。ですので、アセットをSOSクラスタに保存することによるメリットは享受できませんが、ネットワークが復旧した際、あるいはPCをネットワークに接続した際に、PCで作成したアセットとその履歴が直ちにContentServerに保存され、そしてゲームプロジェクトのほかのメンバから参照可能になるのは大きな利点です[注4]。

ContentNavigator

ContentServerのメタデータサーバに障害が発生した場合に、ContentNavigatorが活躍することを上で説明しました。ContentNavigatorは、まずはゲーム開発者がContentServerで管理されているアセットに簡単にアクセスできるようにするためのツールです。ContentNavigatorを使うことで、ゲーム開発者は、

注4) Gitの場合であれば、ユーザが明示的にサーバへローカルの作業履歴を統合しますが、ContentServerでは自動的に統合されます。この違いはGitがソースコード管理を主たる目的としている一方、ContentServerはアーティストのアセット管理を主たる目的としているという差から来ていると考えられます。ソースコードと比較した場合、アセットは個々のアーティストの作業の独立性が高く、自動で統合しても問題が起きにくいといえます。

▼図3　ContentServer管理下のアセットにアクセスするためのツールContentNavigator

ContentServerの内部構造を意識することなく、既存のバージョン管理システムと同じような感覚で使うことができます(**図3**)。

ContentNavigatorはContentServerにアクセスするためのツールというだけでなく、分散システムとしてのContentServerの重要なコンポーネントでもあります。先ほど述べたように、メタデータサーバ障害発生時にはメタデータサーバの障害発生を検知して、必要なデータや情報を利用可能な手段で保存し、ネットワークに伝播させます。

分散オブジェクトストアとしてのSkeedObjectStoreの特徴

■SOSの自立性

前述しましたが、大容量のデータを対象とする分散ストレージや分散ファイルシステムというと、GFS、HDFS、Cephなどのシステムがすぐに想起されるところです(**表1**参照)。そして、ContentServer全体を抽象的な観点から見れば、それはメタデータサーバとSOSという2種類のモジュールを利用することで、おおむねそれらの分散ファイルシステムと似たアーキテクチャを採用していることになるのでした。

それでは、ContentServerの一部分を構成するモジュールであるSOSは、それ自体としては自立できない部品の域を出ないものなのでしょうか。たとえば、HDFSのdata nodeだけを取り出してきても、動作もしないし実用性もないというのと同じように。

この問いに対する答えは、必ずしもそうではない、ということになります。SOSはそれ自体独立して、クラスタを自律的に構築・維持し、その中にデータを冗長化して保管し、I/Oにお

いて高いスケーラビリティとパフォーマンスを実現できるシステムです。前に触れたとおり、SOSのノード群によって構成されるクラスタ内でデータをどのように管理するかは、すべてSOS自身が決めているため、外部にname node相当のものがなければ動作しない、というようなことはありません。

■完全な非集中型クラスタ上での、ブロック分割によるデータ保管

それではSOSのクラスタは、その内部にname node相当のものを抱えているということでしょうか。

この問いに対する答えも「否」です。Content Serverは全体として、中央集権型モジュールと分散型モジュールの二重構造になっていますが、その中の分散型モジュールであるSOSは、純粋に分散型のアーキテクチャによって、データの保存と管理とを実現しています。そしてこの分散型のアーキテクチャは、おもにAmazon.comが開発したKey-Value型データベースであるDynamoに依拠した各種の手法[注6]を採用しています。

このDynamo風の手法によって完全分散型の(fully decentralized)クラスタを構築し、その維持を可能にしたうえで、SOSは管理対象データをブロックに分割して、そのブロック群をクラスタ上で保管しています。具体的には、SOSは管理対象データの実体を均等なサイズのブロックに分割したうえで、個々のブロックのSHA-1ダイジェスト値を計算し、そのダイジェスト値がDHTリング上のどこに落ちるかによって、そのブロックの保管担当ノードを決定します。そして、そのようなブロックのダイジェスト値のリストと、少数のシステム管理属性情報とをオブジェクトヘッダとし、そのオブジェクトヘッダのSHA-1ダイジェスト値によって、そのオブジェクトヘッダの保管担当ノードを決定します。

■データの実体による識別子の決定

SOSでは、オブジェクトヘッダのダイジェスト値が、自動的に当該データ全体の識別子として利用されます。つまり、SOSは大容量のデータを対象とするKey-Value Storeの一種だと言えるのですが、そのKey(データの識別子)を利用者が決定することはできず、本質的には、データ全体から構築されたhash treeのtop hashが、自動的にそのデータの識別子として採用されることになるのです(この識別子は、SOSへのデータ登録時にユーザに返されます)。

およそKey-Value Storeにおいて利用者がキーを決定できない、というのがかなり大きな制約であることは疑い得ません。しかしこの制約を設けることによって、同一のキーに対応する値の更新競合、というめんどうな問題が自動的に消滅していることは、システム全体にとっ

COLUMN 「分散システム流行の理由とは」

脚注6で挙げた論文は、「Consistent Hashingによるノードごとの担当領域分担」、「Gossipingプロトコルによるメンバーシップ管理と故障検知」などの各種手法の組み合わせを活用して、非集中アーキテクチャによるクラスタを実現するための具体的なアーキテクチャを示したものとして有名です。いわゆるP2Pや分散システム研究の成果からとくに有用な要素技術を抽出し、そのあとのCassandra[注4]やRiak[注5]などのOSS分散システム流行への道を開いた、読みやすくおもしろい文献ですので、未読の方にはぜひ一読をお勧めします。本稿では紙数の都合もあり、このDynamo論文で解説されている各種手法について個別に解説することはしません。

注4) http://cassandra.apache.org/
注5) http://basho.com/products/riak-overview/

注6) Giuseppe DeCandia et al., 'Dynamo: Amazon's Highly Available Key-value Store', 2007 .

て大きなメリットです。完全な分散システムにおいては厳密な整合性を求めるべきではなく結果整合性(eventual consistency)で妥協すべし、といったよく言われるトレードオフも、またそのような結果整合性を実現するための実装も不要になるのです。

そして、少なくともContentServerにおいては、前述のとおりユーザにとって意味のあるアセット名称(識別子)や属性を管理し、トランザクショナルな更新機能を提供するメタデータサーバが、SOSとは独立して存在する以上、SOSにおけるデータの識別子を利用者が指定できる必要はなく、結果としてこの設計のメリットだけが残る、ということになります。

このような、「データ実体のダイジェスト値を、そのデータの識別子とするオブジェクトストア」というアーキテクチャは、Gitにおけるオブジェクト格納領域(object store)のアーキテクチャに範をとったものです。このようなデータの管理方法は、古い実体データの削除がほぼ発生せず、ユーザにとって意味がある名前とは独立に、実体の変更履歴の系列を管理しなければならないバージョン管理システムにとって、本質的に適した方法だと言えるでしょう。

このような観点から表現するならば、SOSはGitのオブジェクト格納領域(object store)をDynamo風の手法によって分散システム化したものであり、メタデータサーバは、SOSの外部から、それらのオブジェクトにツリー状の親子構造や時系列で管理された履歴などの意味を与えるものだ、ということができます。

■データ担保ノードと非担保ノード

先述のとおり、SOSではDynamo風の手法を利用して非集中型クラスタを構築しているのですが、このクラスタを構成するノードの中に、自ノード内に責任を持ってブロックを保持する「データ担保ノード」とそうでない「非担保ノード」との区別を設けていることが1つの特徴になっています。

このような区別を設けることで、サーバ上で動作するような信頼性の高いノードと、たとえばゲーム開発者のPC上で動作するような信頼性の低いノードとを同じクラスタ上のピアとして動作させながらも、データの永続化は、信頼できるノードにおいてのみ行う、ということが可能になります。このようにデータ担保ノードと非担保ノードとを同一クラスタ内で動作させることで、非担保ノードも自ノードが持っている当該データの構成ブロックを、必要なノードに渡すことで、クラスタ全体のI/Oスループットに貢献できるのです。

このようにSOSが独自に、エンドユーザのPCまでをも利用したスループット向上手法を活用できるのは、ContentServerにおいてメタデータサーバとSOSとの間の分離の度合を強く設定したことのポジティブな効果であると言えるでしょう。

まとめ

スクウェア・エニックスが開発したゲームエンジンである「Luminous Studio」とSkeedが開発したSOSによるアセット管理システムについて解説しました。P2P・分散コンピューティング技術をベースとして、アセットを効率的に管理する機能をどのように作り上げたのか、そのしくみを理解いただけたでしょうか。次ページ以降では、本システムの開発にかかわったエンジニアによる対談を掲載します。SD

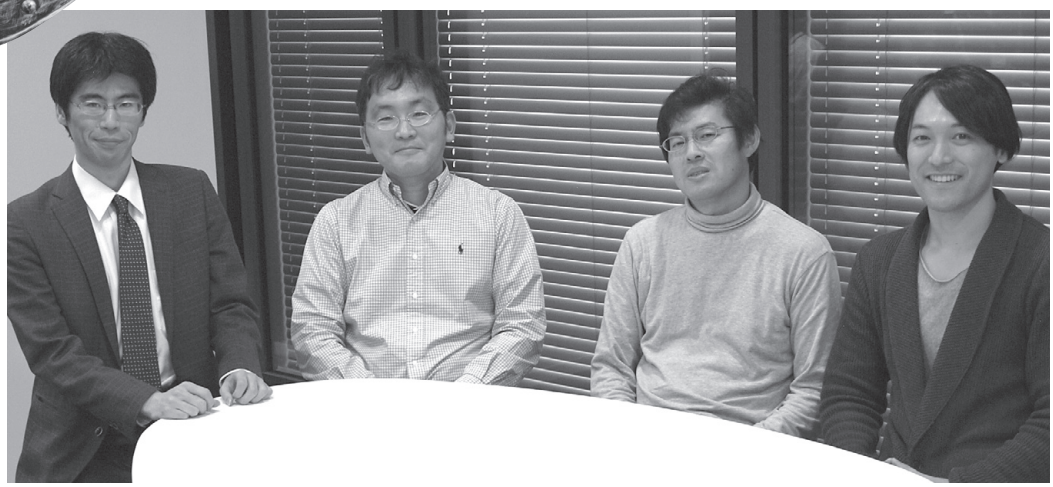
PART 2

SQUARE ENIX + Skeed Presents

ゲーム開発の舞台裏座談会

スクウェア・エニックスとSkeedの技術で生まれたゲームエンジン・Luminous Studioの威力とは？

Software Design編集部・編



皆さんはファイナルファンタジーシリーズをご存じだと思います。ゲームのシーンは実写と見まごうばかりのCGや臨場感たっぷりの音響、そしてキャラクターのリアルな造形と滑らかな動きがごく当然なものになってきています。ゲーム制作の現場では、莫大なサイズになったそれらのデータ（アセット）を管理し、数百名を越えるスタッフで共有します。それらのスタッフが世界各地に分散していることも珍しくありません。こうしたシビアな環境でゲーム制作を進めるためにスクウェア・エニックス社は、ゲームエンジン「Luminous Studio」を開発しています。さまざまなツールやフレームワークを統合したゲーム開発環境のことを、ゲームエンジンと呼びます。ゲームエンジンは、ゲーム本体の開発とアセットのオーサリングをするだけでなく、前述のような背景やキャラクターのCG素材や動画ファイル、音声ファイルのバージョン管理まで行います。そうしたアセットの総量は膨大で、新しく加わった開発メンバーのために環境を構築するのに半日もかかるのも珍しくありませんでした。そこで、独自の通信技術を持つSkeed社と協業を図り、Luminous Studioは、大きく性能を向上させました。現在はDVD1枚分のデータであっても、わずか10秒ほどでダウンロードできます。本稿では、スクウェア・エニックスとSkeedの代表者の皆さんにLuminous Studioの秘密について語っていただきます。

ゲームエンジンとP2Pネットワークの関係とは？

――まずは自己紹介からお願いします。

金子 勇（以降：金子）：Winny作者として、とても名前が売れてしまって困っている金子です。一般的にはP2Pやネットワークの専門家と思われているかもしれませんが、私はもともとゲームやリアルタイム系のプログラミングをやって

㈱Ｓｋｅｅｄ
金子 勇(かねこ いさむ)
ファウンダー兼CINO博士(研究開発担当、工学博士)

いました。たとえばゲームイベントのCEDEC（Computer Entertainment Developers Conference）で、物理演算関連の講演をしたこともあります。大学でもその関連分野をやっていまして、学生を指導する立場でした。プログラミングができる人材として着任したのですが、そんな中なんとなく思いついて作ったのがWinnyでした。一息ついてみると、ネットワークの方で認知されてしまいました(笑)。

柳澤 建太郎(以降：柳澤)：Skeedの技術開発担当取締役の柳澤です。子供のころから趣味でプログラミングをしていました。前職では、Webとデータベースを組み合わせたような業務系のシステム開発をやっていました。Skeedに入社したのが、今から5年ぐらい前です。やはりエキサイティングなプログラミングの仕事がしたかったというのが動機です。今は製品開発をはじめとして、いろいろなプロジェクトの統括をしています。自分でもプログラムを書きます。弊社の事業内容は、金子がやってきた分散コンピューティングやP2Pなどが挙げられます。また最近は、トランスポート層の通信高速化技術と、それを実現する製品の開発に取り組んでいます。

橋本 善久(以降：橋本)：スクウェア・エニックスの橋本です。CTOが、いちばん代表的な肩書です。もともと以前いた会社にゲームプログラマとして普通に入社して、ゲームを作る側にいました。プログラマをしながら企画書を書いたり、ゲームのディレクターもしていました。そこでゲームエンジンと呼ばれる、ゲームを作るためのライブラリ群やツール群を作っていました。弊社でもその経験を活かしてゲームエンジンを作るという仕事を中心にやっています。それが「Luminous Studio」という名前のゲームエンジンです。これを「世界一のゲームを作るための環境」にしようと考え、最先端の技術を導入しようと研究と開発をしていました。そこで、ゲームエンジンのアセット管理システムの部分でSkeedさんと、最先端技術について、いろいろ協業させていただています。あと、ほかにはファイナルファンタジーXⅣの技術ディレクターも兼務したり、さらには他のプロジェクトのさまざまな支援もしています。主業務としては技術を軸に、会社とゲーム開発を改善していくことです。

重国 和宏(以降：重国)：スクウェア・エニックスの重国です。Luminous Studioでは「ContentServer」と呼ばれるアセット管理機能部分を中心に担当しています。そのためネットワーク部分の開発もしています。私が、個人と

㈱Ｓｋｅｅｄ
柳澤 建太郎(やなぎさわ けんたろう)
取締役CTO(技術開発担当)

㈱スクウェア・エニックス
重国 和宏(しげくに かずひろ)
テクノロジー推進部シニアR&Dエンジニア

してネットワーク技術の仕事にはじめて取り組んだのは、セガのバーチャファイター4です。これはアーケードゲームとして、はじめてiモード対応をしたのですが、そのときは社内にネットワーク技術に詳しい人間がまだいなかったので、自分から進んで開発に加わりました。「キミやらない？」みたいな感じでした。それ以降、ネットワーク技術を使ったゲームの仕事を中心にやってきました。

2年ほど前に弊社に転職したのですが、新しいゲームエンジンを作るというチャンスは滅多にないと思い、Luminous Studioのチームに加わりたいというのが、その動機でした。

ちょうど入った直後ぐらいにSkeedとの協業で柳澤さんも参画されることになりまして、今に至ります。その当時、正直なところ私はあまりP2Pはわかっていませんでしたが、金子さんが執筆された本やいろいろな論文を読みながら勉強し、今ではかなり詳しくなりました。ContentServerの開発と運用をうまく進めていくことが私の仕事になります。

――ゲーム開発会社と通信技術会社の協業は珍しいですが、そもそもSkeedさんとの出会いはどこからでしょうか？

橋本：もともとは弊社業務部とSkeed社の方とのかかわりですね。

柳澤：SkeedObjectStore(以降、SOS)の提案より前で、P2P関連のソリューションをご提案したのが始まりでしょうか。たとえば、ダウンロードコンテンツの配信などです。

橋本：セガのファンタシースターオンラインなどで当時使用されていましたね。

柳澤：オンラインゲームのクライアントをインストールさせるために、大容量のデータのダウンロードをどうするかという問題です。オーソドックスなP2P(Peer to Peer)システムを、ファイル交換ソフトから直結する、いわゆるネットワークの高度利用を実現するP2Pシステムとして当社の「SkeedCast」を提案しました。

橋本：SkeedCastの売り込みを受けて、この技術はすごいと思いました。2010年頃に社内で議論しているうちに、この技術の転用先として「我々のゲームエンジンに、P2Pの考え方を応用できないか」と考えるようになったのです。P2Pをゲーム配信に適用するのではなく、ゲーム開発で使われる巨大かつ大量のデータをチーム内に配信するために使おうというものです。作る側の視点でP2Pの考え方を応用し、生産

㈱スクウェア・エニックス
橋本 善久(はしもと よしひさ)
CTO(チーフテクノロジーオフィサー)

性を向上させることができないかという議論が始まりでした。

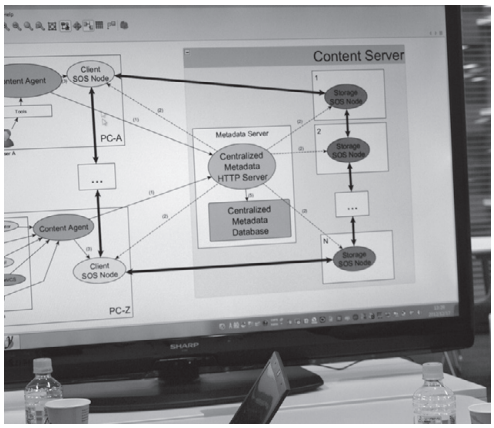
柳澤：P2P技術はI/Oのスケーラビリティを実現できます。すごく抽象的な言い方で申しわけないですが、単一障害点をなくして運用コストを下げること、高額な専用アンプライアンスを何千万円もかけて購入しなくても、1Uの安価なハードウェアを組み合わせて、高度なシステムを構築できることが挙げられます。P2Pのポテンシャルは非常に高いのです。P2Pの種類もいろいろありますが、実はファイル共有ソフトとP2Pはあまり密接な関係ではないのです。

重国：純粋なP2Pでやるべきかどうかで議論になりました。

金子：クライアントが1万ノードを超えてくるとWinnyのような分散的な使い方もありますが、1万ノード以下ではセントラルサーバを置いて、インデックスファイルはそこで管理する設計が普通だという話があります。

橋本：当初、Luminous Studioで使うSOSもセントラルサーバがない状態から始まってましたよね。1万ノード以下ならば、セントラルサーバを作る方向で作ってみようと始めたわけです。

柳澤：セントラルっていうのはアセットの情報が外から与えられるという意味では、もちろんセントラルですが、それは音楽配信P2PのNapsterがセントラルであるというのと意味が違います。そもそもNapsterのネットワーク自体が、中央サーバがないと完結しませんが、SOSはそれ自体で完結します。そこで問題はファイル共有ソフトには検索が必要なことにあります。属性による検索が必要なのです。キーによって値をとってくるとか、ファイルをとってくるなどです。



金子：検索がP2Pのキモなんですよね。検索機能は、Winnyでも手を抜きましたが……。

橋本：そうなんですけど、Winnyが刺激的なのは、そこじゃないですか？

金子：受けましたが、その当時は素人が考えたからなんとも言えません(笑)。

柳澤：P2Pで属性による検索は難しい問題です。今でもちゃんと解決されていません。

Luminous Studioへの SOSの適用

――最初に実験をいろいろされたと思いますが、そのエピソードを教えてください。

柳澤：2011年の年明けぐらいから、弊社で開発したソフトウェアを毎週持参して、デモをしながら、実験・デプロイを半年くらい続けたでしょうか。

橋本：そうですね、いつの間にか夏になってましたね。1つのクライアントが複数サーバにアクセスするという単純なシステムでしたね。そうして試したSOSの売りの1つとして、担保

ノードでファイルを分散して担保し、パラレルに取得するマルチパスが挙げられます。

重国：マルチパス化もありますが、単一の担保ノードからのファイル取得の高速化もありました。性能が出るまでけっこう大変だったという記憶があります。プレスリリースにも出しましたが、最終的には500MB/秒で、つまりDVDサイズのデータも10秒でダウンロードできるようになりました。この場合、回線として10Gb Ethernetが必要ですが。柳澤さんやSkeedの皆さんに頑張っていただきました。最終的に、ファイル取得の高速化は結構いいところまで来たと思います。単純に回線にばっとデータを流すだけならもっと速度が出ますが、Luminous Studioのしくみとして、当然のことながらファイルを分散配置して、どのファイルがどの部分にあるか管理したうえで、それだけの速度を出す必要があります。これは世にあまたのソフトがありますが、その中でも良い成績が出たと思います。

皆でサーバにアセットをわーっと取りに行くと、サーバが重たくなって、なかなかダウンロードできない。コーヒーでも飲んで30分後に戻ると、まだ終わってない。じゃあ、散歩にでも行こうか……みたいな（そんな人はいないと思いますけど）、そういう雰囲気がでてしまうことがあります。なので500MB/秒というのは、ファイルサーバの性能として非常に良い数字です。それをうまく活用すればゲームの開発効率も向上し、ゲームエンジンとしてもいいのかなという感じですね。この実験はかなりしつこくやったところですね。

――テストは通信速度が中心ですか。

橋本：速度がまずは中心ですね。最大の目的が速度を出すってところにあったので。そういう小さくて細かいファイルがすごく大量にあるとか、その逆であるとか、その両方をどう満たすかというテストです。現時点でもまだ検証中であるとも言えますが、とくにこのサーバからローカルにダウンロードするところが先にできあがったんですね。

――性能測定の具体的な方法は？

柳澤：何を計測したいかというと、たとえば複数のコモディティハードウェアを1個のクラスタに組み上げて、それに対するI/Oの絶対的なパフォーマンスはどうかということと、それがノード数の増加に応じてスケールするかということを調べたいので、単純にEnd to Endで計ればいいんですね。何MBのファイルをクラスタに投入し終わるのに、レプリケーションファクタとか、しかじかの条件下で何秒で終わったからユーザにとって意味がある実行スループットは何MB/秒であるといったごく普通のものです。

重国：結局、ゲーム開発者がクリックして、データが手元に来るまでの時間が最終的に重要になるので、今柳澤さんが言われたとおり、リクエストを出して分割されたファイルを取得して、最終的にファイルが全部完成して、利用可能になるまでの時間が基準でした。500MB/秒とは

そういうことです。

橋本：やはり1個1個問題をつぶしましたよね。ボトルネックがハードウェアなのか、それとも、ファイルをダウンロードする前のSOS側で重いのか、ダウンロード後のファイルの再結合のところで重いのかとか……。

柳澤：結局のところクラスタでファイルを保存するとは何かというと、いわゆる伝統的なファイルI/Oのシステムと同じように、大きなバルクデータを小さい均等なブロックに分割して、それを、ある数学的な演算によってどのノードに保存するのかを決めて、書き込むときはブロックを飛ばす、読み取るときには取ってくるっていう……そういうことをやっているわけです。つまり飛ばされる先のディスクI/Oだとか、スレッドのキューイングの問題もあり得るし、すべてを多重に、うまく帯域を消費して持ってこられるか、持ってきた後でバラバラに届いたブロックを再度ストリームに結合して再構成して持ってくる際に、バッファをあふれさせないようにできるか、などなど考えるべきことは山ほどありました。加えてCPUの計算時間も単純に問題になります。チューニングパラメータが何であるべきかとか、そもそも負荷試験プログラムに無駄がないかとかも調べました。

橋本：理想を追求していくと、最終的にはネットワークが一番ボトルネックになるだろうという話が、かなり早期の段階で弊社の情報システム部などからあり、弊社のオフィス移転を機に一部環境を10Gb Ethernetに変えたのです。

金子：制作スタジオでは、End to Endで10Gb Ethernet環境というのは普通だったりしますよね。

橋本：それでもやはり情報システム部は最初は驚いた表情をしました。でも面白い挑戦ですね、ってことでかなり乗り気になって協力してもらえました。10G対応ネットワーク機器の値段が下がるタイミングも予想しました。もちろん全社的にまだ全部を10Gb Ethernetにしたわけではないんですけど、少しずつ様子を見ながら変えています。で、このLuminous Studioができあがっていくところに合わせて、全体をリプレイスしていくのも込みで計画をかなり前に立てたんです。さらに究極的にはサーバ側のハードディスクも、もしかしたらSSDに変更したりすると、キャッシュとして持つとしたら、場合によっては理想的に動作すると、メモリだけでディスクがいらなくなるのではないかとか仮説の話までしました(笑)。

P2P導入で対応すべき課題は何か?

——速度以外のほかの機能要件はどうでしょうか？

橋本：スケーラブルにサーバサイドも変化できるように、なるべくスモールスタートしたかったのです。あとは外注の問題ですよね。たとえば海外のスタジオとやり取りするというときに

も、システムがきちんと稼働していて、東京と海外でどう協調して動かすかとか、そういうところもちゃんと視野に入れて設計していただく必要がありました。

重国：アセットの制作を外注さんにお願いしたときに、こっちから限られたものだけをお渡ししたいけど、向こうからできたものはこっちに引き取りたい、ということがよくあります。なので、P2Pなんだけどアクセスコントロールをきちんとやりたいとか、けっこうこれもP2Pとしては難しいことですけど、やっぱりゲーム開発としてはどうしても必要になってくるところなんですね。なのでやっぱりそこも知恵を絞っていただいているところですね。

――この要望は難問だったんですね。

柳澤：難問といえば難問ですね。

金子：結局、ノード数を制限してもいいんだったら可能です。100万ノードでやってくれと言われたら頭が痛くなっちゃうんだけど。

柳澤：一応こういうI/Oのシステムって究極のところ、人工知能がコンピュータサイエンスにとってずっと夢として牽引役を果たしてきたのと同じように、こういうスケーラブルでハイパフォーマンスで、運用が楽なI/Oシステムって、エンジニアリングにとってずっと夢の領域であるものです。ですが、あるメリットを重視すればそれによってトレードオフの関係にある別の面が損なわれてしまうという現実があります。とはいっても、なるべくせっかく実ユーザである、スクウェア・エニックスさんがいらっしゃるので、実用性ということに対して手を差し伸べることはすごく重要なことだと思っています。それがどういう形で提供できるかは、カスタマイズなのかパラメータチューニングなのか、根本的なアルゴリズムの変更なのか、常に我々にとっても課題ですよね。

金子：P2Pの欠点としては、やっぱり動作の不安定性や不確実性というのがあるんですけどね。全体を見渡せないからですね。あんまり大きくしてセントラルサーバ方式をやめてしまうと、見渡せるものがなくなってしまうから、そこは制御が難しくなるのは確かですね。Winnyだっていろいろ管理しているんです。ローカル部分まで落ちないようにしたり。

柳澤：その上限がいくつなのか、シミュレーションしていることも当然あるでしょうし、現実的な落とし所が、どこにあるかっていうのはなんなんでしょう？

橋本：1万ノードがP2Pの限界とおっしゃっていましたよね。

金子：経験上ですね、結局1ノードでさばける限界っていう感覚があるので、1万あたりでちょっとどうしてもどこか1つのノードに集中するところがあるとそこがボトルネックになってしまうというのがあります。

――セントラルサーバがなくても自律的に動くP2Pシステムが最終的な目標ですか？

橋本：いやそうではなくて、そういうふうにしないで、安定させる方向へ持っていくのでどうですかねっていうのに対して、金子さんとして、こっちのほうが現実解なんじゃないのっていうことなんです。

金子：Winnyもそうなんだけど、自律分散でまったくセントラルサーバがないっていうのはやりたくてやってるというよりは、しょうがないからやっているわけであって、別にやらなくていいわけですよ。それにこだわる必要がないんで。はじめからそうしておかないと、そこがボトルネックになるということです。100万ノードを目指すのならばセントラルサーバ自身も分散しないと。

橋本：やはりアセット制作の中で使う分には、ノード数が100万とかは現実的にはあり得ないので、柳澤さんとしては、そこはいろいろ考えていたのでは？

柳澤：難しいところですね。たとえばHadoopは今でもネームノードがありますが、このことばかり文句をいう人が世の中にいるじゃないですか。そのせいでひどい目にあった人が世の中に何人いるか知りませんけど、Hadoopのネームノードは単一障害点であると。根本的には、属性で検索するっていうこと、しかも高速でやりたいということと、ズバリ識別子をシステムにとってもユーザにとっても、一意の識別子を指定して、稲妻のように高速にズバッとヒッティングしたいというのを両立するのは難しいわけですよね。それでこの5年ぐらいいろいろ試みはされていて、基本的には構造化オーバーレイを使ったDHT（分散ハッシュ表）ベースで稲妻のようにひいてくるのを実現しつつ、いろいろなフラディングの手法を工夫することでトラフィックをあふれさせずに、フラッドしなきゃいけない情報の数やノード数の増加に対してなるべく時間も空間も落ち着くようにというのは工夫されているところです。

ゲーム開発者の特徴とは?

――協業していて、開発に関する考え方の違いはあったんでしょうか?

橋本: プロセスの違いみたいなのは少しありました。弊社でもチームによって仕事の進め方も全然違いますし。

金子: ゲームは、全部やらないとだめですからね。ネットワークは、もちろんのことながら、実はすべて得意ですよね。

橋本: そこはそのとおりですね。ゲーム屋のほうがハードウェアとかコーディングの限界のところを一番攻めたがるというか、設計上良いところのさらにもう一段ギリギリまでっていうところがあります。SOSのプロトタイプができあがったときに、もっとここを速くする余地がないですか?……みたいな。

柳澤: そういうことを、最後におっしゃられて(笑)。

橋本: そうそう、最後に搾り取るみたいな。

金子: 本当は、私はゲームよりなんで、どちらかというとハードウェアの限界までたたいてやろうとか思っちゃって、どんどん下にもぐっていくタイプなんだけど、結局専門もそこになっちゃいましたので。柳澤は違うんですよ。ネットワークはそういうあんまり危険なことはやらないほうがいいという立場です。

橋本: それなので当初かなりこちら側でどこにボトルネックがあるかというのを細かくみて、ここのところが、原因があるんじゃないかみたいな話し合いはさせていただきました。

重国: 動作原理を理解したうえで、こうしたら速くなるのではと提案しました。最初はたぶん100MB/秒とか200MB/秒ぐらいで、もちろんネットワークの制約がある状態でしたが、それによってどんどん速くなっていきました。

橋本: 確かに速いは速い、他と比べても。でももっと速くしなきゃねっていう話をしていて。私たちがやりたいのは画期的に速いものであって、もう一段踏み込みたかった。

Luminous Studioの今後の展開

――ユーザが意識しなくても、自動で使えるしくみを実現したのですね。

橋本: そうですね。普通だったらたとえばPerforceとかSubversionとかでデータをシェアしたり、あるいは普通にファイルサーバを使い、そこに手動でファイルを置いてメールで連

▼図　Luminous Studio　©2012 SQUARE ENIX CO., LTD. All rights reserved.

絡して、といったやり取りすることが多いと思うんですけど、そこもツール経由で全部接続されていて、かつ高速にいろんな検索ができたりとか便利機能がいっぱいあって、っていう独自のアセット管理機構を構築しています。それをSkeedさんと一緒に作りあげてきました。

柳澤：そういう意味では、普通のI/Oツールは、たとえばOSのファイルシステムとか、DBMSを使うと思うんですが、その一部を代替するものと思っていただければいいと思います。

金子：ゲーム会社のほうで作っちゃうというのはすごいですよね。

橋本：今後のゲーム開発で、データが巨大化し多様化していく中で、既存の製品だったら、限界が来るであろうと予測されていて、今回Skeedさんといろいろやるチャンスがきたので、せっかくなら踏み込んでしまえと考えたのです。

金子：作ってしまうのであれば、自分の好きなようにするのが一番ですからね。

橋本：最終的にそのできあがったものがSkeedさんのサービスとしてリリースされる部分もあるだろうし、弊社はゲームエンジンの中に組み

込まれたものとして活用していくわけです。

――いろいろなことに応用できそうですね。

橋本：そうですね。たとえばゲームエンジン自体が映画産業とかでも使われる時代が来るだろうって実は僕たちは想定しています。そうするとゲームよりさらに巨大なデータになるとか、それも含めてそういうのにも耐えられるシステムを未来図として描きながら考えています。一般のゲームエンジンは弊社ほど広くやってないっていうのがあって、一般的なバージョン管理システムを使って、それ以上のことは考えないのがほとんどです。

Luminous Studioを導入することで、プログラマの方は、今までどおりの感覚で明示的なバージョン管理をしてもらったほうがいいと思っているんですけど、ゲームデザイナだったりアーティストだとバージョン管理の習慣があまりないっていうのもあって、今日お見せしたContentNavigatorはLuminous Studioの一番の表に出てくるツールなんですけれども、それ以外にもSubversionに対するTortoiseSVNと似たものですが、これを使うと自然にこのフォルダ以下はいつの間にかバージョン管理されてとか、たとえばPhotoshopとかMayaでセーブするだけでもうどんどんバージョン管理されていて、仮にローカルのデータが吹っ飛んだとしても元に戻せるよとか、そういう機構追加もしていく予定があります。

――Skeedさんからすると、自社の技術を試せる場ですね。

橋本：一番最初に実験台でいいですよ、みたいな感じで言いましたもんね。

柳澤：ゲームエンジンのすべてを把握しているわけでもなんでもないですけど、非常に大規模で複雑で高度なシステムだと思うので、そういう場に適用する機会を与えていただいたことには大変ありがたいですし、本当にこれって大げさでも何でもなく、今後のインフラを含むテクノロジ全体の牽引役になりうるものなんじゃないかと私は思いますけどね。

Amazonのような小売事業者がクラウドシステムの牽引役みたいな感じになっているわけじゃないですか。けれども小売販売システムより、ゲームエンジンのほうが複雑でなく、大規模じゃないってことはいえないですよね。ゲームエンジンっていうものを出発点として、別にクラウドでもインハウスでも何でもいいです、オンプレミスでも何でもいいですけど、なんかそういうストレージとかネットワークとか仮想化とか、そういうI/Oや計算やなんかにかかわるインフラシステムが今後進化して行ってゲーム業界以外のところでもいろんなレベルで広がっていくっていうことはすごく良いですね。

金子：こうした大きな技術がありますので、もし興味がある方がいらっしゃいましたら、我々に加わっていただけると一緒に何か作れます。柳澤も言うまでもありませんが、うちの他の連中も結構優秀で、一人でももちろん作れるし、

大きなものもみんなでコツコツ作れます。

橋本：やはり優秀なエンジニアさんが集まるのは、金子さんの魅力のおかげですね。

重国：私の予定表に金子さんの名前が入っているだけでも問い合わせが大変で(笑)。「あの金子さんと話すんですか？」と。

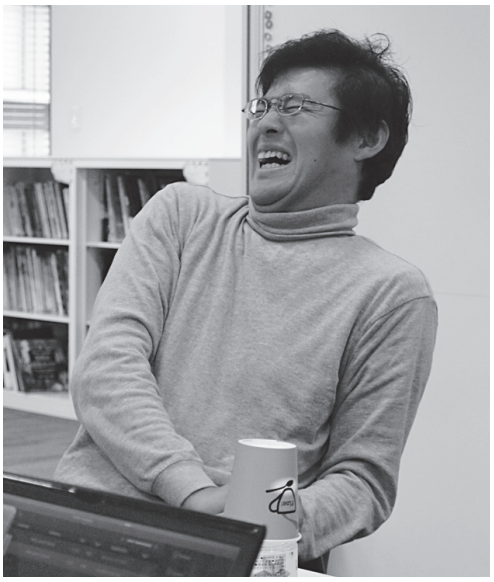
——Luminous Studioの応用例は他に何かありますか？

橋本：あとは普通に建築やインテリアとか、別に衣料・衣服でもいいですし、さらに医療もありえるでしょう。リアルタイムの技術を使う先として、テレビアニメーションもありえます。その場その場で計算するソリューションが求められるものであれば、デジタルコンテンツやデジタルエンターテインメントでなくても、デジタルの何かを作る、映像を中心としてやるところであれば、多分何でも可能性があると考えています。でも、本分はゲームです。そこに全力で向かっています。

エンジニアの理想像とは？

——こうしたLuminous Studioを作り上げて来たわけですが、いっしょに働くとしたら、どのようなエンジニアが理想でしょうか。

橋本：第一に、ロマンに共鳴できる人ですね。ゲーム業界は、とくにSoftware DesignやWEB+DB PRESSの読者にとって、遠い世界に思われているかもしれません。けれど、Skeedさんと弊社の仕事を見てわかっていただけると思いますが、皆さんのスキルを活用できる場がたくさんあります。



重国：そうですね。こうした技術雑誌を読んでいろいろ勉強されている方ならば、すぐにでも活躍できる場が多々あると私も思います。ゲームはグラフィックがあったり、アニメーションがあったり、AI(人工知能)があったり、ネットワークがあったり！

柳澤：言うなれば、総合格闘技ですよね！

金子：めちゃくちゃスゴイことやっているんですよ。ハードウェアも進化していますから。あ、でも、もちろんハードウェアだけがすごいんじゃなくて、作っている人がすごいんです。そう思っているんですけどね！ SD

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7回

Intel VT-xを用いたハイパーバイザの実装その2「/usr/sbin/bhyveによる仮想CPUの実行処理」

浅田 拓也（ASADA Takuya）Twitter @syuu1228

はじめに

前回は、BHyVeの概要や使い方について紹介してきました。いよいよソースコードの解説に入っていきます。今回は、とくに/usr/sbin/bhyveの初期化とVMインスタンスの実行機能の実装について解説をしていきます。

解説対象のバージョン

BHyVeは、現在開発の初期段階です。日々開発が進められており、さまざまな機能が追加されていますが、リリースバージョンが存在していません。

そこで、本連載では執筆時点での最新リビジョンであるr245673を用いて解説を行います。r245673のインストールディスクは、次のアドレスからダウンロードできます。

```
ftp://ftp.freebsd.org/pub/FreeBSD/snapshots/amd64/amd64/ISO-IMAGES/10.0/FreeBSD-10.0-CURRENT-amd64-20130119-r245673-release.iso
```

r245673のソースコードは次のコマンドで取得できます。

```
svn co -r245673 svn://svn.freebsd.org/base/head src
```

/usr/sbin/bhyve と /usr/sbin/bhyveload の役割分担

まずはじめに、前回簡単に紹介した/usr/sbin/bhyveと/usr/sbin/bhyveloadの役割分担について解説します。ゲストOSを起動するには、/usr/sbin/bhyveを実行する前に/usr/sbin/bhyveloadを実行してゲストカーネルのロードを行います。/usr/sbin/bhyveloadを実行すると、/dev/vmm/へVMインスタンスのデバイスファイルが作成されます。このデバイスファイルを通じてゲストメモリ領域にゲストカーネルがロードされ、ゲストマシンのレジスタ初期値やGDT・IDTなどのデスクリプタの設定が行われます。

これに対して/usr/sbin/bhyveの仕事は、初期化済みのVMインスタンスのデバイスファイルをオープンし、VMインスタンスの実行を開始することになります。また、ゲストOSが使用する各種デバイスをエミュレーションするのも/usr/sbin/bhyveの役割となります。

なお、CPUのモードをVMX non root modeに切り替えるなどのハードウェアに近い処理は、特権モードで実行する必要があるためカーネルモジュール（vmm.ko）の仕事になります。

/usr/sbin/bhyveが起動されたら、まずはじめにゲストマシンが使用する各種デバイス（HDD／NIC／コンソール）のエミュレータを使用可能な状態に初期化する必要があります。

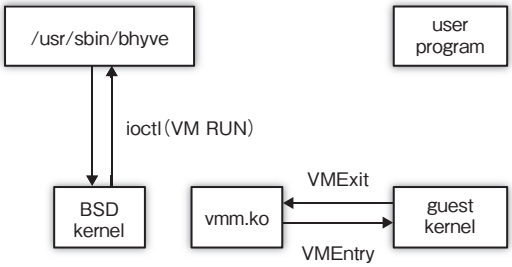
初期化が終わると、/usr/sbin/bhyveは仮想CPUの

数だけスレッドを起動し、/dev/vmm/${name}に対してVM_RUN ioctlを発行します(**図1**)。vmm.koはioctlを受けてCPUをVT-x non root modeへ切り替えゲストOSを実行します(VMEntry)。

VMX non root modeでハイパーバイザの介入が必要な何らかのイベントが発生すると制御がvmm.koへ戻され、イベントがトラップされます(VMExit)。

イベントの種類が/usr/sbin/bhyveでハンドルされる必要のあるものだった場合、ioctlはリターンされ、制御が/usr/sbin/bhyveへ移ります。イベントの種類が/usr/sbin/bhyveでハンドルされる必要のないものだった場合、ioctlはリターンされないままゲストCPUの実行が再開されます。

▼図1 VM_RUN ioctlによる仮想CPUの実行イメージ



それでは、実際にBHyVeのソースコードを読んでいきましょう。**リスト1**と**リスト2**にソースコードを示します。

▼リスト1 /usr/sbin/bhyve/bhyverun.c(丸数字は読解ポイント順を示す)

```
static void *
fbsdrun_start_thread(void *param)
{
  char tname[MAXCOMLEN + 1];
  struct mt_vmm_info *mtp;
  int vcpu;

  mtp = param;
  vcpu = mtp->mt_vcpu;

  snprintf(tname, sizeof(tname), "%s vcpu %d", vmname, vcpu);
  pthread_set_name_np(mtp->mt_thr, tname);

  vm_loop(mtp->mt_ctx, vcpu, vmexit[vcpu].rip);   ← ⑰vm_loop()で仮想CPUを実行する

  /* not reached */
  exit(1);
  return (NULL);
}

void
fbsdrun_addcpu(struct vmctx *ctx, int vcpu, uint64_t rip)
{
  int error;

  if (cpumask & (1 << vcpu)) {
    fprintf(stderr, "addcpu: attempting to add existing cpu %d¥n",
        vcpu);
    exit(1);
  }

  cpumask |= 1 << vcpu;
  foundcpus++;

  /*
   * Set up the vmexit struct to allow execution to start
   * at the given RIP
   */
  vmexit[vcpu].rip = rip;
  vmexit[vcpu].inst_length = 0;

  if (vcpu == BSP || !guest_vcpu_mux){
```

```
    mt_vmm_info[vcpu].mt_ctx = ctx;
    mt_vmm_info[vcpu].mt_vcpu = vcpu;

    error = pthread_create(&mt_vmm_info[vcpu].mt_thr, NULL,
        fbsdrun_start_thread, &mt_vmm_info[vcpu]);
    assert(error == 0);
  }
}
…… (省略) ……
static vmexit_handler_t handler[VM_EXITCODE_MAX] = {
  [VM_EXITCODE_INOUT]  = vmexit_inout,
  [VM_EXITCODE_VMX]    = vmexit_vmx,
  [VM_EXITCODE_BOGUS]  = vmexit_bogus,
  [VM_EXITCODE_RDMSR]  = vmexit_rdmsr,
  [VM_EXITCODE_WRMSR]  = vmexit_wrmsr,
  [VM_EXITCODE_MTRAP]  = vmexit_mtrap,
  [VM_EXITCODE_PAGING] = vmexit_paging,
  [VM_EXITCODE_SPINUP_AP] = vmexit_spinup_ap,
};

static void
vm_loop(struct vmctx *ctx, int vcpu, uint64_t rip)
{
  int error, rc, prevcpu;

  if (guest_vcpu_mux)
    setup_timeslice();

  if (pincpu >= 0) {
    error = vm_set_pinning(ctx, vcpu, pincpu + vcpu);
    assert(error == 0);
  }

  while (1) {
    error = vm_run(ctx, vcpu, rip, &vmexit[vcpu]);
    if (error != 0) {
      /*
       * It is possible that 'vmmctl' or some other process
       * has transitioned the vcpu to CANNOT_RUN state right
       * before we tried to transition it to RUNNING.
       *
       * This is expected to be temporary so just retry.
       */
      if (errno == EBUSY)
        continue;
      else
        break;
    }

    prevcpu = vcpu;
                rc = (*handler[vmexit[vcpu].exitcode])(ctx, &vmexit[vcpu],
                                                       &vcpu);
    switch (rc) {
    case VMEXIT_SWITCH:
      assert(guest_vcpu_mux);
      if (vcpu == -1) {
        stats.cpu_switch_rotate++;
        vcpu = fbsdrun_get_next_cpu(prevcpu);
      } else {
        stats.cpu_switch_direct++;
      }
      /* fall through */
    case VMEXIT_CONTINUE:
    rip = vmexit[vcpu].rip + vmexit[vcpu].inst_length;
```

⑯pthread_create(fbsdrun_start_thread)する。ここまでがハイパーバイザの初期化の処理。ここからはゲストマシンの実行処理に移っていく

㉒/usr/sbin/bhyveに定義されているhandler[]です。I/Oポートへのアクセス、MSRレジスタの読み書き、メモリマップドI/O、セカンダリCPUの起動シグナルなどがハンドラとして定義されている

⑱whileループの中でvm_run()を呼ぶ

㉑ioctlリターンの理由(vmexit.exitcode)に対応したイベントハンドラ(handler[])を呼び出す

㉓handler[]からの返り値により、CPUを現在のripから再開するか／次の命令から再開するか／ハイパーバイザの実行を中止するか、などの処理を行っている。実行が再開される場合は、whileループによりふたたびvm_loop()の実行に戻る

```
      break;
    case VMEXIT_RESTART:
    rip = vmexit[vcpu].rip;
      break;
    case VMEXIT_RESET:
      exit(0);
    default:
      exit(1);
    }
  }
  fprintf(stderr, "vm_run error %d, errno %d¥n", error, errno);
}
……(省略)……
int
main(int argc, char *argv[])
{
……(省略)……

  while ((c =
            (argc, argv, "abehABHIPxp:g:c:z:s:S:n:m:M:")) != -1) {
……(省略)……
  vmname = argv[0];

  ctx = vm_open(vmname);
  if (ctx == NULL) {
    perror("vm_open");
    exit(1);
  }
……(省略)……
  if (lomem_sz != 0) {
    lomem_addr = vm_map_memory(ctx, 0, lomem_sz);
    if (lomem_addr == (char *) MAP_FAILED) {
      lomem_sz = 0;
    } else if (himem_sz != 0) {
      himem_addr = vm_map_memory(ctx, 4*GB, himem_sz);
      if (himem_addr == (char *) MAP_FAILED) {
        lomem_sz = 0;
        himem_sz = 0;
      }
    }
  }

  init_inout();
  init_pci(ctx);
  if (ioapic)
    ioapic_init(0);
……(省略)……
  error = vm_get_register(ctx, BSP, VM_REG_GUEST_RIP, &rip);
  assert(error == 0);
……(省略)……
  /*
   * build the guest tables, MP etc.
   */
  mptable_build(ctx, guest_ncpus, ioapic);

  if (acpi) {
    error = acpi_build(ctx, guest_ncpus, ioapic);
    assert(error == 0);
  }

  /*
   * Add CPU 0
   */
  fbsdrun_addcpu(ctx, BSP, rip);
```

①getoptで処理されなかった一番端の引数がVM名となる。libvmmapiを用いてVM名に対応するデバイスファイルをオープンする

⑤4GB未満のゲストメモリ空間をlomem_addrへマッピングする

⑦4GB以上のゲストメモリ空間をhimem_addrへマッピングする

⑧IOポートエミュレーションの初期化を行う

⑨PCIデバイスエミュレーションの初期化を行う

⑩IO APICエミュレーションの初期化を行う

⑪/usr/sbin/bhyveloadで設定されたRIPレジスタの値を取得する

⑬MPテーブルを生成する。2つ以上のCPU数で起動する時に必要

⑭ACPIテーブルを生成する。無変更なFreeBSDカーネルを起動するのに必要

⑮cpu0（BSP）のスレッドを起動する。実行開始アドレスとしてRIPを渡す

▼リスト2　lib/libvmmapi/vmmapi.c（丸数字は読解ポイント順を示す）

```
…… (省略) ……
static int
vm_device_open(const char *name)
{
        int fd, len;
        char *vmfile;

  len = strlen("/dev/vmm/") + strlen(name) + 1;
  vmfile = malloc(len);
  assert(vmfile != NULL);
  snprintf(vmfile, len, "/dev/vmm/%s", name);

        /* Open the device file */
        fd = open(vmfile, O_RDWR, 0);

  free(vmfile);
        return (fd);
}
…… (省略) ……
struct vmctx *
vm_open(const char *name)
{
  struct vmctx *vm;

  vm = malloc(sizeof(struct vmctx) + strlen(name) + 1);

  assert(vm != NULL);

  vm->fd = -1;
  vm->name = (char *)(vm + 1);
  strcpy(vm->name, name);

  if ((vm->fd = vm_device_open(vm->name)) < 0)
    goto err;

  return (vm);
err:
  vm_destroy(vm);
  return (NULL);
}
…… (省略) ……
char *
vm_map_memory(struct vmctx *ctx, vm_paddr_t gpa, size_t len)
{

  /* Map 'len' bytes of memory at guest physical address 'gpa' */
  return ((char *)mmap(NULL, len, PROT_READ | PROT_WRITE, MAP_SHARED,
         ctx->fd, gpa));
}
…… (省略) ……
int
vm_get_register(struct vmctx *ctx, int vcpu, int reg, uint64_t *ret_val)
{
  int error;
  struct vm_register vmreg;

  bzero(&vmreg, sizeof(vmreg));
  vmreg.cpuid = vcpu;
  vmreg.regnum = reg;

  error = ioctl(ctx->fd, VM_GET_REGISTER, &vmreg);
  *ret_val = vmreg.regval;
```

④vm_device_open()は/dev/vmm/${name}を読み書き用でオープンする

②vm_open()はstruct vmctxをアロケート

③vmctx->fdにファイルディスクリプタを保存する

⑥/dev/vmm/${name}をmmap()する。vmm.koからゲストメモリ空間のアドレスが渡される

⑫/dev/vmm/${name}へVM_GET_REGISTER ioctlを行う。vmm.koからレジスタの値が渡される

```
  return (error);
}
…… (省略) ……
int
vm_run(struct vmctx *ctx, int vcpu, uint64_t rip, struct vm_exit *vmexit)
{
  int error;
  struct vm_run vmrun;

  bzero(&vmrun, sizeof(vmrun));
  vmrun.cpuid = vcpu;
  vmrun.rip = rip;

  error = ioctl(ctx->fd, VM_RUN, &vmrun);
  bcopy(&vmrun.vm_exit, vmexit, sizeof(struct vm_exit));
  return (error);
}
```

⑲/dev/vmm/${name}へVM_RUN ioctlを行う。vmm.koはこのスレッドが実行されているCPUをVT-x non root modeへ移行し、vmrun.ripで指定されたアドレスから実行を開始する

⑳VT-x non root modeでハイパーバイザの介入が必要な何らかのイベントが発生すると、vmm.koの中でトラップされ、/usr/sbin/bhyveでイベントを処理する必要がある場合はioctlがリターンされる。リターンされた理由をvmm.koからstruct vmexitで受け取る

まとめ

/usr/sbin/bhyveの初期化とVMインスタンスの実行機能の実装について、ソースコードを解説しました。極めてシンプルなループによりVMインスタンスの実行処理が実現されていることを確認していただけたかと思います。

今回はユーザランド側の実装のみ解説しましたが、次回はこれに対応するカーネル側の実装を見ていきたいと思います。SD

テキストデータならお手のもの

開眼シェルスクリプト

㈲ユニバーサル・シェル・プログラミング研究所　http://www.usp-lab.com
上田 隆一　UEDA Ryuichi　Twitter @uecinfo

第16回 シェルで画像処理(2) ——awkのパターンと配列を使う

今回は前回に引き続き、シェルスクリプトで画像処理をして遊んでみましょう。前回はコマンドで扱いやすくするために、カラー画像を1ピクセル1レコードにしてから処理しました。ただ、この方法だけだとできることが限られるので、今回は、awkをフルに使って画像処理をやってみます。配列を操作するので、本連載史上、最も「普通の」プログラミングをやります。そうは言っても、普通ではありませんが……。しかし、この人もこんなことを言っているのでよいということにしましょう[注1]。

人生を楽しむ秘訣は普通にこだわらないこと。普通と言われる人生を送る人間なんて、一人としていやしない。いたらお目にかかりたいものだ。——アルバート・アインシュタイン

実行環境

今回は、まったく個人的なことですが、12年間親しんだThinkPadからMacBook Airに乗り換えたことを記念して、Mac上のbashでコーディングします。なぜ乗り換えたかというと、2013年2月号の特集で「Macにはbashが入っているからターミナルを使ってほしい」と書いたときに、「自分が率先しないといかん」と使命感に駆られたからです。カフェでドヤ顔するためではありません。が、もともとカフェ中毒者ですのでドヤリング[注2]などと言われてもしかたありません。言う側から言われる側になって辛いですが、今月号からしばらくはMacでいきます。

図1に、今回の環境を示します。多くのLinuxディストリビューションと違って、Mac(OS X)のawkはgawkではないので注意が必要ですが、今回の内容では出力の違いはありません。

awkのおさらい

パターン

awkは、本連載で何度もワンライナーとして

```
$ cat file | awk 'パターン{アクション}'
```

というように使ってきました。「パターン」はこ

注1) 完全に言いわけに使っています。

注2) カフェなどでMacBook AirなどのApple製品をドヤ顔で使用すること。

▼図1　実行環境

```
$ uname -a
Darwin uedamac.local 12.2.1 Darwin Kernel Version 12.2.1:
Thu Oct 18 16:32:48 PDT 2012; root:xnu-2050.20.9~2/RELEASE_X86_64 x86_64
$ bash --version
GNU bash, version 3.2.48(1)-release (x86_64-apple-darwin12)
Copyright (C) 2007 Free Software Foundation, Inc.
$ awk --version
awk version 20070501
```

れまで使ってきたとおり、入力されたファイルから条件に合う行を抽出するためのものです。パターンはgrepの機能を担っていると考えてよいでしょう。grepは抽出だけですが、awkは抽出した行に対して「アクション」で演算ができます。

図2の例は、パターンで偶数を抽出し、アクションで10で割るというものです。**jot 10**の出力は、**seq 1 10**の出力と同じです。

パターンとアクションの組は、いくつも書くことができます。**図3**のコードはawkのプログラムで、偶数と奇数を数えるものです。パターンは、「START」、「END」も含めて4個ですね。誌面の関係と一行野郎中毒がたたって1行1パターンにしましたが、Cのように改行・インデントをするほうがawkのスクリプトとしてはまともでしょう。

1つの行が複数のパターンにマッチするときは、**図4**のように、パターンに書いた順に何回も出力されます。この辺の挙動は、単なるif文とは違うので注意が必要です。

関数

関数の書き方はJavaScriptに似ています。**function 名前(変数,...){文;文;...}**というように表記します。**図5**は関数の名前の書き方と使い方の例です。

わざと再帰を使ってややこしくしており、例としては少し不適切かもしれませんが、functionの行が関数になっています。この例のように、関数は使う場所より後ろに書いても大丈夫です。

配列

awkは言語ですのでもちろん配列があります。awkの配列は連想配列として実装されています。

▼図2　パターンとアクションの例

```
$ jot 10 | awk '$1%2==0{print $1/10}'
0.2
0.4
0.6
0.8
1
```

▼図3　パターンを並べたawkのコードの例

```
$ cat oddeven.awk
#!/usr/bin/awk -f

START{even=0;odd=0}
$1%2==0{even++}
$1%2==1{odd++}
END{print "奇数:",odd;print "偶数:",even}
$ jot 9 | ./oddeven.awk
奇数: 5
偶数: 4
```

▼図4　複数のパターンにマッチする場合

```
$ echo 1 | awk '{print $1,"a"}NR==1{print $1,"b"}NR!=2{print $1,"c"}'
1 a
1 b
1 c
```

▼図5　関数の書き方

```
$ cat func.sh
#!/bin/bash

echo $1 |
awk '{print scream($1,10)}
     function scream(a,n){return n==1?a:(scream(a,n-1) a)}'
$ ./func.sh あ
あああああああああ
```

▼図6　配列の使い方

```
$ awk 'BEGIN{a["猫"]="まっしぐら";print a["猫"]}'
まっしぐら
```

ですので、**図6**のような使い方ができます。

もちろん、普通の配列としても使えます。配列として使うときは、**図7**のように、インデックスを0からではなく1から始めます。自身で使うときは0からでも動きますが、関数が配列を返すときは1に最初の要素が入っているので、ほかに理由がなければ合わせましょう。

表記に区別がないので、**図8**のようなこともできます。Cでやったら間違いなく怒られますが、awkでは大丈夫です。

こういうことができるので、たとえば、$1はいらないけど$2や$3を配列に入れたいという場合、それぞれ**f[2], f[3]**に入れてやればよいということになります。

ほかの言語では配列と連想配列は区別されることが多いのですが、awkでは実装上も表記上も区別がありません。気軽に使える一方、連想配列ですので、あまり速度は期待できません。

二次元配列は、インデックスをカンマで区切って表記します。もちろん数字も使えます。**図9**に使用例を示します。

察しのよい人にはわかるかもしれませんが、この配列は実際にはCの二次元配列とはまったく異なるものです。awkではインデックスを全部連結した文字列をキーにして、1つの連想配列に記録しているようです。もちろん、文字列の連結は**12,3**と**1,23**が区別できるように行われます。このへんの仕様は、いかにもLL(Light weight Language)の元祖らしい潔さです。

awk中心のシェルスクリプトで画像処理

では、ここから本題です。今回もJPEGなどの画像をアスキー形式のppm画像に変換し、処理します。ImageMagickのインストールをお願いします。

前号でも説明しましたが、アスキー形式のppm画像は、スペースか改行区切りで数字の並んだテキストファイルです。**図10**に例を示します。最初のP3が画像の形式、次の2つが画像のサイズ、次いで画素値の刻み幅(深さ)です。そのあと、左から右、上から下の画素に向けてr(赤)、g(緑)、b(青)の値が並びます。

パターンを使って画素を配列に記録

まず、画像をawkの配列に記録するまでのコードを**リスト1**に示します。6行目で、画像($1に指定する)をppm画像に直しています。11〜14

▼図7　配列の使い方(その2)

```
$ echo 南海 ホークス | awk '{¥
     a[1]=$1;a[2]=$2;for(i=1;i<=2;i++){print a[i]}}'
南海
ホークス
 split関数で文字列を切って配列aに代入
$ echo 'OH!MY!GOD!' | awk '{split($1,a,"!");print a[2]}'
MY
```

▼図8　インデックスが大きくても大丈夫

```
$ awk 'BEGIN{a[123456789]=10;print a[123456789]}'
10
```

▼図9　二次元配列の使用例

```
$ cat hoge.sh
#!/bin/bash

echo $1 $2    |
awk 'BEGIN{
     a["グー","チョキ"] = "グー";
     a["パー","チョキ"] = "チョキ";
     (略)
     }
     END{print a[$1,$2] "の勝ち"}'
$ ./hoge.sh パー チョキ
チョキの勝ち
```

▼図10　ppm画像をheadした例

```
$ head 1.ppm
P3                    ←画像のタイプ
#*                    ←コメント
960 640               ←画像の幅、高さ
255                   ←深さ
125 94 50 126 95 51 127 96 52 128 97 53 128 97 53...
```

行目でppm画像を読み込み、データを縦1列に並べ、中間ファイルに落としています。17〜19行目でヘッダ部分(幅、高さ、深さ)を変数に落としたあと、22行目以降で画像の本体部分の数字をawkに入力しています。

awkに書いてあるパターンは3つで、上から順にそれぞれr、g、bの値を二次元配列に代入しています。パイプから流れてくる数字は、1行目にr、2行目にg、3行目にb、というように3個ごとに値が並んでいるので、rgbそれぞれをフィルタしたければリスト1のように、NR(行番号)を3で割った余りで判定すればよいことになります。

各フィルタに対応するアクションでは、行番号から画像での横位置、縦位置を求めて配列に値を代入しています。横位置は左側から0、1、2……、縦位置は上側から0、1、2……と数えることとしました。awkの掟に反してゼロから数えていますが、`n%w`と`int(n/w)`に1を足すのは面倒ですのでこのようにしています。

光を発射

あとは、これに自分のやりたい処理を実装するだけです……、と言ってもこれは画像処理の本を読むか、Webで調べるかしないとチンプンカンプンな人もいるかと思います。ここでは2つほど例を見せます。

まず、画像の位置を使った処理の例です。**図11**のサンプル画像はUSP友の会の勇壮なLT写真です。よく見えないかと思いますが、後ろの男[注3]は手にビール瓶を持っています。ビール瓶からフラッシュを出してみましょう。

図12に仕上がり、**リスト2**に、この処理を行うawkの部分を示します。配列に値を読み込む部分まではリスト1と一緒で、新たにENDパターンに対する処理と、関数を1つ追加しています。このシェルスクリプトの名前はflash.shで、**図13**のように使ってJPEG画像を得ました。

注3) 筆者です。

▼リスト1　awkの配列にRGBの値を入れるまで(donothing.sh)

```
01 #!/bin/bash -xv
02
03 tmp=/tmp/$$
04
05 ### 画像の変換
06 convert -compress none "$1" $tmp-i.ppm
07
08 ### データを縦一列に並べる
09
10 #コメント除去
11 sed 's/#.*$//' $tmp-i.ppm    |
12 tr ' ' '¥n'             |
13 #空行を除去
14 awk 'NF==1'    > $tmp-ppm
15
16 ### ヘッダ情報取り出し
17 W=$(head -n 2 $tmp-ppm | tail -n 1)
18 H=$(head -n 3 $tmp-ppm | tail -n 1)
19 D=$(head -n 4 $tmp-ppm | tail -n 1)
20
21 ### 画素の値を配列に
22 tail -n +5 $tmp-ppm    |
23 awk -v w=$W -v h=$H -v d=$D ¥
24     'NR%3==1{n=(NR-1)/3;r[n%w,int(n/w)] = $1}
25     NR%3==2{n=(NR-2)/3;g[n%w,int(n/w)] = $1}
26     NR%3==0{n=(NR-3)/3;b[n%w,int(n/w)] = $1}'
27
28 rm -f $tmp-*
29 exit 0
```

▼図11　加工する画像(1.jpg)

▼図12　ビール瓶の先から光線を出す

リスト2のENDパターンでは、まず8行目でppm画像のヘッダ部分を出力しています。そのあとの二重のfor文で、1画素ずつ、r、g、bの順番に値を加工して出力しています。

forのループ内では、まず11、12行目で、その画素が光を出す中心の画素に対してどの位置にあるかを求めています。中心の画素は、筆者が手で調べてハードコーディングしました。変数にしてもよいですね。

そのあと、13行目で、「その画素が光を出す中心に対してどの方角にあるか」を求めています。atan2はCにもある関数ですが、見たことがない人もいるかもしれません。**図14**のように角度を返す関数です。atan2の返した値はラジアン値なので、πで割って360をかけると、いわゆる普通の角度(degree)になります。

ところで、**(x,y) = (0,0)**だとatan2が何を返すか不安ですが、awkですので、

```
$ awk 'BEGIN{print atan2(0,0)}'
0
```

のように実用的な値を返してくれます[注4]。

注4) 全部のバージョンのawkに当てはまるかは未調査です。

▼リスト2　ビール瓶の先から光を出すためのawk(flash.sh)

```
01 ### ビール瓶の先から国民に光を与える
02 tail -n +5 $tmp-ppm      |
03 awk -v w=$W -v h=$H -v d=$D ¥
04     'NR%3==1{n=(NR-1)/3;r[n%w,int(n/w)] = $1}
05     NR%3==2{n=(NR-2)/3;g[n%w,int(n/w)] = $1}
06     NR%3==0{n=(NR-3)/3;b[n%w,int(n/w)] = $1}
07     END{
08         print "P3",w,h,d;
09         for(y=0;y<h;y++){
10             for(x=0;x<w;x++){
11                 ex = x - w*0.87;
12                 ey = y - h*0.32;
13                 deg = atan2(ey,ex)*360/3.141592 + 360;
14                 weight = (int(deg/15)%2) ? 1 : 4;
15
16                 p(r[x,y]*weight);
17                 p(g[x,y]*weight);
18                 p(b[x,y]);
19             }
20         }
21     }
22     function p(n){ print (n>d)?d:n }'
```

リスト2の14行目では、角度15度刻みでweightという変数の値を1にしたり4にしたりしています。完成した画像をよく見ると15度刻みで光っていますが、この準備です。細かい話ですが、atan2が返す値がプラスの場合とマイナスの場合がある影響で360度きれいに15度刻みにならないので、13行目で360を足して、degの値がプラスになるようにしています。

これでいよいよ標準出力に値を出していきます(16～18行目)。白黒でわかりにくいですが、金色(黄色)に光らせたいので、rとgの値にweightをかけて強調します。pという関数は22行目で実装しており、値が最大値dを超えるとdで打ち切って出力するというものです。ところで、awkの変数は基本的にすべてグローバル変数ですので、オプションで定義されたdは、関数の中でも使えます。長いプログラミングをするとちょっと辛いかなと、個人的には思います。

エンボス加工する

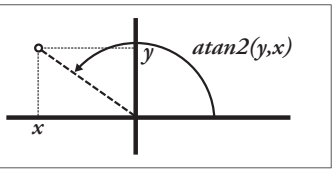
もう1つ例を**リスト3**に示します。これはエンボス加工ふうに画像を変換する処理です。**図15**に処理前、**図16**に処理後の画像を示します。このようなアイコンの処理だけでなく、写真を処理すると絵画のような風合いになります。**http://www.usptomo.com/PAGE=20130113IMAGE**で公開していますので、遊んでみてください。

リスト3の9行目の処理では、変数aに、ある画素とその周囲の画素

▼図13　画像を加工するシェル操作

```
$ ./flash.sh 1.jpg > flash.ppm
$ convert flash.ppm flash.jpg
```

▼図14　atan2(y,x)の返す角度



のg値を比較した値を代入しています。この処理は「sobelフィルタ」と言われるもので、この演算だと、画像の斜め方向で緑色が急激に変わっている画素のaの値が正、あるいは負の方向に大きくなります。**図17**に、aの値でグレースケール画像を作ったものを示します。本当はgだけでなく、r、g、bの値で平均値をとってaの値を求めるべきですが、コードがややこしくなるので緑だけにしています。

このaの値を、10行目のように各rgb値から引くと、色の変化の急激なところが強調されて、人間の目には画像に凹凸があるように見えます。

終わりに

今回は、シェルスクリプト(ただしawk多め)で画像処理をしてみました。筆者は遊びのつもりで始めましたが、テキストにすると処理の流れがわかりやすいので、これは画像処理の教育用によいかもしれません。

今回はawkの説明を充実させました。パターンや配列、関数の書き方などを説明しました。特徴的なのはパターンの存在そのものと、あとは配列の実装でしょう。パターンをたくさん並べてプログラミングをすると、「1行ずつ読み込み、パターンで振り分けて何かする」という、ほかの言語との違いが際立ちます。この動作はシェルスクリプトで使うほかのコマンドと似ており、やはり相性という点でawkとシェルスクリプトには切っても切れない縁があります。逆に言えば、awkが使いこなせることが、シェルスクリプトでなんでもやろうという発想にもつながります。

次回は作りものをいったんお休みして、「コマンドでどうしてもできないややこしい処理」を1行awkで処理する方法を扱いたいと思います。SD

▼図15　エンボス加工前の画像

▼図16　エンボス加工後の画像

▼図17　aの値で画像を作ったもの

▼リスト3　エンボス加工処理

```
01 tail -n +5 $tmp-ppm     |
02 awk -v w=$W -v h=$H -v d=$D ¥
03         'NR%3==1{n=(NR-1)/3;r[n%w,int(n/w)] = $1}
04         NR%3==2{n=(NR-2)/3;g[n%w,int(n/w)] = $1}
05         NR%3==0{n=(NR-3)/3;b[n%w,int(n/w)] = $1}
06         END{print "P3",w-2,h-2,d;
07             for(y=1;y<h-1;y++){
08                 for(x=1;x<w-1;x++){
09                         a = 2*g[x-1,y-1] + g[x-1,y] + g[x,y-1] - g[x,y+1] - x+1,y] - ➥
                           2*g[x+1,y+1];
10                         p(r[x,y] - a); p(g[x,y] - a); p(b[x,y] - a);
11                 }
12         }}
13         function p(v){print (v < 0) ? 0 : (v > d ? d : v)}'
```

最終回

アプリデザインのための基礎知識

スマートフォンの認知度を一般に広めただけでなく、ソフトウェア開発においても新しい波を作り出してしまったiOS。開発者たちは何を見、どう考えているのか。毎回入れ替わりでiOS向けアプリケーション開発に関わるエンジニアに登場いただき、企画・開発のノウハウやアプリの使いこなし術などを披露してもらいます。

勝間田 雅裕　KATSUMATA Masahiro
+Beans URL http://www.plusbeans.com/

アプリとデザイン。ユーザとデザイン。

あなたはデザインに何を求めますか?

iOSアプリに限らず、現在のモバイルアプリにおけるデザインの重要性については、もう皆さま充分に認識されていることかと思います。では、開発者としてアプリデザインを創り上げる際に最も重要視するポイントは何でしょうか?

また、そのデザインによってどのような結果や効果を期待しますか?

さまざまな開発者の方々が執筆されてきた本連載企画ですが、今回は少しベクトルを変えて、デザイナの立場から上記の質問に対するいくつかの答えと、それを実現させるためのヒントを提案させていただこうと思います。いずれもあくまで筆者の個人的な見解ではありますが、あまり公に語られる機会の少ない内容でもあります。アプリのデザインをご自分で手がけられる、または依頼をされる際の参考としてお役に立てる内容になるのではないかと考えています。

ちなみに筆者は現在iOSアプリを中心としたデザインをしておりますが、2009年にiOS Developer Programに登録し未リリースながらも開発も続けています。そのような経緯からもいわゆる専業デザイナよりは少し開発者に近い目線でも解説していけるかと思いますので、肩の力を抜いてお読みいただければ幸いです。

デザインされていないアプリなど存在しない

本題に入る前にとある一節をご紹介します。

デザインが必要か省けるかという問いは、まったく的外れだ。デザインとは避けられないものであり、良いデザイン以外の選択肢は、悪いデザインしかない。デザインがないことはありえない。誰もが、四六時中、それに気づかずにデザイン決定を行っている。(中略)良いデザインとは単純に、正しいステージにおいて、ニーズを生むような他者との打ち合わせを行い、デザイン決定を意識的に行った結果である。

～Douglas Martin～

引用元：『Book Design』Douglas Martin 著 ／Van Nostrand Reinhold刊
日本語訳：Webロケッツマガジン『WEBデザインとは何か? 迷った人に読んでほしい言葉 80+』より
http://webrocketsmagazine.com/entry/20111220/80-design-quotes-for-inspiration.html

つまり、この言葉を言い換えれば**デザインされていないアプリなど存在しない**ということです。仮に、作り手が予算や納期の関係で「デザインは入れていない」と考えていたとしても、それは単純に「そのアプリに最適化されていない形に意図を持たないままデザインされた」状態です。ボタンをどこにどの大きさで置くのか? もっと言えば、ボタンに乗せるテキストの文言

を決めることさえデザインの一部だとも言えますが、そういった1つ1つの選択を「デザインしている」という意識のないまま決定してしまった結果のデザインにほかならないのです。

その点をあらためて心にとめてもらえれば、本当の意味での意図的なデザインの必要性をより実感していただけるのではないでしょうか。デザインが不可避なものであるならば良いデザインを目指したい！　そう考えるのがクリエイターの自然な思考だと筆者は思います。

さらにユーザの視点でも見てみましょう。

ユーザがアプリをダウンロード／使用する際に重視するポイントについての調査報告などがあれば良いのですが、残念ながら見つけられませんでしたので、大まかに推測してみることにします。

2012年後半にインターネット調査のネオマーケティングが実施した調査結果[注1]によると、日本のユーザがスマートフォンを購入する際に重視するポイントのトップは“デザイン”（66.2%）。そして次点が“価格”（61.6%）となったそうです。ハードとアプリでは若干の差はあるかとは思いますが、ユーザのニーズを予想する手掛かりにはなるでしょう。

まずわかりやすくするために、ユーザがアプリを導入する際にデザインをどの程度重視するのかをタイプ別に分けてみましょう。

A. デザインにこだわり、それに見合わないアプリは使わない**こだわりタイプ**
B. まず機能重視で決め、デザインが良ければなお良いと考える**柔軟タイプ**
C. とくにこだわりはなく、デザインの良し悪しは気にしない**無頓着タイプ**
D. デザインにこだわったものには抵抗感があり、良いデザインと言われるようなアプリは使いたくない**天の邪鬼タイプ**

注1）株式会社ネオマーケティング「スマートフォンに関する調査を発表」（2012.10.11）URL http://www.i-research.jp/report/report/r_20121011.pdf

これらのタイプの中で、前述の調査結果でデザインを重視すると答えた6割強のユーザがAとBのどちらかに当てはまると仮定すると、残りの4割弱はCとDに分かれることになりますね。

また、Dは少し強引な気もしますが存在しないとは言い切れないために入れたタイプですので、多く見積もっても1割程度と見て良いのではないかと思います。そこで次の**図1**を見てください。この図はあくまでも予想の域を出ないグラフではありますが、筆者の実感からしてもさほど遠からず当たっているのではないかと感じています。

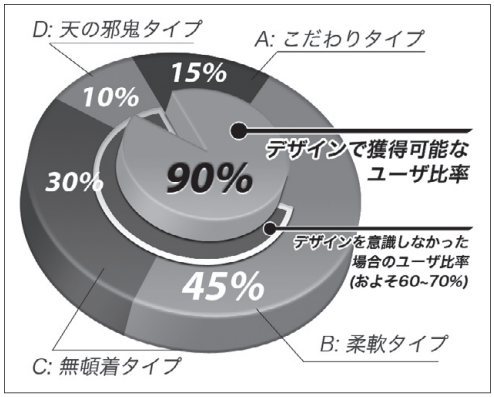
つまり、意図してデザインをした場合とそうでない場合では、獲得可能なユーザ数に明らかな差が出るだろうと予測できます。今一度このことを再認識していただき、次の内容に進むことにしましょう。

アイコンデザインとインターフェースデザイン

さて、ここまで半ばダメオシするような形でアプリにおけるデザインの必要性を強調したところで、はじめの問いかけに戻りましょう。**アプリのデザインに何を求めるか？**です。

見た目を美しくしたい。印象を良くしたい。ダウンロード数を上げたい……いろいろな考えがあるでしょう。ですが一口にアプリのデザインと言っても、アイコンのデザインとインター

▼図1　ユーザタイプ別推定グラフ



フェースのデザインではその目的と役割は違うもなのではないでしょうか？　では具体的にそれぞれどのような違いがあり、デザインを決定する際にどんな視点から判断するのが良いのかを見ていきましょう。

理想的なアイコンとは？

筆者は仕事柄、「アプリのデザインとはどうあるべきか？」という自問自答を常々繰り返しています。それはプログラマの方々がよりスマートな実装方法やアルゴリズムについて常に考えているのに似ているかもしれませんね。そんな中で筆者が考える理想的なアプリアイコンとはどのようなものか？　その現時点での答えを次に示します。

❶ターゲットユーザにとって気になるデザインであること
❷ホーム画面に置きたくなるデザインであること
❸それがアプリにとっての象徴(アイコン)として記憶に残り、アプリを起動するトリガーの役割を果たせるものであること
❹毎日使っても飽きのこないデザインであること

もちろんアプリ内容や販促方法などによって違いはありますが、基本的にはこの4つが重要だと考えています。

■“わかりやすさ”の弊害

まず❶についてですが、これは簡単言えば「広告効果」です。たとえば、まだ有名とは言えない開発者やプロモーションに予算を割けない場合などではとくに重要なのではないでしょうか。

ツール系アプリなどではよく「一目で機能のわかるデザインにしよう」などと言われることもありますが、筆者の意見はこれとは違います。もちろん、シニア向けのアプリやターゲット層が限られているニッチなアプリであればわかりやすさを第一に考えることがベターな場合もあります。ですがアイコンを小さな広告として捉えた場合、一目見ただけでアプリの中身がほとんど予想ができてしまうようなデザインでは効果は薄いと考えたほうが良いでしょう。

良い広告は見る人の心を動かし、行動を変えます。そのために、まずはじめに必要なことは**注目してもらえる／気にしてもらえる**ようにすることです。では、実際にどういうものが気になるのか？　**図2**は筆者がデザインしたアイコンの一例です。これらはいずれも“気になる”という視点を重視して作成してあります。

「novelnove[注2]」はソーシャルで参加する形のショートショート執筆アプリです。「iconsider[注3]」はDropboxを利用することで作成中のアイコンの実機確認が簡単にできるアプリで、リリースされた当時、海外の有名なiOSアプリアイコンの紹介サイトなどに掲載されたこともあります。「まとめ画像一括ダウンロード －まとめDL－[注4]」はその名のとおりサイト上の画像を一括ダウンロードできるアプリです。

たとえばこのまとめDLのアイコンを「一目で機能がわかること」を重視したデザインにするのなら、3つくらいのシンボリックな写真アイコンと円に下矢印のダウンロードマークを合わせたようにするだけで充分でしょう。ですが、それでは足りないだろうと感じています。

図2のいずれのアイコンも、それを見ただけで具体的な機能がすぐにわかるようにはデザインしていません。なんとなくどんなジャンルのアプリであるかは想像できる、といった感じでしょうか。その代わりにこれらのアイコンには、とくに印象付けとタップしたくなる仕掛けを盛り込んでいます。

たとえばそのアイコンを見た人が、ぱっと見でアプリの内容を理解した気になってしまったとします。その場合、その人が頭に浮かべたものが欲しいものでなければアプリの詳細を見る

注2）URL http://novelnove.com/
注3）URL https://sites.google.com/site/iconsiderapp/ja
注4）URL https://itunes.apple.com/jp/app/id583618950

ことはないでしょう。ですが、詳細を見てくれさえすれば、便利そうだと思ってもらえるかもしれないとは思いませんか？　もしかしたら、そのアイコンはユーザを掴むチャンスを逃してしまっているのかもしれないのです。

それを逆手に取ってワザと**わからないことを認識させてあげる**ことは気にさせるためのひとつの秘訣だと思います。わからせないことによって詳細を知るためにタップするという行動を促すのです。

ストアに並ぶアプリの数は今もなお、常に増え続けています。そんな星の数ほどあるアプリの中で、ユーザがあなたのアプリに出会う機会はとても限られています。App Storeやブログやメディアなどでアイコンとアプリ名だけで紹介されたときにそのチャンスをいかに活かすことができるか。それにはアプリ名ももちろんですが、アイコンもとても重要なのです。そこでそのわずかな機会を逃さぬために“気になるデザインにしよう！”というのが筆者からの提案です。

■“主張すること”と“主張しすぎないこと”

次に❷の「ホーム画面に置きたくなる」については、ユーザ視点で考えれば当然のことですね。たまたま友人にホーム画面を見せたときに「そのアイコン、どんなアプリなの？」なんて言われたら少し自慢げに紹介したくなるのではないでしょうか。また、比較的デザインにこだわりのあるユーザからは「アイコンがダサいから入れたくない」といった声を聞くことも少なくありません。ですので変に悪目立ちせず、それでいて特色のあるアイコンが望ましいでしょう。

❸の「象徴(アイコン)として記憶に残り、アプリを起動するトリガーの役割を果たせるもの」については、少し説明が必要かもしれません。これは意外に見落とされがちなポイントでもあるのですが、皆さんもユーザとしてiPhoneやiPadを使っているときに「たしかこんなアプリを入れてたと思うんだけど、どれだっけ？」とアイコンを探した憶えがあるのではないでしょうか？　日々頻繁に利用されるようなアプリであれば良いのですが、そうでないアプリの場合、ユーザにとってアプリの印象とアイコンデザインとの関連付けができていなければ、手元にあるのに見つけてもらえない＝使ってもらえない、という可能性が高まります。つまりそのアイコンは、トリガーとしての役割を果たせていないということになるのです。

ただし、実際には予算の都合などでアイコンのみのデザインをアウトソースする場合も少なくはないと思います。そんな場合にアイコンデザインだけでアプリの中身との関連付けを確実にすることは困難ですが、アイコンの印象をより強いものにすることによって、トリガーとしての役割を果たせるようにすることは可能だと考えています(もし配色や雰囲気をアイコンとUIとで関連づけることができるのであれば、それがより理想的だろうと思います)。

❹の「飽きのこないデザイン」については普遍

▼図2　“気になること”に重点を置いてデザインしたアイコン

的に求められるデザインの形と言って良いでしょう。前述までのように主張することは大切ですが、それと同じくらい**主張しすぎないこと**も大切です。筆者はよく“品があるかどうか？”ということを判断基準にしています。一過性のウケ狙いアプリでない限り、そういった自分なりの判断基準を元に考えていくのが良いでしょう。判断を下す際の指針を常に頭に置いておくことはロングテールなビジネスを目指すうえでは、とても大切なことだと思います。

理想的なインターフェースデザインとは？

では次に、理想的なインターフェースデザインについて考えていきましょう。アイコン同様、インターフェースデザインにも求められる要素がいくつかありますが、シンプルに考えると次の4つに分けられるのではないでしょうか。

①価値ある見た目を提供する
②アプリ独自の世界観やイメージを形づくる
③ユーザをナビゲートする
④情報を整理してアプリの使い勝手を向上させる

これらすべてが実現できていれば、それは良いデザインのアプリだということができるでしょう。それと同時に注目したいのがそれぞれの重要度、プライオリティです。実は前述の4つはプライオリティの低い順に並べています。つまり④が最も重要な項目になります。では、それぞれの内容を見ていきましょう。

まず①の「価値ある見た目づくり」についてですが、これはいわゆる“狭義のデザイン”と言って良いでしょう。ユーザがデザインについて語る場合のほとんどがこれに当たるのではないかと思います。もっと簡単に言ってしまえば「見た目の問題」ですね。

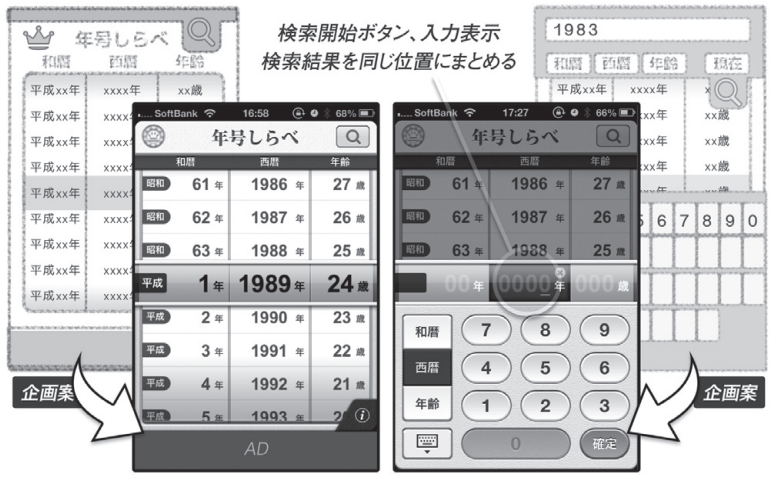
もちろんユーザの心を惹き付けるには見た目も大切です。ですが、ここで言う見た目の良さやその価値は個人的な感覚や趣向、またトレンドなどによって大きく左右される要素であることもまた事実です。それに反して前述の項目の中でも目に見える形としては最もわかりやすいものですので、ともすれば気付かないうちに最重要項目になってしまう恐れもあります。**見た目は大切。しかし最重要ではない**ということを他の項目の解説とともに理解していただければ幸いです。

次に示したのが②の「独自の世界観／イメージづくり」です。ゲームなどエンターテイメント性の高いアプリではとくに重要になる内容ですね。またツール系のアプリなどであっても、作り手が考える「こう使ってほしい」「楽しく心地よく」といったイメージをビジュアル化することによって、ユーザに伝えることが可能になります。そのアプリにとって最適な雰囲気をデザインに落とし込むことによって、ユーザにとってより使い続けたくなるものになるでしょう。

それを実現するにはデザイナと企画／開発者とのイメージの共有が何よりも大切です。もともと言葉では伝えきれないようなイメージを具現化する作業ですので、見た目の格好良さや可愛さとは別のベクトルで“このデザインでユーザに伝わるか？”ということを判断基準にすると良いでしょう。

3つめは「ユーザをナビゲートする」ですね。この項目は先の2つよりもさらに重要です。別の言葉に言い換えれば「おもてなし」と呼ばれる内容になります。パソコンでの業務などで使うプロ仕様のソフトウェアとは違い、iOSアプリの大半は一般向けに作られます。そんな一般ユーザ向けのアプリでありながら「何でもできますよ！　どうぞ自由に使ってください」という形にしてしまうのは不親切なデザインだろうと筆者は考えています。そのようなデザインは「何をすれば良いのかわからない」状態にユーザを陥れてしまう落とし穴だとも言えます。それを防ぐために各タイミングでの選択肢を限定したり、ユーザの動向を予測して自然な視線誘導や指の動きをナビゲートできるようなデザインを考えていきます。

▼図3　情報を整理し、ユーザをナビゲートする一例



また「デザインによって問題を解決する」とは良く語られますが、ユーザをナビゲートできるように熟考していくことによって、企画の段階では気付けなかった問題点が見えてくることも多くあります。それらを1つ1つ探りながら解決策を模索していくこともデザインの大切な役割の1つです。もちろん、それらを実現するにはデザインだけではなくプログラムの力も必要になりますが、そのコストに値するユーザビリティを実現できるという点で重要度の高いテーマであることは間違いないでしょう。

最後の4つめ「情報を整理し、アプリの使い勝手を向上させる」です。これは重要度が最も高いというよりも絶対必要条件と言っても良いかもしれません。日本でデザイナというと、なんだかスタイリッシュでカッコ良い見た目を創造できる人、のようなイメージが強いかと思いますが、Designとは設計であり、計画です。また“目的をより良い形で実現する”ことでもあります。その点からも、**デザインは「情報を整理する」ところからはじまる**、と言っても大げさではないでしょう(**図3**)。

ゲストを我が家に招き入れて楽しいひとときを過ごしてもらうためにすべきことは、ゲスト好みのスタイリッシュな家具をそろえる①ことでも壁紙で自己主張する②ことでもなく、まず、散らかった部屋をキレイに整頓する④ことではないでしょうか？　それが済んでからゲストをどうエスコートして楽しんでもらうか③を考えます。丸数字は前述の項目に当たります。こうたとえるとそのプライオリティの違いがよりわかりやすく感じられるのではないでしょうか？

実際にデザインを選定し決めていく際には、これらの重要度を常に意識してデザインの力を充分に活かせているかどうかを判断基準にすることをお勧めします。またインターフェースデザインですので常に使うシーンをイメージして、そのデザインが動いた状態でどうなるか？をシミュレートしつつ判断していくことも大切です。

進化するためのデザインと変化を嫌うユーザ

ここまでの内容で筆者なりのアプリデザインの指針を示させていただきました。話を先に進めるために、先に述べた判断基準を念頭にデザインを取り入れ開発されたアプリがApp Storeに並んだとしましょう。ですが言うまでもなくアプリのリリースはゴールではなくスタートで

す。そこからユーザを獲得しバージョンアップを重ねて、より多くのユーザに必要とされるアプリを目指していくことになりますね。

現に多くのヒットアプリも時を重ねるごとにさらなる進化を目指して、機能の追加やブラッシュアップにとどまらずデザイン面でも初期バージョンから多くの変更がなされています。そこで、リリース後の重要なポイントとしてのデザインの変更についてお話ししたいと思います。

アプリのデザインを変えるリスク

まず基本として**リリース後に大幅なデザイン変更はしないことが原則**だと考えておいたほうが良いでしょう。初期バージョン時にiOS標準のUIのみでリリースしていた場合などは別ですが、ある程度アプリ独自のデザインだった場合には、その急激な変更にはかなりのリスクが伴います。これはアイコンも同じです。

ご存じの方も多いかもしれませんが、既存デザインの変更はユーザ数が多ければ多いほどユーザを混乱させてしまったり、反感を買ってしまう可能性が高まります。これは変更後のデザインの良し悪しとはさほど関係がなく、もしあなたが気に入っていなかったデザインを素晴らしいと思えるデザインに変えたとしても同様です。ユーザは使い慣れた状態を良しとしていることが多く、極端なほどに変化を嫌うものだと考えておいたほうが良いでしょう。そこを軽視してデザインを変えると、アプリのレビューやTwitterなどで「前のほうが良かった」「改悪」「元に戻してほしい」といったネガティブワードを多く目にすることになってしまいますのでご注意を。

ですので初期バージョンのデザイン決定時には、そういったことも充分に踏まえたうえで判断を下すのが得策です。せっかく貴重な時間を裂き予算をかけてデザインするのであれば、リリース時だけでなく継続的にユーザに使い続けてもらえるような戦略を練りましょう。

リスクを抑えてアプリの価値を高めるために

ではもし、初期バージョンとはデザインを変えたくなった場合はどうするべきでしょうか？もちろん変えないことが絶対の正義ではありません。多くのヒットアプリのようにユーザ数の増加や流行、または刻々と変化していくニーズなどに合わせてデザインをより良くしていく必要もあるはずです。そんな場合には、大まかに分けて次の2つの方法が有効です。

A. 別のアプリとしてリリースする
B. 少しずつバージョンアップしていく

Aは少し極端な例ではありますが、全体のイメージが変わるような大幅な変更であれば別のアプリにしてしまうのもひとつの手です。ですが別の問題としてその時点でのユーザ数を新しいアプリでも取り込めるかどうかは、当然ですが未知数になります。

以前筆者は「AppBank for iPhone」というアプリのメジャーアップデート版のトータルデザインをさせていただいたのですが、それは「楽しいAppBank.net[注5]」という別アプリとしてのリリースでした。こういった手法では確かにユーザの完全移行は難しいと思います。それでもユーザを半ば強制的に移行させるアップデートによる変更ではなく、別アプリとしてリリースすることで、結果的にはユーザに能動的に移行してもらうことになるのです。つまり、ユーザ自らのアクションで新しいアプリとして認識してもらう形になるため「アップデートしたら急に違うアプリのようになっていた」という感覚は抑えられます。

もちろん、同じコンテンツでありながら別々のバージョンのアプリを延々とサポートしていくことは現実ではありませんので、日を置いて旧バージョンのサポートを終了することにはなります。ですが、その少しの猶予期間をユーザ

注5）http://apps.appbank.net/AppBank_for_iPhone.html

に提供することで比較的ユーザに優しい移行になるのではないでしょうか？

次にBの「少しずつバージョンアップしていく」ですが、これがユーザにとっては理想的な進化の形だろうと思います。ただし、毎回部分部分で変更を加えていくと最終的にちぐはぐなデザインになってしまう可能性もありますので、まず最終的なゴールとなるデザインに当たりをつけてから、そこに向かって期間と回数を重ねて近づけていくような戦略をとることができればなお良いでしょう。

これを実際にやるとなると地道な作業にはなりますが、変化を嫌うユーザにも許容できる範囲での改善を続けていくことで、ユーザを教育していくことも可能になってきます。“教育”というと少し大げさに聞こえるかとは思いますが、「このアプリは少しずつ進化していくアプリだ」とユーザに思わせることができれば、デザイン面でのアップデートもより受け入れられやすくなるはずです。

こういったことを上手にクリアして将来のアップデートへの期待感をつくることができれば、その期待感がアプリの価値に加わります。噛み砕いて言えば、現状のアプリの価値が仮に70%だとしても、そこに期待感（10%としておきましょう）が加われば80%の価値になります。つまり、ユーザから見たそのアプリの価値は本来の価値以上になるため、より愛されるアプリになる力を秘めていることになるでしょう。

以上のようにアプリのデザインを変える際の注意点と具体策を挙げてみましたが、まずは初期バージョンをローンチする時点で将来大幅なアップデートをしなくても済むように考えてデザインを決定することが重要です。実際にはそれでもアップデートの必要性は出てくるものですので、前もってリスクを回避する意味でも、これらのことを頭に置いておくと良いのではないでしょうか。

最後に

ここまでアプリのデザインについて書かせてもらいましたが、いかがだったでしょうか？少しでも読者の皆さまのお役に立てる内容であったなら幸いです。

現実でのデザイナとプログラマの共同作業というのは、小さな観点の違いからうまく噛み合ないことも少なくはないと思います。ですが、見ているポイントが違うからこそ生まれる化学反応というのも確かに存在します。目指すゴールが同じであれば、あとはガッチリ組み合って目的地に向かって行きさえすれば些細な違いなど乗り越えられるものだと筆者は考えています。

また、これも筆者の勝手な考えですが「言葉をもって相互理解を促すのはデザイナの担当分野だ」と言っても良いのではないかと思うところもあります。思いや考えを見える形にし、さらにそれをわかりやすく伝わる言葉に代えるのもデザイナの仕事。ですので、もしやり取りでうまくいかないことがあれば、思い切ってデザイナに解決策を提案させるのも良いかもしれませんね。

筆者は地方在住ではありますが、都内近郊で行われるiOSの勉強会などにもときどき顔を出させていただいています。もし何かの機会にお会いすることがあれば、ぜひ気軽に声をかけてみてください。

それでは、皆さまの開発したアプリが1人でも多くのユーザに歓迎されることを楽しみにしています。SD

● 勝間田 雅裕（かつまた まさひろ）

+Beansという屋号でiOSアプリを中心としたデザインを請け負うフリーランス。独学ながら2009年からアプリ開発の勉強もはじめ、2012年には当時話題になったTodoアプリ"Clear"がリリースされた5日後にそのUIを再現したソースコード(Re:Clear)をGumroadで販売した。国内で170万ダウンロードされた"説明書for iPhone"(AppBank)にもデザイン担当として参加している。Twitter @Jacminik URL http://www.plusbeans.com/

第36回 ウィザードを使ったアプリ開発を身につけよう!

モバイルデバイス初のオープンソースプラットフォームとして、エンジニアから高い関心を集めるGoogle Android。いち早くそのノウハウを蓄積したAndroidエンジニアたちが展開するテクニックや情報を参考にして、大きく開かれたAndroidの世界へ踏みだそう!

日本Androidの会 中国支部長
重村 浩二　SHIGEMURA Koji
k-shigemura@android-group.jp

開発環境を使いこなそう!

3月になり、来月からは新入社員という読者の方も多いかと思います。中にはこの4月からAndroidの開発に携わる予定の方も多いのではないでしょうか。Androidの開発環境として提供されているAndroid SDKは日進月歩で進化を遂げており、執筆時点(2月下旬)でr21.1となっています。バージョンアップを遂げることで、初期の頃にはなかったさまざまな機能が追加されてきました。

Androidのアプリ開発を行ううえで避けては通れない開発環境を、もっと便利に使いこなす方法を皆さんにお届けしたいと思います。

Android SDKで追加されたウィザードを使いこなそう!

Eclipseを用いたAndroid SDKのプロジェクト作成ウィザードが大きく変わったことはご存じでしょうか? 書店で販売されている入門書籍に載っている基礎的な使い方はよく見られますが、それ以外の方法では普段あまり使われていない方も多いのではないでしょうか。

この機能、実はかなり力を入れて作り込まれている部分ではないかと筆者は思っています。ウィザードではアクティビティの作成やランチャーアイコンの作成、Androidオブジェクトの作成など、さまざまなことが可能となっています。本稿ではアクティビティの作成を見ていきながら、便利さを実感してほしいと思います。

多彩なアクティビティをテンプレートから作ろう

アクティビティのテンプレートをプロジェクトに追加したい場合には、追加したいプロジェクトが選択された状態で[File]－[New]－[Other]でSelect a Wizardのウィンドウを呼び出して、そこから[Android]－[Android Activity]を選択してウィザードを実行します。アクティビティの作成ウィザードでは、**表1**にあるとおり、5つのテンプレートを提供しています。

使用方法は簡単で、プロジェクトに追加したいアクティビティのテンプレートを選択し、[Next]を押します。

▼表1　アクティビティのテンプレート一覧

テンプレート名	説明
BlankActivity	レイアウトがブランクのアクティビティ
FullscreenActivity	フルスクリーンのアクティビティ
LoginActivity	Webサービスなどで用いられるような、Eメールアドレスとパスワードの入力項目を備えたアクティビティ
Master/Detail Flow	スマートフォンとタブレット両方に対応した画面一式のアクティビティ
SettingsActivity	設定画面のアクティビティ

追加対象のプロジェクトやアクティビティ名を聞いてきますので、それらを確認して問題なければ[Finish]をクリックすることで、アクティビティを追加することができます。アクティビティは独立したモジュールになりますので、**リスト1**にあるように呼び出すアクティビティを明示的に指定したIntentを生成し、startActivity()に引数で指定して呼び出すことで画面遷移を行うことが可能です。

どのアクティビティもそれぞれ特徴を持っていますが、Blank Activityがとくに汎用的なテンプレートになっています。デフォルトの状態で登録した場合は「Hello world!」と画面中央に表示されるアクティビティが生成されますが、**図1**のウィザードで表示される「Navigation Type」を変更することで、次のナビゲーション機能が実装されたアクティビティを生成することが可能です。

- None
- Fixed Tabs + Swipe
- Scrollable Tabs + Swipe
- Dropdown

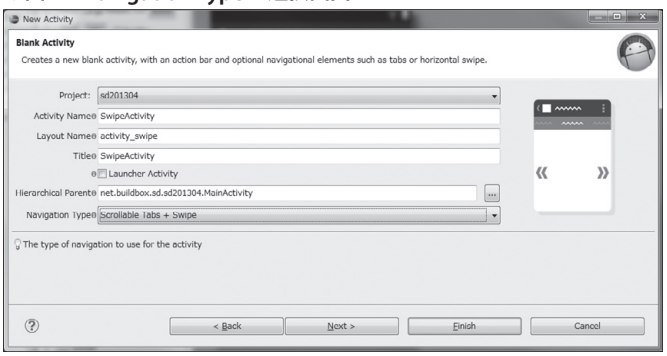
テンプレートを編集しよう

本稿では上記のうち、「Scrollable Tabs + Swipe」のテンプレートを用いてアクティビティを追加した場合の編集方法を見ていきましょう。今回は、テンプレートで3画面の遷移となっているのを5画面に変更し、偶数の画面を変更してみたいと思います[注1]。

▼リスト1　アクティビティの呼び出し

```
public void dispSwipe(View v) {
    // SwipeActivityの明示的な呼び出し
    Intent intent = new Intent(this, SwipeActivity.class);
    startActivity(intent);
}
```

▼図1　Navigation Typeの選択画面



アクティビティを追加しよう

まずは前項で説明した手順でアクティビティを追加しましょう。追加するアクティビティ名は「SwipeActivity」とし、「Navigation Type」を「Scrollable Tabs + Swipe」として追加してください。アクティビティを追加するプロジェクトはBuild TargetがAPI Level 11以降となっている必要があります。

ウィザードでアクティビティを追加すると、プロジェクトに次の変更が追加されます。

- [src]配下にSwipeActivity.javaの追加
- [res]－[layout]配下にactivity_swipe.xmlの追加
- [res]－[menu]配下にactivity_swipe.xmlの追加
- [res]－[values]配下にあるstrings.xmlにリソースの追加
- AndroidManifest.xmlにSwipeActivityを呼び出すための定義の追加

アクティビティを追加しただけでは画面遷移で呼び出すことはできませんので、リスト1にあるようにSwipeActivity.classを明示的に指定

注1）テンプレートを編集した結果をサンプルプログラムとして用意してあります。参考にしてみてください。URL http://sd.gihyo.jp/archive/2013/201304/support/

したIntentを生成し、startActivityで画面遷移を起こすようにdispSwipe()をMainActivity.javaに追加してください。

activity_main.xmlに**リスト2**を記述することで、アクティビティ間の画面遷移ができるようになります。

追加されたアクティビティを編集しよう

続いて、テンプレートで追加されたSwipeActivity.javaを見ていきましょう。SwipeActivityはFragmentActivityから継承されており、次のメソッドが自動生成されます。

- onCreate()
- onCreateOptionsMenu()
- onOptionsItemSelected()

このうちonCreate()では、**リスト3**の個所でスワイプでページを切り替えるためのSectionsPagerAdapterを生成し、クラスのメンバ変数として定義されたViewPagerのオブジェクトmViewPagerにsetAdapter()でバインディングされています。findViewById()で取得してきているレイアウトは、activity_swipe.xmlに定義された<android.support.v4.view.ViewPager>(**リスト4**)です。

■表示ページ数を変更する

SectionsPagerAdapterはFragmentPagerAdapterを継承した内部クラスとして実装されています。SectionsPagerAdapterでは次のメソッドが定義されます。

- SectionsPagerAdapter()
- getItem()
- getCount()
- getPageTitle()

このうちgetItem()で画面上に表示されるフラグメントを、引数positionを元に生成しています。自動生成されるコードでは、内部クラスとしてFragmentから継承したDummySectionFragmentを生成し、画面上にページ番号を表示しています。getCount()では、合計の表示ページ数を戻り値として返しています。getPageTitle()では、各ページのタイトルを引数positionによって返すようにできています。この機能でデフォルトではタイトルバー直下に"SECTION

▼リスト2　activity_main.xmlに追加する画面遷移用ボタンの実装

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:onClick="dispSwipe"
    android:text="@string/labelSwipe" />
```

▼リスト3　onCreate()の実装

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

    // Show the Up button in the action bar.
    getActionBar().setDisplayHomeAsUpEnabled(true);

    // Create the adapter that will return a fragment for each of the three
    // primary sections of the app.
    mSectionsPagerAdapter = new SectionsPagerAdapter(
            getSupportFragmentManager());

    // Set up the ViewPager with the sections adapter.
    mViewPager = (ViewPager) findViewById(R.id.pager);
    mViewPager.setAdapter(mSectionsPagerAdapter);
}
```

▼リスト4　activity_swipe.xmlの実装

```
<android.support.v4.view.ViewPager xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SwipeActivity" >

    <android.support.v4.view.PagerTitleStrip
        android:id="@+id/pager_tab_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#33b5e5"
        android:paddingBottom="4dp"
        android:paddingTop="4dp"
        android:textColor="#fff" />

</android.support.v4.view.ViewPager>
```

▼リスト5　getCount()の戻り値の変更

```
@Override
public int getCount() {
    // ページ数(5)を返す
    return 5;
}
```

▼リスト6　getPageTitle()の実装

```
@Override
public CharSequence getPageTitle(int position) {
    switch (position) {
    case 0:
        return getString(R.string.title_section1).toUpperCase();
    case 1:
        return getString(R.string.title_section2).toUpperCase();
    case 2:
        return getString(R.string.title_section3).toUpperCase();
    case 3:
        return getString(R.string.title_section4);
    case 4:
        return getString(R.string.title_section5);
    }
    return null;
}
```

1"、"SECTION 2"……と表示されます。レイアウトaction_swipe.xmlでは、<ViewPager>の子要素としてリスト4で示した<android.support.v4.view.PagerTitleStrip>を定義する必要があります。

今回は表示するページ数を**リスト5**のように、getCount()の戻り値を「3」から「5」に変更しましょう。

そして、getPageTitle()で4ページ目と5ページ目のタイトルを**リスト6**のように返すようにしましょう。getPageTitle()の引数positionは0から始まりますので、ページ数-1した値のpositionでタイトルを設定します。ここでは**リスト7**のように、[res]－[values]のstrings.xmlに表示テキストも忘れず設定しましょう。

■数ページの表示を変更する

偶数のページで表示するフラグメントを変更

するために、今回はリスト8にあるColorSectionFragmentを内部クラスとして用意しました。デフォルトのページと区別するために、画面全体にTextViewを表示し、背景色を青色としています。そして、フラグメントの生成を担っているgetItem()で、リスト9にあるように偶数ページのときにColorSectionFragment()を返す実装としてあります。

▼リスト7　strings.xmlへのリソースの追加

```
…省略…
<string name="title_section5">5ページ目</string>
<string name="title_section4">セクション4 </string>
<string name="title_section3">Section 3</string>
<string name="title_section2">Section 2</string>
<string name="title_section1">Section 1</string>
…省略…
```

Strip部分を変更しよう

テンプレートではPagerTitleStripが実装されますが、簡単に変更することが可能です。ここでは、PagerTabStripに変更してみましょう。activity_swipe.xmlを開き、<android.support.v4.view.PagerTitleStrip> を <android.support.v4.view.PagerTabStrip>に変更することで、タブ画面のStripに変更されます。UIの操作面ではほとんど変化はありませんが、タイトルに下線が表示され、どのページが選択されているのかがよりわかりやすくなります。タイトル下部の下線は2種類あります。タブ全体に引かれた下線の表示有無をsetDrawFullUnderline()で、タブのインジータ部分は色をsetTabIndicatorColor()で変更することが可能です(リスト10)。

▼リスト8　ColorSectionFragmentの実装

```
public static class ColorSectionFragment extends Fragment {

    public ColorSectionFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        TextView textView = new TextView(getActivity());
        // テキストビューを画面全体に配置
        textView.setLayoutParams(
                new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.MATCH_PARENT));
        // テキストビューの背景色を青色に変更
        textView.setBackgroundColor(Color.BLUE);
        return textView;
    }
}
```

▼リスト9　getItem()の変更

```
@Override
public Fragment getItem(int position) {
    // 偶数のページの場合に生成するFragmentを変更する
    if (position % 2 != 0) {
        Fragment fragment = new ColorSectionFragment();
        return fragment;
    } else {
        Fragment fragment = new DummySectionFragment();
        Bundle args = new Bundle();
        args.putInt(DummySectionFragment.ARG_SECTION_NUMBER, position + 1);
        fragment.setArguments(args);
        return fragment;
    }
}
```

Column

前画面への遷移も簡単に実装可能!

ウィザードの中で「Hierarchical Parent」に親となるアクティビティを指定すれば、前画面への遷移もウィザードで実装を行うことが可能です。

用意したサンプルであれば、「SwipeActivity」の実装時に「Hierarchical Parent」にて「net.buildbox.sd.sd201304.MainActivity」を指定することで、MainActivityに画面遷移するための次の実装を自動生成しています。

SwipeActivity#onOptionsItemSelected() でgetItemId()の戻り値がandroid.R.id.homeの場合に、Android Support Libraryで提供されているNavUtils#navigateUpFromSameTask()を呼び出しています。

この機能を利用するには、AndroidManifest.xml上のSwipeActivityの<activity>タグの定義で、android:parentActivityNameに"net.buildbox.sd.sd201304.MainActivity"と定義し、<meta-data>としてandroid:nameが"andorid.support.PARENT_ACTIVITY"、android:valueが"net.buildbox.sd.sd201304.MainActivity"と定義される必要があります。

また、onOptionsMenuSelected()でandroid.R.id.homeの呼び出しが行われるように、onCreate()内でgetActionBar().setDisplayHomeAsUpEnabled(true)を呼び出しています。

▼リスト10　PagerTabStripの制御

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    …省略…

    PagerTabStrip pagerTabStrip = (PagerTabStrip) findViewById(R.id.pager_tab_strip);
    // タブ部分に下線を表示する
    pagerTabStrip.setDrawFullUnderline(true);
    // タブインジケータの色を黄色に変更する
    pagerTabStrip.setTabIndicatorColor(Color.YELLOW);
}
```

まとめ

今回はウィザードを用いた場合のアクティビティの追加方法を見ていきました。ここでは紹介しなかったその他のアクティビティも同様に、テンプレートから既存のプロジェクトにアクティビティを追加して、独自の実装に変更することが可能です。ただし、ここまでの解説でもみてきたとおり、ウィザードでアクティビティを追加する際の機能には、Build TargetのAPI Levelが11以降でないと利用できないものがあります。当面は、Gingerbread(API Level:9)以前のバージョンでも活用できるように、独自の実装なども必要になるでしょう。

利用の制限があることからも、すべての場面で使えるわけではありませんが、テンプレートの利用方法を覚えておくことで定型的なコーディングを減らし、開発の生産性を向上させることもできます。ぜひ活用してみてください。SD

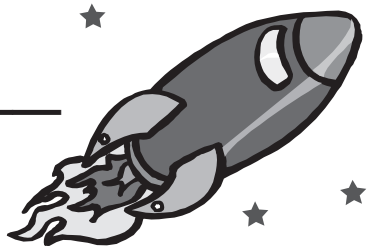
重村 浩二(しげむら こうじ) 日本Androidの会 運営委員 中国支部長

日本Androidの会にて運営委員、中国支部長として毎月山口県・広島県を中心に自作アプリの発表やハッカソン、ハンズオンなどを中心とした勉強会を精力的に開催。個人としても、雑誌への寄稿などを中心に活動中。
URL http://buildbox.net/
Twitter @shige0501

2

Debian Developer　やまねひでき　henrich@debian.org

Debian界隈の最近の動き——環境整備とイベント報告



Debian Hot Topics

はじめに

前回の記事についてVine Linuxの開発者の方と雑談した際に「そんなに細かくルールが決まってるのはDebianぐらいだよー」と言われてしまいました。ポリシー周りの話はディストリビューションごとに温度差がありそうですが、パッケージ管理周りについてはパズルの楽しさがあるというのはPlamo Linuxのこじまみつひろさんも仰っている[注1]ので、Debainに限った話ではないようです。

さて、本連載のタイトルは「Debian Hot Topics」なのですが、このタイトル、じつは第1回の原稿が書き終わるころに決まったので、前回の3月号ではあまりホットなトピックを挙げられませんでした。今回は目にとまった出来事をいくつか取り上げたいと思います。

debian-mirror.sakura.ne.jp稼働開始

3月号の発売と前後しますが、Debian JP Projectでは、さくらインターネット(株)の協力のもと、Debianリポジトリのミラーサーバを追加設置させていただきました(宣伝)。さくらのVPSを利用されている方は、とくに意識せずにftp.jp.debian.orgとして新サーバを参照いただけます。

こちらで利用されているGeoMirror実装は開発者でプロジェクトメンバーの荒木靖宏さん(@ar1)によってGitHubにソースが公開されていますので[注2]、興味のある方はご参照ください。

ミラーサーバですが、みなさんの協力のおかげでかなり充実してきているので、将来的には、通常のリポジトリ以外にもアジア太平洋地域に現状存在していないsecurity.debian.orgのミラーを提供できれば……と考えています[注3]。

clang.debian.net

LLVMのフロントエンドであるclangですが、Debianではclang2.9から継続してすべてのDebianパッケージに対してビルドを行い、そのエラー状況をトラッキングしています[注4]。clang3.2では18264パッケージに対して2204(12.1%)がビルドに失敗しており、3.1のときと比べて対象パッケージ数が増えたものの、ほぼ同一の割合でビルドできているとのことです。この背景としては開発元がclangでの問題を把握し対処し始めている、あるいはパッケージ側でパッチを当てているものが増加していること

注1) URL http://gihyo.jp/lifestyle/serial/01/ganshiki-soushi/0042?page=2

注2) URL https://github.com/armaniacs/CnameQRV4

注3) 結構要件が高く、潤沢なネットワーク接続以外にも、Debianシステム管理チームがrootとして管理／仮想マシン不可／SSH以外のシリアルコンソールなどのアクセス提供／ファイアウォールによるアクセス制限なし／ハードウェアの冗長構成(保証付き)といったことが求められるのでなかなか難しいです。相談にのっていただける方がいましたら、ぜひ筆者まで連絡をお願いします。

注4) URL http://clang.debian.net/

が挙げられています。どんなエラーが出ているのか興味がある方はサイトを確認してください。手元の環境でビルドチェックしてみたい方もいらっしゃると思いますので、現在簡単に確認できるように対応を進めています。しばらくお待ちください(いったんはpbuilderにオプションを付けるというハック[注5]を行ったのですが、pbuilderのメンテナから「オプションを増やしたくない」と拒否されてしまったので、別の方法を試みています)。

Debianプロジェクトリーダー(DPL)選挙

さて、年に一度のDPL選挙が近づいてきました。Debianでは毎年プロジェクトのリーダーを投票によって選んでいますが、今回はこれまで3選を果たしていた現DPLのStefano Zacchirolliは立候補しない旨表明しているので、本命がいない状態です。すると、なぜか「DPL Game」なる遊びがDebian開発者間で流行り始めました。これはFrancesca Ciceriさんが始めた「みんなにDPLに相応しいと思う人を知らせよう!」と、Fantastic Four＝すばらしい人を4人挙げるものです。ただし「挙げた人に対してDPLに立候補するようプレッシャーをかけるのが目的じゃないよ」と一応釘を刺しています。この話題に興味がある方はハッシュタグ「#DPLgame」をどうぞ。

パッケージング道場開催、OSC浜松／東京

昨年開催された大統一Debian勉強会[注6]の開催地である京都大学で、2013年2月9日に「Debianパッケージング道場」という名前でDebian開発者の岩松信洋さん(@iwamatsu)が師範となってワークショップが行われました。参加者はハードながらも濃密な時間を過ごせたようです。また同日に、Debianメンテナとして活躍中の赤部晃一さん(@vbkaisetu)が中心となって、OSC浜松[注7]に東京エリアDebian勉強会として初めての出展が行われました。東京エリアDebian勉強会は同月のOSC東京でも出展／発表を行うなどアクティブに活動していたようです。

関東／関西地域では毎月Debian勉強会が開催されていますが、福岡などでも有志による活動が行われています。Debian関連イベントが気になる方はDebian JP Projectのサイトをチェックしてください。

Debian6.0.7リリース

みなさんお待ちかねのDebian7.0は……、この記事を書いている時点ではまだリリースされていませんが、6.0のアップデートが行われ6.0.7となりました。セキュリティ修正の反映のほか、6.0リリース後に見つかった重大な問題の反映、ならびにLinuxカーネル更新によるサポート対象ハードウェアの追加が行われています。

安定版の更新の裏側

さて、上記の更新版についてですが、セキュリティ修正はともかくとして、ほかの更新はどのようにして適用が決まっているか、というのは案外知られていないようですのでここで解説をしましょう。

(0) パッケージメンテナが安定版のパッケージに影響がある「重大な問題」に気づくところからすべては始まります
(1) debian-releaseメーリングリストにおいて、パッケージメンテナが「この変更を安定版のパッケージに適用したいのだけど」と、問題に対する変更分のdiffを添付して提案します
(2) リリースマネージャーがGoサインを出し、パッケージメンテナは更新したパッケージをマスターサーバのp-u-newキューへとアップロー

注5) URL http://bugs.debian.org/700290
注6) URL http://gum.debian.or.jp/
注7) URL http://www.ospn.jp/osc2013-hamamatsu/

ドします。これでユーザがパッケージを「stable-proposed-updates」リポジトリで取得可能になります

(3) 一定期間とくに問題が報告されず、リリースマネジャーが承認すれば次の安定版の更新へ取り込まれます

「重大な問題」はDebianバグトラッキングシステムでいうところの「important」以上の重要度のバグに当たります。ただ、問題が大きくても変更点が巨大過ぎる、修正によるほかのパッケージへの影響が多過ぎるなどと判断されると、残念ながらproposed-updates(更新候補)からstableへの反映は行われません。このあたりは厳格な条件があるわけではなく、リリースマネージャーの「説得」次第で、ケースバイケースとなっています。

筆者は2chブラウザ「JD」パッケージのメンテナなのですが、過去のリリースではたいへん不幸なことに安定版のリリース後に2ch側の仕様が大幅に変更されてしまい、そのままでは利用できなくなってしまったことがありました。対応として、修正が行われた新バージョンから必要な変更点のみをピックアップしたパッチを作ってレビューを受け、proposed-updatesへとアップロード→安定版へ反映、という作業を行った経験があります。

ただ、正直なところ、対応が必要な部分だけを探し出していく作業はしんどいわりに、新しいバージョンの機能が使えるようになるわけではないのでユーザ受けもよくありません。筆者個人としてはbackportsのほうへ新しいバージョンを入れるほうが気が楽ですね。

安定版でサーバを運用する方へ

複数のサーバを安定版で運用しているユーザは、安定版の更新に伴って「急にアップデートが何十個も降ってきた!」と叫ぶのではなく、事前にテスト用のマシンを用意してproposed-updatesの設定をapt lineに追加して変更内容が問題ないか、をチェックしておくのが賢い運用です。proposed-updatesリポジトリに置いてあるパッケージは「問題がなければ次のアップデートリリースに入るパッケージ」です。ただ逆に言うと、変更によって生じた問題が存在している可能性もそこそこあります。もちろん、何か問題が見つかった場合はリグレッションとしてメーリングリストやバグトラッキングシステムへ報告するのもお忘れなく。

また、実際の更新時期を知りたい場合は、debian-stable-announceメーリングリスト[注8]という告知専用のものがあるので併せて購読しておくとよいでしょう。

debian-installer7.0RC1リリース

さて、一方インストーラについては進展がみられており rc1がリリースされました。大幅なドライバ追加によるハードウェアサポートの充実や、メディアのマウントやネットワーク設定の修正などが含まれています。なお、比較的最新のマシンにはBIOSではなくUEFIが搭載されていて「CSM(Compatibility Software Module)」と呼ばれるBIOS互換のモジュールが動作するようになっています。しかし、ごく一部の環境ではこのCSMが無効にされていることがあり、その絶対数が少ないため十分なテストが行われていない可能性があります。UEFIでの既知の問題については修正がこのリリースで行われているものの、上記関連で何かしらの問題にあった方は解決のためにメーリングリストでの相談やバグレポートなどをお願いします。

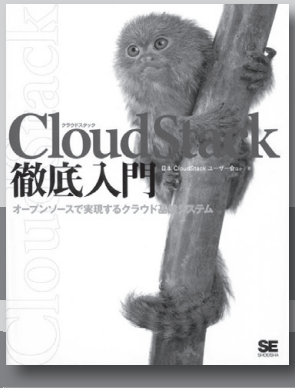
最後に

読者のみなさん方の興味を引くような話題はありましたか？ ぜひご意見ご感想を@gihyojpまたは@debianjpまでお寄せください。SD

注8) URL http://lists.debian.org/debian-stable-announce

BOOK no.1 CloudStack徹底入門

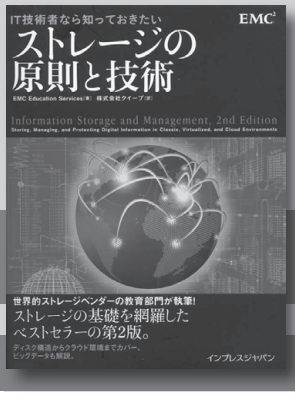
日本CloudStackユーザー会ほか 【著】
B5変形判、 392ページ／価格＝3,990円 （税込） ／発行＝翔泳社
ISBN＝978-4-7981-3058-3



本誌3月号の特集でも取り上げたCloudStackの導入から活用までを解説。本書でその機能の概要をとらえることができる。CloudStackはデータセンター各社で非常に推されている。それはプライベートクラウドを容易に構築できるからで、ユーザとデータセンターのメリットがぴったり一致するからだ。同様にOpenStackも合わせ鏡のような存在。オープンソースでのIaaS構築については、この両雄がしばらく鎬を削っていくのだろう。これから発展していく技術なので今が乗り込むチャンスなのだ。ユーザー会執筆のためか、読者目線でさまざまな技術が解説されている。本書で自分なりのクラウド環境を構築してみるのはいかがだろうか。

BOOK no.2 IT技術者なら知っておきたい ストレージの原則と技術

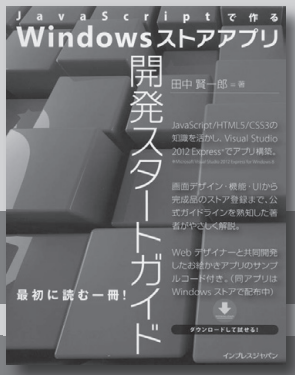
EMC Education Services 【著】 ／㈱クイープ 【訳】
B5変形判、 464ページ／価格＝4,830円 （税込） ／発行＝インプレスジャパン
ISBN＝978-4-8443-3351-7



本書は米国マサチューセッツにあるEMC Corporation（VMwareの買収などでも有名）のトレーニング部門、EMC Education Services著の翻訳書である。全体は4つのセクション（①ストレージシステム、②ストレージネットワーキングテクノロジー、③バックアップ／アーカイブ／レプリケーション、④ストレージインフラストラクチャのセキュリティと管理）に分けられ、基本的なHDDの構造からデータセンターの運営にいたるまで幅広く解説されている。第2版ということで、最新のFCoEやフラッシュドライブ、ビッグデータなどの項目も追加され、ストレージの初歩から最新状況まで幅広い知識をまんべんなく学習することができる。

BOOK no.3 JavaScriptで作る Windowsストアアプリ開発スタートガイド

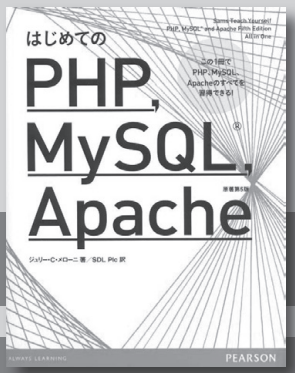
田中 賢一郎 【著】
B5変形判、 340ページ／価格＝2,940円 （税込） ／発行＝インプレスジャパン
ISBN＝978-4-8443-3352-4



Webアプリ開発経験者がWindows 8のネイティブアプリである「Windowsストアアプリ」の開発をはじめるのにお勧めの本書。JavaScript/HTML/CSSで開発が可能になったこと、統一感を維持するためにデザインコンセプトが導入されたことなど、新しいことづくめのWindowsストアアプリに関する知識がしっかりとまとめられており、開発者心得書とも言えそうだ。Webアプリ開発の経験をWindowsストアアプリ開発へ活かすために、アプリの構造や動作のしくみ、開発ツールやJavaScript/CSSの書き方のポイントなどがていねいに説明してある。物理演算エンジンを使ったサンプルアプリで、実践的なコーディングも学べるだろう。

BOOK no.4 はじめてのPHP,MySQL,Apache（第5版）

ジュリー・C・メローニ 【著】 ／SDL Plc 【訳】
B5変形判、 624ページ／価格＝3,990円 （税込） ／発行＝ピアソン桐原
ISBN＝978-4-86401-111-2



本書は、Webシステムの開発未経験のエンジニアがWeb開発の概要を短時間で学ぶのに最適な1冊だ。PHPの文法、MySQLやApacheの使用方法といった情報はエッセンス程度にとどめ、半分以上はこの3つのソフトウェアの連携方法や組み合わせたときのチューニング方法の解説だ。PHPやMySQLに詳しくなくてもほかの言語やDBの知識があれば、十分に理解できる。簡単なWebシステムを作成しながら、PHPでMySQLにアクセスする方法、DB設計の指針、Apacheを使用したWebシステムのアクセス制御、PHPとMySQLを使った簡単なアクセス追跡、パフォーマンスチューニングなど、Webシステム開発の要点を効率よく学べる。

第7回

Unique OSS
――日常業務から感じる「OSSならでは」

三木 雄平
MIKI Yuhei
レッドハット(株)
JBossサービス事業部
シニアソリューションアーキテクト

本日のお題

こんにちは。レッドハットでミドルウェアのソリューションアーキテクト(以下、SA)を担当している三木と申します。SAとは前回で説明しましたが、要はプリセールスエンジニアです。ユーザ先に伺い製品の説明をしたり、最適なシステム構築するにはどの製品のどの機能を使うべきか等々、日々提案活動を中心に業務をこなしています。筆者が担当しているJBossはJava EEアプリケーションサーバを中心とした、企業向けのOSSミドルウェアになりますが、今回の連載では日頃業務を遂行するうえで感じる「OSSならでは」について紹介したいと思います。

効率的な製品のキャッチアップが可能

前回の連載で紹介させていただきましたが、JBoss Enterprise Middlewareは多くの製品から構成され、扱う技術も多岐にわたります。また、新製品も続々リリースされるため、多くの製品を早急にキャッチアップする必要があります。その際、ソースコードが閲覧できるのは非常に効率的です。

以前、筆者は別の外資系ミドルウェアベンダで働いておりましたが、通常、プリセールスエンジニアは製品のソースコードを閲覧できません。下手をするとサポートエンジニアですらソースコードを読むことができない会社もあります。その点、レッドハットではサポートエンジニアはもちろん、SA、コンサルタント、および(見ないとは思いますが)営業と役割に関係なく、誰でもソースコードを閲覧し、解析できます。

個人的には新しい製品や技術を効率的にキャッチアップするには、まずはドキュメントを軽く読みながら、動かしてみて全体像をつかみ、いわゆる、“腹落ちフェーズ”でソースコードを追いかけるのが最速だと思っています。また、ソースコードを読むことでドキュメントでは表現できないようなソフトウェアの性格的なものが理解できると感じています。あと、コードは読めば読むほど、その製品に愛着が湧いて(たとえ、あり得ないようなトラブルが発生したとしても「困ったヤツだな」くらいで許せる)という不思議な感覚に襲われるのは筆者だけでしょうか……。

技術的な裏付けが簡単に取れる

通常、レッドハットでは製品の動作に関する質問などはサポートチームで対応するのですが、SAもユーザさんとの会話の中で「あの機能ってどうなってましたっけ？」的な質問を受けて、挙動を調べることが多々あります。その場合、ドキュメントを確認したり、有識者に聞いたり、または実際に製品を動かして調査するわけですが、ここでもソースコードを自分で確認できるというのは非常に有益です。ソースコードを確認することで絶対的な裏付けを取ることができますし、結果、ユーザさんにも自信をもって回答することができます。

この点ではトラブルシューティングのときにも同様のことが言えます。トラブルシュートもサポートチームを中心に対応することがほとん

どですが、プリセールスでもそのトラブルの本質を理解する必要があります。とくにそのトラブルの原因が製品のバグであった場合、そのバグの本質を理解するには該当部分のソースを見ることが一番の近道になります。

エンジニア同士の熱い会話ができる

ただ、当然ですが、ソースコードを閲覧できるのは我々だけでなく、誰でも(つまりエンドユーザさんも)見ることができます。さらに言うとソースコードだけでなく、開発者のやりとりや、ロードマップに関する方向性、クリティカルなバグ修正の経緯など、ほとんどの情報をインターネット上で手に入れることができます。このため、客先訪問すると時々ものすごくJBossに詳しい人に出会うことがあります。人によっては「ソースコードを読むのが大好きで、とくに好きな製品のソースコードを読むのは至福のときです」という強者(過去、何人かこういう人に会っています)や、トラブル時に、しかたなく読み進めているうちに詳しくなった方、それぞれ事情は異なりますが、そういった"JBoss通"のエンジニアと会話すると、非常にエキサイティングで有意義な時間になります。「同じ釜の飯を食う」と言う言葉がありますが、同じソースコードを読んで、同じ疑問や悩みを解決したエンジニアとの会話は格別です(そのまま飲みに行きたいくらいです)。

ただ、こちらがあまりその機能に精通していなかった場合……、正直なところ冷や汗をかきまくることになります。ということで、なるべくそうならないように、我々SAは日々、情報収集を行い、ソースコードを読み、検証を繰り返すという"修行"を積む必要があるわけです。

ユーザさんと戦友になれる

OSSの世界では「俺がJBossを開発してやる!」と考えるユーザーさんが出てきたりします。企業によっては(きっとそんなに簡単ではないとは思いますが)「じゃ、やっとく?」ということでJBossの開発に参加したいと表明されるエンジニアおよび企業さんがいらっしゃいます。通常、一般的な製品ベンダであれば難しいと思いますが、レッドハットの場合、そういった企業やエンジニア(実力と情熱必須)を開発チームに受け入れる体質が整っているので「あ。どうぞ、どうぞ」という感じで受け入れることが結構あったりします。

こうなってくると、「ユーザさん」というよりは同じソフトウェア製品を扱う『戦友』的な関係になり、より「一緒に仕事をしている感」がさらに沸いてきます。

こういったことができるのは「OSSならでは」ですし、開発に参加したエンジニアは技術的モチュベーションを、派遣した企業は最新技術情報と直接、要望や機能拡張要求などをインプットできるパスを、レッドハットとしてはお客さんとの強いコネクションと直接フィードバックがもらえるしくみを確立でき、まさに皆がハッピーになれる関係を構築することができます。

ということでJBossの開発者を輩出し、今もっとも日本でJBossに詳しい某大手ユーザの戦友の皆さんと究極のJBoss本である『JBoss Enterprise Application Platform 6完全ガイド(仮)』を執筆中です。6月には発売予定です。もちろん、出版社は技術評論社です。

最後に『恵比寿』について

恵比寿通信ですので最後にとって付けたように恵比寿という土地(駅周辺?)について。恵比寿はお洒落なレストランやカフェがたくさんありますが、1人で気軽に入れるバーもたくさんあります。個人的にはウィスキーが好きですので、遅くまでOSS製品と格闘した日、ふらっと1人で立ち寄って1〜2杯軽く飲んで帰れる場所がたくさんあって非常に良い感じです。SD

第36回 Ubuntu Monthly Report

Ubuntu Minimal CDでフットプリントの小さな環境を構築する

今回はUbuntu Minimal CDでメモリとHDD/SSDの消費量が小さなGUI環境を構築します。

Ubuntu Japanese Team　あわしろいくや　AWASHIRO Ikuya　ikuya@fruitsbasket.info

Ubuntu Minimal CDって何?

今さら言うまでもなく、Ubuntuとその派生版の魅力の1つはCD/DVD1枚で必要な環境が一通りそろうことです。一方、最小のパッケージから構築したいという用途のために、Ubuntu Minimal CDも用意されています。これは本当に小さなISOイメージ(30～35MB)で、ほとんどのパッケージはネットワーク経由で取得します。ごく基本的なパッケージだけがインストールされ[注1]、ほかに必要なものは自分で選択できます。CUI環境であればとくに困ることはないのですが、GUIの場合はいくつかポイントを押さえておかないと正しく動作しない部分があるので、今回はその方法の紹介です。

なぜそのようなことをしようと思ったのかというと、在庫処分品のASUS EeeBox PC EB1020を手に入れたからです。スペックは**表1**のとおりですが、APU[注2]はさておきメモリが2GB、SSDが8GBだとUbuntuをインストールして活用するには厳しいスペックです。メモリは足すことができますが、SSDの交換は不可とのことです。頑張ればできるようではありますが、そもそも安価で手に入れたものに高価なSSDを追加するのは何かもったいないというか、そのようなことをするのであればもっと高価なPCを購入すべきでしょう。メモリに関しては余っているものがあれば使用してもいいでしょうし、2GBのSO-DIMMであれば2,000円もしないで購入できるようですので、SSDの使用量は削るもののメモリはそれほど削らないことにしました。そもそもいわゆるネットトップ[注3]として使用するにしても、昨今のWebブラウザを使用するのであればタブの20や30は平気で開きますし、そうなるとメモリ2GBでは到底足りませんので、増設を検討するべきでしょう。

注1) その割にはインストールしただけで1GBを超えますが……。
注2) ざっくりCPU＋GPUだと思ってください。

表1　ASUS EeeBox PC EB1020のスペック

CPU	AMD C-60 APU with Radeon HD Graphics
メモリ	2GiB DIMM DDR3 同期 1600 MHz
グラフィック	Wrestler［Radeon HD 6290］
SSD	8012MB SanDisk SSD U100
無線LAN	AR9285 Wireless Network Adapter
有線LAN	RTL8111/8168B PCI Express Gigabit Ethernet controller

Ubuntuのバージョン

このようなPCを使用する場合、考え方は2つあります。1つはLTSを使用することにより、可能な限り長く使うようにすることです。12.04 LTSであればあと4年間は使えるので、1万5,000円で購入したPCであることを考えると、元が取れたと思えるでしょう。もう1つは最新バージョンを乗り継いでいく、すなわち12.10をインストールして13.04にアップグレード、さらに13.10にアップグレード、と性能が

注3) 最近はあまり耳にしなくなりましたが、Web端末用のPCのことです。

追いつく限りアップデートしていく方法です。確かにインストールしてあるパッケージが少なければ少ないほどアップグレードに失敗する確率は下がります。よって、今回は両方とも紹介します。本当は12.04と13.04で紹介したかったのですが、タイミングが悪かったのかどうやっても13.04がインストールできませんでした。

Ubuntu Minimal CDの入手

Ubuntu Minimal CDの情報はWiki[注4]にまとめられています。今回は12.04[注5]と12.10[注6]を使用しますが、13.04[注7]もあります。悩ましいのはCDというところで、残念ながら[スタートアップ・ディスクの作成]でイメージをUSBメモリに転送できず、CDドライブを用意しなければいけないところです。ASUS EeeBox PC EB1020にはないため、今回は手持ちの外付けDVDドライブを使用しました。

12.10で環境構築

検証していて驚いたのですが、12.04と12.10で構築の方法、具体的にはインストールするパッケージがずいぶん異なります。よって、インストールする方法は別立てで紹介することにしました。まずは12.10からです。13.04は前記のとおり検証できませんでしたが、おそらく12.10とほぼ同じだと思います。

インストーラは、今はなくなってしまったAlternate CDと同じで、原則としてデフォルトのままインストールすればいいのですが、8GBのSSDにメモリと同じだけ、すなわちメモリを増設していない場合は2GBものスワップ領域を、増設したらその分だけさらにスワップ領域を取ることになるので、メモリの増設はインストール後にするのがいいでしょう。今回はスワップ領域は用意しませんでした[注8]。[ディスクのパーティショニング]で[手動]を選び(**図1**)、[新しいパーティションのサイズ]を[8GB]にします。その後[ソフトウェアの選択]が表示されますが、とくに何も選択しなくてもいいですし、確実にSSHを使うのであれば[OpenSSH server]にチェックを入れるといいでしょう。

インストールが完了して再起動したあと、実際にパッケージをインストールしていきます。まず、X関連のメタパッケージがインストールされていないため、Xが起動しません。というわけでインストールします。

```
$ sudo apt-get install xorg --no-install-recommends
```

次にGNOMEの動作に最低限必要なパッケージをインストールします。これはメタパッケージですので個別にインストールしてもいいのですが、労が多い割に益が少ない、すなわちあまりSSDの使用量削減にはつながらないので、手間と効率を考えてメタパッケージをインストールすることにしました。もっと削減したい場合は、インストールするパッケージを吟味してください。

```
$ sudo apt-get install gnome-core --no-install-recommends
```

注4) https://help.ubuntu.com/community/Installation/MinimalCD

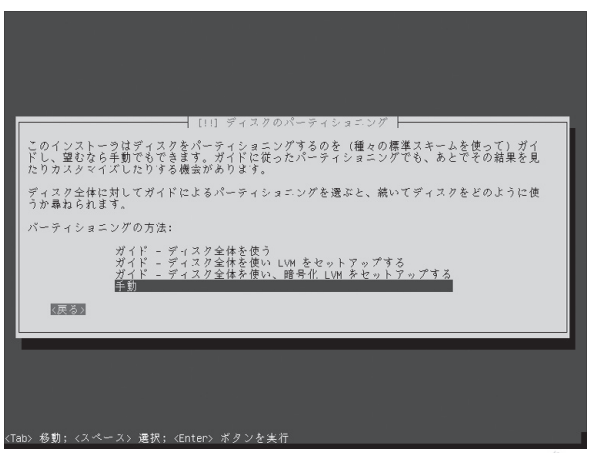
注5) http://archive.ubuntu.com/ubuntu/dists/precise/main/installer-amd64/current/images/netboot/mini.iso

注6) http://archive.ubuntu.com/ubuntu/dists/quantal/main/installer-amd64/current/images/netboot/mini.iso

注7) http://archive.ubuntu.com/ubuntu/dists/raring/main/installer-amd64/current/images/netboot/mini.iso

注8) 2GBでスワップ領域なしだと心許ないのは事実です。

▼図1　スワップを作成しない場合はここで［手動］を選択する



さらりと“--no-install-recommends”というオプションを付けましたが、これはいったい何なのでしょうか。パッケージにはそのパッケージの動作に必要なパッケージ、つまり依存関係があります。Debianパッケージでは依存関係の強弱に応じていくつかの段階がありますが、よく使われるのは動作に絶対に必要なもの(Depends)、絶対ではないものの必須と言えるもの(Recommends)、必須とは言えないもののあったほうがいいもの(Suggests)です。ほかにも衝突するものや置き換えるものなどもありますが、今回は省略します。そのうち、DependsとRecommendsを一緒にインストールするのがデフォルトの動作で、どうしてもインストールするパッケージが多くなります。Recommendsをインストールしないことによってssdの使用量を削減するオプションが“--no-install-recommends”というわけです。

これでGNOMEが起動するところまできましたが、日本語入力ができません。通常はMozcがあればいいと思いますので、インストールします。メモリ消費量は多いですが。

```
$ sudo apt-get install ibus ibus-mozc
```

あとは再起動するか、次のコマンドを実行してGDMを起動します。

```
$ sudo service gdm restart
```

これで最低限のセットアップは完了ですが、Network Managerが使用できないので有線／無線LANの設定が手間です。今回は無線LAN搭載モデルですのでなおさらです。この原因は知ってしまえば割に単純で、Network Managerは/etc/network/interfacesを手動で編集していると起動しないのです[注9]。よって、今回の例だと、

```
auto p5p1
iface p5p1 inet dhcp
```

という下2行を削除し、再起動すればNetwork Managerが起動するようになります。無線LANはとくに何もしなくても認識していました。

APUの性能を引き出すため、プロプライエタリなグラフィックドライバーをインストールしたい場合は、[ソフトウェア・ソース]を使用するのが簡単です。次のコマンドでインストールします。

```
$ sudo apt-get install software-properties
```

[ソフトウェア・ソース]を起動し、[追加のドライバー]タブからインストールしてください。さらに“gvfs-fuse”パッケージもインストールしておくと便利です[注10]。

Ubuntu 12.04で環境構築

12.04での環境構築は若干手間がかかります。Minimal CDからインストールするところまでは同じで、X関連メタパッケージとgnome-coreのインストールもおおむね同様です。

```
$ sudo apt-get install xorg
$ sudo apt-get install gnome-core --no-install-recommends
```

しかし、日本語関連パッケージはIBusだけではなく、フォントなどいくつか足りないものがあるので一緒にインストールします。

```
$ sudo apt-get install ibus ibus-mozc poppler-data cmap-adobe-japan1 fonts-takao-mincho fonts-takao-gothic
```

NetWork Managerを有効にする方法も同じで、/etc/network/interfacesの、

```
auto eth0
iface eth0 inet dhcp
```

を削除して再起動してください。これはどの環境でも(ethが複数ない限りは)同じだと思います。

プロプライエタリなグラフィックドライバーをインストールする方法は12.10とは異なり、“jockey-gtk”パッケージをインストールし、[追加のドライ

注9) 今回はインストーラが手動で設定しています。

注10) これがないと~/.gvfs以下にマウントされず、ファイルサーバに置いてあるドキュメントに保存するといった作業ができません。

バー]を起動してください。“gvfs-fuse”パッケージをインストールしておくと便利なのは同様です。

図2は起動直後のメモリとSSDの使用量で、SSDには4GB程度の空き容量があることがわかります。sdc1はUSBメモリで、環境とは関係ありません。下はメモリの使用量で、700MB程度しか使用していないことがわかります。希望通り、フットプリントの小さな環境になりました。

Cinnamonのインストール

GNOME Shellのインターフェースにイマイチ慣れることはできないとか、誰かに使ってもらうのにGNOME Shellだと都合が悪いといった場合には、Cinnamonを使用するといいのではないでしょうか。GNOME Shellから派生した、旧来のWindowsに似せてあるインターフェースで、比較的馴染みやすいと思います[注11]。ただしUbuntuのリポジトリから取得できるのは13.04からであり、12.10と12.04ではPPAからインストールする必要があります。PPAの登録を簡単に行うadd-apt-repository（apt-add-repositoryでも可）コマンドはインストールされていないのでインストールする必要があるのですが、12.04と12.10で収録されているパッケージが異なります。12.04は“python-software-properties”で、12.10は“software-properties-common”です。該当のパッケージをインストール後、次のコマンドを実行します。

```
$ sudo add-apt-repository ppa:ikuya-fruitsbasket/cinnamon
$ sudo apt-get update
$ sudo apt-get install cinnamon muffin
```

インストール後再起動し、ログイン時に“Cinnamon”を選択してください。

図3はCinnamonです。実機ではうまくスクリーンショットが取れなかったので仮想環境からですが、パッと見て役割がすぐ理解できそうであればCinnamonを選択するといいでしょう。

注11）筆者の感覚的には、Windowsに似ているというよりもKDEに似ている気がしてしかたがないのですが……。

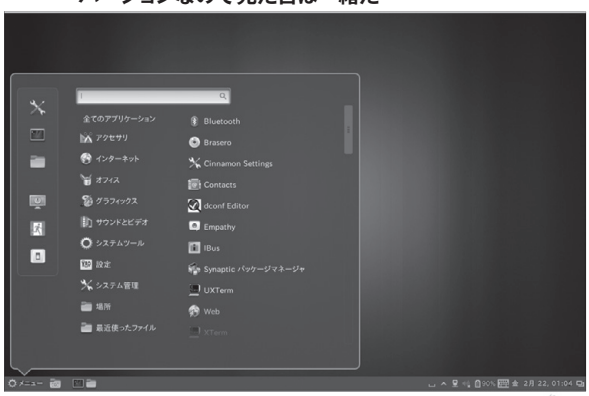
オフィシャルではないパッケージ

1つの考え方ですが、「可能な限り少ないパッケージで環境を構築し、よく使うパッケージはオフィシャル以外のものを使用する」というのはありだと思います。具体的にはGoogle ChromeやLibreOfficeです。Google Chromeはオープンソースのchrominmがありますが、LibreOfficeともども最新版のリリースに追随するのが遅かったり、あるいはUbuntuのバージョンを上げないと最新版にできないといったことがあります。いずれも紙幅の都合上インストール方法は紹介できませんが、問題なく動作していることを確認しています。Google Chromeをインストールする場合は、事前に“libnss3-1d”“libxss1”“xdg-utils”パッケージをインストールしておいてください。SD

▼図2　起動直後のメモリとSSDの使用量

```
ikuya@eeebox: ~
ファイル(F)  編集(E)  表示(V)  検索(S)  端末(T)  ヘルプ(H)
ikuya@eeebox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       7.4G  2.9G  4.2G  41% /
udev            796M  4.0K  796M   1% /dev
tmpfs           322M  396K  322M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            804M  376K  804M   1% /run/shm
/dev/sdc1        29G   19G  8.8G  68% /media/usbdata2
ikuya@eeebox:~$ free -m
             total       used       free     shared    buffers     cached
Mem:          1607        700        907          0         21        227
-/+ buffers/cache:        450       1156
Swap:            0          0          0
ikuya@eeebox:~$
```

▼図3　12.04にインストールしたCinnamonだが、12.10も同じバージョンなので見た目は一緒だ



第13回 perfコマンドの解説

Text：青田 直大　AOTA Naohiro

Linuxのソースツリーのtoolsディレクトリの中には、カーネルと併せて開発されているツールのソースコードが収められています。その中でもperfはもっとも古くからここで開発されているツールです。今回はこのperfに焦点をあて、CPUのどのような機能を使っているのか、カーネルはその機能へのインターフェースをどのように提供しているのか、そしてperfではどのようなことができるのかを解説していきたいと思います。

パフォーマンスカウンタ

最近のx86やARMのCPUには、指定した「イベント」の回数を計測する「パフォーマンスモニタリングユニット（PMU）」と呼ばれるカウンタがあります。ここで計測できるイベントとは、たとえば次のようなものです。

- CPUサイクル数
- 実行された命令の数
- キャッシュミスの数
- 分岐予測失敗の数

こうした計測値を使用することで正確なパフォーマンスの測定が可能になり、プログラムの最適化に役立てることができるようになります。「具体的にどう役立てるのか?」はのちほど見てみるとして、まずは、そのイベントをどうやって設定するのか、カウンタにどうやってアクセスするものなのかを見てみましょう。

パフォーマンスカウンタは、x86ではIntelでもAMDでもモデル固有レジスタ（MSR）として実装されています。MSRはその名のとおりモデルに固有で使うことができるレジスタです。読み出すにはECXレジスタに読み出したいMSRの番号をいれて、“rdmsr”命令を呼ぶとMSRの64bitのうち上位32bitがEDXレジスタに、下位32bitがEAXレジスタに入ってきます。モデル固有というその名のとおり、どのモデル番号でどのような値が返ってくるかはCPUのモデルごとに異なっています。たとえば、Intelであれば0x19CからCPUコアの温度情報を取得できます。

MSRからの読み込み

では、実際にMSRからの読み込みをやってみましょう。“rdmsr”命令はCPUの「特権レベル0」でしか実行できません。これはつまり、「カーネルモードでなければ実行できない」ということです。ですので、カーネルモジュールを書くか、あるいは「/dev/cpu/<コア番号>/msr」からの読み出しで書くかの2択となります。ここでは/dev/cpuからの読み出しでやってみましょう（**リスト1**）。

Nehalem以降のIntelプロセッサであれば、こ

うすることでCPUコア0の温度を取得できます。簡単にコードを解説しておきます。まずCPUコア0のMSRを取得するために/dev/cpu/0/msrをopen()しています(リスト1-❶)。このファイルデスクリプタからoffsetをMSR番号に指定して64bit幅で読み出すと、指定したMSRの値を取り出すことができます(リスト1-❷)。ここでは"tjmax"としてMSR[IA32_TEMPERATURE_TARGET]の23:16(16bit目から23bit目)を、"temp"としてMSR[IA32_THERM_STATUS]の23:16を読み込んでいます(リスト1-❸)。"tjmax"のほうはCPUの温度制御回路(TCC)が作動する温度になります。一方で"temp"のほうは、この"tjmax"からの差分が取得できます。ですので"tjmax - temp"がCPUコアの温度となります(リスト1-❹)。このプログラムの出力とlm_sensorsパッケージのsensorsコマンドの出力を見れば、MSRを正しく読み出せていることがわかります。

パフォーマンスカウンタの設定

パフォーマンスカウンタへと話を戻しましょう。パフォーマンスカウンタのMSR番号は0xC1から始まり、PMC0は0xC1に、PMC1は0xC2というように、それぞれのモデルのパフォーマンスカウンタの数だけ連続して配置されています。また、どのカウンタにどのようなイベントをカウントするかの設定もMSRへの書き込みで行われます。こちらは0x186から始まりカウンタのほうと同様にPERFEVTSEL0は0x186に、PERFEVTSEL1は0x187にというように配置されています。パフォーマンスカウンタの数は、cpuid命令を使えば取得できるようになっています。通常はcpuidコマンドを使えば良いでしょう。このコマンドの"Architecture Performance Monitoring Features(0xa/eax):"の"number of counters per logical processor"の部分がカウンタの個数となります。

MSR[PERFEVTSELx]は**図1**のような構造になっています。この中のEvent(7:0)とUMASK(15:8)に観測したいイベントに対応した値を設定し、またEn(22bit目)に1を設定することで、カウンタを有効にできます。また、USER(16bit目)とOS(17bit目)によって、ユーザモード(特権レベル1から3)の場合に記録するかどうかと、カーネルモード(特権レベル0)

▼リスト1　/dev/cpuからのMSRの読み出し

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define MSR_IA32_TEMPERATURE_TARGET 0x1A2
#define MSR_IA32_THERM_STATUS 0x19C

int main()
{
  unsigned long long msr;
  ssize_t ret;
  int tjmax, temp;

  int fd = open("/dev/cpu/0/msr", O_RDONLY);  ……❶
  ret = pread(fd, &msr, sizeof(msr), MSR_IA32_TEMPERATURE_TARGET);  ……❷
  if(ret != sizeof(msr)) return 1;
  tjmax = (msr >> 16) & 0x7F;

  ret = pread(fd, &msr, sizeof(msr), MSR_IA32_THERM_STATUS);  ……❸
  if(ret != sizeof(msr)) return 1;
  temp = (msr >> 16) & 0x7F;
  close(fd);

  printf("Core 0: %d\n", tjmax - temp);  ……❹
  return 0;
}
```

の場合に記録するかどうかを設定できます。これによってカーネル空間のイベントだけを記録したり、あるいは逆にユーザ空間のイベントだけを記録できるようになっています。

たとえば、イベントに0x3C、UMASKに0x00を設定すると走ったCPUのサイクル数(haltしていた分などが除かれています)をカウントできますし、イベントに0xC0、UMASKに0x00とすれば実行が終了した命令の個数をカウントできます。

さらに後のバージョンでは機能拡張がされています。自分のCPUで実装されているバージョンは先ほどのcpuinfoの"Architecture PerformanceMonitoring Features (0xa/eax):" の "version ID"を見ると知ることができます。バージョン2ではfixedカウンタというものなどが導入されていますし、バージョン3ではHyper-Threadingに対応するためにMSR[PERFEVTSELx]の21bit目にAny Threadというフラグが導入されています。fixedカウンタは役割が固定されているカウンタです。たとえばMSR[PERF_FIXED_CTR0(0x309)]は、先ほど例として挙げた実行した命令の個数のカウンタに割り当てられています。fixedカウンタはもとから役割が固定されているので、当然のことですがイベント・UMASKを設定する必要もありません。バージョン2ではMSR[IA32_PERF_GLOBAL_CTRL(0x38F)]の32から34bit目でそれぞれfixedカウンタの1から3番めを有効にできます。バージョン3ではMSR[IA32_FIXED_CTR_CTRL(0x38D)]でこれらのカウンタに対して、ユーザ空間・カーネル空間でのオン・オフを切り替えたりなどの調整ができるようになっています。 Any Threadのほうはその名から想像できるように、物理コアを共有するすべての論理コア上でのイベントを集積してカウントするためのフラグとなっています。

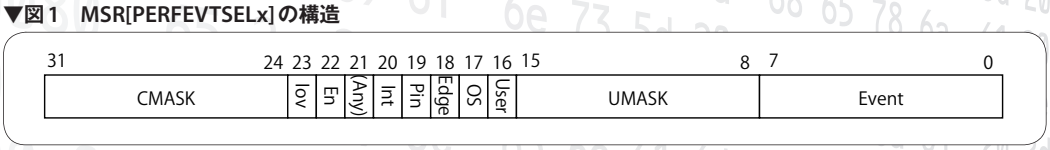
rdpmc命令

ここまでMSRとして実装されているパフォーマンスカウンタを"rdmsr"命令で読み込むとしてきましたが、パフォーマンスカウンタの読み込みに関しては"rdpmc"という特殊な命令が用意されています。この命令は、ECXレジスタの0から30bit目まででカウンタの番号を指定し、MSRの場合と同様にEDXレジスタに上位32bitが、EAXレジスタに下位32bitが入ってきます。また、ECXレジスタの30bit目がセットされていないときには通常のカウンタが、セットされているときにはfixedカウンタが選択されます。つまり、ECXレジスタに0x00000001を指定すれば通常のカウンタの2つめが、ECXレジスタに0x40000002を指定すればfixedカウンタの3つめが選択されるというわけです。

rdmsrの代わりにrdpmcを使うほうが読み込みにかかるサイクル数を低減できます。perfのカーネル側実装で可能であれば、rdpmcを使うようにしたpatch[注1]に、実際どの程度サイクル数が少なくなるのかを計測するカーネルモジュールのコードが添付されています。rdpmc-module.cとMakefileにあたる部分(obj-mの行から)を保存し、makeしてinsmodでカーネルモジュールを読み込めば計測が行われ、ログに結果が出力されます。たとえば筆者の環境では**図2**のようになりました。rdpmcには89.0サイクル、rdmsrには270.1サイクルかかっていて、なかなか大きなサイクル数の差がでていますね。

注1) http://permalink.gmane.org/gmane.linux.kernel/1255785

▼図1　MSR[PERFEVTSELx]の構造



perfのカーネルインターフェース

さて、ここまでIntelでのパフォーマンスカウンタの読み方を見てきました。Intelの中だけでもMSRの名前どおりモデルごとの差異がありますが、AMDではさらにその差異は大きくなります。さらに言えば、ARMにもパフォーマンスカウンタがついていますので、これも考え出すとかなり面倒な場合分けが必要になってしまいます。そこでLinuxカーネルではこれらの抽象化した仮想的なカウンタのインターフェースを提供しています[注2]。

このインターフェースは、ハードウェアパフォーマンスカウンタがないアーキテクチャ向けにソフトウェアで実装されたパフォーマンスカウンタなどへのアクセスも提供しています。さらに、カーネルの各所に設定されたトレースポイントも計測できます。

たとえばEXT4でファイルシステムのsyncを行う、ext4_sync_fs()関数の中には"trace_ext4_sync_fs(sb, wait);"という行があり、この部分が実行されたタイミングやそのときのsuper block(sb)、waitの値を見ることができるようになっています。

perfのカーネルインターフェースは**図3**のようなシステムコールとなっています。引数を適切に設定することで、指定したイベントのカウンタとなるファイルデスクリプタが返ります。これを64bit幅でread()することで、そのイベントが何度計測されたのかを知ることができます。詳しい中身は割愛しますが[注3]。perf_event_attr構造体によってどのようなイベントを計測するかを指定します。pidとcpuは**表1**のように、イベントの計測範囲を設定します。group_fdは新しく作るファイルデスクリプタが所属するグループを指定するためのものです。これについては後述します。flagsは今のところ使われていません。

注2) 「生の」ハードウェア・カウンタへのアクセスも可能なように作られています。

注3) linux/tools/perf/design.txtを参照してください。

▼図2　筆者の環境での計測結果

```
# insmod ./rdpmc-module.ko
# dmesg
...
[98241.490646] COUNTER_OVERHEAD: vendor 0, family 6, model 42, stepping 7
[98241.490714] COUNTER_OVERHEAD: NITER == 64
[98241.490732] COUNTER_OVERHEAD: loop overhead is 834 cycles
[98241.490736] COUNTER_OVERHEAD: rdtsc cost is 60.9 cycles (4736 total)
[98241.490740] COUNTER_OVERHEAD: rdpmc cost is 89.0 cycles (6536 total)
[98241.490741] COUNTER_OVERHEAD: rdmsr (counter) cost is 270.1 cycles (18122 total)
[98241.490743] COUNTER_OVERHEAD: rdmsr (evntsel) cost is 279.6 cycles (18734 total)
[98241.490744] COUNTER_OVERHEAD: wrmsr (counter) cost is 455.0 cycles (29954 total)
[98241.490745] COUNTER_OVERHEAD: wrmsr (evntsel) cost is 415.5 cycles (27430 total)
[98241.490746] COUNTER_OVERHEAD: read cr4 cost is 21.5 cycles (2210 total)
[98241.490748] COUNTER_OVERHEAD: write cr4 cost is 315.5 cycles (21028 total)
[98241.490753] COUNTER_OVERHEAD: sync_core cost is 721.8 cycles (47030 total)
[98241.490759] COUNTER_OVERHEAD: read fixed_ctr0 cost is 87.3 cycles (6426 total)
[98241.490762] COUNTER_OVERHEAD: wrmsr fixed_ctr_ctrl cost is 445.5 cycles (29350 total)
```

▼図3　perfのカーネルインターフェースのシステムコール

```
int sys_perf_event_open(struct perf_event_attr *hw_event_uptr,
    pid_t pid, int cpu, int group_fd,
    unsigned long flags);
```

▼表1　pidとcpu

	pid == 0	pid > 0	pid < 0
cpu < 0	今のタスク(すべてのCPU上)	指定したタスク(すべてのCPU上)	無効
cpu >= 0	今のタスク(指定したCPU上)	指定したタスク(指定したCPU上)	すべてのタスク(指定したCPU上)

常に0にしておく必要があります。

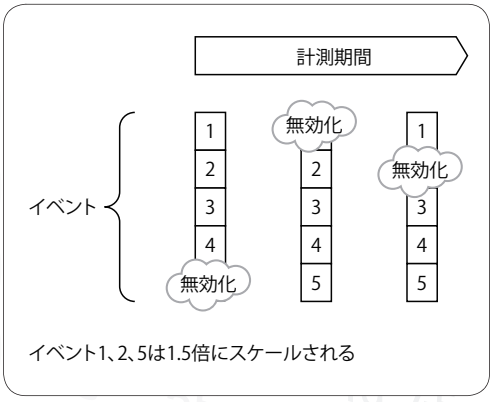
先ほどのハードウェアのパフォーマンスカウンタの項目で書いたように、ハードウェア上のPMUの数は限られています。たとえば、Intel Core i7 2xxxではコアあたり4つになっています。つまり、同時に計測できるハードウェアカウンタは(コアあたり)4つまでになります。

一方でperfのカーネルインターフェースはファイルデスクリプタの個数の上限にさえひっかからなければいくらでも多くのカウンタを計測できます。つまり、ここには1つトリックがあるというわけです。

perfのカーネルインターフェースは、登録されたイベントがハードウェアカウンタに収まりきれない場合、ラウンドロビンにイベントをハードウェアカウンタへと登録します。この場合、実際にカウンタに登録された期間と、計測が行われた期間とが取得できるので、計測値をスケールして計測期間中ずっとカウンタに登録されていたとしたらどの程度の値になっていたのかを計算できます(**図4**)。

もちろんこのようなスケールを行うと、正確さはある程度犠牲になってしまいます。とくに2つのイベントの比較を行いたいのに、片方がカウンタに登録されていて、もう一方が登録されていない期間があると、正確な比較はどうしても難しくなってしまいます。そこでイベントのグループ化を行います。今挙げた例のように、同時にハードウェアカウンタに登録されていてほしいものを1つのグループにまとめます。グループにまとめられたものは、同時に計測が行われるので計測値を問題なく比較できるようになるというわけです。

▼図4　ラウンドロビン上での計測値の予測



perf stat

では、最後にperfコマンドについて見ていきましょう。一番てっとりばやい使い方は“perf stat <コマンド>”とすることです。**図5**のように何もオプションを付けなくても、主要なイベント計測してコマンドの実行中での計測結果を表示してくれます。

“-eオプション”で観測したいイベントを指定できます。複数のイベントは“,”で区切って指定します。どのようなイベントが観測できるかは、“perf list”コマンドで見ることができます。イベント名に“:u”や“:k”を付けると、それぞれユーザ空間／カーネル空間のみの観測となります。イベントをどんどん追加していくと、**図6**のように右側に何%との表示が出て

▼図5　pref statの計測結果の例

```
$ sudo perf stat dd if=/dev/zero of=/dev/null count=1000000
1000000+0 records in
1000000+0 records out
512000000 bytes (512 MB) copied, 3.66726 s, 140 MB/s

 Performance counter stats for 'dd if=/dev/zero of=/dev/null count=1000000':

       3370.233208 task-clock                #    0.917 CPUs utilized
              461 context-switches          #    0.137 K/sec
               13 cpu-migrations            #    0.004 K/sec
              229 page-faults               #    0.068 K/sec
       2689019057 cycles                    #    0.798 GHz
       1545983395 stalled-cycles-frontend   #   57.49% frontend cycles idle
   <not supported> stalled-cycles-backend
       2918697869 instructions              #    1.09  insns per cycle
                                            #    0.53  stalled cycles per insn
        548553035 branches                  #  162.764 M/sec
          988847 branch-misses              #    0.18% of all branches

       3.673373164 seconds time elapsed
```

くるかと思います。これは先ほど紹介したとおり、観測対称が4つを超えてしまい、ハードウェアカウンタに入らなくなったためスケールした値になっていることと、どの程度スケールされたのかを示しています。

では、何か実際に比較してみましょう。**リスト2**のようなコードを準備します。foo()とbar()はbufへのアクセスのインデックスの順序が異なっている以外は同じです。この順序によってメモリアクセスの効率が変わってくるといいます。これをキャッシュミスの回数を見ることで比較してみましょう。

このコードを-DFOOと-DBARを付けてそれぞれコンパイルし、“perf stat -ecache-misses”をとってみます。

図7のように確かに-DBARのほうがキャッシュミスが大きくなっています。

▼図7　perf stat -ecache-missesの実行結果の比較

```
# -DFOOの場合
% ./perf stat -e cache-misses ~/prog

 Performance counter stats for '/home/naota/prog':

           306999 cache-misses

       0.067339607 seconds time elapsed

# -DBARの場合
% ./perf stat -e cache-misses ~/prog

 Performance counter stats for '/home/naota/prog':

         15460884 cache-misses

       0.243723796 seconds time elapsed
```

まとめ

今回はperfについて、CPUレベルからコマンドまで見ていきました。perfコマンドについてはまだまだ基本的な部分しか紹介できていません。ほかの機能についてもいつか紹介できればと思います。

◆

このたび思いがけず日本OSS奨励賞を受賞し、OSC東京2003で賞状をいただきました。日々オープンソース活動を続けていますが、こうして表彰されるとまた感慨深いものがあります。本連載も受賞の大きなバックアップになったと思います。この場を借りて、読者のみなさん、編集部のみなさんに感謝します。これからも開発／広報の両面でオープンソースに貢献していきたいと思います。カンファレンス／勉強会などで見かけたときは、ぜひお声がけください。SD

▼リスト2　キャッシュミスの比較

```
#include <assert.h>
#define SIZE 4096
unsigned char buf[SIZE][SIZE];
int i,j;
void foo()
{
#ifdef FOO
  for(i=0;i<SIZE;i++)
    for(j=0;j<SIZE;j++)
      assert(buf[i][j] == 0);
#endif
}
void bar()
{
#ifdef BAR
  for(i=0;i<SIZE;i++)
    for(j=0;j<SIZE;j++)
      assert(buf[j][i] == 0);
#endif
}
int main()
{
  int i,j;
  for(i=0;i<SIZE;i++)
    buf[i][0] = 0;
  foo(); bar();
  return 0;
}
```

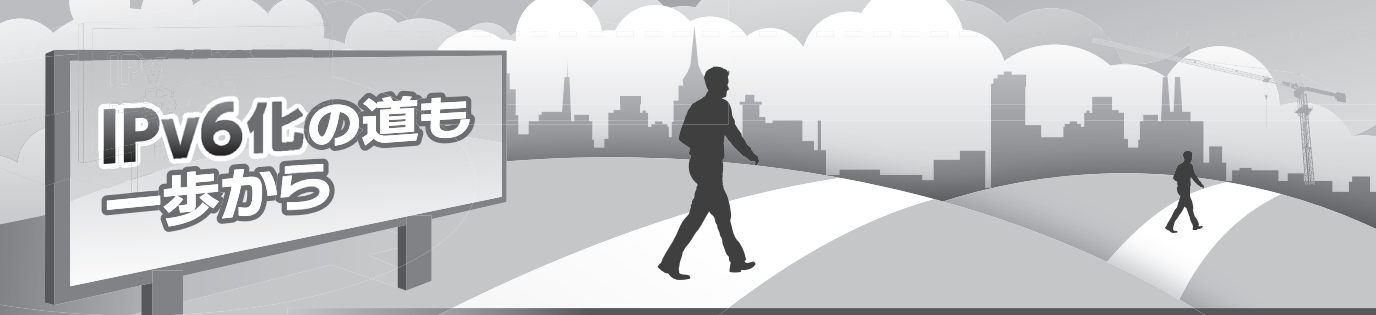
▼図6　イベントの観測結果の表示

```
$ sudo perf stat -e cycles:u,cycles:k,cycles,cycles,cycles dd if=/dev/z/null count=1000000
1000000+0 records in
1000000+0 records out
512000000 bytes (512 MB) copied, 2.06006 s, 249 MB/s

 Performance counter stats for 'dd if=/dev/zero of=/dev/null count=1000000':

        219622245 cycles:u                #    0.000 GHz                     [79.94%]
       1415005967 cycles:k                #    0.000 GHz                     [80.12%]
       1634769312 cycles                  #    0.000 GHz                     [80.12%]
       1634735060 cycles                  #    0.000 GHz                     [80.12%]
       1634741029 cycles                  #    0.000 GHz                     [79.71%]

      2.063599489 seconds time elapsed
```



第5回 アドレス計画時の注意点と落とし穴

IPv6普及・高度化推進協議会　IPv4/IPv6共存WG　アプリケーションのIPv6対応検討SWG
廣海 緑里 HIROMI Ruri　渡辺 露文 WATANABE Tsuyufumi　新 善文 ATARASHI Yoshifumi　藤崎 智宏 FUJISAKI Tomohiro

今回の想定

今回から構築の各段階での作業内容について触れていきます。今回はネットワーク構築について、注意点を中心に解説します。「ネットワーク構築」では、ネットワーク設計に基づいて機材を調達し、設定、テスト、導入という流れを想定して、話を進めます。

導入の基本的な考え方

本連載の第1回目（本誌2012年12月号）で導入の目的について、

(a) 顧客拡大のため（B2Cサイトや B2Bサイト、クラウドベンダ、データセンター事業者など）
(b) IPv6サイトを閲覧可能にする（ISP、社内インフラなど）
(c) IPv6アドレスのみの顧客／利用者とのメール送受信を可能にする（ISP、社内インフラなど）
(d) （他の目的で）新規にシステム／サイトを構築もしくはリニューアルするため、これを機にIPv6に対応させておく

などがあると書きました。

とくに公開サイトや外部とのメール転送のIPv6対応が、現在、最も対応が急がれる部分です。というのも、クライアントOSがIPv6に対応していて、すでにIPv6を使っているケースが増えているからです。今、新しいPC端末を購入すると、それはIPv6対応（デフォルトで有効）のものです。今回ははじめの一歩ということで、IPv6対応端末が導入されても困らないように、ネットワークの準備をします。具体的には、以下の3点を目標とします。

①IPv6の試験環境を作る
②既存のIPv4環境に、IPv6が有効になった端末を接続した場合の挙動を確認する
③IPv6が有効になった端末を接続するためのデュアルスタック環境を構築する準備を開始する

IPアドレスの選び方と設定

IPv6の疎通確認用のネットワークを作る場合に使えるIPv6アドレスの種類は3つあります（**表1**）。試験環境であれば、3つのいずれを使っても構築／検証できます。しかし、将来のネットワーク拡張や外部との接続などを考えると、最初からグローバルユニキャストアドレス（GUA）の運営ができるように備えるのがよいでしょう。

組織がGUAを取得するには、接続先のISPから調達する方法とレジストリ（日本ではJPNIC）からプロバイダ非依存のアドレスを調達する方法があります。いずれの場合も、調達コストがかかりますので注意が必要です。ちなみに筆者の所属する組織では、研究目的にプロバイダ非依存のIPv6アドレスの割り当てを受けています。

また、リンクローカルアドレスだけでルータ間のリンク（LLA）を構築、運用する方法とその

メリット／デメリットについて、draft-ietf-opsec-lla-only-03[注1]という文書が提出されています。今回は、ルータ間だけではなく、サーバやクライアントも接続するネットワークという点で参考にするには注意が必要です。ユニークローカルアドレス(ULA)についても、RFC 4864、RFC5375、RFC6204などの文書で企業網内での利用を推奨しています。IPv6で外部に接続しないのであれば、セキュリティ対策などでULAがより安全になります。

今回はGUAで運用する前提としますが、便宜上、ドキュメント記述用アドレス(2001:db8::/32)[注2]を用います。

まずは、1つのセグメントを作るところから始めます。1セグメントは/64としてサブネットを切り出します。最初のセグメントはシンプルに2001:0db8:0000:00001::/64(省略形は、2001:db8:0:1::/64)としてみます。

次に、アドレスを組織内でどう割り当てていくかを考えます。IPv4アドレスの割り当てと同じようにしたいという欲求も強いと思いますが、IPv4を忘れて一度まっさらな状況で整理したうえで検討してみましょう。また、これまで表計算ソフトウェアで管理していた場合には、これまで何をどのように管理していたのかを振り返りつつ、IPv4とIPv6の2つをうまく管理するための管理ツールの導入を検討してみてもよいかもしれません。

ただ、今回の想定では、小さな試験環境ですので、運用しながら規模が大きくなってきた場

注1) http://tools.ietf.org/html/draft-ietf-opsec-lla-only-03

注2) マニュアルや設定サンプルに記述する目的であらかじめ予約されているアドレス。実際の運用では使ってはいけないことになっている。

▼表1　疎通確認で使用できるIPv6アドレスの種類

アドレスの種類	説明	アドレス範囲	表記例
リンクローカルアドレス(LLA)	同一リンク(ネットワーク)内でのみ利用可能なアドレス	fe80::/10	fe80::6aef:bdff:fe61:4d13
ユニークローカルユニキャストアドレス(ULA)	IPv4のプライベートIPアドレスに相当するアドレス。運用事例が少ない。運用方法の議論中	fc00::/7	fd20:87ec:3bb5::/48
グローバルユニキャストアドレス(GUA)	IPv4のグローバルIPアドレスに相当するアドレス。一般的に利用されている。 PAアドレス……ISPから割り当てられる。/48などPIアドレス……レジストリ(JPNIC)から割り当てられる。/48より大きいブロックも申請可能	2000::/3	2001:db8::/32

COLUMN 利用アドレスに関するIPv4での実際の問題事例

IPv4アドレスでも、ドキュメント記述用アドレス(192.0.2.0/24)があります。しかし、多くの文献ではプライベートアドレスを用いて記述されていることが多い状況です。外部に接続しないイントラネットワーク内で使う分には対外的な害がないという点は、ドキュメント記述用アドレスもプライベートアドレスも同じです。ただし、ドキュメント記述用アドレスは使われないアドレスとしてフィルタ項目の中にプリセットされている機材やサービスもありますので、注意が必要です。

IPv4アドレスの運用で見かけるものとして、外部に経路を流さないからといって、グローバルなアドレススペースをレジストリ管理外で利用するケースがあります。内部利用のためのグローバルアドレスを使いながら、別のグローバルアドレスで外部接続を持つ場合があり、不必要なNATを導入したり、経路情報漏洩事故が発生したときの対処が必要になったりと不経済な運用につながります。新規導入するIPv6アドレスは、こういった運用を避けるようにアドレス設定を行いましょう。

合にはどのように管理するかを評価項目の1つとして検討してもよいでしょう。

サーバ／クライアントのアドレス設定方針

サーバやルータなど運用上変化しないアドレスが付与されていたほうがいいものには、静的にアドレス設定します。クライアント端末など変化してもよいものには、アドレス自動設定のしくみを利用します。自動設定を利用すると、ULAとGUAを共存させる場合やリナンバリングする場合、ISPからも割り当てを受けてマルチプレフィックス環境にする場合などに作業が楽になります。

また、アドレス自動設定のしくみを使う場合、ステートフルかステートレスで行うか、下位64ビットにはMACアドレス由来のIDを使うか、一時アドレスを利用するかといった方針を決める必要があります。

IPv6アドレス自動設定のしくみ

IPv6のアドレス自動設定は、サーバなどで端末の状況を管理するステートフルか、管理をしないステートレスかの2つに大別されます。IPv4のDHCPでの運用管理の踏襲や、昨今のセキュリティ管理強化の流れでは、ステートフルを選択したいと思うところかもしれません。IPv6では近隣探索（以降、NDP）というプロトコルとICMPv6メッセージが大活躍します。DHCPサーバでネットワーク上の端末が利用しているアドレスを管理するのが目的なら、NDPのアドレス解決、重複アドレス検出（DAD）、近隣到達不能検出（NUD）といった機能をうまく使うことで、DHCPサーバ（のログ）に頼らずに実現できます。また、サーバも1台減らすことが可能になるかもしれません。今回はSLAACと呼ばれるアドレス自動設定を取り上げて説明します。

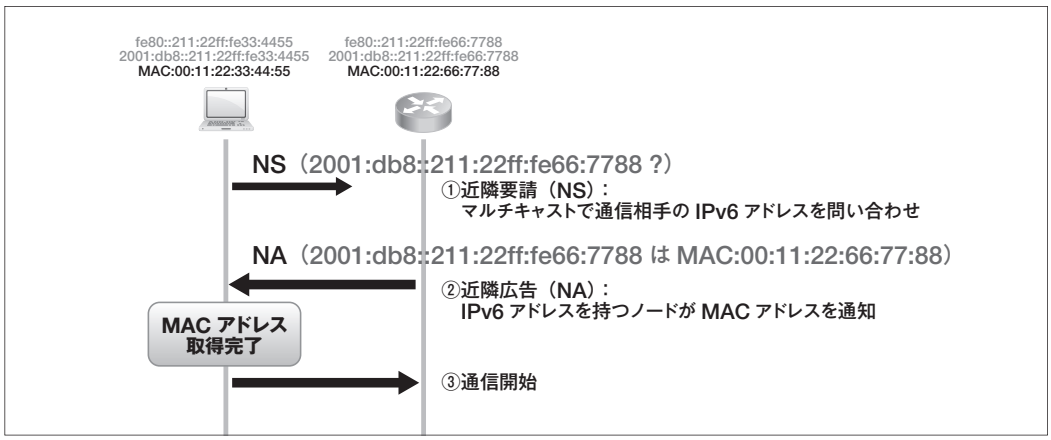
NDPはIPv6の最も基本と言えるプロトコルです。**図1**は、通信相手のリンクレイヤアドレス（MACアドレス）を取得する様子です。NS（近隣要請）メッセージをマルチキャストアドレス宛に送ると、該当する端末がNA（近隣広告）メッセージを送って通知します。続く**図2**は、アドレスプレフィックスやデフォルトルータの情報を得る様子です。ここでやりとりされるパラメータによって、DHCPサーバを使うかどうかがわかります。ルータに対してRS（ルータ要請）メッセージが送信され、ルータからRA（ルータ広告）を得ます。図2のRAメッセージでは、O flag=1（Other Configuration flag）がセットされていて、DNSサーバなどのサーバ情報がDHCPv6で取得できることを示しています。ここまでがNDPの基本動作です。NDPでできることを**表2**にまとめました。

図3は、RAで受け取ったプレフィックスと端

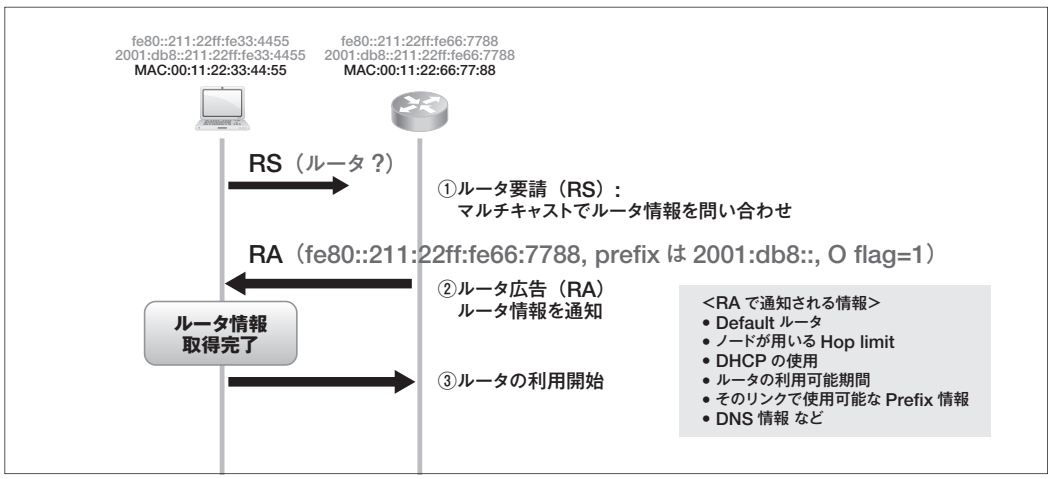
▼表2　近隣探索（NDP）の機能

機能	概要
ルータ探索（Router Discovery）	ホストが同一リンク上のルータを探索する
プレフィックス探索（Prefix Discovery）	プレフィックスを得る
パラメータ探索（Parameter Discovery）	リンクMTUなどの情報を得る
アドレス自動設定（Address Autoconfiguration）	インターフェースに自動的にIPv6アドレスを設定する
アドレス解決（Address resolution）	通信相手のIPv6アドレスに対応するリンク層アドレスを調べる
次ホップ決定（Next-hop determination）	あるパケットの終点に対する、転送先ノードのIPv6アドレスを得る
近隣到達不能検出（Neighbor Unreachability Detection）	近隣ノードへ到達できないことを確認する
重複アドレス検出（Duplicate Address Detection）	同一リンク上の同じユニキャストアドレスを持つノードを検出する
リダイレクト（Redirect）	ルータがもっとよい次ホップをホストへ通知するためのNDPメッセージ

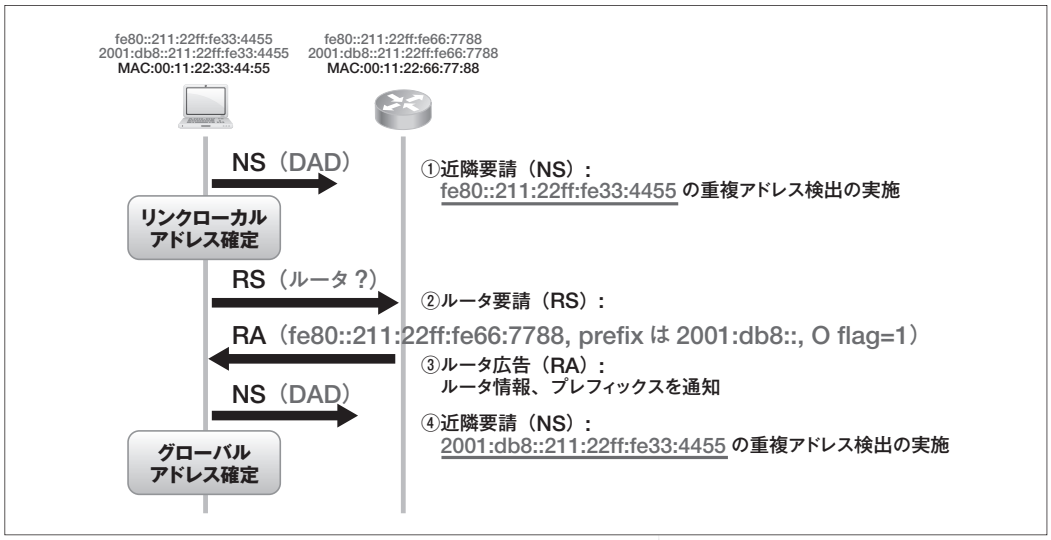
▼図1　近隣要請（NS）と近隣広告（NA）リンクレイヤアドレス解決



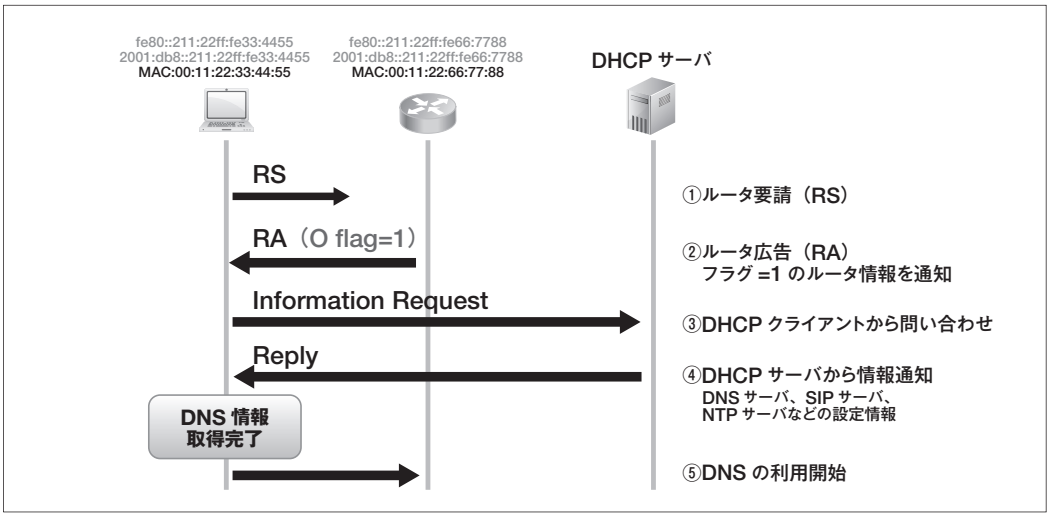
▼図2　ルータ要請（RS）とルータ広告（RA）によるルータ情報の通知



▼図3　SLAACによる端末のIPv6アドレス自動設定



▼図4　DHCPv6を用いたDNSサーバ情報の取得



▼表3　端末の主な設定方法の比較

設定方法	関連RFC	IPアドレスの設定	DNSとサーチリストの伝達	デフォルトルータの伝達	補足
手動	4291	—	—	—	GUIではIPv6アドレスの登録ができないOSがある
SLAAC	4861、4862	○	×	○	多くの端末OSで実装済み
DHCPv6（ステートレス） ※アドレス以外の情報取得	3315、3736、3646	×	○	×	DNSサーバとサーチリストオプションをDHCPサーバから伝達。DHCPv6クライアント未実装の端末OSがある
SLAAC+DHCPv6	4861、4862、3315、3736、3646	○	○	○	RA（Oフラグ=1）とする
SLAAC+RDNSS、DNSSL	6106	○	○	○	RDNSSやDNSSLオプションに対応していないルータや端末OSがある
DHCPv6（ステートフル） ※アドレス情報など	3315、3736、3646他	○	○	×	RA（Mフラグ=1）とする。DHCPv6クライアント未実装の端末OSがある
リンクローカル	4862	○	×	○	NDPを使わず固定設定する端末OSがある

末のインターフェースIDからIPv6アドレスを合成し、DAD（重複アドレス検出）を行って、他に同じアドレスを使っている端末がないことを確認して、利用開始に至る様子です。**図4**では、RAのO flag=1を受け取って、DHCPクライアントを起動してDHCPサーバにInformation Requestを出してReplyを得る様子です。これで、IPv6の通信開始に必要な基本情報がそろいました。

表3は自動設定の方法ごとに、DHCPサーバを使うか、DNSやデフォルトルータの情報伝達はどうなるか、を整理したものです。

今回は、以下の方針で進めます。

- サーバ類には固定のアドレスを割り当て、DNSにも登録する
- サーバなど固定で割り当てる場合のアドレス割り当てルールや利用する範囲を決めておく
- クライアント端末にはアドレス自動設定で動的にアドレスを割り当てる

- クライアント端末の下位64ビットはMACアドレス由来のEUI-64形式を用いる

端末に付与するIPv6アドレスは、グローバルルーティングID部には2001:db8:0:1::/64、下位64ビットにMACアドレス由来のmodified EUI形式のインターフェースIDを使って、IPv6のステートレスアドレス自動設定(SLAAC)を利用することにします。これでアドレス設定の基本計画は完了です。

機材の選び方と設定

ここで機材の選び方について考えてみます。まず、IPv6をサポートしているOSであることが第1条件としてあります。これはアドレス自動設定が可能であること、静的な設定を行えること、疎通確認のためのコマンドが整備されていること、設定確認ができることなどが挙げられます。

クライアントOSについては、IPv6対応のものが多くあります。ただし、iOSやAndroid、CentOSなどのLinux系OSではDHCPv6クライアントの実装がされていない場合があり、DHCPv6を利用する場合には注意が必要です。また、GUIでは設定の確認ができないものや、バージョンによってpingなどの確認コマンドがないものもあります。OSにコマンドがなくても、アプリケーションをインストールすれば大丈夫な場合もあります。

ルータに関しては、少し前までは、「IPv6対応」という名のもとに「IPv6パススルー機能」(L2のブリッジ機能)を搭載したものが多くありました(現在もあります)。これらの「IPv6対応」というのは、NTT東西の提供するIPv6閉域網サービスへのアクセスを提供するために開発されており、IPv6によるグローバルインターネットサービスへ接続できない場合があります。IPv6アドレスを設定したり、経路情報を交換したり、パケットを転送するには、「IPv6パススルー機能」のルータは使えないので、注意が必要です。

正式なIPv6対応かどうかの1つの選定基準としては、「IPv6 Ready Logo」を取得しているものが指標として使えます。財団法人電気通信端末機器審査協会(JATE)が実施している「IPv6 Ready Logo認証」プログラムは世界共通の認証プログラムで、IPv6に必要とされる機能を満たしていることを証明しています。JATEのWebサイトで、認定された機材のリストが公開されています[注3]。

注3) http://ipv6.jate.jp/approved_list

COLUMN ULAを使う場合

ULAには、グローバルルーティングプレフィックス(/48)の算出をするための計算ルールがあります。ULAは、MACアドレス、EUI64アドレス、時刻データを用いてRFCに規定されるアルゴリズムから計算されます。追加のULAを作成する場合に同じMACアドレスを用いても、時刻のパラメータの値が変わるため、重複することは理論上ありません。算出した/48のプレフィックスからは、/64サイズで16bit分(65536)のサブネットが得られます。このPseudo-Randomアルゴリズムについて正確に知りたい場合は、RFC4193の3.2.2章にサンプルコードがあります。もっと簡単に計算結果だけ知りたいという人は、インターネット上の公開サイトを利用することもできます。信頼できるサイトをいくつか挙げておきます。

①KAMEプロジェクト
http://www.kame.net/~suz/gen-ula.html
②SixxSプロジェクト
http://www.sixxs.net/tools/grh/ula/

最新のクライアントOSの多くはIPv6がデフォルトで有効になって出荷されているので、IPv6の運用がされているネットワークに接続するとすぐに利用できます。企業向けルータなどについては、機能は搭載済みで、設定するとすぐに使えるものが増えてきています。

ルータ、クライアント、サーバの設定の参考資料としては、これまで何度か取り上げているIPv4アドレス枯渇対応タスクフォースが公開している文書があります[注4]。ルータベンダーのWebサイトで、通信事業者やネットワークの種類ごとにお勧めの設定を掲載している場合もあります。

注4） http://www.kokatsu.jp/blog/ipv4/data/user.html

試験環境の構成

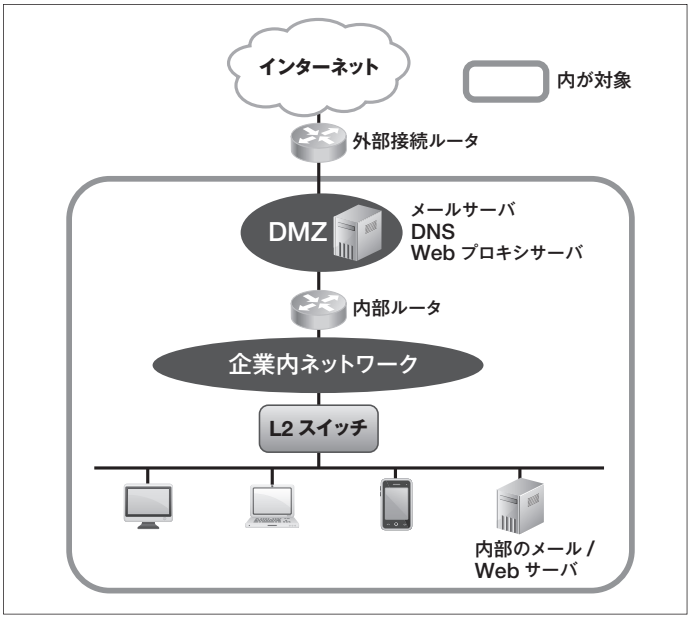
今回はクライアントとしておもにWindows 7を例にとります。IPv6はデフォルトONですので、そのまま接続するだけです。サーバとルータには、自身に固定のIPv6アドレスを手動で設定します。ルータには、同じセグメントに接続する端末にアドレス自動設定を行うための設定を行います。サーバはDNSとWebサーバを立ち上げて、クライアントから情報を見られるかどうか確認してみましょう。これらの情報を**図5**と**表4**に整理しておきます。

終わりに

今回は、おもにアドレス設定について解説しました。利用できるアドレスや設定方法がいろいろあって、ややこしいと思われたかもしれません。組織のネットワークにとっては、構築方法の選択肢が複数あって柔軟性があるとも言えます。自分の組織ならどうするか、ぜひ検討してみてください。

次回は、今回整理した情報をもとにIPv6の検証環境を作って、最初に挙げた3つの目標を達成させたいと思います。お楽しみに。SD

▼図5　試験環境構成図



▼表4　試験環境のGUAアドレス設計の整理

	アドレス	割り当て	設定方法
ネットワーク	(サーバセグメント)2001:db8:0:1::/64 (クライアントセグメント)2001:db8:0:2::/64	―	―
ルータ	(サーバセグメント)2001:db8:0:1::1/64 (クライアントセグメント)2001:db8:0:2::1/64	固定割り当て	手動設定
サーバ	2001:db8:0:1::10	固定割り当て	手動設定
クライアント	2001:db8:0:2:fffe:[EUI64アドレス]	動的割り当て	自動設定

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3月号

第1特集
オープン環境でスキルアップ!
もっとクラウドを活用してみませんか?

第2特集
光、ギガビット、高速ネットワークを体験!
実践! ワイヤリングの教科書

一般記事
・「SSDストレージ」爆発的普及の理由

1,280円

2013年2月号

第1特集
UNIXコマンド、fork、pipeを復習し、高度なスクリプティングへ
シェルスクリプティング道場

第2特集
忙しいITエンジニアのための
超効率的勉強法

一般記事
・Samba 4.0.0ファーストインプレッション

1,280円

2013年1月号

第1特集
いざというときに備える
システムバックアップ

第2特集
IT業界のキーパーソンに聞く
2013年に来そうな「技術」・「ビジョン」はこれだ!

特別付録
法輪寺電電宮情報安全護符シール

1,380円

2012年12月号

第1特集
判断をあおぐ／経緯を説明する／手順の理解を得る文章を書くためのアタマの整理術
なぜエンジニアは文章が下手なのか?

第2特集
高速・高機能HTTPサーバ
Nginx構築・設定マニュアル

一般記事
・エブリデープログラマの発想と実践

1,280円

2012年11月号

第1特集
もし、OpenFlowでやれと言われたら?
SDN、仮想化でネットワークはどうなる

第2特集
サーバの運用支援に
グラフィカルなリソース監視ツールを!
Muninが手放せない理由

一般記事
・SkeedSilverBulletとは?

1,280円

2012年10月号

第1特集
サーバ管理自動化の恩恵とリスクを見直しませんか?
Chef入門

第2特集
lprコマンドが動く裏側のしくみがわかる!
Linuxプリント環境の教科書

一般記事
・SSH力をつけよう!
・JSX入門［前編］

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。

April 2013

NO.18

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

ITをより良いものに —労働面も技術面も—

今回は、昨年12月に福岡で、そして今年の2月に浜松で行った研究会の模様をお伝えします。

福岡大会

■IT業界を良くする方法
【日時】2012年12月8日（土）13:00～13:45
【会場】KCS福岡情報専門学校 7階 708教室
【講師】小室 文
（サーバーワークス/JAWS-UGクラウド女子会）
【司会】法林 浩之（日本UNIXユーザ会）

▲ 写真1　小室文氏

毎年、12月に福岡で行われるオープンソースカンファレンスの中でjus研究会を実施しています。今年は福岡を拠点に活動されている小室さん（**写真1**）を講師にお迎えし、IT業界における働き方についての考察をお聞きするとともに、どうすれば良くなるのかについて議論しました。参加者は21人でした。

■男性の就業環境改善、そして女性の労働参加へ

はじめに、ITエンジニアの労働状況に関する資料の紹介と、小室さんの考える改善策が提示されました。IT業界の仕事は3K（きつい、厳しい、帰れない）と言われますが、これを是正するにはまず男性の就業環境を改善すべきであるというのが小室さんの意見です。日本では、育児は女性、労働は男性という意識が根強く、30代女性の労働者率が20代や40代に比べて低いという統計があります。しかし、その労働形態は、男性にとっては家族を養っていくのは自分しかいないという責任感に追われてしまい、精神的にも良くない状態に陥りやすいです。男性にも育児休暇を認めれば家庭や自分に時間を割くことができるようになり、それが女性の負担を軽減することにもつながり、結果として女性も仕事に力を入れることが可能になるでしょう。

このような考えに基づいて就業環境を工夫している例としては、男性の育児休暇を認める（サイボウズなど）、週2～3日は定時で強制的に帰らせる（ワコールなど）、オフィスに子供が遊べる場所や保育園などを作る（サマンサタバサなど）、などがあります。VPNなどを用いて家でも仕事ができるようにするのも工夫の1つと言えるでしょう。また、こうした動きを推進するには行政の協力も必要で、たとえば時短をきちんと取得した個人や会社にインセンティブを出すなどの特典を設けてほしいとか、労働者側も特技を持つとか人のつながりを作るなど、人として力をつける必要があるという見解も述べられました。

■参加者それぞれが考える改善案

この後、講師からの提案で、参加者を3つのグループに分け、IT業界を良くするにはどうすればよいかについてグループディスカッションを行いました。最後に各グループから議論の内容が報告されましたが、それによると、部署の単位を小さくするこ

とで個人間のコミュニケーションの密度を上げる、人を増やして各自の仕事量を減らす、IT業界の人でもITの正しい活用法ができていない場合が多いので、正しく活用できるようレベルの底上げをする、といった意見が出たようです。

今回の研究会はいわゆる技術的な話ではありませんが、ITという世界で仕事をしていくうえでは大切なテーマだと思います。研究会としては珍しく参加者同士で議論する時間も持つことができ、有意義なセッションになりました。

浜松大会

■LANの見える化
【日時】2013年2月9日（土）14:30～14:45
【会場】浜松市市民協働センター
【講師】牧野 秀彦（ヤマハ）
【司会】法林 浩之（日本UNIXユーザ会）

2月もオープンソースカンファレンスの中でjus研究会を行いましたが、静岡県でのjus研究会は初開催となりました。講師として浜松市に本社のあるヤマハから牧野さん（**写真2**）をお迎えし、ヤマハのルータや管理ツールを使ってLANの状態を可視化する方法について発表していただきました。展示会場の中にセミナーブースが設けられていたこともあって観覧者が多く、60名の方にご覧いただきました。

■複雑なケーブル配線は配線図で表示

講演は、L2スイッチを使ったLAN配線の見える化と、無線LANアクセスポイントを使用するときの電波状態の見える化の2点が主な内容でした。前半はLAN配線の見える化に関する取り組みです。ヤマハのネットワーク機器は小規模LANでの利用が多いのですが、そこでよく目にする問題として、配線の複雑化による配線ミスやケーブル抜けがあります。これらの問題を引き起こす要因の1つに、ネットワークがどんな状態なのか、何が起きているのかが目に見えないというのがあります。そこでヤマハのL2スイッチSWX2200シリーズには、ルータを介して遠隔からスイッチを監視／管理するしくみが用意されています。ルータの管理画面にはLANの配線図が表示され、接続されている各機器の稼動状態を見ることができます。また、正常な状態をスナップショットとして記録しておき、現在の状態との差分を示すことでトラブルの原因を推測できる機能もあります。


▲ 写真2　牧野秀彦氏

■電波の混雑状況はグラフで表示

後半は無線LANの見える化についてです。最近はとくに2.4GHz帯を使用する機器が増加し、電波干渉により無線LANが十分に使えないという話をよく耳にします。ほかにも、不正なアクセスポイントの存在、持ち込みモバイルルータの増加なども影響しています。そこでヤマハの無線LANアクセスポイントWLX302には、電波状態をグラフで表示する機能が搭載されています。横軸に周波数、縦軸に電波強度が表示され、どれぐらい混雑しているかが一目瞭然でわかります。また、信号強度やエラー率などをもとに混雑状況を判定して色分け表示したり、電波状態のスナップショットを保存するなどの機能により、障害発生時の原因調査もしやすくなっています。そして最後に、見える化の本質は可視化ではなく、次のアクションを決めるための情報を見やすいかたちで出力することであり、見える化を進めることでネットワーク管理業務の効率化に貢献していきたいというコメントがありました。

持ち時間が15分という短い研究会になりましたが、発表に続いて展示ブースで実演を見ることにより、内容をさらに深く理解することができました。SD

Hack For Japan
エンジニアだからこそできる復興への一歩

Hack For Japan

第16回 OpenDataとハッカソンで変わる世界

“東日本大震災に対し、自分たちの開発スキルを役立てたい”というエンジニアの声をもとに発足された「Hack For Japan」。本コミュニティによるアイデアソンやハッカソンといった活動で集められたIT業界の有志たちによる知恵の数々を紹介します。

● Hack For Japanスタッフ
及川 卓也　OIKAWA Takuya
Twitter @takoratta
高橋 憲一　TAKAHASHI Kenichi
Twitter @ken1_taka

Developers Summit 2013 パネルディスカッション

2013年2月14日と15日に翔泳社の主催による「Developers Summit 2013」(以下、デブサミ)で「OpenDataとハッカソンで変わる世界」というセッションが行われました[注1]。Hack For Japanからは OpenDataを積極的に推進している関治之がモデレータとして、また、及川卓也がパネリストとして登壇しました。他のパネリストの方々は次のとおりです。

- 渡邉 英徳(首都大学東京)
- 東 富彦(国際社会経済研究所, Open Knowledge Foundation)
- 中井 康裕(経済産業省)

※本文中敬称略

まずは自己紹介も兼ねて各登壇者からそれぞれの取り組みについて話がありました。

OpenDataの拡大とアプリによるフィードバック── 関氏

このセッションを通じて、OpenDataとハッカソンでどうやって世の中を変えていくのか、ということを伝えていきたいと考えています。

OpenDataとは、政府や自治体などが持つ公共データをオープンに公開、活用することで、より良いアプリケーションを生み出すムーブメントです。OpenDataが日本でも広がるには、良いアプリケーションが必要です。良いアプリケーションやアイデアは、データの提供側にポジティブなフィードバックを与えます。Hack For Japanでは、OpenDataをテーマにしたハッカソンを何度も開催してきました。OpenDataハッカソンで世界を変える。そのためにはエンジニアが必要なのです。


関 治之氏(Hack For Japan)

これまでHack For Japanで開催してきたハッカソンには、エンジニア以外の人も参加しています。社会的課題をテクノロジーで解決するという観点で考えると、OpenDataとハッカソンの相性はとても良いと思います。

project311に見るビッグデータ活用の有効性── 渡邉氏

ヒロシマ・アーカイブの作成に関わっています。福島原発からの距離の同心円のマッピングは、Twitterで呼びかけたところ2時間ほどでできました。これは当時1週間で212万回の表示がされています。

昨年GoogleとTwitterが共同開催した「project311」という震災時のビッグデータへの取り組みがあり、東大の早野先生の伝により集められたデータによってできた成果をspeedi.mapping.jpで見ることができます。これを見ると放射性ヨウ素が南に向かって飛んでいたということがわかり、もし

注1) スライドはこちらにあります。URL http://www.slideshare.net/hal_sk/open-data-16547949

これらのデータが最初からオープンになっていたら被爆量を減らすことができたのではないかと考えられます。

Hack For Japanの活動をふりかえって── 及川氏

Hack For Japanは震災直後からハッカソンをITでの復興支援手段と位置づけています。これまでに被災地を含む全国各地でハッカソンを開催し、多くの人に参加していただく中で、常に自分たちに問いかけている重い言葉があります。

それは、「果たしてハッカソンは本当に有効だったのか？」というものです。

これは別の言い方をすると、「もう一度、同じことが起きたときにハッカソンという手法を使うだろうか？」という質問に言い換えられます。実際、スタッフミーティングで私たちはこのことを真剣に考えました。

今回の震災でHack For Japanの活動を通じてさまざまなことを学んだ我々が、今度もし別の地域で自然災害などがあった場合、ハッカソンという方法を使って復興支援を行おうと考えるだろうかという問いが、ハッカソンに対しての振り返りをするには必要です。活動当初よりさまざまな人に、IT支援と口で言うのは容易(たやす)いが、それを完遂するのは難しいと言われてきました。多くの人が瓦礫処理をしている横で、スマートフォンで、タブレットで、デジタルカメラを抱えて自分が何をできるかを考えなければいけません。

IT支援というのはそれほどまでに、成果を生み出すことが難しいものです。

さらに、ハッカソンを開催する際に、それを声高には叫ばなかったにせよ、少なくとも心の中では「このイベントを楽しんでほしい」というメッセージを常に持っていました。それは、ハッカソンというものがそもそもは楽しむイベントであるからです。技術者が集まり、あるテーマを元に限られた時間で腕を競う。それがハッカソンです。基本的には楽しいイベントです。果たして、この「楽しむ」というイベントが震災復興支援の手段として適していたのかどうかは反省しなければなりません。

そのような「楽しむ」というハッカソンの精神は残したまま震災復興支援活動を行いたいと考えていたのですが、このように自己矛盾を抱えたままで実施したためか、今に至るまでその注目度と比較すると、目に見える成果はあまり多くないと自分たちでも認めざるを得ません。しかし、それでも、ハッカソンという手法は条件をきちんと整えたならば、震災復興支援という活動にも有効だと考えます。

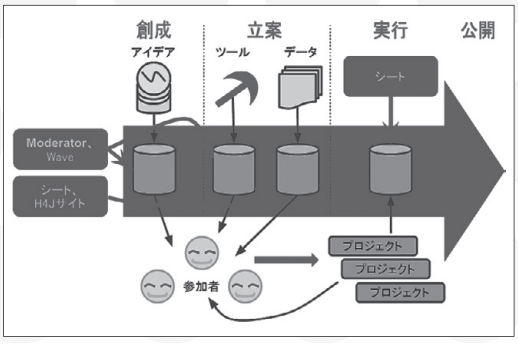
図1は2011年の4月にはスタッフで共有していたもので、当初のHack For Japanのサイトにも掲載していたものです。この図には、プロジェクトを遂行するに際しての必要な要素である、プロジェクトメンバー、技術(データとツール)、そして実際にそれが使われるための被災地との連携が描かれています。ハッカソン成功に必要な条件とは、実は当初考えていた、この図に描かれていたのです。

Hack For Japanのハッカソンには熱意のある人が多く集まってくれましたが、プロジェクトによっては必要なメンバーが欠けていることがありました。デザインができる人がいなかったり、プロジェクト・マネージメントの経験のある人が不足していたり。そのようなメンバーのマッチングを行うことも本来、スタッフ側でもう少し考慮すべきでした。

必要な技術面のサポートについては、クラウドベンダーを始めとして、実は多くのベンダーから無償サポートの申し出を当初からいただいていました。それを十分に使い切れなかったということも反省点として挙げられます。

また、途中から被災地でのハッカソンを行うなど

◆図1　Hack For Japanの全体像



して実際のユーザに対する意識を強めてはいたのですが、未だに被災地が必要なものを組み上げるようなベストなアプローチを提示できていません。実際のユーザの声を聴く手段を用意すべきでした。

Hack For Japanの活動と並行して、ほかの団体のハッカソンや同様のイベントに参加しましたが、そこからも多くを学びました。その1つにメンタリング[注2]の必要性があります。「Startup Weekend」という活動[注3]は、2日半でスタートアップのシミュレーションを行うことができるものですが、ここでは2日目の午後にメンタリングテーブルと言って、さまざまな専門家に相談ができる場が提供されています。法律の専門家、ベンチャーキャピタル、技術の専門家などがボランティアベースで相談に乗ります。Hack For Japanのスタッフ間でもこのようなメンターの必要性は議論されたことがあったのですが、実現には至りませんでした。

そして何よりも、データの必要性。とくに当初のハッカソンでは、必要なデータがないために開発できなかったものが多くあります。今日のテーマともつながりますが、OpenDataが用意されつつある今ならば、ハッカソンはより有効であるはずです。

整理すると、データ（OpenData）、メンター、そしてユーザの声。これらをきちんと整えることでハッカソンは実社会に役立つものを開発する手段となりえます。

OpenData活用の3つのポイント── 東氏

NPO法人のETIC.[注4]などと一緒に社会起業の活動をしています。Open Knowledge Foundationの日本支部[注5]（本部はイギリス）もやっており、今年初めくらいにようやく機能してきました。

OpenDataの3つの目的に沿って海外の事例を紹介します。

①政府の透明化

「Ad Hawk：政治広告の監視」

テレビやラジオの音をスマホで録ってサーバにあげると、それが政治広告かどうか判定して返してくれる

②公共サービス向上

「Love Lewisham：市民参加型環境向上」

不法投棄されたマットレスを発見した市民からのレポートで、当局が対処中かどうかわかるようになっている

③経済活性化

「Total Weather Insurance：農家向け収入保障保険」

“データを公開するのはいいが、ビジネス的にどうなんだ?”という問いは必ず出てくるが、その答えのひとつ。元Google社員が起こした会社ということもあってシミュレーションの仕方が半端ないレベルで行われており、10兆地点の気象シミュレーションポイントを元にしているとのこと

政策としてのOpenData── 中井氏

経済産業省の情報プロジェクト室に所属しており、G空間[注6]情報の活用や、使いやすいデータをどうやって出していくのかなどを進めています。

OpenDataへの取り組み状況

昨年3月から政府全体でOpenDataに取り組むことになりました。個人情報の問題もあるのでできるところからという状況です。また、公共データWGの中で経産省の持っているデータを試行的に出してみて、これを政府全体の話に持って行くための試行的取り組みもしております。昨年12月には電子行政OpenData実務者会議を設置し、2013年度以降どのように進めていくかロードマップを作成中です。

ほかにもOpen Data METI[注7]サイトを設置して、Creative Commonsによってデータの再利用をしや

注2）助言者（メンター）と対話を行うことで、気づきと助言による被育成者本人の自発的・自律的な発達を促すという、人の育成・指導方法のひとつ。
注3）URL http://startupweekend.org/
注4）URL http://www.etic.or.jp/
注5）OKFJ URL http://okfn.jp/
注6）政府が打ち出した次世代の地図や位置連動システムを表現するための言葉。
注7）DATA METI構想：公共データの提供、技術や制度の検討、ポータルサイトの構築、ユースケースの創成と共有、国民や事業者による利活用、ニーズや課題の把握、というサイクルをできる限り短いサイクルで回す。

すくし、DATA METI活用パートナーズの募集、実際にデータを活用できる人を募集しています。執筆時点の2013年2月20日現在、オープンデータアイデアボックス[注8]には、292名の方が登録、83のアイデアが登録されています。

OpenDataはどうあるべきか ── パネルディスカッション ──

5人の登壇者の話が一通り終わった後、Hack For Japan 関のモデレートでパネルディスカッションへと移りました。

◉◉◉

関:データを出す側の話をもっと聴きたいのですが、経産省としてはどういう計画があるのでしょうか。

中井:PDF形式になっているものを機械可読可能な状態、検索可能な形にしていきたいと考えています。中には法律、条例などで公開が難しいものもあります。

関:どの組織がどういうデータを持っているのかというのはまとまっているのでしょうか。

中井:正確に把握している人はほとんどいないのが実情です。まずは棚卸しするところからです。省庁のほかに、独立行政法人もその対象として考えています。

関:我々が行うハッカソンからのアウトプットは役に立つものでしょうか。

中井:もちろん役に立ちます。あるとないとでは大きく違います。

及川:OpenDataと言った場合の、「Open」と「Data」を分けて考えてみると良いのではないかと思います。つまり、「Closed」な環境でさえ「Data」が使えるようになっていなかったならば、「Open」にしようとすることは無理なのではないか、もしくは厳しい言い方かもしれませんが、使えないデータを公開するだけのことになってしまわないかと思います。

このように考える背景には、昨年に行った復旧復興支援制度データベースAPIハッカソンでの気付きがあります。このハッカソンでは、すでに用意されているAPIを利用したのですが、APIが残念ながら使い物にならなかったのです。多くのエンジニアが集まったにもかかわらず、1日のハッカソンで完成したアプリケーションは皆無でした。たとえば、データのフォーマットがまちまちであったり、必要なAPIが用意されていないなど、実際に使うという観点で見た場合に足りないものだらけでした。半角や全角が統一されていなかったり、日付も西暦や元号が混ざっていたりしました。

関:やはり生まれたものを育てていくプロセスが重要だと思うのですが、我々のハッカソンでアイデアは良いけどハッカソン終了後に続かなかったものがいくつかあります。

東:ハッカソンはとてもいいのですが、プロジェクトに命をかけるほど深刻な課題を抱えている人がもっと必要なのではないでしょうか。参加者に当事者意識が弱いと、終わってしまえば人ごとになってしまいます。各地域で、本当に困っている人たちが集まってハッカソンをやると良いと思います。それを解決するにはどうすれば良いか自分たちで考えるはずです。

関:継続して進めていくには、いかに自分ごととしてかかわれる人がプロジェクトの中心にいるか、ということが重要だと思います。

最後に

この号が出る頃には終了していますが、2月23日には「International Open Data Day in Japan」も開催されます。Hack For Japanでもこのイベントに協力していますので、本連載でもレポートする予定です。

最後に、このセッションは次の言葉で締めくくられました。

世界をよりよい場所に変えるのは、あなたです!

SD

注8) URL http://opendata.openlabs.go.jp/

温故知新
IT むかしばなし

第21回

第5世代コンピュータとProlog

たけおかしょうぞう　TAKEOKA Shouzou　take@takeoka.net

はじめに

今回は、1972年に登場した非手続き型プログラミング言語の“Prolog”と1982～1992年の国家プロジェクト“第5世代コンピュータ”を振り返ります。

Prologをご存じですか?

日本はProlog大国です。Prologというのは、イギリスのエジンバラで考案された論理型プログラミング言語です。計算の原理は、定理証明と同じしくみで「レゾリューション」と呼ばれています。その1ステップは、三段論法を1つ行うのと同じです。違う見方をすると、「物事の関係を論理式で書き表しておくと、それをプログラムとして実行できる」という、ほかのパラダイムでは表現も実現もできないことを行っています。

ソクラテスは死ぬか?

Prologは、ユニフィケーションという、双方向代入を備えたパターンマッチングと、バックトラックという2つの独特なメカニズムを備え、そのプログラム実行は、ほかのプログラミング言語からは想像しにくい振る舞いをします。次のプログラム例と結果をご覧ください。

```
human(ソクラテス).
mortal(*X) :- human(*X).

?- mortal(ソクラテス).
yes.
```

この上の2行がプログラムで、「human(ソクラテス).」という行は“ソクラテスはhumanである”という事実を記述しており、「motal(*X) :- human(*X).」[注1]という行はhumanである*X（*Xは変数）はmortal[注2]である、という関係を記述しています[注3]。

「?-」は対話プロンプトであり、プロンプトの後ろが、ユーザによる問い合わせです。ユーザから問い合わせを入力されると、Prolog処理系は実行を開始します。

Prologは、現在、並列向き言語として人気のあるErlangの先祖でもあり、そのパターンマッチング機構はPrologから受け継いでいます。

Prologマシンの数々

Prologは、関数型言語マニアが推す単一代入（変数に一度だけしか代入できない）であり、副作用がなく、並列処理記述に向いていると考えられました。1980年代後半、日本の第5世代コンピュータプロジェクトでは、ICOT（Institute for new generation COmputer Technology）という組織を作り、並列な知識処理マシンを開発します。しかも、ある種の機械は、当時のプログラマの憧れ、ワークステーションとして1人1台のためのものとして実現されました（当時、UNIX系ではΣプロジェクトというものもあり、1人で高性能マシンを占有するというエンジニアリングワークステーションは、強く必要とされていました）。個人用Prologワークステーションは、PSI（プサイ）として量産されました（商品版はMELCOM PSI[注4]）。また、ICOTでは大型Prologマシンとして、CHIも開発されました。日本では、

注1) ":-"は←の意味。

注2) 死を免れないという意味の英語。

注3) 「ソクラテスは人間である。人間というものは死を免れない」というプログラムで、「ではソクラテスは死ぬか」の問いに「yes.」と回答されている。

注4) 三菱電機　MELCOM PSI　http://museum.ipsj.or.jp/computer/other/0009.html

Prologマシンはさかんに研究され、神戸大学 PEK[注5]、東京大学 PIE[注6]、京都大学 KPR[注7]なども研究、開発されました。

Prologによる並列マシン

そもそもPrologの実行の一面は、知識データベースから導かれる候補木を探索し、問い合わせに合致するものを探すというものです。Prologのプログラムは副作用がないので、候補木の探索は独立に行え、それは並列に実行ができます。これは、Prologの論理式のORを並列に探索するので、OR並列と呼ばれます。次はOR並列の例です。

```
human(ソクラテス).
dog(ポチ).
dog(シロ).
cat(タマ).
cat(クロ).
mortal(*X) :- human(*X).
mortal(*X) :- dog(*X).
mortal(*X) :- cat(*X).

?- mortal(*ANS).
```

mortal(*X)は、各節は独立であるから並列に実行できます。ということで、ICOTの設立前は、PrologによるOR並列で並列マシンを作ろうという機運が高まっていました。当時も現在も、高い並列度を引き出して、多くのコアを駆動することが難しいと考えられていました。それに対してOR並列は並列度がどんどん出せるので、良いだろうと思われました。

並列度の爆発の懸念

しかし、より深くPrologの並列の性質について考えたところ、当時はOR並列は並列性を制御するのが難しく、並列度が爆発(いわゆるスレッドが大量に生成され)し、計算の途中で資源が枯渇し、計算が進まなくなるだろうと思われ始めました(当時は、複雑な要素プロセッサで構成された並列計算機の規模は、数十〜100コア未満程度でした)。

そのころ、上田和紀氏がGHC[注8]というAND並列なProlog系言語を考案され、並列度は爆発せず、きれいに並列プログラムが書けるようになりました。ICOTでは、GHCをもとにKL1(Kernel Language One)を開発し、オブジェクト指向な並列Prologシステムを完成させました。ICOTの重要な成果として、KL1を並列に実行するマシンである、並列推論マシンPIM[注9]があります。

AND並列というのは、ある見方をするとCSP(Communicating Sequential Process)やErlangと同様な並列プログラミングパラダイムを持つものです。AND(',')でつながれた複数のプロセスが、変数を通信チャンネルとし、通信しながら並列に計算を行うというものです。

日本のProlog研究

ICOTの成果は、日本国内では評価が不当に低いのですが、海外ではICOTという組織を作ったこと、ICOTを始め日本のProlog研究は高く評価されています。

日本製のProlog処理系も数多くあり、AZ-Prolog[注10]やK-Prolog[注11]は、現在も開発が継続され販売されています。PC-9801時代に有名だったProlog-KABA[注12]の開発者は筆者の古い友人です。また、SWI-Prolog[注13]は、オープンソースで現在も開発が活発に行われており、Linux、Windows、Mac OSなどで動作し、GUIアプリケーションの開発も行えます。

Prologは現在もソフトバンクBBのクラウドシステム[注14]で使われたり、形を変えてSemantic Web[注15]として生きています。SD

注5) 神戸大学 Prologマシン PEK http://museum.ipsj.or.jp/computer/other/0015.html

注6) 東京大学 並列推論エンジンPIE http://museum.ipsj.or.jp/computer/other/0016.html

注7) 京都大学 論理型言語向き並列マシンKPRのORリダクション・ユニット http://ci.nii.ac.jp/naid/110002869429

注8) Guarded Horn Clauses http://ja.wikipedia.org/wiki/Guarded_Horn_Clauses

注9) http://www.jipdec.or.jp/archives/icot/ARCHIVE/Museum/FinalReport/appendix/node2.html

注10) ソフネック㈱のProlog処理系 http://www.az-prolog.com/

注11) ㈱KLS研究所のPrologコンパイラ http://www.kprolog.com/

注12) PC-9801で動作したProlog処理系。当時の京都大学数理解析研究所の桜川貴司氏、柴山悦哉氏、萩野達也氏によって作成された。

注13) http://www.swi-prolog.org/

注14) Prologベース プロダクションシステムによるクラウドサービスの運用監視 http://www.az-prolog.com/201010_jirei.pdf

注15) http://www.w3.org/2001/sw/

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

The Linux Foundation、「CloudOpen Japan」LinuxCon Japanと同時開催決定

The Linux Foundationは、クラウドのオープン化を促進する技術カンファレンス「CloudOpen Japan 2013」を5月29日～31日にホテル椿山荘東京で開催する。「LinuxCon Japan 2013」と同時開催となる。

CloudOpenは、クラウドソリューションのデプロイや開発に責任を負うソフトウェア開発者およびIT管理者を対象とするイベント。Chef、Gluster、Hadoop、KVM、Linux、oVirt、Puppet、Xenをはじめとする技術コンテンツのほか、ビッグデータ戦略、オープンクラウドのプラットフォームやツールなどを特集する。

参加登録は以下のWebサイトから行う。登録料金は、準早期登録料金がUS$350（3月1日～4月30日）、通常登録料金がUS$425（5月1日以降）、学生登録料金がUS$100となっている。

■CloudOpen Japan 2013 Webサイト
https://events.linuxfoundation.jp/events/cloudopen-japan

CONTACT The Linux Foundation
URL http://www.linuxfoundation.jp

Software

キヤノンITソリューションズ、米国Sencha社のHTML5準拠のWebアプリ開発フレームワーク「Sencha」シリーズを発売

キヤノンITソリューションズ㈱は2月14日、米国Sencha社とHTML5準拠でマルチデバイスに対応したWebアプリ開発フレームワーク「Sencha（センチャ）」シリーズのディストリビュータ契約を締結し、日本語ドキュメント版のライセンス販売ならびに技術サポートを行うことを発表した。

Senchaは、デスクトップからスマートデバイスまで一貫した開発をクロスプラットフォームで行えるフレームワーク。Webベース（HTML5/CSS3/JavaScript）のJavaScriptライブラリとアプリ制作環境で構成されており、効率的でデザイン性も高く、魅力的なクライアントアプリの制作ができる。

同社は3月14日からSenchaを発売する。価格は次のとおり。

▼Sencha（ライセンス製品）販売価格

製品名	製品内容とサポート	希望小売価格（税別）
Sencha Complete	Sencha Touch、Ext JS、Architect 日本語ドキュメントとE-mail サポート	120,000 円
Sencha Ext JS	日本語ドキュメントとE-mail サポート	78,000 円
Sencha Animator	日本語ドキュメント	14,800 円

CONTACT キヤノンITソリューションズ㈱
URL http://www.canon-its.co.jp

Software

ネオジャパン、次世代グループウェア「desknet's NEO」の小中規模向けパッケージ版を出荷開始

㈱ネオジャパンは2月26日に、グループウェア「desknet's」の後継となる新製品「desknet's NEO」の小中規模向けパッケージ版を発売した。

desknet's NEOは、累計276万ユーザ以上の販売実績を持つdesknet'sの後継製品。インターフェースにHTML5を全面採用し、従来型のWebアプリケーションとは一線を画す見やすさと使いやすさを実現している。PC、タブレット、スマートフォンの標準ブラウザで利用でき、各端末へのインストールは不要。

今回出荷開始となる小中規模向けのdesknet's NEOスモールライセンスは、5ユーザが39,800円、100ユーザが378,000円。さらに、他社グループウェア製品からの乗り換えの場合、50ユーザが59,400円、100ユーザが113,400円となる（いずれも税抜価格）。

従来の小規模製品は、組織の階層管理、時差対応、外国語メールの送受信、大容量データ管理などの機能は利用できなかったが、今回のスモールライセンスではエンタープライズライセンスと同じ機能を、そのまま利用できるようになった。

CONTACT ㈱ネオジャパン
URL http://www.neo.co.jp

Software

エクセルソフト、「インテルSystem Studio 2013 Linux版」を販売開始

エクセルソフト㈱は、Intel社が提供する組込み／モバイルシステムの開発ツールスイート「インテルSystem Studio 2013 Linux版」を2月27日より販売開始した。

同製品はインテルAtomプロセッサ、インテルCoreプロセッサファミリー、インテルXeonプロセッサを搭載したLinuxの組込み機器、モバイルデバイス向けソフトウェアを開発、最適化することができる。

C、C++、アセンブラ言語をサポートするとともに、GNUクロスビルドに対応し、EclipseのC/C++開発用プラグインCDTやYocto Projectなどのアプリケーション開発ツールキットへ統合できる。

価格はシングルユーザライセンスで499,800円（税込）となっている。次のWebサイトから評価版もダウンロードできる。

■評価版ダウンロードWebサイト

http://www.xlsoft.com/jp/products/download/intelj.html

CONTACT エクセルソフト㈱
URL http://www.xlsoft.com/jp

Service

at+link、「at+linkプライベートクラウドライト」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、業界最安値（同社調べ）の低価格で提供する「at+linkプライベートクラウドライト」を2月27日より開始した。

同サービスは、サーバ1台をまるまる自社専用のクラウド環境として提供するホスティングサービス。初期費用0円、月額15,000円の定額で、必要なときに専用の管理パネルから手軽に仮想サーバを作成できる。Webサーバやメールサーバだけでなく、テスト環境などの運用にも、物理ノードのリソース範囲内で自由に運用できる。

▼スペックと料金

	プランA	プランB（3月末日提供予定）
CPU	Xeon E3-1265Lv2	Xeon E5-2603
メモリ（最大）	8GB（16GB）	32GB（48GB）
HDD（最大）	500GB×2（1TB×2）	1TB×3（1TB×8）
利用可能VM数	無制限	無制限
グローバルIPアドレス数	/28（16）	/27（32）
初期費用（税別）	0円（台数限定キャンペーン）	180,000円
月間利用料（税別）	15,000円	25,000円

CONTACT at+link
URL http://www.at-link.ad.jp

Hardware

ロジクール、Windows 8に対応したマウスなど、4製品を発売

㈱ロジクールは、Windows 8のタッチジェスチャを表面全体で操作できる「ロジクール タッチマウス t620」を2012年11月より販売している。

同製品は「2012 CES イノベーション アワード」を受賞したM600と同じくスタイリッシュで高級感のあるデザインを採用したワイヤレスマウス。水平／垂直スクロールやページの進む／戻る、スタート画面の呼び出し、アプリケーションの切り替えなどの操作をジェスチャで快適に操作できる。また、同社のWebサイトよりソフトウェアをダウンロードすれば、ジェスチャをカスタマイズすることも可能。価格は6,980円となっている。

また、同社ではt620同様にWindows 8のタッチインターフェースに対応したタッチデバイス製品として「ロジクール ゾーンタッチマウス t400」（4,980円）、「ロジクール ワイヤレス充電式タッチパッド t650」（7,980円）、「ロジクール ワイヤレス タッチキーボード k400r」（4,980円）も販売している。

※価格はいずれもロジクールオンラインストアでの税込価格。

CONTACT ㈱ロジクール
URL http://www.logicool.co.jp

Letters from Readers

読者の表紙に対する感想は？

最近、本誌の読者アンケート（http://sd.gihyo.jp/）に表紙についてコメント欄ができました。「シンプルでよい」「季節を感じる」という感想がとても多いです。なかには「『Design』を名乗る雑誌にふさわしい、すてきな表紙」との意見も。ありがたや～。そんなお花シリーズの表紙も今回でちょうど1年。次からは新しいシリーズが始まりますよ。どんな絵柄になるのでしょう？　お楽しみに!

2013年2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 シェルスクリプティング道場

日常の簡単な作業程度ならプログラミングよりもシェルスクリプトのほうが必要なときに必要な作業をすばやくこなせます。そんなシェルスクリプトを使いこなせるようになるため、シェルの動作原理にまで踏み込んで、実践で役立つワザを解説しました。

シェルスクリプト道場の記事を読み始めてからずっとシェルにはまっています。

福岡県／めたぼさん

たいへん参考になりました。FreeBSDに限定した内容もありますが、OSに依存しない内容のほうがよかったと思います。

東京都／psiさん

新しいこと、知らなかったこと、忘れていたこと、総合的に短期間で見直せたことなどたいへん役立ちました。

富山県／YKB84さん

このレベルの内容はWebではほとんど得られず、雑誌の特集だからこそ得られる内容だと感じる。これからもこういったWebでは得にくい、濃く充実した記事を掲載していただきたい。

石川県／かっぱ大王さん

シェルスクリプトは身近に使っているせいか、非常に反響の大きい特集でした。動作原理に踏み込んだところが好評だったので、今後も深い内容の記事を検討していきたいと思います。

第2特集 超効率的勉強法

忙しい毎日であっても常に最新技術を学び、身につけることが求められるITエンジニア。そんなみなさんのために、「勉強の達人」森川滋之氏による物語仕立ての特別講座をお届けしました。

社会人2年目で、うまく勉強の時間が取れないと感じていたので、とても参考になりました。

東京都／佐伯さん

おそらく出来内さんとほぼ同い年です。

大阪府／neo-the-catさん

私は30歳過ぎなのですが、こんな年齢になって今さら自分のもの覚えの悪さ、要領の悪さを痛感し、何が自分に足りないのかをひたすら考えていました。そんなときに本特集が非常にありがたかったです。さっそく「伊達直人式超速読法」を実践してみました。「4色ボールペン」や「講義」もどこかで実践してみます。

神奈川県／miwarinさん

いつも本を読む時間がなくて困っていました。活用したいと思います。

佐賀県／佐々木さん

勉強時間を確保するのに苦労する、という声が多く寄せられました。弊誌が少しでもみなさんの技術力UPのお役にたてれば幸いです。

一般記事 Samba 4.0.0がやってきた

2012年11月にリリースされたSambaの新バージョンについて、新機能とインストール方法を取り上げました。

小中高校などで、校内LANを構築するとなるとクライアント環境がWindowsであることが、ほぼ前提条件のようになるので、必然的にWindows Serverの導入を求められます。今回、Samba 4.0.0でActive DirectoryのDomain Service機能が実装されたとの記事を拝見し、今後、学校現場でも導入できるのではと期待しています。かなりのコスト削減につながるのではないでしょうか。

宮崎県／maehrmさん

2つの異なるディストリビューションでインストール方法が書かれているのはとても参考になった。

愛知県／長屋さん

SambaはUNIX系OSを用いて、Windowsのファイルサーバを構築する際にはなくてはならない存在です。実際に使用している読者もいたのではないでしょうか。

一般記事　マイクロアドが開発／運営する広告配信システムの裏側

今、Web広告業界で注目を集めているのが、リアルタイムにWeb広告枠の売買を行う手法「RTB」です。そのRTBを支えるインフラ技術を紹介しました。

RDBとKVSの特徴がわかりやすく、勉強になりました。

神奈川県／つねさん

技術的な内容もおもしろかったですが、RTBの存在を知らなかったのでとても新鮮でした。

神奈川県／星野さん

Web広告は当たり前のように目にしますが、その裏側にRTBというしくみがあるのはご存じでしたか？　技術もさることながら、このしくみそのものに興味を持たれた読者も多かったようです。

連載

連載「ハイパーバイザの作り方」の濃さが好きです。これからもパンチのある記事をよろしくお願いします。

東京都／山下さん

本連載はかなり高度な内容を扱っていますが、本連載を好む読者はなかなか多いです。仮想化技術を使っても、内部のしくみを詳細に知る機会はあまりありません。そこがハイレベルな読者にウケているようです。

フリートーク

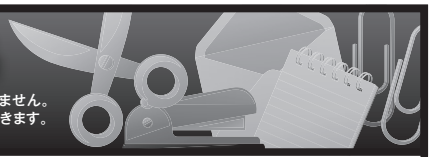
UNIX環境に慣れたくて、自宅PCをデュアルブートにしています。それがなかなかうまくいかず何度もOSを入れ直すはめに。GUIDやMBRなどマルチブート環境を作る記事を希望します。

愛知県／TMZFSさん

担当は複数のOSを使う必要があるときは仮想化ソフトを使っています。しかし、これもメモリが十分にないと満足に動きません。マルチブートにしろ、仮想化にしろ快適な環境を用意するには苦労が伴いますね。

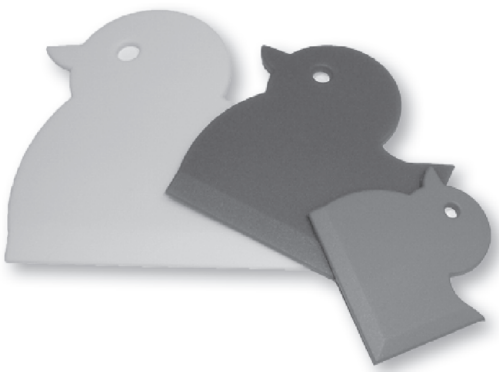
エンジニアの能率を高める一品

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。



ほこりトリ

650円（LMS 3枚入り）／おおかわ　http://www.ookawa1976.com



今回はディスプレイ用の掃除グッズを紹介します。「ほこりトリ」はシリコーンゴムでできた掃除用マットです。ディスプレイの画面上を軽くなでるだけで、ほこりがマットに吸い付いてきれいになります。ティッシュや布巾だと画面の隅だけはどうしてもホコリが残りがちですが、ほこりトリの場合、最初にマットの平面で大まかにホコリをかき集め、最後にマットの角で画面の隅にかたまったホコリを掻き出すことができます。ただ、マットにホコリが吸着したまま使い続けると、一度吸着したホコリが再度画面上に付いてしまいます。水洗いすれば、またきれいに拭きとれますが、スムーズに掃除を進めたいなら、事前に2、3枚のマットを用意しておき、適宜、取り替えながら使用するのがベストです。掃除は本格的にやると面倒ですが、こういうアイテムが手元にあると、気になったときに、すぐにきれいにできるのがよいですね。（読者プレゼントあります。8ページ参照）

祝　2月号のプレゼント当選者は、次の皆さまです

①Wi-Fi SDカードリーダー REX-WIFISD1 ..... 京都府　鈴木和貴様
②パスワードマネージャーミルパス ..... 長崎県　中尾義之様

★その他のプレゼントの当選者の発表は、発送をもって代えさせていただきます。ご了承ください。

次号予告

Software Design
May 2013

2013年5月号
定価1,280円　176ページ

4月18日発売

［第1特集］
新しい季節へキミと「ビギナーのための55冊」

良書に出会うためには運も必要です。良い本に当たれば、読者を大きく成長させてくれます。とくにIT業界は技術の流行が変わりやすいため、本がたくさん出版されています。その本の中から良い本に出会うのは大変なことです。そんなときに必要なのが先輩からのアドバイスです。経験に裏打ちされた良書を読んで、IT業界で強く生きていくための確実な基礎を身につけてみませんか？　IT書の目利きが選んだ珠玉の書籍で新しい季節をキミに!

［第2特集］
正規表現をマスターしていますか?

パターンを見抜き、欲しい処理を一発grep!　文字コードの種類も押さえながら、検索、データ整形などなど、仕事が捗る正規表現をマスター。

［一般記事］
セミナー講師は見た「春の新人あるある物語」

休載のお知らせ

「システムで必要なことはすべてUNIXから学んだ」（第10回）は都合によりお休みいたします。

SD Staff Room

●2月号は某大型書店チェーン店様のデータで、実売率79%を記録しました。おかげさまです。皆さんシェルが好きですね～。そんなわけでコンピュータプログラミングの需要はどんどん増えていくと思います。ということで特集1では、みんな気軽にプログラミングしてよね、というのがテーマでした。（本）

●最近、家の玄関に小猫がよく来る。ちょっと前に通販で買ったでかい鱈の甲（かぶと）をやったのが好評だったらしい。玄関にネットカメラがあるので、職場マシンの片隅に表示した画像を見ているとちょこんと座っていて可愛い。遠隔操作で餌をあげられるようなメカでも作ろうか思案中。某猫狂著者に教えを請わねばw（幕）

●インフルエンザ（B型）にかかりました。前号の制作終了間際だったので、最終作業を編集部のみんなに押しつけることに。今回のことで、私がいなくても雑誌は出版され……いや、そんな後ろ向きなシメではなく、やはりバックアップはITに限らず大切だと認識させられ、仲間のありがたみが身に染みました。（キ）

●最近、会社帰りにスマホで電子版の漫画を読むのが日課です。漫画は漫画喫茶でまとめて読む習慣ですが、ここ2、3年は漫喫に行く暇がなかったので、読みたい作品がたまっています。その欲求をここぞとばかりに解放しています。最近読んだ作品は『進撃の巨人』『深夜食堂』『月下の棋士』。（よし）

●先月号で滑らずに歩けると書きましたが……代わりに車で滑ってしまいました。車はレッカーされるほどの事故でしたが、スリップした場所がまだよかったのか、奇跡的に単独&救急車を呼ぶほどの怪我はしなくて済みました。あとでよくよく考えてみるとわたし今年は厄年。厄払いに行く必要はもうなさそうです。（年女）

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4月号

発行日
2013年4月18日

●発行人
片岡　巌

●編集人
池本公平

●編集
金田富士男
菊池　猛
吉岡高弘
＊
細谷謙吾（書籍編集長）
取口敏憲

●編集アシスタント
松本涼子

●編集協力
坂井直美
金子卓也（トップスタジオ）

●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。