春を感じよう。案ずるより動き出そう。

Renewal!!

USP MAGAZINE

for the sophisticated shell scripters

日本で唯一のシェルスクリプト総合誌

よりぬき版

2014 4 April Vol. 12 ¥500

「シェル芸」に効く AWK処方箋

新創刊記念対談

「東京」的なものに染まり過ぎない生き方とは

教えて先輩♥

サーバー運用お助けTips

Haskellでやってはいかんのか？

太田智美がスゴい人と盃をかわしつつインタビューする企画

今月のエクストリームエンジニア

USP MAGAZINE

2014 April vol.12

Contents

USP MAGAZINE

新創刊記念対談

吉田 基晴
×
當仲 寛哲

[前編]

「東京」的なものに染まり過ぎない生き方とは

暗号化技術を活用したデジタルデータの著作権保護システム（DRM）などの開発・販売を手がけるサイファー・テック株式会社は、徳島県の南東にある海部郡美波町に 2012 年に開発拠点を開設、2013 年には本社を同地に移転した。また同社代表取締役社長の吉田基晴氏は地域活性を手掛ける「あわえ」（地元の方言で路地という意味）を立ち上げた。限界集落に近いと言われる町に移った理由とは、そして地元で生きるエンジニアの生活とは。吉田基晴社長と、當仲寛哲 USP 研究所 代表取締役所長が語り合った。

吉田　2003 年の設立以来、サイファー・テックは東京に本社を構えていました。しかし、2012 年に徳島県海部郡美波町に開発オフィスを設立したのをきっかけに 2013 年に同町へ本社を移転しました。この町は僕の生まれ故郷で、NHK の朝の連続ドラマ「ウェルかめ」の舞台にもなった海に程近い町です。

當仲　海亀の産卵で有名なところですね。僕、学生時代に徳島から高知の室戸岬まで歩いたことがあるんですよ。途中で日が暮れて駅のベンチで寝たりしながら。

吉田　ひょっとしたら美波町の近くも通ったかもしれませんね。美波町は農業と漁業、観光の町で人口は約 7500 人。日本有数のサーフィンエリアとしても知られていて、当社オフィスの目の前には海が広がっているので、朝一番に波乗りしてから仕事に来るエンジニアや、IT ワークの間にオフィスに隣接する水田で稲作に励む社員もいます。釣り、舟遊び、農業体験、お祭りや運動会などのイベント、防災活動の参加。どれも特別なことではなくて、生活に溶け込んでいます。結構忙しいですが僕らはそのスタイルを「半 X（エックス）半 IT」と呼んでいます。

■求人広告に反応なし

吉田　僕は月の半々くらいで徳島と東京を行ったり来たりしています。サイファー・テックでは主に、暗号化技術を用いて違法コピーを防止するデジタルコンテンツの著作権管理（DRM）製品の提供や、企業の情報漏洩対策の支援などを柱にしています。また美波町の本社では、地域の人といっしょにさまざまな活動をしています。地元の中学校に社員がお邪魔して生徒にコンピュータの使い方を教える出張授業をやったり、高齢者の方にパソコンや SNS の使い方を教える教室を開いたり、さらに毎年夏には全国各地の大学生を対象に地域活性につながるアプリを開発するインターンシップ（就業体験）を実施。2013 年のインターンでは「電球が切れて付け替えられない」などの困り事を高齢者の方がつぶやくとその内容が支援者のスマフォ画面に表示される「みなみてボタン」というアプリをその学生達が作り、徳島県から表彰を受けました。美波町初の IT 企業として僕らにできることを探し、取り組んでいます。


美波町で稲作を行う
サイファー・テック

當仲　仕事があれば地方で暮らしたいという人は多いようですね。先日、佐賀県で地域貢献に取り組む企業の方にお会いしたときもそう伺いました。

吉田　ええ。当社の社員も地元出身者だけでなく、関東など町外から移住してきた人がいます。東京に本社があったときより、一緒に仕事をしたい思う仲間が集まりますね。創業まもない頃は東京で求人広告をかなり出したのですが、応募がほとんどなくて、採用には苦戦しました。暗号化技術にはセキュリティに関する高い専門性が必要です。田舎と違って東京のように人が大勢いる場所ならば、と

思い込んでいたのですが、反対でしたね。美波町にオフィスを作ってからは、社員の数は３倍に増えました。サーフィン好きな人、農業に取り組みたい人とか、IT ワークの合間を縫って遊びや暮らし、地域の活動に飛び回る人が多いですね。

當仲　うちはお客さんの業務システムを作る SI を主体にやっている会社ですが、2005 年の創業のころは有限会社というだけで相手にしてくれない人もいました。会社にブランドも信用もないので、社員集めはもっぱら呑み屋さんで話をして「君コンピュータいじれる？」「Excel ならちょっと使えます」「そう、じゃあうちきて勉強しながら仕事しない？」。ナンパみたいにいろんな人に声をかけてました。

■美波町に本社を移したきっかけ

吉田　僕は 2008 年頃から千葉で仲間と稲作に挑戦していました。「やるならできるだけ機械を使わず、手作り、そして無農薬でやろう」なんて調子のいいことを言っていましたが、そんなに生易しいものじゃありませんでした。かがんで田植えをすると、なまった足腰がすぐ立たなくなります。夏場の草取りは 35 度の炎天下で、熱中症になりかけてふらふらでした。でも、泥の中に足を入れると大人になってついぞ味わったことのない気持ちよさがあり、えんえん草をむしっていると無心になれるんです。見よう見まねでやって、秋には人生で初めて自分たちの手で作った米の収穫を迎えたときは、すごくうれしかったですね。味のことよりも、収穫できたことへの充足感だけで満足でした。いつも口に入れている食べ物はこうやって作られているんだ、そうした仕事に支えられながら、僕らのような暮らしや IT や金融や流通などのビジネスが成り立っているんだなと、まさしく身体で実感しました。

セキュリティ絡みの仕事はあまり表に出てこない裏方の世界で、中にはあまりよいイメージを持っていない方もいます。この仕事をやる意義とはなんだろうと悩んだ時期もありましたが稲作をやって迷いは吹っ切れました。僕らは僕らのやれることをやろう、自分の信じる道を進もうと思えるようになりました。こうした実体験に基づいた気づきのチャンスの多さが地方の魅力だと思っています。

現金収入を得るための「稼ぎ」は大事ですが、属する社会の中で支えあう「務め」、家族や自分を支える「暮らし」、人間らしい「遊び」の４つのバランスが大事だと思うようになったんです。私は 30 代くらいまでは稼ぎを増やし、遊ぶことに力を入れていましたが、「務め」や「暮らし」をしっかりと担えてこそ、本当の大人だと思えるようになってきました。務めというのは、共同での草刈りや、神社の傷んだ部分をみんなで修繕するといったコミュニティでのちょっとした活動です。務めや暮らしにもエネルギーを使っているから、美波町の地域社会がなんとか回っているんだと気付きました。

當仲　東京にいると、いいこと、楽しいこともあるけれど、「毒」もたくさんもらうんですよね。テレビをつけると、おいしいお店とか綺麗な服とか、おしゃれな街だとか、どうもバイアスのかかった情報がセレクトされてやたらたくさん流れてくる。東京にいるとかえって外の世界がわからなくなります。お金があれば世の中何でもできるような気にさせられる。お米もお茶も机でも椅子でも誰かが作っていて、そのそれぞれは小さくても膨大な営みが重なって僕らがこうやって仕事したり生活したりしているのに、それが見えなくなっていくのが東京のコワいところですね。

吉田　徳島県は全体で見ると、65 歳以上の高齢者が５割を超える限界集落の比率が日本で最も高い地域です。美波町も高齢化率は 40%台で、過疎化を食い止めるために残された時間はそう多くありません。

この美波町の中で仕事や遊び、暮らしや務めをしていると、お互い様の精神で地域の文化や暮らしが成り立っていることがよくわかります。大工さんが隣の家のちょっとした修理をしたり、収穫した農作物や釣った魚などをおすそ分けし合ったりと、お互い様だから人が少ないけれどもなんとか回っているんです。みんながそれぞれ得意なところを持ち寄って、子育てや地域の手助けをしたりしている。当社の若い社員も地域の輪にだんだん加わっています。僕も家族もお世話になりっぱなしです。だから少しでも恩返しをしたい。美波町が全国に先駆けたモデルとなって何か新しい仕組みのようなものを生み出せないか。それを模索しているところです。

＝次号後編へ

Profile

吉田 基晴

1971 年徳島県生まれ。株式会社ジャストシステムなどを経て、2003 年にサイファー・テック株式会社を設立。デジタルコンテンツの著作権管理（DRM）や違法コピー防止のための暗号化技術、企業の情報漏洩対策の支援などを行う。2013 年に本社を故郷の徳島県美波町に移転。同年株式会社あわえを設立し、地域活性を事業として手掛ける。趣味は釣り。

當仲寛哲

1966 年兵庫県生まれ。株式会社ダイエーでシステム改善を推進。2005 年に USP 研究所を設立。2008 年「ユニケージ開発手法および同開発コマンドセット」が IPA 主催「ソフトウェア・プロダクト・オブ・ザ・イヤー 2008 システム・基盤分野」を受賞。趣味は、全国各地を巡る年一度の社員旅行の企画。「移動時間が長すぎ」としばしば社員からブーイングを受けている。

No.4 スズラボ通信

山に登るのはそこに締切りがあるからだ　～Redmine のすすめ～

Author : **すずきひろのぶ**

Email: suzuki.hironobu@gmail.com

締切りがなければ人間何もしません

もちろん世の中には誰に言われなくても、きちんきちんと仕事をこなしていく、そんな方がいるのは分かっています。ですが、少なくとも私の場合、締切りのない仕事は出来たためしがありません。「無限大∞」は値ではなく状態であるわけで、つまり、締切りがない＝無限大の締切り日なのです。そして私という仕事関数は無限大の時に収束しません。

一方で締切り日があったとしても遅れるのが人の常。では、どうやって締切りを守るのか、あるいはどうやってスケジュールを管理していくのか。そこが問題です。

タスクを明確化し、それに締切り日をつける。さらにそのタスクを細分化し、複数のサブタスクに分割し、各々に締切り日をつけ、それを管理していくと割と便利です。本来はタスクの管理はプロジェクト管理あるいはプロセス管理の中で、どうタスクを管理していくかという一部でしかないのですが、これをタスクを中心にもってきて考え～それはタスク・オリエンテッドあるいはタスク・ドリブンと呼ぶのも良いかもしれませんが～管理してみます。主従が逆転しているじゃないか、とお叱りを受けるかも知れませんが、是非はともかく、バンバンとタスクを切って作業、スタート日、終了日、作業量を見積って管理していくと割と楽です。今回はそんなわけでプロジェクト管理支援 CMS の Redmine です。

Redmine

Redmine はプロジェクトのメンバーや関係者とプロジェクトの情報を共有するための CMS です。実際に使うと文章管理ツールや掲示板ツールとして使われるシーンが多いのですが、Redmine を紹介する際の最大の特徴は、やはりチケットによるプロジェクトのタスク管理です。

Redmine では、タスクを明確化して、スタート日、終了日、担当者などの属性情報を加えて登録し、マイルストーンを管理することができます。Redmine では、その１つ１つの登録のことをチケットと呼びます。チケットを扱うシステムとしては、有名な所ではバグ・トラッキング・システム（BTS: Bug Tracking System）の Bugzilla や課題トラッキング・システム（Issue Tracking System）の Trac などがあります。

Bugsilla: **http://www.bugzilla.org/**

Trac: **http://trac.edgewall.org**

バグ・トラッキング・システムや課題トラッキング・システムのチケットの使われ方と Redmine のチケットの違いは、前者は１つ１つのチケットが独立的なイベントであるのに対し、後者は時系列的でプロジェクトのスケジュールを意識している部分です。もちろん運用方法で前者も後者も違いがない使い方が出来ます。しかし Redmine は最初から、いわゆる線表的なプロジェクト管理をするように作っています。筆者はプロジェクト管理だけではなく、ミーティングや出張などのスケジュールを入れてカレンダー代わりにしています。そうすることで、関係者の中で最小限抑えておくべきスケジュールを共有できるからです。他に使えるアイデアとしては、大学の研究室などで指導する学生の進捗管理や、ゼミ運営などにも使えるでしょう。チケットはチケットを発行することができるロール（権限）があればいいので、権限さえあれば他人の担当するチケットもばんばん切れます。

Redmine をインストール

そうはいってもまずはインストールして使い始めなけ

ればいけません。Redmine 2.4.2 を CentOS 6.4 にインストールする話をしたいと思います。Redmine を触る前にまずいくつかやることがあります。それは SELinux をどうするかです。あと iptables で 80 ポートを開けるのも忘れずに。

SELinux をどう設定するか

まず最初に重大な決心をする必要があります。それは……**SELinux をオフにするか否か。**

筆者は SELinux を Enforcing(有効にして制限をかける) にしたままと、Permissive(有効だがログしか取らない) の両方でインストールしたことがありますが、SELinux の仕組みを知らない人が SELinux の制限をかけたまま Redmine をインストールするのは、不可能とはいいませんが、正しい設定をするには大変骨の折れることであろうと思います。ただし、そのかわり万が一外部から（たとえばバッファーオーバーフローなどで）システムに侵入しようとした時、大変心強い仕組みを持っています。Enforcing にしていると、マルウェアがちょっとやそっとじゃ動かないのと同じ理由で、上手にインストールしないと動きません。

```
$ getenforce
Enforcing ← Enforce のモードになっている
```

Permissive は「有効だがログしか取らない」のでインストールが難しくなるようなことはありません。先ほどの高いセキュリティを保つことはできませんが、一方でログはどんどん記録されていくので監査としては意味があります。が、しかし、これもまた正しく SELinux のための設定してはいないので、大量にログが吐き出されるという資源の浪費が問題になります。ですから、SELinux の設定を適切にせずに運用するのは、あまり褒められたものではありません。

さて、多くのブログやインストール説明では次の一文が一番最初にきています。

SELinux を disabled（完全オフ）にすることを推奨する。

残念ながら筆者の経験では、この一文を強く否定することは出来ません。それが現実だからです。SELinux を設定しつつ運用したい方は次の URL を参考に自分の環境に合わせてください。

http://www.redmine.org/projects/redmine/wiki/RedmineAndSELinuxOnCentOS

次に CentOS 6.4 は デフォルトで iptables により不必要なポートは閉じられているのでポート 80 を開ける作業をします。

iptables / ファイアウォール

次の一文を ssh(ポート 22) を許可する記述の後ろあたりに加えます。

```
# vi /etc/sysconfig/iptables
```

設定後システムをリブートしてください。

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80
-j ACCEPT
```

コンパイル環境や apache と mysql のインストール

コンパイル環境は Ruby のライブラリ群である Rubygems をインストールする時に必要になります。apache と mysql は redmine で使います。

ImageMagick は画像処理、ImageMagick などで使う ipa-pgothic-fonts はフォントですが、これは Redmine のデフォルトでインストールする際に要求されるので入れておきます。

```
# yum groupinstall "Development Tools"
# yum install httpd mysql-server
# yum install zlib-devel curl-devel openssl-devel httpd
-devel
# yum install apr-devel apr-util-devel mysql-devel
# yum install ImageMagick ImageMagick-devel ipa-pgothic
-fonts
```

mysql にはインストール後、すぐに最低限のセキュリティ対策をツールを使って行います。そのために mysql_secure_installation はデフォルトでインストールされています。余計なアカウントを消したり、ここで mysql の root (管理者) のパスワードが変更されたりします。

```
# service mysqld start
# mysql_secure_installation
# chkconfig httpd on
# chkconfig mysqld on
```

Ruby と Rubygem をインストール

CentOS 6.4 のシステムにインストールされているデフォルトの Ruby を使わず自分でコンパイルした独自の Ruby を利用します。利点は Redmine のバージョンとシステムデフォルトの Ruby とのバージョン齟齬を起こしていきなり動かなくなるようなケースを避けられるこ

とです。安定した運用ができるかわりに、Ruby に脆弱性があった時など、自分で責任をもってアップデートさせなければいけません。どちらのリスクを取るかですが、せっかくなので独自にインストールすることにします。

/opt/ruby の下にインストールします。Ruby のソースコードもそのディレクトリの下に一括管理します。ここで利用する Ruby のバージョンは ruby-2.1.0 です。

```
# mkdir /opt/ruby
# cd /opt/ruby
# mkdir src
# cd /opt/ruby/src
# wget http://cache.ruby-lang.org/pub/ruby/2.1/ruby-2.1.0.tar.gz
# tar zxvf ruby-2.1.0.tar.gz
# cd ruby-2.1.0
# ./configure --prefix=/opt/ruby
# make
# make install
```

今度は Rubygem をインストールします。

```
# cd /opt/ruby/src
# wget http://production.cf.rubygems.org/rubygems/rubygems-2.1.11.tgz
# tar zxvf rubygems-2.1.11.tgz
# cd rubygems-2.1.11
# /opt/ruby/bin/ruby setup.rb --prefix=/opt/ruby
```

自分でもよく間違えるのが、この setup.rb は、先にインストールした /opt/ruby/bin/ruby でなくてはなりません。もしダウンロードしてきた Rubygems が古いバージョンであれば、次のコマンドを実行すると自動的に Rubygem 自身でアップデートしてくれます。

```
# /opt/ruby/bin/gem update --system
```

このあとの作業の時、システムの Ruby と無用なコンフリクトを起こさないように作業中は次の環境変数を有効にしておきましょう。筆者は作業が終わるまで .bashrc に入れています。

```
export PATH=/opt/ruby/bin:$PATH
export RUBYPATH=/opt/ruby
export RUBYLIB=/opt/ruby/lib
export GEM_HOME=/opt/ruby/lib/ruby/gems/2.1.0
export GEM_PATH=/opt/ruby/lib/ruby/2.1.0/rubygems
```

次に Rubygem で bundler と mysql2 と passenger をインストールします。bundler はインストール時に必要なツール群です。mysql2 は Redmine で採用している mysql 接続のためのライブラリです。以前は Rubygem の後ろに 2 のつかない mysql を使っていたので、むかしと同じだと思っている人や、古いドキュメントを読んでいる人はひっかかります。というか筆者がひっかかりました。先ほどの環境変数が有効であるという前提で次のように Ruby のライブラリをインストールします。

```
# gem install bundler
# gem install mysql2
# gem install passenger
```

この時、いろいろとコンパイルが始まりますので、ちょっとビックリするかも知れません。最初に開発環境を入れたのはそのためです。

Redmine 本体のインストール

標準的であろう /var/www の下にインストールします。

```
# wget http://www.redmine.org/releases/redmine-2.4.2.tar.gz
# cd /var/www
# tar zxvf redmine-2.4.2.tar.gz
# mv redmine-2.4.2 /var/www
# chown -R apache.apache /var/www/redmine-2.4.2
```

bundle を使って必要な Rubygem ライブラリを自動インストールします。

```
# cd /var/www/redmine-2.4.2
# bundle install
```

設定ファイルの設置

Redmine のコンフィグレーションを設定します。まずデーターベースの設定部分から。あくまでも例ですが、次のように決めたとします。

データベース : db_redmine

ユーザ名 : user_redmine

パスワード : 38m65tcub0

それをデータベース設定ファイル内に設定します。

```
# cd /var/www/redmine-2.4.2/config/
# sudo -u apache cp database.yml.example database.yml
ファイルを apache の持ち物にするため。
# vi database.yml
```

次に、mysql でデータベース、ユーザを作ります。

```
# mysql -uroot-p
```

以下は SQL 文です。

```
create database db_redmine default character set utf8;
 grant all on db_redmine.* to user_redmine identified by '38m65tcub0';  flush privileges;  exit;
```

次に一般の設定です。

```
# sudo -u apache cp configuration.yml.example configuration.yml  # vi configuration.yml
```

production の設定セクションを有効にします。次にrmagick_font_path を設定します。

```
production:
  email_delivery:
    delivery_method: :smtp
    smtp_settings:
      address: "localhost"
      port: 25
....
rmagick_font_path: /usr/share/fonts/ipa-pgothic/ipagp.ttf
```

ファイルの最後に有効になっている production: を無効にします。これがあると最初に有効にした production: の設定が上書きされ無効になってしまいます。これでハマりました。メールの配送に関しては、そのサーバーに MTA がインストールされていれば配送されます。

Redmine の初期化

まず generate_secret_token を実行しトークンを作ります。次にデータベースのテーブル類を作り、そしてデータベースに初期値を設定します。すべてスクリプトがあります。

```
# cd /var/www/redmine-2.4.2
# bundle exec rake generate_secret_token
# RAILS_ENV=production bundle exec rake db:migrate
# RAILS_ENV=production bundle exec rake redmine:load_default_data
```

Ruby on rails のアプリケーションとして Redmine が正しくインストールされているかを確認します。

```
$ rails server -e production
```

ローカル上で localhost:3000 へブラウザでアクセスします。このタイミングで Redmine の admin のデフォルトパスワード (admin) を変更しておくと安心でしょう。Redmine の使い方がわからないので後回しにしたとしても、この管理者 admin、パスワード admin は最悪なので、早い時点で変更しておいてください。

Passenger と apache との同期

Passenger は Redmine(Ruby on rails アプリケーション) を効率よく使うためのフレームワークです。Redmine は cgi でも動かすことが可能ですが、通常は Passenger を使います。インストールには既に入っているインストール用のコマンドを使います。

```
# passenger-install-apache2-module
```

コンパイルが始まりモジュールがインストールされます。最後に「この設定を記録して、設定ファイルに入れろ」というメッセージが出ますが、これはいつでもわかるので、とりあえずそのままにしておきます。

/etc/httpd/conf.d/ に下に passenger.conf を作成し必要なパラメータを設定します。

```
# passenger-install-apache2-module --snip > /etc/httpd/conf.d/passenger.conf
```

加えて passenger を apache のユーザ ID で動かすために passenger.conf に次の記述を入れます。最後の行は最新のバージョンの Redmine の設定ではシンボリックリンクが無効なので、それを有効にしておくためです

```
PassengerUserSwitching OFF
RailsDefaultUser apache
PassengerResolveSymlinksInDocumentRoot ON
```

httpd のバーチャルホスト設定

ここでは /etc/httpd/conf.d ディレクトリ以下に redmine.conf というファイル名で以下の記述を入れました。redmine.example.net をアクセスすると Redmine にアクセスできます。

```
<VirtualHost *:80>
  ServerName redmine.example.net
  DocumentRoot /var/www/redmine-2.4.2/public/
  <Directory "/var/www/redmine-2.4.2/public">
    Options Indexes FollowSymLinks
    Order allow,deny
    Allow from all
    AllowOverride all
    Options -MultiViews
   </Directory>
</VirtualHost>
```

あとは httpd をリスタートさせてください。デフォルトでは admin/admin が有効なので速攻で安全な変更をしてください。

さいごに

これで誌面がつきてしまいました。次回に Redmine の使い方を説明したいと思います。

いま私たちは何を学ぶべきか

人間とコンピュータの可能性

第12回 コミュニケーションと図形理解

written by 慶應義塾大学 名誉教授 **大岩 元**

これまでの連載で抽象化を軸に情報技術の学びについて解説してきた。しかし今回から少し視野を広げて、技術者として必要な技術以外の能力についても考えていこうと思う。最初はコミュニケーション能力を取り上げる。

ブランコ

右ページに掲載した６コマのマンガは 1973 年にロンドン大学の計算センターで発表されたものである。新たな情報システム開発を行なう場合の困難を説いたものとして関係者に広く愛用されている。あるソフト開発企業の社長は、このマンガがたいそう気に入って社屋の入口に貼り出したケースまであったと聞く。

プロジェクト依頼者の考えていたシステムは要求書にまとめられた途端に曲解される。その３枚の板が３本のロープに変わるように。これを分析して設計書にまとめると、なぜかロープの本数は２本に戻るが、今度は板がロープにつながれるも地面の上に置かれることになる。この設計書に基づいて作られたプログラムを見ると、板は幸いにして再びロープでつり下げられるが、木の幹に遮られて、スイングさせられない代物が出来上がっている。

こうした事態は実務上ではよく起こるらしい。だが何億円、何十億円もかけて作ったプログラムであればこのような状況下でも後戻りは許されず、何とか稼働させなければならない。火消しと呼ばれる「有能」な技術者が投入されて、膨大な費用をかけて稼働させることになるが､その結果､利用者に導入されたプログラムは５番目の、見るも無残なブランコと化してしまう。特に、官庁関係のシステムにこうした類のものが多く、パスポート１枚を発行するのに１千万円以上の費用がかかったりするのである。

この話にはオチがついている。ブランコを望んだユーザーは、実は古タイヤを木にぶらさげて遊びたかっただけだった、というものである。

困難な開発には保険制度が必要

このような悲劇が繰り返し起きることは、IT がムーアの法則に従って発展を遂げてきた以上、必然的というほかはない。コンピュータの応用分野には、常に新たな技術や製品・サービスが生まれ続けている。その結果として、だれも見たことがないシステムが常に開発されてきた（いまも、そうである)。したがって複数の関係者が描くシステム像は前述のマンガのように、いずれもどこかしら似たところを含みつつ、立場によって異なるものになってしまっても致し方ないことなのである。そして、その中から少なからずブランコの意味を成さないものが作られてしまう。

こうした事態を防ぐ方法として、保険制度がある。実際に韓国でシステム開発をした人の話によると、開発費の１割程度の金額を失敗に備えた損害保険料の支払いに費やしたそうだ。保険をかけている場合、「これ以上の開発を続けても、完成後に得られるメリットより損害の方が大きい」と判断されたならば開発を中止することができる。欧米の IT システム開発で中止される案件が多いのは同様の保険制度があるからであると耳にしたことがある。ただし保険でカバーするためには、それまでの開発が正当なものであったということを第三者に証明する必要がある。しかし、そもそも開発工程の正当性を過去にさかのぼって証明しうるような案件自体が、日本にはあまりないのではなかろうか。ゆえに保険制度が機能する基盤がほぼ無いというのが、日本の実情である。「い

やいや、欧米に較べて日本の情報システムは開発における失敗が少ないのだ」という議論があるが、実のところ失敗を隠してしまうために問題が表面化しないだけなのではないだろうか。

コミュニケーション能力

ブランコをきちんと動かすには、突き詰めると関係者間での相互理解が完璧に行なわれるほかに手立てはない。そのためには関係者が、自分がシステムをどのように理解しているのかを他人に分かりやすく伝えられなければならないし、伝えられた人はその内容を正確に受けとめなければならない。

こうした能力を育成するのに、前回解説したKJ図解が威力を発揮する。KJ図解においては、漠然とした意味を、カードの幾何学的な配置に反映させる作業がキモである。人間は、ことばで説明したことが理解できなくても幾何学的な配置に対しては直感的な、鋭い理解力を持っている。同じ近さでカードが配置されていても、2枚のカードが上下に配置されているか、左右に配置されているかの差を敏感に感じ取るのである。その結果、自分の直感に合わないと感じる配置に関して、関係者間で議論が始まるのである。

こうした議論が生み出される前提として、カードに書かれる日本語が正確に記述されていることが必要となる。配置に関して議論が生じた時には多くの場合、カードに書かれたことばの意味の理解が人によってずれていたためであることが多い。この結果、配置の議論をきっかけに、カードに用いられる単語の意味理解が深まっていくのである。KJ法が1970年代のビジネスパーソンに愛用された一つの要因はこの点にあるのではないかと思う。

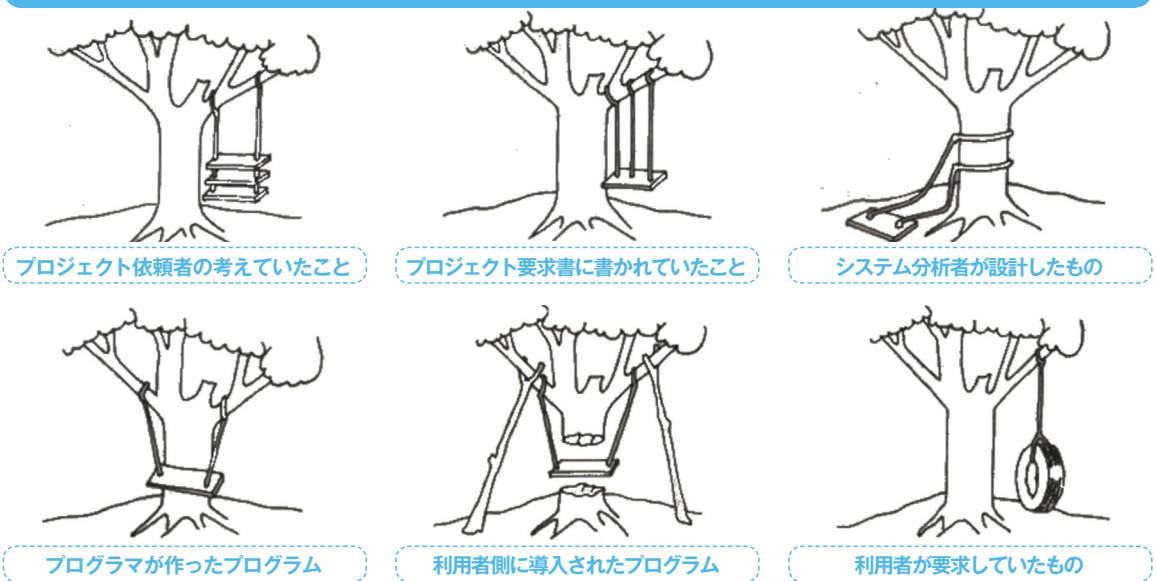
KJ図解を作っていく過程で、参加者の間の問題理解が進み、知識や状況が正確に共有されるようになり、結果としてその後の集団活動が効率的に展開できたのではないだろうか。

二次元図形に関する人間の理解力

二次元図形に関して人間の直感力が優れている理由は、動物の視覚がその生存のために数億年以上の長い時間をかけて磨き抜かれた能力であるから、だろう。動物は視覚から得られた情報を手がかりに危険を察知し、身を守る行動を起こすことができる。わずかな視覚情報からでもその意味を汲み取れる能力が生存の決め手になってくる。こうして磨き抜かれた視覚情報の理解力には、コンピュータでも簡単にはシミュレートできないものがある。例えば、囲碁の名人の盤面理解には、このような能力が縦横無尽に駆使されているものと思われる。

情報システムに関する人間の意味理解を、図形で表現できれば、その理解は深いものになると同時に、それを他人にも伝えやすくなるのである。

ソフトウェア開発の問題点



シェルスクリプト大喜利

第十二回

司会：『もっと吹く』編集長・みかん

皆様こんにちは。三か月のご無沙汰、シェルスクリプト大喜利（略して sh 大喜利）のコーナーです。

このあいだのソチオリンピックは盛り上がりましたねぇ。なぜか四輪になり、あれって実はわざとなんじゃないかと噂された開会式があったり、ロシアの女子フィギュアスケートのあの選手の言動がとても正直で人気を博し、なぜか日本で萌えイラスト化されたり、ボブスレーではジャマイカが 12 年ぶりに参加してクール・ランニング。結果は最下位だったのに、なぜか金メダルを取ったチーム（どこでしたっけ？）より大々的に報道されたり。

そんなオリンピックが盛り上がっていた裏で、もう一つの競技会が密かに盛り上がっていたことはあまり知られていません。え、何の競技会かって？この大喜利ですよ。ほら、前回出したお題をよく見直してみてください。有名な競技が含まれているじゃありませんか。ソチオリンピックを記念して、そういう問題を仕込んでいたんですよ。

長話はこれくらいにして、そろそろ始めましょう。シェルスクリプト大喜利、まずは本コーナーのご説明から。

シェルスクリプト大喜利とは

シェルスクリプト大喜利特有のルール

一、sh 大喜利はクイズやテストではありませぬ。なので決まった答えというものはないのです。あえて言うなら面白いスクリプトが正解！

二、面白いスクリプトとは、例えばこんなもの。

イ、人が考えつかない意外性がある

ロ、美しい、芸術的、記述がシンプル、高速、など

ハ、アイデア・こだわりが光る

ニ、ネタになるバカバカしさ、くだらなさがある

など。でも最後のは段位強制返還の恐れありよ。:-)

三、スクリプト動作環境は Linux とし、基本的に Linux JM(http://linuxjm.sourceforge.jp/) に記載されているコマンド及び機能のみ使用可能とします。これは多くの人が楽しめるようにするためなのです。（JM にあるので、C シェル系での解答も OK ！ **あと ksh、zsh も OK にしました**）

四、sh 大喜利はシェルスクリプトを披露する場なので、**Perl や Ruby、Python などは使っちゃダメ**です。そもそも JM にも載っていません。逆にシェルスクリプトにとって不可欠な awk や sed 等は OK です。JM にもありますし。でも、よっぽど面白ければ、Perl とかなきにしもあらず ??

五、Open usp Tukubai(http://uec.usp-lab.com/) も使用 OK ！　但し、使う意義がそれなりに感じられないと採用はキビシいですよ〜。

ルールもおさらいしたところで、さあ始めましょう！

本番開始

<一問目>

1 ～ n（n は引数で指定）の自然数の中に存在する『安全素数』を全て求めるシェルスクリプトを書いてください。

毎度お馴染み、整数問題。安全素数というのは、「素数 p のうちで、$(p-1)/2$ もまた素数になっているもの」です。情報セキュリティー分野で有用なために、「安全」の名が付いているそうですよ。

でも素数を列挙してくれる primes コマンドがあるのでスッキリした解答が多かったですね。

◎K 師範の解答

```
1 #!/bin/sh
2 primes 1 $(((($1-1)/2)) |
3 while read i; do
4   a=$((2*$i+1))
5   primes $a $(($a+1))
6 done
```

まずは師範になったKさんから。はい、元の定義をそのままシェルスクリプト語に翻訳したかのように素直です。大変に師範らしい明解な解答ですね。

大喜利的に見れば特にヒネリの無い解答だとも言える。でも、シンプルなものはシンプルなまま美しく書けるシェルスクリプトのよさが出ていい、と思ったので採用。いいね！**初段授与**。

◎gori.sh 師範の解答

```
1 #!/bin/bash
2 primes 1 $1 | sort > list
3 cat list | awk '{print $1*2+1}' | cjoin0 key=1 li
  st | sort -n
```

こちらも、師範になったgori.shさんから。素数pと$2p+1$の表を作り、Tukubaiコマンドのcjoin0でINNER JOIN相当をやって、$2p+1$の数列の中から素数のものだけを絞り込んでます。

そうそう、こういう解法の投稿も期待してたよ。同様の処理をPOSIX標準のjoinコマンドで頑張ってくれた投稿もあったんだけど、見た目のシンプルさでこちらを採用。でも、べ、べつにOpen usp Tukubaiを贔屓してるわけじゃないん、だからねっ！シンプルイズビューリホー、**初段授与**。

◎おとどさんの解答

```
1 #!/bin/sh
2 primes 1 $1|awk '{P[$1]=1}END{for(p in P){if((p-1
  )/2 in P){print p}}}'
```

師範達とはうってかわって、ややごちゃごちゃ系の回答ですね。AWKの連想配列を使って$(p-1)/2$もまた素数になっている素数pを絞り込んでいるようです。

師範のキレイな模範解答もいいけどこういうのもまたいいね！添付のコメントによればコードの短さを意識したというんだけど、確かに一問目への投稿の中で最短だったよ！前回の初投稿ではあえなく段位を授与できなかったけど、二度目の挑戦にして今度こそ**初段授与**だ。

◎TSB_K2K さんの解答

```
(編注)予めシェル変数nに最大値を代入しておく)
$ echo "x=5; while(x<=$n){ f=1; y=(x-1)/2; if(x%2&&
  (y==2||y%2)){ i=3; while(i*i<=x&&f){ if(x%i==0||y
  %i==0){f=0}; i+=2 }; if(f==1){x}; }; x+=2 }" | bc
```

これもシンプルとは対極にある解答ですね。他の投稿では軒並み使われていたprimesもfactorも、それにAWKまでも使わず、素数の計算から安全素数の絞り込みまで、bcコマンド1つでこなしてます。

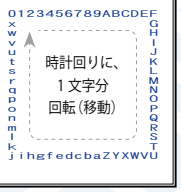
いやぁ〜頑張ったねぇ。AWKだけで何でもこなす解答はよく来るけど、bcでやっちゃうところに感心したよ。まぁ、「nは引数で指定」という問題文に準じてシェルスクリプト化してないぞ、とかツッコミどころはあるが、あまり細かいこと言うのも野暮ってもんだ。よし、**初段授与**！

◇　◇　◇

いつもの整数問題で体を温めたところで、次の問題にいきましょう。

<二問目>

図のように、16×16の領域の外周部分に、60個の英数字が並んだテキストデータがあります。これを時計回りに1文字分回転（移動）させるシェルスクリプトを書いてください。



というわけで、これがソチオリンピック開催記念「競技」です。ほら、**回転ですよ、回転。**（意味が違う！）でも、この競技だって、スピードで競うもよし、技（コード）で競うもよし、という具合にいろいろ見どころがあって、熱い戦いが繰り広げられたのです。では、入賞者のコードを紹介！

◎gori.sh 師範の解答

```
1 {c=b;b=a;a=$0;}
2 NR==2{print substr(a,1,1)                substr(
  b,1,15)}
3 NR>2{ print substr(a,1,1)"               "substr(
  c,16,1)}
4 END{  print substr(a,2,15)               substr(
  b,16,1)}
```

まずは、銅メダル解答。これは、おぉ〜AWKのコードですよ。添付コメントによれば「このテキストをkaiten.awkなどの名前で保存して、awk -f kaiten.awk < inputdataなどのように実行すればよい」とのことです。

うーん、力業でこなしてるって言ってるんだけど、そのわりに最終的なコードはわりとスッキリしてるねぇ〜。また、なんでこれで回転ができるのか、よーく読まないと理解できないのもトリッキーでいいね！わがまま言うと、最初の行に#!/usr/bin/awk -fなーんていうシバンを付けて単体でコマンドとして動くようにしてくれたらさらにカッコよかったけどね。とにかく素晴らしい、**一段授与**！これで二段だ。

◎TSB_K2K さんの解答

```
1 #!/bin/sh
2 cat curve |  tr "[0-9A-Za-z]" "[z0-9A-Za-y]"
```

次は、銀メダル受賞作。出題したアタクシの意表を突いたナイスな解答ですよ！先に補足しておきますが、2行目の curve とは元データの入ったテキストファイルだそうです。

久しぶりに「うぁ〜やられたー！」と思った解答だったよ。マジメに回転するんじゃなくて、回転方向にみて各々一つ手前の文字に置換しているだけ。だからもちろん、英数字がアルファベット順に並んでなきゃ通用しないんだけど、アタシャ確かにアルファベット順に並べたよ！

苦し紛れに文句を付けるなら、この解答は元データを名前固定のファイルから受け取るようにしてるとこがオシいな。標準入力から受け取るようになっていてくれれば、このシェルスクリプトを 60 個パイプで繋いで 1 回転とかやって遊べるからね。でもよく考えてくれたわ。単純がゆえに全出場選手（投稿者）のなかでスピードも最速だったし。「恐れ入った！」の**二段授与**。これで三段だ！

◎札幌 awker さんの解答

```
1 #!/bin/sh
2 awk ´NR==1{print " "$1} NR>1´ abcring.txt |
3 awk ´NR==1{l=substr($0,length($0));print substr($
  0,1,length($0)-1)} NR>1{print substr($0,1,length(
  $0)-1) l;l=substr($0,length($0))} END{L=sprintf("
  %0" length($0)-1 "d",0);print L l}´ |
4 awk ´{L3=L2;L2=L;L=$0;if(L3!=""){print L3}} END{p
  rint substr(L2,1,1) substr(L2,3,length(L2)-3) sub
  str(L,length(L)) substr(L2,length(L2));print subs
  tr(L2,2,1)}´ |
5 awk ´{L2=L;L=$0;if(L2!=""){print substr(L,1,1) su
  bstr(L2,2)}}´
```

そして、栄光の金メダル作がこちら！これの何がスゴいかというと、テキストの一辺の文字数が一般化されてるんですよ。だから 16×16 はもちろん、3×3 だろうが、20×20 だろうがちゃんと回してくれるのです。

コードはゴチャゴチャ系なんだけど、とにかく一般化されてるのには恐れ入ったよ。それはそうと、どういう解法で回転させてるんだろうと思ったんだけど、awk を 1 つずつ動かしてみたらよくわかったよ。なるほどねぇ。（読者の皆さんも、「百聞は一見にしかず」だ！）第 11 回冬季オリンピック開催地、札幌の意地ってやつなのかねぇ。いろいろとすばらしい！金メダルってことで、一気に**三段授与**だ！

あ、そうそう。札幌の**北海道大学生協さん**でも本誌を置いてもらえることになったので近くにお住まいの方はヨロシク！

◇ ◇ ◇

二問目はたくさんの解答が来て感謝。中には正しく動作せずに失格になったものもあったんですけど（grep -o を用いた投稿は面白かったんだけど残念）、載せきれなかった皆さんごめんなさい。さて三問目。

<三問目>

前回の大喜利で ping コマンドで時間を稼ぐものがありましたが、他の何らかのやり方やコマンドで、時間稼ぎをする方法を教えてください。ただし、次の条件があります。

- **秒数をある程度予測できること（単純なループはダメよ）**
- **なるべく珍しいもの（sleep とかはちょっとねぇ）**

三問目は毎度お馴染み、ちょっと変り種の問題です。物事は何でもかんでも速ければいいってもんじゃありません。時には時間稼ぎも必要です。シェルスクリプトで時間を稼ぐなら sleep コマンドという便利なものがありますが、もしそれが使えなかったらどうする？ということでこの問題です。

これまた色々な解答が来ましたが、結構やり方あるもんですね。

◎tomi さん、K 師範、猿二号さん、TSB_KZK さんの解答（代表して tomi さん）

```
$ timeout X yes > /dev/null      # Xに秒数を指定
```

4 名の方からいただきましたが同種の解答だったため、まとめさせてもらいました。これらの解答の肝は timeout コマンドです。後ろに無限ループする何らかのコマンドを指定し、それを所定の時間が経過したら強制終了させるというわけですね。

アタクシ、timeout コマンドの存在をすっかり忘れてたよ。そういやあったなぁ、これ。そんな自分への反省を込め、各自に一段授与。tomi さん三段、K 師範二段、猿二号さん五段、TSB_KZK さん四段ね。

◎さいたま帝国さんの解答

```
1 #!/bin/sh
2 watch -n $1 ´(if [ -f done.tmp ]; then kill -TERM
  $PPID; rm done.tmp; else touch done.tmp; fi)´ >/d
  ev/null
```

これはちょっと面白いですよ。他の投稿で、top コマンドや vmstat コマンドを利用していたものがありましたが、それらは一定時間待機した後に勝手に終了してくれます。でも watch コマンドはご存知のとおり一定間隔ごとに指定コマンドの実行を繰り返すので、終わりません。そこで、二度目の実行時に

watch プロセスを殺しにかかってるんですね。

うーん、一捻り加えてくれてありがとう。この投稿が来なかったら top コマンドや vmstat コマンドによる解答を採用していたところだけど、これ見せられちゃぁねぇ。こういう展開もまた、sh 大喜利の面白さと思うよ。よし、**一段授与**。只今四段。

◎$C_6H_8O_7$（クエン）さんの解答

```
1 #! /bin/sh
2 goal=$(echo $(date +"%s.%N")+$1 | bc)
3 while awk 'BEGIN{exit ('$(date +"%s.%N")'>'$goal')?1:0}';do continue; done
```

添付のコメントによれば、「精度にこだわって、ナノ秒まで対応してみた（笑）」とのことです。date コマンドで UNIX 時間をナノ秒単位まで求め、指定された時間が経つまでループするという戦法ですね。

シェルスクリプトでホントにナノ秒精度の時間が計れるのかよ！とツッコミ入れるのはお約束かな？でもアイデアは面白いね。**一段授与**！これで四段かな。

◎イタローさんの解答

```
 1 #!/bin/sh
 2 mkdir wait
 3 cd wait
 4 touch -t `date +%Y%m%d%H%M%S | calclock 1 - | awk -v t=$1 '{print $2+t}' | calclock -r 1 - | sed 's/.* ¥(.*¥)¥([0-9][0-9]¥)/¥1.¥2/'` timeup
 5 while :; do
 6   touch now
 7   ls -1t | head -1 | grep now > /dev/null && break
 8   sleep 1
 9 done
10 rm -rf ../wait
```

はい来ました、ヘンな解答が……。Tukubai コマンドの calclock を使って時間の計算をしていたり、ファイルのタイムスタンプの新旧を、しかも ls コマンドで比較していたりするなど、せっかく見どころを作ってくれたはずなのに、8 行目にナゼか sleep。添付されていたコメントには「コンピューターに無駄な負荷を掛けちゃいかんと思って sleep を挿みました。僕エライでしょ」と書いてありまして……。

「うんうん、そのとーりだね。エライぞっ！」……ってコラァー！！！ sleep 使うんだったら**そもそも sleep 一回呼ぶだけでいい**じゃないか。と、大いなるツッコミどころ持った投稿ありがとう。**一段剥奪**しちゃうからね。これで二段に逆戻りだけど、懲りず今後も投稿よろしくねっ！

◇　◇　◇

といったろころで本日の大喜利はこれにてお開き！読者の皆さん、投稿してくれた皆さん、今回もありがとうございました。

投稿大募集

次回のお題

一、1 ～ n（n は引数で指定）の自然数の中に存在する「ルース＝アーロン・ペア」を列挙するシェルスクリプトを書いてください。

二、標準入力から来る HTML テキストを読み込み、その中に含まれる URL 文字列を全て抽出するシェルスクリプトを書いてください。タグの属性値として記載されているものもタグの間に挟まれているものも全てです。

三、巷で「アセンブラ短歌」というものが流行ってるらしいですね。それに対抗して、シェル俳句（川柳）を募集します。基本ルールは次のとおりです。

イ、5+7+5 で 17 個のコマンドをパイプで繋ぐ（5、7、5 個ごとに改行していると望ましい）

ロ、必要に応じてその句の意味の解説を付ける

投稿のしかた

お題への解答は、お名前（ペンネーム）、解答したいお題番号、解答スクリプト、簡単な補足の四点セットで下記の宛先へ。一人何問でも何個でも解答可！

尚、次回締め切りは**５月１２日（月）午前０時**とします。雑誌は月刊化しましたけど、本コーナーは今までのペースでゆる～く続けていきます。約二か月後の締め切りまでじっくり考えてください。それまで何度でも解答の投稿と修正を受け付けます。

お題もどしどし送ってくださーい

お題の投稿も大募集。こっちは締め切りなしでずーっと募集中。そして、考えてくれた方にも段位を授与します。自分で出題して解答するのも、OK ！

投稿先

どちらも投稿先は、mag@usp-lab.comです。面白ければ（ワリと）何でもあり！　じゃんじゃん投稿待ってます！

上記のサービス内容は 2014 年 3 月現在のものです。
最新の情報は弊社ホームページにてご確認ください。

ISBN 978-4-9048-0706-4
C3455 ￥500

9784904807064

USP 研究所
定価（本体 500 円＋税）

1923455005002