月刊 USP MAGAZINE
Shell Script Is The Supreme Glue Technology
FOR THE SOPHISTICATED SHELL SCRIPTERS

世界で唯一の

シェルスクリプトマガジン

よりぬき版

2014
10 October
Vol. 18
¥500

特集

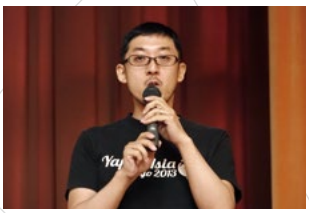
インフラエンジニアのお仕事

シェル芸勉強会後追い企画

Haskellでやってはいかんのか？

人間とコンピュータの可能性

CONTENTS



月刊 USP MAGAZINE
世界で唯一の
シェルスクリプトマガジン
2014 Octobar vol.18

神無月

今月の特集は「インフラエンジニアのお仕事」。

彼らの苦労は知る人ぞ知る。今回、彼らにスポットを当ててみました。

特集企画

インフラエンジニアのお仕事

がんばらない**運用**を目指して

「教えて先輩♡サーバー運用お助けTIPS」では日常のサーバ運用に関するTIPSを掲載しています。ところで皆さん普段はどれだけ手を動かしていますでしょうか?コンピューターにできることはコンピューターに任せ、ルーチンワークをいかに省力化するか、障害によるサービス影響をいかに少なくするか、つまり「極力手を動かさないようにするか」はエンジニアの腕の見せ所だったりします。とかく日本人は苦労を美徳としがちですが、エンジニアたるもの楽をする目的以外の苦労はしてはいけません。では、運用を楽にするにはどうしたらよいのでしょう?答えは「正しい設計をする」に尽きます。行きあたりばったりで作ったネットワークやサーバーは、障害発生の確率が上がるだけでなく、障害発生時にも原因特定や解決が困難になったりします。

ここでは、サービスを新規に立ち上げる場合どんな設計を行うかの一例を紹介します。現場によってより細かな工程に分割されたり、用語の定義が異なることもありますが、おおよそは表1に示す内容が網羅されているかと思います。

がんばらない**運用**を目指して

表1　設計工程の一覧

要件定義

機能として実装するべき項目や性能、セキュリティなどの要件を明確にする。

基本設計

要件定義をもとに機器構成や機能概要をまとめる。画面や帳票などの操作やデータの入出力、生成されるデータの概要やネットワーク機器の構成図などの仕様をまとめる。

詳細設計

基本設計で定義された仕様をプログラム単位に分割したり、サーバーやネットワーク機器のパラメーターを決定する。

運用設計

運用業務として行うべき項目を定義する。監視設計や拡張設計も運用設計に含む。運用業務とは何か？については本文にて記載する。運用設計書をもとに運用手順書などのドキュメントが作成される。

表1　設計工程の一覧に記されている順に設計書ができあがり、詳細設計と運用設計の間に実際の構築作業が行われる(または、構築作業と運用設計が同時進行する)ようなスケジュールが一般的ではないでしょうか。これが新規サービスではなく既存システムの更新である場合には、データ移行に関する設計も行うことになります。

要件定義においては、大きく「機能要件」「非機能要件」の2つを定義します。機能要件の一例は「一般ユーザーはショッピングサイトに表示されている商品をクリックして買い物を行うことができる」「会員ユーザーは買い物をするとポイントが貯まる」など、サービス内容に直結する機能を指します。一方で「サーバー1台に障害がおこってもサービスに影響が出ないこと」「同時接続数1000でもシステムがダウンしないこと」などの性能要件的なものや、「ショッピングサイトはSSLで保護されていること」「カード情報を扱うため、PCIDSS Ver3.0に準拠すること」などセキュリティに関する要件は、非機能要件として定義されます。
サーバー・ネットワークエンジニアは主に非機能要件かつネットワーク機器やOS、ミドルウェアにて実装される内容を担当することになるでしょう。

基本設計や詳細設計ですが、おおよその内容としては、構築を行うのに必要な要素が漏れ無く記載されていることが最低条件です。要件定義書から設計への落とし込みに過不足がないか、項目ごとに矛盾がないかなど抑えるべきポイントはありますが、誌面に限りがあるため、ここでは省略します。

運用設計は、サービス開始後に行うすべての作業について記載されることになります。具体的には、表2に示す項目がその一例となるでしょう。

表2　運用設計書に記載する項目(例)

業務に関する項目	運用スケジュール(24時間365日運用か、あるいは平日の日中帯の運用か、メンテナンス時間帯)、運用体制、責任分界点、通常作業内容(cronなどのジョブとして自動化するものも手動オペレーションするものも、一旦すべてリストアップする)について定義する
ジョブに関する項目	ジョブ管理内容一覧
セキュリティに関する項目	個人情報などの保護対象となるデータや保全対象となるデータを定義したり、OSやソフトウェアの更新、ファイアウォールやアンチウィルスソフトの運用などについて定義する
ログに関する項目	ログファイル一覧、ログ保存期間(ローテーション頻度、世代数)、ログ監査に関する内容について定義する
バックアップに関する項目	バックアップ取得対象、頻度、バックアップ取得方法、保存先、バックアップ保存期間、リストア方法について定義する
監視設計	監視や稼働統計の項目、閾値、閾値超えまたは復旧時のアクション、SLA(サービスレベル方針)について定義する
障害時運用設計	障害時の運用パターン(片系運用や縮退運転など)、影響範囲、対処方法について定義する
拡張設計	リソース逼迫時のスケールアップ、スケールアウト基準および方針について定義する
その他	作業用アカウント一覧、パスワードに関するルール、名前解決や時刻同期について etc...

目次だけでこれだけのボリュームなので、1つ1つ解説するとなるとそれこそ本1冊書けるくらいの内容になってしまいますが、運用設計で決めるべき項目を思いつく限り挙げてみただけでもこんなにあるということは、これらが曖昧なまま運用がスタートすると、トラブル時だけでなく通常運用作業において「迷い」が生じ、不本意なインシデントがおきてしまうことにもなりかねません。

それでは、運用ドキュメント(運用設計書、運用手順書)の量が多ければそれで良いかといえば、答えはNoです。必要なことがどこに書いてあるのか探しにくいと、見落としや誤解が生じてしまいます。運用ドキュメントに限らず、プロジェクトで作成したドキュメントは、どこに何が書いてあるのかを明確にし、内容についてもドキュメント間で矛盾や齟齬が生じていないか、読み手の読解力に過度に依存する内容でないか、暗黙知をなるべく形式知にできているかを確認するようにします。「運用でカバー」のカバレッジが上がれば上がるほど、運用チームは疲弊していくということになるので、運用設計においていかに曖昧さを排除するか、ひいては要件定義においてどれだけサービスを理解するかにかかっていると言えましょう。

とかくドキュメント系の作業は「エンジニア」に好まれない作業になりがちですが、プロジェクトメンバー間やステークホルダーとの意思疎通を円滑にするには欠かせません。また、設計根拠は決して事実のみが基になるのではなく、良くも悪くも人に依存することが往々にしてありがちです。それ自体は決して褒められたものではありませんが、せめて暗黙知のまま運用でカバーするのではなく、途中からプロジェクトにアサインされたメンバーでも短期間で既存メンバーにキャッチアップできるような粒度で書かれていることが理想ではないでしょうか。

そしてもう1点、エクセル方眼紙のように、罫線や方眼紙のマスなどの「本来のアウトプットの品質として評価されるべきではない」箇所で多大な工数を割くことは、あまり良いイメージを持たれないことが多いですが、何かしら理由があってドキュメンテーションツールが選定されていることも理解しましょう。それよりも大事なのは、冒頭にも記した通り、どれだけ正しい設計ができているかこそが評価の主軸であるべきです。理不尽や無理解を憎んでツールを憎まず、弘法筆を選ばずです。

現場視点からの運用方法論については、少し古い資料ではありますが、運用設計ラボ合同会社の波田野裕一氏(USP Magazineにも連載がありますね!)がthinkitに寄稿している記事(http://thinkit.co.jp/book/2010/12/02/1902)が参考になります。今回の特集では運用ドキュメントの目線でどう改善していけばよいかを書いてみましたが、運用業務そのものをどう見なおしていけばよいかのヒントになるでしょう。

1つのサービスやプロジェクトに長く携わっていると、技術的なスキルアップは(Webなどの情報によって)行うことができても、設計スキルに関しては、本当にこれでよいのか?という比較がなかなかできないのも事実です。しかし、案外周囲を見渡してみると同じような悩みを持つエンジニアは少なくないはずなので、可能な限り勉強会やカンファレンスに顔を出して仲間を募り、業務情報などの漏洩には気をつけつつも設計スキルについてケーススタディを持ち寄って勉強会を開いてみるのもよいかも知れません。

written by 濱田康貴

教えて先輩 サーバー運用お助けTips

4回目 連載番外編

サーバーエンジニアの業務にどうしても欠かせないもの1つに「障害対応」があります。しかし、どこの会社でも研修でプログラミングやサーバ・ネットワーク構築を教わることはあっても、障害対応だけはなかなか体系的に教わる機会がありません。あなた自身が関わる業務が自社サービスなのか保守運用代行なのか、お客様との契約内容、外注先の有無によっても、どこまで対応するべきなのかの責任分界点が異なるため、一般的に決まった正解がないのが障害対応業務の特異性でもあるので、教える側もなかなか教えにくいという事情もあるかと思います。今回は1人でも多くの「障害の矢面に立つ」方に近い立ち位置で、障害対応について参考になることを書いていきたいと思います。

エピローグ（障害原因は意外と少ない?）

「なあ@nullpopopoよ、WEBサーバの障害対応って実は簡単なんだぜ?原因は❶ハードが壊れるか❷大量のアタックが来るか、❸デプロイされたプログラムがおかしいかの3つしかないんだ。」と、ある日@orz001氏(某大手ホスティングサービスでサーバ運用していた方)が語ってくれました。ハードは予備機と交換すればよいですし、大量のアタックはファイアウォールの機能でブロックすればよく、プログラムがおかしければ一旦公開を止めて直せばよいので簡単ですね!というわけで、今月の連載はおしまい・・・というのは冗談です。おっしゃる通り設定不備や操作ミスなどの人為的な障害は別として、外からの障害はだいたいこれら3つの原因に大別されます。

少し補足すると、ハード障害はどんなサーバーでも避けがたいものなので、ダウンタイムがどれだけ許容できるかなどの要件や予算に従い、あらかじめ冗長化を行ったり、予備部品を確保しておいたりします。物理サーバーの運用が嫌だ!という動機でVPSやクラウドに移行するという考えもアリでしょう。

大量のアタックも、サーバーがインターネットに繋がっている以上これも避けがたく、あなたが管理しているサイトが誰から恨みを買ったりしているわけでもないとしても、中小サイトであってもそのリスクはつきまといます。有効な対策としてはファイアウォール(パケットフィルタリング)やWAF(Web Application Firewall)の導入になりますが、常に最新の脆弱性情報を入手するなど、ある程度の人的リソースも必要です。

ホスティング業者によっては、ファイアウォールの運用を代行してくれるところもありますので、検討してみるとよいでしょう。

最後のプログラムがおかしいパターンですが、サーバーエンジニアは最低限、アクセスログやエラーログから原因を特定しなければなりません。また、自らアクセスしてみて「どのプログラム(実行ファイルやスクリプト)に何がおこっているか?」を検証することもあるかと思います。サーバー上で公開しているプログラムがすべて自分たちの内製であれば、誰がどう対応すればよいかのフローも自ずと決まってくるでしょうし、解決までの時間も短いでしょう。

CMSソフトやそのプラグインなどは管理画面から簡単にアップデートできますが、自分たちで開発したものでない場合は、バグを踏んでしまったら障害期間が長期化する可能性もあるでしょう。なので、予めコンテンツとDBのバックアップをとっておき、万が一表示がおかしくなった場合にはすぐロールバックできるようにしておくことをおすすめします。また、こうした運用手順においていかにダウンタイムを短くするかを考えることは、サービスやシステムを理解することに繋がりますし、単に手順をなぞるよりも大きくスキルアップできるチャンスだったりします。究極には、既成のCMSに不満を抱いて、自らCMSを開発するという選択肢もありますので、ご興味を持たれた方は是非「フルスクラッチから1日でCMSを作る　シェルスクリプト高速開発手法入門(USP研究所監修 上田隆一、後藤大地　著　発行:株式会社KADOKAWA)」を一読してみてください(※1)。

障害対応のゴール

顧客や上司からは、一刻も早い障害復旧を求められることもあるでしょう。しかし、安易に「デーモンやOSを再起動すれば直るから」といって、再起動に逃げるのはいけません。障害発生時の状況を破壊してしまうと、後から原因を追求することが非常に困難となってしまいます。これはすなわち、原因もわからずいつ再発するかわからないという危険性を内包するに等しいのです。いたずらに時間をかければよいというわけではありませんが、障害原因の特定と根本的な解決策の実施を再優先にしましょう。もし原因不明な状態が長時間続くようであれば、早いうちに複数のエンジニアで対応にあたることをおすすめします。また、その際に最低1人は、5w1hを簡潔に時系列で記録する役割であると、なお良しです。

また、ApacheやNginxなどのhttpサーバーが出力するアクセスログには、ステータスコードが記録されることがほとんどです(※2)。また、エラーログは、プログラムが出力するより詳細なメッセージが記録されることもあるので、「ログは口ほどに物を言う」の精神で、真っ先にどちらも確認しておく癖をつけておきましょう。

障害対応ロールプレイ (いきなりサーバー管理者になっちゃったぞ)

さて、あなたがたった今サーバー管理者になって、前任者からSSHログイン情報とrootのパスワードを受け取ったばかりなのに、顧客から「http://nullpopopo.blogcube.info/ が見えないぞ」とクレーム電話が入って対応しなければならないとしましょう。そんな無茶苦茶なと思われるかも知れませんが、案外サーバー管理者にとってはこの手の話は1度や2度じゃないものです。顧客は怒りのあまり、すぐに電話を切ってしまいました(※3)。なお、本ロールプレイではCentOS 6.5(x86_64)を想定しています。

たまたまかかってきた電話が非通知だったので、折り返し電話をかけて詳細をヒアリングすることもできず、前任者がちゃらんぽらんなため、ログイン情報以外のドキュメントもありません。泣きたい気持ちをグッと堪え、サーバーにログインします。しかし、WEBサーバーがApacheなのかNginxなのかもわかりません。ひとまず80番ポート(httpで用いるTCPポート)がListenしているかを確認します(リスト1)。

State列が「ESTAB(ESTABLISHの意)」となっているので、httpポートはListenしているようです。私を含めおっさん世代のために少し補足しますと、ssコマンドはnetstatの代替となります(net-toolsパッケージがメンテナンスされなくなり、今後はiprouteパッケージで提供されるコマンド群を使う必要があります)。次にファイアウォール(iptables)も穴が開いていることを確認しましょう(リスト2)。

ここまでの切り分けで、このサーバーは外部からhttpで接続できることがわかりました。さて、一体どのアプリケーションがhttpポートを待ち受けているのでしょう。ある通信ポートをどのアプリケーションが待ち受けているかは、lsofコマンドで確認します(リスト3)。

リスト1　80番ポートをListenしているかの確認 (接続元IPアドレスは例示のため「あり得ない」数字に置換加工しています)

```
[hamada@www2201gi ~]$ ss -n | egrep '(^State|:80)'
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43868
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43869
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43861
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43870
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43866
ESTAB      0      0      133.242.174.215:80               123.45.678.9:43867
```

リスト2　ファイアウォールの確認

```
[hamada@www2201gi ~]$ su -
パスワード:
[root@www2201gi ~]# iptables -L -n | grep 80
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80
```

リスト3　通信ポートとプロセスの確認

```
[root@www2201gi ~]# lsof -i:80
COMMAND  PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nginx   1329  root   18u  IPv4  10053      0t0  TCP *:http (LISTEN)
nginx   1330 nginx   18u  IPv4  10053      0t0  TCP *:http (LISTEN)
nginx   1331 nginx   18u  IPv4  10053      0t0  TCP *:http (LISTEN)
```

上記のように、lsofコマンドに引数「-i:<ポート番号>」をつけることで、rootユーザがPID1329番で待ち受けているかがわかりました。先頭のCOMMAND列に「nginx」とあるので、NginxがWEBサーバのようですね。顧客からの接続状況(またはエラーの有無)をログで確認したいのですが、そもそもこのサーバーにインストールされているNginxがrpmパッケージからインストールされたものなのか、ソースコードからコンパイルしてインストールしたものなのかわかりません。ここでもlsofが大活躍です(リスト4)。

リスト4　Nginx実行ファイルがどうやってインストールされたかを確認する

```
[root@www2201gi ~]# which nginx
/usr/sbin/nginx

[root@www2201gi ~]# lsof -p 1329 | egrep '(^COMMAND|nginx$)'
COMMAND  PID USER   FD   TYPE             DEVICE SIZE/OFF    NODE NAME
nginx   1329 root  txt    REG              252,3  1303512 1049447 /usr/sbin/nginx

[root@www2201gi ~]# rpm -qf /usr/sbin/nginx
nginx-1.7.0-1.el6.x86_64

[root@www2201gi ~]# yum list nginx | grep ^nginx
nginx.x86_64                            1.7.0-1.el6                     @CentALT

[root@www2201gi ~]# rpm -ql nginx | grep conf
/etc/nginx/conf.d
/etc/nginx/conf.d/geo.data
/etc/nginx/conf.d/ssl.conf
/etc/nginx/conf.d/upstream-fair.conf
/etc/nginx/conf.d/virtual.conf
/etc/nginx/fastcgi.conf
/etc/nginx/fastcgi.conf.default
/etc/nginx/nginx.conf
/etc/nginx/nginx.conf.default
/etc/sysconfig/nginx
/usr/share/doc/nginx-1.7.0/nginx.conf.mod_zip
```


リスト3を少し補足すると、ここで実行したlsofコマンドで、Nginxがhttp接続を待ち受けていることを確認しましたが、USER列がrootになっている1329番プロセスが親プロセスであることも暗に示しています。これはTCPやUDPの0～1023番ポートまでを「特権ポート」といい、UNIX系OSではrootしかバインドできない(http接続は80番でrootユーザが待ち受けるけど、リクエストはnginxユーザで実行する子プロセスにforkして、子プロセスからクライアントに返答させる)ので、必然的にこれが親プロセスだとなるわけです。

リスト4では、Nginxの実行ファイルがrpmパッケージからインストールされたものかどうかを確認しています。whichコマンドでnginxコマンドがどこにあるかを確認し、次にlsofコマンドの「-p」オプションで、PID1329番のNAME列と一致することを確認します。そしてこれがrpmファイルからインストールされたものかどうかをrpmコマンドの引数「-qf」で確認し、yumコマンドでどのリポジトリからインストールしたかを表示します。最後に、rpmコマンドに引数「-ql　nginx」を与え、grepで「conf」文字列を含むものを絞り込みます。そうすると、/etc/nginx/conf.d　ディレクトリの配下に設定ファイルがあり、そこにログに関する記述があるという当たりをつけることができます(リスト5)。

リスト5　設定ファイルの当たりをつける

```
[root@www2201gi ~]# ls -l /etc/nginx/conf.d
total 2584
-rw-r--r-- 1 root root      1467 Apr 25 08:33 001_LB_WEB_NULLPOPOPO.conf
-rw-r--r-- 1 root root      1707 Apr 25 08:31 101_WWW_WEB_NULLPOPOPO.conf
(略)
-rw-r--r-- 1 root root      2582496 Apr 26 01:13 geo.data
-rw-r--r-- 1 root root      466 May 13  2008 ssl.conf
-rw-r--r-- 1 root root      370 May 13  2008 upstream-fair.conf
-rw-r--r-- 1 root root      283 May 13  2008 virtual.conf

[root@www2201gi ~]# egrep '(access_log|error_log)' /etc/nginx/conf.d/*NULLPOPOPO*
/etc/nginx/conf.d/101_WWW_WEB_NULLPOPOPO.conf:    access_log   /home/logs/nullpopopo.blogcube.info/access_log   main;
/etc/nginx/conf.d/101_WWW_WEB_NULLPOPOPO.conf:    error_log   /home/logs/nullpopopo.blogcube.info/error_log;
```

しかし、ここまで手間をかけずとも、最初のlsofコマンド(リスト4にて実行)で以下のようにgrepしてあげれば、実は当たりがつくのですが、たまたまログファイル名に「log」がついているから気づけるだけのことでもあります(リスト6)。

リスト6　実はこれでも当たりをつけられちゃう

```
[root@www2201gi ~]# lsof -p 1329 | egrep '(^COMMAND|log)'
COMMAND   PID     USER   FD      TYPE    DEVICE SIZE/OFF      NODE NAME
nginx     1329    root   2w      REG     252,3                19842 1704992 /var/log/nginx/error.log
nginx     1329    root   4w      REG     252,3                19842 1704992 /var/log/nginx/error.log
nginx     1329    root   5w      REG     252,3                25968363 1704987 /var/log/nginx/access.log
nginx     1329    root   6w      REG     252,3                289656 2497917 /home/logs/nullpopopo.blogcube.info/access_log
nginx     1329    root   7w      REG     252,3                9582 2497952 /home/logs/nullpopopo.blogcube.info/error_log
(略)
```

さて、そんなことをしているうちに、先ほどの顧客より催促の電話がかかってきました。

顧客「ところで http://nullpopopo.blogcube.info/ は繋がるようになったのかね?」

ぼく「恐らく大丈夫だと思いますので http://nullpopopo.blogcube.info/detarame.html を表示してもらえますか?」

顧客「『見つかりません ごめんなさい。リクエストされた投稿が見つかりませんでした。検索すれば見つかるかもしれません。』と表示されたぞ!」

ぼく「(少なくともWordPressがエラーを返してあげたな、よしよし)すみません、最初に障害に気づかれたのはいつ頃でしょうか?」

…(略)

電話がかかって来た時、実は顧客にカマをかけていました。そこそこアクセスのあるサイトの場合、普通にトップページへアクセスしてもらっても、顧客の接続環境がわからないと他のログに埋もれてしまいます。そこで、わざとデタラメな(404エラーになるような)URLを顧客に表示させて、ログに顧客のIPアドレスを目立つように残します(リスト7)。

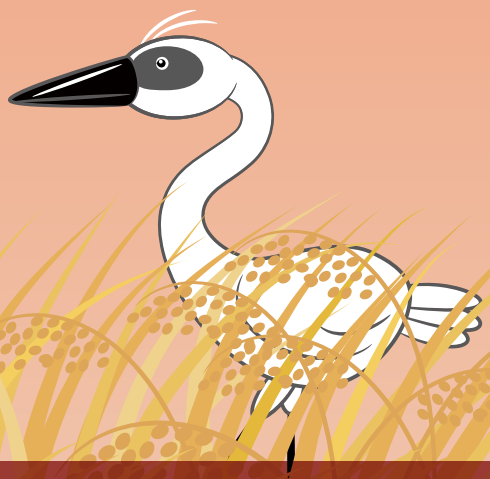
リスト7　デタラメなURLの表示をお願いした結果 (接続元IPアドレスは例示のため「あり得ない」数字に置換加工しています)

```
[root@www2201gi ~]# tail -f /home/logs/nullpopopo.blogcube.info/access_log | grep detarame
123.45.678.9 - - [23/Jul/2014:01:17:25 +0900] "GET /detarame.html HTTP/1.0" 404 4449 "-" "Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36"
```

少なくとも、今電話でやり取りした時点で顧客からWEBサーバへ接続できたことが、WordPressが生成した404エラー画面(画面に表示された文言で判断)が表示されていることで確認できました。アクセスログでも、ステータスコードは404を示しています。その後、ログの精査を行い、顧客が繋がらないと申告してきた時間帯は、他の回線から繋がっていることが判明したこと、その他エラーログに深刻なエラーがなかったこと、前後の時間帯で同じ接続元IPアドレスから接続ができていることを説明し、穏便にお引き取りいただくことができました。

今回のような障害発生時において、詳細をヒアリングしたいこちらの気持ちはお構いなしに、顧客は大抵お怒りモードです。少なくとも「答えが欲しい」と思っている顧客に対して「当時の状況は?今は?」とこちらからあれこれ質問をするのは、得策ではありません。このように、障害対応は時としてこのような胆力を求められることもありますが、これは「会話の主導権をこちらが握る」ことが目的だったりしますので、案外バカにできないスキルだということも頭の隅に置いておくとよいでしょう。

いかがでしたでしょうか。障害対応は、MSP事業者でない限り、それ自体がお金を生み出す業務ではありませんので、なかなか会社から評価されにくいかも知れません。また、顧客や上司からの「詰め」に心が荒むこともあるでしょう。しかし、こうした場数を乗り越えたぶんだけ、必ずタフになれます。そして、インフラエンジニアとしてスキルアップする最大のチャンスでもありますので、是非積極的にトライしてみましょう。



インフラエンジニアのお仕事

※1:　実はこの本のプロローグに私が登場しています。

※2:　httpのステータスコードは「Hypertext Transfer Protocol (HTTP) Status Code Registry」
http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml でIANAが一覧を公開しています。

※3:　ただし、振りかかる無茶振りをすべて真正面から受け止めていては身体と心がいくら頑丈でももちません。1人で悩まず、上司や仲間に相談しましょう。

July Tech

さる6月22日(土)に、JTF2014: July Tech Festaという**インフラ系の話題を中心としたエンジニアイベント**が産業技術大学院大学(品川シーサイド)で開催されました。拙文ながら、当日の模様をレポートしたいと思います。
2013年に続き、第2回目の開催となる今回のテーマは**「インフラ・テクノロジーマップ 2014 ～ 明日 楽をするために、今日 自動化を頑張るエンジニアの集い ～」**です。ハンズオン、ランチなども含めると7トラック27セッション、来場者数は約470名と、お祭りと呼ぶのにふさわしい盛り上がりでした。
以下、ざっくりですが当日の見どころをご紹介します。

written by 鍋島理人

July Tech Festaとは?

クラウドの普及とともに、ITの技術トレンドはすさまじい速度で移り変わり、技術習得と情報交換、人脈構築の必要性が高まっています。これらのニーズに答えながら、ITに携わるエンジニアの知的興味を満足させるためのエンジニアのお祭りです。ITインフラ関連エンジニアと、ソフトウェア開発エンジニアの両方を対象にしています。

基調講演

基調講演はServerspecの開発者で、元paperboy&co.(現GMOペパボ)の宮下剛輔氏によるセッション「Serverspecに見る技術トレンドを生み出すヒント」です。Serverspecとは、サーバーの状態をテストするフレームワークで、文字通りRubyによるテストフレームワークRSpecをベースにしています。最近のトレンドでもあるテスト駆動インフラ／インフラCIは、宮下氏自身も2007年頃から同様の構想を持っていたそうです。Puppetによる構築自動化の後、構築後のチェックを自動化する目的で、当初はAssurerというツールを自作しました。しかしServerspecと違って、振る舞いテストのツールであったことや、様々な機能を盛り込み過ぎ、複雑なツールになってしまいました。Serverspecほどコンセプトが明確でなかったことや、宮下氏が現場から離れPuppetにも触らなくなってしまったため、次第に宮下氏自身も使わないツールになってしまいました。その後、再びPuppetを使い始めた宮下氏が、既存のテストツールで良い物がないか探し始めたところ、Chefは充実しているのにPuppetにはRSpec-puppetぐらいしかなく、しかもモジュールのテストにしか使えないという現状にぶつかりました。それなら自作してしまうか!ということで、同僚がLXCのコンテナをテストするためにRSpecで書いていたコードを参考に、一日で作り上げたものがServerspecだそうです。

その後Serverspecはrubygems.orgで30万ダウンロードを記録し、各種賞・メディアで紹介されるなど大きな成功を収め、さらに発展しつつあります。なぜServerspecは成功し、トレンドを生み出すことができたのでしょうか。宮下氏によれば、ソフトウェア開発で成功したプラクティスをインフラの世界に持ち込んだこと、最初から普及を意識した活動(ネーミング、キーワードを意識、英語でドキュメントを書くことなど)、シンプルかつ拡張可能であること、いろいろな人を巻き込むこと、既存のトレンドや影響力のある人にうまく乗っかることなどが理由です。
最後に宮下氏は、日本人にはOSS開発者が少ない現状に触れながら、技術的トレンドを生み出す近道はないこと、ただ自分がほしい物、面白いと思うものを作り公開し続けることの積み重ねが、素晴らしいものを生み出すことに繋がると力説し、以下の言葉で講演を締めくくりました。

> Shut the fuck up and write some code
> Openness is our driver for excellence
> 知之者不如好之者好之者不如樂之者

とにかくコードを書こうってことですね:D

Festaレポート

分科会

基調講演後は、各トラックに別れ、セッションやハンズオンが行われました。話題のDockerの解説・事例や、まだまだ情報の少ないSwiftの解説も行われるなど、インフラにとどまらずITエンジニア全般にとって有意義なコンテンツでした。ざっとテーマを書き出してみると、クラウド、ストレージ、ネットワーク、インフラ、Immutable　Infrastructure、オートメーション、CI/CD、サーバーテスト、DevOps、WebAPI/OpenAPI、オペレーション、プログラミング、機械学習、ビッグデータ、オープンデータ、SDN/SDS/SDI、Security、CTFと様々なエンジニアのニーズに応えようとしていることが分かりますね。

ランチタイム

各社のスカウティングセッションを聞きながらのランチタイム。ヘルシーなお弁当で話題の「渋弁」をはじめ、数種類のお弁当から自由に選べ、飲み物まで着いているという素晴らしいおもてなしでした。ランチタイム以外でも、常時おやつが用意されており、セッションに集中して疲れた脳の栄養補給もバッチリです。スカウティングセッションでは、ニフティやDMM.comラボなどがショートセッションを持ち、会場に集ったエンジニアに向けてアピール合戦を展開していました。年収のホンネ話や、事業にまつわるちょっとエッチな話題が出るなど、際どいトークが繰り広げられ、会場を湧かせていました。

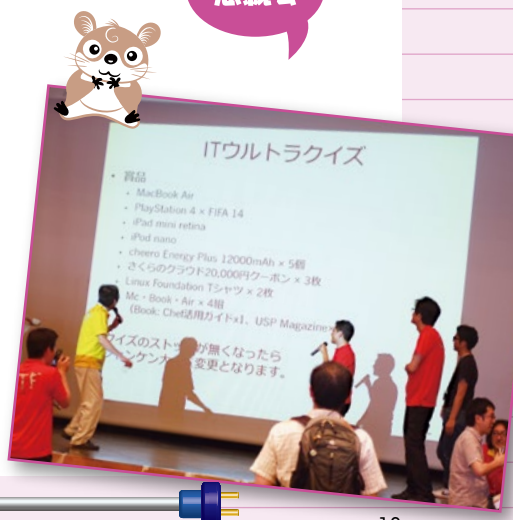
英会話カフェ

今や英語で技術情報を参照するのは当たり前の時代、そしてこれからのエンジニアは、英語で海外に向けて発信したり、人脈を築いていくスキルも必要です。先ほどの基調講演でも、宮下氏は英語の大事さに触れていました。いつまでも苦手意識を持っているわけには行きません。そこで、JTFの目玉企画の一つである英会話カフェに参加することにしました。最初はマグノリア・コンサルティングの渡部泰子氏による、英語の勉強法についてのセッション。日本語と英語を交えながら、「声に出す」ことを主体とした効率的な勉強法を学びます。恐れず、声に出してみることが大事との教えを頂きました。そして休憩を挟んで、あくしゅのBruce　McIntosh氏による実践編。英語による人脈作りのポイントを、英語で学びます。ここから先は原則日本語なしなので、冷や汗が走ります。とにかく声に出してみることが大事と自分に言い聞かせ、身振り手振りを交えて必死で拙い英語をしゃべりました。

懇親会

美味しいビールと食べ物に舌鼓を打ちながら、来場者・スタッフの垣根なく親睦を深めます。豪華賞品を賭けて「ITウルトラクイズ」が行われるなど、楽しい時間でした。

本レポートではあまりご紹介できませんでしたが、インフラ色の強いイベントにも関わらず、コードについてのセッションが多かったことが印象的でした。インフラエンジニアもコードを書かねばならず、アプリ開発者もインフラのことを知らねばならないのが、クラウド・自動化時代のトレンドなのかもしれません。両者の領域が接近しつつある現状を感じることが出来ました。そしてガイダンスやスタッフの親切さなど、イベントのおもてなし力の高さも印象的でした。July　Tech　Festaの名の通り、来年もまた初夏の季節に開催されるはずです。くつろいだ雰囲気の中で、知識と刺激と仲間を得ることができるイベントです。まだ行ったことのない方は、是非参加してみてください。



シェルスクリプト大喜利

第十四回

司会：『もっと吹く』編集長・みかん

皆さんこんにちは。三か月に一度のお楽しみ、シェルスクリプト大喜利のコーナーのお時間です。

もうすぐ 10 月ですね（この記事を作っているのは 8 月なんですけどね）。いやぁ〜すっかり涼しくなりましたねぇ（ホントか!?）。9 月 9 日のスーパームーンはホントに大きかったですねー（天気が悪いと見えないけどホントに見えたのか!?）。次のスーパームーンは……、え、9 月 28 日なんですか、もうすぐじゃないですか（いやそれは 2015 年だ！）。

というわけで、とにかくもうすぐ 10 月です。『10』といったらテンパズルですよ。ほら、あの電車の切符の 4 桁の数字を使って 10 を作る遊び。昔よくやりませんでしたか？今日はそれをシェルスクリプトにやらせようというお題に挑んでもらいます。他にもお題はありますが、これが**一番燃えましたな**。結果は読み進めてのお楽しみ。

ではまず本コーナー、シェルスクリプト大喜利のルールのご説明からまいりましょう。

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『半素数』を全て求めるシェルスクリプトを書いてください。

一問目の整数問題。sh 大喜利向けのお題を選ぶのが苦しくなってきた感じがするんですが、いくとこまでいきますよー。

さて今日求めてもらうのは半素数。これは因数分解すると 2 つの素数になる、という数です。つまり、それ自身が素数というわけではないんですが、でも factor とか primes といったコマンドの使い道はありそうですね。さて、投稿された解答は？

◎TSB_KZK さんの解答

```
1 echo "
2   scale=0
3   n=$1
4   for(x=2;x<=n/2;x+=1)
5     for(i=2;i<=n/x;i+=1){
6       if(sqrt(x) >= i) if(x%i==0)break
7       if(x <= i){
```

```
 8          jupper=sqrt(i)
 9          for(j=2;j<=jupper;j+=1) if(i%j==0)break
10          if(jupper < j)
11            print x, ¥" & ¥", i, ¥"¥n¥"
12        }
13      } " | bc
```

いやぁ、これはオシいんですよ。半素数を求めてもらいたかったんだけど、このコードだと答えとなる半素数を構成する素因数が表示されてしまいます。

うーん、これは本来なら残念ながら没にしてしまうとこなんだけど、今回の投稿で（積を求めるように改造してみても）最速だったんだよね。上手い具合に試行する組み合わせを削減していて…。掲載のみで段位授与はナシ。次のお題で健闘してるし勘弁してね。

◎K 師範の解答

```
1 #!/bin/sh
2
3 primes 1 $(($1/2+1)) |
4 while read p; do
5   primes 1 $(($p+1)) | xargs -I@ echo $p @
6 done |
7 while read a b; do
8   echo $(($a*$b))
9 done |
10 while read s; do
11   [ $s -le $1 ] && echo $s
12 done |
13 sort -n
```

こちらは一転してオーソドックスな解答。素数の列挙は 2 か所の primes コマンドに任せ、得られた素数ペアの積が指定範囲内かどうかチェックしています。

流れは大体解説した通りですな、師範、いつもありがとう！**一段授与**、只今三段。

◎gori.sh 師範の解答

```
1 #!/bin/bash
2 loopx <(seq $1) <(seq $1)|awk '$1%$2==0'|yarr num
  =1|awk '(NF==4&&$3^2==$1)||(NF==5&&$3^2!=$4)'|sel
  f 1
```

こちらはもう一人の師範、gori.sh さんの解答。Tukubai コマンドを駆使して短くしていますね。

これは、primes や factor コマンド、それに平方根等の高度な関数を使わずに素数を出しているんだけどコメントによれば「割り切れる数値が、自分自身の平方根のみか、自分以外で２つだけある場合」とのこと。なるほど！１つ目の AWK と yarr コマンドで１つの数値の因数を一行に列挙にしてるのか！そして列挙された因数の数を数えて判定するというわけか。うーん、ナイスアイデア！これは**二段授与**だ。現在三段。

◎ふじやんさんの解答

```
1 #!/bin/sh
2 seq 1 $1 | xargs -n 1 factor | awk 'NF==3{print
  $1}' | tr -d :
```

初参加者ですね。ありがとうございます。おぉ、これは短いですね！

なるほど、素因数分解するコマンド factor で出てきた素因数の数が 2 つだったものだけ表示するというわけか、シンプルだね。シンプルなおかげか、そこそこ速いし。よし、**初段授与**。これからもよろしくね！

◇ ◇ ◇

さて、それじゃお待ちかね。今回最も熱くなったこのお題行ってみましょう。

<二問目>
0-9 のいずれかの数字の引数が 4 つ与えられている。これらの数字に何らかの四則演算を施して 10 にできる場合のみ戻り値 0 を返すシェルスクリプトを作成してください。

はいっ、というわけで 10 月号に相応しくテンパズルです。1 桁の数字 4 つを四則演算して、10 にするというアレです。10000 通りのうちの 8147 通りが 10 にできるので成功率は高いのですが、中には難問だったり、どう頑張っても 10 にできないものがあったりして悔しい思いをすることも。なので、実際のところその 4 つの数字は 10 にできるのかどうかを、シェルスクリプトで判定しようというわけです。

ちょっと難しいお題だったかと思いきや、皆さん頑張って解いてくれましたよ。

◎東京 awker さんの解答

```
 1 #!/bin/sh
 2 echo $1 $2 $3 $4 |
 3 awk '{
 4   split("1234 1243 1324 1342 1432 1423 ¥
 5          2134 2143 2314 2341 2431 2413 ¥
 6          3214 3241 3124 3142 3412 3421 ¥
 7          4231 4213 4321 4312 4132 4123",o1);
 8   for (i=1;i<=24;i++) {
 9     split(o1[i],o2,"");
10     print $o2[1],$o2[2],$o2[3],$o2[4];
11   }
12 }' |
13 awk '{
14   op[1]="+";op[2]="-";op[3]="*";op[4]="/";
15   for (i=1;i<=4;i++) {
16     for (j=1;j<=4;j++) {
17       for (k=1;k<=4;k++) {
18         print $0, op[i], op[j], op[k];
19       }
20     }
21   }
22 }' |
23 awk '{
```

```
  print $1,$2,$5,$3,$6,$4,$7,"p";
  print $1,$2,$5,$3,$4,$6,$7,"p";
  print $1,$2,$3,$5,$6,$4,$7,"p";
  print $1,$2,$3,$4,$5,$6,$7,"p";
}' |
awk '{
  cmd = "echo ¥"" $0 "¥"|dc 2>/dev/null";
  cmd | getline ans;
  close(cmd);
  if (ans==10) { ret=1;exit; }
}
END {exit 1-ret;}'
```

おっとこれは久しぶりの東京 awker さん。しかし結論から言いますと、**残念!これ不正解**なんですよ。10000 通り試してみたところ、"0269" などの 10 にできない数の一部もできると判定されてしまいました。原因は dc コマンドの計算精度の考慮しなかったことですね。計算精度を考慮しないと解けないテンパズルの難問の一つに "1199" というのがあって、(1+1/9)*9 という有理数計算をしないと解けないものがあるんですけど、これは意外にも 10 にできると判定。どーやってるんだろうと思ったら、1+1/9+9 の 1/9 を 0 に四捨五入して、それが結果的に合っていたみたいです。

うーん、dc に着目したのはよかったんだけど。おしいねぇ。残念ながら段位授与ならずだ。現在六段。

◎gori.sh 師範の解答

```
#!/bin/bash
[ -z $4 ] && exit 0
tmp=tmp

echo a$1 b$2 c$3 d$4 | tarr > $tmp-a
echo '+ - * / r- r/' | tarr > $tmp-b
echo p > $tmp-p

loopx $tmp-a $tmp-a $tmp-a $tmp-a    |
sed '/¥([a-d]¥).*¥1/d'               |
tr -d abcd                           |
loopx - $tmp-b $tmp-b $tmp-b $tmp-p  |
tee $tmp-d                           |
dc -e 10k -f - 2> /dev/null          |
grep -q '¥(^9¥.9*$¥)¥|¥(^10$¥)¥|¥(^10¥.0*$¥)' ||
cat $tmp-d                           |
self 1 2 5 3 4 6 7 8                 |
dc -e 10k -f - 2> /dev/null          |
grep -q '¥(^9¥.9*$¥)¥|¥(^10$¥)¥|¥(^10¥.0*$¥)'
```

こちらは gori.sh 師範の解答。10000 通り試したところ……、**はい正解!!**さすが師範です。dc コマンドを使うアプローチは東京 awker さんと同じですが、こちらは dc に -e 10k オプションを付け、grep で端数判定をしてちゃーんと見てますね。しかも速い！総当たりで組み合わせを生成する loopx コマンド (Tukubai) を使っているものの、一時ファイルをうまいこと活用したり、交換法則の成り立たない減算と除算で dc のスタック順序を入れ替えるなどの工夫が随所に見られて、確かにこれは熱いコードですな。

いやぁ、抜かりない配慮と工夫で速さと正確さを両立したね。またまた素晴らしい、**二段授与**して五段だ。

◎TSB_KZK さんの解答

```
#!/bin/sh
nums=`echo $1 $2 $3 $4 | xargs -n 1 | sort | tr -d ' ¥n'`

res=`curl -s "http://ja.wikipedia.org/wiki/テンパズル" |
tr -d "¥n"                                             |
sed "s/<table/¥n<table/g"                              |
grep "¥"make10¥""                                      |
grep -o "<td>[0-9]*</td>"                              |
grep -c "$nums"`

if [ $res -eq 0 ] ; then exit 1; fi
echo "make 10!"
```

はい、一問目に続き投稿ありがとうございます。しかし、こ、これは……。

君、そういうことするかね。Wikipedia の答えを使うのはチートなのでボツね。……なーんてこと言わない言わない。面白ければ何でもアリよ、大喜利は！

インターネットを引けば答えがすぐ見つかるというのも、いい時代になったもんだよ。確かこういうのを**Web スクレイピング**って言うんだよね。三度目の正直で段位授与だ。**二段授与**して、七段。

◇ ◇ ◇

さて、最後はいつものヘンなお題。

<三問目>

名前付きパイプを作るコマンド mkfifo がありますね。でもアタクシ、なかなかうまい使い道を思いつきません。
そこで、あなたの知ってる mkfifo（名前付きパイプ）のウマい使い方を披露してください。

皆さん使ったことありますか？このコマンド。というか、名前付きパイプって知ってます？まぁ、名前付きパイプの存在は知っていたとしても、なかなか使い道無いですよね。そこであえてお題として募集してみたんですが……。うーん、やっぱり難しいみたいですね。投稿作品をいくつか紹介しましょう。

◎イタローさんの解答

```
$ mkfifo detonator
$ (cat detonator >/dev/null; echo 'Bomb!') &
```

投稿コメントには「echo 'Bomb!' の部分が rm -rf ~/* とかだったらきっとイヤでしょ？つっかえている

catコマンドをkillすると即座にBomb!するという**ブービートラップ**がありますからね」と書いてありました。

うーん…mkfifoの応用例では、こんな感じの「つっかえ棒を外すと事を起こす系」の投稿がいくつかあったけど一番陰湿な君のを採用したよ。確かにこれを安全に解除するのは一手間だ。catの親プロセスIDを探してkillしてから、catのプロセスをkillせんといかんよね、こういう具合に。あぁ、何て厄介な。

```
$ ps -o args,ppid,pid | grep "^cat detonator" | tr -d a-z | xargs -n 1 kill
```

mkfifoの役立つ例じゃなくてむしろ余計な例だけど、まぁいいか。**一段授与**。これで四段だよ。

◎たまさんの解答

```
$ mkfifo p1 p2
$ cat p1 | md5sum &
$ cat p2 | wc -c &
$ cat $file | gzip | tee p1 | tee p2 > $file.gz &
```

投稿コメントによれば、こちらはダウンロードファイル作成時に必要とされがちなファイルサイズとMD5値を、名前付きパイプでいっぺんに求められますよ、という投稿です。

うーん、アイデアはいいと思うんだけど、gzipしてできたファイルに後からmd5sumとwcをすればいいんじゃないかなぁ。いまいち実用的じゃないよねぇ。というわけで残念ながら段位授与無しかな。

◎shu-7さんの解答

```
$ mkfifo foo
$ ls -F foo
$ rm foo
```

最初これの何がいいのかさっぱりわからなかったんですが、コメントによればスマートフォンからターミナルにログインしている時、パイプ記号“|”を入れるのが意外と面倒で、こうやって“|”を表示させてコピペするのだそうです。

なるほど！ls -Fって確かにファイル名の後ろに“*”、“/”、“@”、“|”といった記号類を表示するねぇ。中でも“|”はパイプに多用するから、知ってると便利なのか。mkfifoの意外な実用性を知ったよ。よし**初段授与**！

◇　◇　◇

いやぁ、今回は極めて面白い投稿があった一方で段位授与せずに終わった投稿も多かったねぇ。

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「回文数」を列挙するシェルスクリプトを書いてください。単純なお題ですから、ひとヒネり利かせてくださいよー。（変わった解き方や、特定のコマンド使わない縛り、珍しいコマンド使うなどなど）

二、横10桁×縦10行でハイフン“-”が敷き詰められ、そのうちのいくつかがアスタリスク“*”で書き換えられたテキストがあります。これを入力データとし、マインスイーパーのように、自分のいるマスの周囲8マスにある“*”の数を数えて1個以上であれば数字に置き換えるシェルスクリプトを書いてください。

三、カレンダには日曜始まりのものもあれば月曜始まりのものもあり、世界を見渡せば土曜始まりなんてものもあります。しかし、UNIXのcalコマンドは日曜始まりでしかありません。

そこで、引数で0（日曜）～6（土曜）の数値を与えると、その曜日始まりで今月のカレンダーを表示するシェルスクリプトを書いてください、

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**11月10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。面白ければ何でもあり！じゃんじゃん投稿待ってます！

ISBN 978-4-9048-0712-5
C3455 ¥500

9784904807125

USP 研究所
定価（本体 500円+税）

1923455005002