ビックなデータで、どどん・どどん・どどんどん！

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 7 July Vol. 15
¥500

特集
愛されデータサイエンティスト

新連載
Linux女子部誌上講座
リナ女のたしなみ
～写真管理編～

金融システムにおける
UNIXの歩み

USP MAGAZINE

2014 July vol.15

Contents

コードレビュー Vol.4

CODE Review

USP 研究所技術研究員
written by 大内智明

前号に続き、本部・店舗間の発注・伝票処理に伴う連絡・通知をシンプルに実現する常駐型シェルスクリプトをご紹介します。この処理フローの一部を担う集積処理に今回はフォーカスします。

定期的にファイルを集積する

本部や店舗で行われる業務には、決まった締め日時があります。その時期になると本部や各店舗で多くのトランザクションが集中的に発生します。結果として、システム負荷が大きくなり、他システムへの連携に影響する場合があります。こうしたシステム負荷が大きくならないように、ユニケージ開発手法でよく使用するのが、前回登場した常駐型シェルスクリプトです。

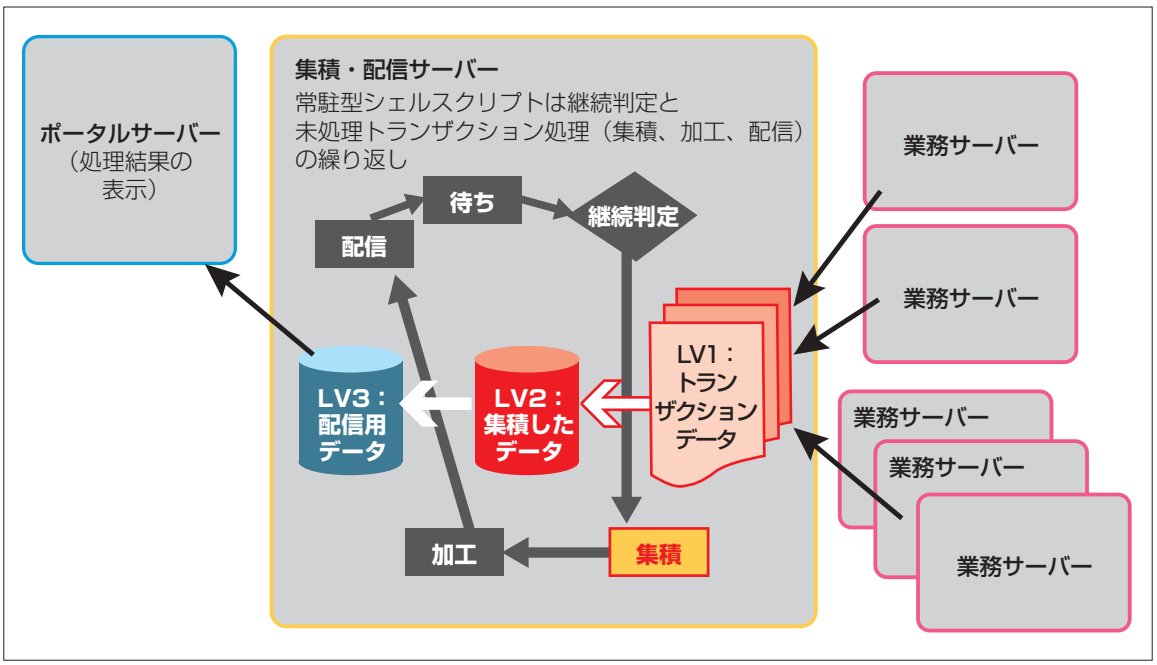
前回は、常駐型シェルスクリプトの全体像を説明しましたが、今回はその中の重要な一機能である集積処理部分のコードを見ていきます（図 1）。集積処理の役割は、各業務サーバーで発生した複数のトランザクションファイルを定期的に集め、1 ファイルにして保存することです。

技術的な概要

このシェルスクリプトは、発注処理・伝票処理といった各業務から発生するトランザクションデータ、すなわちユニケージ開発手法における LV1（システムに入力された生データ）を一定間隔ごとに集積して、LV2 ファイル（LV1 を変換・整形したテキストデータ）として保存する役目を果たします（図 2）。

発生するトランザクションには、実データファイルとセマフォファイル（→補足 1）の 2 種類があります。トランザクションを集積する（発生したトランザ

図 1　常駐型シェルスクリプトの処理フロー



クションを集める)際の注意点は、同一の実データファイルを重複して集積しない、ということです。

もし、重複してデータを集積すると不正確な処理結果が出力されることになります。そうならないように集積処理が終了すると、元のファイル名（未処理ファイル)をリネームしています。処理済みのセマフォファイルには、ファイル名に ZUMI（済み）という文字を付けて未処理のファイルと区別するのです。リネームはセマフォファイルのみに対して実施します。セマフォファイルのみをリネームする理由は２つあります。

- セマフォファイルを、未処理（処理済み）の判定に使用しているため
- 実データファイルは LV1（生データ）なのでそのままの状態で残す必要があるため（→補足 2)。

【補足 1】ここで使用するセマフォファイルとは、実データファイルが作成途中に取り込まれることを防ぐために、トリガファイルとして実データファイルが完成した直後に生成されるファイルのことです。実データファイルとセマフォファイルは 1 対 1 で存在します。それは、データ取り込みに過不足がないようにセマフォファイルの件数だけ取り込み処理を実施していることが理由です。実データファイルとセマフォファイルのファイル名は同一にします（画面 3)。

【補足 2】ユニケージ開発手法では、LV1 データ（生データ）は消去せず保存します。サーバーのディスク容量が不足する場合には、バックアップサーバー等で保持します。プログラムは消去されても後から再生することができますが、データは一度失われると二度と再生できないからです。

コードの見どころ

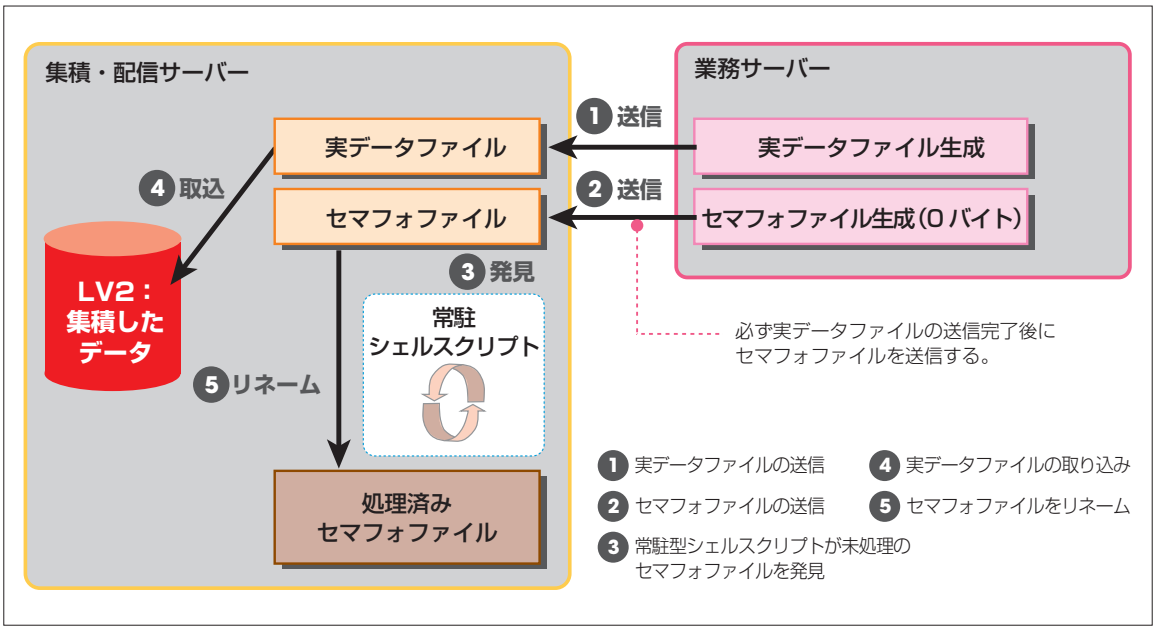
掲載したソースコードは、繰り返し実行される処理（集積･加工･配信･継続判定）の中に含まれている一工程（集積）です（図2)。各トランザクションファイルを集積してファイルをまとめる処理ですが、これはLV1からLV2を作成する処理に該当します。その部分を大きく3つのポイントに分けて説明します。

[1]前回集積した後に発生した新規の未処理ファイル（セマフォファイル）を抽出します。補足1で述べたように、セマフォファイル名は実データファイル名と同一にしているため、セマフォファイル名だけを抜き出して、実データファイルのフルパスを作成します（75～87行目)(→補足３)。

[2]重複してデータを取り込まないようにセマフォファイルを処理済みのファイル名にリネームします（98行目)。

[3] 集積したファイルをLV2として保存します（102～105行目)(→補足4)。

図２　集積・配信サーバーと業務サーバーの動作



リスト１　集積処理を行うプログラム

```
 1 #!/bin/ush -xve
 2 # システム名     : ＵＳＰシステム
 3 # サブシステム名: 連絡アラートの配信(日次)
 4 # 業務名         : 連絡アラート
 5 # プログラム名   : 連絡アラートＬＶ２の集積・通知
 6 # 備考(Usage)    : LV2MAKE.TIMES.ALERT_TSUCHI [YYYYMMDD]
 7 # シェル名       : LV2MAKE.TIMES.ALERT_TSUCHI
 8 # 作成日         : 2013/07/31
 9 # 会社名         : USP
10 # 作成者         : A.Asaba
11
12 #////////////////////////////////////////////////////////////////////////////
13 # 初期設定
14 #////////////////////////////////////////////////////////////////////////////
15
16 # 走行ログの記録
17 logd="${HOME}/LOG"                                                  # ログディレクトリ
18 logf="${logd}/LOG.$(basename $0).$(date +%Y%m%d)_$(date +%H%M%S)_$$"  # ログファイル名
19 echo  "${logf}" &> /dev/null
20 log 2> ${logf}
21
22 # パスの定義
23 PATH=/home/UTL:/home/TOOL:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:${PATH}
24 LANG=ja_JP.UTF-8
25
26
27 #-----------------------------------------------------------------------
28 # 変数の定義
29 #-----------------------------------------------------------------------
30 tmp=/tmp/$$-$(basename $0)_$(date +%Y%m%d%H%M%S)            # 一時ファイル
31 hostname="$(HOSTNAME)"                                     # サーバー名
32 semd="${HOME}/SEMAPHORE"                                   # セマフォディレクトリ
33 subd="ALERT "
34 rcvd=/home/RCV/DATA/ALERT
35 rcvsemd=/home/RCV/SEMAPHORE/ALERT
36
37 lv2d=/home/DATA/LV2
38 outd="${lv2d}/${subd}"
39
40 # エラー時の終了処理定義 (ushエラーハンドラ)
41 err ERROR_EXIT(){
42   echo "${hostname} $(basename $0)_${sday} ERROR $(date +%Y%m%d%H%M%S) ${logf}" >> ${logd}/UPCNT
43   touch ${semd}/$(basename $0).${hostname}.ERROR.${sday}
44   exit 1
45 }
46
47 # 処理日付の確認
48 sday=$(date +%Y%m%d)
49 # 引き数の確認
50 [ $# -eq 1 ] && sday=$1
51
52 # 現在時刻
53 stime="$(date "+%H%M%S")"
54
55 # 簡易YYYYMMDD日付チェック
56 if ! isdate ${sday} ; then
57   echo "Parameter DATE error:[${sday}]"
58   ERROR_EXIT
59 fi
```

画面１　echo でファイルリストを出力

echoでファイルのリストを出力すると1行に全てが出力される。そこで横型データを縦型に並び替えるコマンドtarrを使用すると、ls同様にデータは縦に並んだ状態で出力される。

```
$ echo /usr/bin/*
/usr/bin/411toppm /usr/bin/BMPtoRGB /usr/bin/DllPl
ugInTester
$ echo /usr/bin/* | tarr
/usr/bin/411toppm
/usr/bin/BMPtoRGB
/usr/bin/DllPlugInTester
```

lsは外部コマンドなので全件抽出できない場合がある。

```
$ ls /usr/bin/* | head
/usr/bin/411toppm
/usr/bin/BMPtoRGB
/usr/bin/DllPlugInTester
```

```
60
61 # 前回セマフォの消去
62 rm -f ${semd}/$(basename $0).${hostname}.*.${sday}  >/dev/null 2>&1
63
64 # 起動時刻の記録
65 echo "${hostname} $(basename $0)_${sday} START $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
66 touch ${semd}/$(basename $0).${hostname}.START.${sday}
67
68 ////////////////////////////////////////////////////////////////////////////
69 # データ処理部
70 ////////////////////////////////////////////////////////////////////////////
71
72 # 初期化
73 : > ${tmp}-lv2_data
74
75 echo ${rcvsemd}/${sday}/*.*.${sday}?????? |
76 tarr                                       |
77 ugrep -v '¥?'                              |
78 ugrep -v '¥*'                              |
79 xargs -J % find % -type f                  |
80 xargs basename                             |
81 # ファイルを変換
82 while read fname; do
83   rcvf="${rcvd}/${sday}/${fname}"
84   [ -e ${rcvf} ] || ERROR_EXIT
85
86   # ファイルの集積
87   cat ${rcvf} >> ${tmp}-lv2_data
88   # 1:アラートメッセージ番号      2:発信者区分              3:発信会社コード
89   # 4:問合せ先                  5:重要度                  6:種別
90   # 7:件名パラメータ1           8:件名パラメータ2          9:件名パラメータ3
91   # 10:発信日時                 11:作業期限               12:掲載開始日
92   # 13:受信会社コード           14:受信者ログインID       15:未読フラグ
93   # 16:業務キー1                17:業務キー2              18: 業務キー3
94   # 19:業務キー4                20:業務キー5
95   # 21:更新フラグ    22:適用日    23:ユーザID    24:入力日時
96
97   # セマフォを済みとする
98   mv ${rcvsemd}/${sday}/${fname} ${rcvsemd}/${sday}/${fname}.ZUMI
99 done
100
101 ## LV2の集積
102 if [ -s ${tmp}-lv2_data ]; then
103   mkdir -p ${outd}/${sday}
104   cp -p ${tmp}-lv2_data ${outd}/${sday}/${subd}.${sday}${stime}.${$}
105 fi
106
107 ////////////////////////////////////////////////////////////////////////////
108 # 終了
109 ////////////////////////////////////////////////////////////////////////////
110 # 終了時刻の保存
111 echo "${hostname} $(basename $0)_${sday} END $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
112 touch ${semd}/$(basename $0).${hostname}.END.${sday}
113 #ワークファイルの削除
114 set +x
115 rm -f ${tmp}-* > /dev/null  2>&1
116 set -x
117 echo "$(basename $0) exit 0"
118 exit 0
```

「コードの見どころ」および画面2を参照

echo でファイルリストを出力（画面1を参照）

セマフォファイル（画面3を参照）

ugrep は grep のラッパー。指定パターンが見つからなくてもエラーステータスは「1」でなく「0」となる。

[1] 未処理ファイルを抽出して1ファイルに集積（75～87行目）

実ファイル（画面3を参照）

[2] セマフォファイル（未処理）を処理済みに変更（画面3を参照）

[3] 集積したファイルをLV2として保存（102～105行目）

「コードの見どころ」および画面2を参照

画面2　開始・終了を記録するUPCNTファイル

```
$ grep LV2MAKE.TIMES.ALERT_TSUCHI ${logd}/UPCNT
.......
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 START 20131210085500 ← 開始 ┐
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 END 20131210085520   ← 終了 ┘ 処理時間20秒
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 START 20131210090020   開始 ┐
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 END 20131210090030     終了 ┘ 処理時間10秒
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 START 20131210090535   開始 ┐
USPSV01 LV2MAKE.TIMES.ALERT_TSUCHI_20131210 END 20131210090552     終了 ┘ 処理時間17秒
.......
```

画面3　セマフォファイルの生成

```
$ ls -l /home/RCV/DATA/ALERT/20131210/ALART_MSG.END.20131210090000
-rw-rw-r-- 1 usp usp 1120  12月 10日 /home/RCV/DATA/ALERT/20131210/ALART_MSG.END.20131210090000 ●----- 実ファイル

$ ls -l /home/RCV/SEMAPHORE/ALERT/20131210/ALART_MSG.END.20131210090000
-rw-rw-r-- 1 usp usp 0  12月 10日 /home/RCV/SEMAPHORE/ALERT/20131210/ALART_MSG.END.20131210090000 ●----- セマフォファイル

取り込み処理が終了するとセマフォファイルをZUMI付きに変更

$ ls -l /home/RCV/SEMAPHORE/ALERT/20131210/ALART_MSG.END.20131210090000.ZUMI
-rw-rw-r-- 1 usp usp 0  12月 10日 /home/RCV/SEMAPHORE/ALERT/20131210/ALART_MSG.END.20131210090000.ZUMI
```

各業務サーバーからのトランザクション（実ファイル）が生成された後に、0バイトのセマフォファイルが生成される

ユニケージ開発手法においては、開始・終了を記録するUPCNTファイルを用意します（65、111行目）。UPCNTファイルには、サーバー名・シェルスクリプト名・処理日・ステータス（開始／終了／エラー）・日時などの有用なデータを記録します。これらの情報から、シェルスクリプトの処理時間やCGIの場合には時間帯別アクセス回数などを算出することができます（画面2）。

さらにディレクトリ内にあるファイルを抽出して、ファイルリストを作成します（75行目）。ファイルを抽出する代表的なコマンドにecho、find、lsがあります。抽出する条件に「*」や「?」などを付けて実行した場合、echoはシェル（bash、ushなど）の内部コマンドであることから、件数が多くてもデータを抽出することができます。一方、findとlsは外部コマンドであるため件数が多すぎると「引数のリストが長すぎます」といったメッセージが出てすべて抽出できない場合があります。こうした違いを考慮して使い分ける必要があります。echoコマンドの使用例は画面1（P.28）をご覧ください。

【補足3】今回のケースでは複数のトランザクションファイルを1ファイルにまとめましたが、1ファイルに集積できない場合、1トランザクションを1ファイル（LV2）として保存する場合もあります。

【補足4】ユニケージ開発手法では、入力業務の種類や時系列（日時など）ごとにディレクトリを分けてデータ配置します。以下は、システムのディレクトリ構成の例です。

リスト2　ディレクトリ構成

```
                                  データ受信（LV1）
/home/RCV/DATA/ALERT/（処理日）/（実データファイル）
         /SEMAPHORE/ALERT/（処理日）/（セマフォファイル）
/home/DATA/LV2/ALERT/（処理日）/（LV2実データファイル）
                                  LV1の集積データ（LV2）
```

まとめ

セマフォファイルを上手に扱うことで、正しく実データを集積することができます。今回のようにセマフォファイル（トリガファイル）を使用しないで、実データファイルを作成途中で集積すると、中途半端な状態で取り込んでしまい、後々トラブルの種になる場合があります。

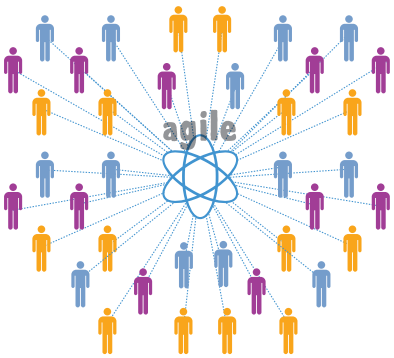
さて、次回は、配信用LV3データを他サーバーへ配信する処理について説明します。

組織文化を変える汗と涙の物語

アジャイル改善塾

「第三回 キックオフ」

written by 山海一剛



経営層の問題意識をとりまとめ、その対策として、すでに多くの会社で改善塾の指導をしていたIT業界出身の高木徹氏の指導を仰ぐ、ということで、社内の合意を取り付けた筆者。時はすでに3月。次年度の施策として前例のない取り組みをスタートさせるには、解決しなければならない課題が山積みでした。それは単に、「時間」の問題だけではありませんでした。

★塾生を集める★

さて前回、豊田マネージメント研究所の高木徹氏（現在、同社エグゼクティブ・コンサルタント）を招いて、社内にリーダーを育成する、というところまでお話ししました。ご存知の方もいらっしゃるかもしれませんが、高木徹さんのプロフィールを少しだけ Web から拝借すると―。2004 年より IT 系企業の赤字転落を防ぎ、ES（従業員満足度）の向上に貢献。このホワイトカラーの職場カイゼン活動実績が評価され、トヨタ自動車に出向。3 カ月でトヨタのグループ会社のソフトウェア開発の生産性を 20%以上改善。多くの企業の「カイゼン塾」で 1,000 人以上の塾生を輩出し 5,000 人以上のカイゼン経験者を育成……、という方です。

高木氏の指導スタイルは、週に 1 日、リーダー層を集めて「塾」を開催するというものです。リーダーとは、管理をしているだけではなく、自分自身が現場で業務に携わり、手取り足取り若手を直接指導している中堅社員をイメージしています。しかし、2 つの課題が横たわっていました。

まず、現場の「働き頭」を週に丸 1 日拘束することになります。とても部長やマネージャが許可するとは思えません。しかし高木氏の実績をみると『幹部候補生を毎週 1 日塾に出席させ、成果を上げている』とのこと。1 週間という短い期間内に「塾で学び、現場で実践し、結果を次の塾に持ち寄る」というサイクルを繰り返すことが最大の特徴です。そして私もこの点に、現場を変える可能性を見出していたのですから、「週に 1 回」はどうしても譲れません。ただ丸一日拘束することにしてしまうと塾生が集まらないだろう……板挟みです。結局、塾の開催は毎週木曜日午後、つまり毎週だけど半日ということにしました。これは私自身にとっても、中途半端な妥協という印象をぬぐえず、今でもこの判断が良かったのかどうか悩んでいます。

もうひとつの課題は塾生をどのように募集するかでした。悩んだ末に「肩書、部署、仕事の内容は問わないが、10 人程度のメンバーを率いている立場の人を」という表現で、本部長・部長・マネージャを通じて「塾生を選抜してもらう」という方針を掲げました。塾の主旨である自律性という意味では立候補制が理想なのですが、「塾の活動は仕事の一環である」ことを、本人にも上司にもしっかりと認識してもらうために、人事部から各所属長への人選指示という形にしたのです。各本部に向けて正式な人選依頼を行ったのは 3 月の下旬。さて何人ぐらい集まるだろうか。とても気になりました。

待っているだけでは心配なので、考えに共感してくれそうな人には「来年度からこんな制度を立ち上げようしているのだけど、もし上長から相談されたら、参加を希望してね」と説明して回りました。ある人には、出張帰りの新幹線の席で熱く語って勧誘。疲れているところにさぞかし迷惑だったことでしょう。でも、こ

の人はⅠ期の塾生として参加、しかも非常に熱心に活動してくれたのでした。

★高木氏のスケジュールが空いてない★

塾を毎週実施するためには、上記２つの課題のほかに別の壁がありました。正式決定した３月初めには、すでに高木氏のスケジュールが埋まっており、なんと２週間に一度しか時間が取れないとのこと！　これも、前回記事の最後に述べた、塾開講に至るまでに多々あったハードルのひとつです。それでも週１回の開催は譲りたくありません。結局、高木氏が来られない日は、私が講師として塾をファシリテートすることにしてしまいました。周囲に協力してもらいながらここまで来たのに、今さら私が逃げ出すわけにはいきません。私自身が高木氏から直接指導してもらえる機会を得たわけですから、無謀は承知の上で引き受けました。１週間おきに高木氏が訪れるので、来られた日に高木氏のそばで指導の仕方を学びながら次週にすべきことを指導してもらいました。それをもとに翌週、私が場をファシリテートする、という見習い講師としてのサイクルも回すことになったのです。

★命名！「アジャイル改善塾」★

最後に残ったのが名前です。他社の例では、単に「改善塾」とか、あるいは「自律改善塾」とか、それぞれの思いを込めた名前が付けられているようです。数年前からうちの会社は「アジャイル開発」というものに力を入れてきました。「アジャイル」とは、「機敏な」「身が軽い」を意味する形容詞で、機動力を重視したソフトウェアの開発手法のことなのですが、これについては今後詳しくお話ししましょう。

実はこのソフトウェア開発スタイルは米国発祥でありながら、車というハードウェアを作るトヨタ生産方式から大きく影響を受けています。短い期間で計画・実践・振り返りを繰り返すこと、開発メンバーの自律性や持続的改善活動を重視することなど、改善塾と共通の部分がたくさんあります。改善塾もトヨタの人づくりがルーツであることを考えれば、むしろ当然のことです。そこでやっと「アジャイル改善塾」という名前を付けるに至りました。

「アジャイル改善塾」は、ソフトウェアの開発手法を学ぶ塾だと思っている人が今でもいるぐらい、紛らわしいネーミングなのですが、塾で学ぶことで両者に共通する普遍的な価値観に気づいてほしいと考えたのです。

記念すべき第１回目のアジャイル改善塾で行われたワークショップの様子

★キックオフ★

会議室の予約やら、社長にスピーチをお願いするやら、慌ただしくしているうちにどんどん時間が流れ、とうとうキックオフの日がやってきました。2013 年 4 月 16 日です。集まった塾生は 15 人。ソフトウェア開発部門だけではなく、データセンターの管理部門、営業事務、総務、人事、経営企画など、実に様々な部門から塾生が集まりました。これから毎週木曜日、何が起こるのでしょうか？ 私自身も期待と不安が入り混じった思いでした。［続く］

第４弾 @自動販売機

今月の 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

エクストリームエンジニア

eXtreme Engineer

interviewed by 太田智美

自動販売機、それはぼくのロマン

……と、あるエンジニアは言った。そう、今回のゲストはアイティメディア 技術開発本部 和田良太氏。彼は自動販売機を愛し、自販機巡りのしすぎで車を1台壊してしまった男だ。

http://shuma2.net/

和田氏は、仕事では Ruby や Perl を、趣味では Ruby on Rails や Node.js、WebSocket を使ってサービスやアプリケーションを開発しているエンジニア。そして自動販売機を語らせれば彼の右に出る者はない。

彼のお気に入りの場所は、成田にある「オートパーラーシオヤ」。そこには東京近郊では珍しいハンバーガーの自販機があり、屋根もテーブルも椅子もあるので、プログラミングには最適な空間だという。

そんな和田氏に連れられて、今回のインタビュー会場へと向かう……。ここは……！！！

ズラーっと並んだ自動販売機。その自販機を見つめながら、和田氏は語る。「自販機には、その時代の先端技術が最大限に詰め込まれている。限られた空間や材料で美味しい一品をいかにして作り上げるか。可能性をぎりぎりまで追い求める。そこに自動販売機エンジニアの魂があると思う」。

さりげない自動販売機だが、それを作るにはかなりのチャレンジ精神が必要らしい。

最強の自販機とは？

太田: なぜ、自動販売機に惹かれていったのでしょうか？

和田氏：もともと古い機械が好きで、からくり時計のように自動で動くものが好きでした。ディズニーランドにもジェットコースター（アトラクション）のレールを見るのが楽しみで行っていました。そういうところから徐々に自動販売機の魅力に取り憑かれていったんです。

太田：これまでに見てきた自動販売機の中で、印象に残っているものはありますか？

和田氏：いろいろあります。技術者として唸ったのは、うどんの自販機です。あの狭い中で、ちゃんと湯切りするところがすごい。豪快だったのが、かき氷専用自販機。この自販機には「いちご」と「レモン」のボタンくらいしかなくて、ボタンを押すと固まったかき氷がドーンと出てきて、その上にシロップがかかるのですが、シロップが全部あふれてしまうんです。カップが小さいのか、氷の量が多いのか……食べにくかったですね。お味噌汁の自動販売機も忘れられません。具材がボンって落ちてきて、とろとろぬる〜いお湯が入る……。最強の自販機は、カレーの自販機です。その自販機コーナーにはインターホンが付いていて、押すとおばあちゃんがカレーを持って出てくるっていう伝説が……！

日本中どこにでもあると思っていた自動販売機。奥が深い……。そろそろ、お腹も空いてきたので「チャーシューおにぎり＆からあげ」を購入。

和田氏によると、冷凍されたおにぎりと唐揚げを、自販機の真ん中にある電子レンジで温めてから出す構造になっているそうだ。

いただきまーす！

こうして取材は無事終了。和田氏は自動販売機に囲まれながら、幸せそうに杯を乾かしていた。

自販機型の貯金箱

……今日も１人のエンジニアが、どこかで戦っている。

インタビュアー
太田智美

国立音楽大学を卒業後、慶應義塾大学大学院メディアデザイン研究科に入学。大学院在学中に、情報処理学会 第12回「Internet and Operation Technology(IOT)」研究会において「おところりん：ソーシャルネットワークにおける偶然性を用いた音楽生成」を発表し、学生奨励賞を受賞。同大学院を修了後、アイティメディア株式会社に入社。営業配属を経て、@IT統括部に異動。@ITでは記事執筆を担当するほか、年間100以上のコミュニティ・イベントに参加し、ドラ娘、マスコットキャラクター、コメンテーター、凸撃娘としてさんざん暴れまわり、愉快な生活を楽しんでいた。ところが！ 2014年4月よりねとらぼ編集部に異動という軽い大事件が発生。これから何をやらかすかは未知数である。
Facebook:http://www.facebook.com/tb.bot
Twitter:@tb_bot

シェルスクリプト大喜利 第十三回

司会：『もっと吹く』編集長・みかん

さぁ今日も、始まりますよ、このコーナー。こんにちは、シェルスクリプト、大喜利よ。きもちいね、ちょうど今頃、いい陽気。みなさんは、いかがおすごし、ですかぁぁ？（なんじゃそれ、最後の「ぁぁ」が、意味不明）

……あまりやるとしつこいのでこのくらいにしときます。えー、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。

冒頭がなぜ五、七、五だったのかと言いますと、今日のお題で恐らくシェルスクリプト界初の試みであろう「シェル俳句（川柳）」をやるからなんですよ。コマンドを五、七、五個並べて思いの丈を述べてもらうルールにしたんですが、果たしてどんな句が寄せられたのか。乞う御期待、あぁ乞う御期待、乞う御期待。

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。

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『ルース＝アーロン・ペア』を全て求めるシェルスクリプトを書いてください。

いつまで続くか一問目の整数問題。今日は、ルース＝アーロン・ペアというやつですよ。隣り合う 2 数のうち、各々の素因数の和が等しいものだそうです。例えば 8 と 9 などが該当しますので factor コマンドあたりで素因数調べて確かめてみてください。

というわけで factor コマンドを使うのが定石なんですが、果たしてどんな解答が来たでしょうか。

◎K 師範の解答

```
1 #!/bin/sh
2 seq $1 |
3 factor |
4 while read h r; do
5   echo ${h%:} $(( $(echo $r | tr ' ' +) ))
6 done |
7 awk '
8 {
9   if (b==$2) {print "(" a "," $1 ")"}
10  a=$1
```

```
11    b=$2
12 }´
```

毎度投稿ありがとうございます。わかりやすいコードを心掛けているということで、下手にヒネらないのは師範らしい解答ですね。

下手なヒネりはないけど、factor の出力から元の値の抽出と、素因数の和を求める５行目のやり方はオモシロイね。**一段授与**、これで二段だ。

◎むっつーさんの解答

```
 1 #!/bin/sh
 2
 3 IFS=$´: ¥t¥n´
 4
 5 seq 2 $1 |
 6 factor |
 7 while read Num Factors
 8 do
 9 echo "$Num $(( $(echo $Factors | tr " " "+") )) "
10 done |
11 uniq -D -f1 |
12 cut -d" " -f1 |
13 xargs -n2
```

そして、これは偶然か必然か？ 似ている解答が来ましたよ。もちろんどちらがどちらに似ているということはありませんが、こちらは IFS 変数や uniq -D オプションといったやや希少な技を使ってますね。

添付コメントによれば、uniq の cut のオプションで韻を踏んでいるらしい。なるほどね。俳句とは関係ないお題で韻を踏むとはアタクシの予想の先を行っていてて感心したよ。よし、**二段授与**。只今五段。

◎$C_6H_8O_7$（クエン）さんの解答

```
1 #!/bin/sh
2 seq 2 $1|xargs factor|ysum num=1|self 1 NF|tr -d:
  |uniq -1 -c|self 1 2|grep -v ^1|sed ´s/¥(.*¥) ¥(.
  *¥)/seq ¥2 $((¥2+¥1-1))|tr ¥¥¥¥n ,/e´|sed ´s/,$//
  ´
```

こっちは実質ワンライナーですな。Tukubai コマンドも使ってくれてますね。ありがとうございます。実はこの解答は二度修正投稿されたもので、最初は途中の sed の部分以降が AWK になってて「AWK が消せなくて悔しい」とのことでした。しかし次の投稿で「一応 AWK 消せました」ということで while; do 〜 done に差し替わっていました。しかし「while もなんかイヤだった」ということで来たのがこの解答です。

うーん、AWK や while を排除したかったその気持ちはちょっとわかる。しかし途中の sed はスゴいことになってるねぇ。置換コマンド “s///e” の中に更にワンライナーがあるね。細かいこと言うとこの解答の場合、素因数の和が等しいものが３つ連続している場合は、ペアじゃなくて３つ並ぶけどね。でも全投稿中最速だったし、お疲れ様〜。労をねぎらい**一段授与**。これで五段ね。**クエン酸補給して被曝回復してね。**

◎TSB_KZK さんの解答

```
1 #!/bin/sh
2 echo "for(x=1;x<$1;x+=1){ y=x+1; sum=0; nx=x; ny=
  y; for(i=2;i<=sqrt(y);i+=1){ if(nx%i==0){sum+=i;w
  hile(nx%i==0)nx/=i;}; if(ny%i==0){sum-=i;while(ny
  %i==0)ny/=i;}}; if(sum == 0){print ¥"x=¥";print x
  ;print ¥", y=¥";y;}}" | bc
```

これは factor コマンドに頼らない、bc コマンドによる華麗なる解答ですね。前回もこの手法で安全素数のお題を解いてくれました。

ほとんど bc のコード。引数 “$1” さえどうにかなれば、シェルスクリプトじゃなく “#!/usr/bin/bc” とかやって純粋な bc にもできちゃうんじゃなかろうか ??? これからも bc 派の回答をヨロシクね！一段授与、これで折り返しの五段だ。

◇　◇　◇

今回の整数問題の難易度はどうでしたかねぇ。さて、それじゃ次のお題に行きますよ。

<二問目>

標準入力から来る HTML テキスト内にある 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。

次はフィルタリングの問題ですね。HTML から URL を抜き出すフィルタということで、これはそこそこ実用的なお題じゃないですかねぇ。

「大喜利が実用的なお題出してどーする！」と思われてしまった、かどうかは定かではありませんが、ちょっと投稿が少なめ……。お題としてのヒネリが足りなかったかな？

◎TSB_KZK さんの解答

```
1 sed "s/¥"/¥n/g" | egrep "^(http|https)://"
```

いやぁ、これは短い！たったこれだけでいいのか！？と思って curl が吐き出す HTML を食わせたみたけど、確かに URL が出てきて動いてます。

うーん、やるなぁ……と思ったけど、このお題の文章をよく見てもらうと「タグの間に挟まれているものも」というのがあるんだな。つまり、

URL2

というのがあったら *URL1* の部分だじゃなくて *URL2* の部分も取ってきてねということなのだ。これは残念！というわけで、段位は授与できず。他に何かヒネリがあれば、授与してたかもしれないけどね。

◎伊佐坂劇物さんの解答

```
1 sed ´s/¥(http¥|https¥|ftp¥)/¥n¥1/g´ | sed ´s|[^]A
  -Za-z0-9._~:/?#@!$%&´"´"´()*+,;=[-].*$||´ | egrep
  -i ´^(http|https|ftp)://[a-z][a-z0-9.-]+[a-z]¥>(/
  [^]A-Za-z0-9._~:/?#@!$%&´"´"´()*+,;=[-]*)?´ | sor
  t | uniq
```

こちらはえぇと、タブの間の URL も取り出しているようですな。そうそう、こういうことです。添付コメントによると、RFC3986 で定義された URL として正しい文字が続く範囲を取り出してるとのことです。

なるほど、途中に出てくるたくさんの記号はそのためか。ftp も対象としてたり、sort&uniq で整理してたり丁寧なのもいいね。よし、**一段授与**。これで二段。

◎イタローさんの解答

```
1 #! /bin/sh
2 mkdir hoge
3 cd hoge
4 wget -Hrl 1 $1
5 find . -type f | sed ´s!^.!http:/!´
```

この人はいつもヒネりというかヒネクレた解答を送ってくれますね。添付コメントには「デッドリンクを除去する点が自慢です。他の細けぇことは気にすんな」って書いてありました。まあ面白くはあるので紹介しますけどねぇ。ツッコミどころ多数につき、こりゃダメですよ。

https だろうと ftp だろうと全て http と決めつけてるし、GET メソッドの CGI 変数部分が取り出せてないし、そもそも標準入力から来る HTML っていう問題文を無視してるじゃないか！あまり堅いこと言うのは趣味じゃないけど、なんかツッコミ入れられるところは全部ツッコめと言われてるような気がするのでとりあえずしといたよ。でも、wget で取り出すというアイデアは面白いね。よし、**零段授与**！三段のままだ。

◇　◇　◇

さて、お待ちかね。このお題いきましょう。

<三問目>

シェル俳句（川柳）を考えてください。基本ルールは……

- **5+7+5 で 17 個のコマンドをパイプで繋ぐ**
- **必要に応じてその句の意味の解説を付ける**

巷で人気の「アセンブラ短歌」に対抗してのお題です。どんな作品が来るか楽しみにしてたんですよ。まあ初めての試みなのでとにかくやってみましょう。

◎K 師範の解答

```
1 #!/bin/sh
2 sleep 3 | sleep 3 | sleep 3 | sleep 3 | sleep 3 |
3 sleep 4 | sleep 4 | sleep 4 | sleep 4 | sleep 4 |
  sleep 5 | sleep 5 |
4 sleep 3 | sleep 3 | sleep 3 | sleep 3 | sleep 3
5 #一分間
6 #寝ようとしたが
7 #すぐ起きた
8 #             K
```

なるほど。登場する sleep の秒数を合計すると 60 秒であるものの、実際に実行すると 5 秒で終了してしまうというオチですな。

全てが同時に開始されるという、パイプで繋がれたコマンドの性質だよね。はい、**一段授与**。これで三段。

◎たまさんの解答

```
1 echo 俺の | echo 俺の | echo 俺の | echo 俺の | e
  cho 俺の |
2 echo 俺の | echo 俺の | echo 俺の | echo 俺の | e
  cho 俺の | echo 俺の | echo 俺の |
3 echo 俺の | echo 俺の | echo 俺の | echo 俺の | e
  cho 俺の話を聞けぇぇぇぇぇぇ >/dev/null
4 # 聞いちゃいねぇ
```

あぁ、これはアレですね。唄えばいいんですね。まぁ、最後が /dev/null なのが侘しいですな。

いやぁ、/dev/null が付いているから動作的には全く意味がないんだけど、/dev/null を付けるだけでとたんに侘しくなるね……。/dev/null ごときでこんなにしんみりさせられるとは！深い、深いよ旦那。苦労して生きてきた貴方の人生が目に浮かぶようだ。この先もどうか強く生きていってもらいたい。よし、応援の気持ちを込めて二段授与。（なにやってるんだ、アタクシは……）

◎猿二号さんの解答

```
1 [ $RANDOM -ge 16384 ] && [ $RANDOM -ge 16384 ] &&
  [ $RANDOM -ge 16384 ] && [ $RANDOM -ge 16384 ] &&
  [ $RANDOM -ge 16384 ] &&
2 [ $RANDOM -le 16383 ] && [ $RANDOM -le 16383 ] &&
  [ $RANDOM -le 16383 ] && [ $RANDOM -le 16383 ] &&
  [ $RANDOM -le 16383 ] && [ $RANDOM -le 16383 ] &&
  [ $RANDOM -le 16383 ] &&
3 [ $RANDOM -ge 16384 ] && [ $RANDOM -ge 16384 ] &&
  [ $RANDOM -ge 16384 ] && [ $RANDOM -ge 16384 ] &&
  rm -ir ~/*
```

編集部注：**このコードの実行はくれぐれも自己責任で!!!** 損害が起きても責任とれませんからねっ！

えぇと、添付のコメントによりますと「気の小さ

い私のための**辞世の句**」だそうです。うーん確かに……。$RAMDOM 変数は 0~32767 の範囲をとるから各条件文の確率は 1/2 で、最後のコマンドが実行される確率は最終的に 1/65536。しかも実行されるコマンドはホームディレクトリをまっさらにするコマンドなんだけど、万が一起動しても確認プロンプトが出るという用心っぷりです。

これは rm という危険なコマンドを付けたことに対して叱るべきか、それとも 1/65536 という低い確率で rm を起動して、しかも -i オプションを付けるというあまりのいくじのなさに対して叱るべきか悩むね。いやぁ、シェル俳句って作者の性格がよく現れるね。なんかモヤモヤするけど、**一段授与**。これで六段だ。

◎t_kawai さんの解答

```
1 echo -e ´¥x1B[2¥x1B[6Cふ´ && echo -e ´¥x1B[6Cる´
  && echo -e ´¥x1B[6Cい´ && echo -e ´¥x1B[6Cけ´ &&
  echo -e ´¥x1B[6Cや´ &&
2 echo -e ´¥x1B[5A¥x1B[3Cか´ && echo -e ´¥x1B[3Cわ´
  && echo -e ´¥x1B[3Cず´ && echo -e ´¥x1B[3Cと´ &&
  echo -e ´¥x1B[3Cび´ && echo -e ´¥x1B[3Cこ´ && ech
  o -e ´¥x1B[3Cむ´ &&
3 echo -e ´¥x1B[7Aみ´ && echo -e ´ず´ && echo -e ´
  の´ && echo -e ´お´ && echo -e ´と¥n¥n´
```

コードを見ると「ふるいけや…」って書いてあるんです。でも何やらエスケープシーケンスのようなものがくっついていまして、色でも付けるのかと思って実行してみると……、あらま。せっかくなので実行結果を載せてみました。

```
$ echo -e ´¥x1B[2¥x1B[6Cふ´ && echo -e ´¥x1B[6C……
み か ふ
ず わ る
の ず い
お と け
と び や
   こ
   む
$
```

エスケープシーケンスでカーソルを移動して、縦書きにしたのか。これは、俳句を詠むというよりは俳句を詠む環境を作る作品だなぁ。詠んでる句も芭蕉だし。うーん、まあいいか。**一段授与**。これで二段だ。

◇　◇　◇

シェル俳句は、作者の心情や性格が顕れて面白いね。でも、全体的に同じコマンドの繰り返しが多かったなぁ。もっとたくさんのコマンドを駆使した句が見たいね。またそのうち開催しようかな。

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「半素数」を列挙するシェルスクリプトを書いてください。

二、0~9 のいずれかの数 4 つが引数で与えられる。これらの数字に任意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。

イ、4 つの数の順番は、入れ替えアリとするのでもナシとするのでも、どちらの方針でも OK。

ロ、カッコを付けるのをアリとするのでもナシとするのでも、どちらの方針でも OK。

要は面白いコードになるならどっちも OK よ。

三、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。

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**８月１１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 6 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 978-4-9048-0709-5
C3455 ¥500

9784904807095

USP 研究所
定価（本体 500 円+税）

1923455005002