

# WORD

60秒以内に  
WORDを用意  
出来なかったら  
WORD無料券を  
プレゼントします  
号

港町に行こう!

mbed系男子になろう

Google Apps Script入門

GRな日々。XIV

tmuxを便利に使う

New comer Haswell

車好きの18きつぶ

WORD民のためのLaTeX2e入門

とよさと!

書籍紹介

母校が志望校になった日 リターンズ

山口喜教教授 退職直前インタビュー

突撃!編集部員の所持品検査

オスっちメスっち育成日記

25

2013.2

## 目次

|   |    |
|---|----|
| 港町にいこう！                                 | 3  |
| mbed 系男子になろう                            | 7  |
| Google Apps Script 入門                   | 17 |
| GR な日々。XIV                              | 21 |
| tmux を便利に使う                             | 25 |
| New comer Haswell                       | 31 |
| 車好きの 18 きっぷ                             | 43 |
| WORD 民のための $\text{\LaTeX} 2\epsilon$ 入門 | 46 |
| とよさと！                                   | 57 |
| 書籍紹介                                    | 62 |
| 母校が志望校になった日 リターンズ                       | 64 |
| 山口喜教教授 退職直前インタビュー                       | 71 |
| 突撃！編集部員の所持品検査                           | 74 |
| オスっちメスっち育成日記                            | 83 |
| WORD 編集部への誘い                            | 94 |

# 港町に行こう！

文 編集部 ジオン

## 1.はじめに

鳥取の鼓動シリーズ第3回目となる今回。前回はあまりにも衝撃的なネタだったため地元の良いところを全く紹介できなかった。その反省を活かし、今回は冬休みに私が訪れた鳥取県の観光地の紹介をする。

## 2.名港 境港を行く

### 2.1 日本有数の漁港境港

今回訪れた土地は鳥取県境港市。第1回目の水木しげるロード紹介の際にも取り上げた町である。この町は妖怪の町として有名になったのとは別に、古くから日本有数の水揚げ量を誇る良港として的一面を持っている。実績を調べると驚くことにマグロとズワイガニ系の水揚げ量が日本一である。さらに平成23年の主要漁港別の総水揚げ量ランキングでは銚子、焼津に続く3位にまでなっている。

### 2.2 境港水産物直売センター

境港市のホームページ<sup>\*1</sup>に「さかなと鬼太郎のまち」と書いてあるように漁業も同市の立派な観光資源である。境港水産物直売センターもその中の一つで、だだっ広い駐車場と魚のイラストが描かれたシンプルな看板のそばによくありそうな白い倉庫が建っているだけという殺風景な外貌だが、観光客に媚びていないと考えればなかなかに好感が持てる。

中に入ると魚の生臭い香りと共に商店の人々の活気のある声が耳に入ってくる。私が訪れたのは年末だったということもあり、センター内は多くの人で賑わっていた。このセンターを何度か訪れたことがあるが、ここまで人が多いというのは初めての体験だった。恐らく観光客に、年末年始の食卓を飾る食材を探しに来た地元民が加わったのが原因だろう。この時期の名物は何と言っても松葉ガニだ。どの商店をみても一番目立つ場所を陣取っているのは松葉ガニである。松葉ガニとはこの地域で獲れるズワイガニのブランド名であり、一般のズワイガニと比べ殻は固く、ずつしりと重く、鋏は太いといわれている。当然高価であり地元の人間も日頃から口にできるようなものではないが、年末のこの時期



アクセスは境港駅からバスで39分。  
米子空港からバスで11分。



通路を挟んだ両サイドに商店が並ぶ。通路は全長50メートルほど。

\*1 <http://www.city.sakaiminato.lg.jp/>

ばかりは地元の人間も地元で獲れた海の幸に手が伸びるようだ。松葉ガニとひとえにいっても生なのか加工済みなのか、完全な形なのか欠損しているのかなどにより値段は様々である。自分の財布に合わせてカニを購入することができるのが良いところ。

松葉ガニの他にも豊富な海の幸を取り扱っており、生の物から加工した物までなんでもそろっている。普段魚と縁の無い人は、市場いっぱいに並べられたそれらをただ見るだけでも十分に楽しむことができるだろう。魚を眺めた後、食したことの無い珍しい魚を購入してみるのもまた一興かもしれない。商店の人に質問をすれば魚をどのように調理したらいいのか教えてもらえる。このように販売者と購入者の距離が近い事も直売センターの魅力だ。



物によっては通常のズワイガニの2、3倍の値がついている。

## 2.3 最後に

まんが王国を名乗り町おこしを進めている鳥取県。しかし、その地域本来の魅力も忘れてはならない。鳥取県に訪れる事があれば是非、その豊かな海の幸を堪能していただきたい。



小さい魚は一皿単位でしか購入できない分お得。

# そいやっさ!!

今回は豪華二本立て！また鳥取×サブカル関係でおもしろいネタが発掘された。



### 3.『きみわた』プロジェクト

#### 3.1 はじめに ぱーと2

前回『まんが王国とつとり』に合わせて制作された『きみうたつむぎ 常』を紹介した。しかし、『きみうた』以外にもサブカルチャーで町おこしをしようとする活動は調べれば調べるほどでてくる。今回も前回に続き、それらの活動の一部である『きみわた~君と私と想い街~』を紹介する。

#### 3.2 『きみわた』って何？

『きみわた』プロジェクトは鳥取県中部の倉吉市で始まった企画で、女子高生二人の物語を通じ鳥取県の豊かな自然や観光名所などをアピールしていくものである。ただの萌えによる町おこしではなく、二人の女の子の視点で現代に生きる人々の心の葛藤などを描いたりとメッセージ性も強い作品となっている。物語はイラスト、漫画、小説と複数の媒体を通して公式サイト<sup>\*2</sup> や pixiv<sup>\*3</sup> などで公開されている。

#### 3.3 あらすじ

宇佐美祐花（うさみゆか）と稻田なつみ（いなたなつみ）は中学1年の時からの親友である。  
中学3年生の頃、なつみの進路に関する想いのすれ違いと祐花の受験の失敗により、一度はわだかまりを抱えてしまった二人。 だが祐花の家出をきっかけにお互いの素直な気持ちを打ち明ける事が出来た二人はもう一度親友として手を取り合う事が出来た。  
4月から倉吉市の別々の高校で、それぞれの新しい生活が始まる。かつての幼馴染との再会。なつみの兄へのほのかな想い。小学生の無邪気な女の子とそれに振り回される男の子。 新たな出会いは二人に何をもたらすのか。

**君と私の物語はここから始まる。**

——公式サイトから引用

#### 3.4 登場人物紹介

宇佐美祐花（うさみゆか）

倉吉の私立高校に通う女子高生。明るい性格で周囲の人を引っ張っていくリーダータイプだが、その反面寂しがり屋。将来したいことや夢がないことに焦りを感じている。勉強はできないが生きるうえでの知識は豊富。なつみとは中学生時代からの親友。



\*2 <http://www.kimiwata.com/>

\*3 <http://www.pixiv.net/member.php?id=4938038>

稻田なつみ（いなたなつみ）

倉吉のある農業高校に通う女子高生。小学生のころに虐められた経験から人の輪に入ることに消極的になり、人付き合いも苦手で引っ込み思案。自分のやりたいことに対しては努力を惜しまないタイプで、将来の夢である保育士になる為に日々努力をしている。勉強はできるが、天然で普段はその片鱗をみることはない。祐花とは中学生のころに出会い親友となった。



### 3.5 県公認の活動として

前回の『きみうた』はどうしてこうなったの嵐だったが『きみわた』はそれに比べてしっかりとした活動をしている。両者で最も異なっているのは県から補助金がでているか否かということだろう。鳥取県は鳥取力創造運動支援補助金<sup>\*4</sup>という制度を導入しており、地域をよくするための活動を行う団体に補助金を出している。『きみわた』プロジェクトはこの団体の一つとして補助金を受け取っている。そのため中途半端なことはできないのか『きみわた』の公式サイトはしっかり整備され、本編の作品としてのクオリティーも安心して見れるものとなっている。

グッズ販売もしっかりとしており、まんが博でのブース出展や公式サイトから通信販売も行っている。通信販売では『きみうた』のキャラクターのカーテンや抱き枕カバーなどが1万5000円前後で売っており、誰が買うのだろうかと見ていて面白い。現在公式サイトでは4月に発売されるドラマCDのPV<sup>\*5</sup>が公開されている。声優はやはり高校生が担当しているが、PVを見る限り悪くない。このクオリティを維持したまま活動を続けてくれることを心から願う。

### 3.6 最後に ぱーと2

鳥取県の町おこしの定番としてやはり知名度は低い『きみわた』だが、前回の某同人ゲームと比べると本腰を入れて地元のアピールと町おこしを行おうとしている姿勢が見て取れる。私としてもこの記事を読んで興味を持った方には是非公式サイトを訪れていただきたいと思う。この企画の特徴として、公式サイトで公開されているイラストは最低限の利用規約さえ守ればだれでもどこでも使用できるフリー素材となっている。二次創作も自由である。気に入った方はイラストを様々な場所で紹介したりして『きみわた~君と私と想い街~』を宣伝して頂きたい。

---

\*4 <http://www.pref.tottori.lg.jp/127928.htm>

\*5 [http://www.youtube.com/watch?feature=player\\_embedded&v=ugVmPxqWN9I](http://www.youtube.com/watch?feature=player_embedded&v=ugVmPxqWN9I)

# mbed系男子になろう！

～ 出力関数の時間を計ろう編～

文 編集部 ,無季

## 1 はじめに

### 1.1 事の始まり

,無季は激怒した<sup>1</sup>。必ず、かの邪知暴虐な制御の遅れを取り除かねばならぬと決意した。,無季には制御はわからぬ。,無季は野良の技術屋である。プログラムを書き、mbedマイコンと遊んで暮らしてきた。けれども機械制御に対しては、人一倍敏感であった。ものづくりをしているうちに,無季は制御の様子を怪しく思った。異音がしている。しばらく歩いて老爺に逢い、語勢を強くして質問した。老爺は答えなかった。,無季は両手で老爺のからだをゆすぶって質問を重ねた。老爺は、あたりをはばかる低声で、わずか答えた。

「printf関数は制御周期の遅れを招きます。」

「なぜ遅れるのだ。」

「処理時間が数百 ms かかっている、というのですが、誰もそんな、遅延を持って居りませぬ。」

「驚いた。printf関数は乱心か。」

「いいえ、乱心ではございませぬ。処理時間を、保障することができぬ、というのです。」

聞いて、,無季は激怒した<sup>2</sup>。「呆れた関数だ。使ってはおられぬ。」

.....

というわけで、今回は mbed マイコンの printf 関数にかかる時間を計測しました。モータ等の制御ループ内で制御ログを保存することができるのか、また処理時間はどれほどかかるのかを確認したいと思います。実用的な目安は 1ms 以内だと思います。

### 1.2 mbedマイコン概要

mbed マイコンはラピッドプロトタイピング向けの ARM マイコンです。

開発環境が Web 上にあることから煩わしい環境構築が不要で、非常に簡単に始められます。ライブラリが豊富にあるのも利点の一つです。コンパイルが通ると実行ファイルがダウンロードされます。また、mbed マイコンを USB ケーブルでパソコンに接続すると、USB フラッシュメモリのよう

に認識されるので、ダウンロードされた実行ファイルを GUI 上でコピー &

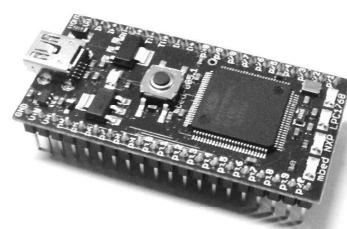


図 1: mbed マイコン

\*1 ※してません。

\*2 ※してません。大事なことなので 2 回言いました

ペーストするだけで書き込みが行えます。詳しい使い方や製作例は過去の WORD<sup>\*3</sup> で紹介しました。

## 2. 計測方法

### 2.1 計測方法概要

mbed マイコン LPC1768において「Hello World!」という文字列、または int 型の変数 1つを 10進数で 1000 回出力する処理時間を計測します。mbed マイコンの Timer を用いて、次の 4つの出力方法における時間をマイクロ秒 (us) 単位でそれぞれ計測しました。

- ・PC にシリアル通信で出力（通常の printf 関数）
- ・mbed マイコン評価用ベース基板☆ board Orange<sup>\*4</sup> 上の液晶ディスプレイ（LCD）に出力
- ・mbed マイコンのローカルディレクトリへのファイル出力
- ・☆ board Orange 上の microSD カードへのファイル出力

### 2.2 PCに出力する

mbed マイコンでは、printf 関数は mbed マイコン上の USB コネクタ（書き込みに使うところ）から PC にシリアル通信で出力します。プログラムをリスト 1 に示します。

```
1: #include "mbed.h"
2:
3: int main(){
4:     printf("Hello mbed world! $\n");
5: }
```

リスト 1: 標準出力のプログラム

出力を確認するためには、screen コマンドを用いるか（Mac、linux）、それ用のソフトウェアを用います。

- ・Mac の場合

mbed を PC に接続した状態から、USB デバイスの接続先を確認します。

```
$ls /dev/tty.usbmodem*
```

すると、/dev/tty.usbmodem1412などの文字列が返ってきますので、screen コマンドで表示を行います。

```
$screen /dev/tty.usbmodem1412
```

- ・Windows の場合

1. 「the mbed Windows serial port driver」をダウンロード、およびインストールします。
2. 次に Tera Term を起動し、「シリアル」にチェックし、ポート「mbed Serial Port」（Tera Term ver. 4.72）を選択します。

---

\*3 過去の記事は WORD Press (<http://www.word-ac.net/>) で読めるよ！

\*4 「すたーぼーどおれんじ」と読む。Ethernet や USB、microSD カード、電源のコネクタや LCD が取り付けられた基板

## 2.3 LCDを使う

☆ board Orange 上の LCD に表示するためには、TextLCD というライブラリをインポートします。インポート手順は次の通りです。なお、詳しいインポート手順は「WORD vol.22 学類誌と称する事実上の薄い本号<sup>\*5</sup>」中の「使ってみよう～中級編：他人のソースコードを利用してみる～」に紹介しました。

1. 開発環境のページ左上、「Import」をクリックする
2. 検索画面のページ下部のフォームより「TextLCD」を検索する
3. 検索されたものの中から、適切なものを選択し、「Import!」をクリック

TextLCD に関しては以下のサンプルコードが与えられています。☆ board Orange で使うためには 6 行目をリスト 3 のように変更します。

```
1: // Hello World! for the TextLCD
2:
3: #include "mbed.h"
4: #include "TextLCD.h"
5:
6: TextLCD lcd(p15, p16, p17, p18, p19, p20); // rs, e, d4-d7
7:
8: int main() {
9:     lcd.printf("Hello World!\n");
10: }
```

リスト 2: TextLCD のサンプルコード

```
6: TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
```

リスト 3: ☆ borad Orange 用に変更後

## 2.4 ローカルファイルを使う

mbed マイコンのローカルディレクトリにファイルの作成や入出力を行うには LocalFileSystem を使います。

LocalFileSystem では、`fprintf` 関数や `fscanf` 関数などの見慣れた (?) ファイル操作の関数が定義されています。これに関しても、(前略) 薄い本号中の「使ってみよう～初級編：ファイル入出力～」で紹介しましたので、ご参照下さい。

```
1: #include "mbed.h"
2:
3: LocalFileSystem local("local"); // Create the local filesystem under the name "local"
4:
5: int main() {
6:     FILE *fp = fopen("/local/out.txt", "w");// Open "out.txt" on the local file system for writing
7:     fprintf(fp, "Hello World!");
8:     fclose(fp);
9: }
```

リスト 4: LocalFileSystem のサンプルコード

<sup>\*5</sup> WORD Press (<http://www.word-ac.net/wp-content/uploads/2012/09/22.pdf>)

## 2.5 microSDカードを使う

microSD カードを使うには SDFFileSystem というライブラリを使います。そして、SDFFileSystem を使うにはさらに FATFileSystem というライブラリをインポートする必要があります。よって、SDFFileSystem をインポートする際には、FATFileSystem をインポートしているものを使いましょう。例えば、Simon Ford 氏の SDFFileSystem<sup>\*6</sup> がそれです。

```
1: #include "mbed.h"
2: #include "SDFFileSystem.h"
3:
4: SDFFileSystem sd(p5, p6, p7, p8, "sd"); // the pinout on the mbed Cool Components workshop board
5:
6: int main() {
7:     printf("Hello World!\n");
8:
9:     mkdir("/sd/mydir", 0777);
10:
11:    FILE *fp = fopen("/sd/mydir/sdtest.txt", "w");
12:    if(fp == NULL) {
13:        error("Could not open file for write\n");
14:    }
15:    fprintf(fp, "Hello fun SD Card World!");
16:    fclose(fp);
17:
18:    printf("Goodbye World!\n");
19: }
```

リスト 5: SDFFileSystem のサンプルコード

## 2.6 Timerを使う

Timer は短い時間<sup>\*7</sup> を計測するために用いられるものです。mbed マイコンの場合、 $2^{31}-1$  us、すなわち 35 分程度計測できます。Timer のサンプルプログラムはリスト 6 のように与えられています。これは標準出力に"Hello world!"を出力する時間を計測し、出力するプログラムです。これをベースにプログラムを書くことにしました。

```
1: #include "mbed.h"
2:
3: Timer t;
4:
5: int main() {
6:     t.start();
7:     printf("Hello World!\n");
8:     t.stop();
9:     printf("The time taken was %f seconds\n", t.read());
10: }
```

リスト 6: Timer のサンプルコード

---

\*6 mbed の Simon Ford 氏のユーザーページ SDFFilesystem (<http://mbed.org/users/simon/code/SDFFileSystem/>)

\*7 時や日、年単位のような長い時間には time がある。time はリアルタイムクロック (RTC) を利用して時間（時、分、秒）や日付（年、月、日）を読み出すことが可能である

## 2.7 作成したプログラム

以上のことから、作成したプログラムはリスト 7 のようになりました。16、18 行目は、ローカルファイルを扱う場合と SD カード上のファイルを扱う場合で、どちらかをコメントアウトしてコンパイルしました。加えて、31 から 33 行目も同様に出力方法に合わせて、コメントアウトして使いました。

```

1: #include "mbed.h"
2: #include "TextLCD.h"
3: #include "SDFileSystem.h"
4:
5: #define LOOP 1000
6:
7: Timer t;
8: TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
9: LocalFileSystem local("local"); // Create the local filesystem under the name "local"
10: SDFileSystem sd(p5, p6, p7, p8, "sd"); // the pinout on the mbed Cool Components workshop board
11:
12:
13: int main() {
14:
15: /* File open */
16:     FILE *fpout = fopen("/local/out.txt", "w");
17:         // Open "out.txt" on the local file system for writing
18:     //FILE *fpout = fopen("/sd/out.txt", "w");
19:         // Open "out.txt" on the SDcard file system for writing
20:     if(NULL == fpout){
21:         lcd.printf("File open error.\n");
22:         exit(1);
23:     }
24:
25:
26: /* Timer start */
27:     t.start();
28:
29: /* Output */
30:     for(int i=0; i<LOOP; i++){
31:         //printf("Hello World!\n"); // Output to PC
32:         lcd.printf("Hello World!\n"); // Output to LCD
33:         //fprintf(fpout,"Hello World!\n"); // Output to File
34:     }
35:
36: /* Timer stop */
37:     t.stop();
38:
39:     fclose(fpout);
40:
41: /* Output the result */
42:     lcd.printf("Time: %d [us]\n", t.read_us());
43:
44:
45:     return 0;
46: }
```

リスト 7: 使用したプログラム

### 3. 結果と考察

#### 3.1 結果

プログラムを 5 回繰り返し実行し、その平均時間を求めました。それぞれ結果は表 1、表 2 のようになりました。また平均時間を図 2 に示します。

表 1: 文字列出力の計測結果

| hello world! 1000回 | try1[us] | try2[us] | try3[us] | try4[us] | try5[us] | ave[us]    | ave[ms]    |
|--------------------|----------|----------|----------|----------|----------|------------|------------|
| printf, pc         | 13539761 | 13539768 | 13539712 | 13539708 | 13539759 | 13539741.6 | 13539.7416 |
| printf, lcd        | 4155003  | 4154996  | 4154990  | 4155001  | 4155000  | 4154998.0  | 4154.998   |
| fprintf, local     | 377276   | 372051   | 402030   | 376388   | 375235   | 380596.0   | 380.5960   |
| fprintf, SDCard    | 226657   | 262260   | 226666   | 228501   | 226874   | 234191.6   | 234.1916   |

表 2: 変数出力の計測結果

| Loop var 1000回  | try1[us] | try2[us] | try3[us] | try4[us] | try5[us] | ave[us]    | ave[ms]    |
|-----------------|----------|----------|----------|----------|----------|------------|------------|
| printf, pc      | 4050140  | 4050125  | 4050165  | 4050165  | 40500077 | 11340134.4 | 11340.1344 |
| printf, lcd     | 1012243  | 1012246  | 1012255  | 1012244  | 1012253  | 1012248.2  | 1012.2482  |
| fprintf, local  | 112570   | 110432   | 106027   | 106085   | 105994   | 108221.6   | 108.2216   |
| fprintf, SDCard | 65615    | 64981    | 109469   | 64736    | 65052    | 73970.6    | 73.9706    |

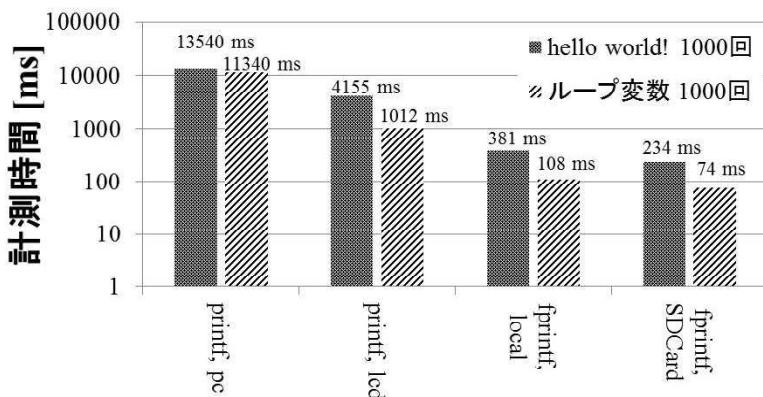


図 2: 1000 回出力時の計測時間

やはり、PC にシリアル通信で出力するのは遅いのか……。1 回当たり約 14ms もかかっていらっしゃる。一方、ファイル出力勢が健闘していますね。1 回当たり 1ms を下回っています。これは制御ループ内で使えると思われます。ただいくつか遅いデータ（表 1 fprintf, local の try3、fprintf, SDCard の try2、表 2 fprintf, SDCard try3 など）がありますね。

### 3.2 ,無季に電流走る……！

「printf って、バッファリングしてるんじゃなかつたっけ……？ひょっとすると、バッファに溜めてからまとめて出力している場合、制御周期を乱す事になるのでは……？」

そんなわけで各ループにおける処理の時間を計測することになったのです。



図 3: 感電したわけではない

## 4. バッファリングを検証

### 4.1 プログラムの変更

各ループの処理時間を保存するための配列 int time[LOOP]を宣言します。そして、ループ終了後にそれをファイル出力するようにプログラムを書き換えました（リスト 8）。これを 3 回実行し、それぞれの処理時間の変化を確認しました。

```

1: #include "mbed.h"
2: #include "TextLCD.h"
3: #include "SDFileSystem.h"
4:
5: #define LOOP 1000
6:
7: Timer t;
8: TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
9: LocalFileSystem local("local"); // Create the local filesystem under the name "local"
10: SDFileSystem sd(p5, p6, p7, p8, "sd"); // the pinout on the mbed Cool Components workshop board
11:
12:
13: int main() {
14:     int time[LOOP];
15:     FILE *fpout, *fplog;
16:
17:
18: /* File open */
19:     fpout = fopen("/local/out.txt", "w"); // Open "out.txt" on the local file system for writing
20:     //fpout = fopen("/sd/out.txt", "w"); // Open "out.txt" on the SDcard file system for writing
21:     if(NULL == fpout){
22:         lcd.printf("File open error.\n");
23:         exit(1);
24:     }
25:
26:
27: /* Timer start */
28:     t.start();
29:
30: /* Output */
31:     for(int i=0; i<LOOP; i++){
32:         //printf("Hello World!\n"); // Output to PC
33:         lcd.printf("Hello World!\n"); // Output to LCD
34:         //fprintf(fpout,"Hello World!\n"); // Output to File
35:         time[i] = t.read_us(); // Record time
36:     }
37:
```

```
38: /* Timer stop */
39:     t.stop();
40:
41:
42:     fclose(fpout);
43:
44: /* Output the result */
45:
46:     fplog = fopen("/local/TimeLog.txt", "w");
47:         // Open "out.txt" on the SDcard file system for writing
48:     if(NULL == fplog){
49:         lcd.printf("TimeLog File open error.\n");
50:         exit(1);
51:     }
52:
53:     for(int i=0; i<LOOP; i++){
54:         fprintf(fplog, "Time %d : %d [us]\n", i,time[i]);
55:     }
56:     fclose(fplog);
57:
58:     return 0;
59: }
```

リスト 8: バッファリングを検証するために変更したプログラム

## 4.2 PCへの出力時における計測結果

PCへの出力では、図4のような時間経過になりました。PCへの出力では、特にバッファリングは見られませんでした(だって、シリアル通信ってことになってるしね。通信をバッファリングして遅らせたらエライことだ！)。ただし、1回の出力では約13.5 ms かかっています。

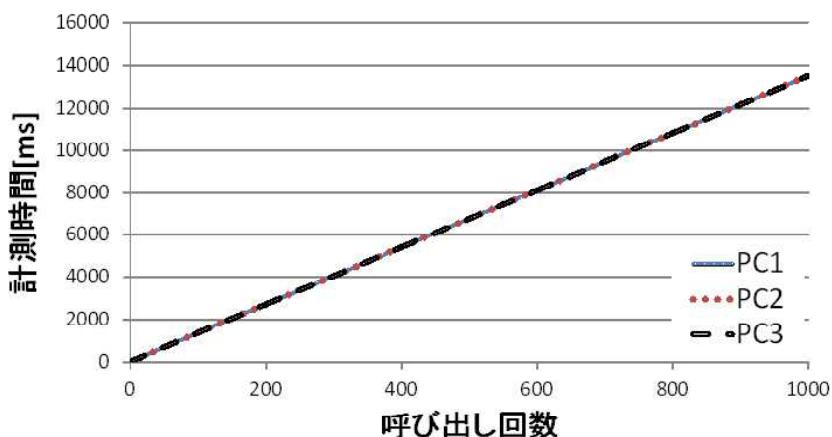


図4: PCへの出力時における時間経過

### 4.3 LCDへの出力時における計測結果

LCDへの出力も図5のようにバッファリングは見られず、コンスタントに約4.15 ms かけていました。

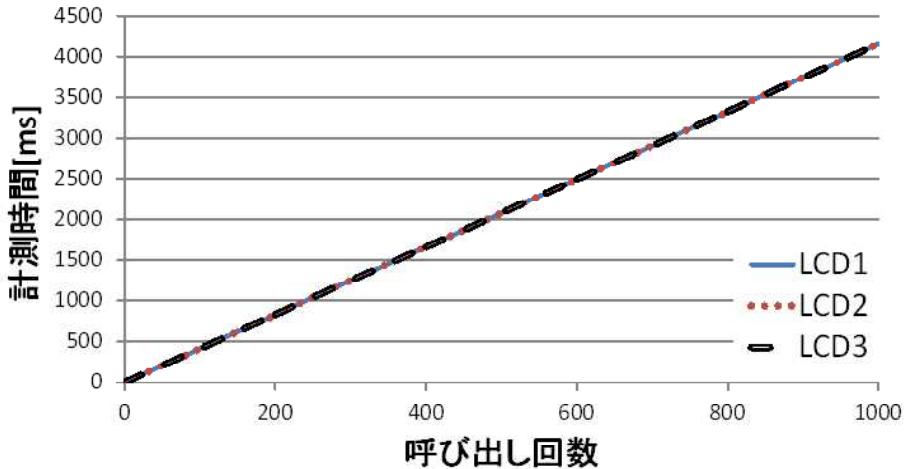


図5: LCDへの出力時における時間経過

### 4.4 ローカルファイルへの出力時における計測結果

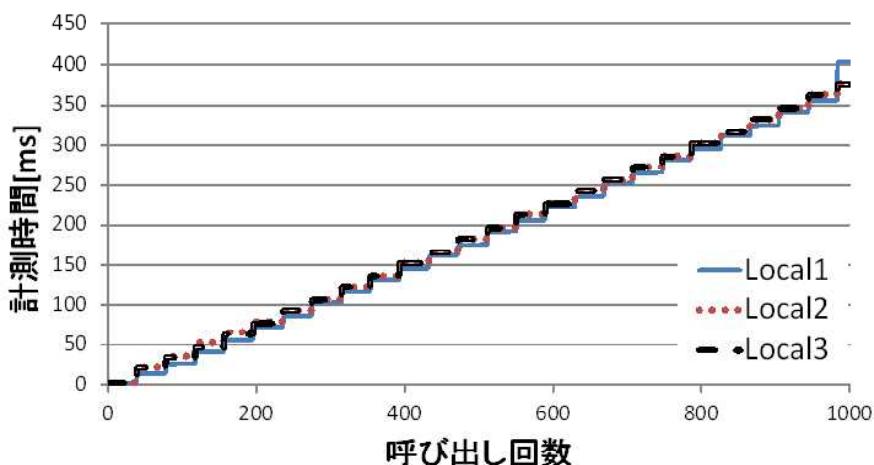


図6: ローカルファイルへの出力時における時間経過

これは、バッファリング……！？

40回に1回、13msほどの処理時間が定期的に現れています……。

#### 4.5 SDカードへの出力時における計測結果

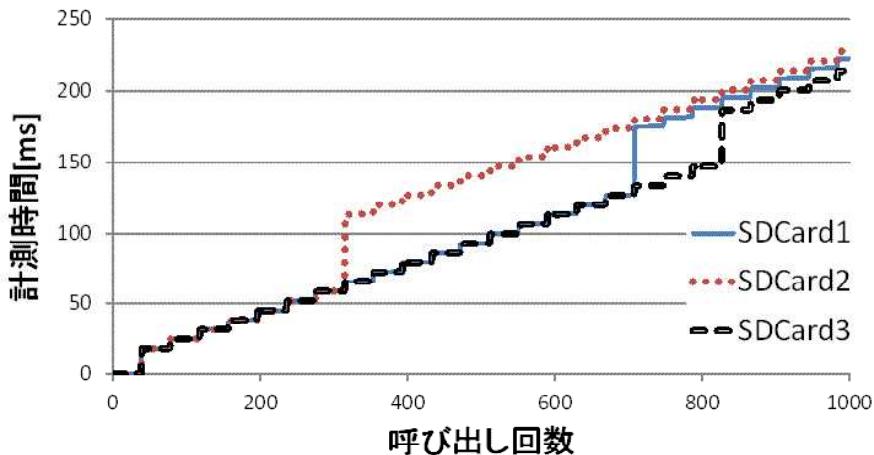


図 7: SD カードへの出力時における時間経過

うつ……！  
こ、これは……！？

60 回に 1 度 6ms のバッファリングがあるし、そんなことより、なんだあの処理時間は……？ とてもでつかい処理時間がいらっしゃるではありませんか！ 48ms ですよ、48ms !  
こんな感じや制御にならないんだよ（激怒）。

### 5. おわりに

2 つの実験結果より、ファイル出力は平均 1ms を下回っているものの、処理時間にムラがあることが分かりました。それらを吸収できるほどのゆっくりとした制御周期の場合でないと、制御中に出力するってことは避けた方が良いと思いました（小学生並みの感想）。

次回は、モータ制御に関する記事も書きたいなって思っています。

# Google Apps Script入門

文 編集部 えりつく

## 1 Google Apps Scriptとは

公式サイトによると、Google Apps Script(以下 GAS)とは"A JavaScript cloud scripting language that provides easy ways to automate tasks across Google products and third party services"だそうです。Google Apps や Google App Engine などと混同している方もいると思うのでそれぞれの定義を挙げると、

- ・Google Apps(GA) : Gmail や Google カレンダーを独自・ドメインで使えるようにする有料サービス。
- ・Google App Engine(GAE) : Google 保有のクラウドを用いた Web アプリケーションのプラットフォーム。
- ・Google Apps Script(GAS) : Google のサービスなどを簡単に操作するスクリプト言語。

といった感じです。<sup>\*</sup>

Gmail・Google Drive・Google カレンダーなどの主要な Google の Web サービスを非常に簡単に操作できる API が用意されているため、JavaScript が書ける人ならすぐに手に馴染むと思います。Web 上の GUI アプリケーション、Google スプレッドシートの独自定義関数、JDBC によるデータベースを用いたアプリケーションなどが簡単に書けてしまうため、仕事の自動化にピッタリの言語ですよ！

筆者は学園祭実行委員会のシステム開発を担当しており、雙峰祭グランプリ 2012のために、携帯メールによる投票システムを開発していました。「引継ぎを簡単にするために、Postfixなどを使わないで実現したい」と考え、GAS と(罪の意識を強く感じながら)PHP を使ってこのシステムを開発することに決めました。

今回はそんな GAS について、例を挙げて紹介したいと思います。写経ゲーともいいます。

## 2 メール応答システムを作ろう

今回は「自分の Gmail アカウントに送られてきた特定のメールを、校正して返信するメール応答システム」を GAS で作ってみましょう。校正には Yahoo! が公開している校正支援 API を用いてみます。

GAS は、Web 上に IDE があります。まずは Google にログインし、Google Drive にアクセスします。そこで「作成」の「もっと見る」から「スクリプト」を選択します。そして「空のプロジェクト」でプロジェクトを作成しましょう。「コード.gs」が選択されていますので、ここにコードをすらすらと書いていきます。プロジェクト名とかコード名とかは適当に変えちゃってください。

\*1 このように、GAS と GA はインド人とインディアンくらい関連がありません。紛らわしいですね。

まずは Yahoo! の API を叩くところを書きます。API を叩き<sup>\*2</sup>、XML を解析しています。この API を叩くには Yahoo! のデベロッパー登録が必要なのでちょいと面倒ですが、便利なのでこの際登録してみてください。

```
function proofread_text(str) {
  var text = "校正結果は以下のとおりです。¥n";
  var proofreaded = proofread(str);
  if(proofreaded.length == 0) return text + "指摘点がありません。";
  for(var i = 0; i < proofreaded.length; i++) {
    text = text + proofreaded[i].surface + "->";
    text = text + proofreaded[i].word;
    text = text + "(" + proofreaded[i].info + ")¥n";
  }
  return text;
}

function proofread(sentence_raw) {
  // Yahoo! テキスト処理 API を叩く
  var uri = "http://jlp.yahooapis.jp/KouseiService/V1/kousei";
  var appid = "自身で取得したアプリケーション ID"*3;
  var sentence = encodeURIComponent(sentence_raw);
  var option = {
    method : 'post',
    payload : "appid=" + appid + "&sentence=" + sentence
  };
  var doc = Xml.parse(UrlFetchApp.fetch(uri, option).getContentText(), true);
  var response = doc.getElement().getElements();
  var result = [];
  // パースした結果をオブジェクトにして返す
  for(var i = 0; i < response.length; i++) {
    result.push({
      surface : response[i].getElement('Surface').getText(),
      word : response[i].getElement('ShitekiWord').getText(),
      info : response[i].getElement('ShitekiInfo').getText()
    });
  }
  return result;
}
```

---

\*2 <http://developer.yahoo.co.jp/webapi/jlp/kousei/v1/kousei.html> を参考にしてください。

\*3 Yahoo! デベロッパー登録をするともらえます。

これで、文章を受け取れば校正結果を生成できるようになりました。あとは Gmail に届いたメールを取得し、返信すべきメール(proofread ラベルがついているもの)だけに返信するところを追記すれば終わりです。

```
function handle_mails() {
  var label = GmailApp.getUserLabelByName("proofread");
  var threads = label.getThreads();
  for(var i = 0; i < threads.length; i++) {
    var messages = threads[i].getMessages();
    for(var j = 0; j < messages.length; j++) {
      // 校正結果を返信する
      messages[j].reply(proofread_text(messages[j].getBody()));
      messages[j].markRead();
    }
    threads[i].removeLabel(label);
    threads[i].moveToArchive();
    threads[i].markUnimportant();
  }
}
```

それぞれの本文を校正し、校正結果を返信しています。校正後には既読にしてアーカイブしています。

さて、ここまでできたところでこの `handle_mails` 関数を 1 分ごとに実行させるように<sup>\*4</sup> 設定します。IDE の「リソース」タブから「すべてのトリガー」を選択し、`handle_mails` を 1 分ごとに実行させるように登録してください。あとは Gmail の設定で「件名が『proofread』のとき、ラベル proofread をつける」というフィルタを作成すれば完成です。

以下のメールを自分の Gmail アドレスに送信すると、

Subject : proofread

Body :

校正支援 Web API は、24 時間以内で 1 つのアプリケーション ID につき 50000 件のリクエストが上限となっています。また、1 リクエストの最大サイズを 100KB に制限しています。詳しくは「利用制限」をご参照ください。

<sup>\*4</sup> 「実行」メニューから関数名を選択しても実行できます。残念ながら 2012 年現在、メールを受信したイベントを受け取ることはできません。

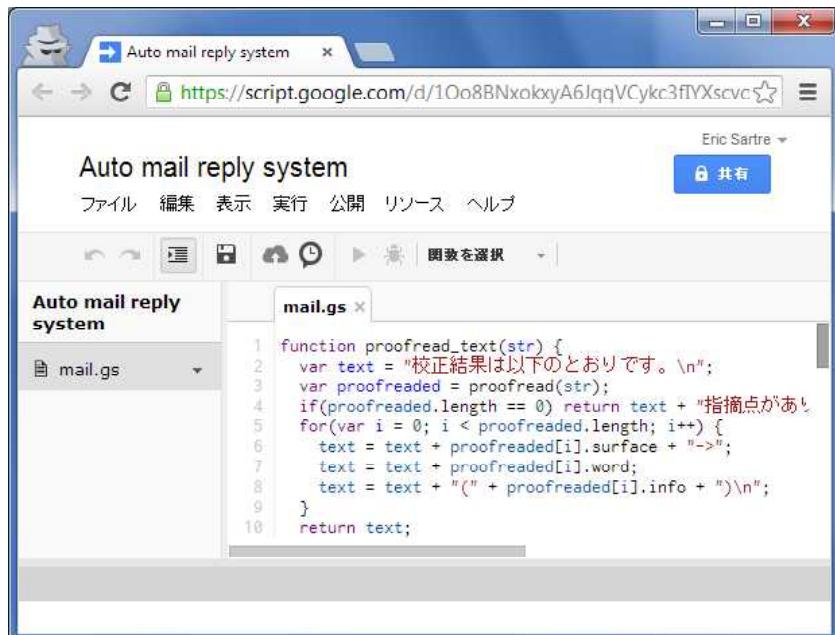
こういうのが返信されてきます。

```
Subject : Re:proofread  
Body :  
校正結果は以下のとおりです。  
Web->ウェブ(用字)  
KB->キロバイト (リテラシレベルに応じて KB も OK) (用字)
```

わあ便利。

## このように

GAS を使うととっても簡単に素敵なシステムを作ることができます。みなさんも是非、GAS を使って作業をサポートしましょう（はあと）



The screenshot shows the Google Apps Script interface. The title bar says "Auto mail reply system". The left sidebar shows a project tree with "Auto mail reply system" expanded and "mail.gs" selected. The main editor area contains the following code:

```
function proofread_text(str) {  
  var text = "校正結果は以下のとおりです。\\n";  
  var proofreaded = proofread(str);  
  if(proofreaded.length == 0) return text + "指摘点がありません";  
  for(var i = 0; i < proofreaded.length; i++) {  
    text = text + proofreaded[i].surface + "->";  
    text = text + proofreaded[i].word;  
    text = text + "(" + proofreaded[i].info + ")\\n";  
  }  
  return text;
```

## GRな日々。XIV

文 編集部 葡萄酒

つくば市から車で1時間半、茨城県と栃木県を結ぶ「<sup>もおか</sup>真岡鐵道」という路線がある。そこでは、昨今ではもう珍しい蒸気機関車が、今もなお人々を乗せて走っているという。今回は、その蒸気機関車「SL もおか」を紹介する。

真岡鐵道・真岡線は、茨城県は下館<sup>しもだて</sup>から栃木県の茂木<sup>もてぎ</sup>に至る、40km ほどの鉄道路線である。普段はディーゼル気動車によって運行されているのだが、土日は1日1往復「SL もおか」号が運行される。この「SL もおか」はC12 66とC11 325の二両<sup>\*1</sup>を用いており、前者は新潟県の水原町から、後者は福島県川俣町から譲り受けたもので、1994年から運行が始まった。これらはそれぞれ1934年と1946年に製造された車輌で、どちらも静態保存されていたものを復元して運行している。

さて、今回は茨城県側の下館駅から「SL もおか」に乗ることにしよう。通常の乗車券<sup>\*2</sup>に加えて、500円の「SL 整理券」を窓口で購入する必要がある。JRと共に改札を抜けて右手に進むと真岡鐵道のホームがあり、遠くには見慣れないレトロな雰囲気の客車が見える。胸騒ぐ期待に足取りも軽く歩を進めると、溜め込んだ熱を持て余すかのように煙を吹きあげる、黒い機関車が姿を表した。



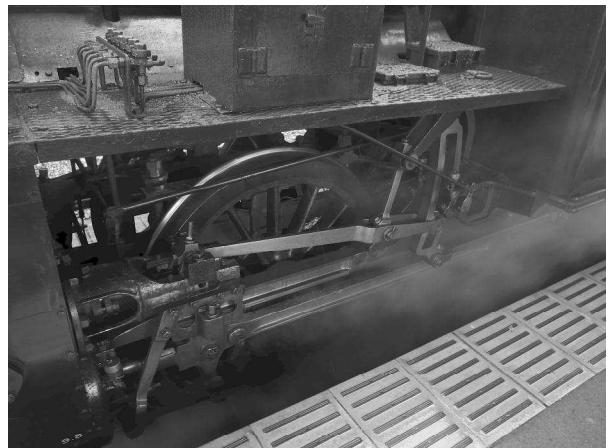
\*1 主に真岡線内ではC12が用いられ、代わりにC11はJR東日本に貸し出される場合がある。

\*2 出発駅の下館から終点茂木まで全線乗り通すと1000円になる。

ちょうどこの日は雪が降っており、吐き出す蒸気と相まった白い視界の中にたたずむ重厚な存在感は、さながら雲海に沈む山岳のようである。ずっとこのまま眺めていたいのは山々だが、発車時刻はそう遠くない。さっそく客車に乗り込むことにしよう。これから約1時間半、このロマンあふれる列車に揺られて茂木に向かうことになる。

座席に座って少しすると、列車はゆっくりと動き出した。電車やディーゼル気動車とはまた違った、独特の振動が伝わってくる。はっきり言って乗り心地は快適とは言い難いが、窓の外を流れていく白煙を見ていると、そんな振動さえも心地良く感じるような気がする。機関車から力強い汽笛が聞こえると、否応なしに気分も高揚するというものだ。

暫く走ると、真岡駅に到着した。ここで暫く停車する時間があるので、少し外出することにする。先程は詳しく観察する時間の無かった機関室を覗いてみよう。中には幾つものパイプやバルブが並んでおり、各所に蒸気を送り出す姿はまさに機関車の心臓だといえる。機関室を見終えたあとは、動力を他の車輪に伝える連結棒などを観察しているうちに、発車時刻が迫ってきた。座席に戻って発車を待とう。



再び動き出した列車は、降り積もる雪など物ともせずに走り続ける。気づけばいつしか、車窓からの風景は真

つ白に染まっていた。ここまで景色が変わると、日帰りだということも忘れてすっかり小旅行気分<sup>\*3</sup>である。

そして蒸気機関車の旅路を堪能しているうちに、行程も最後の茂木駅に到着した。この心地良い振動に別れを告げ、駅のホームに降り立つ。しかし、楽しみはこれで終わりではない。終着駅にはまた別の「お楽しみ」が用意されているのだ。



ターンテーブル

ここ茂木駅ではホームから線路の近くまで繋がる歩道が用意されており、転車台の上で方向転換する機関車をごく近くで眺められるようになっている。写真を撮る観光客のために回転を一時停止してくれるなど、サービス精神も満載である。この時点でかなりの雪が積もっていたが、何人もの「同業者」が寒さに震えながら写真を撮っていた。せつかくなので、転車台で方向転換する様子をご覧いただこう。



\*3 取材当日は1月14日、関東全域で大雪が降った日なので、実際本当に「気分」だけである。おそらく筑波にいても雪景色自体は見られただろう。

完全に後ろを向いた機関車を見届けて、今日のイベントは漸く全て終了となった。名残惜しいが、この辺りで機関車に別れを告げることにしよう。こうして蒸気機関車は80年もの時を超えて、人々の夢とロマンを乗せて今も走り続けているのである。



余談ではあるが、駅の待合室では地元の方々が観光客向けに温かい黒豆茶などをサービスしており、寒い日でも十分に暖まることができた。次は時期を変えて紅葉の季節にでも訪れてみたいものである。山沿いを走るこの路線ならば、雪景色とはまた違った表情を楽しませてくれるに違いない。

# tmux を便利に使う

文 編集部 坂口和彦

## 1 はじめに

tmux(terminal multiplexer)<sup>\*1</sup> というソフトウェアがある。tmux はターミナル上で複数のターミナルの画面を管理するためのソフトウェアであり、図 1 のような見た目をしている。

```
[pi3027@pi3027-mba] (0)% ls
#Nat.v# Example.glob Nat.vo embed.m4
Bool.glob Example.v README.mkd lib.ps
Bool.v Example.v.d Template.glob lib.ps.m4
Bool.v.d Example.vo Template.v stdlib_ext.glob
Bool.vo Makefile Template.vo stdlib_ext.v
Core.glob Makefile_cod Template.v.d stdlib_ext.vd
Core.v Nat.glob build stdlib_ext.vo
Core.v.d Nat.v config.el
Core.vo Nat.v.d coq2ps

[pi3027@pi3027-mba] (0)% [redacted]
* 738a49d (origin/master, github/master, master) subst_eval
* 2d242db revert 2 commits
* 10876d9 fix
* 05b9ce5 evalpartial
| * f3b4fe5 (origin/slides, github/slides, slides) 2013-01-21
* | 7e29dd8 evalpartial
* | 476ec57 add config.el
* | 1ec4e0f rename
| * 74aa070 2013-01-21
* | 1c8248c simplify
* | 7c59dd1 ssreflect
| * 1cbe743 2012-12-21
* | 033234f improve the postscript library
| * b647784 2012-12-21
* | 8382b41 ssreflect
| * e3c2f31 2012-12-21
* | 87e673e embed.m4
:
[pi3027-mba] (0) [0: ~/word/2013-02/articles/pi3027% omake -P] [1: ~/wrk/formalized-postscript%] [2: %] [3: % vim]

[109] exists i3 : inst, instnat_spec (Plus.tail_plus n m) i3 /\ 
[110] (i2 :: i1 :: vs, instnat_add_tail :: cs) |=> (i3 \
[111] :: vs, cs).
[112] Proof.
[113] eexists=> n m i1 i2 vs cs H H0.
[114] evalpartial' evalswap.
[115] evalpartial' evalpush.
[116] evalpartial' evalswap.
[117] evalpartial' evalexec.
[118] =>evalpartial (eval_instnat_repeat n).
[119] clear H11; move: n m i2 H0; elim=> [ | n IH] m i1 H \
[120] ; simpl.
[121] - by evalauto.
[122] - edestruct (instnat_succ_proof m i1) as [i2 [H1 H2]\ 
-UUU:>--F1 Mat.v 25% L116 Git-master (Coq Scr
[123] i2 : inst
[124] vs : stack
[125] cs : stack
[126] H : instnat_spec n i1
[127] H0 : instnat_spec m i2
[128] =====
[129] exists i : inst,
[130] instnat_spec (Plus.tail_plus n m) i /\ 
[131] (i1 :: ?1967, i2 & vs), [i1 :: i1, ?2034 & cs]) |=> (i \
[132] :: vs, cs)
-UUU:>--F1 *goals* Bot L10 (Coq Goals) ----
-UUU:>--F1 *response* All L1 (Coq Response) ----
-UUU:>--F1 *response* All L1 (Coq Response) ----

[pi3027-mba] (0) [0: ~/word/2013-02/articles/pi3027% omake -P] [1: ~/wrk/formalized-postscript%] [2: %] [3: % vim]
```

図 1: tmux

tmux は主に以下の機能を提供する。

- 画面を縦や横に分割し、分割したそれぞれの領域に画面を配置する機能<sup>\*2</sup>
- ウィンドウを複数作成し、それらを切り替える機能
- ペインのリサイズ、移動、プリセットレイアウトによる自動再配置
- キーボード操作だけでできるコピー&ペーストの機能と、コピー&ペーストのための複数のバッファ
- 強力かつドキュメント<sup>\*3</sup>の充実したスクリプトを書くのに便利なコマンドインターフェース

\*1 <http://tmux.sf.net/>

\*2 分割された下にあるそれぞれの画面をペインと呼び、分割される元の画面全体をウィンドウと呼ぶ。

\*3 [man tmux](#)

この記事は、既に tmux を使っている人に向けて tmux の便利な使い方を紹介することを目的としている。WORD 19 号「一斗缶 special edition 号」には Flast さんによる「GNU screen から tmux へ」というより入門向けの記事があるので、tmux を使ったことがない読者はまずこれを読むのが良いだろう。

また、私が zsh<sup>\*4</sup> ユーザであるために zsh を使っていることを前提とした項目が多数ある。zsh ユーザでない tmux ユーザにはあまり向いてない記事かもしれないが、目的を達成する上で使った仕組みについても少し触れているので、再実装して試せるかもしれない。

## 2 コマンドの実行終了をステータスラインで通知する

[pi8027@pi8027-mba] (0)% █ [~]

[pi8027-mba][1] [0:~% sleep 10] [1:~%] [2:~%] [3:~%] [4:~%] █

図 2: 0 番目のウィンドウで sleep 10 を実行している tmux

[pi8027@pi8027-mba] (0)% █ [~]

[pi8027-mba][1] [0:~%] [1:~%] [2:~%] [3:~%] [4:~%] █

図 3: 0 番目のウィンドウで sleep 10 を実行し終えたことを通知している tmux

tmux の下で実行終了までに長い時間かかるコマンドを実行した場合、しばらく別のウィンドウを見ていてそのコマンドの実行終了に気付けない場合が多い。この問題を解決するために、コマンドの実行が終わるとステータスラインの該当するウィンドウの部分の色が変わるようにしたい。

---

\*4 UNIX シェルの一種。詳しくは <http://www.zsh.org/> を参照。

この色を変える方法はいくつかあるが、ペル文字が出力されてからまだ見ていないウィンドウは色が変わるようになっているので、ここではそれを利用する。つまり、コマンドの実行を終える度にペル文字を出力するようにシェルの設定を書く。例えば、zsh では `precmd` フックを使って以下のように書ける。

リスト 1: コマンドの実行終了をステータスラインで通知するための zsh の設定

```
if [[ -n $TMUX ]]; then
    function _tmux_alert(){
        echo -n "\a"
    }
    add-zsh-hook precmd _tmux_alert
fi
```

この設定を試すには、`sleep 2` を実行して直後にウィンドウを切り替えてみると良い。2 秒後に `sleep` コマンドの実行が終了すると、ステータスラインのそのウィンドウの部分の色が変化する。

### 3 ペインの pstree を見る

図 4: ペインの pstree を見る - 実行例

`tmux` を使っていて、特定のペインの下のあるプロセスの PID を調べたい場合がある。このような場合に、特定のペインの `pstree` を見られると便利である。これを `tmux` のキーバインドから呼び出せるようにするには、設定に以下の行を追加する。

リスト 2: ペインの `pstree` を手軽に見るための `tmux` の設定

```
bind-key p run-shell "pstree -p $(tmux display-message -p '#{pane_pid}')"
```

これを追加すると、`prefix-p` で今見ているペインの `pstree` をいつでも見られるようになる。

## 4 tmux のバッファを編集する

tmux のバッファを単純にコピー&ペーストのために使うだけでなく、テキストエディタでバッファを編集したい場合がある。tmux のバッファの読み出し、バッファへの書き込みは、`save-buffer`、`load-buffer` コマンドができるので、一時ファイルとこれらのコマンドを使うとバッファを編集するコマンドを作れる。以下は、zsh での設定の例である<sup>\*5</sup>。

リスト 3: tmux のバッファを編集する `tmux-editbuf` コマンド

```
tmux-editbuf(){
    local buffer=$1
    local file=' tempfile'
    if tmux save-buffer -b "${buffer:-0}" $file ; then
        buffer=${buffer:-0}
    elif [[ -n $buffer ]] ; then
        return 1
    else
        sleep 0.5
    fi
    $EDITOR $file
    if [[ $(wc -c < $file) -eq 0 ]] ; then
        [[ -n $buffer ]] && tmux delete-buffer -b $buffer
    else
        tmux load-buffer ${buffer:+-b $buffer} $file
    fi
    rm $file
}
```

上に示した `tmux-editbuf` コマンドは、引数としてバッファの番号を取れる。指定しなければ 0 番を使う。エディタは `EDITOR` 環境変数の値を使用するので、これを必ず設定しなければならない。

## 5 tmux のバッファとクリップボードの間で複製をする

tmux のバッファと (OS などが持っている) クリップボードを同期したいと考える人は多いだろう。これはかなり難しい問題だが、その代替として tmux のバッファとクリップボードの間で文字列を複製することはできる。

私は普段 Linux(X11) 環境と Mac OS X 環境を使っているので、まずはこの差を吸収するために `clipboardin`、`clipboardout` というスクリプトを書いた。

リスト 4: `clipboardin`

```
#!/bin/bash
case `uname` in
Linux) xclip -i -selection clipboard;;
Darwin) pbcopy;;
*) exit 1;;
esac
```

---

<sup>\*5</sup> 少し試した限りでは bash でも問題無く動いた。

## リスト 5: clipboardout

```
#!/bin/bash
case `uname` in
Linux) xclip -o -selection clipboard;;
Darwin) pbpaste;;
*) exit 1;;
esac
```

これを使って、tmux のバッファとクリップボードの間で複製を行うコマンドを書いた。

## リスト 6: tmux のバッファとクリップボードの間で複製を行うコマンド

```
tmux-loadcb(){
    clipboardout | tmux load-buffer $@ -
}

tmux-storecb(){
    tmux save-buffer $@ - | clipboardin
}
```

`tmux-loadcb` を使うことで、クリップボードの中身を tmux のバッファに複製できる。`tmux-storecb` を使うことで、tmux のバッファの中身をクリップボードに複製できる。

## 6 新しいペインでコマンドを実行する

今、tmux の下である Git<sup>\*6</sup>リポジトリのログを見たかったとしよう。しかし、単にログを見たいのではなく、ログを見ながら作業を続けたいとする。このとき、`git log` と入力して、そのまま作業を続けると表示されたログは流れていってしまう。ペインを分割すればログを見ながら作業できるが、ペインを分割して同じディレクトリに移動してから `git log` と入力するのは、冗長な操作である。これを単に `np git log` と入力するだけで済むようにする `np` コマンド<sup>\*7</sup>を実装した。ただし、この例では Git のページャとして `less`などを設定<sup>\*8</sup>していなければすぐに新しく作られたペインごと消えてしまうので、その設定をする必要がある。

現時点での `np` コマンドの実装は zsh 向けのものであり、補完関数を含めて `tmux-np.zsh` という 1 つのファイルになっている。これを `$ZDOTDIR/tmux-np.zsh` に複製して、zsh の設定ファイルに以下の記述を追加すると、`np` コマンドを使える。

## リスト 7: np コマンドを使うための zsh の設定

```
. $ZDOTDIR/tmux-np.zsh
```

\*6 バージョン管理システムの一種。詳しくは <http://git-scm.com/> を参照。

\*7 <https://github.com/pi8027/config/blob/master/zsh.d/tmux-np.zsh>

\*8 `less` を設定するなら `git config --global core.pager less` を実行する。

## 7 ペインを最大化する

tmux で分割をし過ぎて小さくなつたペインを、分割されていないウィンドウと同じ大きさにしたい（最大化したい）場合がある。これだけなら `break-pane` コマンドを使えば良いのだが、最大化したペインを元の大きさ、位置に戻したいとなると、別の方法を考えなければならない。これを実現するソフトウェアとして、今まで `tmux-zoom`\*<sup>9</sup>が良く知られていた。しかし、`tmux-zoom` は以下に挙げる複数の問題を抱えている。

- ウィンドウのタイトルを自動的にアップデートするように設定をしていると、上手く動かない。
- 最大化したペインが元々あった位置には別のペインが作られるが、普通の新しいペインと区別が付かない。そのペインを消すと元の位置に戻す操作ができなくなる。元の位置に戻す操作をするとそのペインは消えるので、そのペインを使っていると意図せずに情報が失われる場合がある。

そこで、`tmux-zoom` を参考に `pane-maximize`\*<sup>10</sup>というソフトウェアを開発した。これは、上に挙げたものを含む `tmux-zoom` が抱えていた多数の問題を解決し、とても使いやすいツールとなっている（と思う）。`pane-maximize` を使うにはそのスクリプトを適当なパスの通った位置にコピーし、`tmux` の設定ファイルに以下の行を追加する。

リスト 8: `pane-maximize` を使うための `tmux` の設定

```
bind-key -r m run "pane-maximize -a"
```

これで、`prefix-m` でペインを最大化でき、すでに最大化されているペインは元の位置に戻せる。これは今までの内容の中で最もおすすめしたい設定である。

## 8 まとめ

ここまで、6つの `tmux` の便利な使い方、設定を紹介してきた。このうち半分以上は、シェルスクリプトとしてその一部分もしくは全てを実装している。`tmux` の魅力の1つは、このようにして簡単なスクリプトを書くことで機能を追加できるところである。もう1つ挙げるとすれば、Cプログラミングができれば容易に改造し機能を追加できる点である。`tmux` の類似ソフトウェアとして `GNU Screen`\*<sup>11</sup>があるが、長い歴史を抱えているために実装はとても複雑で、簡単には手が出せなくなってしまっている。`tmux` を使っている人はここに紹介したものに限らず、自分が `tmux` に望む機能を自分の手で追加してみると良いのではないだろうか。

また、`tmux` に限らず、私が使っている設定ファイル集はそのほとんどの部分を公開\*<sup>12</sup>している。もし興味があれば、見てみると良いかもしない。

---

\*<sup>9</sup> <https://github.com/jipumarino/tmux-zoom/>

\*<sup>10</sup> <https://github.com/pi8027/pane-maximize/>

\*<sup>11</sup> <http://www.gnu.org/software/screen/>

\*<sup>12</sup> <https://github.com/pi8027/config/>

# New comer Haswell

文 編集部 iorivur

## 1 おまじない

Haswell<sup>1</sup> がそろそろ出そうなので、ざつとどんなマイクロアーキテクチャなのか紹介します。といつても、Intel アーキテクチャをすべて概観するわけにはいかないので、あたらしいおもしろ機能を中心に緩く紹介します。

## 2 パッケージ

デスクトップ向けのパッケージは LGA1150(SocketH3) となります。え、聞いたことないって？はい、マザーボードも買いましょう。サーバ用途の製品のパッケージは、LGA2011 を置き換えて SocketR3 になる、らしいです。

チップセットについては、コードネームは Lynx Point と呼ばれている、Z87, Z85, H87, Q87, Q85 及び B85 がリリースされると言われています。Haswell の次の CPU、Broadwell でも同じチップセットが使えるという噂もあります<sup>2</sup>ので今買っても安心です<sup>3</sup>。

## 3 FIVR

さて、最初のおもしろ機能は、FIVR<sup>4</sup> という、内蔵 VRM です。

### 3.1 VRM とはなんぞ

VRM(Voltage Regulator Module) とは、文字通り、電圧のレギュレータモジュールです。いまどきのナウでヤングな半導体は、3.3V や 5V といった暴力的電圧で動作させようとしてはいけないので、それ以下、1.5V 以下の電圧に下げる必要があります。さて、いまどきの CPU たちは、電流を大量に要求するので、効率よく降圧を実現するには、LDO のような損失の大きなレギュレータではなく、よりもむしろ DC-DC コンバータ<sup>5</sup>で実装することになります。

そして、そのモジュールはマザーボード上に配置されていました<sup>6</sup>。当然、マザーボード上に VRM 回路があると、箱の電源モジュールからケーブルを引き出すよりもずっと配線の引き回しが減りますし、CPU の近くに配置できるので、インピーダンス計算もずっと手間が省けます。

さて、VRM がどのように動いているか、さくっと説明しましょう。サクッとそれっぽく書いた図がこちらになります。

<sup>1</sup> Ivy Bridge の次のマイクロアーキテクチャ。マイクロアーキテクチャというのは、ISA(Instruction Set Architecture, いわゆるアーキテクチャ。Intel 64 とか MIPS とか Thumb2 とか) をさらにどうやって実装するのかという、さらに細かいアーキテクチャを指す

<sup>2</sup> なんでも、チップセットに新たな機能を追加する必要性が薄れてきているためとか

<sup>3</sup> 要出典

<sup>4</sup> fully integrated voltage regulator: ふあいう“あー”と発音するらしい

<sup>5</sup> 直流 - 直流電圧変換。(そのまんまじゃないか……)

<sup>6</sup> 元は電源メーカから提供されるような仕様でした

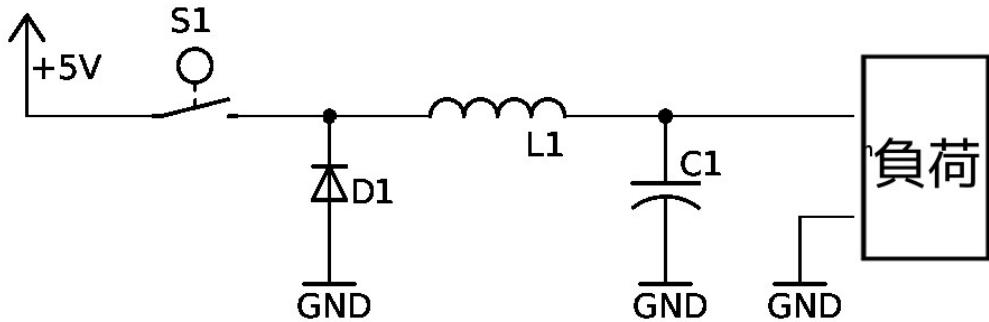


図1 簡易 VRM 回路

高校物理を履修なさった皆さんならばコイル L1 には定常的な電流が流れるようにする能力があるとご存知でしょう。つまり、

$$V = -\frac{dI}{dt}$$

なので、S1 が瞬時に切れても、D1-L1-負荷-GND の閉路の電流が一定になるように L1 に誘導起電力<sup>7</sup>が発生します。それでもって S1 がオンになったりオフになったりを高速で繰り返すと、ある一定の電圧の付近で常に負荷に電圧がかかることになります。

なので、この S1 のデューティ比をきちんと制御する事で、一定の欲しい電圧を作り出すことができます。

ちなみに今時の CPU は (Core i7 3770K を例にとると)、TDP77W 程度で定格電圧が 1.1V 程度ですので、ざっと見積もって 70A くらい流れているわけです。誤解を恐れずに言えば、アパートの契約電流<sup>8</sup>が 20A くらいですので、精密半導体に流している電流とは思えぬ膨大な電流が流れている、要求されていることに気づきます。これらの電流はすべてマザーボードの隣にレイアウトされた VRM から流れています。

<sup>7</sup> 力とかいてあるものの、実際の次元は V です

<sup>8</sup> 100V 交流では事情が異なりますが

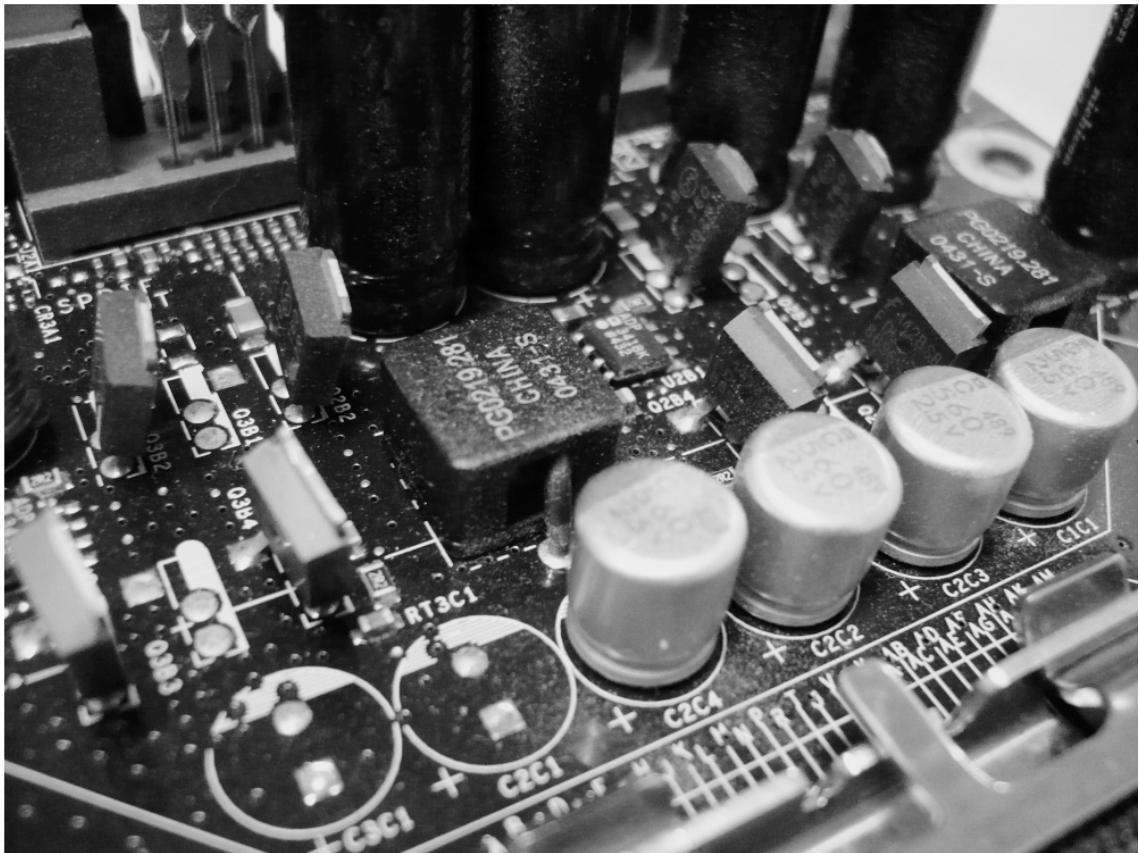


図2 Intel 製 LGA775 マザーボードにおける 2 フェーズ VRM 回路

### 3.2 なにが変わるの

ところで、みなさんは 0 距離射撃ってご存知ですか。

Intel は 0 距離射撃を始めます。VRM を CPU の近くにレイアウトするよりも CPU に内蔵した方が、より制御もこまめにできるようになり、CPU の機能ブロックごとに小分けしてそれぞれに適正な電圧を動的に生成できるようになります。コアごと、GPU 部、キャッシュ部、それぞれ別の電圧を与えられるようになります。

マザーボードベンダによる違いも吸収できるという利点が得られます。ぎょえ～～、変態だよう～～。やっぱり近距離恋愛よりも同棲だね！

さて、上で見たようなパワー系の回路をどのようにしてひとつの CPU パッケージに収めるのでしょうか。いままでも CPU ダイにそれだけの電流が流れていたわけですが、電圧変換をするとなると、それなりのさらなる<sup>9</sup> 熱損失を覚悟しなければなりませんし、そもそもそんなでかいインダクタどこにつければいいのか皆目検討がつきません。CPU

<sup>9</sup> FET の寄生ダイオードによる順方向の電圧降下がいちばん大きいんじゃないですかねえ……

パッケージからフェライトコアが二本、まるでプルートゥに襲われたかのごとくつきだしているのは避けたいところです<sup>10</sup>。

安心してください、いくら LGA1150 なぞ聞いたことがないとは言え、そんな事はありません。FIVR ではひとつ CMOS モジュールに集積されて、CPU ダイと一緒にひとつのパッケージになかよく載ることになります。

ナムサン！ひとつの CMOS モジュールだって？そんなことがあっていいのでしょうか。コイルはどこにいったのでしょうか。実はちゃんと CMOS プロセスとなかよくひとつのダイに生成されるようになっています。さて、それにはどのようなテクがつかわれているのでしょうか。

今までの CMOS プロセスも、銅配線はダイの中に生成できました。まあ、それはあたりまえですよね。ケイ素と仲良くなりにくく<sup>11</sup>銅も、バリア<sup>12</sup>を生成してからならば、Si や SiO<sub>2</sub> 中に拡散して特性を害することもないです。やった！有機金属気相成長法バンザイ！仲を取り持ついい奴だ！

そんなこんなで、CMOS プロセスを知らなかつたテレビの前のみんなも、明日から CMOS プロセス中に銅のワイヤが生成できるようになりました。しかしどう考えても一本の銅ワイヤだけではインダクタンス足りないですよね<sup>13</sup>。そこで、次にこのインダクタンスを高めたいのですが、いい方法はないでしょうか。

コアを付けたいところですが、コアの周りにワイヤを巻き巻きするのはたいへんです。そこで逆にワイヤの周りに強磁性体を巻きましょう。え？それで本当に磁束密度が高まるかって？ご安心ください、コアを主体に巻き巻きしなくとも、当然、ビオサバル<sup>14</sup> の法則から、直線導線中を流れる電流 I にも、その導線の周辺の点における磁束密度 B に影響する権利があることがわかります。手で確かめるときは、

$$d\mathbf{B} = \mu_0 \frac{Idl \sin \theta}{4\pi r^2}$$

の、lだけを積分すればいいわけでないことに注意してください。ほかの変数も dl の位置によって影響を受けます。座標系に置いた方が簡単かもなあ。

ま、そういうわけで、銅ワイヤの周りに磁性体の筒があつても力強いインダクタンスが得られたことにします<sup>15</sup>。

しかし、これで納得というわけには行きません。どうやって CMOS プロセス中にそんな強磁性体を入れられるのでしょうか。どんなに小さなフェライトコイルをつくってるの？そんなに小さいフェライトコイル、コアメモリ以来だよ<sup>16</sup>！いやいや、フェライトではありません。

---

<sup>10</sup> 個人的には、CPU パッケージに水冷用のノズルが二本あいてそこに液体を流すとかあっても夢があつていいと思う

<sup>11</sup> アルミニウムに比べ

<sup>12</sup> バリアが生成できなかつた時代はアルミニウム配線でした

<sup>13</sup> 逆に足りてたらむしろ今まで困ってただろ

<sup>14</sup> バザールでござ～る♪

<sup>15</sup> じゃあなぜ普通のコイルは巻き巻きしてあるかって言うと、何度も巻いた方が一つの貫通する磁束を何度も拾えるから、もっと正確に言うと、見かけのループの面積 S が大きくなるから

<sup>16</sup> んなアホな

本当は CoZrTa をフィルム状にしてラップするように生成している<sup>17</sup> ようです。この磁性体フィルムの層を 1 つではなく、2 つ入れ子にかぶせてやることで、さらにぐんと威力を發揮します。おまけに、中の銅ワイヤもがんばって巻いてやると、ある資料<sup>18</sup> では 2 層の磁性体と 8 回巻の銅ワイヤで 50nH まで<sup>19</sup> インダクタンスを向上させています。

それでもいままでよりはインダクタンスが小さいはずです。どうしてうまく動くのでしょうか。それはスイッチング周波数を高めることにあります。プロトタイプはだいたい 30MHz から 140MHz くらいで動かしているそうです。たしかに超高速スイッチングは Intel にとってお手の物ですからねえ。インダクタンスが小さくてもローインピーダンスじゃないから恥ずかしくはあんまりない！

しかもこれならば、高い頻度でデューティを変えられるので、もっと細かい時間単位で電圧制御をしたいという欲求に応えることが出来、時間方向での電圧制御の粒度をあげることもできます。

すごい！やった！制御時間粒度もロリの時代！

そんなわけで、CPU のオンパッケージ、もしくはオンダイ VRM が実現する時代が来ました。これによって、CPU の消費電力削減がさらに進むでしょう。CPU にフィードするマザーボード上の VRM を一系統に減らすことができるようになり、マザーボードもよりシンプルになります。

そういえば Ivy からすでに Tri-Gate を使ってるんでしたね。これもおもしろいのですが、ここで紹介するには趣旨が違う上、紙面が足りぬです。簡単に紹介すると、フランスパンを焼くとき、片面からよりも 3 面から焼いた方が早くこんがりと焼きあがるのと同じことです。でも、FET のゲートをあんな形に生成するのすごいよなあ。どうやってるんだろう。やっぱり Si をもっと深くえぐれるようになったからかなあ。

## 4 Hardware Transactional Memory

次のおもしろ機能は Transactional Synchronization eXtension です。トランザクションを知らないみなさんも、よっぽどアイドル性が高くなき限りはトランザクションしたことがあると思います。たとえばトイレとか。

### 4.1 とらんざくしょなるめもりい ??

Wikipedia がトテモ詳しいのですが、説明します。ここに、オンライン対戦恋愛シミュレーションゲームを作るとしましょう。好感度変数がいつ変更されるかスレッド別に考えてみます。

まず世界に男性プレイヤは「お兄ちゃん」と「同級生」しかおらず、世に女性は「攻略対象ちゃん」1 人しかいないと仮定します。「お兄ちゃん」の好感度は、「攻略対象ちゃん」と「お兄ちゃん」が一緒にいる間だけ、つまりアットホームシチュエーションだけしか変更されず、「同級生」の好感度は「攻略対象ちゃん」と「同級生」が一緒にいる間だけ、つまり学校シチュエーションだけしか変更されないとします。

また、自分が思っている予想好感度は、実際の好感度の差分から推定することにします。

そうすると、彼らのアルゴリズムはこうなります。

<sup>17</sup> 他の材料でも研究されているらしいが

<sup>18</sup> [http://pc.watch.impress.co.jp/docs/column/kaigai/20121227\\_580258.html](http://pc.watch.impress.co.jp/docs/column/kaigai/20121227_580258.html)

<sup>19</sup> @10MHz、1GHz では 20nH くらい

**Algorithm 1** お兄ちゃんのアルゴリズム**Require:** アットホームシチュエーション

```

 $k \leftarrow k + \Delta$  好感度
 $x \leftarrow$  会話メソッド()
if  $x$  is Excellent then
    好感度 ++
 $k ++$ 
else
    if  $x$  is GOOD then
        --
 $k --$ 
    else
        if  $x$  is BAD then
            好感度  $\leftarrow 0$ 
 $k \leftarrow 0$ 
        end if
    end if
end if

```

**Algorithm 2** 同級生のアルゴリズム**Require:** 学校シチュエーション

```

 $k \leftarrow k + \Delta$  好感度
 $x \leftarrow$  会話メソッド()
if  $x$  is Excellent then
    ++
 $k ++$ 
else
    if  $x$  is GOOD then
        --
 $k --$ 
    else
        if  $x$  is BAD then
            好感度  $\leftarrow 0$ 
 $k \leftarrow 0$ 
        end if
    end if
end if

```

とりあえずこれで完璧です。

しかし、このゲームが進行していくにつれ、ついに「攻略対象ちゃん」が「同級生」を家に連れ込みにくるようになりました。「同級生」も、ついには Require 学校シチュエーションではなくなつたのです。ここで、「お兄ちゃん」は兄らしい毅然とした態度に出て、二人の会話に混じることになりました。これが悲劇の始まりだとは、このときだれも予想していなかつたのです。

この 3 人が同時に話すようになったので、「お兄ちゃん」プロセスと「同級生」プロセスは同時に走るようになりました。

この状況で、上のアルゴリズムでお互いが動いたらどうなるでしょう。たとえば、「お兄ちゃん」がせっかく「攻略対象ちゃん」とはなしているのに、途中から「同級生」がはなしかけました。そして、「お兄ちゃん」ははなし終えたので好感度がせっかくアップしました(たとえば、3 から 4 へ)のに、後から同級生が「お兄ちゃん」の悪口を吹き込んで、「お兄ちゃん」の好感度を後からひとつ下げました。このとき、「お兄ちゃん」の好感度はいくつでしょう。

ほんとうは、3 から 4、そして 3 に戻るはずが、3 から 4 になったのを無視して 3 から 2 に下がつてしまします。もし「お兄ちゃん」の好感度が 1 つ下がっている、と「お兄ちゃん」が後から知つても、ほんとうは 2 になつてしまつているのに、好感度は 3 になつたと思いつづけてしまいます。こうして二人はすれ違つて行くのですね。

こういった整合性のミスを起こさないために、いくつか方策があります。そのうちの一つは、ひとりずつ話そう、ということです。

**Algorithm 3** ロックなお兄ちゃんのアルゴリズム**Require:** アットホームシチュエーション**Require:** LOCK is FreeLOCK  $\Leftarrow$  Taken

恋愛処理

LOCK  $\Leftarrow$  Free**Algorithm 4** ロックな同級生のアルゴリズム**Require:** 学校シチュエーション or アットホームシチュエーション**Require:** LOCK is FreeLOCK  $\Leftarrow$  Taken

恋愛処理

LOCK  $\Leftarrow$  Free

しかし、これでは3人でたのしく話すことができません。並列度を上げた高速な知的恋愛ゲームを楽しむためには、もうちょっと改良する余地がありそうですね。

並列に「お兄ちゃん」と「同級生」が走りながら、かつさつきみたいなずれ違いをなくしたい。変更されないようにロックするのではなく、逆に、代入の直前に、もういちど本当にその資源が最初に参照されてからのちに変更されていないかどうか、書き込む側が確かめればよいのではないでしょうか。

**Algorithm 5** 好感度アップデート処理好感度<sub>old</sub>  $\Leftarrow$  好感度

中処理

**Require:** LOCK is FreeLOCK  $\Leftarrow$  Taken**if** 好感度<sub>old</sub> is 好感度 **then**    好感度  $\Leftarrow$  好感度<sub>new</sub>**else**

Abort();

**end if**LOCK  $\Leftarrow$  Free

ここでLOCKは、変更されていないことを確認してから書き込むまでの間にはかのライバルに書き換えられてしまわないように設置してあります。これによって並列に動作しないかもしれない部分は検査と代入の箇所だけになります。しかも、その間に別のプロセスが別のことをしていればまったく並列に動きます。Abort()は、今までのトランザクションを放棄して、以前の状態にロールバックする関数とします<sup>20</sup>。

こういうわけで、この様な処理によって上述の利点を享受しようというのがトランザクショナルメモリです。

<sup>20</sup> 実世界での実装がまたれますね

## 4.2 ハードウェアなトランザクショナルメモリと Intel

そもそもトランザクショナルメモリはハードウェア実装がオリジナルで、これを無理やりソフトウェアで擬似的に実現したものがソフトウェアトランザクショナルメモリ(STM)というものです。しかし実際はそのコスト上の問題からSTMのほうが多く用いられてきました。それでは今までIntelは何をしてきたんでしょうか？トランザクショナルメモリに興味がなかった？

いや、そういうわけではないのです。いままでもSTMを支援する機構としては、SSE3時代から命令セットレベルでサポートされていました。例えば、monitor命令などがそれです。これを使えば、こうして競合状態になるかもしれないメモリを全部監視対象にしておいて、もし誰かに書き換えられてトランザクションが失敗したらmwaitをしていた別のプロセスが仲介をする、たとえば今回残念だった方のプロセスに人生をやり直してもらうなどをするような親の監視プロセスを作つて置いておくなどが可能になります。

しかし、Haswellから搭載されるTransactional Synchronization Extensions(TSX)は、さらに進みます。この拡張が提供する機能は大きく分けて2つです。それはHardware Lock Elision(HLE)とRestricted Transactional Memory(RTM)です。

まず前者について説明します。

### 4.2.1 HLE

ここに、ロックを用いたコードを書いたとします。

---

#### Algorithm 6 『時そば』の支払い処理

**Require:** LOCK is Free

```

LOCK ← Taken
while recentUsedNumber < 16 文 do
    受け取る
    recentUsedNumber++
end while
LOCK ← Free

```

---

ここで、アトミックな処理が会計処理でないと、「今何刻デエ」という別のクエリによってrecentUsedNumberが破壊されてしまうかもしれませんので、ぜひともロックをかけたいところです。

しかし、「今何刻デエ」ときかれて答えている間会計処理のできない愛想のない店員にはなりたくありません。江戸でそんなことをしたらたちまちチャキチャキの粋な江戸っ子原理主義者に潰されてしまいます。そんなわけで、recentUsedNumberが破壊されない限りにおいては並列にクエリを裁ける粋な店員になります。

ここで、ロック変数を参照するのに特別な機能のあるXCHGや、LOCK prefixの付いているADD, ADC, AND, BTC, BTR, BTS, CMPXCHG, CMPXCHG8B, DEC, INC, NEG, NOT, OR, SBB, SUB, XOR, XADDなどに、さらにXACQUIRE prefixを付けた場合について説明します。つまり、HLEを利用する場合です。ちなみに、ここで言うprefixというのは、命令に付いているprefixのことです。Intel 64とIA 32の命令には、Instruction Prefix

という 1 バイトの印が 4 つまでつけられます。命令のフォーマットが分からぬ場合は Intel SDM<sup>21</sup> の Volume2 Chapter2 の Instruction Format のページをみるとわかりやすい図が上の方に出ています。どんな prefix があるかは、Intel SDM Volume2 Appendix A.3<sup>22</sup> の Table A-2 を参照してください。

さて、LOCK が Free か確かめるコードに XACQUIRE を付けたとします。こうしてから、LOCK で守られているはずのセクションに、ほぼ同時に 2 つのプロセスが侵入してきた場合どうなるのでしょうか。

実は後発のプロセスにもロックがかからず、どちらもアトミックセクションに侵入できます。

そのあとに二八そばを売りつけたら、最後に LOCK を Free するところで今度は XRELEASE prefix を付けて Free すると無事解決です。先にお会計が終わらなかつた方は残念でした。人生を、二八そばを買うところからやり直してください。

さて、じゃあどうしてロックがかからなかつたのか、また、後発のトランザクションが無事失敗したのか<sup>23</sup> 謎ですね。これは、ケイ素を採掘したアフリカ奥地の不思議な呪力が……ではなく、きちんとしたカラクリがあるのです。

いまどきの CPU たちはナウでヤングなので、それぞれのコアにキャッシュを持っていまして、本当にトランザクションが commit される<sup>24</sup>のは、そのキャッシュがメモリに吐き出される時です。

まず、L1 や L2 キャッシュがどのように働いているか概観します。キャッシュは、メモリを仮想的に近くに配置するためあります。メモリからデータを持ってくるときに、一度に大量に持ってきてキャッシュに吐き出して（メモリはバーストで読み出した方が速いから）、キャッシュから取り出します。

キャッシュ同士がもし排他制御されていなかつた場合には、ひとつのアドレス空間から 2 つのキャッシュがそれぞれ独自に値を持ってきてしまいかねません。そしてキャッシュが書き戻すときになって初めてその先のメモリの値が変わっているのに気づいて驚き桃の木山椒の木、自分は浦島太郎か、よもやこのアドレス空間に間男かやと、仰天してしまいます。仰天ですめばいいのですが、実際は気づかずにプログラムが暴走するのでやつてられません。そこで、キャッシュ同士は互いに紳士協定<sup>25</sup>を結んでひとつのアドレスとふたつのキャッシュが同時に付き合いして互いがそのメモリに書き込んだりするなどということはないようにしてあります。

<sup>21</sup> Intel® 64 and IA-32 Architectures Software Developers Manual <http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>

<sup>22</sup> Intel SDM Volume2 Appendix A.3 ONE, TWO, AND THREE-BYTE OPCODE MAPS

<sup>23</sup> 出遅れたのに成功したらむしろそのトランザクションは正しくないということに注意

<sup>24</sup> 成功して値が書き戻される

<sup>25</sup> MESI プロトコルとか

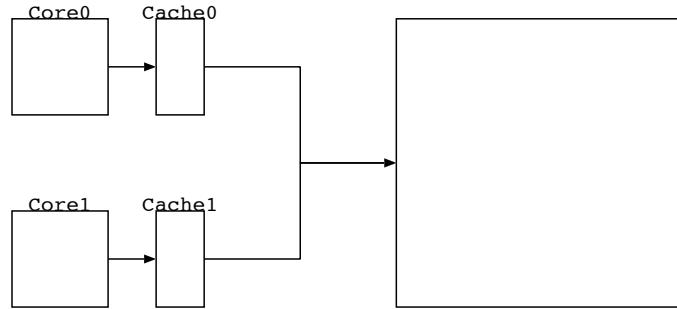


図3 矢印のようにデータがメモリに書き戻されていく

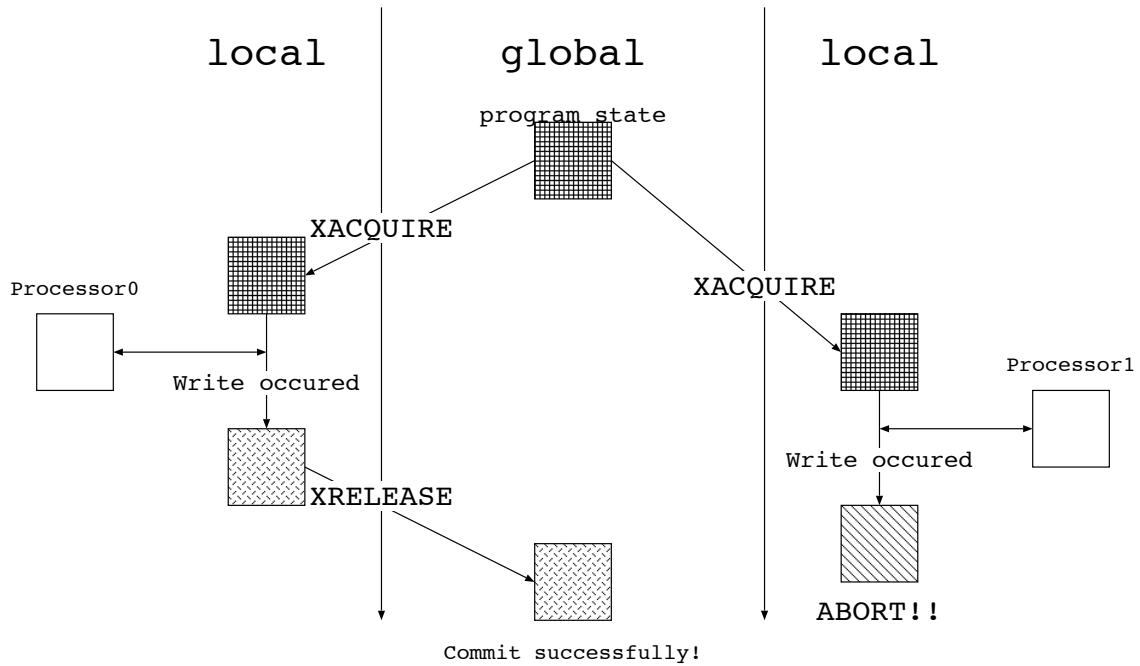


図4 ステートが一度ローカルに保持される

さて、翻って XACQUIRE と XRELEASE は何者でしょうか。この prefix がついた命令が発行されると、レジスタの状態をふくむそのプロセスの状態が一度保存され、そして新たにプロセスの状態がローカルにキャッシュされます。その新しい状態は、何が変更されたのか、何が参照されているのかというフラグを持ち、アトミックセクションだけなわになるにつれて汚れて行きます。いわば、メモリがレイヤをもってマスクされた状態ですね。

そうして、トランザクションが最後まで完走すると、その状態は最終的に commit されます。具体的には、Write された部分が書き戻されます。

キャッシュの例でみると、『汚した』アドレスを二つのプロセッサがお互い独自に持つことは今までよくないこ

と、避けるように実装されてきたことでしたが、ここではあえて恣意的に、commitされるまで別の状態を持ちながら走りつづけ、XENDが発行された時にはじめてそのキャッシュの整合性を得るようになっています。

失敗したトランザクション、図6右側は、先ほど保存したプロセスの状態にロールバックされます。「汚した」状態はちゃんと後からロールバックされます。つまり、変更はすべてキャッシュ止まりでメモリに書き込まれていなかつたものだから、キャッシュが無効になってしまえば破棄されたも同然、ということでしょうか。

実際にはこのreadとwrittenのアドレスの管理はキャッシュラインベースで行われるらしいので、無辜のアドレスも、近くのアドレスが書き込まれると「汚れた」状態だとみなされて肅正され、場合によっては実際必要な頻度よりも多くトランザクションが中止されますが、ひとつひとつ管理するコストを考えるとこうなるのでしょうか。

#### 4.2.2 RTM

HLEは、古いハードウェア<sup>26</sup>でも動かせるようにするために、わざわざ新しい命令を導入する事をやめてinstruction prefixを導入したのですが、一方RTMでは新しく3つの命令が追加されました。それがXBEGINとXABORT、そしてXENDです。さらに、HLEでも使えるXTEST命令も使えます。

さて、それぞれの命令について考えます。

XTEST命令はHLEでも使える命令なのですが、ここで説明してしまいますと、今現在がトランザクショナルメモリをサポートするこの状態(Transactional Execution)の実行下にあるかどうかを調べる命令です。

XBEGIN命令は、このトランザクションの始まりのアドレスを相対指定します。トランザクションが不幸にも失敗した時には、このアドレスに飛びます。そのとき、EAXレジスタは原因レジスタとなって、アボート事由がかかるれます。

XEND命令は、トランザクションが終了する箇所を指定する命令で、この命令まで無事にたどり着いた暁にはそのトランザクションの成功を意味します。バンザイ！そしてそのトランザクションはcommitされて二つの並行世界はマージされます。世界線は収束します。

不幸なトランザクション失敗の時はいつ起こるのでしょうか。もちろんXABORTが発行されたときにも失敗します。トランザクションのレース状態以外にも、海より深く山より高いオトナの事情というものがあるので、そういうときにソフトウェア的に失敗を宣言できます。デバッグに便利なようにちゃんと8bitの識別用のコードを埋め込みます。もちろん、トランザクションの失敗はハードウェア的に検出されます。ハードウェア的な検出のためにわざわざかち合う可能性のある、監視が必要な資源を登録しておく必要はなく、自動的にReadなメモリ領域を監視・マークしているので自然にプログラムがかける、らしいです。

すばらしいですね。

---

<sup>26</sup> 現在市販されているすべてのCPUは古いということになるが……

## 5 欲しい！いつ買えるの??

こんなに魅力満載の Haswell Intel CPU は 2013 年第二四半期発売です！しかもまだまだあたらしい機能があります！TDP10W 程度の製品も出る、らしい。すごいゾ！……うーん、やっぱりタブレット向けに食い込めないと、まずいのだろうかこのご時世。それなりに長い間ずっと移動端末向けにプロセッサ作ってたけど ARM に食われっぱなしだし XScale は売っちゃったしナア……

ところで Atom とかどうするんだろう、Atom にも最新プロセス投入するとかしないとかっていう話も聞くし、Intel も組み込み・移動端末アレルギーのままじゃまずいって判断したんだろうか。

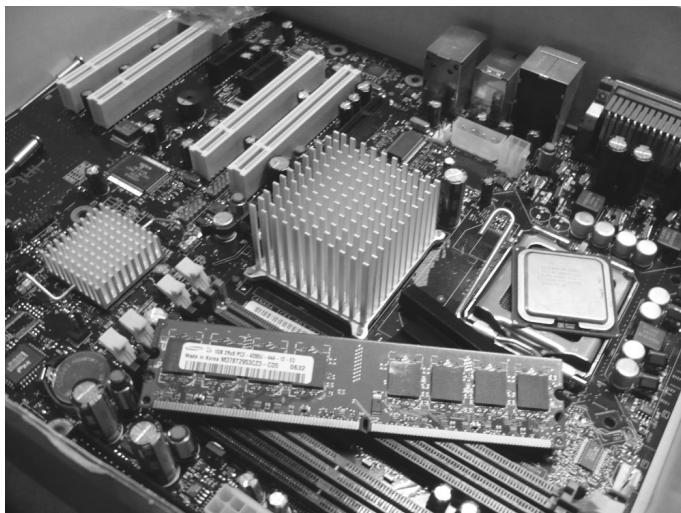
とにかく Haswell で組み込み向けにも Core i 系 CPU が組み込めたらいいかも知れない。この前、実家の近くに巨大ショッピングモールができる、大きさのあまり地図が見にくかったのですが、隣にテーブル UI の端末が置いてあって、それで業種別に検索できるなどずいぶんと便利だった。ただ少し動作が鈍かったように思うので、ああいった端末がきびきび動くとうれしいなと感じた。でもこの場合据え置きだからなあ。

移動端末で CAD を動かしたりするのかしら。医療用？

## 6 粗品の紹介

紙面が微妙にあまつたので粗品の紹介。Intel 特集、Intel デスクトップ向けマザーボード事業撤退発表にちなんで、筆者が特別に粗品 (Intel 詰め合わせ) を追加！(先着 0 ~ 1 名、気まぐれご奉仕)

- Intel 製 LGA775 マザーボード (詳細不明、シリアルポート、パラレルポート、光オーディオ出力あるよ！)
- あり合わせジャンクメモリ (DDR2) 512MB ~ 1GBx1 ~ 2 度、気まぐれで増えたり減ったり
- 廃棄予定だった Core 2 Duo E6550(クーラつき)



# 車好きの18きっぷ

文 編集部 かづきお

車が好きだ。

車と言ったら、自動車のことになると決まっているだろう。

電車とか、汽車なんてのは車ではない。

そもそも自分で運転できないわルートが決まっているわな乗り物に乗って何がうれしいのか。

そんな自分が、青春18きっぷとか言う18歳しか使えない切符<sup>\*1</sup>で帰省することになった。

中部のG県まで片道9時間の旅である。

何が悲しくて9時間もタダ座っていないといけないのか。

車で帰ったとしても5時間で到着するというのに、まったくテツドウ<sup>\*2</sup>というのは時間がかかる。

その上、ノリカエとかいう儀式のためにタイムスケジュールが決まっていて、目に付いたところで車を止めて買ひ食いしたり、疲れたから昼寝とか、そういういたた楽しみが一切無い。

挙げ句の果てに、朝4時半に新宿発とか、頭湧いてんのか。

寒い。寒い寒い寒い。

だいたい、何なんだこのシートは。まるで切り立った崖に座ってるような物だ。

リクライニング出来ないなら出来ないで、レカロとかスバルコにシートを作らせればもう少しましになるに違いない。

往きはまだ良い。暖かい太平洋側を進んでいくからな。

富士山も綺麗に見えたし、太平洋だって拝めた。

問題は帰り。エッチュウとかシナノとかいう恐がい國を通る<sup>おぞ</sup>という。大雪が降りすぎて電車<sup>\*3</sup>とか埋まってくれないかな。

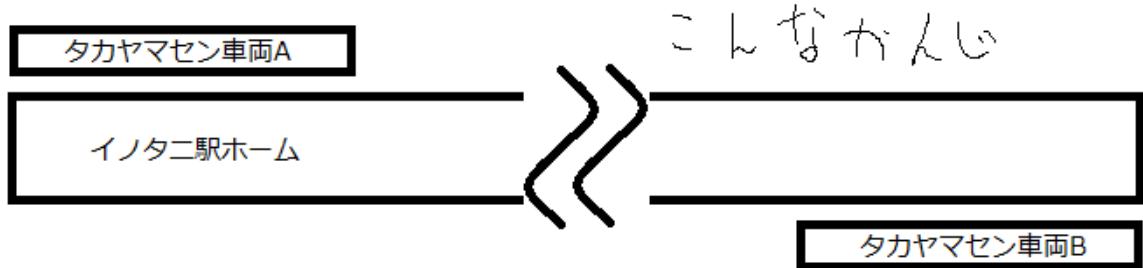
---

\*1 18歳しか使えない切符：そんなことはない

\*2 テツドウ：電車、電車と言っていたら、これは汽車だと怒られた。そうだよな、駆動方式は重要だ。たとえディーゼルであったとしても。

\*3 電車：このあたりは未開の地なので、電化されておらず汽車が正しい

まずタカヤマセンとか言うのを使って富山まで行く。途中イノタニとかいう携帯の電波も入らないような秘境で放り出される。乗り換えはホームの反対側とアナウンスされたが、反対側と言っても車両が止まってるの遙か向こうじゃないか。



<----- 1光年くらい ----->  
なるほど、このホームの真ん中あたりにJR東海と西日本の境界線があるのだろう。

タカヤマセンは、電化されてないのでディーゼルエンジンで動く汽車である。

どうやらこの車両はキハ40系とかいう車両で、しかもエンジンスワップ！されているという。胸が熱くなるな！

標準のエンジンは、ターボチャージド直6エンジン。15000CC！で300PSである。それを、キハ85系で用いられている米カミンズのターボチャージド直6エンジン、14000CC、350PSに積み替えて使用しているらしい。排気量小さいのにパワーはあるんだな。

# POOWWWEEEERRRR!!!!!!

富山に着いてからは、オオイトセンというシナノの國を通る鉄道を使って南下する。

雪がやばい。寒い寒い寒い。

途中雪に埋もれた車も何台かいた。



雪に埋まれ息絶えたパジェロミニ。

さすがに4WDでもこうなってしまったら  
どうしようもない。

空いていた車内だが、白馬からおびただしい数の人間が乗ってくる。スキーヤーとか言う人種だ。

まあ、なんとか D の高橋兄も冬を制したものは夏を制すと言っていたし、みんな無心になって冬の山を攻めているんだろう。

松本を過ぎた頃には物珍しさもなくなり、読書に没頭し、気づいたら土浦に居た。

まったく、美濃太田を出発してから、30 時間が経過している。車を使えば 5 時間だというのに。

まあでも、じっくり外を眺めることも出来たし、昼寝しても読書しても目的地まで連れて行ってくれる。

何より安い！ここ重要。

たまにはこんな旅行も悪くないなと思った。

ちょっとだけだよ。ホントだよ。

# WORD 民のための LATEX 2ε 入門

文 編集部 吉村 優

## 始めに

最近 WORD では LATEX を採用することに成功し、一太郎<sup>\*1</sup>くんとの両方を使って記事を作っています。

ただこの二つのソフトウェアは恐らく大抵の人が御存じの通り、全く別の用途に用いられるソフトウェアです。かたや論文などのドキュメントの作成・組版、こなたワープロや EPUB 文章などの編集ツール。また、LATEX がテキストファイルをソースとして PDF を生み出すのに対して、一太郎は独自アプリ「TARO22.EXE<sup>\*2</sup>」以外による編集を許さぬ独自のバイナリファイルをソースとして、PDF などを生み出します。

これらの違いから、「LATEX マンセー派」と「一太郎マンセー派」が WORD 編集部内に生まれることになりました。

僕は普段から LATEX で小説を書く程度には LATEX を使っている LATEX マンセー派なので、今回は編集部内に巣喰う一太郎マンセー派を切り崩すために、この記事を書きました。

## 第 1 章

\section や \subsection を使って記事を書き進めるのは、情報科学類の生徒であればリテラシーの授業か何かでやったことがあると思うので、その辺の基本的なことは飛ばします。

とりあえず最初の章では、みんなが LATEX で苦戦する様々な事柄を、便利なパッケージで解決しようかと思います。

### 1.1 フロートオブジェクトの配置

LATEX では、画像や表などをフロートオブジェクトと呼んでいます。この配置に苦しむ方が多いのではないかと思います。大抵は LATEX が空気を読もうとした結果、凄まじい位置に物体が配置されてしまうというものです。

例えば次のように、\includegraphics を使って画像を読み込んだとします。

```
1 \begin{figure}[htbp]
2   \centering
3   \includegraphics[width=70mm]{img/img.eps}
4 \end{figure}
```

するとこのように、あらぬところ<sup>\*3</sup>へ画像が吹っ飛んでしまうことがあります。これは TeX 処理系がスペースを吟味し、どうやらここには挿じ込めないだろうと踏んで、すっ飛ばしたのでしょう。しかし一太郎などその手のソフトウェアは WYSIWYG<sup>\*4</sup> という考え方から、ソフトウェアが勝手にオブジェクトをどこかへワープさせるなど言語道断なのです。

従って、ソースコードで出現する位置と実際に表示される位置が、だいたい同じになるように調整する必要があります。

---

\*1 JustSystems が作ったワープロソフト。WORD では伝統的に使われている。

\*2 一太郎の実行ファイルです。

\*3 今回の場合は次ページへワープしています。

\*4 ディスプレイに現れるものと処理内容（特に印刷結果）が一致するように表現する技術のこと。

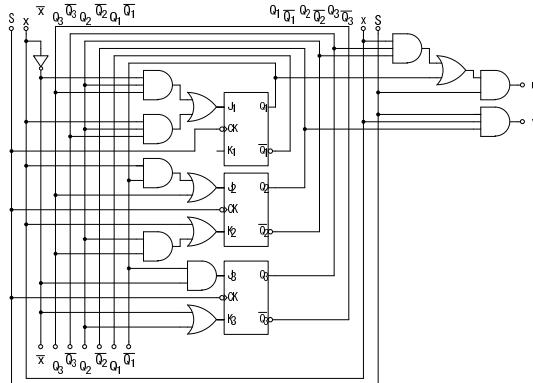


図1 吹っ飛んだ画像

このような時に用いるのが *here* パッケージ<sup>\*5</sup>です。

`\usepackage`

LaTeX でパッケージを用いるには、C 言語の `#include` と同じような `\usepackage` コマンドを用いて、パッケージを読み込む必要があります。例えばこの *here* パッケージを読み込む場合は、`\documentclass` コマンドの直後に、

```
1 \usepackage{here}
```

として、*here* パッケージを読み込みます。

*here* パッケージを用いた場合は、次のように `\figure` の位置指定に、`H` を用います。

```
1 \begin{figure}[H]
2   \centering
3   \includegraphics[width=70mm]{img/img.eps}
4 \end{figure}
```

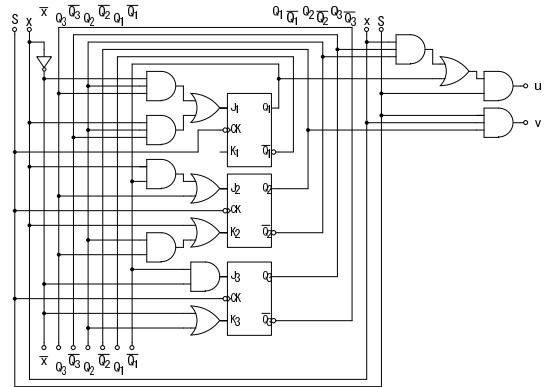


図2 吹っ飛んでない画像

これで大体大丈夫な感じです。

画像などが表示されない場合

`\documentclass` の引数に `draft` を指定している場合、画像は表示されずに、その画像と同じ大きさの枠だけになります。

また、`listings` などの位置を調整する場合は、脚注 5 の `float` パッケージを用いるのが良いです。詳しくは <http://mirror.hmc.edu/ctan/macros/>

\*5 このパッケージは中身で `float` パッケージを呼び出しているだけなので、実質は `float` パッケージを使っていることになります。このパッケージは、あらゆるものをフロートオブジェクトへジョブチェンジさせ、位置を強制するようなマクロなどがありとても有用です。

`latex/contrib/float/float.pdf`などを参照してください。

## 1.2 段組

段組とは、ページを何列かに分けて文章を記述する方法です。論文などに多く見られるレイアウトで、この記事も二段組になっています。

最も一般的な手法は、`\documentclass` に `twocolumn` オプションを使うことですが、これはちょっと敷居が高い<sup>\*6</sup> のでお勧めしません。

かわりに `multicol` パッケージ<sup>\*7</sup>を使うのがお勧めです。これは、`\begin{multicols}{N}` と `\end{multicols}` で囲った箇所を、 $N$  段組にするというパッケージで、これを使っておくのがかなり便利です。この記事の最初のように、ある場所までは一段組で、ある場所から二段組にするというような処理も、このパッケージを使えば容易に出来ます。

また `multicols` 環境の中で `multicols` 環境を用いることも出来ます。

```
1 \begin{multicols}{2}
2 Who killed Cock Robin?
3 I, said the Sparrow,
4 with my bow and arrow,
5 I killed Cock Robin.
6 \end{multicols}
```

|   |  |
|---|--|
| Who killed Cock<br>Robin? I, said the Spar- | row, with my bow and<br>arrow, I killed Cock<br>Robin. |
|---|--|

このように、なかなか汎用性の高いパッケージなので、こちらを使う方が楽かと思います。

### multicols 環境内のフロー

`multicols` 環境の中では、基本的にフローは置けません。ただ、先程紹介した `here` パッケージを使えば、問題なく表示することが出来ます。もし `figure` 環境を使いたいのであれば、かわりに `figure*` 環境を用いるか、`float` パッケージの `\newfloat` を使うのが良いです。

## 1.3 ソースコード

情報系の学生はソースコードを文章中に書きたくなる時があるかと思います。このような時、`listings` パッケージを使えばよいというのは、たぶんほとんどの人が知っているかと思います。しかし、`listings` パッケージは初期設定がゴミすぎて、なんの設定もせずに使うと正直失望しかありません。

### 1.3.1 設定方法

`listings` の設定は、次の二つの方法で変更出来ます。

`\lstset`

全てのスタイルに対して適用されるルールを設定します。

`\lstdefinestyle`

これは `listings` のスタイルを定義する命令です。`listings` は `\begin{lstlisting}[style=hoge]` で、`hoge` スタイルを適用出来ます。

ちなみに、`\lstdefinestyle` は次のように使えます。

```
1 \lstdefinestyle{Perl}{
2   language=Perl,
3   morekeywords={say}
4 }
5
6 \lstdefinestyle{HTML}{
7   language=HTML,
8   numbers=left
9 }
```

<sup>\*6</sup> 脚注も二段組になったり、二段以上の段組に出来ないなど不自由が多いです。

<sup>\*7</sup> このパッケージのドキュメントは三段組を使って書かれていて、非常に読みにくいので、正直二段組以上の段組はお勧めしません。<http://www.ctan.org/pkg/multicol>

これらは、

```
1 \begin{lstlisting}[style=Perl]
2 ... Perl Code ...
```

などとすると、設定を使うことが出来ます。

### 1.3.2 設定項目

ここでは *listings* の基本的な設定項目について述べてゆきます。

#### language

*listings* の中に書いてあるものが、何言語なのかを記述します<sup>\*8</sup>。

#### basicstyle

*listings* 内の普通の文字のスタイルを決めます。

#### keywordstyle

キーワードの文字スタイルを決めます。

#### breaklines

`breaklines=true` とすると、自動改行が有効になります。見た目上の行と実際の行は区別されまます。自動改行がないと *listings* の枠からはみ出る可能性があるので、有効にする方がいいでしょう。

#### numbers

行番号の位置に関する設定です。値は次のものを取ります。

- none
- left
- right

意味は察してください。

#### firstnumber

行番号の初期値です。これに数値を設定するとその番号から始まるのですが、`auto` または何も指定しないと、前回の番号の次の値から始まります。

#### frame

枠に関する設定です。引数はいろいろあるのですが、とりあえず `tbrl` を入れておけば、四角で囲われます。

#### tabsize

タブの幅がスペース何個分かを、数値で与えます。

#### xleftmargin, xrightmargin

左右のマージンを決めます。

この他にも様々なオプションがありますが、多過ぎるので詳しくはドキュメント<sup>\*9</sup>を見てください。

### 1.3.3 このドキュメントの設定

僕が使っている設定はこんな感じです。

```
1 \lstset{
2   language=[LaTeX]TeX,
3   basicstyle=\footnotesize\tt,
4   keywordstyle=\footnotesize\bf,
5   moredelim=[is][\it]{}/},
6   breaklines=true,
7   numbers=left,
8   showstringspaces=false,
9   firstnumber=1,
10  numbers=left,
11  xleftmargin=12pt,
12  xrightmargin=1pt,
13  tabsize=2,
14  backgroundcolor={\color[gray]{.9}},
15  basewidth={0.57em, 0.52em}
16 }
```

## 1.4 ページ余白

ページの余白などは本来、`jsarticle` などといったクラスファイルの中で文字サイズなどから計算して決定されるものです。なので、それを変更するのは本来厄介なのですが、L<sup>A</sup>T<sub>E</sub>X をよく知らない人にとって、クラスファイルの改造は敷居が高すぎるかと思います。クラスファイルの改造などをせずカジュアルに余白を変更したい方は、`geometry` パッケージを使って余白を設定するのが良いかと思います。

<sup>\*8</sup> 記述の仕方が独特なので注意が必要です。詳しくは後述する *listings* のドキュメントを参照してください。

<sup>\*9</sup> <http://www.ctan.org/pkg/listings>

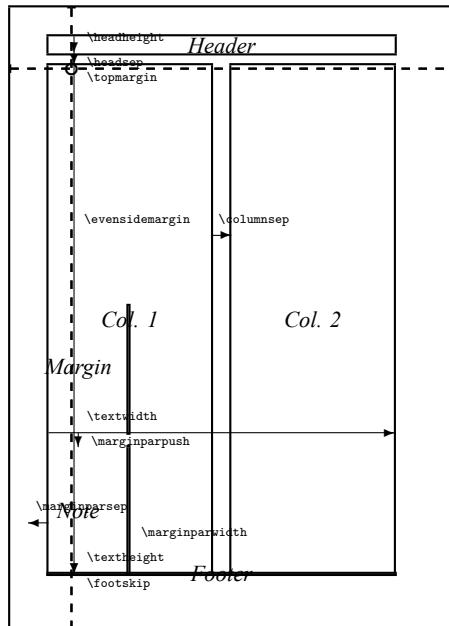
### 1.4.1 ページレイアウトの取得

とりあえず、現在のレイアウトがどのようにになっているのかを把握します。これには *layouts* パッケージを使うのが便利です。

次のようにすると、現在のページレイアウトが表示さ

れます。

```
1 \currentpage
2 \twocolumnlayouttrue*10
3
4 \pagedesign
```



```
\paperheight = 731.23582pt
\paperwidth = 517.84015pt
\hoffset = 0.0pt
\voffset = 0.0pt
\evensidemargin = -26.74559pt
\oddsidemargin = -26.74559pt
\topmargin = -38.1267pt
\headheight = 20.0pt
\headsep = 14.22636pt
\textheight = 597.50789pt
\textwidth = 406.79134pt
\footskip = 0.0pt
\marginparsep = 23.58018pt
\marginparpush = 17.0pt
\columnsep = 23.58018pt
\columnseprule = 0.0pt
\lrm = 8.5pt
\lex = 3.79543pt
```

\* この図は *Footer*などのスペースが全くありませんが、これは現在のページのレイアウトを反映しているからです。

*layouts* このように、分かりやすく各部分の大きさを取得できます。

パッケージはこの他にも、脚注やフロートの情報などを視覚的に表示出来るので、組版を調整するにしても有用です。詳しくはドキュメント<sup>\*11\*12</sup>を見てください。

### 1.4.2 ページ全体の余白設定

これは *\usepackage* のオプションを次のようにするだけです。

```
1 \usepackage[top=1mm, left=1mm, right=1mm,
bottom=1mm]{geometry}
```

<sup>\*10</sup> *\twocolumnlayouttrue* とは、ページが二段組であるかどうかの設定です。二段組ではない場合は特に必要ありません。

<sup>\*11</sup> <http://ftp.yz.yamagata-u.ac.jp/pub/CTAN/macros/latex/contrib/layouts/layman.pdf>

<sup>\*12</sup> このドキュメントでは *\setparameters{textsize}* というマクロがあることになっていますが、実際には存在せず、ソースを読むと *\setparameter{textfont}* が存在するので、こちらの間違いではないかと思われます。

これは、上下左右のマージンが全て 1mm であるということになります。

### 1.4.3 あるページの余白設定

あるページだけ余白設定を変えたいという事例があるかもしれません。`geometry` には `\newgeometry` コマンドなど、そういう時に便利なコマンドがいくつか用意されています。

次のようにすることで、そのページから以降全てのページについて、余白を変更することが出来ます。

```
1 \newgeometry{left=3cm,right=1cm,bottom=0.1cm}
```

また、`\newgeometry` を使う以前の余白設定に戻す場合は、`\restoregeometry` を使います。

```
1 \restoregeometry
```

このようにすることで、臨機応変にページの余白を変更することが出来ます。

## 1.5 他のパッケージ

最後にこの文章を作るために用いたパッケージを列举しておきます。ただし、今までに紹介したものは除きます。

### 1.5.1 CTAN<sup>\*13</sup>収録

#### *txfonts*

Times New Roman っぽいフォントを使うパッケージです。

#### *type1cm*

Computer Modern Type1 Fonts に多様な大きさを提供するパッケージです。

#### *enumitem*

`description` 環境において、見出しの後で改行し、本文をインデントしてから始めるレイアウトにするパッケージです<sup>\*14\*15</sup>。

#### *url*

URL を適切に表示するパッケージです。スペースを URL に入れたい場合 `obeyspace` オプションを使うとよいです。

#### *graphicx*

画像を入れるパッケージです。

#### *color*

色を使うためのパッケージです。WORD コマンドや `listings` の背景色のために使っています。

#### *otf*

ヒラギノのボールドなど、多彩な日本語フォントを使うためのパッケージです。

#### *dirtree*

ディレクトリ構造を綺麗に書くためのパッケージです。2.3節で扱っています。

### 1.5.2 CTAN 未収録

#### *ascmac*

`itembox` 環境など、枠付きの箱を作るパッケージです。

### 1.5.3 この記事における使い方

この記事ではどのように `\usepackage` しているのかを示します。

```
1 \usepackage{txfonts}
2 \usepackage{ascmac}
3 \usepackage{enumitem}
4 \usepackage{layouts}
5 \usepackage{dirtree}
6 \usepackage[final]{listings}
7 \usepackage{jlisting}
8 \usepackage{float}
9 \usepackage{multicol}
10 \usepackage[obeyspaces]{url}
11 \usepackage[dvipdfmx]{graphicx}
12 \usepackage[dvipdfmx]{color}
13 \usepackage[multi, deluxe, bold]{otf}
```

\*<sup>13</sup> TeX のパッケージをまとめてある物体。

\*<sup>14</sup> `enumitem` パッケージを読み込み、`style=newline` を指定すれば良い。

\*<sup>15</sup> これを使っても見出しと本文が同じ行に出力されることがあります。このような場合は、`\item[hoge]\hfill\`{ }` とするのが良いです。

## 第 2 章

一太郎はドキュメントの完成予定図を常に見ながら作業出来ます。しかし L<sup>A</sup>T<sub>E</sub>X は一般に、ソースと出来あがるもの<sup>\*16</sup>が一致していないので、コンパイルするまではどのような見た目になるのか分からぬという問題があります。

また、L<sup>A</sup>T<sub>E</sub>X<sup>\*17</sup>は目次や索引の関係で、何回かコンパイルしなければならないという問題もあります。

これらを解決する手法として最も一般的であるのが、Makefile を書くことです。ただ最近はただの Make ではなくて、ちょっと高機能な Make である OMake というものがあります。

この章では OMake の使い方、そしてそれを用いた T<sub>E</sub>X ファイルのコンパイルについて解説します。

ただ、OMake は比較的強力なデータ構造<sup>\*18</sup>や制御文、さらにはファイルの入出力も出来る高機能な物体なので、この文章だけで全てを説明するのは無理があります。そこで、ひとまずはこの記事をコンパイルする時に使っている OMake を紹介します。

### 2.1 OMake の準備

OMake はパッケージ管理ソフトなどを使ってインストール出来るかと思います。適当にインストールしてください。

今回の環境ですが、カレントディレクトリに *main.tex* とし、それは *word.cls* という専用のクラスファイルに依存しています。この *main.tex* をコンパイルすることを考えます。通常はターミナルで次のようなコマンドを叩きます。

```
1 $ pdflatex --kanji=utf8*19 main.tex
2 $ dvipdfmx -p jisb5*20 main.dvi
```

これらを OMake くんにやっていただくわけです。

### 2.2 コンパイルしてみる

とりあえず細かい説明はおいておき、どのような物を用意すればコンパイルが出来るのかを見てみます。

#### 2.2.1 初期化

まずは OMake に必要なファイルを作成します。OMake に必要なファイルは基本的に二つで、*OMakeroot*

と *OMakefile* です。これら二つを生成します。

```
1 $ touch OMakeroot
2 $ touch OMakefile
```

この二つのファイルの意味は次の通りです。

*OMakeroot*

すべての *OMakefile* の中から参照出来る関数や変数を定義します。

*OMakefile*

基本的には、*OMakefile* の置かれているディレクトリのファイルをコンパイルするルールを書きます。

\*<sup>16</sup> 例えば PDF ファイルや POSTSCRIPT を出力します。

\*<sup>17</sup> ここでは DVI ファイルを経由する処理系を指します。PDF ファイルを直接吐くような処理系 (pdfTeX など) は無視します

\*<sup>18</sup> 配列やクラス、オブジェクトなどを備えています。

\*<sup>20</sup> これは入力ファイルの文字コードが UTF-8 であることを主張しています。別の文字コードである場合は適宜変更する必要があります。

\*<sup>20</sup> 紙のサイズが JIS B5 であることを主張する引数です。

## 2.2.2 OMakefile

ここではとりあえず、*OMakeroot* はおいておき、*OMakefile* を次のように編集してみます。

```

1 LATEX = platex --interaction=nonstopmode --
2   kanji=utf8
3 DVIPDFM = dvipdfmx
4 DVIPDFMFLAGS = -p jisb5
5 TEXDEPS = word.cls
6
7 LaTeXDocument(main, main)
8
9 .DEFAULT : main.pdf

```

このような内容で保存します。それでは、各行の説明をします。

### 1, 2 行目

コンパイルに用いる処理系を指定しています。この設定では L<sup>A</sup>T<sub>E</sub>X に *platex* を、*dviware* に *dvipdfmx* を使ってています。

### 4 行目

*dvipdfmx* に渡すオプションを指定しています。ここでは、紙のサイズを JIS B5 にしています。

### 5 行目

コンパイル対象が依存しているファイルを指定します。

### 7 行目

コンパイルを実行します。

### 9 行目

デフォルトでコンパイルするものを指定します。

### 依存ファイル -TEXDEPS-

5 行目ですが、依存ファイルを指定しています。OMake は -P オプションでファイルの変更を監視出来るので、TEXDEPS に T<sub>E</sub>X のクラスファイルなど依存しているファイルを挿入することで、監視対象のファイルを追加出来ます。よって、依存ファイルを指定します。

*omake* コマンドを使えばコンパイルされ、*main.pdf* が得られます。

## 2.3 分割されたファイルのコンパイル

T<sub>E</sub>X の特徴は、\inputなどを用いて分割されたファイルを読み込むことにあります。

僕が L<sup>A</sup>T<sub>E</sub>X を使って小説を書くと、時々百ページくらいになってしまうことがあります。そういう時に一つのファイルに死ぬ程文字が書いてあると、Vim が重くなってしまってストレスが溜まり執筆不能に陥るでしょう。なのでファイルを分割しようという話なのですが、Omake を使っている状況でファイル分割を行うと、分割ファイルが増える度に、

- 分割された T<sub>E</sub>X ファイルをまとめるファイル<sup>\*21</sup>
- *Omakefile*

の二つを編集する必要があります。

こんなことをファイルが増える度にしていては、精神が破滅しかねませんし、ミスを犯す可能性が増加することになります。そこで Omake を使って分割された T<sub>E</sub>X ファイルのコンパイルに挑みます。よって、次のようなものを Omake で作ります。

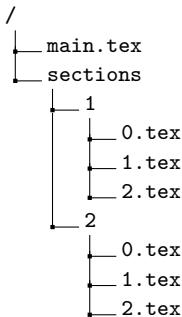
1. 依存している分割 T<sub>E</sub>X ファイルを取得。
2. フォルダの階層構造から、\section や \subsectionなどを判断し、T<sub>E</sub>X ファイルを結合する部分の作成。

<sup>\*21</sup> これは \input が列挙された T<sub>E</sub>X ファイルということになります。

### 3. コンパイル

という動作を行う必要があります。

また、ここでは依存のある分割ファイルが次のようなディレクトリ構造を持つとします。



この、*sections/1/0.tex* や *sections/1/1.tex* などに、*main.tex* が依存しているとします。また *0.tex* は一段組、それ以外の部分は二段組とします。

よって、依存する TeX ファイルを結合している TeX ファイルは次のようなものになるとします。

```

1 \section{ }
2 \input{sections/1/0.tex}
3 \begin{multicols}{2}
4 \input{sections/1/1.tex}
5 \input{sections/1/2.tex}
6 \end{multicols}
7 \section{ }
8 \input{sections/2/0.tex}
9 \begin{multicols}{2}
10 \input{sections/2/1.tex}
11 \input{sections/2/2.tex}
12 \end{multicols}

```

このファイルを *main.tex* から *\input* で読み込みます。

#### 2.3.1 依存ファイルの取得

OMake は関数を書けるので、ここでは関数を書いてみることにします。書く関数は、

1. ディレクトリを渡すと、その直下のディレクトリを全て返す関数
2. ディレクトリの配列を受け取り、そのディレクトリに含まれる全ての TeX ファイルを返す関数。

の二つです。

#### 直下のディレクトリを取得する関数

まずは一つ目ですが、これは OMake がサポートしている *\$(ls)* を使ってこのようにします。

```

1 FindDirs(dir) =
2     return $(set $(dirname $(ls $(dir)*)))

```

用いられている関数は以下のようなものです。

*\$(set)*

これは文字列の集合をソートして、重複を取り除きます。

*\$(dirname)*

ファイルからディレクトリ名を抜き出します。

*\$(ls)*

シェルコマンドの *ls* と同じです。

#### 直下の TeX ファイルを返す関数

これも、*\$(ls)* を使えば簡単です。

```

1 DepsUpdate(dirs) =
2     deps =
3         foreach (d, $(dirs))
4             value $(filter %.tex, $(set $(ls $(d)))))
5
6     return $(deps)

```

用いられたものは、

*foreach*

渡された配列を先頭から一つずつ変数に代入してループします。

*value*

*foreach* 文の返り値を指定します。

*\$(filter a b)*

*b* の内、*a* に該当する物のみを拾い集めます。*%* はワイルドカードなので、この場合は最後が *.tex* であるファイルを集めます。

というような意味です。

#### 2.3.2 分割 TeX の結合ファイルの作成

これは先程作成した関数などを使えば簡単に取得出来ます。

```

1 section
2   fp = $(fopen $@, w)
3   foreach(dir, $(dirs))
4     fprintln($(fp), \section{})
5     foreach(zero, $(filter %0.tex, $(set $(ls $(dir)))))
6       fprintln($(fp), \input{$(zero)})
7     texts = $(filter-out %0.tex, $(filter %.tex, $(set $(ls $(dir)))))
8     if $(gt $(length $(texts)), 0)
9       fprintln($(fp), \begin{multicols}{2})
10      foreach(file, $(texts))
11        fprintln($(fp), \input{$(file)})
12      if $(gt $(length $(texts)), 0)
13        fprintln($(fp), \end{multicols})

```

ファイルを操作するための新しいコマンドが出現しました。

`$(fopen)`  
ファイルを開きます。`w` は書き込みモードで開くことを意味します。

`$@`

コンパイルした成果物のファイルを指します。

`$(fprintln)`

ファイルに出力します。

`if`

条件分岐です。この条件分岐は、フォルダの中に `.tex` しかなかった場合に、`\begin{multicols}` と `\end{multicols}` を出力しないようにするものです。

### 2.3.3 完成品

最終的に次のようになりました。

## OMakeroot

```

1 open build/LaTeX
2
3 DefineCommandVars()
4
5 FindDirs(dir) =
6   return $(set $(dirname $(ls $(dir)*)))
7
8 DepsUpdate(dirs) =
9   deps =
10  foreach (d, $(dirs))
11    value $(filter %.tex, $(set $(ls $(d)))))
12
13  return $(deps)
14
15 .SUBDIRS: .

```

## OMakefile

```

1 .PHONY: clean
2
3 LATEX = platex --interaction=nonstopmode --kanji=utf8
4 DVIPDFM = dvipdfmx
5 DVIPDFMFLAGS = -d 5 -p jisb5 -f cid-x-local.map
6

```

```
7  dirs = $(FindDirs ./sections/)  
8  TEXDEPS = $(DepsUpdate $(dirs)) word.cls word.clo main.tex default-style.sty  
9  
10 word.cls : ../../texfiles/word.cls  
11   cp $< $@  
12  
13 word.clo : ../../texfiles/word.clo  
14   cp $< $@  
15  
16 body.tex : :value: $(DepsUpdate $(dirs))  
17   section  
18     fp = $(fopen $@, w)  
19     foreach(dir, $(dirs))  
20       fprintfn($fp), \section{}  
21       foreach(zero, $(filter %0.tex, $(set $(ls $(dir)))))  
22         fprintfn($fp), \input{$(zero)}  
23       texs = $(filter-out %0.tex, $(filter %.tex, $(set $(ls $(dir)))))  
24       if $(gt $(length $(texs)), 0)  
25         fprintfn($fp), \begin{multicols}{2}  
26       foreach(file, $(texs))  
27         fprintfn($fp), \input{$(file)}  
28       if $(gt $(length $(texs)), 0)  
29         fprintfn($fp), \end{multicols}  
30  
31  
32 LaTeXDocument(main, main)  
33  
34 .DEFAULT: main.pdf
```

word.cls, word.clo

WORD では記事と専用のクラスファイルを別に管理しているため、このように別のディレクトリからコピーしています。

今回は詳しく説明していないので、OMake の詳しい

ことは全然分からなかったかもしれません、とりあえず、分割された TeX ファイルであっても、OMake で自動コンパイル出来るということです。

OMake について詳しく知りたい方は、<http://omake-japanese.sourceforge.jp/> を参照するといいと思います。

## 終わりに

LATEX は非常に自由度の高いソフトウェアなので、最初はちょっと難しいかもしれません、使えるようになれば簡単に美しい文書を書けるようになると思います。

また LATEX はそれ単体ではちょっと苛つくことがあります。しかし、そういう時は CTAN などのパッケージで解決出来ることが多いので、とりあえずは CTAN に収録されたパッケージを調べてみましょう。

そしてコンパイルなどは、テキストエディタで TeX を書いて、その都度ターミナルからコマンドを叩いているので手間がかかるので、OMake など何かしらを使うことをお勧めします。

次号は最初からクラスファイルを作る話でも書こうかと思います。

# とよさと!

文 編集部 こぼる



「そうだ京都、行こう。」この有名な文言を借りたとしても、私が9月末に敢行した小旅行を表現することは出来る。現に私は京都へ行き、京都にある建造物を拝見し、京都の空気を堪能したからだ。しかしそれで表されるのは旅の本意そのものではない。正しくはこう書くべきである。「そうだ豊郷、行こう。」

滋賀県の方々には申し訳なく、また自分の浅学さが恥ずかしい限りだが、滋賀県の名所と言われて自分は二ヶ所しか思い浮かべることができない。それは琵琶湖と豊郷である。琵琶湖はともかくとして、豊郷と言われてもぴんとくる人間は少ないであろう。滋賀県犬上郡豊郷町。そこにあるのは「豊郷町立豊郷小学校」という名の小学校、及びその旧校舎である。今回私が豊郷を訪れたのはその旧校舎に目的がある。2004年に学び舎としての務めを終えて以来この校舎は一般に開放されており、映画「逆転裁判」や、最近のものでは2013年公開予定の映画「だいじょうぶ3組」などのロケ地として使われたりしている。だが、恐らくここに今日最も多くの人間を引き寄せているのは、大人気を博した漫画およびアニメ「けいおん!」の学校のモデルになったという事実であろう。つまり、いわゆる「聖地」なのである。今回私がここを訪れたのも他でもない、聖地巡礼が目的である。

かねてから念願であったこの聖地巡礼を行うため、私は5日間に及ぶ旅行を計画した。残念ながらこのとき我々の大学では学期の真っ只中であった為、真面目な一学徒として学業との兼ね合いは考慮しなければならなかつた。熟慮の結果、平常通り講義のある木曜日の夜に出発し、月曜日の朝に帰宅し2限からの講義に出席するというエクストリームスケジュールが爆誕した。「あれ、金曜日には講義があるはずでは……」などという疑問を抱いてはならない。私が同日に履修していた講義は何故か既に大方雲散霧消していた為、何の問題もなく金曜日から始まる3連休を練成することが可能だったのである。

私が豊郷を訪れたのは旅行行程の4日目、日曜日のことであった。金曜日に大阪、土曜日に京都を満喫した私は、この日滋賀県豊郷に向かっていた。旅の道連れには大阪、京都それぞれの大学に通う友人を召喚した。余談になるが、けいおん!の聖地とされている学校は滋賀県豊郷にあるものの、学校以外の聖地とされる場所の多くは京都の某大学近辺にある。当然私はいわゆる京都らしい(寺院巡りなどはそっちのけで聖地巡礼に最大限の時間を割いた。話を戻して)日曜日、京都の友人宅から豊郷に向けて出発する。間が悪くこの日は台風が接近しており、朝から小雨が降っていた。京都から近江八幡、近江八幡から八日市、八日市から豊郷と電車を乗り継いでいる内、

雨は大降りに変わった為、私は途中八日市で傘を買わなければならなかつた。奮発して買った1000円もするこの傘が4時間と経たぬ内に台風によって無残に破壊されてしまうことを、この時まだ知らない。

近江八幡より先は近江鉄道に乗って豊郷へ向かう。二両編成の電車に乗っていくつもの無人駅を通り過ぎる間は、雨で濡れた窓の外に延々と田んぼや山々が広がるといった風情であった。

八日市からは20分程で豊郷に着いた。駅は二つのプラットフォームが向かい合うこれといって変わつた所のない簡素なホームからなつていた。電車を降りた後に待ち受ける何の変哲もない看板が、目的地への到着を告げていた。



飛び出し君けいおん!キャラバージョン。最初に出迎えてくれたのはこのムギちゃんであつた

駅を出て、事前に調べた通りの道筋を辿るべく歩み出す。だが、事前調査の必要は無かつたことをすぐに思い知る。経路のそこかしこに「ようこそ聖地へ」という文字と共に、けいおん!のキャラクターが描かれた幟旗が設置されていたのだ。最下部には「豊郷町観光協会」の文字。けいおん!が町から公式に認知されていることに驚いた。それだけではない。見れば、周りの店の窓や掲示板にもけいおん!関連のポスターが数多く貼られている。かの有名な飛び出し君のけいおん!キャラバージョンもあちこちに散見された。なるほど、これらけいおん!記号を辿つていいくだけ学校には着けそうである。駅から10分程歩いて学校には到着した。学校は校舎、講堂、図書館の3つの建物からなる。最初に目に入ったのは、講堂の裏だった。

ようやく念願の聖地である。感動と喜びでしばし、校門前で雨の中友人と狂喜乱舞する。先日の京都での聖地巡礼から幾度か味わっている感覚だが、学校の外観は正に作中に描かれているものそのもので、作品の世界に入り込んでいるかのような錯覚を覚える。

校舎、講堂、図書館と選択肢は三つあるがまずは正面の校舎に入る。スリッパを履いて上ると、見覚えのある階段と、手すりの「兎と亀」の寓話をモチーフにしたブロンズ像が目に入る。これが部室へつながる階段であることはすぐに分かった。3階までとりあえずは上る上る。途中手すり上のブロンズ像で「兎と亀」のエピソードが展開されていて面白い。2階-3階の踊り場に立つと著しく見覚えのあるアングルに。

……遂にここまで来てしまった。桜高軽音部の部室である。逸る気持ちを抑えることもなく入室すると……



著しく見覚えのあるアングル



うおー。感嘆せすにはいられない。部室はこれから何までそつくりである。黒板や長椅子、寄せられた机から隣の音楽室へと通じる奥の物置に至るまで、作中のものと殆ど同じだ。強いて言えば、部屋の構造が作中ではL字形であったのに対しモデルは四角形だが、これは作品上あまり大きな違いではない。

中央に据えられている、お馴染みの寄せられた4つの机に注目すると、上にはケーキやティーカップなどでお茶会の風景が再現されている。引き出しの中を見ると、作中に登場した絵やキャラクターの入部届けなどが入っている。こういったものや、隅の机に飾られている数々のグッズ等は皆有志の提供によって集まつたものだろう。黒板やホワイトボードは本来来場者による数々の書き込みで埋め尽くされている筈だが、今日はたまたま有志による年に一度の大掃除の日と重なったらしく、何も書かれていなかった。しばらく椅子に座ってみたり、窓からの景色を眺めたりして雰囲気を記憶に刻んでから教室を後にした。



ところで先程から楽器の演奏が聞こえている。部室隣の音楽室でバンドの練習を行っているらしい。察するに彼らもまたけいおん!のファンで、ここ音楽室は公共の練習場所となっているようだ。この部屋は2期第7話「お茶会!」にお茶会が行われた場所だ。しかし、実際にあれだけの人数を集めるには実物は些か小さ過ぎるようであった。練習の邪魔にならぬよう手短に見て回り、その場を後にした。



校舎は基本的に2階建てで、音楽室と部室に当たる音楽準備室だけが3階になっている。2階以下の教室は殆どが閉め切られていて入ることが出来なかつたが、廊下に面した窓から中を覗くことは出来た。机などは作中のものと類似性が無く、また教室自体も一回りも二回りも小さい印象を受けるが、覗き込んだ窓をはじめ全体的な雰囲気としてはよく似ていた。

粗方廻って、校舎内で見ることのできる場所は残すところ1階入り口正面、作中では職員室に当たる部屋のみとなった。ここは警備員の監視の下、豊郷小学校の歴史を展示する部屋として開放されていた。中では旧校舎の建築に関する情報や学校として使用されていた当時の写真、そして校舎全体の模型などを見ることができる。そういう展示物を見つづ、部屋の構造などを見ると確かに作中の職員室であることが確認できた。

これにて校舎の観覧は終了だ。しかし、見所はまだ講堂、図書館と残されている。校舎の次は講堂を見てみることにしよう。



幸いなことに校舎から講堂は屋根が続いていたため、更なる強まりをみせる雨風に当たること無く移動することが出来た。

講堂の中は、正面のステージから据え付けられた椅子に至るまで作中のものとそっくりである。ただし、壁に掛けられた校歌は豊郷小学校のものだ。作中では恐らく確認できないが、実はこの講堂には二階がある

ことを発見した。ステージの反対側、入り口を入ってすぐの場所に、階段を上っていくちょっとしたスペースがある。二階席というわけだ。ここもしばらくの間眺めてから退館した。

さて、巡礼も残すところ図書館のみとなった。校舎に入るときスリッパに履き替えた為、一旦校舎入り口まで戻り、図書館へと向かう。校舎から図書館へ行くには短い間だが外に出なければならなかつた。暴風雨の中、ほんの20秒程度ダッシュしただけで、傘による防護も虚しく衣服はじっとりと濡れてしまった。

この図書館では週末、カフェが営業されている。私が来訪を週末に据えた理由の一つがこれである。カフェではけいおん！に因んだ軽食等を販売している他、お土産も多数販売している。早速、雨でぐしょ濡れの靴を脱いで上がってみることにした。

本来は図書館であり、作中の図書館のモデルにもなっているものの、最早そこには本棚や蔵書は残されていない。代わりにカフェの為のテーブルがいくつか設えてあつた。奥には小さな空間があり、ここにもまた有志提供の楽器、フィギュア、雑誌、キャラクターパネルなどが飾られている。こうしたものをひとしきり眺めた後、カフェで軽食を注文しテーブルで一息つくことにした。



カフェで注文した「澤先輩焼きそばパン」と「あずにやんたいやぎ」



図書館の様子。二階より

しかしそれにしても、ここには意外な程来訪者が多い。周囲を見渡してみても、ざっと 10 人から 20 人はここ図書館にいるだろう。中にはどことなく常連とわかるような集団も居た。校舎や講堂を廻っている間も同様にそれなりの人間には遭遇していた。この頃は 9 月末である。一部の大学でこそ夏休みであるものの、日曜日とはいえ世間の大部分では長期休暇などではあるまいし、ましてや今日は台風が接近、どころか今や直撃している。それにも関わらずこれだけの人間が毎週末、あるいは毎日のようにテレビ放映から 2 年が経過した作品の為に来訪していることは、けいおん!の根強い人気を表しているようであった。

しばらく土産品を物色したりだらだらしたりしている内に、天候も手伝って空が暗くなってきた。名残惜しいが、そろそろ帰路につくことにしよう。私は友人と連れ立って暴風雨の校庭に歩み出した。吹き荒ぶ風に殴られて新品の傘がバキバキに破壊されたのは、校舎から豊郷駅へと歩く途中のことだ。その後は悪天候による電車の運行停止のため駅で震えながら待たされるなど、天候には終始散々な目に合わされた。しかし、友人の巧みな路線捌きのおかげで何とか無事帰還し、月曜 2 限の単位は安寧を得るのであった。

以上が私の豊郷紀行文である。私は京都、豊郷それぞれに一日ずつかけゆっくりと廻ることができたが、それでも巡礼できていない場所はまだまだある。いずれまた機会を見つけて、今度はより天候に恵まれた日に聖地巡礼を行ってみたいと思う。読者の中にももしけいおん!や聖地巡礼に興味のある方がいたら、是非一度豊郷を訪れてみていただきたい。とても良い体験となることを私は確信している。それでは以下に豊郷小学校に関する情報をまとめて本稿を終わりとする。

### 豊郷町立豊郷小学校

所在地: 529-1169

滋賀県犬上郡豊郷町大字石畠 518

京都駅からの行き方(自分とは異なる Google 先生により最適化されたルート):

- |                                 |              |
|---------------------------------|--------------|
| 1. JR 京都駅 JR 東海道・山陽本線 米原経由近江塩津行 | 48 分, 1110 円 |
| 2. 彦根駅 近江鉄道本線 貴生川行              | 18 分, 400 円  |
| 3. 豊郷駅                          | 10 分, 徒歩     |

### カフェについて

営業日: 土日祝日(イベント時等臨時休業あり)

更なる詳細情報については、「今日の部室」でぐぐると幸せになれそうです。

## 書籍紹介

文 編集部 ,無季、UTM

### 書籍紹介とは

この書籍紹介は、情報科学類生による情報科学類生のための書籍紹介です。技術的に役立つものから知つていて得をしないものまで、書籍をピックアップして紹介します。

#### プレゼンテーションzen

書籍名 : プrezentation zen

著者名 : ガー・レイノルズ

訳者名 : 熊谷小百合

発行 : ピアソン桐原

ISBN : 978-4-89471-328-4

値段 : 2300円

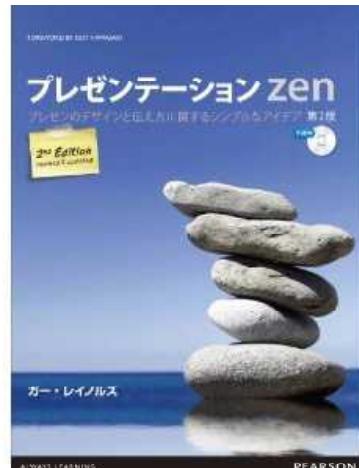
頁数 : 256頁

発行日 : 2009年9月10日

筆者は、現在多く見られる「お粗末な」プレゼンテーションスライドに異議を唱え、ビジュアルの威力を發揮するスライドの作り方を紹介している。プレゼンテーションを作るに当たって、どのようにストーリーを考え、画像や文章を用い、話をするのか。そういうことを、実際のスライド例を挙げながら、議論している。スライドの在り方を検討している筆者だけあり、本は極めて読みやすい。

私はこの本を読んで、プレゼンテーションの主役というものを再度考えることになった。プレゼンテーションスライドはあくまでも資料であり、主役は発表者なのだ。聴衆がスライドの文字を読むのに必死になった結果、発表者の主張を聞き逃すという事故を無くすために、発表者が話している内容をサポートするスライド作りが求められると思う。

この本では非常に鮮やかで、しかも綺麗なスライドが様々紹介されており、パラパラと眺めるだけでもとても有意義な時間を過ごすことが出来るだろう。



※画像は第2版

## 病苦を超える最後の天行力

書籍名：病苦を超える最後の天行力 初版第12刷

著者名：福永法源

発行：株式会社 アースエイド

ISBN：978-4-900331-21-1

値段：1000円

頁数：280頁

発行日：1995年1月30日

「天行力」とは、自然に則った生活を送ることができている者のみが用いることができる、生命の根源エネルギーのことである。この力により、筆者は人の過去を見たり、病を癒したりして人々を救ってきた。しかし、今日自然に則った生活を送っていない人類があまりにも多く、このままでは災厄を引き寄せ 2001 年には人類は滅亡してしまうと警鐘を鳴らす。本書では、天行力の能力を膨大な具体例を用いて紹介し、それをもたらしている天の意志と人類滅亡回避のためにどのように生き方を変えればいいかを解説している。2001 年の滅亡に備えるために、我々も筆者のように自然に逆らわない生き方を身につけなければならない。



掲載した画像は、以下のウェブサイトから引用しました。

- Amazon.co.jp (<http://www.amazon.co.jp/>)

# 母校が志望校になった日 リターンズ

文 編集部 UTM

## 1 前置き

### 1.1 「母校が志望校になった日」とは

社会学類誌「そおしあ～る（以下そしあ）」が毎年行っている、センタープレ模試の受験のことである。今回はWORDも参加し、そしあと点数を争った。前回のセンタープレ対決<sup>\*1</sup>では惜しくもそしあに負けてしまったWORDだが、今回の対決で雪辱を果たす。

### 1.2 ルール

- ・駿台センター試験プレテストを受験し、結果を競う。
- ・志望校には、筑波大学情報学群情報科学類前期、社会・国際学群社会学類前期を指定する。
- ・受験する科目は、社会1科目・国語・英語リスニング・英語・数学2科目・理科2科目である。理科と社会の科目は、受験する上での制限はないため各人が自由に選べる。
- ・勝負の判定は、最高点・最低点・平均点の3点で、2つ以上優れていたほうを勝ちとする。

### 1.3 参加者一覧

今回の参加者はWORD2人、そしあ6人である。WORDは2人と人数が非常に少ないとみ、どちらか1人が点を落とすと平均点に大きく影響する。また、そしあは受験を経験したばかりの1年生が3人も参加するため、手強い戦いになりそうである。

参加者のリストは表1の通りである。

表1:WORD 参加者リスト

| 名前  | 学年       | 入学方式 | 意気込み     | 備考     |
|-----|----------|------|----------|--------|
| UTM | 学類3年     | 後期   | 現役超え(ヤツ) | 前期は落ちた |
| ,無季 | 博士課程後期1年 | 推薦   | 同上       |        |

### 1.4 時間割

今回のセンタープレは表2の時間割で実施された。一日で全ての科目を消化するというヴェリーハードなスケジュールである。朝早く起きるのが嫌なので、情報科学類、社会学類ともにセンター試験では社会を2科目受けれる必要はないので、2時間目の地歴・公民から受験した。

---

\*1 14号「WORDはSIMフリーです号」参照。今回の記事のタイトルが「リターンズ」なのは前回の勝負を踏まえてのことである。この記事も前回の記事の形式を流用踏襲している。

表 2:試験の時間割

|       |             |
|-------|-------------|
| 地歴・公民 | 8:25~9:25   |
| 地歴・公民 | 9:35~10:35  |
| 国語    | 10:45~12:05 |
| リスニング | 12:15~13:00 |
| 昼食・休憩 |             |
| 受験登録  | 13:40~13:55 |
| 英語    | 14:00~15:20 |
| 数学①   | 15:30~16:30 |
| 数学②   | 16:40~17:40 |
| 理科①   | 17:50~18:50 |
| 理科②   | 19:00~20:00 |

## 2 当日の様子

人生を左右する戦いを控えた強者<sup>\*2</sup>に挑む我々は、御茶ノ水にある駿台お茶の水2号館で2012年12月16日に受験をした。当日はそしあの参加者と共にTXつくば駅にて集合し、TXと総武線を乗り継ぎ会場に到着した。

今回のセンター模試は高3生・高卒生が対象であり、かつ本番のセンター試験まであと1ヶ月ということもあって、会場の様子は非常に重苦しいものだった。会場内の椅子が固いうえに多くの机が並んでいたため通路も狭く、あまり居心地のいいものではなかったが、暖房が効いていたため室温はそれなりに快適だった。模試の半分が終わるころには長い間座っている事による肩こりと背中の痛みで非常に辛かった。現役の時はこんなのを1年に何回も受けているのかあ。最後の理科②では途中退室が許されていたので、さっさと問題を片付け退室した。その後は、会場近くにあるマクドナルドで夕飯をとりつつ自己採点し、鉄道でつくば駅に戻り解散となった。

## 3 結果発表

気になる結果は以下に示す通りである。受験した科目の結果を時間割の順番で発表する。また、そしあの点数は最高点、最低点、平均点のみ掲載する。

---

\*2 現役生のこと。

### 3.1 地歴・公民

表3:社会の点数

| 社会     | 受験科目 | 点数   |
|--------|------|------|
| UTM    | 倫理   | 52   |
| ,無季    | 地理   | 56   |
| WORD平均 |      | 54   |
| そしあ最低  | 日本史  | 52   |
| そしあ最高  | 世界史  | 68   |
| そしあ平均  |      | 57.7 |

社会の点数は大きく引き離されてはいないが、さすが社会学類と言うべきか、そしあがリードした。筆者はここで社会の最低点を取ってしまう。なんで倫理にしたんだろ……。

### 3.2 国語

表4:国語の点数

| 国語     | 点数    |
|--------|-------|
| UTM    | 162   |
| ,無季    | 99    |
| WORD平均 | 130.5 |
| そしあ最低  | 128   |
| そしあ最高  | 157   |
| そしあ平均  | 143   |

国語は筆者の圧勝である。国語は筆者の圧勝である。<sup>3</sup>が、平均にするとそしあの勝ちである。,無季さん、まじ戦犯。絶対に許さない

### 3.3 英語・リスニング

表5:英語・リスニングの点数

| 英語     | 英語    | リスニング | 合計(200点) |
|--------|-------|-------|----------|
| UTM    | 113   | 36    | 119      |
| ,無季    | 114   | 26    | 112      |
| WORD平均 | 113.5 | 31    | 115.5    |
| そしあ最低  | 91    | 22    | 90       |
| そしあ最高  | 156   | 38    | 150      |
| そしあ平均  | 128.7 | 32.7  | 128.7    |

英語はそしあが13点の差を付けての勝利である。そしあは1人だけ何か低い点数を取っているが、そのほかがいい点数をとっているので、平均では大きく離された。

---

\*3 大事なことなので2回言いました。

### 3.4 数学I・A

表 6: 数 I・A の点数

| 数IA    | 点数   |
|--------|------|
| UTM    | 94   |
| ,無季    | 59   |
| WORD平均 | 76.5 |
| そしあ最低  | 26   |
| そしあ最高  | 86   |
| そしあ平均  | 59.7 |

数学では WORD が圧勝……と思いきや、そしあにも 80 点以上の高得点を取る人が 2 人いた<sup>\*4</sup>。数学は問題数が多いので時間配分が重要である。ちょっとでもわからないと感じたらすぐ飛ばして次の問題をやつた方がいい。

### 3.5 数学II・B

数 II・B の結果は表 7 に示す。

表 7: 数 II・B の点数

| 数II・B  | 点数   |
|--------|------|
| UTM    | 72   |
| ,無季    | 37   |
| WORD平均 | 54.5 |
| そしあ最低  | 1    |
| そしあ最高  | 37   |
| そしあ平均  | 21.7 |

こちらは WORD の圧勝となった<sup>\*5</sup>。そしあ最低点 1 点……あつ……（察し）。WORD 最高点である筆者 (UTM) と比較すると、実に 72 倍もの差がついている。そしあ最高点と比べても倍近い。なぜなのか。

数 II・B も数 IA と同じく楽勝かと思われたが、1 問目の図形問題で和積の公式をど忘れしてしまったためかなりの問題が空欄のままになってしまった。



図 1: そしあ最低点を見たときの筆者

\*4 だが筆者の圧勝である。

\*5 筆者が圧勝したおかげである。

### 3.6 理科

理科は受験科目が受験者によってまちまちなので、2科目を表8にまとめて発表する。

表8:理科2科目の点数

| 理科     | 化学   | 生物 | 物理 | 理科総合 |
|--------|------|----|----|------|
| UTM    | 43   | 68 |    |      |
| ,無季    | 49   |    | 88 |      |
| WORD平均 | 62   |    |    |      |
| そしあ最低  |      | 27 |    |      |
| そしあ最高  |      |    |    | 68   |
| そしあ平均  | 47.3 |    |    |      |

理科も理系であるWORDに軍配が上がった。,無季さんが他を大きく引き離す88点を取る。化学なんて無かった

生物も国語と同じく現役の時は自信のあった科目だったが、暗記系であるためかかなり点数を落としてしまう結果となった。予定運命図<sup>6</sup>なんてもう覚えてないよ。

### 3.7 総合点数と志望校判定

総合点数は表9の通りになった。なお、総合点では英語とリスニングの点数を200点に圧縮し、全900点として算出される。

表9:総合点数と志望校判定の結果

| 総合     | 総合点(900点) | 情報科学類判定 | 社会学類判定 |
|--------|-----------|---------|--------|
| UTM    | 610       | C       | C      |
| ,無季    | 500       | E       | E      |
| WORD平均 | 555       |         |        |
| そしあ最高  | 579       | D       | C      |
| そしあ最低  | 369       | N       | E      |
| そしあ平均  | 491.5     |         |        |

見事にC～E判定ばっかりである。個人成績表裏面の「個人成績表の見方」によると、B判定は合格可能性60パーセント以上、C判定は合格可能性40パーセント以上、D判定は20パーセント以上、E判定は20パーセント未満となっている。つまり、全員合格可能性60パーセント未満であるため、本番の試験での筑波大学合格は絶望的だろう<sup>7</sup>。

なお表中の「N」というのは、「未受験科目があるため判定不可能」を表している。理科を一科目しか受けていなかったため、情報科学類の判定が出なかったものである。

---

\*6 フォークトさんがアレして作ったやつ。原基分布図ともいう。

\*7 ちなみに、デジタルハリウド大はA判定だった。

### 3.8 勝敗

全員が当時の学力を全く維持できていないことがバレてしまったが、今回の勝負は総合点を比べることであるためバカだったとしても相手がそれ以上にバカならいいわけで全く問題が無い。最高点・最低点・平均点それぞれの勝敗の結果を表 10 にまとめた。

表 10: 勝敗の結果

|      | 最高   | 最低   | 平均    |
|------|------|------|-------|
| WORD | 610  | 500  | 555   |
| そしあ  | 579  | 369  | 491.5 |
| 勝ち   | WORD | WORD | WORD  |

すべて WORD の勝利!!



図 2: 勝利が確定したときの筆者

# (・'ー・')どや

そしあ側には1年生が3人、しかもそのうち2人は前期試験を通過してきているため、結果を受け取るまでは「コリヤ負けるかもなあ」と思っていたがそんなことはなかったぜ!

ちなみに、最高得点・最高得点者・最低点・最低得点者は次の表11のようになつた。戦犯が一目でわかるぞ<sup>\*8</sup>!  
これをみると、WORDの勝因はそしあ内に足を引っ張る人がいた最高点を人数の少ないWORDで多く取ることができたからだと考えられる。

表11:最高点・最低点とその取得者

| 科目    | 社会         | 国語  | 英語  | 数学IA | 数学IIB | 理科2科目 |
|-------|------------|-----|-----|------|-------|-------|
| 最高点   | 68         | 162 | 150 | 94   | 72    | 88    |
| 最高得点者 | そしあ        | UTM | そしあ | UTM  | UTM   | ,無季   |
| 最低点   | 52         | 99  | 90  | 26   | 1     | 27    |
| 最低得点者 | UTM<br>そしあ | ,無季 | そしあ | そしあ  | そしあ   | そしあ   |

## 4 おわりに

WORD vs. そしあセンタープレ対決はいかがだつただろうか? 前回の雪辱を見事に果たす大勝利だった。「そしに1年生が3人もいる」、「都合により4~5人いたはずのWORD側の参加者が最終的には長老2人で挑む羽目になった」などの理由により、前回以上の大敗を喫してしまうかと思われた。が、なぜか大差をつけて勝つてしまつた。<sup>\*9</sup>。(・'ー・')イヤーナンデ' タ' ローネ

そしあは1年生が3人いたがその全員が3年である筆者に負けてしまつてゐる。こんなんじや勝負にならないんだよ(棒読み)。社会・国語の文系科目と英語ではそしあに負けてしまつたが、数学2科目と理科2科目では大差をつけて勝つことができた。

今回の結果、そしあに勝つことができたことは非常にうれしいが、ひとつ心残りがあるとすれば、目標にしていた現役超えが達成できなかつた<sup>\*10</sup>ことだろうか。結構疲れる上1日潰れてしまうのは正直辛いが、また機会があれば挑戦したいと思う。次はス●ブラか麻雀にしない?

\*8 と思ったけど最低点取得者はほとんどがそしあである上に、そしあ側は全て「そしあ」と表示されるのでわからなかつた。ザンネ!

\*9 もちろん筆者の圧勝である。

\*10 現役の時はたしか自己採点で680~700点程度だったと記憶している。センター試験の点数を通知してもらえる成績通知を申し込んでいたが、結局届かなかつた。800円返して!

# 山口喜教教授 退職直前インタビュー

文 編集部 はろぺり

前学類長、山口喜教先生（シス情系・教授）は、今年度をもって本学を定年退職する。これを機に本誌では2時間に及ぶインタビューを行った。ここではその内容を整理した上で掲載する。

## 本学に着任する前に教えてください。

生まれは別府市ですが、小学校に上がる際に立川へ引越し、その後30歳くらいまで住んでました。駿台御茶ノ水校で1年間浪人し、東大に入りました。浪人生活もけっこうおもしろかったです。コンピュータよりも、回路やトランジスタ、オーディオなどが好きだったので、東大ではそっちの道に進みましたが、卒論ではミニコンで並列計算を行うようなことをやりました。当時研究室で指導していただいたのは、のちに第五世代コンピュータを推進した元岡先生です。

公務員試験を受け、研究職かそれ以外かで悩みましたが、こちらが合ってるだろうと思って電子技術総合研究所<sup>\*1</sup>に入所しました。配属先は計算機方式研究室というところでした。そのころはMITでAIの研究が盛んで、電総研でMITの人が講演する機会などがあり、人工知能を使われたLISPに興味を持ち、そのような関係で私はLISPを早く実行するためのアーキテクチャ（LISPマシン）の研究をしていました。その後1980年代になると、データ駆動<sup>\*2</sup>の考え方に対する興味を持ち、新しく提案したアーキテクチャを実装して評価したりしていました。

## いつから筑波に来たのでしょうか？また、本学に着任する経緯を教えてください。

入所したころ、電総研は田無と永田町の2か所に分か

## 略歴

山口喜教（やまぐち・よしのり）

博士（工学）。専門は並列計算機アーキテクチャ、並列実時間システムなど。昭和24年大分県別府市に生まれ、小学校時代に東京都立川市に引越す。昭和47年、東京大学工学部電子工学科を卒業し、電子技術総合研究所に入所。在勤中に英・マンチェスター大学へ留学。平成11年に本学教授に着任し、学術情報メディアセンター長、情報科学類長を歴任。平成3年情報処理学会論文賞、平成7年市村学術賞受賞。著書に『データ駆動型並列計算機』（共著）がある。



れていて、私は永田町にいましたが、本学開学から2年後の昭和54年に筑波研究学園都市に移転しました。ちなみに永田町のほうは、今の首相官邸の場所です。その後平成11年1月、筑波大学の教授に着任しました。

筑波大学に来たのは、学術情報メディアセンター（学情センター）に教授がほしいという話があり、「近いし、いいかな」と思って引き受けたからです。でも最初はあまり大学教員になりたくなかったんですよね。私が学生のころは大学闘争が盛んだった時期で、教室がロックアウトされて授業がなかつた期間もありましたし、学生に吊るし上げられる先生も

\*1 産業技術総合研究所の前身のひとつ

\*2 データ駆動：計算モデルのひとつ。ある処理で生成されたデータに応じて、次に行るべき計算を連鎖的に起動する

いて、大学教員は大変だと思っていました。でも時代も変わったようなので、いいかなと考えが変わりました。

**本学ではどのようなことをやりましたか？**

情報科学類長以外の役職としては、学情センター長があります。学類の計算機室を持つ情報科学類生には馴染みがないかもしれません、全学の教育計算機システム(サテライト)の導入、メンテナンス、トラブル対処などを行い、けつこう大変でした。

研究面では、予算の都合もあり電総研でやっていたものよりも、ややコンパクトなことをやっていました。

FPGAによるネットワークの侵入検知の研究がその1つです。2000から3000個くらいのルール(パターン)を1つのチップに収まるような回路にするものです。圧縮するだけでなく、速度も重要でした。また、最近ではGPGPUによる並列計算の研究もやっています。

**先生は平成20年度から5年間、学類長を勤めましたね。学群学類再編の直後でしたが、何か苦労はありましたか？また、他に変わったことはありましたか？**

前任の北川先生(現CS専攻長)がCS専攻長になり、私に学類長の話が回って来ました。最初は1年くらいかと思っていたのですが、気づいたら5年やってました。

たしかに平成20年度は第三学群情報学類から情報学群情報科学類に再編した年ではありますが、これに関する苦労は特にありませんでした。私の前に学類長を務めた田中二郎先生(現学群長)、北川先生は苦労なさったんじゃないかと思います。ですが、それでも学類長としての仕事は忙しかったですね。

変わったこととしては、まずはCOJT<sup>3</sup>や情報科学基礎

実験<sup>4</sup>といった体験型科目の充実をはかりました。情報科学類生には閉じこもりがちな学生もいるので、横つながりや先生との交流を深めてほしいと思っていました。

あと、新歓行事で合宿をするようになりました。3回目となる来年度からは学類の公式行事になるようですね。実は付き添いの先生方が大変といった議論もあったのですが、学類の横のつながりを強めるいい機会だと思います。

**逆にまだできていないことなどはありますか？**

情報科学類の位置づけ、特に情報メディア創成学類(メ創)とのちがいについて考えてました。どちらも理系で高校生への説明が難しく、まだモヤモヤしています。

それと、極端に少ない女子学生を増やしたかったです。ガリガリとプログラミングばかりするわけでもないし、情報科学は男性に偏った学問ではないと思います。Webサイトのデザインを変えたりもしましたが、こういうところではメ創のほうがうまくいっている部分があります。



\*3 COJT:組み込み技術キャンパスOJT。平成21年度から開講されている情報科学類、情報メディア創成学類3年次向けの授業

\*4 情報科学基礎実験: 平成23年度から開講されている1年次向けの必修授業。Arduinoを使った実習を行う

**授業は「論理システム」と「コンピュータネットワーク」の担当ですね。これはどのような授業だったのでしょうか。また、心がけたことや、授業を通して感じたことなどはありますか。**

論理システムは長くやっています。最初は「論理回路」の発展としてやっていましたが少しずつ内容を変え、ハードウェア記述言語（Verilog）を教えるようになりました。これにはハードウェアに対するアレルギーをなくしたいというねらいもあります。コンピュータネットワークは、退職された海老原先生に代わって、佐藤聰先生と分担してやりました。私は前半部で、IPと、TCPの入り口くらいまでを教えました。

授業をしていて感じたこととしては、他の学類もそうかもしれないけど、学生があまり質問をしてくれないことが気になります。学生がどういうところに興味を持っているのかわからぬし、自分のペースだけで授業を進めていくしかないんですね。みんな「ついてくる」といった感じです。静かで、試験の出来も大体いいのですが、もうちょっと前のほうに座って聞いてほしいと思います。

**ではもう少しプライベートな話をお伺いさせてください。まず、OSやコンピュータは何をお使いですか？**

MacBookも持っていますが、Windowsがメインです。Macも好きだし、家に置いていた時期もありましたけど、仕事ではWindowsのほうが便利なので。

**プログラムを書く機会はどれくらいありますか？また、言語は何でしょうか？**

電総研ではSimulaというオブジェクト指向の言語やCで並列計算のシミュレータなどを書いていましたが、こちらに来てからはあまり書かなくななりました。何かを書く時はCを使います。あとは、プログラミング言語ではなくハードウェア記述言語ですが、Verilogも書きます。

**休日はどのようなことをしますか？**

特にコンピュータ関係の趣味はないですね。美術館に行ったり、音楽を聞いたりします。

音楽は、グレンガールドなどのピアノやクラシックなど、それにThe Beatles、Queenなどのロックも聞きます。最近の曲だと、平原綾香やいきものがかりも好きです。イギリス留学から帰国する際に買ったQUAD製の大きなコンデンサスピーカーをかれこれ25年近く使っています。

それから録画したテレビ番組を見ます。簡単に録画できるのでつい録り溜めてしまい、すると全部見なきやいけない気がして、2倍速で見たりもします。ただし聞き取れなかったところだけは戻して通常再生しますね。

**退職後はどのようなことをしたいですか？**

一人でゆっくりできるような研究テーマを探しているところです。ひとつは、どうすれば生物の知能はコンピュータで実現できるのかを考えてみたいと思っています。人工知能と生物の知能とのちがいについて本質的なことは分かっていません。最近では処理能力が高まって知能に近いものができるようになっていますけど、それは本当の知能とは違うように見えます。

**最後に学生へエールを送っていただけませんか？**

情報科学類は、それなりにセレクトされた学生が集まっています。これからグローバルな社会において、昔よりも大変なことはいっぱいあると思います。しかし「自分たちが新しい時代を作っていくんだ」と思って、もっと頑張ってほしい。それだけの能力がみなさんにはあります。

すでに自分たちでいろいろやっている人も情報科学類にはいます。そういう人たちは、もっと伸ばしていってください。

# 突撃！編集部員の所持品検査

文・写真 編集部 はろぺり

## はじめに

みなさんこんにちは。突然ですが、友だちのカバンの中身を隅々まで見たことはありますか。そうですよね、ないですよね、気になりますよね。ということで、編集部員のカバンの中身を漁って激写して尋問してみました。大体みなさん同じようなものを持ち歩いてますが、細かいところに個性が出るようです。この記事では目に付いた部員の荷物を、尋問の様子を交えつつ取り上げます。なお、筆者の手際の悪さゆえ、ポケットの中身までチェックしそびれた人もおり、検査にはムラがありますがご了承ください。

## 登場人物紹介

は:はろぺり(筆者)

ゆ:ゆうたん

む:, 無季

い:いおりん

ひ:ひだるま

み:みみずのひもの

か:かづきお

PJ : PJ

げ: Genyakun

ふ:ふたばちゃんになりたい

## 所持検

PC/外部ディスプレイ用ケーブル/マウス/充電器/ポストイット/筆入れ/筆入れからあふれたペン/財布/手帳×2/講義資料/ノート×2/ケーブル類(携帯電話のケーブル、Dock ケーブル、オーディオの延長、USB micro 、USB mini )/USB 充電器/分配器

筆箱とは別にゴソゴソ出てくるペン。

は:うわー、この先輩きたねえ

ゆ:きたねえって言われたよ……

は:このケースはなんですか？ああ、ケーブル類か

ゆ:これ 100 均の DS ケースなんだよね

む:ところで USB メモリ持ってるんだ。情報科学類生って

## ゆうたん

### 基本データ

はじめはゆうたん先輩。部内屈指の一太郎スキルを持ち、本誌連載『WORD 読者アンケート』の現筆者。独特のユーモアとセンスを持ち合わせる WORD 内の古株

の一人。家電量販店でバイトをしている。

|      |              |
|------|--------------|
| 名前   | ゆうたん         |
| 学年   | 2 年次 5 年     |
| PC   | Asus UX31A   |
| 携帯端末 | GALAXY NEXUS |



みんな宅鯖<sup>\*1</sup>とか Dropbox とか使うよね

ゆ: 人に渡す時にたまに使うんですよ

む: 『ユーエスピー<sup>\*2</sup> でちょうどいい』みたいに？ (笑)

ゆ: そうですね(笑)

は: あ、この手帳私のと色違った。オソロのイロチだね☆ ところ

でなんで手帳が 2 冊も？

ゆ: 1 冊買ったあとに、バイト先でもらったのでとりあえず入れてる

は: イチオシの持ち物はありますか

ゆ: それはこの Windows 7 の販促品のボールペンでしょう。ペンライト付き！

は: !!!これをどこで手に入れた！？

ゆ: これもバイト先で……

電気屋さんのバイトはいろいろ手に入るらしい。

## ，無季

### 基本データ

現役部員の中では最古参の、無季さん。学年だけではなく、工シス出身であるという点でも部内では異色。さまざま相

|      |                        |
|------|------------------------|
| 名前   | ，無季                    |
| 学年   | 1 年(博士後期)              |
| PC   | MacBook Pro 15" Retina |
| 携帯端末 | SH01C                  |

談に応じてくれ、1 年生からも慕われる目標にできる先輩。酔うとセクハラしてくるところ以外は……。

### 所持検

ティッシュ×4/キレイキレイ×3/応急手当キット/うちわ/マウス/マウスパッド/eneloop/eneloop 充電器、携帯電話充電器/ケーブル類(充電、iPad 用、携帯充電 × 2)/歯ブラシ/ハンドクリーム、制汗剤、プレスケア/iPad mini/分配ノギス/根付(除夜の鐘をついたらもらった)/タオル/買い物袋/PC/筆入れ/コンデジ/財布/静電気対策リストバンド<sup>\*3</sup>/ケンジントロック<sup>\*4</sup>

は: うわ、なんだこの除菌厨。なんでキレイキレイがいっぱい？

す: 増えた

は: 増えるキレイキレイ……。ケンジントロックは何に使うの？

す: 学会の展示とかに使います。でもこの MacBook Pro、付ける所ないんだよね



，無季さんのカバンの中身

\*1 宅鯖: 自宅においてあるサーバーのこと。この場合、自前で管理するサーバーというような意味。分かりやすく言えば、「インターネット経由でどこからでもアクセスできるパソコン」

\*2 USB ( Universal Serial Bus ) はあの穴に関する規格の総称。「USB メモリ」と呼ぶのが正しいが、それを「USB」と呼ぶ人は後を絶たない

\*3 静電気を防ぐために体を接地するプレスレット。電子工作に使う

\*4 盗難防止のために PC につけるワイヤー錠。電器屋や展示スペースなどで使われる

は:持ち歩いてる意味あるんですか？

す:哲学！！……まあ学会で他人のPCを使う機会もあ

るし

は:一番気に入っているものは？

す:ノギスです。いつも使うので(追真)新しいカードゲームのスリーブを買うためにカードの大きさを測ったり、ものづくりに使ったりもするし

## ひだるま

### 基本データ

1年のひだるまくん。寡黙だけどその内に秘められた

能力は未知数。最近では編集部の一太郎のマクロ機能でプログラミングをはじめたという噂も……。

|      |              |
|------|--------------|
| 名前   | ひだるま         |
| 学年   | 1年次 1年       |
| PC   | ThinkPad X61 |
| 携帯端末 | N031C        |

### 所持検

エスペラント語入門/ノート×2/独和辞典/Pythonの入門書/手帳/単語カード/欠席届/HDD(1TB)/AMER/WORD×3/筆入れ/クリアファイル(講義資料)/PC

は:言語の入門書ばっかりだね

ほ:ひだくんドイツ語は？

ひ:30分くらい遅れても大丈夫だし

は:WORDがいっぱいあるなあ



ひ: mbed の記事が読みたかったので

は:手帳大きくない？

ひ:いや、そんなことないと思いますよ

は:イチオシの荷物は？

ひ:過去の WORD です(キラッ☆！)

は:編集部員の鑑だ。HDD の中身はエロ画像とか？

ひ:も、入ってますね

は:も、入ってるのかよ！アメが入ってるけど、いつもお菓子常備してるの？

ひ:いや、もらったものがそのまま入ってるだけです

は:ノートはどうやって使い分けですか？

ひ:授業用とメモ用です

は:手帳はスケジュール管理？

ひ: Google カレンダーと手帳で管理してる

は:同期はどうしてる？

ひ:とりあえず手帳に書いて、あとで時間がある時に

Google カレンダーに入力します

は:で、ドイツ語は大丈夫なのか？

ひ:結構遅刻しても、「(遅刻)」程度なので

は:あ、そうなのか。でもごめん。急ぎたまえよ

## みみずのひもの

### 基本データ

「見た目は大学1年生、ちょっとお茶目な男の子。アイドルをマスターする感じのゲームと、マジックをギャザリングする感じのカードゲームが大好きなん DA ! 皆からはホモホモ呼ばれてるけどそんなことないもん、ぶんぶん(憤慨)! 気がついたらちょっぴりくさそうでえっちな動画をオススメされて、言うがままに見せられてるだけなんだからね(暗黒微笑)！」と本人談。

|      |                |
|------|----------------|
| 名前   | みみずのひもの        |
| 学年   | 1年次 1年         |
| PC   | Aspire M5-481T |
| 携帯端末 | T003           |

## 所持検

筆入れ/PSP/ワイヤーの鍵/免許証/傘入れ(傘なし)/パスケース(中身は空)/印鑑/ノート×2/千早さんのクリアファイル(講義資料など)/使い捨てカイロ/メガネ拭き兼メガネ入れ/メガネ拭き(ルーヴル美術館で買った)/ウォークマン/電子辞書/iPod Touch/扇子

す:免許証の写真イケメンだ。キリッとしてるぞ  
は:なんかムカツきますね。ところで自転車のワイヤー錠は  
なんでここにあるの?付けてないの? そういえば自転車  
のワイヤー錠盗まれたとか言ってなかつたっけ?

す:なんでワイヤー錠だけ盗まれたの?  
み:さあ、なんででしょう……。盗まれたから新しく100均  
で買ったんだけど、それとは別に新しいのが親から届き  
ました。なんかに使うかなーと思って持ち歩いてます  
は:パスケースが空なのはなんで  
み:たしかMTGのカードを入れる何かだったと思う  
は:アイマスの缶バッヂがこんなにあるのはどうして  
み:同人で1冊買うとついてくるんです。どうせつけるなら

身近な所につけておこうと  
は:このクリアファイルの子はアイマスの千春ちゃんだっけ?  
み:千早ちゃんです!  
?\*:千春ちゃんは【禁則事項です】学類の【禁則事項で  
す】学のお方です!!!!  
は:ミスドの布巾が入ってる。これも君の?  
み:ちがいますよ。これは僕のじゃない\*  
は:ノート2冊はどうやって使い分けて?  
み:授業と落書きで分けてます  
は:プリキュアのスタンプ台紙はなんなの?  
み:ミニストップのバイトの帰りに、スタンプを押したもの。

4種類集めるとステッカーがもらえます



は:特に気に入ってる文房具はある?

み:このペンは親から引き継いだもの

は:ふーん、まあそれはいいや

み:あ、はい。すいません……

は:一番よく使うのは?

み:PSPとかウォークマンですね

は:といえばドイツ語を取ってたね。今ドイツ語Bの授業らしいけど?

み:い……1学期に切りましたorz

## かづきお

### 基本データ

自動車部にも所属しているかづきお氏。車の知識において、

| 名前   | かづきお   |
|------|--|
| 学年   | 3年次 6年   |
| PC   | ThinkPadX200                                   |
| 携帯端末 | BlackBerry 9810、<br>GalaxyNote、<br>GalaxyNote2 |

て、部内で彼の右に出るものはいない。サングラスやZIPPOライターなど、他の情報科学類生にはないような渋い荷物が検出された。

\*5 編集室のどこかから聞こえてきた声です

\*6 編集室は散らかっています。ご迷惑をおかけして申し訳ございません

## 所持検

空の封筒/手袋/筆入れ/外付けの DVD ドライブ/USB ケーブル(スマフォ用)/デジカメ/サングラス/目薬/小物入れ(カフェイン錠剤、頭痛薬イブ、データ通信機、USB の AC アダプタ、Bluetooth アダプタ)/2 枚組 CD × 2 /Panaloop ( panasonic の充電池)/懐中電灯/ウェットティッシュ/SL のキーホルダー  
ポケットの中: 携帯 × 3/ZIPPO ライター/カイロ(オイルで暖かくなる。ライターで点火)/ティッシュ × 2/ボールペン

か: 空の封筒とか、ゴミとかはいいよね。見られて困るゴミもあるし  
は: ホテルのライターとか注射器の針とかですか?

い: ひやー

か: そういうこっちゃ

(まじかよ。お巡りさんコイツです)

か: 手袋じゃろ、財布じゃろ

い: え? 財布! ?

か: 財布くらい持っててもいいだろう。……はははは! これ

は筆箱やった orz

(お疲れ気味である。話しかけてよかつたのかな)



かずきおさんのカバンの中身

は: お、筆箱からメモリが? ……いや、定規か  
か: RIMM<sup>7</sup> のターミネータだね。あと筑波大学のシャーペンもあるよ

は: おお、IMAGINE THE FUTURE.ですか。いいですね。私も持っています

か: いや、入学当初 UT-Shop で買ったやつ。あの頃は ITF.なんてなかった

は: あ、そっち系か? ……orz サングラスは運転用ですか?

か: いや、かけようと思ったらいつでもかけるよ。眩しい時とか

か: わしは全部上着の中に入れて持ち歩くからなあ。(ポケットをガサゴソ) 真岡鐵道<sup>8</sup>の乗車証明書。さつき行つてきた

は: 携帯が1つ2つ……3つ! ? たくさんありますね

か: ないよ。3つしかないよ。情報科学類生にしては3つくらい少ないほうじゃないかな。とか言ってみる

は: 私は手負いの iPhone しか持っていないませんけどね

か: あとはボールペン、目薬。これで以上、かな? ほんとかな? ……あ、これで以上だ(ゴミを捨てながら)。何もおもしろいものはないね

は: イチオシは?

か: エンジンのスパークプラグのキーホルダー。エンジンを点火させるための電極

は: ガスコロを点火する時のバチバチする部品みたいなやつですね。なんでこんなものが?

か: エンジンの高温にも耐えられるイリジウム、白金などの丈夫なもの。電極の先は尖って危ないから切つてある(そうじゃなくて、なんでそんなの持ってるのさ)

か: あとは……ティリリーン! MUSTANG のライター

は: 格好いいですね。ところで全体的に黒いものが多いで

\*7 メモリの規格のひとつ

\*8 詳細は今号の『GR な日々。XIV』を参照

すけど、趣味でしょうか？『男なら、黒に染まれ』みたい  
な？

か：いや、たまたまなつただけ。大体、黒いものしか売ってな  
いでしょ

は：たしかに、なんとなく買ってると黒いものばかりになって  
しまいますよね。

か：片付けんのめんどうくさいな

は：すいません。ご協力ありがとうございました

## PJ

### 基本データ

|      |                 |
|------|-----------------|
| 名前   | PJ              |
| 学年   | 2年次3年           |
| PC   | MacBook Pro 13" |
| 携帯端末 | iPhone4、iPad    |

### 所持検

三脚/Vita/薬/デジカメ、充電器/懐中電灯×2、電池  
/1W レーザー/血圧計/PCM レコーダー/整髪料  
/PC/iPad/関数電卓/鍵/GPS レコーダー/エネループ/メモ  
リカードリーダー/マウス、マイク(Bluetooth)/ディスプレ  
イの変換ケーブル(mini display port to D-sub / DVI)  
/USB コンセント/USB ハブ/イヤフォン/LAN ケーブル/ス  
トレート・クロス変換アダプタ

は：うわあ、大物だ。工具箱みたいなごつついかいばんだね。

いつも持ち歩いてるの？

PJ：はい

は：D800 とは別にデジカメがもう一つあるんですね。デ  
ジカメに予備なんて必要？



PJさんのカバンの中身

PJ：天井裏を撮る時とか

は：これは懐中電灯と、懐中電灯と、懐中電灯と……

PJ：いや、3つ目はレーザー

は：レーザー？ レーザーpointer？

PJ：1W だよ

は：1W がどれくらいの出力なのか、読者にも分かるよう  
に説明してもらえますか？

PJ：普通のレーザーpointerの1000倍。目で直接見  
たら焼けるレベル

(こわつ。ちかよらんとこ)

は：GPS レコーダーは、今も記録してるんですか？

PJ：いや、旅行に行く時だけ

は：むやみにログを公開すると、時間と位置からスピード違  
反がバレてしまいますね

PJ：そうだね(意味深)

は：これは財布？

PJ：いや、資格証ケース<sup>9</sup>

\*9 自動車免許、陸上特殊無線技師、特別教育(アーク溶接/フォークリフト/クレーン/高所作業車/ゴンドラ取り扱い/小型車両建設機械/整地)、技能講習(玉掛け/ガス溶接)

は:いっぱいありますね。こっちの場所はケーブル類ですね。大きいカバンを使ってると、ポーチなしでも整理できるのはいいですね。重いけど。黒いものが多いのは趣味ですか？

げ:男子の特徴だよね

PJ:といふか、白いものって汚れそうだよね

は:そうかもしれませんね。ところで、三脚は重そうだけど、よく使うんですか？

PJ:全然使わない

は:それでも持ち歩くの？なんで？ねえなんで？

PJ:いざ使う時にわざわざ取りに帰らなきゃいけないことを考えると、持ち歩いていたほうがいい

は:なるほど。ワックス使うんですか？

PJ:前髪が伸びた時は使う。サラサラしてるからね

ふ:むつかつくよね

(彼女(ふ)の髪は太くて硬い)

は:むつかつくよな

(筆者も同じく)

PJ:本当にさらさらなんだよねー^^

は:一番大事なのものはなんですか？

PJ:iPad……いや、MacBookだね。あ、そういうば懷

中電灯も、なにげに1万円するんだよね

(※彼は末期の懐中電灯マニアです)

## Genyakun

### 基本データ

部内のみならず、各所のネットワークを管理するGenya

|      |                              |
|------|------------------------------|
| 名前   | gennyakun                    |
| 学年   | 2年次4年                        |
| PC   | Let's Note                   |
| 携帯端末 | F-02D / F-05A / iPad / BF01D |

kunさん。やはりネットワーク屋らしい荷物が多く検出されました。

### 所持検

チロルチョコ(きなこもち)/筆入れ/USBケーブル類(Foma、スマフォ、PSP、PS Vitaなど)/ウォークマンのオーディオ入力ケーブル/Fomaの通信ケーブル/薬箱×2/シリアルケーブル×2/シリアルケーブルのオスメス変換/ストレート・クロス変換アダプタ/充電池×3/ポイントカードケース/Vita/レジ袋/iPad/ヘッドセット

豪快にカバンをひっくり返す Genyakun 氏。

げ:おら、きなこもちやるよ

PJ:わあい！

は:ものが少ないです。ケーブル類が多い

げ:『軽くて、持ち運びがよくて、機能性のあるもの』がカバンのコンセプトです。最低限のものに絞って、あとは必要に応じて追加します。PS Vitaもその内外れるかもしれません

は:素晴らしい。何が素晴らしいって、カバンのコンセプトを語ってきた人はあなたが初めてですよ

は:ケーブル類が多いんですけど、ポーチは使わないんですか？

げ:はい。もの少ないし、カバンの中のポケットで整理できるので



は:思い入れのあるものは？

げ: Let's Note かなあ。高 1 でノート PC を買う時に、バッテリー駆動時間が公称 8 時間で軽くて丈夫なものが欲しかった。ThinkPad か Let's Note に絞られたけど、国産でバッテリーの持ちも長く、サイズが手頃だったので買った。値段は手頃じゃないんだけどね

は:なにか欲しいものがありますか

げ: デジタルガジェットいっぱい欲しい！ 機種変で新しいスマフォ欲しい！！

## ふたばちゃんになりたい

### 基本データ

|      |                 |
|------|-----------------|
| 名前   | ふたばちゃんになりたい     |
| 学年   | 3 年次 3 年        |
| PC   | MacBook Pro 13" |
| 携帯端末 | iPhone 4        |

### 所持検

財布/パスケース(テレfon カード、ミミロルのシール、  
Pasmo、お守り、非常時における本学の安否確認の説明書き)/使い捨てカイロ/ヘアゴム/B6 の便箋/鍵  
/WORD24 号/下敷き/クリアファイル(講義資料)/PC/トラックボール/折りたたみ傘/イヤフォン/ポーチ/筆入れ(シャーペン、ペン、4 色ボールペン×2)/ケーブル類

は:何これ。折りたたみ傘？

ふ:うん。ケースのポケットに入っているスポンジで、チャリのサドルを拭けるんだよ

は:それは便利。すごい。というか、チャリ乗るんだ

ふ:乗れるよ。バリッバリやで！

は:このお守りはどこの？



ふたばちゃんになりたいさんのカバンの中身

ふ:受験期にもらったので分からぬ

は:よく持ち歩いてるね。これのおかげで単位も来る

ふ:来るかなあ……？

は:テレfon カードに XP のライセンスシールが付いてる……？

ふ:まあそういうこともあるよ

(なるほど分からん)

は:ところでリュックの外側のポケットに鍵を入れるのは、落としそうじゃないですか？

ふ:まあリュックも危ないけど、ポケットに入れておいて、iPhone を取り出した時に落とすほうが怖い

は:私の場合、右ポケットが iPhone、左が鍵、後ろが財布と決めてます。ところで、筆入れが小さい割に 4 色ボールペン 2 本もあるんですね

ふ:このコンパクトなのはゆずれない。片方は組み立てられるやつで、色を選んで買うやつなんだよ。水色と茶色と橙色と桃色

は:ノートはカラフルに取るの？

ふ:ブロック図を書く時だけ DANA。入力・出力・その他いろいろとで、色を分けたいシーンがある

は: B6 のノート使うのはどうして？

ふ:机にPC置くと、場所がないんだよね。PCの端に乗せて書ける小さいのがいい。そもそもあの机ちっちゃいんだよね

は:このポーチに描かれてるブタさんは?

ふ:モノクロブー。高校の時にはやった。友だちにゲシゲシされて、端っこが削れてるけど

は:トラックボールの長所ってなんですか?

ふ:ベッドでもマウスパッドなしで使える。マウスよりは疲れない。そして振り返るのが楽(FPSゲーム的な意味で)

は:このリュックはどうしたものでs(ry

ふ:シマムラ!えっと、1990円くらい。前使ってたのが破れたから買った

は:目下、欲しいものはありますか?

ふ:圧倒的にペンの色が足りない。軽くて色の多いペンか色鉛筆が欲しいかな

は:ご協力ありがとうございました

ふ:お、おういえー

## おわりに

全体的にケーブル類が多いと感じました。たくさん持っている人はケースに入れたり、カバンのポケットを活用して整理しているようです。ほぼ全員がPCを持ち歩いていました。重いから持ち歩かないという人もいましたが、これは情報科学類生の特徴のひとつだと思います。

## 注意

この記事では編集部員と筆者周辺の情報科学類生の所持品検査を行いました。ここで一点注意していただきたいのですが、ここでご紹介したのはあくまで個人の荷物であり、情報科学類生の荷物には個人差があります。ですから例えば読者諸氏周辺の情報科学類生がiPhoneの充電ケーブルを持っていなかったからといって、「お前情報科学類生のくせに持ってないのかよ。使えないやつめ」などと言わないでください。筆者もたまに思いますけど声に出さないでください。決してデジタル便利屋ではないので。よろしくお願ひします。

# オスっちメスっち育成日記

文 編集部 みみずのひもの

## 1.導入

メスっちが死んだ。

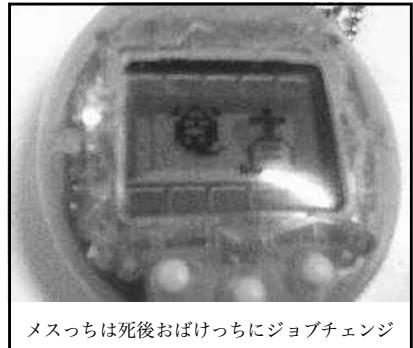
享年は4才、発見時の体重は僅か68グラムしかなかった。恐らく私がバイトに行っている間、誰にも見送られること無くたった独りで息を引き取ったのだろう。何故だ……一体何故こんなことに……。丸2日間飯を与えるのがいけなかったのだろうか？ひょっとしたら排泄物の処理を怠ったのが良くなかったのかもしれない。幽体へとその形を変えた彼女は、画面から恨めしげに私をぢつと見つめていた。

4日間。彼女と共に歩んだ育成の日々はあまりにも短く、私はただ彼女の食欲と睡眠欲を満たしながら下水処理を施す程度のことしかできなかった。授業の合間にぬってえさボタンを連打した日々。ごきげんパラメータを上昇させる為ミニゲームを繰り返した日々。実は思い出と呼べるような思い出もそれほど残らなかった日々。唐突に訪れた彼女の死は、正直なところ私にとって割とどうでもよかった。こうして私がメスっちと過ごした短い冬は終わりを告げた。

待て、みみずのひものよ。本当にこれで終わって良いとでも思っているのか。何よりもまだ私はたまごっち最大の機能、ブリード（たまごっち同士がくんづほぐれつすること）を観察していないのだぞ。生殖活動を行わない人間ならまだしも、生殖活動を行わないとたまごっちに一体何の価値があるというのだろうか。しかもたまごっちの世代交代を行うことによってTMP（たまごっちポイント）が上昇し、たまごっちの進化系統が変化するというのだから、これを試さずしてたまごっちライフを終えるなど愚の骨頂ここ極まる。

しかし皆様お察しの通り、たまごっちにおける生殖活動は「オスっち（別売り）」と「メスっち」が準備されることによって成立する。つまり私がブリードをする為には、オスっちを現役で育てているなどという天然記念物並に希少な小学生女子に頼み込んで、「おじさんのメスっちと君のオスっちをブリードして楽しい一夜を明かさないかいハアハア？」とお茶でも誘うかのような爽やかさで口説く必要性が生じるのだ。恐らく通報までは秒読みだろう。

まあ落ち着け、みみずのひものよ。別にオスっちも自分で育てればよいだけの話じゃないか。私はamaz○nの検索バーに「オスっち」と叩き込んだ。中古1600円相当、あっ……（察し）。元々このメスっちがWORD編集部室ロッカーから発掘されたタダ同然の代物であることを考えると、オスっちの為に支払う1600円という追加料金は私の購入をためらわせるのに十分な額だった。ぶっちゃけたまごっちなどという旧世代生物の育成に1600円も



メスっちは死後おばけっちにジョブチェンジ

つぎ込むなら、最新のたまごっちを買えばいいんじゃないかという身も蓋も無い残酷なご意見も聞こえてきそうである。

もはやこれまで。そもそもオスっちメスっちが発売から 15 年も経過していることを考えると、既に絶滅の危機に瀕しているのも致し方ない。たまごっちも今やカードとか通信で遊ぶ時代<sup>\*11</sup> であることを考慮すると、育成機能をメインに据えているような旧世代たまごっちなどもはや過去の産物に過ぎないのである。ということで彼女は生涯喪女として生きていくことを定められていたのだと、私はポジティブに考えることにした。

そんなメスっちにお見合いの話が舞い込んだのは、まさに奇跡としか思えないほどの偶然だろう。話によれば本メスっちが発掘された際同時にオスっちも発見され、彼は現在 WORD 編集部員ふあいさんの管理下に置かれているとのこと。恐らく先駆者はたまごっちという種族の絶滅を想定して、男女つがいのたまごっちを同じ場所に保存したのだろう。偉いぞ先駆者。

さて、こうなると私もふあいさんのオスっちを惹きつけるような、フェロモンむんむんのアダルトっち（♀）をブリードの為に準備しなくてはならない。こうして私は昔の女を切り捨て、新しいメスっちとの共同生活を始めたのである。

## 2.オスっちメスっちとは

オスっちメスっちシリーズは 1997 年に発売されたたまごっちシリーズの 1 つで、シリーズ 8 番目にあたる。前述の通り本作を特徴付ける最大の機能は、オスっちメスっち両個体をアダルトっちまで育て上げてから本体上部にある端子を接合することで行う「ブリード」だ。ここで「ブリードって何？」などという澄んだ目をした小学生並の無粋な質問を投げかけられる読者様がいらっしゃるかもしれないが、私は心を鬼にして次に進めたいと思う。

さて、ブリード後メスっちは雌雄のベビーっちを 1 匹ずつ生み、オスっちメスっちがそれぞれ同性の方を 1 匹ずつを引き取って実家に帰宅する。行為後即出産即電撃離婚という芸能界も顔負けのドライなたまごっちの関係には、思わずチェリーボーイの私もたじたじする他に無い。そのうちに親のたまごっちの方はたまごっち星へと

最近のたまごっちはカラー



ブリード中の様子

\*11 12月13日から稼働を始めたアーケードゲーム「たまごっちリズム」のこと。ジャンルはリズムゲーム。4枚のたまごっちカードを組み合わせてパーティを構成し、芸術作品の制作を目指し奮闘する。ちなみに子機のたまごっちピースと通信することで、育成中のたまごっちを呼び寄せることも可能。

帰還するので、プレイヤーは新ベビーたちを育成しアダルトたちまで育て上げるというわけだ。

それにしても自分のたまごたちと他人のたまごたち間で生殖活動を行うということについて、私のみならず読者の皆様までもが妙な生々しさを感じてしまうかもしれない。が、公式から既にそういったことを意識しているようなので<sup>2</sup> どうやらたまごたち界隈における性事情は人間界のそれよりもあけっぴろげのように見える。

その他は従来のたまごたちシステムに準拠する。お腹が空いたらごはんを与え、飴玉や旗振りゲームなどでご機嫌を取り、排泄物は水洗で流し、頭の上にドクロマークが浮かんだら病気状態なので治療する。小学生でも遊べる実に簡単なゲームだ。

それでは本紙みみずのひもの編集部員がメスっち及びオスっちに挑んだ約1ヶ月に渡る育成の記録をここでご紹介しよう。

### 3.育成観察日記

#### ・12月24日

前メスっちが死んだ。クリスマス一緒に過ごそうとしていた彼女が死んだ。とうとう私も独りぼっち。まあア○ガミもあるしそこまで気にはならなかった。

#### ・12月25日

前日までは幽体として形をとどめていた前メスっちは、朝どろどろに溶けた状態で発見された。その状態からマニュアルに従ってボタン操作をすると、新しい卵が登場する。2、3分後に無事孵化。こうして私のたまごたちライフは新たな始まりを向かえた。

私は愛着を持ちやすくなるよう、彼女に「梨穂子」という名前を付けた。いや、ア○ガミは特に関係無い。さすがに育成初日だけあって、私自身まだ意識も高かった。授業中の合間にぬつて梨穂子にえさをやることも忘れない。

1才

23グラム

---

\*2 「俺ブリードしちゃった……」「誰と?」「……大人の女の人の(ドヤ)」というセクシーな発売当時のCM。

気になる方は調べて、どうぞ。

## ・12月26日

もう朝の10時になろうかというのにまだ寝ているあたりは、さすがWORD編集部原産たまごっつの名に恥じない。梨穂子は起きるなり食物を要求してきた。食い意地の張ったところはまるで本家の梨穂子<sup>\*3</sup>を感じさせる。

12時7分脱糞。



睡眠中の梨穂子

食欲、ご機嫌パラメータの減少があまりにも早いので、とりあえず死なないように食欲だけでも最低限満たすように心がける。夜ちょっと世話を忘れていたら、梨穂子は排泄物を枕元に携えながら深い眠りに落ちてしまった。

その眠りが最後になるとも知らずに……

2才

32グラム

## ・12月27日

悲劇は真夜中に起こった。なんと某WORD編集部員Itosugi氏がいきなりリセットボタンを押したことにより、私の可愛い可愛い梨穂子が突如消失したのだ。まさか育てている生物をリセットで吹き飛ばされるなどとは想像もしていなかつたが、WORD編集部という過酷な環境ではそういったことが日常茶飯事なのだと断っておこう。加えてリセットという特殊な埋葬方法では、梨穂子との最期の時間どころか余韻に浸る暇すらも与えられない。画面には卵がただ1つだけ残され、梨穂子との思い出らしき痕跡は跡形も無く消え去っていた。

こうして私と梨穂子の僅か48時間に渡る愛の共同生活は、唐突に終わりを迎えた。私は梨穂子の死による悲しみと喪失感から、思わず10時間ほどア○ガミをプレイして現実逃避を図ってしまった。

享年 3才

だがいつまでも感傷に浸っている場合では無い。私はブリードという最終目的に達する為、いかなる犠牲を払ってでもメスっちをアダルトっちまで育て上げなくてはならないのだ。私は直ぐに新しいメスっちの育成へと取りかかる。

11時52分脱糞。

実は「おかし」を食べることで「ごきげん」が直るという、このゲームの根本的なシステムに今日気づく。お

\*3 梨穂子ちゃんとはア○ガミに登場するヒロインの一人で、幼馴染みぼっちやり系女子。甘いものを作ることも食べるのも大好き。ゲーム中有名なシーンの1つとして、嫉妬した梨穂子をなだめるため「梨穂子はかわいいなあ!」「梨穂子はかわいいなあ!!」「梨穂子はかわいいなあ!!!」の3つから好きな選択肢を選ぶというものがある。筆者は何となくさよ教のこよりルートを思い出した。分かるかしら。

かしで満腹度が満たされないのはそういう理由だったのか。

彼女の名前については後日決めたいと思う。

1才

59グラム

この体重増加速度は先代2名と比べても圧倒的。おかしの食べ過ぎが原因か？

## ・12月28日

新しい子の名前は「梨穂子二世」にした。

このゲームごきげんパラメータという値が非常に重要なようだ。このパラメータが0になるとメスっちは機嫌を悪くし、どんなに空腹であってもごはんを食べなくなるのだ。食欲よりも機嫌を優先させるあたりが、メスっつの風格の高さを伺わせる。

さて、ごきげんパラメータを上昇させる方法は、「おかしをやる」と「ごきげんアップゲーム」の2つだ。以下にごきげんアップゲームの説明を付随する。

「たまごっちが右か左の旗を上げるので、AボタンかBボタンを押してどっちの旗を出すか選んでね。たまごっちと同じ向きでとまつたら成功！5回勝負で3回以上当たっていれば、ごきげんがアップします。」（ごきげんアップゲーム解説、公式HPより）

だがこのゲーム重要なウェイトを占める割には、無駄なエフェクト表示に時間がかかる、単調でつまらない、効果が薄い<sup>\*4</sup>と三拍子揃っている。一方おかしは病気になりやすかったり体重が増えるというデメリットがある代わり、2秒補給できるので非常にお手軽だ。

今まで梨穂子二世にはさながらドーピングのようにおかしをじゃんじゃん流し込んでいた。だが今日の最大体重が90グラム近くまで上昇したことを考えると、どうしてもおかしのやりすぎ感が否めない。何しろたまごっちの体重が99グラムでカNSTする考えると、ひょっとしなくても梨穂子二世はぼっちやり系女子をとうに通り越しているのでは無いかという疑念が浮かび上がるのも当然だ。これからはごきげんアップゲームを繰り返すことで減量を目指したいと思う。

Twitter上のやりとりの末、ふあいさんからオスっちも引き取ることになった。こちらはボタン電池が死んでいるそうで、準備しなければならない。

梨穂子二世は脱糞し病気状態のまま睡眠をとる。どうやらたまごっちにとっての欲求優先順位は  
睡眠欲 > 機嫌 >>> (越えられない壁) >>> 食欲らしい。

2才

83グラム

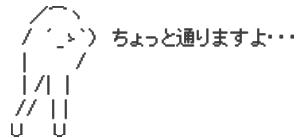
---

<sup>\*4</sup> ごきげんパラメータが最大4つのうちハート1つ分しか上がらず、体重も1グラムしか減量しない。

### ・12月29日

朝起きると梨穂子二世から足が生えていた。この種類は「はわいこっち」というそうだ。お兄さんこの体型は正直気持ち悪いと思います。何となく「ちょっと通りますよ」というAAのことを思い出した。

3才  
90グラム



### ・12月30日

梨穂子二世の体重が99グラムを迎えたことにより、ディスプレイ一面が顔面で埋まる。さすがに乙女なだけあって自身の体型を気にし始めたのだろうか、どんなに空腹であってもごはんの摂取を拒否するようになった。嫌な予感しかしない。

4才  
99グラム



ぱつちやりを通り越し肥満体型と化した

### ・12月31日

案の定梨穂子二世絶食のあまり死ぬ。彼女は残念ながら年を越すことができなかつた。段々とたまごっちの死に対して無感情になっていくみみずのひもの編集部員が、そこにはいた。

享年 5才  
99グラム

※ここで冬休みの為しばらく育成作業は休暇。

### ・1月6日

ついにボタン電池を購入したので、さっそくオスっちにインサート。が……画面に何も表示されてない。ああ、何ということでしょう。僕もオスっちの液晶はほぼ死んでいたのです。画面に対してほぼ水平に視線を傾けることでようやくほんの少しだけ映るその貧相な液晶からは、どうやら黒っぽい生物がうねうねしている程度のことしか観測できなかつた。

となればパラメータを読み取ることもできない。先述の「ごきげんアップゲーム」に至つては、肝心の旗をどちらに振っているかが分からない。ましてや画面からオスっちの機嫌が悪いのか体調が悪いのか脱糞したのかすらも判別できないとなると、私は決められた時間に決められた回数だけボタンを押すというよく分からない単調作業を毎日こなすことになりそうだ。もはやこれを育成ゲームと呼ぶことができるのだろうか心配だ。

ちなみにオスっちとメスっちを接続しブリードを試みたが、メスっちが否定の意思を示した。まだ大人の階段を上るには時期尚早ということだろうか。バ○ダイの心遣いには畏れ入る。

### ・1月7日

オスっちごはんをやり忘れて死ぬ。存命中画面上の様子が良く分からぬくせに、死後オスっちの画面には墓場の姿がくっきりと映っていた。

### ・1月8日

後を追うようにしてメスっちも死ぬ。彼女は名前を付ける前に死んでしまったことが悔やまれる。

### ・1月9日

オスっちメスっちを同じタイミングで復活させる。俺達の育成はこれからだ！



### ・1月10日

たまごっち？ そんなことより中間試験だ！

### ・1月11日

中間試験にうつつを抜かしていたら、いつの間にかまたメスっちが死んでいた。

### ・1月12日

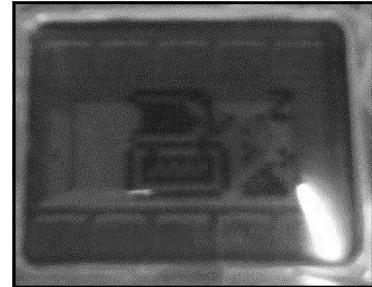
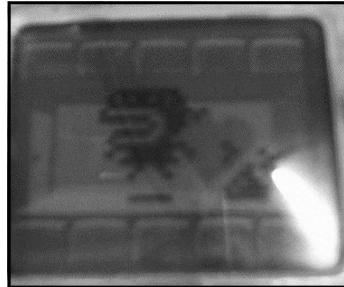
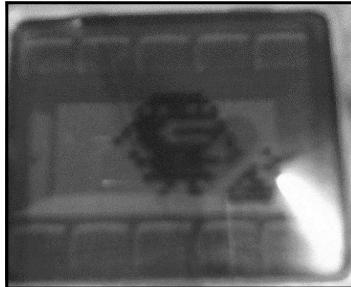
一縷の望みをかけて WORD 編集部員ふあぼきやつする氏にオスっちの修理を依頼するも、あえなく撃沈。

主治医からのコメント「無理」。

こうしてたまごっちを殺したり生き返らせたりを繰り返していくが、なかなか件のアダルトっちまで成長させることができないみみずのひもの編集部員であった。

そして時は流れ、来る 1 月 19 日。原稿締め切りまで後 2 日というところで、ついに私は梨穂子六世ことアダル

トっちの姿を捉えることに成功した。それではその全貌を明らかにしよう。



4 才

87 グラム

なんだこれ。

公式 HP によると彼女の名前は「ガンコっち」というそうだ。彼女は育成の合間をぬって日焼けサロンにでも通いつめたのだろうか、その肉体は黒一色に染め上げられていた。ひょっとしたらガンコというのは汚れのことかも知れないが。

とにかくたまごっちとしては致命的なほど可愛くない。何しろ見た目はヤマンバ、笑うと不細工寝顔も不細工、いびきが五月蠅い、おまけに夜活動し昼に寝るという昼夜逆転生活を送るものだから、父親の立場からするとこれほど将来が不安になるような娘も他にいないだろう。

ちなみにガンコっちで検索したところ、落語家の三遊亭ぬう生氏が本記事の私と同じようなことをして、同様に可愛くないという意見を述べていた<sup>5</sup>。人間考えることは皆一緒のようである。

だが何はともあれ私はアダルトっち（♀）の生成に成功した。これで後アダルトっち（♂）さえ用意できれば念願のブリード達成。たまごっち達もめでたく大人の階段をステップアップするという算段だ。もう締め切りには間に合わないけれども、オスっちの成長が待ち遠しい。

## ・1月21日

昼下がりの午後、ピーッピーッという断続的な電子音が WORD 編集部内を響かせた。携帯か時計のアラームか、それとも電池切れの知らせか？ そのうちにどうやらその音の発生源は、私の上着ポケットからだということが分かった。

嫌な予感がする。私は恐る恐るポケットに手を差し込み、音の発生源でもあるメスっち本体を取り出して、思わず絶句した。画面に映されていたのは、苦悶の表情を浮かべるガンコっちの姿だった。その頭上にはドクロマークが浮かび、傍らには卵らしきものが添えられている。

---

\*5 三遊亭ぬう生氏のブログ

<http://nuushou.blog99.fc2.com/blog-entry-299.html>

「たまごちは今際の際に卵を産み落とす」

そんなネット上でのたまごっち解説が脳裏を過ぎた。もはやガンコっちの死が秒読みまで迫っているのは一目瞭然だ。またか、また俺はメスっちのことを……。

いや、まだ間に合う。A ボタンを 6 回、B ボタンを 5 回連打すればメスっちへの治療が実行され、直に良くなるはずだ。

周期的に繰り返される電子音。

コンマ数秒の戦い。

私が A ボタンに手を掛けた刹那——

「ピーッ」

あつ。

享年 6 才

94 グラム

こうして私の育成日記はブリードすること無く終わりを迎えた。この日記が終わったとしても、私は育成のループを繰り返すことになるだろう。いつか彼女がブリードするその日まで……。

最後に読者の皆様へ一言。

お客様の中でオスっち育成現役プレイヤーの方がいらっしゃいましたら、是非 WORD 編集部まで遊びに来てブリード、しよう(直球)！

## WORD 編集部への誘い

文 編集部 吉村 優

我々WORD 編集部は情報科学類の公式団体であり、情報科学類誌「WORD」の発行をしています。これは筑波大学内では3つしか存在しない学類誌のうちの一つです。WORDはコンピュータに関するのみならず、様々なことを取り上げる総合的な雑誌であり、年に数度、主に第三エリア3A、3B、3C棟にて配布されています。

WORD 編集部にはこんな部員がいます。例えば、

- ACの中のAC ranha
- バイリンガルなスーパーハカー zer0day
- 漆黒の堕天使 はろべり
- 定理証明とPOSTSCRIPT pi8027
- 単位爆散 いおりん

他にも多くの個性的な編集部員がいます。また部員の大半は情報科学類生ですが、院生なども存在しています。

WORDは我々が真心を込めて執筆、編集、印刷、製本しています。といっても編集部における拘束時間はほとんどなく、週一回の編集会議、年に数度の赤入れ・製本作業といった程度です。なので、他のサークルとの掛け持ちは何の問題もありません。実際、他のサークルと掛け持ちしている編集部員もたくさんいます。

以下の項目に当てはまる人は是非学生ラウンジ隣の編集部室（3C212）へ。常に開いているので、いつでも見学に来てください。

- AC入試で入学した人
- それ以前にACな人
- ネットワーク管理者を経験したことがある人
- 印刷や製版に興味がある人

また、現在WORDは一太郎とLaTeXを使って組版をしておりますので、それらの知識に詳しい人材を募集しています。

- LaTeX2 $\varepsilon$ に詳しい人
- 特に、XeLaTeXにおける縦書き組版の経験がある人
- 「一太郎 承」に詳しい人
- 特に、SuperPlayRiteを用いたマクロ作成の経験がある人
- JustSystemsで働いたことのある人
- 親など近しい人がJustSystemsで働いている人

その他質問がある方は、word@conis.tsukuba.ac.jpまでメールしてください。



# 情報科学類誌

# WORD

From College of Information Science

60秒以内に WORD を用意出来なかったら WORD 無料券をプレゼントします号

発行者

情報科学類長

編集長

吉村優

制作・編集

筑波大学情報学群  
情報科学類 WORD 編集部  
(第三エリア C 棟212号室)

2013年2月15日 初版第一刷発行

(512部)