電気通信大学MMA

百萬石

弐阡壱拾弐年　春号

部長挨拶

百萬石をお手に取って頂きありがとうございます。

MMA(Microcomputer Making Association)[*1]は、コンピュータに関わる様々なことに取り組んでいるサークルです。サーバ構築・管理[*2]やプログラミング[*3]、ハードウェア製作、ガジェット収集など、部員の活動は多岐に渡ります。それら活動の一端を年に二回、冊子の形で報告しているのがこの百萬石です。[*4]

詳しくを語るのは各記事に任せるとして、この冊子を読まれた後にさらなる興味を持たれた方は、ぜひサークル棟 2F 左手奥にある部室へお越しください。コンピュータに関心がある方を歓迎します。これまで Unix に触れたことが無くても、これから自ら手を動かして触れてみる姿勢があれば大丈夫です。

お待ちしております。

2012 年 4 月
部長　clear

[*1] Mixed Martial Arts ではありません、念のため
[*2] 部内用のサーバ複数台の他、サークル棟ネットワークの管理も行っています
[*3] 5 月にポーランドで行われる ACM 国際大学対抗プログラミングコンテスト (ACM-ICPC) 世界大会に MMA の部員で構成されたチームが出場します
[*4] バックナンバーはこちら: `http://wiki.mma.club.uec.ac.jp/Booklet/`

目次

工房のススメ

KHe7

khe7@mma.club.uec.ac.jp

はじめに

新入生向け、各種工房のススメ

1　前提

この冊子を手にしているのは電通大の新入生、と仮定して話をしています。

とはいえ、I 科の選択専門科目である情報工学工房と電子工学工房を履修していない学生には有益かもしれないので目を通すのもいいでしょう。

2　はじめに

電気通信大学へのご入学おめでとうございます。本学は、地名の入っていない唯一の国立大学であり、TDU[*1]、F ラン、私立と勘違いされるなど少し寂しい立場にあります。

このような名前ではありますが、2 年前の私のように情報系の大学という認識で入学してきた人も多いのではないでしょうか。ですが、誠に残念なことに、C 言語を使ったプログラミングでさえ、後期から始まります。

前期はコンピューターリテラシーという微妙な科目があります。[*2]そして、後期は基礎プログラミング通論および演習という科目で、C 言語の変数・定数から、配列、入出力、関数、ポインタと一気に扱ってしまい、初学者には速くて難しい[*3]上に、演習課題の量が多くて面倒というこれまた残念な仕様となっています。更に、I 科では、2 年前期にプログラミングの講義が無くなるのに、後期はポインタをフル活用する演習が登場するなどそれはもう残念な出来のカリキュラムとなっています。[*4]

とはいえ、この記事はそんなカリキュラムを放置して、電子工学工房と情報工学工房という I 科のボーナス科目を紹介していきます。

3　工房の存在

という訳で本題に入りますが、主に I 科の人をターゲットとしています。[*5]

電通大は“工房”という形で、授業や課外活動として自主的にいろいろなことを出来る場を整備しています。その一環として I 科には、1 年から 4 年までいつでも取ることができる、電子工学工房と情報工学工房という 2 つの科目があります。

それぞれ、1 年間にわたって行われる通年の科目ですが、2 単位にしかなりません。それでもあえてこの科目を紹介し、推奨していくには 2 つの理由があります。

ひとつ、1 年前期から実習形式でそれなりに本格的な内容に触れることができる。

ふたつ、この 2 つの科目は、選択専門科目という分類に入っていて、履修して単位を取っておくことで 3,4 年次の難しい数学や物理、情報の科目の必要単位数を減らすことができる。

という点です。

内容については次章に委ねます。また、試験前に普通に勉強していれば、正直単位なんて落としません。しかし、なぜか電通大の学生はやたら単位を落としていくので 4 年卒業率が 7 割を切っていたりします。もし、3 年次に 2 年の再履に追われているとしたら、こうやって予め必要単位数を減らしていくことには一定の意味があるでしょう。

[*1] 東京電機大学。秋葉原にある (むしろあった？)

[*2] コンピューター入門というか……なんというか……内容も残念。ただ、LaTeX の環境を言われるままに整備しておくとよいです。

[*3] 知ってると意味なし

[*4] 学科再編は改悪…

[*5] 他学科でも取れるかもしれないけど抽選あるし…でも、説明会は行ってみると良いかと。

4　工房の内容

私は 1 年次に情報工学工房を、2 年次に電子工学工房を履修しました。同時履修が推奨されていなかったためですが、1 年次に両方とったという猛者もいたようです。

工房は 4 月の終わりから授業が始まります。内容についての説明会と履修の希望調査は 4 月中旬、新入生合宿研修で予告があると思いますのでその場でのアナウンスと掲示に気をつけて下さい。

4.1　情報工学工房 (2010 年度)

情報工学工房では、C 言語や Java、OCaml、MATLAB といった様々な言語から 1 つを選んで言語やテーマに別れて学習を行います。前期はそれぞれの言語について文法や特徴について実際に JED[*6]や情基[*7]等でプログラミングしながら学習します。後期は「ICPC の過去問を解く」や「GPU プログラミング」といったテーマを選択してテーマごとに別れてそれぞれ別々に様々な学習をします。

私は前半は Java、後半は最中限を選択しました。担当は寺田実准教授です。

前期は、前述のとおり、Java を用いたプログラミングを行います。先生の説明を聞いて実際に eclipse 上でプログラムを組んで走らせて提出してコメントを貰っての繰り返しです。

Java は手続き型のプログラミング言語なので C 言語とほとんど変わりませんが、オブジェクト指向について、思想とイメージの説明もあるので結構有意義でした。

後期は最中限というカードゲームの AI を作っていました。最中限は、簡単にまとめると 3 人でトランプのカードを出しあい真ん中のカードを出した人がその数の点数を貰う。一定回数繰り返して得点の合計が真ん中の人が勝つというゲームです。

学生がプログラムする部分は、なにを出しますかと聞かれてこれを出しますという部分だけなので AI と言っても構える必要はありません。どんな戦略を立ててどのように出していくと強いのかを考えて試行錯誤するのが目的の授業です。とりあえず楽しくやっておけば損にはなりません。

参考までに講義のページ[*8]の URL を下に載せておきます。　情報工学工房は学部再編に合わせて 2010 年度に新しくできた科目だったので今では少し内容に変化が生じているかもしれません。

4.2　電子工学工房 (2011 年度)

電子工学工房では、情報工学工房と違って比較的プログラミングよりも電気回路の方に重きが置かれています。

前期は、「多様な電子回路素子とその動作」、「いろいろな電子部品の特性の計測」、「小型コンピュータの動作」という 3 つのテーマをそれぞれ 4 回の時間を使って行いました。

それぞれのテキストは講義のページに残っているので見てもらえば内容はわかると思います。実際に組んでそれぞれの性質、特性について考えるのは素直に楽しいと思えました。

その中でも 3 番目に扱った小型コンピューターの動作では、SSC[*9]という 8bit コンピューターを使って簡単なプログラムを組んでいきます。2 年前期の計算機通論を勉強していると復習になりますが 1 年の時点でも十分に理解できる内容と言えます。簡単なコンピューターの動作原理を確認するのに適した機材です。また、1,2 コースでは 3 年次実験に使うということなのでここにも工房の性格が現れています。どのセンサーを使うか、センサーの入力をどう処理するか、そしてどうプログラムに落としこむか。LabVIEW という言語に戸惑いつつも楽しくやれたのでこちらもおすすめしておきます。講義のページの URL は次のとおりです。

[*6] I 科の計算機室
[*7] CL で使う計算機室
[*8] http://pr.ice.uec.ac.jp/kobo2010/
[*9] Slow Scan Computer

http://space.ice.uec.ac.jp/edu/ugrad/eew/Welcome.html

5　まとめ

電子工学工房、情報工学工房という講義は、前期に基本的な事項について学習して、後期にひとつのテーマについて試行錯誤を繰り返しながら様々なことを学び取るという自主的な学びの場の提供と言えるでしょう。

また、情報工学工房については 2 度目以降の参加[*10]もテーマによっては歓迎されています。

他にも、ロボメカ工房やピクトラボ、ものつくりセンターなど“工房”としての場の提供は様々な形で行われています。院生、研究室所属など利用に条件がある場合もありますが、積極的に使えるものは使う方針でいると学生生活がより有意義になるのではないかと思います。

もちろん、講義やサークル活動、趣味、バイトなどに打ち込むのもよいでしょう。他学科の科目や、他大学に単位互換制度で突撃するのも面白いかもしれません。何事も積極的に 4 年間を楽しんでください。

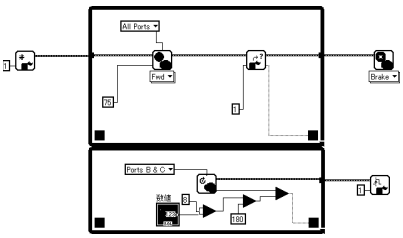

図 1　LabVIEW のプログラム例

[*10] 単位は出ません。というか、出ないはずなのですが、2012.03.13 現在でなぜか出ている不思議なケースが存在。おそらく取り消される。

たのしい Twitter クライアント作成

kakakaya

kakakaya@mma.club.uec.ac.jp

はじめに

プログラミングは授業で C 言語をやった程度、python の知識はほぼ 0、まともに動くアプリケーションの作成経験は完全に 0 ながら、一念発起して CUI がメインの Twitter クライアントを作ってみました。[*1]ちなみに、クライアント名は、いもうと大陸です。

恋する妹は、とどまるところを知らないっ！

1 作ろうと思った動機

- 周りの流行
  これを作った時点で、自分を除くアクティブな MMA1 年のうち 7 割以上が何らかの自作 Twitter クライアントを作っていた。[*2]自分も流行に乗り遅れてはいけないと感じ、何らかのクライアントを作ってみたいと思うようになっていた。
- 放置していたクライアント登録
  上記の理由もあり、某オープンソースのクライアントを弄り、クライアント認証部分だけ自分で登録したものに変更して、クライアント名、認証時の説明、及びリンク先だけ違う自作のクライアントっぽく装おうと考えた。しかし、自分の知識不足か、はたまた愛が足りなかったのか、認証部分を変更したところコンパイルすることが出来なかった。そのため、頑張って考えたクライアント名だけ残り、なんとも無念であった。
- 何か動くものを作ってみたい、という願望
  春休み中、なかなかに暇な生活を送っていた。そのため、せっかく授業で C を覚えたのだからと思い、AOJ や codeforces のようなプログラミングのコンテストに手を出して痛い目にあったりして遊んでいた。しかし、自分の環境で動くのに提出すると上手く動かなくて WA、というのに辟易して、また、数字の入出力をずっと続けていくのも余り建設的でないと感じ、何かもっと視覚的で生産的な物を作りたいと考えるようになってきた。
- もっと軽量で思い通りの Twitter クライアントが欲しい
  Twitter に関しては比較的古参だったりするので[*3]、色々なクライアントを試してきたのだが、やはり GUI 依存だと起動していない場合は投稿まで時間がかかる。一方、CUI 系については疎いものの、引数による投稿機能のあるものは見つからなかった。それに、どうせ CUI 系なら、自分で作り、さらに公開することも出来るのではないか？と考えた。

上記の理由により、いっそ自分で Twitter クライアントを作ってしまおう！と一念発起、作ることに決めた。また、使う言語を Python にしたのは、入学当初もなにかやってみようと思い Python 関連の書籍を読んだ

[*1] 「一週間で作る！」と、記事のタイトルにしようと思ったのが締め切りの一週間前だったものの、まだ人に見せられるようなレベルじゃないのでお茶を濁しました。

[*2] 嘘は付いていない。

[*3] 登録日：2007 年 09 月 07 日 01 時 58 分 03 秒 (JST)

ものの、当時は BASIC 以外全く知らなかったため手も足も出なかった……ので、リベンジをしようと思ったからである。

2　作ってみよう

簡単な作り方の説明。

2.1　必要なもの

- Twitter アカウント
- Python2.7
- eclipse+PyDev
- tweepy
- オリジナルの Twitter クライアントに関するアイデア
- Google

2.2　基礎部分の作り方

(1) 環境作成
Python とか eclipse とか tweepy とか。環境によっては他にもいくつか必要かもしれない。eclipse は無くても良いけど有ると便利。tweepy のインストールには依存するものがいくつかあるため、先にそっちからやらなくてはならない。要ドキュメント参照。

(2) クライアント登録
`https://dev.twitter.com/apps` でクライアント登録をし、Consumer key と Consumer secret を貰う。認証機能を付けないなら、Access token と Access token secret も貰っておく。

(3) 実装

```
import tweepy

auth = tweepy.OAuthHandler('consumer key','consumer secret')
auth.set_access_token('access key','access secret')
api = tweepy.API(auth_handler=auth)
api.update_status("""I'm at Sister Continent
 with 100 younger sisters! #sister_continent""")
```

クォーテーションで囲まれた部分に先程登録したものを代入し、test.py とかみたいな感じで保存して実行するだけで投稿は出来る。言うなれば基礎部分である。
ちなみに、ググると一杯同じようなコードが出てくるが、これは Python の言語仕様に依るところが大きい。Perl などでは"TIMTOWTDI"という「一つのことをするのに沢山のやり方がある」という思想があり、人によりコードは多種多様であるが、Python はコードの読みやすさを重視したコードであるため、綺麗な

コードを作ろうとすると大体同じようなものになる。いわゆる、たったひとつの冴えたやりかた、である。

2.3 機能追加

これだけでは、ひとつの内容を投稿するのにわざわざ書き換えなくてはいけない。また、投稿するだけのクライアントというのも非常に味気ないので、ここからいくつかの機能を付けてみた。以下に具体例を記す。[*4]この辺が作っていて面白く、創意工夫の意欲が刺激される。

- 複垢対応、複クラ対応
  既存のクライアントにも、複数のアカウントを制御出来るものは多く存在するが、恐らく複数のクライアントに対応したクライアントというのは史上初であると思われる。ちなみに、複数のクライアントに対応といっても、某 StS の twicca モードのようなものではなく、ユーザが Twitter に登録したクライアント名を使えるというだけである。[*5]
- ini ファイル読み込み
  python での bool は数値型の 0 などが False ということで、[' 項目']=0/1 で行った。読み込んで辞書に格納する。以下のあぶない機能において、意図しないで投稿されてしまうことを防ぐためのロックや、コンソールへの出力の有無、通常利用するアカウントやクライアントなどを設定する。
- 引数読み込み
  起動時の引数をリストに格納し、一定の内容だったら定型句投稿や、使用するアカウントを決定し、それ以外ならその内容を投稿する。引数が無かったら投稿内容の入力を促す。当初、sys.argv を受け取り直接解析していたものの、argparse という python2.7 から追加されたモジュールを利用することにより、より柔軟な処理を可能にした。
- 各種エラー時の修正促し
- 各種定型句投稿機能
  某大陸系クライアントに搭載されていた文字列＋時刻を投稿する機能を付けてみた。実際、時刻が付いていてもなんら意味はないのだが、本家のリスペクト及び重複投稿でエラーが出るのは面倒なのでそのようにした。
- あぶない機能
  狂気的なクライアントを目指していたので、スクリーンショット投稿、クリップボードの文字列を投稿といった一歩間違えると取り返しが付かない機能を付けようとした。だが、環境ごとに処理を付けるのが面倒なので、断念してしまった。他に、USERNAME 等のような環境変数を投稿する機能とかも考えている。ネタの範疇を超えている気もする。
- kakikomi.txt
  付けようと思って忘れていた。そのうち付ける予定…。

さて、これで Twitter クライアントのようなものが出来上がった。
ちなみに、`http://wiki.mma.club.uec.ac.jp/kakakaya/Product/SisterContinent` で公開予定である。興味があったら試してみて欲しい。

[*4] 本当はコードも付けたかったが、大したものは書いていない上に、未だまともに動くとは言いがたい状態なのでやめておいた。
[*5] 途中までそのような機能を付ける予定は無かった上に、アカウントの認証状態をクライアント毎に保存する必要が生じたため、大幅な仕様変更が必要になり、現在制作中断中。うまい方法を模索中…

3　活用法

コマンドライン上でしか使えないクライアントは果たして便利なのか？ということで、いくつかの活用方法を考えてみた。

- CUI で Twitter を使いたい時に使う。
  多分不向き。TL 表示機能がある CUI クライアントは普通に存在するので、そちらを使うほうが便利。
- 引数指定の出来るキーボードランチャから起動する。
  筆者はこのようにして使っている。何もない状態から 6 打前後、引数指定まで事前に登録しておけば 3 打で即投稿出来る[*6]、というのは便利である。
- ショートカットのプロパティで引数を指定しておき、開くとすぐに投稿出来る。
  一応ネタとしての使い方の提示ではあるものの、需要は案外あるかもしれない。でも、大量に作っておくよりは引数指定で好きなように起動するほうが便利である。
- 珍しいクライアントを使いたい欲を満たす。
- 自己主張をクライアント名で行う。
  正しい使い方。
- いもうと大陸には妹がいっぱいいるんだ…アハハ……
  いもうと大陸に於ける妹とは、豊富な引数指定による半自動投稿機能の事を指します[*7]。認証画面などに書いてあるとおり、最終的に 100 種類の半自動投稿方法 + ユーザ任意の文章の投稿が出来るようにしたいと考えています。

4　総括

上記のコードをコードをコピーして自分の欲しい機能を付け加えてみるだけで自分のクライアントが作れます。やってみると案外楽しい上に、特に失敗しなければ一日で完成するかもしれないので、興味が湧いたらやってみるといいと思います。なんか動かない時は、恐らく愛が足りていないので、Google 先生と一緒に頑張ってみるといいでしょう。

[*6] [半角/全角][/][t][(spc)][-][j] で「地震だ！(以下時刻が付加される」というのが投稿される。登録済みなら t 打鍵直後に投稿される。

[*7] リモート（telnet、SSH の使える環境なら一応それらしいことが出来る）と妹を掛けている説もあり。

便利ツール・Valgrind

wataj

wataj@mma.club.uec.ac.jp

はじめに

解放していないメモリはないか、不安でしょうがない。そんなときに役立つのがコレ、「Valgrind」。無料です。

1　はじめに

こんにちは。今回はメモリデバッグツール「Valgrind」(ヴァルグリンド) を紹介します。

Valgrind は CUI ベースのプログラムで、主に実行形式ファイルのメモリ関連の不具合を調べるのに適しています。

2　C プログラミングとメモリ

さて Valgrind の便利さを存分に知るためには、多少 C 言語のプログラミング知識が必要ですが、「メモリを確保して、使い終わったら、それらを解放しなければならない」ことさえ念頭におけば、大丈夫でしょう。確保したメモリをどこかで解放しなければ、いずれはメモリが足りなくなってしまうかもしれないからです。

そこで、作ったプログラムがちゃんとメモリを解放できているのかどうかを調べる道具「Valgrind」の登場です。

3　早速つかってみる

3.1　テスト用のプログラム

まず次のようなソースコードを見てみましょう。

これは（普通は）５バイトのメモリを確保して、すぐに解放するだけのコードです。このプログラムは正常です（free 関数に NULL が入っても大丈夫です)。

good_01.c

```
#include <stdlib.h>

int main(){

  char *a= (char *)malloc(sizeof(char)*5); // メモリを確保

  free(a); // メモリを解放

  return 0;
}
```

3.2 Valgrind の出番だ

まずどうにかしてソースコードファイルからプログラムを作ります（コンパイルと言います)。ここでは gcc 4.4.3 を使用しました。

なおコンパイルコマンドのオプションに「-g」をつけると、後々見やすくなって良いです。

普通はこんなコマンドになります (コマンドなので、% は入力してはいけません)。

```
% gcc -g good_01.c
```

すると、a.out と言う名前のプログラムが生成されます（試しに実行してもいいですが、なにしろメモリを確保してすぐに解放するだけのプログラムなので、何か表示されたり、音が鳴ったり、ゲームが起動するわけではありません。残念でした。私も残念です)。

さてやっと Valgrind の出番です。次のコマンドを打ちましょう (コマンドなので、% は入力してはいけません)。

```
% valgrind ./a.out
```

すると、画面にズラズラと色々表示されます。１５行くらい出ます。もしも 1 行、

```
bash: command not found: valgrind
```

だとか

```
valgrind コマンドが見つかりません
```

とかが出てそれっきりならば、お使いのパソコンには Valgrind は入っていません。UNIX 系 OS なら、頑張って入れることができます。場合によってはとても簡単です。でも詳しくはここでは書きません。

あ、Windows ですか？Windows 版は存在しません。今日から Ubuntu にしません？Ubuntu はいいですよ。でも詳しくはここでは書きません。

閑話休題、以下は Valgrind が入っていた人（もしくは入れた人）が対象です。こんな感じの表示が出たと思います（完全に同じ、ではありません。特に各行の左の数字「7521」は、多分毎回変わります）。

実行結果 01

```
==7521== Memcheck, a memory error detector
==7521== Copyright (C) 2002-2009, and GNU GPL'd, by Julian Seward et al.
==7521== Using Valgrind-3.6.0.SVN-Debian and LibVEX; rerun with -h for copyright
info
==7521== Command: ./a.out
==7521==
==7521==
==7521== HEAP SUMMARY:
==7521==     in use at exit: 0 bytes in 0 blocks
==7521==   total heap usage: 1 allocs, 1 frees, 5 bytes allocated
==7521==
==7521== All heap blocks were freed -- no leaks are possible
==7521==
==7521== For counts of detected and suppressed errors, rerun with: -v
==7521== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 12 from 7)
```

これが、何もエラーが見つからなかったときの、Valgrind の表示です。最下行に「ERROR SUMMARY: 0 errors」と書いてありますね。

では次に、確保したはいいが解放せずにそのまま終わる、というプログラム bad_02.c を Valgrind にかけてみましょう。

bad_02.c

```
#include <stdlib.h>

int main(){

  char *a= (char *)malloc(sizeof(char)*5); // メモリを確保

  return 0; // そのまま終わり
}
```

結果は次のようになります。

実行結果 02-1

```
==7619== Memcheck, a memory error detector
==7619== Copyright (C) 2002-2009, and GNU GPL'd, by Julian Seward et al.
==7619== Using Valgrind-3.6.0.SVN-Debian and LibVEX; rerun with -h for copyright
info
==7619== Command: ./a.out
==7619==
==7619==
==7619== HEAP SUMMARY:
==7619==     in use at exit: 5 bytes in 1 blocks
==7619==   total heap usage: 1 allocs, 0 frees, 5 bytes allocated
==7619==
==7619== LEAK SUMMARY:
==7619==    definitely lost: 5 bytes in 1 blocks
==7619==    indirectly lost: 0 bytes in 0 blocks
==7619==      possibly lost: 0 bytes in 0 blocks
==7619==    still reachable: 0 bytes in 0 blocks
==7619==         suppressed: 0 bytes in 0 blocks
==7619== Rerun with --leak-check=full to see details of leaked memory
==7619==
==7619== For counts of detected and suppressed errors, rerun with: -v
==7619== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 12 from 7)
```

よく見てください。真ん中あたりに「LEAK SUMMARY:」が出ていますね！このプログラムはマズイということをちゃんと Valgrind は検出できました。

「definitely lost: 5 bytes in 1 blocks」、つまり 5 バイト確保したのが解放されずに終わりました、ということが書かれています（メモリリークです）。これだけでも便利なのですが、さっきの「-g」オプションを活用しましょう。この実行結果 02-1 の下の方に、「Rerun with --leak-check=full to see details of leaked memory」とあります。書いてあるとおりに、「valgrind --leak-check=full ./a.out」としてみましょう。すると、次のような実行結果 02-2 が得られます。

実行結果 02-2

```
==7666== Memcheck, a memory error detector
==7666== Copyright (C) 2002-2009, and GNU GPL'd, by Julian Seward et al.
==7666== Using Valgrind-3.6.0.SVN-Debian and LibVEX; rerun with -h for copyright
info
==7666== Command: ./a.out
==7666==
==7666==
==7666== HEAP SUMMARY:
==7666==     in use at exit: 5 bytes in 1 blocks
==7666==   total heap usage: 1 allocs, 0 frees, 5 bytes allocated
==7666==
==7666== 5 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7666==    at 0x4024F20: malloc (vg_replace_malloc.c:236)
==7666==    by 0x80483F8: main (bad_02.c:5)
==7666==
==7666== LEAK SUMMARY:
==7666==    definitely lost: 5 bytes in 1 blocks
==7666==    indirectly lost: 0 bytes in 0 blocks
==7666==      possibly lost: 0 bytes in 0 blocks
==7666==    still reachable: 0 bytes in 0 blocks
==7666==         suppressed: 0 bytes in 0 blocks
==7666==
==7666== For counts of detected and suppressed errors, rerun with: -v
==7666== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 12 from 7)
```

真ん中あたりに、「プログラムの中のどの地点で確保されたメモリが、解放されずに終わったか」が書かれています。これによると、bad_02.c の 5 行目です。確かに、メモリを確保した場所です。ここで確保したメモリが、解放されずに終わりました。それを探知できたのです。これが Valgrind のすごいところです。

4 Valgrind にもできないことはある

Valgrind は、不適切な添字をもつ配列要素へのアクセス（確保した配列範囲を越えたアクセス）を検出できません。残念でした。私も残念です。

5 おわりに

Valgrind は万能ではありませんが、それでもなお便利です。ここには書ききれなかった機能も、沢山あります。有名ですから既に使いこなしている人もいるでしょうが、もしこの記事を期に Valgrind を知って頂けたら幸いです。以上で記事を終わります。ありがとうございました。

nginx を用いた Web サーバ最適化

alstamber

alstamber@mma.club.uec.ac.jp

はじめに

Web サーバといえば、数あるサーバの種類の中でもおそらく最もメジャーなものであり、構築経験のある人も多いことだろうと思う。サーバを運営する上で可用性 (availability) を高めることは重要な課題である。今回は nginx と呼ばれるものを使って、高負荷な状態にも耐えられるサーバ構築を実践してみることにした。

1 nginx とは

1.1 概要

nginx[*1]はロシアで開発されている Web サーバである。つまり Apache などと同類のものである。現在そのシェアは Apache と IIS についで 3 番目だという報告もある。シェアはそれらのソフトウェアに及ばないが、有名なサイトでの採用例は多い。日本で知られているサイトであれば FC2 動画や favstar.fm などが採用しているようである。nginx は Apache などに比べて高負荷の環境に強いと言われており、またメモリの消費量も抑えられるという利点がある。欠点を挙げるとすれば、Apache などで築いてきたコンフィグファイルなどの資産を活用できないことだろう。

nginx の大きな特徴は、Apache などとリクエストの処理方法が違うことであろう。Apache などの従来の Web サーバはリクエストに対して応答するプロセスが一度に複数の HTTP リクエストに対して応答することができない。そのため複数の HTTP リクエストを受けた際には先行するリクエストの処理を待つか、新しい応答用のプロセスを立ち上げるかをしなければならない。しかし、プロセスを立ち上げることが出来る数には当然限界がありその限界を超えると Web サーバはすべての HTTP リクエストを処理することができなくなり途端にその性能を落としてしまう。この性能の低下は特に Wordpress で構築したサイトなどバックに重いプログラムが付随しているサイトにおいて顕著になってしまう。

それに対して nginx は「イベントループ」と呼ばれる手法を用いて非同期処理を行うことで、シングルスレッドで複数の HTTP リクエストを同時に処理することが可能になっている。Web サーバに何らかのイベントが発生した時そのイベントはイベントクエリと呼ばれる場所に登録され、イベントに対応したイベントハンドラが実行されることになる。イベントクエリには同時にコールバック関数と呼ばれるものが登録され、イベントハンドラの実行結果はコールバック関数を実行することで返されることになる。これによってマルチスレッド化しなくても並列して処理を行うことが可能となっているのである。マルチスレッド化しないので、Apache などに比べてメモリ使用量を抑えることも可能である。

1.2 リバースプロキシ

nginx の利点としてリバースプロキシを構築するのが簡単であることが挙げられる。例えば Wordpress のような PHP と MySQL を組み合わせたような HTML の構築に果てしない時間がかかるようなシステムを使っていると、HTTP リクエストへの応答性能をいくら上げても可用性の向上には限界がある。そこでリバースプロキシをはさみ、リバースプロキシに HTML をキャッシュすることを考える。こうすることで重

[*1] えんじんくす と読む。

い処理である PHP での処理や MySQL での処理を減らすことができ、結果として応答性能を向上させることができる。

2 試してみる

2.1 環境

今回 nginx の導入を行ったのは次のような構成のサーバである。

- さくらの VPS512
- CPU……仮想 2Core
- メモリ……512MB
- HDD……20GB
- ネットワーク……共有 100Mbps
- OS……Debian Squeeze

2.2 nginx のインストール

nginx の導入を行う。

```
% sudo aptitude install nginx-full
```

2.3 PHP のインストール

```
% sudo aptitude install php5-common php5-cli php5-cgi php5-fpm
```

インストールが終わったら php5-fpm を起動しておく。

```
% sudo /etc/init.d/php5-fpm start
```

2.4 MySQL のインストールと設定

```
% sudo aptitude install mysql-server
```

設定ファイルを書き換え、UTF-8 をデフォルトの文字コードとするようにする。書き換えた部分だけを抜粋して示す。設定を反映するために再起動する。

```
[client]
default-character-set=utf8

[mysqld]
character-set-server=utf8

[mysqldump]
default-character-set=utf8
```

2.5 Wordpress のインストール

MySQL クライアントを起動して、Wordpress 用のデータベースを作成しておく。Wordpress をダウンロードし nginx の root ディレクトリに展開する。展開しディレクトリに移動して、コンフィグファイルを書いておく。

2.6 nginx の設定

ここがメインである。nginx の設定は Apache などと違う部分が多いためこの部分に非常に苦労した。Debian の場合、nginx の設定は/etc/nginx/nginx.conf に記述する。すべてを記述するといくら紙面があっても足りないので重要な部分だけを抜き出したいと思う。

```
    proxy_cache_path   /var/cache/nginx levels=1:2 keys_zone=czone:4m
        max_size=50m inactive=120m;
    proxy_temp_path    /var/tmp/nginx;
    proxy_cache_key    "$scheme://$host$request_uri";
    proxy_set_header   Host     $host;
    proxy_set_header   X-Real-IP    $remote_addr;
    proxy_set_header   X-Forwarded-Host   $host;
    proxy_set_header   X-Forwarded-Server $host;
    proxy_set_header   X-Forwarded-For    $proxy_add_x_forwarded_for;
    proxy_buffers      32 16k;
```

デフォルトで用意されている nginx.conf には http{ で始まるブロックが存在する。ここにサーバについての設定を書いていく。上に示したものはそこに記述する内容である。細かい内容は説明しないが、リバースプロキシでキャッシュする際のキャッシュの設定について記述されている。

```
upstream backend {
    ip_hash;
    server localhost:8080;
}

server {
    listen 80;
    server_name example.com;
    root /path/to/wordpress;
    index index.html index.php;

    location /wp-admin/ {
        proxy_pass http://backend;
        break;
    }
    location /wp-login.php {
        proxy_pass http://backend;
        break;
    }
    location / {
                if ($http_cookie ~* "comment_author_[^=]*=(
    [^%]+)%7C|wordpress_logged_in_[^=]*=([^%]+)%7C") {
                    set $do_not_cache 1;
                }
                proxy_no_cache     $do_not_cache;
                proxy_cache_bypass $do_not_cache;
                proxy_cache czone;
                proxy_cache_key "$scheme://$host$request_uri$is_args$args";
                proxy_cache_valid  200 301 302 10m;
                proxy_cache_valid  404 5m;
                proxy_cache_use_stale  error timeout invalid_header updating
                                       http_500 http_502 http_503 http_504;
                proxy_pass http://backend;
                proxy_redirect http://example.com:8080/ /;
        }
}
```

続くこの部分がリバースプロキシに関する設定である。リバースプロキシは 80 番ポートで待ち受けているので、普通にサイトにアクセスするとリバースプロキシにアクセスすることになる。login 画面を表示する PHP ファイルなど一部のキャッシュされると困るファイルに関しては除外するように設定している。

```
    server {
        listen 8080;
        server_name _;

        location / {
            root /path/to/wordpress;
            index index.php index.html;
            if (!-e $request_filename) {
                        rewrite ^(.+)$  /index.php?q=$1 last;
                    }
        }
        location ~*\.php$ {
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME /path/to/wordpress$fastcgi_script_name;
        }
    }
}
```

リバースプロキシにキャッシュがなく、新たに動的にページを生成しなければならないことがわかったときは 8080 番ポートで待ち受けている普通の Web サーバに処理が渡される。その 8080 番ポートで待ち受けているサーバの挙動を記述したのが上の部分である。PHP に関しては php5-fpm に処理を投げることによって PHP の実行速度が少しでも向上するようにしている。

3 結果

nginx の導入によってどれほど応答性能が向上したのかを調べようと考え、次のコマンドを別サーバから発行してみた。

```
% ab -n 100 -c 10 http://my.url/
```

これは Web サーバのベンチマークに相当するコマンドである。今回はベンチマークの結果から Request が 1 秒間に何個さばけたかという指標である Requests per second を比較することとした。比較のために Apache の導入も行い、Apache だけが起動している状態、nginx だけが起動している状態でそれぞれテストを行った。その結果、Apache では 0.9[requests/sec] だったのに対し、nginx では 68.11[requests/sec] という数値であった。nginx の導入によって明らかに Web サーバの応答性能が向上したのがよくわかる。応答性能が飛躍的に向上したので、より可用性が高くコネクション数が増えても落ちないサーバを目指しやすくなるのである。MySQL や PHP の設定をチューンすれば、更に性能の向上も期待できるだろう。

続 · 誰も得しない自宅サーバ構築記

mernao

mernao@mma.club.uec.ac.jp

はじめに

念願の自宅サーバ構築からかれこれ 1 年以上が経過。ちょうど 1 年前の百万石に書いた記事からその後一体何があったのか。答えは「大した事はしなかった」ということ。続 · 誰も得しない自宅サーバ構築記のはじまりはじまり。

1 自宅鯖の概要

サーバを建て始めたのは 2010 年 12 月下旬頃から。その頃は UNIX やらネットワークやら何も分からず右往左往していました[*1]。今見てみると当時は大分適当なことをしていたんだなぁと思います。当時のサーバの仕様はこんな感じです。とはいっても今もそれほど変わりませんが。

MotherBoard	Intel D510MO
CPU	Intel Atom 510 (1.66GHz × 2)
Memory	DDR2-800 2GB × 2
OS	FreeBSD8.1-RELEASE amd64
Works	http, imap, smtp, dns(内向け), ftp, irc, samba, etc...

もっと当時の事を詳しく知りたいという方は百万石 2011 年春号の私の記事をご覧ください。そういう方が居たらの話ですが (笑

2 外向け DNS サーバにする

折角固定 IP が振られているのでうち向けだけじゃなくて、外向けにも DNS サーバを建てたら面白そうだねということで、外向け DNS サーバを建ててみました。自分で DNS サーバを建てると、AAAA レコード (IPv6 のレコード) が書けてしまう[*2]というのがポイントです。既に内向け DNS を運用していたのでそれほど設定は苦ではありませんでした。ちょっと設定を晒しておきます。

/etc/namedb/named.conf

```
acl localnet {
        192.168.111.0/24;
        127.0.0.1;
};

view "internal"{
```

[*1] 今でもそうですが。

[*2] ちゃんと対応しているところは対応していますが。私の使っている DOT tk の DNS サーバは非対応だったんです。

```
        match-clients { localnet; };
  (中略)
};

view "external"{
        match-clients { any; };
  (中略)
};
```

敢えてここで DNS の設定について詳しく書く必要は無いと思うので大分割愛しています。ポイントは ACL を使っているところです。内部ネットワークの IP アドレスからアクセスがあった場合は localnet の ACL が適用されて、view "internal"の内容を見に行きます。これは match-clients localnet; が指定されているためです。一方外部からの場合は view "external"の内容を見に行きます。なので今までの内向けの設定を全て internal に書いておいて、外向けの内容を external に書いておけばいいわけです。

とりあえずこれで自分のサーバに対する設定は終わりです。ですがこれでは外部から名前を引いても自鯖に問い合わせは来ません。最後に今まで使っていた DNS サーバ (権威のあるサーバ) にグルーレコード (glue record) を書かないといけません。グルーレコードってなんじゃらほいと言うことですが、グルーレコードは簡単に言えば*.xeph.tk のレコードが ns1.xeph.tk というサーバにあって、ns1.xeph.tk の IP アドレスはいくつだよ、という情報のことです。逆にグルーレコードが無い場合を考えてみます。

まず我がサーバの xeph.tk の名前解決を行いたいとします。まずは.tk に対して xeph.tk の名前解決を依頼しますが、.tk は xeph.tk の IP アドレスを知らないので返答できません。しかしグルーレコードを持っていれば、「俺はそのレコードは持っていないから ns1.xeph.tk に聞いてくれ。IP アドレスはこれだよ。」を返答することができます。その返答を受け取って、ns1.xeph.tk に再び問い合わせに行けば良いわけです。最初はこの辺の仕組みがこんがらがっていたり、設定が反映されるまでそれなりに時間がかかったりで苦労しました。実際、DOT tk の Web 設定ページでは、Custom DNS を選択して、Host Name の欄に ns1.xeph.tk(named で設定した DNS サーバの名前),IP Address の欄に自宅鯖の IP アドレスが登録されていれば OK。他は何も登録する必要はないです。

確認のために dig コマンドで名前を引いてみれば、自分のサーバで名前解決をしていることが確認できるでしょう。AAAA レコードも引けちゃってなんだか感動です。

3 6to4 対応

外向け DNS については大分抽象的に書いてしまったので、6to4 については具体的に活用できるように書いていきたいと思います。

そもそも 6to4 とはなんぞ？という話ですが、超大雑把に説明すると、ネイティブに IPv6 が使えない環境でも IPv6 が使えちゃうという仕組みです。ただし条件があって、グローバル IPv4 アドレスが割り当てられている環境が必要です。まぁ今の IPv6 の普及状況だと、IPv6 が使えて嬉しいと実感できるのは踊る亀が見られる[*3]ぐらいな気もしますが、MMA 部員の場合は IPv6 が使えると部内のサーバに直接アクセスできるので割と必要なのです。

[*3] IPv6 でアクセスすると亀が踊り出すことで有名 http://www.kame.net/

設定の前に自宅のネットワーク環境の話をしておきますと、自宅サーバに直接グローバル IPv4 アドレスが割り振られているわけではなくて、ブロードバンドルータにグローバル IPv4 アドレスが割り振られていて、ブロードバンドルータの NAT 配下に自宅サーバがあります。こういう環境でも 6to4 は使用可能です。家ではブロードバンドルータの WAN 側へのアクセスを全て自宅サーバに転送しているので問題ないですが、そうでない場合はルータ側でプロトコル番号 41 を自宅サーバに向けて転送する設定をする必要があるようなので注意が必要です。

とりあえずサクッと使ってみましょう。とその前に rc.conf に `ipv6_enable="YES"`を加えておく必要があった気がします。多分。

```
$ sudo ifconfig gif0 create up
$ sudo ifconfig gif0 tunnel 192.168.111.2 192.88.99.1
$ sudo ifconfig gif0 inet6 2002:D294:CC::1/48
$ sudo route add -inet6 default ::1 -ifp gif0
```

はい。とは言ってもこの設定はうちのサーバ向けの設定なのでそのままでは使えません。まずは 2 行目の 192.168.111.2 ですが、大体想像がつくと思いますがこれが自宅サーバの NIC に割り当てられている LAN の IP アドレスです。こいつを適宜書き換えてください。192.88.99.1 の方は最寄りの 6to4 ルータ[*4]の IP アドレスなので書き換え無いでください。

次に 2 行目の 2002:D294:CC::1 ですが、まず 2002 から始まる IPv6 アドレスは 6to4 で個人に割り当てられるアドレスと決まっています。このアドレスは [2002:自分の IPv4 アドレスの上 16bit:自分の IPv4 アドレスの下 16bit::1] という構成になっています。つまり D294:CC の部分はうちのグローバル IP アドレス 210.148.0.204 を 16 進数に変換したものになっています。という訳で、この部分も適宜書き換える必要があります。

とりあえず書き換える部分は以上です。実際にコマンドを実行してみて、ifconfig で gif0 というインターフェイスが出来ているか確認してみてください。ちゃんと繋がっているかの確認は、www.kame.net のような IPv6 Ready なサーバに `ping6` でもしてみれば分かる思います。踊る亀さんが見れない場合は、ブラウザが IPv4 で接続しに行ってしまっています。仕方ないですね。

さて、このままだと起動時にいちいち手動で 6to4 の設定をしなければならないのでかっこ悪いです。起動時に上のコマンドを実行するスクリプトを仕込んでもいいんですが、折角なので rc.conf にちゃんと設定しておきたいです。意外と rc.conf に 6to4 の設定を書けばいいのかというのは調べても出てこなかったので、`man rc.conf` を読みつつ適当に設定しておきました。rc.conf の設定は最終的にこんな感じになりました。記事の最初にも書きましたが、FreeBSD 環境下です。

```
cloned_interfaces="gif0"
ifconfig_gif0="inet tunnel 192.168.111.2 192.88.99.1"
ipv6_enable="YES"
ifconfig_gif0_alias0="inet6 2002:D294:CC::1/48"
ipv6_static_routes="def"
ipv6_route_def="default ::1 -ifp gif0"
```

[*4] m 最寄りとは限りません。うちはひどいときはアメリカの 6to4 ルータに行っていました。

IP アドレスなどの細かい設定は適宜読み替えてください。cloned_interfaces は最初に書かないとインターフェイスを作ってくれないので、その後の設定が反映されず詰みます。あと、ipv6_enable="YES"を忘れても詰みます。こういう設定は初めて書いたので少し苦労しました。もう少し設定すれば v6Gateway にできて家中の PC に IPv6 が配れたりしますが、面倒だったりセキュリティ的に面倒そうだったりするのでまだやっていません。気が向いたらやると思います。

4　その他ゲーム用鯖化など

色々設定をしてきましたが、どれも大して仕事していないです。本当です。なので CPU 負荷率を見てみてもかなり暇そうです。だからという訳ではないですが、Minecraft というゲームを買ったので MultiplayServer を建ててあります。java で動いているサーバなので、特に設定はいりませんが、最初は IPv6 でしか LISTEN してくれないという謎挙動に悩まされました。後から気が付いたんですが、java の JDK をコンパイルしたときのコンパイルオプションで IPv6 が有効になっていたのが原因だった気がします。今は友達の WindowsPC で動いていた MinrcraftServer を移植[*5]して 24 時間稼働させています。忙しい時は CPU 負荷率が 1 を超えているので割とお仕事しまくりで、munin[*6]のグラフも賑やかです。ついでに TeamFortress2 という FPS のゲームのサーバにもしようとしましたが、さすがにマシンスペックが足りずに 16 人以上捌こうとするとかなり厳しかったので断念しました[*7]。

5　おわりに

とまぁこんな感じで運用しています。その他記事にするほどでもないような細かいことはちょくちょくやっています。ほとんどがセキュリティ関連の更新ですが。ちょうどこの記事を書いている今は libpcre に更新がかかったせいで libpcre に依存する ports を全て更新しているところです。先が見えないです...。そういう点では全部バイナリパッケージの OS の更新作業を見てると、あースマートなだなぁなんて思っちゃいますね。でもこの作業はこれはこれで楽しかったり... いやなんでもないです。そうそう、悲劇を産まないために/usr/ports/UPDATING はちゃんと読みましょう[*8]。ちなみに kernel は Generic のものを使っています。Generic だと freebsd-update が使えて便利ですからね。でも FreeBSD9.x にするときにはそろそろカーネルコンパイルもやってみようかななんて考えてたり。

これからサーバを建ててみようかなという人は FreeBSD なんていう選択肢もどうでしょう。そして春は新生活が始まる時期ですが、少し落ち着いたらサーバを建ててみるのも良いんじゃないでしょうか？色々な意味で毎日が変わりますよ。是非。

[*5] 勝手に色々いじったりもして
[*6] サーバの状態をグラフ化できるソフト
[*7] 最大 24 人程度で遊ぶゲームなので
[*8] 部長もそう言ってました！

大学を飛び出して、勉強会に参加してみて

hayakawa

hayakawa@mma.club.uec.ac.jp

はじめに

この数年、勉強会が各地で盛んに開催されている。このようなコミュニティに参加し始めて一年以上経過した。参加に至ったきっかけから今に至るまでの記した。

1　きっかけ

学外に飛び出した最初のきっかけは、2010 年 11 月 6 日に開催された ubuntu10.10 のリリースパーティーである。普段から読んでいたブログに、秋葉原でリリースパーティーを開くという告知を見て、少しだけ勇気を出して参加してみた。そこには、70 人を超える ubuntu ユーザがいて、日頃の ubuntu との関わりとかを話しあっていた気がする。参加者の中には、当時は私も ubuntu に関して素人であったが、会場で熟練の方に教わりながら、インストールしている風景も見ることが出来た。また、参加者は学生から社会人まで幅広く、九州から参加している人もいた。さらには、このパーティーを知るきっかけとなったブロガーの他に、有名なブロガーがいたり、Linux に関わる雑誌を書いている人も参加されていた。このパーティーは、昼過ぎから開催され、夜遅くまで続いた。その間、アルコールを飲み続けていたため、後半の記憶が少し怪しい。ここで、非常に楽しめたことが次のイベントへ参加の動機づけとなった。

2　R 言語のコミュニティに乗り込む

再び、あのような楽しい場に参加したいと、ubuntu のリリースパーティーから数週間後の 11 月 27 日に開催された” 第一回 Japan.R” に参加した。R 言語とは、統計解析を得意とするプログラミング言語であるが、 2 年生の時に大学の講義で習ったのをきっかけに、興味を持ち続けていた。このイベントは、Japan.R という名の通り、日本中の R コミュニティが集まる大規模なイベントで、午前中から開催された。午前の部は、“はじめての「R」“というテーマで、当時初心者であった自分にとって俺得すぎるテーマで、R 言語の環境構築から基本的な使い方の説明であった。午後からは、R ユーザの方々が、実際にどのような現場で R を使っているかを聞くことができて、R 言語を学習する良い動機づけが出来た。

次に参加したのは、Japan.R を主催していた方が開催している Tokyo.R というコミュニティで、今までの内容と比較出来ないような高度な内容な発表であった。そこで、自分の知識不足を痛感した。次に参加しようと考えていた” にこにこテキストマイニング勉強会” という勉強会の運営者が Tokyo.R で発表をされていて、懇親会にて話しをすることが出来た。その人が、” 豆乳システム” というプログラミング言語及び開発環境を知っていたことは、感動的であった。なぜなら、この言語を知っている人に出会えたことが、今までに一度も無かったからである。その時に、twitter でフォローしたことが、今後の急展開を招いた。

3　初めての発表

Tokyo.R で知り合った方が開催する勉強会は、初心者向けのテキストマイニング勉強会であった。テキストマイニングに対しては、興味を持っているものの未知の世界であった。そのため、どんな世界かを知る手がかりになるのに良いと思ったオライリーの出版物を読みたいなあっと何気なく twitter で呟いていたら、運営者に” 次の勉強会で発表してくれ !! “といった内容の事を言われて、断ることが出来ないような状況に追い込まれて、“発表します” と答えてしまったことが、勉強会での初めての発表となった。勉強会の当日になるまで、必死に準備をして臨んだ発表では、初心者が躓きやすいポイントやテキストマイニングの流れが分かりやすかったらしく、思いの外に高評価を得て、驚いてた。そして、いつの間にかその勉強会でレギュラー発表者になっていた。

4　増え続ける発表会数と意外な展開

テキストマイニングの勉強会でレギュラー発表になってしまったせいか、勉強会で発表することに抵抗を感じなくなり、これ以外の勉強会でも発表するようになった。勉強会に参加していくなかで、勉強会が職探しの場となっていることも知った。当時の自分は、コンビニでアルバイトをしていて、勉強に専念するためにも辞めようかと思っていたが、勉強とバイトが同時に行うことが出来たら素敵だなと思って、発表に使用したスライドの最後に、“バイト探しています。“という主旨のスライドと興味のある分野を列挙した。そしたら、ある日突然に、“良かったらバイトしませんか？“という話を頂き、非常に嬉しかった。その後は、懸賞論文に投稿したものが受賞してしまったり、本を執筆した方々にサインを頂いたり、ベンチャー企業や研究開発のお手伝いをさせてもらえたりと、エキサイティングな生活を送っている。

5　最後に

初めは、イベントやコミュニティに参加するのは、緊張の連続であったが、少しだけ勇気を出して、学外に飛び出してみて本当に良かったと思ってる。今までに知らなかった世界に出会うことができたり、社会人の方や他大の学生と話をすることができたりと、面白い経験を数多くした。最近では、MMA も外部と積極的に交流を深めていて、今まで以上に刺激的なサークル活動をすることが出来ているように感じる。

計算機バカ

thunder_lab

thunder_lab@mma.club.uec.ac.jp

はじめに

thunder_lab が一年間[*1]で買った計算機やらパーツやらの記録

1 はじめに

百萬石の春号といえば新歓を意識したものが強い。この記事もそれを念頭としたものであるということをご了承いただきたい。

さて、今回は私が去年一年間で購入した計算機に関連するものの購入記録を示そうと思う。去年は CPU において、Intel からは SandyBridge、AMD からは Bulldozer と、両メーカから新しい CPU が発売されたり、タイの洪水により HDD の価格が高騰したりと、様々なことがあった。その中で、自分でも自身が馬鹿だろうと思う程度にはパーツを買い漁った気がする。現状で一人暮らしながら部屋に計算機がデスクトップ/ノート合わせて 6 台という状態である。それぞれのマシン構成は MMA の wiki 上[*2]に記しているので合わせて参照していただければと思う。

また、新入生の中には入学を機にマシンを自作しようと思っているものもいるであろう。この記事を読んで少しでもパーツ選定などの役に立っていただければ本望である。

2 買ったもの一覧

というわけで買ったもの一覧である。

- CPU
  - AMD FX-8120
  - AMD AthlonII 245e
  - Intel Core i5 2500k
  - Intel Celeron G530
- CPU クーラー
  - サイズ APSALUS 120
  - ANTEC KUHLER-H2O-920
- マザーボード
  - MSI 970A-G45
  - ASUS P8H61-M LX PLUS REV 3.0
  - MSI Z68A-GD55
- メモリ
  - Corsair CMZ8GX3M2A1600C9B [DDR3 PC3-12800 4GB 2 枚組]

[*1] 年度換算

[*2] http://wiki.mma.club.uec.ac.jp/thunder_lab/computers

  - TEAM TED38192M1333C9DC-AS [DDR3 PC3-10600 4GB 2 枚組] x2
  - SUPERTALENT W1333UX4GM [DDR3 PC3-10600 2GB 2 枚組]
  - ADATA AX3U1600GC4G9-2G [DDR3 PC3-12800 4GB 2 枚組]
  - CFD D3N1333Q-2G [SODIMM DDR3 PC3-10600 2GB]
- HDD
  - WD WD20EARX [2TB SATA600]
  - WD WD25EZRX [2.5TB SATA600]
  - WD WD5000AAKX [500GB SATA600 7200]
  - HGST HDS721050CLA662 [500GB SATA600 7200]
- 電源
  - HEC WIN+ 550W HEC-550TE-2WX
  - 玄人志向 KRPW-G530W/90+
- ケース x2

3 評価

購入したものの列挙が終わったところで各購入物をかいつまんで評価していこう[*3]。

まず最初に録画マシン用として AthlonII を調達した。これと今までメインで使っていたマザー（AM3）を組み合わせた次第であるが、この様な用途では非常に快適である。また、BIOS 設定により定格よりも大幅に電圧を落とした運用をしている。とはいえ、今このクラスの CPU を買うなら既に APU しか無い訳だが。

次に、メイン機用に FX-8120 を調達した。これは発売日に講義をぶっちぎってまで発売日に買いに行った代物だが……組み合わせたマザー、既存の電源もろとも今は亡きものとなっている。レビューや、死亡した詳細は前号の百萬石を参照されたい。

というわけで、急遽メイン機用に調達されたのが Intel の 2500k である。それに合わせてマザーも調達した。現在は 4.5GHz での常用をしている。2500k のベンチマークを取っていて非常に悲しい思いをしたのだが、2500k>>> 壁 >>>FX-8120 であったことはここに記しておく。また、消費電力についても同様に大きな壁があった[*4]。これも悲しくなるほどに。

そして、最後に調達された CPU が Celeron の G530 である。Celeron と言いながら、この世代のものは完全に予想を越えた高性能/低消費電力で非常に驚いた。旧世代の Core2Duo と完全に並ぶ程度の性能ながら 3000 円ほどで購入できる。メインでの使用は厳しいかもしれないが、サブ用途などでは非常に有力である。私はこれを自宅サーバー用途で調達したが、非常に快適である。また、この石は TDP65W となっているが、35W でもいいのではないかという程には非常に優れた低消費電力を実現している。

CPU の総評として、SandyBridge が非常に強力であることがとても印象強い。Bulldozer は非常に厳しい状況にある[*5]。オーバークロックで差を埋めようにも、Sandy の方が皮肉なことに耐性は高いし、もうすぐ IvyBridge も登場し差は広がってしまうばかりである。しかし、Bulldozer も最近になって価格改定が入るなどしているので、検討の余地はあるかもしれない。

[*3] HDD はお察しください。
[*4] どっちがどっちかはお察しください。
[*5] 談:元 AMD 信者

メモリに目を向けると、かねてからの価格暴落もあって、非常に買いやすい状態にある。最近では DDR3 では 1333 と 1600 の価格差がほとんど見られないので、今買うなら後者にするべきである。こんな価格なおかげでポンポン買ってしまった。Corsair メモリは悪くなかったのだが、ある時突然不安定になったので memtest を走らせたところ、セットのうち 1 枚でエラーが起こってしまっていた。残念なことに購入時のレシートを無くしてしまったので永久保証による交換ができなかった……。この代替で購入したのが ADATA メモリである。最近、低価格もあって話題となっている ADATA だが、まさか自分で使うときが来るとは思わなかった。しかしこいつ、意外と優秀である。memtest は一晩 13 パスしたので常用しているが、非常に快適である。爆安だからといって敬遠するかもしれないが、予算によっては充分検討する価値はあると思う。

CPU クーラーは購入したものすべてが簡易水冷である。やはり安物は値段相応だと思った。最近は値段もこなれてきているので、比較的手も出しやすいと思う。冷却力を求めているなら空冷からの変更を考えてもいいと思う。

電源は最近は BRONZE 認証のものが非常に多い。電源の質が低いと色々なパーツを巻き込んでお亡くなりになることもあるので、できる限りいいものを購入したいところである。購入した玄人志向の電源は秋葉原で特価だった GOLD 認証品で、これも自宅サーバー用途で購入したが、非常に音も静かで発熱も低く、これぞ GOLD 電源だと思わせる動きをしている。

4　最後に

去年一年間で購入したものについて、大雑把ではあるが評価してみた。自分のような貧乏学生としては、パーツを購入する際も常に予算というものがまとわりついてしまい、あまり高級なものが買えていないと思う[*6]。しかし、安くてもいいものはもちろん存在するし、それらを狙って買ってみる人柱精神も非常におもしろい。こんなお粗末な記事ではあるが、冒頭でも述べたようにこれからパーツを買おうと考えている人の参考に少しでもなれば幸いである。

それにしても、我ながらよくこれだけ一年で買い込んだものだ……

[*6] 安物を集めれば高価なもの買えるじゃんという突っ込みはなしの方向で

似非ガジェ充日記

mernao

mernao@mma.club.uec.ac.jp

はじめに

MNP 新規ユーザ優遇による既存ユーザへの冒涜——端末価格は 0 円へと収束する。2012 年、端末はキャリアに支配されていた。実質と一括が入り交じる世界、0 円の存在さえ疑う。現金でないキャッシュの存在。相次ぐ通信障害。増え続ける CB 額。背くことの出来ぬ違約金——そしてプリモバイルが 9,975 円へと収束し世界は転換期を迎える。

1　はじめに

これは、ガジェ充になろうとしたある大学生の記録である。筆者は、この内容によって生じた損害に関して一切責任を負えない。

2　全てのはじまり

元々ガジェット類に興味のあった私は当時初代 W-ZERO3 のメモリ増強版である WS004SH を使用していた。WS004SH は買った当時 4 万円以上していた。実は WS004SH は 2 台持っていて、1 台目はイヤホン端子が故障してしまったため、ちょうど安く買えたこともあり同じものを買っていた。それでも私の中ではガジェットと言えば数万はする高級なものでそうポンポン買えるものではないという認識があった。しかし IS01 の登場がその考えを変えてしまった。

2010 年 9 月、MMA の合宿で yajima 先輩から IS01 に関する話を聞いた。何でも、最初の事務手数料と月々のユニバーサル手数料 7 円を払うだけで IS01 という Android 端末が au から買えてしまうという話だった。信じられなかったが、実際に調べてみると IS デビュー割[*1]というのを適用すると確かに最初 3,000 円程度払うだけで、本体代金は一括 0 円で買えてその後も維持できてしまい、更新月に解約すれば高い違約金も発生しないらしい。これはフューチャーフォンとの 2 台持ちをしやすくするための au の施策だったのだと思うが、今までたとえ本体代金が 0 円だとしても高い維持費を払わなければ維持は不可能だと思っていた自分にとっては驚き以外の何物でもなかった。急いで契約をしようと思ったが、生憎不具合修正のため販売停止になっていた。だが IS01 を探す中でたまたま IS02 が同様に一括 0 円で売られていたので契約した。しばらくすると IS01 の販売が再開され、IS01 も無事 GET することが出来た。同じ話を聞いた MMA 部員数名も同様にして IS01 を契約したので、MMA 部員で IS01 を持っている人は実は結構いる。

ちなみに IS01,02 の両端末は、料金プランを工夫することで 1 台につき 1,050 円分の無料通話料が手に入れられた。もちろん維持費は変わらない。つまり、毎月 7 円しか払わないのに無料通話分を錬金することが可能なのだ。これには大分お世話になっている。

こうして私の充実した端末ライフが始まった。

[*1] 今は存在しない

3 DesireHD の放出

翌年 2011 年 3 月に DesireHD という HTC 製のかなり良い端末が SoftBank から一括 0 円で販売されるという騒ぎがあった。かなり良い端末と書いたが、DesireHD はその当時日本では最高峰のスペックを誇っていたと思う。今でも退色無く愛用している端末である。ハッキリ言って IS01,02 のスペックは良いとは言えなかった。IS01 は良いキーボードこそ備えているが、深刻なメモリ不足と Android1.6 からのアップデート見込みがないなどで、メガネケース[*2]と揶揄されるほど散々だった。IS02 に関しても OS が WindowsMobile6.5 という時点でお察しくださいという状況だった。そんなこともあり、DesireHD のようなまともな端末は是非欲しかった。

DesireHD の一括 0 円は IS01 の時と同じように、学割適用で月額 7 円維持ができるというものだった。この時に厄介だったのがフォトビジョンの存在だった。フォトビジョンとは、いわゆるデジタルフォトフレームなのだが、こいつを契約してしまうと通常の 1 回線としてカウントされて更新月以外の解約時に 1 万円近く請求されてしまう。本体代金こそ 0 円だが、維持には毎月 680 円程度必要なため、塵も積もればで 24ヶ月契約するとなるとバカにならない。なのでフォトビジョンの契約はなるべく避けたかった[*3]。そこで数店舗回ったが、某ヨド○シカメラはフォトビジョンとの抱き合わせ販売で、店員に聞いてもどうしてもフォトビジョン無しは無理との事だったので諦めるしか無かった。しかし某石○電気は違った。なんと普通にフォトビジョン無しで販売していた。ヨド○シカメラの店員の言葉が蘇る。私の心の中に店員を疑う心が芽生えてきた瞬間でもあった。こうして無事 DesireHD を手に入れることが出来た。その日の夜は何故か熱を出して寝込んでいた。ガジェ充になるには体力が必要らしい。

この DesireHD なかなか素晴らしいので、root 化して Proxy を通せるようにしたり SIM ロック解除をして遊んだりもした。唯一の欠点は解約したり SIM を抜いたりすると画面がロックされて何も出来なくなってしまうことである。SIM ロック解除をしていなかったら危うく何も出来なくなるところだった。これは SoftBank の仕様のようなので仕方がない。

だがこの DesireHD を境に、新規一括 0 円端末に出会うことは少なくなっていった——

4 EVO 祭り

この時点で既に端末をいくつか手に入れていたが、依然としてメイン機は WS004SH だった。一度ハードキーボードを手にしてしまうとなかなか他には移れなかったためだ。しかも WILLCOM 回線は、かなり遅かったが 980 円でパケットし放題でかなりリーズナブルだったので、なかなか乗り換えられずにいた。とはいえ、そろそろ WILLCOM の PHS 回線にも嫌気が差してきていた。そこで出会ったのが EVO WiMAX(ISW11T) だった。当時 au は密かに WiMAX のエリア確認ツールとして、一週間 EVO を無料貸し出ししていた。どんなもんなのかと丁度帰省をするときに借りてみたところ、PHS ユーザだった私にとって想像を絶する快適さだった。

でもお高いんでしょう？——そう私は思っていた。だが意外にもどれだけパケットを使っても 4,000 円程度で済むという事が分かった。これは毎月割という仕組みのおかげで、毎月 2,850 円が引かれるためだった。これは元々毎月の料金を 24ヶ月値引きすることで、端末代が実質 0 円！ということを謳うためのものだっ

[*2] 見た目がメガネケースっぽいのでそのようなあだなが付けられた。

[*3] 不要な上に使い道が無く、非常に不評だったのでガジェット好きからはゴミフレームなどと厳しく揶揄されることも多かった。

た筈なのだが、店によっては EVO 本体が一括 0 円で売られているので、その場合は単純に月額使用料が安くなるだけになるのである。一度 3G 回線の味をしめてしまうともう後には戻れない。しかも思っていたほど高くなかったので契約を決めた。この回線を主回線として今に至っている。契約したのは京都のヨド○シカメラだったが、東京の某ヨド○シカメラでは付いてこなかった SMT-i9100 という SAMSUNG 製のタブレット端末が付いてきた。東京では MNP 契約者には付いてこないと言われていたので、あれれ？という感じだった。ちなみに今回からは普通の新規一括という買い方ではなくて MNP 新規一括という買い方だった。犠牲にしたのは DesireHD の回線だった。というのも、MNP でないと本体代金は 0 円にならない仕組みだったからだ。この頃から MNP でお得に買えるという流れが既に出来ていたと思う。だがこの当時はまだプリモバイルの事を知らなかった。そう、まだ——

5　EVO 祭りの終焉 ·CB 時代の到来

そうこうしていると EVO の毎月 2,850 円は急に無くなり、お得感は急激に薄れて行った。前にも述べたが、この頃既に端末が新規一括で買える案件は少なかった。更に最低維持しても月に 221 円程度どうしても掛かってしまうものしか無かった。もう端末は安く買えないのかと一時は思ったが、実はそうではなくむしろもっとお得な、もはやもらえるのは端末だけではない時代に突入していた。

ここで登場したのがプリモバイル (以下プリモバ) とキャッシュバック (以下 CB) の組み合わせである。この頃からか分からないが au への MNP 転入者に対してのみ高額のキャッシュバックを行い優遇するという販売店が増え始めた。キャッシュバックがあると、仮に解約して違約金が発生したとしても相殺することができる。それどころか手元にある程度残る。しかしキャッシュバックを受けるためには MNP をしなければならないので、MNP 転出元を確保する必要がある。その MNP の転出元、通称 MNP 弾丸にするために多く使われたのが SoftBank のプリモバイル携帯である。このプリモバイル携帯はいわゆるプリペイド携帯で、MNP 解約時に違約金が発生しないのが大きな特徴である。そこで、プリモバイルの回線を MNP に利用すれば、転出手数料 2,100 円 + 初回チャージ料 3000 円の合計 5,000 円ちょっとで MNP の転出元が確保できてしまう。一方の au の CB はその当時多いものだと 5 万円ぐらいあったので十分に相殺が可能であった。とは言っても、やり過ぎると SoftBank のブラックリストに載るという風に言われているので、やり過ぎには注意が必要である。

早速やろうと考えて、まずは MNP 弾丸の確保に奔走した。とは言ってもどこのソフトバンクショップに行ってもプリモバイル携帯本体が売り切れであった。仕方がないのでとりあえず送料無料だったので SoftBank のオンラインショップで 740SC を 2,079 円で購入した。が、数日後に友人から新宿の某モ○ワンで X04HT が売っているという情報を聞いてすかさず買いに行ってしまい、740SC は無駄な買い物になってしまった。ところで X04HT とはというと、2008 年 11 月発売の WindowsMobile6.1 搭載とかなり古いが、本体が 1,000 円程度で手に入ってしまうのでかなり人気のある機種だった。古いが一応スマートフォンという点も面白い。こうして私の CB 時代が幕を開けた。

6　MNP 弾丸で契約 · メイン機も交代

MNP 弾丸が手に入ったところで、いよいよ端末の購入に踏み切った。この後ずっとお世話になるのが某テ○ルという携帯ショップである。このお店は CB の条件がなかなか良い。まずは 12 月上旬に IS04FV を MNP 本体一括 0 円、3.5 万円 CB 付きで契約した。その後、解約することになるが、MNP 弾丸作成費を考

えても差し引き 1 万円以上手元に残る計算だ。端末を買ったはずなのに逆にお金が増えるなんて信じられない話だと思うが、事実である。IS04 は REGZA Phone 第一弾であるが、その出来栄えはネット上での叩かれ具合[*4]からお察しくださいといったところだ。現在はお風呂用防水端末として活躍中である。IS04 については色々と言いたいことがあるが、ここでは述べないことにする。

勢いに乗って 12 月下旬に EVO 3D(ISW12HT) も MNP 本体一括 0 円、3.5 万円 CB 付きで契約した。こいつは本命機種で、メイン機として使おうという魂胆だ。しかし、8 月に買った初代 EVO の方が毎月割が大きいので、EVO 3D をそのまま運用するとお得感が薄れる。そこで SIM 入れ、SIM 出しという技を使った。旧 EVO も EVO 3D も普通の端末と異なり、SIM カードスロットを持たないいわゆる ROM 機と呼ばれる機種であった。通常 au の持ち込み機種変は 2,100 円を請求される[*5]が、何故か ROM 機から SIM 機への機種変などの場合は無料で持ち込み機種変が可能になっている。そこで、まず EVO 3D の回線を SIM 機である IS02 に持ち込み機種変を行った。この行為を俗に SIM 出しと呼ぶ。なぜならこの作業で IS02 で使う為の SIM が新しく発行されるからだ。次に旧 EVO の回線を EVO 3D に入れる。最初からこれをやれば良いと思う人もいるかもしれないが、ROM 機の場合、中に生きた回線があると、他から機種変出来ないという制約があるため 2 回に分ける必要がある。これが非常に厄介で、きちんと理解している au ショップの店員が少ないために、たまに変なことを言われたりするが、そういう時はセンターに確認を取ってもらうしか無い。そんなこんなで色々と苦労したが、無事電話番号も変わること無く、お得に最新機種の EVO 3D を手に入れることが出来た。肝心の EVO 3D の使い心地だが、旧 EVO と比べるとさすがにかなり良かった[*6]。

CB 時代に突入してのポイントは、店を選ぶという事だ。実質 0 円[*7]などという紛らわしい言葉が使われていたり、同時にデータ回線を掴まされたりする店が多いので注意が必要だ。

7　ガジェ充達のプチツアー ·3 台同時契約芸

2012 年 2 月になり、無事 12 月に契約した際の CB が帰ってきた。CB は通常、数ヶ月後に指定口座に振込みになっていたりするので、振り込まれるまでに解約を行ったりすると CB が帰って来ない事が大半である。

CB も帰って来た事だし、そろそろ新しい端末を買ってみたい欲に駆られてきたので、友人を誘って某テ○ルへと向かった。今回は CB が帰ってきた au2 回線を docomo に MNP しつつ、あわよくば au で 1 台契約という計画だった。CB が帰ってきた回線はそのまま解約でもいいのだが、+2,100 円払って MNP 弾丸にしてしまったほうが、また新しい端末が手に入って面白そうなので docomo に転出しようと考えた。

2 月の後半の某日にツアーは決行された。私が自宅車を運転して、MMA 部員計 4 人で押しかけるという割と大規模なものになっていた。普段なら数時間で契約も済むだろう——そう考えていた。だがこの日は違った。あちこちで年度末の処分セールが行われていたのだろう、au や SoftBank のセンター自体が混雑していて、契約に非常に時間がかかり、ソフトバンクショップに着いたのが 11 時頃だったが、テ○ルで最終的に受け取ったのは夜の 18 時半を過ぎていた。

さて今回の私の契約は、まず au2 回線を docomo に MNP した。機種は 2 台とも L-07C を契約。MNP

---

*4 アアアッフォン (半角カタカナ) で検索すればおわかり頂けるだろう。

*5 そもそも機種変なんて SIM カードを入れ替えればいいじゃないかと思う人も居るだろうが、au の端末は SIM を入れ替えても有料のロック解除を申請しないと使えないようになっている。

*6 そもそも旧 EVO は日本で発売された時期が海外で発売された時期よりかなり遅れていたため、新商品なのに時代遅れという端末だった。

*7 実質 0 円は全然お得じゃないので注意。

新規一括本体 0 円、3 万円 CB だった。実質手元に残るのは一台につき 1 万円なのでやはり損はしないどころかお得になる。その後、プリモバイル弾丸を使って au の PHOTON を契約した。MNP 新規一括本体 0 円、5 万円 CB だった。これもやはり損しないどころの話ではない。ちょうど PHOTON は前から欲しかったので良物件だった。そんな感じで私は計 3 台、友人らは EVO 3D を 1 台ずつ契約した。EVO 3D の条件は PHOTON と同じであった。するとどういう事だろうか、4 人全員で 26 万円の CB がくっついてきた。もはや月給レベルである。さらに 4 人同時契約特典などで 3.5 万円分の商品券が付いてきたりでてんこ盛りだった。そういうわけで一日を終えた。皆幸せな端末ライフを送ろうとしていた。皆またこんな風に端末を買いに行きたいと考えていた。そう、この時は誰もこんな未来は予想していなかった——

8　そして終焉へ

2012 年 3 月 5 日、それは唐突に起こった。SoftBank からプリモバイルから MNP 転出を含む解約を行った場合、違約金 9,975 円が発生するという発表がなされた。つまり今までのように気軽にプリモバイルを MNP 弾丸に出来なくなってしまうわけだ。どうやらプリモバイル回線が MNP に利用されることがかなり多いらしく、それに対する対策ということらしい。この条件の適用は 2012 年 4 月 4 日契約分からである。つまりそれまでに作った MNP 弾丸に関しては適用されない。そこで今までこの方法を常用してきた人達は焦りを見せた。これで MNP 祭りは終りを迎えたかのように思えた。しかし、既にポストペイド回線を多数所有している者に対しては、実はそれほど意味がないのである。例えば私の場合 docomo の回線を再び au に MNP し、また docomo に MNP という荒業をしていけば良いのである[*8]。とはいえやりにくくなった事は事実である。

9　批評

今まで一通り説明してきたが、何かおかしいかと思わないだろうか？そう、このやり方はどう考えても新規顧客、しかも他のキャリアの契約者数を確実に奪える MNP での新規ユーザをなんとしてでも増やそうという考えのもとに行われているバラマキである。しかしこの場合既存ユーザはどうなるだろうか？損をしているように感じないだろうか？更に端末の 0 円化が加速していないだろうか？これでは発売日にまともな価格で買った人があまりにも不利である。中には発売当初から 0 円の端末も出てきているようである。数年前に総務省からの通達で 0 円端末は無くなるような素振りを見せたかと思えばスマートフォンの登場で一気に崩れ去ってしまい、あまり健全な状況では無いように感じる。安く端末を買えている私に言う資格は無いかもしれないが、このやり方には正直疑問を感じざるを得ない。かく言う私も途中からこのやり方に嫌気が差して、まともに端末を買う気が失せてしまった人間である。やはり維持費は安いほうが良いに決まっている。今後状況がどう変わっていくかが気になるところである。

10　おわりに

本来ならばこの端末群を活かして Android アプリ開発を... というところだったが、eclipse を導入して AndroidSDK の環境などを整えたところで時間が来て力尽きてしまった。よく考えたら私はプログラミングというのはそれほど得意では無いのだった。次回の百万石までには何かが出来ていると良いが... 構想として

[*8] とは言っても docomo にもブラックリストがあるのではないかと一部では囁かれているので心配は残る。

は端末一台一台をポップンのコントーラのようにして... という非実用的な事を考えている。

結局今は 15 台[*9]端末を持っていて、9 回線を運用している。世の中には 40 台以上端末を持っていたり、20 回線以上運用していたりする人が居るので恐ろしいものだ。まだ迂闊にガジェ充は名乗れなさそうである。

記事中では紹介しなかったが、MNP の為に必須な 3,000 円のチャージは一見無駄になりそうだが、チャリティダイアルに電話をすることで全額寄付が可能だ。そのまま無駄にするよりは是非寄付をすることをおすすめする。とは言ってももうこの記事が読まれているときにはプリモバイル MNP は難しくなっているだろうが。

いかがだっただろうか？今後端末を買う際の参考になれば幸いである。そう、端末は賢く買えば安く買えるのである。とは言ってもプリモバイルが使えなくなったので少々難しくはなるが、それでも au にもぷりペイドというプリペイドのサービスはあるし[*10]、docomo にも B ナンバーという仕組みが存在し、それらを上手く使えば MNP 弾丸として利用する事が可能である。少々面倒ではあるが、試してみる価値はあるのではないだろうか。その際はくれぐれもよく調べた上で、自己責任でお願いいただきたい。

それではみなさんの素敵な端末ライフを願っております。

図 1　所有ガジェットたち

*9 この記事を書いている最中に X02T と IS12T が仲間に加わった。IS12T は学割 +U25 割適用で WiMAX を使っても月 2155 円という格安回線になる。

*10 この記事を書いている最中に SoftBank に引き続き、au もプリペイドの規約を変えてきて、弾にかかるコストが高くなってしまったので要注意。

百萬石 2012年 春号

2012 年 4 月 4 日　初版第一刷発行　【本書の無断転載を禁ず】
2012 年 4 月 6 日　初版第二刷発行

著　者　alstamber, hayakawa, kakakaya, KHe7, mernao, thunder_lab, wataj
編集者　clear
発行者　電気通信大学 MMA
発行所　電気通信大学 MMA 部室
〒182−8585 東京都調布市調布ヶ丘 1−5−1 サークル会館 208 号室
http://www.mma.club.uec.ac.jp/
印刷所　電気通信大学 MMA 部室
製本所　電気通信大学 MMA 部室
表　紙　電気通信大学 MMA


我々はろくろを回すことを……
強いられているんだッ！