

XX 独立

工学部機械システム工学科

森山 雅雄

matsu@welcome.mech.nagasaki-u.ac.jp

1 はじめに

ども! 色物担当の森山です。いーかげんにしろよ! おめー! と思う方もいらっしゃると思いますが、今年も書かせてもらいます。この題目を見てピンときた人、あなたは 70 年代中ごろから 80 年代中ごろにかけて `bit`^{†1} 読んでましたね。今でも `bit` はありますが、あのころは、伊理正夫先生、森口繁一先生、一松信先生などが数値計算に関する記事をお書きになられていてとても勉強になりました。あっ、それと島内剛一先生の、なんでも日本語で表現する^{†2} とてもユニークな記事もあって、計算機科学への興味をそそる本でした^{†3}。80 年代後半にはいると、ICOT^{†4} がらみの人工知能関係の記事が増えてきて段々つまらない本になってしまいました。

今回の題目である「XX 独立」というのは、私が `bit` を読んでいたころ時々出てきた UNCOL (Universal COmpiler Language) という、ちょっとめずらしいコンパイラ作成のアプローチの話で出てきた言葉です。確か、前田英明先生か THEMSKY さん^{†5} の記事だったと記憶しています。UNCOL というのは、N 種類の計算機 (CPU) と M 種類のプログラム言語があるとき、コンパイラは N×M 個作成しなくてはいけないのですが、UNCOL のアプローチは、M 種のプログラム言語を、一旦中間言語 (これが UNCOL) にコンパイル (トランスレート) し、それを N 種の CPU のネイティブコードにコンパイルする形式にすると N+M のコンパイラで済む、というコンセプトの話だったと記憶しています。UNCOL は結局 UCSD p-system^{†6} くらいしか成功しなかった^{†7} のですが、UNCOL の基本概念である「CPU 独立、言語独立」というコンセプトは、その後いろいろな形式で実用化されています。今回はこのような「XX 独立」のいろいろな XX の部分にスポットをあてた話を書こうと思っています。

2 プリンタ独立

以前、この話は書いたことがあるんですが、プリンタに印字するとき、制御コードがメーカ独自なので、いろいろと大変だというわけで、共通のプリンタ制御コードをつくっちゃえ!^{†8} というまさに「プリンタ独立」なノリでできたのが Postscript です。下の図を見て下さい。Postscript printer には、普通のプリンタが持っているプリントエンジン (印刷用ハードウェアと制御装置) の他に、Postscript interpreter というものが入っています。

^{†1} 共立出版から出ている計算機科学の雑誌です。

^{†2} プログラム=算譜、データ=算料、ファイル=算帳、カーソル=遊標などです。

^{†3} 当時は機械科の学生だったんですが、製図嫌い、力学大嫌い、そんでもってやたら小生意気なダメ学生でした。

^{†4} 第 5 世代コンピュータなんかとかんとか、というわけのわからない通産省のプロジェクトです。

^{†5} 何人かの著者の頭文字を並べたものだそうです、もしかしたら前田先生も入っていたかも知れません。

^{†6} 80 年代前半のパスカルコンパイラが有名です。Turbo PASCAL の前に統合環境を実現していました。

^{†7} MINIX のタネンバウム先生が ACK というコンパイラコンパイラを作った時、このアプローチを使ったそうです。

^{†8} こんなに下品ではないとおもいますけど。

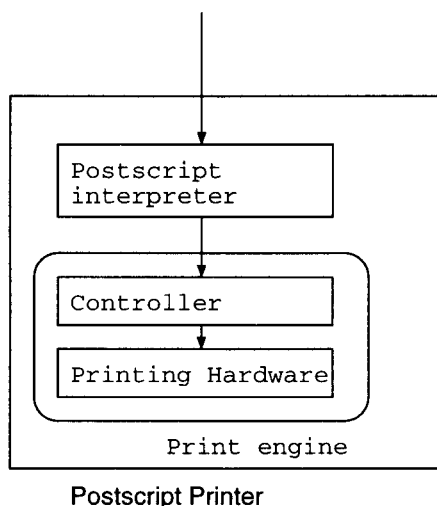


図 1: Schematic of Postscript printer

従来のプリンタは、プリントエンジンの制御装置のコマンドが制御コード (ESC/P とか LIPS とかが有名ですね) と呼ばれていて各社まちまちでした。Postscript Interpreter は、Postscript という一種のプログラム言語を対応するプリントエンジンの制御コードに変換する機能を持ちます。このようなプリンタ制御用言語を標準化したおかげで、各プリンタメーカーは自社のプリンタに Postscript interpreter をつけるだけで、簡単に Postscript printer を製品ラインナップに加えることができるし、ユーザは割と安価に購入できるというわけです。

3 端末独立

3.1 キャラクタ端末の場合

UNIX を端末から使っている場合を想定して下さい。端末の種類ってどのくらいあると思いますか？ /etc/termcap があれば見て下さい、テキストファイルですから、less などのコマンドで内容が確認できます。まさに死ぬほどありますよね。less コマンドにしても、エディタにしても、これら端末の種類をすべて認識して作られているのでしょうか？ 答えはいいえです。先ほどの termcap は、端末別の制御コード^{†9} が書いてあるデータベースです。UNIX マシンで `man termcap` と入力して、その内容を確認してみてください。概念的に言えば vi のようなエディタだの、less のようなファイルビューアだのは、一文字送りだの、一ページ送りだのといった抽象的なコードを出力し、それが termcap みたいな端末データベースを通り抜けて実際の端末制御コードが送られるような機構になっています^{†10}。これがあるおかげで、プログラミングがだいぶ楽になります。98 時代を知ってる人は思い出してください、エスケープシーケンスでカーソルをいろいろな場所に動かしたり、テキストの色を変えたりしてましたよね。でもそういうプログラムって 98 専用だったでしょ、DOS/V マシンというへんてこな名前と呼ばれる PC が出てきたとき、ちょっと凝った画面制御をする 98 用のプログラムは動かなかったですね。これは、98 独自の画面制御コードを使っているため、「端末独立」なプログラムじゃないからです。

3.2 GUI の場合

とまあ、ここまでは昔の bit に載ってたことで、結構忘れていたんですが、95 年に留学^{†11} したとき、これに類することを体験する機会がありまして、昔の bit の記事がフラッシュバックしました。それが端末独立な GUI って奴です。

^{†9} 一文字送りとか、一画面スクロールとかです。

^{†10} 実際は、termplib とか curses とかの UNIX のライブラリ中の一文字出力とか一画面スクロールとかの関数を呼ぶ形でプログラムして、実行時にそのときの端末の種類を termcap から探して、それなりの信号を出力する形式で実行されます。

^{†11} しつこいですが、機械にあたったのに学振ではなかったんです。忘れねえぞ！ > 一部の連中

GUIってのは Graphic User Interface のことで、マウスでうりうりという人間を駄目にする無駄で有害なユーザインターフェースです。ちなみにアメリカの連中は「グイ」と発音してまして、ちょっと耳に違和感がありました。GUI 対応のプログラムってえらく面倒なんですよ。んだってさ、Windows はあるわ、Mac はあるわ、X-Window はあるわだし、コンパイラやツールキットとかいうライブラリが違って、まるっきりプログラムが違っちゃうんですよ。んな無駄なことに頭使うよりプログラム本体をしっかりと考えたいですよ。

私は、NASA/JPL (Jet Propulsion Laboratory) に留学してまして、そこで、1998 年打ち上げ予定の衛星に搭載されるセンサのデータ利用プロジェクトチーム^{†12} に所属していました。そこには、地質屋さんが結構たくさんいらして、彼らは Windows NT 3.51 を使っていました^{†13}。私はデータ処理が専門なので、衛星観測データから、温度、放射率などの物理量を推定するアルゴリズム開発に携わっていました。夏の暑いところに、私がちょっとしたプログラムをつくって皆に見せた時、地質屋の人たちから、「GUI つくれ！」というリクエストが来ました。私のプログラムは、パフォーマンス重視で、無駄なモンは一切つけないハードボイルドなものです。コマンドラインから引数をとって、テキストを標準出力に出すだけのものです。そういうポリシーでプログラム作成してるのに、GUI だなんて! と断ろうとした^{†14} んですけど、我々が利用していた WS に httpd がインストールされていて、内部的に利用^{†15} していたことに気がつきました。これは、個人のプログラムやデータをプロジェクトの各メンバに利用してもらおうという意識^{†16} で運用されていました。よくよく考えてみると、web を見る (Mosaic とか netscape といった web クライアントを使う) というのは、web クライアントが用意した GUI を使っているんですね。ということは、GUI の部分は web クライアントにまかせて、GUI ベースでないプログラムに、HTML 言語で書かれた httpd とのインターフェースをかぶせて GUI をつけてしまうという連係プレーを使えば、GUI に無駄な労力を割かなくて済みますね。

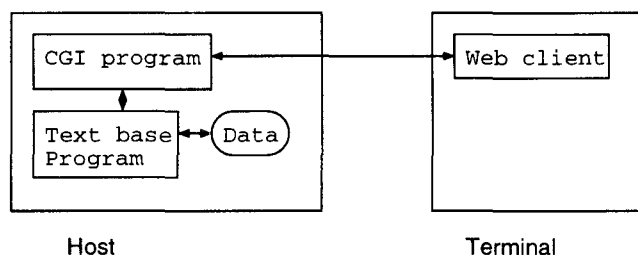


図 2: Schematic of the web based GUI

左の図は、web を用いた GUI の模式図です。CGI(Common Gateway Interface) というのが HTML でかかれた GUI のプログラムです。この模式図の場合、データは Host におかれて、Host 側のプログラムで処理され、結果が Terminal の web クライアントによって表示されます。私が JPL で作成したのはこの形式です。

これ以外にも、Host 側からプログラムとデータを転送し、Client 側でプログラムを実行させるタイプの実装もできます^{†17} が、私は使いませんでした^{†18} ので今回は説明をしません。

というわけで、例をあげて説明してみましょう。以下のプログラムは、 $y = x^2 - a/x$ を反復させて $\sqrt[3]{a}$ を求めるプログラムです。コマンドライン引数として、 a 、収束半径、初期値の 3 つを取ります。

^{†12} <http://asterweb.jpl.nasa.gov>

^{†13} 前回書いたように、私は UNIX を利用していました。

^{†14} WS に login して、コマンドを打ち込んで実行してもらおうと思ってました。

^{†15} 我々が所属していたサブネットからのアクセスを許して、それ以外からのアクセスを禁止するような設定のことです。

^{†16} XX のプログラムを俺の web から持って行ってよ、という会話が交わされていました。良いアイデアですね。

^{†17} JAVA などはこの形式みたいです。

^{†18} だって JAVA 知らねえもん。

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* y = x**2 - a/x */
int main(int argc, char *argv[])
{
    double Xn;
    double EPS, a;
    double Fx, Fdx;
    int cou;

    if(argc!=4)
    {
        fprintf(stderr, "rt2 a EPS X0\n");
        exit(1);
    }
    a=atof(argv[1]);
    EPS=atof(argv[2]);
    Xn=atof(argv[3]);

    cou=0;
    while(1)
    {
        Fx=Xn*Xn-a/Xn;
        Fdx= 2.0*Xn+a/(Xn*Xn);
        printf("%d %le %le\n", cou, Xn, Fx);
        if((fabs(Fx) < EPS) || (fabs(Fdx) < EPS)) break;
        Xn= Xn-Fx/Fdx;
        cou++;
    }
}

```

実行ファイル名を `rt3` とした場合の実行例が以下のようになります。

```

coke:matsu>rt3 2 1.0e-06 2
0 2.000000e+00 3.000000e+00
1 1.333333e+00 2.777778e-01
2 1.260073e+00 5.753184e-04
3 1.259921e+00 5.596412e-12

```

この例では、 $\sqrt[3]{2}$ を求めるのに、初期値を2、収束半径を 1×10^{-6} としたもので、反復毎に反復回数、 $\sqrt[3]{2}$ の近似値、 $y = x^2 - a/x$ を表示しています。これに web ベースの GUI をかぶせてみましょう。

3.2.1 GUIの設計

まずは、見た目からいきましょう。「あのなあ、計算機ってなあ、プロンプトにキーボードで入力すんだよ、このタコ!」という気持ちをぐっと押さえて、下のような画面を考えてみました。

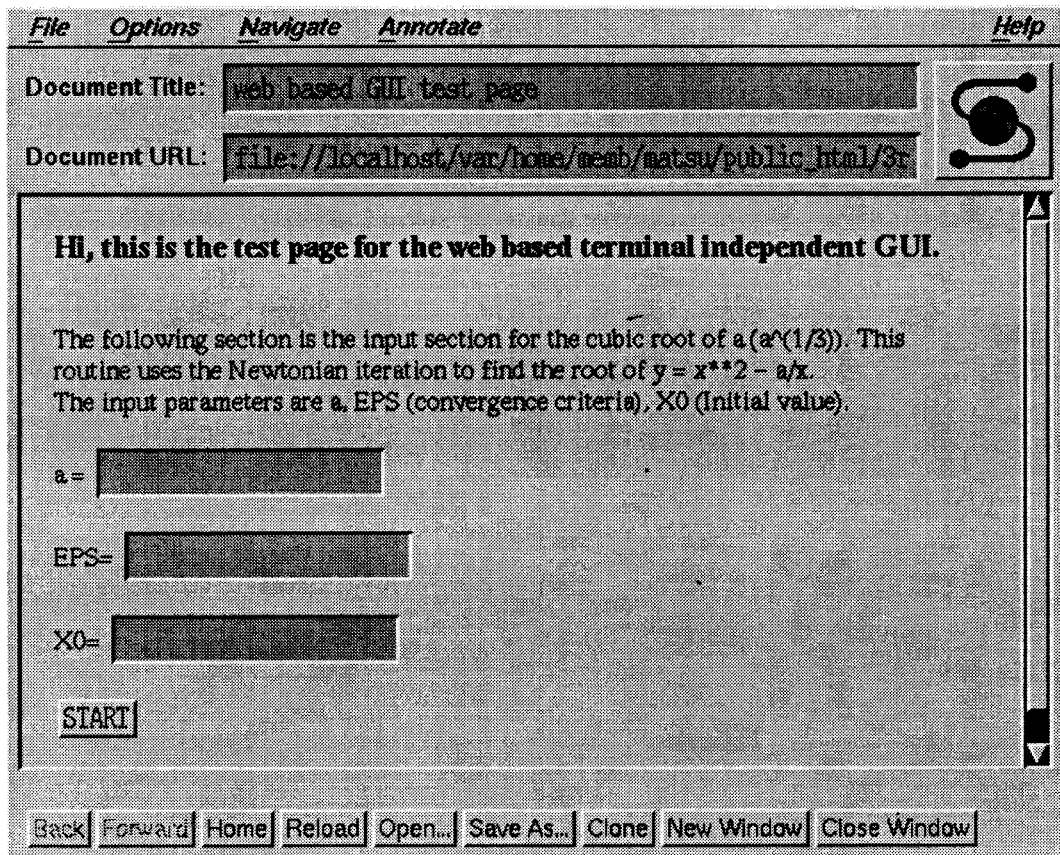


図 3: Input GUI

これを実現する HTML は以下のようなものです。

```
<html>
<title> web based GUI test page </title>
<h2> Hi, this is the test page for the web based terminal independent GUI. </h2>
<br>The following section is the input section for the cubic root of a ( $a^{1/3}$ ).
This routine uses the Newtonian iteration to find the root of  $y = x^2 - a/x$ . <br>
The input parameters are a, EPS (convergence criteria), X0 (Initial value).
<form action="cgi-bin/3rt.cgi" method="GET">
<P> a = <input type="text" name="a">
<P> EPS= <input type="text" name="EPS">
<P> X0= <input type="text" name="X0">
<P> <input type="submit" value="START">
</form>
</html>
```

割と簡単でしょ。留学してたもので HTML 本が手元になくて、オンラインドキュメントや、他人のページを view source して私でも作れました。キーは、`form` のところですね。input というのがありますが、これが、Client 側に入力パラメータのウィンドウを作って、パラメータを取ってくるための仕掛けです。

3.2.2 CGI の設計

Client 側の GUI が出来ましたので、Host 側のプログラムにパラメータを渡して実行させ、結果を Client へ送る仕掛けを作りましょう。GUI 側の HTML ファイル中に、

```
<form action="cgi-bin/3rt.cgi" method="GET">
```

という記述がありましたけども、ここで指定されている `cgi-bin/3rt.cgi` というファイルが、実行ファイルになります。この場合は自分の web ページを置くディレクトリの下での `cgi-bin` というサブディレクトリの下に `3rt.cgi` という名前の実行ファイルを作成することになります。

では `3rt.cgi` の中身を考えてみましょう。答から先に書きますと、下のようになります。

```
#!/bin/sh
echo "Content-type: text/html"
echo ""
#

eval `/usr/local/bin/cgiparse -form`
#
echo "<pre>"
echo "Cubic root of " $FORM_a
echo "Initial value: " $FORM_X0
echo "Conversion criteria: " $FORM_EPS
echo ""
echo "# trial_value x*x-x/a"
./rt3 $FORM_a $FORM_EPS $FORM_X0
echo "</pre>"
```

最初の 2 行は、表示用の HTML の始めの部分です。おまじないだと思ってください。次の `cgiparse` の文は、GUI の HTML ファイルで 3 つのパラメータを入力させましたけども、それらを、`$FORM_xx` (`xx` は `name` で指定した名前) という環境変数に入力してくれるプログラムです。下から 2 行目が私が作成した Newton 法のプログラムです。結果を以下に示します。

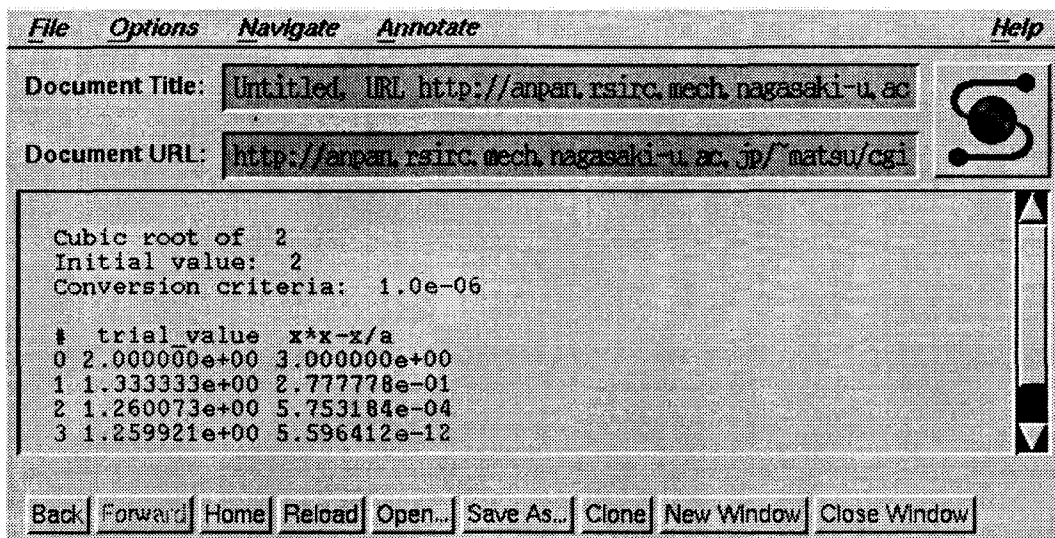


図 4: Result

実は、これに似たことを 95 年度の卒論でやらせたんですが、諮問会の人に、なんで便利なのか？ということが理解してもらえませんでした。機械科だったんでしょがないんですけど、ちと学生が

可哀想でした。ごめんね。> ○○君、××君。

4 XX 独立を疎外するもの

このあたりで、ちょっと視点を変えてみましょう。いろんな XX 独立を話しましたが、このコンセプトが定着しにくいのはなぜでしょう？ UNCOL (CPU 独立) を例にとって話をしてみましょう。ソフト屋から見れば UNCOL ってとってもありがたいコンセプトなんですよ。だって、言語毎に機械語よりもわかりやすい UNCOL へのコンパイラ (トランスレータ) を作っておけばいいわけですからね。極論すれば、N88-BASIC の中間言語みたいな感じで OK なわけですよ。でも、CPU 屋からみたらどうでしょうか？ せっかく高速な CPU のアイデアを考案しても、中間言語である UNCOL にそれを活かす機能が無かったらせっかくの努力が無駄になりますよね。つまり、個々の CPU メーカーの思惑 (ソフト屋からみたら我侭) が、UNCOL をツブしてしまったわけです。極論すれば、XX 独立というのは、共通仕様ですから、身勝手な奴がいるとうまくいきません。

5 おわりに：センタ独立

もう一度、XX 独立の話です。うちの研究室は、1993 年からセンタ独立というコンセプトで、研究室のネットワークを管理しています。何だそれ？ といわれるかもしれませんが、うちの研究室はなるべく情報処理センタに迷惑をかけないように、ということを中心に掛けています。具体的には、ワークステーションを導入して、研究室の教官、学生に独自アカウントを発行する、サブネットを導入し、不用なパケットを基幹線に流さない、学生のネットワーク教育を研究室内で実施する、等々です。つまり、センタへは、回線確保、メール/ニュースの配送、ネームサービス (98 年度からこれもうちでやる予定) だけをやってもらって、あとはこっちでやりますよ、という意識です。なんでまた？ といわれるかもしれませんが、センタにおんぶに抱っこというのは怖いんですよ、だってさ、うちのセンタって管理してるのが実質的に二人ですからね、あんまり物頼むとパンクしちゃうかも知れないでしょ。それに我々も、我々の流儀というのがありますので、それを全学の皆さんに押し付ける気もありません。ただ、我々はこういう方針ですけども、センタを無視するということはしていません。ネットワーク上でのエチケットやセンタの方々が定めたルールは守っているつもりです。つまり、我々のネットワーク上の活動において、センタの方々の手を極力患わせないようにしてます。センタの方々には、その分、回線確保に力をそそいでもらいたいと願っております。

我々は、このコンセプトを「センタ独立」と呼んでいます。我々は、サブネット切ったり、いろいろハデにやってますけど、PC 1 台しかない方でも、それなりのセンタ独立になれます。難しいことじゃないんですよ。センタ独立を疎外するものって何だと思いますか？ UNCOL の場合と同じで、各自の我侭なんですよ。自分の PC が調子悪くても、「ネットワークがおかしい!」とか言ってませんか？ センタの人たちが、不正アクセス対策をいろいろとしてくれているのに、自分のパスワードを堂々とホワイトボードに書いてませんか？ 自分のところの学生に「ネットワーク教育」のとかかりもあたえずに、「うちのセンタはユーザ教育をしない」とかワメいていませんか？ 申請無しで PC をネットワークにつないでませんか？ 何も考えずにディスクやプリンタの共有をやってませんか？ ひとつのアカウントを複数人で共有していませんか？^{†19} こういうことをすると、センタの方々の仕事^{†20} が増えてしまうんですよ。確かに Mainframe の頃は、特別な人でないと計算機が利用できませんでした。でも今は違うよね。少しはこの方向のことも考えてみてはいかががでしょうか？

^{†19} これらは、すべて学内で見かけたことです。内心「犬!」と叫んでましたけども

^{†20} なくてもいい仕事です。