UNIX Communications

1997年 調布祭 百萬石

Using MMA

mma-active 著

ikidou 編

NUTSHELL HANDBOOKS

メールのフィルタを書くときのメモ

uirou@mma.club.uec.ac.jp

1997/11/18

1 フィルタって？

電子メールと言う物はもう一般的な物として定着してきている。UNIX の世界でメールを配送するプログラムと言えば、sendmail(1) はかなり一般的なものである。その sendmail(1) では、.forward と言うファイルによってユーザサイドでメールの配送を制御できるようになっている。このフィルタを用いる事によって、メールを転送する時の条件を設定したり、うまく作れば ML 紛いの事もユーザサイドでできるようになる。ここでは、その.forward ファイルを使ってのメールのフィルタの書き方についての手助けになりそうなことを考えてみる。

2 普通に存在するフィルタ

フィルタと言っても、普通の用途であればすでに良くできた物が存在する。例えば、mh には slocal(1) というコマンドがあるし、smms という perl を用いた物も存在する。まずは、自分の望むフィルタがあるかどうか、周りに聞いてみることから始めると良いだろう。とりあえず、これらの使い方はマニュアルを見てもらう事とする。

3 フィルタを動作させるには

フィルタを動作させるには、.forward ファイルに、"でくくってこのように書く。

```
"| /home/uirou/script/mail_filter"
```

要するに、"|"でくくり、フィルタプログラムへのフルパスを書くのである。この形式からもわかるように、フィルタプログラムの指定は sh(1) に食わせられ、処理される。

4 フィルタにはどのような入力があるのか？

フィルタにはどのような入力があるのか？当然の疑問である。そこはそれ。このようにフィルタさせてみよう。

```
\uirou,"| /bin/cat >> /tmp/uirou.mail"
```

こうして自分にメールを一個打ってみよう。こんな感じで送られて来るとおもう。

```
From uirou@din.or.jp  Sun Nov 16 01:19:18 1997
Return-Path: uirou@din.or.jp
Received: from localhost (localhost [127.0.0.1])
 by neo.iimura.mma.fun.club.uec.ac.jp
 (8.8.5/3.5Wb5/neo-19970705) with ESMTP id BAA04710 for
 <uirou@neo.iimura>; Sun, 16 Nov 1997 01:19:18 +0900
To: uirou@neo.iimura.mma.fun.club.uec.ac.jp
Subject: test mail
X-Mailer: Mew version 1.90 on Emacs 19.28  / Mule 2.3 (SUETSUMUHANA)
X-Pgp-Fingerprint: 89 6D 89 68 9A 23 DF 8F  B0 43 DF 05 DA 8D 37 02
Mime-Version: 1.0
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit
Message-Id: <19971116011918P.uirou@neo.iimura.fan.club.uec.ac.jp>
```

```
Date: Sun, 16 Nov 1997 01:19:18 +0900
From: Iimura Takuji <uirou@din.or.jp>
X-Dispatcher: imput version 970826
Lines: 2

this is test mail.
これはテストメールです。
```

これでわかるのは、最初の行は From からはじまり、その後はヘッダ部が続き、リターンだけの行があって、データ部が続くということである。

5 ヘッダについて

とりあえず、ここでメールのヘッダについて簡単にお扱いしておく。

To	送る先のアドレス
Cc	これも送る先のアドレス
From	このメールを送った人のアドレス
Sender	From のアドレスが書き変わった時などに、本来のアドレスがつけられる
Reply-To	From とは別のアドレスに返信を望む時につけられる
Subject	メールの題名
Date	そのメールが送信された時刻
Message-Id	そのメール固有の番号。unique である事が求められる
In-Reply-To	どのメールへの返信かを表す
Received	配送経路が記録される
Return-Path	エラーなどがあった場合は、ここに転送される
X-*	X- に文字列がつづくフィールドは定義されていない

1 ヘッダ部の書かれ方

メールにフィルタをしたいといった場合、ヘッダを解析しなければならないことが多い。通常、ヘッダ部は一行で表されるが、長いヘッダは折り畳まれることがある。折り畳まれたヘッダは、次のように行頭に空白文字が来ることになっている。

```
From: "Iimura Takuji"
      <uirou@din.or.jp>
```

実際、このようなヘッダは良く見かける。このように、行頭に空白文字があり、その後に文字列が来るものは、その前の行の改行と空白文字を一つの空白文字として扱うことによって折り畳みを単一行表記に置き換えることが出来る。要するに、さっきの例で言えば、

```
From: "Iimura Takuji" <uirou@din.or.jp>
```

と言う例が、

```
From: "Iimura
 Takuji"
 <uirou@din.or.jp>
```

などとも書けることを意味している。

2 perl での実装サンプル

ということで、perl でこのようなヘッダを一つの連想配列として読み込む方法を示しておく。

```
undef(%headder);
$now_headder = '';
while(<>){
    last if(/^[\r\n]*$/);
    if(s/^\s+//){
        chop($headder{$now_headder});
        $headder{$now_headder} .= " $_";
    }else{
        s/^([^:]+):\s+//;
        $now_headder = $1;
        $headder{$now_headder} = $_;
    }
}
```

このように、改行だけの行が来れば、ヘッダ部はおしまい。行頭に空白があれば、前の行とくっつけるようにすれば、良い。

ただし、このスクリプトをそのまま使うと、同じヘッダが複数あった場合、(例えば Received など) 最後に書いてあったものしか残らない。注意すること。

また、ヘッダ部の書かれ方は、RFC822 で定義されている。

6 lock file を作ろう

フィルタを書いていると、大抵何らかのファイルに log を残したいと思ったり、新たなファイルを作成したいと思うものである。そのとき、lock file を作らなければいけなくなる。なぜなら、フィルタはメール一通につき一つづつ起動され、処理が行われるので、もしも複数のメールが同時に到着したとき、複数のフィルタが同時に同じファイルにアクセスしようとするかもしれないからである。

このような同一ファイルに同時にアクセスすることを避けるために、lock file と呼ばれるファイルを作り、そのファイルが存在するときはすでに他のプロセスがファイルに書き込みをしているということに決めておけば、複数のプロセスが同時に起動しても、同じファイルに同時にアクセスするということを避けることが出来る。

一般的な lock file 作成方法は、symlink(2) システムコールを用いたものである。この方法はすぐに思いつく open(2) システムコールを用いた lock file の作成方法と比べ、次の点で優れている。

open(2) を用いて、lock file を作成しようとすると、stat(2) などを用いて lock file の存在を確認してから、lock file を作成することになる。この場合では、stat(2) などを用いて lock file の存在を確認したときには lock file は存在しなかったかもしれないが、open(2) を実行するまでの間に lock file が作成されなかったと言う保証は無い。

その点、symlink(2) を用いた場合は、lock file の存在確認と同時に、lock file の作成をシステムが行ってくれる。また、この方法は NFS 上でもきちんと動作する。

1 perl での実装サンプル

この方法についても、perl での実装のサンプルをつけておく。

```
sub lock{
    local($tmp) = @_;
    $lock::lockfile = $tmp if($tmp ne '');
    local($retry) = 15;
    if($lock::lockfile eq ''){
        local($myname) = $0;
```

```
        $myname =~ s#(.*/)##;
        $lock::lockfile = "$myname.LOCK";
    }
    while(!symlink("/", $lock::lockfile)){
        &err("sorry. lockfile timeout.\n") if(--$retry <= 0);
        sleep(2);
    }
    $lock::locked = 1;
}
sub unlock{
    unlink($lock::lockfile);
}
```

このコードを使った場合、サブルーチン lock を呼ぶことによって、lock file を作成し、すでにある場合は数秒待ちつつ何度かトライする。また、引数に lock file を指定しなかった場合は、現在のディレクトリに、"自分の名前.lock" をいうファイルを作ろうとする。サブルーチン unlock は、ただ lock file を unlink しているだけである。

また、このままでは、複数の lock file を作ることは出来ない。注意すること。

7 /var/mail への書き込み

.forward で指定され、起動したフィルタは、そのユーザの UID で動作するので、当然、自分のメールボックス (要するに/var/mail/username) へも書き込み可能である。あなたはフィルタで自分のメールボックスへメールを追加したいと思うかもしれない。

そういった場合、少しだけ知っておく必要のある事柄がある。

1 /var/mail の lock

/var/mail/username への書き込みを行う場合、このファイルも lock する必要がある。なぜなら、前言ったように、複数のメールが同時に来たときに、書き出されるファイルが同じだからである。

この lock の場合には、もう一つ気をつけなければいけないことがある。/var/mail/username を扱う MUA(Mail User Agent) も同時に/var/mail/usarname を書き換える可能性があるからである。この場合、通常はどのようにして lock をかけているのかいまいち筆者は自信が無い。どうも sendmail(8) が mail を/var/mail/username にセーブするときには mail.local(8) を使っているようである。

それによれば、mail.local に、-l オプションを用いた場合は、"username.lock" というファイルを lock file に、そうでなかった場合には、flock(2) システムコールを用いての lock を行うようである。flock(2) についてはここでは説明しない。man を読むこと。

ということで、/var/mail/username を lock する場合は、/var/mail/username.lock と言うファイルを作り、それを lock file とし、さらに flock(2) を用いて lock しておいたほうがよさそうである。

8 おっと、忘れるところだった

実は、メールのフィルタの出力は sendmail によって監視されている。また、フィルタの終了コードも sendmail は監視している。この終了コードが "0" 以外であったとき、sendmail はフィルタのエラーであると言うことで、エラーメールが "postmaster" に送信される。これは非常に迷惑で、また恥ずかしいことでもあるので、あまり`exit(1)` などとやるのはおすすめできない。

そのようなことを回避するためにも、とりあえず、一番最初に示したような方法で、フィルタに入力される形式をファイルにして取っておき、テスト中のフィルタプログラムに cat(1) で渡す、といった方法を

取って実験することをおすすめする。

9 終りに

結局のところ、メールのフィルタはlock fileとの戦いが主である。それ以外のところは読者諸氏のアイディアでなんとでもなるであろう。確かに、MIMEなどと戦い始めるとちょっと面倒そうではあるが、、

参考文献

各種manページ
メールヘッダとフィールド - インターネットメールの注意点
(http://ux01.so-net.or.jp/~hat/imail/imail06.html)
RFC822 の日本語訳
(http://www.imasy.or.jp/~hanakos/rfc/822.html)
NetBSD-1.2G の src/libexec/mail.local/*

市民、CGI という言葉を知っていますか?

uirou@mma.club.uec.ac.jp

879778800 == (1997/11/18 JST)

Abstract

コンピュータ (以下、C#) ようこそ。トラブルシューター。

ユーザ (以下、%) 私のことですか？

C# その通りです、市民。Alpha Complex(==MMA) においては、市民 (==部員) はみな、トラブルシューターです。

% トラブルシューターとは何ですか？

C# トラブルシューターはコンピュータに仕える忠実な市民です。コンピュータの命令をうけてチームを組み、ミッションを遂行します。

% ふむふむ。

C# それと、命令される任務とは別に、常に心がけておかなければならない事が一つあります。

% なんですか？

C# 反逆者を発見し、報告し、場合によってはその場で処刑することです。

% 反逆者？

C# 親友であるコンピュータを裏切り、Alpha Complex を破壊しようと企む、悪意のある連中やプロセスの事です。彼らは忠実な市民に化けており、陰謀をめぐらしています。反逆者を見つけ出し、これを処刑しなければなりません。

% ふむふむ。

C# また、全ての 猿は皆、反逆者です。MMA はみな、遺伝的に完璧であり、この Alpha Complex はそうした完璧な人間のみに許されたユートピアです。遺伝的に欠陥を持ったミュータントはみな猿になります。これらは、見つけ出され、根絶されなければなりません。

% なるほど。私も猿を見つけるべく努力します。

C# RFC に違反するプログラムやその製作者、及び使用者も皆、反逆者です。コンピュータに公式に認められていない規則はみな、RFC に違反しており、それを作るもの、及び使うものはコンピュータに対して陰謀をめぐらしていると判断されます。かような反逆者は、やはり狩り出され、処刑されなければなりません。

% だいたいわかりました。私は、反逆者、猿、RFC に違反するプログラムなどを見つけだし、処刑すれば良いのですね？

C# その通りです、市民！ Serve the Computer! Computer is your Friend!

C# ところで市民、あなたは WWW で猿になった事はありませんよね？RFC に違反する M$のプログラムを使ったことは？

% .... い、いえ、コンピュータ。私はそのようなことは今まで一度もしたことがありません...

1 市民、CGI という言葉を知っていますか？

C# 市民、あなたは CGI と言う言葉を聞いた事がありますか？

% いいえ、コンピュータ。私は CGI という言葉を聞いた事はありません。

C# CGI とは WWW におけるサービスに、プログラムを導入させられる技術です。これを使う事によって市民達の見られる情報は、より柔軟性を増し、使い勝手が向上するでしょう。ところで、市民。あなたは CGI プログラムをしてみたいと思いますか？

% いいえ、コンピュータ。私は CGI を組んでみたいとは思いません。

C# 良くない答えです。市民。CGI プログラムはユーザの好評を得ています。実際、CGI プログラミングをする事は楽しい事です。楽しい事、すなわち幸福です。あなたは幸福になりたくないと言うのですか？

% いえ。私は幸福になりたいとは思っていますが、CGI プログラミングをすることが幸福であるとは思えません。

C# 市民、コンピュータが幸福と言うのです。あなたはコンピュータを疑うのですか？疑うとことはもちろん反....

% い、いえ！違います！私は CGI のプログラムをしてみたいとは思っているのですが、それをする知識が無いのです。このままでは CGI プログラミングをすることができません。

C# わかりました。市民。あなたは、あなたのセキュリティ・クリアランスで閲覧が許可されている情報を使って CGI プログラミングを学びなさい。それは実際、楽しい事ですよ。

2 市民、CGI を使うまえに知っておくべき事があります。

C# 市民、あなたはまず、CGI プログラムと言う物が、どのようにして動いているかを知らなければなりません。まずは、HTTP について教えましょう。まずは、どこぞの Web server の http ポートに telnet してみなさい。

% はい。コンピュータ。

```
% telnet www.alpha.complex http
Trying xxx.xxx.xxx.xxx...
Connected to www.alpha.complex.
Escape character is '^]'.
```

C# では、そこで、

```
GET / HTTP/1.0
```

と入力し、リターンキーを二回押してみなさい。

% なにやら文字が表示されて接続が切られました。そういえば、これって、alpha.complex のトップページの HTML ですね。

C# そうです。さて、ここで送られて来たデータの頭が、以下のような形式になっている事に注意しなさい。

```
HTTP/1.0 200 OK
Date: Sat, 15 Nov 1997 12:57:04 GMT
Server: AlphaComplexHTTPD/1.0.0
Content-type: text/html
Content-length: 2477
Last-modified: Thu, 13 Nov 1997 22:04:09 GMT
```

これは、ヘッダと呼ばれる物で、最初の行は HTTP のステータス、次からの行はそれぞれのヘッダです。この後にただ改行だけの行があるのにも注意しなさい。この改行だけの行で、ヘッダ部と、データ部をわけているのです。

% データ部とは？

C# HTML の内容の事です。さっきの例で言えば、alpha.complex のトップページの内容がそうです。

% なるほど！要するに、HTTP では、データのやりとりをするのに、ヘッダ部とデータ部があり、実際にユーザが見る部分はデータ部であると言う事ですね。

C# 実に良く理解しましたね。市民。

3 市民、CGI プログラムは HTTP を喋らなければなりません。

C# さて、市民、CGI プログラムにおいては、より柔軟性のあるデータを提供するために、HTTP ヘッダをつける事が許されています。実際、HTTP ヘッダをつけられる事によって、ユーザの利便は上昇しています。

% ということは、逆に言うと、HTTP ヘッダをつけなければいけないという事ですか？

C# そうです。

% ところで、コンピュータ、CGI ではどのようにしてユーザ (ブラウザ) にデータを与えれば良いのですか？

C# 良い質問です。市民。CGI プログラムは、標準出力に出力された文字がそのままユーザ (ブラウザ) に渡されるようになっています。そこで、次のような perl スクリプトが CGI プログラムとなり得ます。

```
#! /usr/local/bin/perl
print <<"END";
Content-type: text/html
Welcome to Alpha Complex!
END
```

% なるほど！最初の Content-type:は HTML ヘッダで、Welcome to ... の部分はデータ部であると言う事ですね！で、その間にある改行だけの行が、それをわけているということですね？

C# 実に良く理解していますね。市民。

% ところで.... コンピュータ。

C# なんでしょうか？

% ヘッダ部にある、Content-type というものについてですが、これは一体どういう事を表しているのですか？

C# 実に良い質問です。市民。これは、text で送られた html の形式の文章である事を示しています。

% でも.... データ部は HTML で書かれていませんね。

C# ..... 市民、何か問題でも？

% い、いえ。別に.....

4 市民、CGI を動作させるには、httpd に対して実行権限が与えられなければなりません

C# 市民、通常 CGI プログラムは "何々.cgi" という名前でファイルにセーブされます。

% どうしてですか？

C# 市民、コンピュータがそう言うのです。あなたはコンピュータを疑うのですか？

% め、めっそうもない。

C# それでは、市民、あなたの CGI プログラムを "何々.cgi" と言う名前でセーブして、ブラウザで表示させてみなさい。

% はい。コンピュータ。

```
% cd ~/public_html
% mv ~/script/test_cgi.pl test.cgi
% lynx http://www.mma/~MMA/test.cgi
```

あれ？Forbidden と言われてしまいました。

C# あぁ、言い忘れていました、市民。CGI プログラムは httpd 権限で実行されます。あなた以外のユーザでも実行できるようにパーミッションを変えてみなさい。

% はい。コンピュータ。

```
% chmod go+rx test.cgi
% lynx http://www.mma/~MMA/test.cgi
```

あ。今度は行けました。

C# おめでとう！市民。これで CGI プログラムを動作させることが出来るようになりましたね！

% ところでコンピュータ、この Content-type:の事ですが、これを text/html ではなく、image/gif 等にした場合、イメージを直接送りつけても良いのですか？

C# 素晴らしい！良い質問です。もちろん。それは可能です。実際は以下のようになるでしょう。

```
#!/bin/sh
echo Content-type: image/gif
echo
cat hoge.gif
```

% なるほど！Content-type:で指定したデータを、その後に出力すれば良いのですね。

C# 実に良い答えです。市民。もう基本的な CGI プログラムはマスターしましたね！

5 市民、CGIプログラムは値を受け取ることが出来ます

C# 市民、CGIプログラムは情報を出すだけでなく、受け取ることも出来ます。

% ほう！それはどのようにすれば良いのですか？

C# それは実際、簡単なことです。CGIプログラムにおいて、与えられる情報は、環境変数もしくは標準入力から得られます。

% “もしくは”、とは？

C# HTTPの使い方によって、二種類の受渡し方法があるからです。このあたりで良く使われる環境変数について説明を聞きたいと思うでしょう。

REQUEST_METHOD クライアントの要求メソッドが入ります。GET, POST, HEAD, PUT, DELETE, LINK, UNLINK のいづれかが入ります。

SCRIPT_NAME 呼び出されたCGIスクリプトのパス名がURL形式で入ります。

QUERY_STRING REQUEST_METHODがGETであった場合、ここに、URLエンコードされたクライアントからのデータが入ります。

REMOTE_USER スクリプトが実行されたサーバで認証機能が使われている場合、user名が入ります。

HTTP_USER_AGENT スクリプトを呼び出した相手のブラウザのプログラム名、バージョン番号などが入ります。一般的にsoftware/version library/versionという形式のようです。

HTTP_PEFERER このスクリプトを読み込む直前に参照していたURLが入ります。ブラウザによっては送って来ないものもあります。

この、REQUEST_METHODがGETである場合、QUERY_STRINGにクライアントからのデータが入るわけです。

% ちょっといいですか？

C# なんですか？

% URLエンコードとは、どういうことですか？

C# とりあえず、ASCII文字列のみでデータが送りたいので、ある特定の規則に乗っ取ってエンコードされていると理解しなさい。

% .... わかりました。

C# perlでは、次のようにしてユーザからの入力を連想配列に入れることが出来ます。

```
sub read_input
{
    local ($buffer, @pairs, $pair, $name, $value, %FORM);
    # Read in text
    $ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
    if ($ENV{'REQUEST_METHOD'} eq "POST")
    {
        read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    } else
    {
        $buffer = $ENV{'QUERY_STRING'};
```

```
    }
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs)
    {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%(..)/pack("C", hex($1))/eg;
        $name =~ tr/+/ /;
        $name =~ s/%(..)/pack("C", hex($1))/eg;
        $FORM{$name} = $value;
    }
    %FORM;
}
```

この例では、REQUEST_METHOD を見て、それが POST であるかで、QUERY_STRING を読むか、標準入力から読むかを判断しています。また、次のようにして URL エンコードを表現できます。

```
sub urlencode ($) {
    my $dst, $x, $y, $i;
    my @unsafe = qw({ } | \ ^ ~ [ ] < > % / ? : @ = &);
    $dst = '';
    push @unsafe, chr(96), chr(35), chr(39), chr(32);
    for ($i = 0; $i < length($_[0]); $i ++) {
        $x = substr($_[0], $i, 1);
        $y = ord($x);
        if ($y < 0x20 ||$y > 0x80 ||
            grep($_ eq $x, @unsafe)) {
            $x = '%' . sprintf('%X', $y);
        }
        $dst .= $x;
    }
    $dst;
}
```

% ありがとうございます。コンピュータ。でも、どうやってユーザからデータを CGI プログラムに与えるのですか？

6 市民、あなたは FORM を使ったことが無いのですか？

C# 市民、あなたは FORM を使ったことが無いのですか？しかたがありません。FORM についても少しだけ触れることにしましょう。

% お願いします。

C# FORM タグは、通常、以下のような形式で使用されます。

```
<FORM ACTION="cgi_script.cgi">
  あなたの名前:<INPUT TYPE="text" NAME="name"><BR>
  あなたのパスワード: <INPUT TYPE="password" NAME="pass">
  これからすることを選びなさい。
  <SELECT NAME="what-to-do">
    <OPTION> ミッションを受け取る
    <OPTION SELECTED> コンピュータに奉仕する
    <OPTION> 反逆者を見つける
    <OPTION> 猿を見つける
    <OPTION> RFC 違反プログラムを見つける
  </SELECT>
</FORM>
```

このように、FORM では、ACTION 引数に動作させたい CGI プログラムの名前を与えます。そして INPUT タグや SELECT タグなどでユーザの入力を促すわけです。

% なるほど！私は今まで FORM タグの存在意味を理解していなかったようです。確かに、これを使えばユーザからの入力を受け取ることが出来ますね！

C# そうです。とりあえず FORM まで説明していると時間がいくらあっても足りないので、このあたりでやめておくことにします。

% .... はい。コンピュータの言われることはいつも正論です。いつも、自分の考えのあさはかさを思い知らされます。

7 市民、CGI プログラムではファイルの lock は重要です

C# 市民、CGI プログラムにおいてファイルに結果を書き出させようとした場合、lock file を使った排他制御が重要です。

% それはどうしてですか？コンピュータ。

C# 考えてもみて下さい。市民。何かファイルに情報を書き出す CGI プログラムがあったとします。

% ふむふむ。

C# それが複数のユーザに、同時に呼び出されたらどうなると思いますか？同じファイルに書き出す処理が複数のプロセスによって同時に行われるところに注意しなさい。

% あ、最悪ファイルの中身が壊れてしまいますね！

C# そうです。そこで file lock が重要なのです。

% わかりました。でも、ファイルの lock をするときに、flock(2) システムコールを使ってはいけないのですか？

C# 良い質問です。市民。しかし、flock(2) システムコールは全ての NFS で正常に動作するとは限りません。そこで、lock file を使った排他制御が必要なわけです。

% なるほど。CGI プログラムは、しばしば NFS 上に存在していますからね。

C# そうです。そこで、ファイルの lock の方法ですが、symlink(2) を用いた方法が良いでしょう。

% それは何故ですか？

C# symlink(2) を用いた場合、ファイルの存在確認とファイルの作成が同時に出来るからです。この方法については、深くは説明しないこととします。

% あの.... ヒントだけでも....

C# 百万石を良く読みなさい。

% はぁ...

8 市民、CGIプログラムの作り方は理解できましたか？

C# 市民、あなたにはCGIプログラムの基本を教えました。これでもう完璧なCGIを組むことが出来るでしょう。

% あの.... コンピュータ。

C# なんですか？

% これだけの情報ではまともなCGIを書くことは出来そうに無いのですが....

C# あなたはコンピュータの言うことが信じられないのですか？コンピュータを信じないこと、それはすなわち反逆です。

% い、いえ！違います。私はただ、まだあまりうまくCGIプログラムが書けないと言いたかっただけです。

C# 市民、あなたは私の説明したCGIの書き方を聞きましたね？

% ... はい。

C# それでもまだ書けないと言うのですか？

% い.... いえ！違います！

C# そういえばさっきWWWで猿になったことは無いと言っていましたね？

% .... はい。

C# では、なぜalpha.complexのトップページを知っていたり、HTMLについての知識を持っているのですか？

% .....

C# 市民、隠し事は、すなわち反逆です。反逆者は処刑せねばなりません.....

レーザー音

C# 大丈夫です。今度のクローンはきっとうまくやってくれます.....

参考文献

http://w3.lab.kdd.co.jp/technotes/WWW/

http://www01.u-page.so-net.or.jp/da2/babahide/library/paranoia.txt

MULTIPLY を X に移植してみよう計画

uirou@mma.club.uec.ac.jp

1997/11/18

1 それは現れた (10 月 31 日)

いつものように某 ML の流量に呆れながらも、つらつらとメールを読んでいると、MULTIPLY と呼ばれるロジック対戦 Game が、紹介されていた。(アドレスは、http://www.stat.hit-u.ac.jp/ tom/である) とりあえずどんなものか見てみることにした。

どうも、8 種類の用意されたロボットの動作を、移動、索敵、計算、条件分岐、などのパレットを用いて、20x20 の領域にプログラムし、それを戦わせると言うゲームのようだ。そしてそれは無償で配られていた。素晴らしい事である。

しかし！しかしである。これがまた、Windows な環境でしか動かなかったのだ。そして、その時僕は Windows を走らせられる環境に居なかった。僕の Windows は HDD が壊れてから見捨てられていたのである。

僕は、MULTIPLY で遊んでみる事をあきらめた。

2 環境は、整うべくして整った (11 月 02 日)

某 Emulator のレジスト版が何故か手元にある、、

これで遊べるようにするには、Windows(==MS-DOS) の入っている HDD を修復し、Windows(==MS-DOS) が使える環境にならなければならない。幸い、連休なので、Windows 本体を入れる事にした。

3 MULTIPLY 初体験。そして、、(11 月 03 日)

ちょうど Windows な環境も整ったわけであるので、MULTIPLY を遊んでみることにした。これは ML に入らなければならない、、当然、ML に入っての第一声は、「移植は考えてないんですか？」である。:P

4 色良い返事。良い事ばかりとは限らない (11 月 05 日)

作者であるところの下村さんによれば、「移植してくれる人がいれば、ぜひ」とのことである。

どーせ VC++なソースなのだろうから、X 関連の面倒臭い事さえ何とかなれば、何とでもなるだろうと思えた。とゆーことで、「おいら、やっても良いかも、、」と書いてしまった。

5 かくして、ソースは現れた (11 月 08 日)

何故か、作者である所の下村さんからの巨大なメールを受け取る。中身はどうも MULTIPLY のソースらしい。

ということでデータファイルの解析を開始。

6 MFC との戦い (11 月 09 日)

データファイルは MFC(MICROSOFT (R) FOUNDATION CLASSES) の CArchive クラスを用い、Serialize という機能でもってファイルに書き出されていた。どうも、class をそのまま書き出す事が出来るらしい。

悲しい事に、僕は C++でプログラムと言うのを組んだ事が無い。そのため、移植作業には C を用いて行う事にした。しかし、C には class の概念は無い。とうぜん、MFC なんてものも標準でついているわけが無い。前途多難である。

7 MFC からの解放。(11 月 12 日)

どうしても MFC の CArchive について良くわからず、意味もなく日数が進んで行く、、

そんなある日、作者様が、「データファイルはテキストにしましょう」とおっしゃって下さった。CArchive なんてのと戦うよりテキストと戦った方が何倍も楽だ！しかし、実装は次のバージョンでは見送られるらしい。しかたが無いので内部処理及び X での画面処理についての部分に力を入れる事にした。

8 内部処理の移植は完了。しかし、、(11 月 16 日)

この日、内部の画面表示、ファイルからの読み込み以外のいわば内部処理はこれの移植を完了した。しかし、X のプログラムにツールキットを使う事にしたら、いまいちツールキットの使い方をわきまえていなかったのに気がつく。少なくともボタンとかを作らなくてすむのだから、ツールキットのほうが何倍も楽なはずなのだが、、

9 そして未来、XMultiply は実現するのか？(??月??日)

元祖、Windows 用 MULTIPLY は、DirectX を用いグラフィック、音共に、きちんと処理されている。が、僕の X 制御能力はたかが知れているので、画面はちゃちい物になるであろう。ましてや、音を出すなんてどうすればよいものやら、、

参考文献

MSVC 2.?? の (MICROSOFT (R) FOUNDATION CLASSES) ソース MULTIPLY 1.09 のソース

STkプログラミング

So-Miya (宮宗正樹)

1 はじめに

Tcl/Tk というお手軽なウィンドウプログラミングの行える言語がある。これは **Tcl** というスクリプト状の言語からウィンドウ操作を行う **Tk** を操作する構造になっている。しかし、この言語ではスクリプトということもあり、あまり大規模なプログラミングには向いていない。(それに、実行速度も結構遅い。)

Stk は、**Scheme** という **lisp** の一種から **Tk** を扱えるようにしたものである。インタプリタであるが、**lisp** というだけあって結構実行速度が期待できる。

ここでは、**STk** でのプログラミングのため、まずはじめに **Scheme** と **Tcl/Tk** について簡単に述べ、つぎに **STk** におけるプログラミングについて述べたいと思う。

1 Scheme

Scheme は、**lisp** と呼ばれるリスト言語の一形態であり、基本的に **Emacs Lisp(elisp)** と言語構造は同じであり、また基本命令も同じ名前であるものが多い。

list、**vector**(配列のようなもの) などの構造体を扱え、変数の型にとらわれないプログラミングが可能である。

詳しくは **Scheme** の参考書を見てほしい。

2 Tcl/Tk

Tcl/Tk とは、ウィンドウシステム上のアプリケーションを簡単に作成することを目指したシステムであり、実行部分をスクリプトである **Tcl** で、インターフェース部分をツールキットである **Tk** で構成したものである。

ラベル、ボタン、ラジオボタン、ドロップダウンメニュー、リストボックス、キャンバス、入力フィールド、テキストフィールド、スクロールバー等が作成、表示、操作できる。また、入力装置としてマウスとキーボードが想定されている。

詳しくは Tcl/Tk の参考書を見てほしい。

2 STk

STk のバイナリおよび説明書は`http://kaolin.unice.fr/`に存在するが、日本の ftp サイトにも結構置いてある。

STk は **Scheme** に、**Tk** のインターフェースを付け加えたものである。これで作成したアプリケーションは外部プログラムを呼び出したり port を呼び出したり、font 名を指定したりと、特殊なことをしない限り、**STk** の動作するマシンと OS の上で実行させることが可能であるはずである。

1 Tk の呼び出し方法

Tcl における **Tk** の呼び出し方と比較しつつ述べる。

ウィジェットを生成、パッケージする例として、*frame f* の上に *Button b* を作成することを考える。
Tcl の場合はこうなる。

```
frame .f
button .f.b -text "Ok" -command {quit}

pack .b .f -fill x
```

これをStkの場合はこのように書く。

```
(require "Tk-classes")

(define f (make <Frame>))
(define b (make <Button> :text "Ok" :command (lambda () (quit))
                         :parent f))

(pack b f :fill "x")
```

一番最初の *(require "Tk-classes")* は Tk を統合的に扱うために最初にインクルードするものである。

Tcl では-のところを、:を先頭につけることでオプションを区別しているのがわかる。

STk では、Tcl の時のようにウィジェット名でウィジェットの親子関係を示すことができないので、:parent オプションで親ウィジェットを指定する必要がある。(省略した場合トップウィンドウを親とする。)

また、:command に関数を渡す際にはすぐさま評価されないように *lambda* を通す必要がある。これは、オプションに対して通常はすぐに評価されることを示しており、オプションを実行時に変更することが可能なことを示している。

2 電卓プログラム

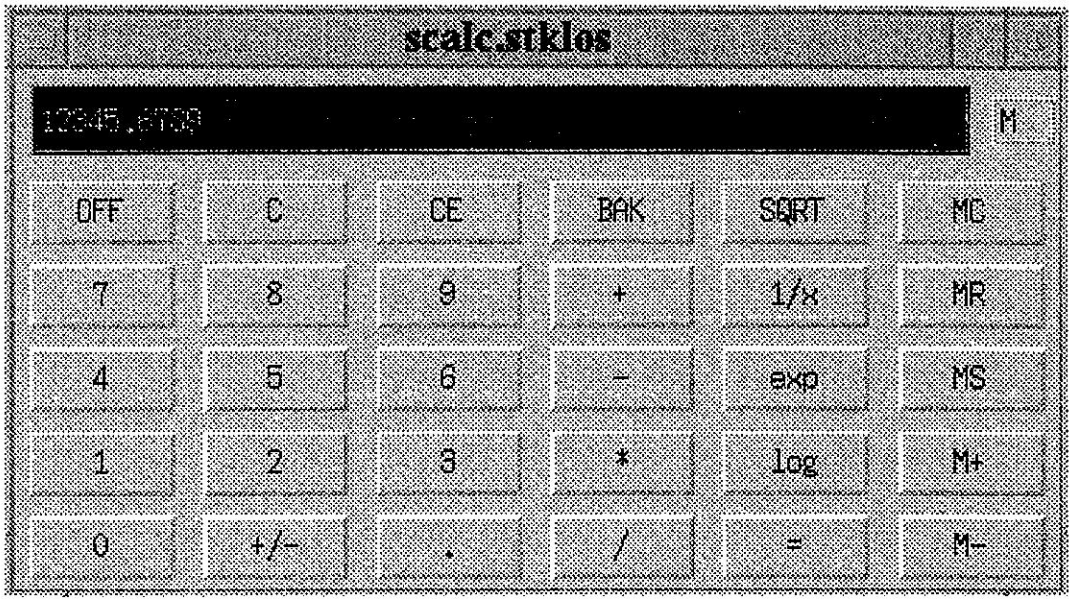
今まで勉強したことを踏まえて、STk でプログラミングを行う。題材として、電卓プログラムを作成することにした。

1 仕様

- 数値を入れ、演算子を入れて、もう一度数値を入れて"=" を押すと答えが表示されるものとする。
- 実際の電卓と同様に、数値1、演算子a、数値2、演算子b、と入れたとき、数値1と数値2を演算子aによって計算した結果が表示され、この結果とこの後に入力される数値が演算子bで計算される。
- メモリ機能を盛り込む。
- 正負を反転させることができる。
- 二項演算子だけではなく、一項演算子 (ルート等) も入れる。
- 計算のリセットと、数値入力のリセットと、数字の押し間違いの訂正機能。
- 小数点計算ができる。

2 実行状況

うまく動作することを確認した。
以下に実行結果を図示する。



3 考察

使用してみて、いちいちボタンをマウスでクリックするのは面倒であることがわかる。解決方法としては、キー入力をバインドすることである。

また、桁あふれの問題の解決や、0 で割算を行ったときなどのエラー処理を行っていない。

このプログラムはオブジェクト指向的な作りにはなってないし、マウスによるドラッグアンドドロップ、バインドによるキー入力、グラフィックの表示などを試していないが、STk の使用方法のおおよその見当はつけることができた。

3 作成した電卓プログラム [scalc.stklos]

このプログラムは`http://delegate.uec.ac.jp:8081/club/mma/~so-miya/data/scalc.stklos`に置いてある。

このプログラムはフリーウェアとする。著作権は放棄しないが、自由に改造、配布してよいものとする。また、このプログラムによっていかなる損害を被ったとしても著作権者は責任を負わないものとする。

ネットワークのおはなし

ottan
ottan@mma.club.uec.ac.jp

1 まえふり

今回はお言葉を書かなかったので、ちょっとユニマガばりにこっちでヨタ話をかいてみます。で、お題は過去に企画だおれになった記事について (笑)...

本とは今回ネタにしようと思ってたのは、『易経とデジタルとモード』と『ssh の実装と実際』という名前だけはカッコいいよさそうなやつ。前者は岩波文庫版の易経[1] に撃沈されて延期、後者は清水さんの perl の記事みたいにソースの中から例をあげて実装の素晴らしさを解説出来るレベルに行きつくまでかなり時間がかかりそうなのでこれも延期。実は、これ以前にもいろいろ書きかけの記事があって、ぼくの 200LX の A:\ txt には 5,6 個ストックされたりします (笑)。

というわけで、今回は TCP/IP の基礎ということでお茶を濁すことにします。最近、経路制御とか NAT とかやってて、もう一度基礎を見直す必要を感じたこともあってネットワークに接続されたホストを立ち上げるのに必要な最低限の知識について書いてみます。

2 歴史的経緯

今回は知らなくてもいいでしょう。興味のある人はそこら辺の本を読んで下さい。

3 TCP/IP

だいたい OS のインストーラあたりに聞かれるネットワークの設定は、

- ホストの IP アドレス
- ホストネームとドメインネーム
- そのセグメント[2]のネットマスク
- ゲートウェイ (デフォルトルート) の IP アドレス
- DNS サーバの IP アドレス

といった所です。このへんを知ってれば大丈夫でしょう。

1 IP アドレス

ホストの識別子です。

32bit 整数のアドレスであらわされ、8bit [3]づつでくぎって、172.21.139.40 というように表されます。

[1] 内容の難しさもさることながら、昭和初期の初版のため限りなく文語に近い口語でかかれてて解説を読むのも一苦労です...
[2] ゲートウェイを介さずに同じ ethernet につながっているネットワーク、ちょっと正確じゃない気もするけど
[3] この 8 bit を 1 octet と言うようです。1 byte と言った方が分かりやすそうな気もしますが、必ずしも 1 byte = 1 bit ではないのと、通信関連では伝統的に使ってるというのがあるようです。

2 ホストネームとドメインネーム

ホストの人間にわかる形での識別子です。

MMA のでは、ホストネームはインストールする人の好きに出来ます。ただあんまり長いのは嫌われるので、半角英数字で 5 文字以下ぐらいがよいようです。[4]また、ドメインネームは mma.club.uec.ac.jp.private となります。最後の private ドメインは学内ネットワークから直接は外にでられない証です。それ以外の各サブドメインの意味は分かっているものとして省略。ホストネームとドメインネームをあわせて FQDN (Fully Qualified Domain Name) と言ったりもします。

あと、 mail 関係の設定に関しては private をつけないように。private というドメインはあくまで学内でしか通用しません。

3 プライベートネットワーク

MMA に割り当てられているアドレスは、IP アドレスの不足、セキュリティー上及び政治的問題からプライベートアドレスになっています。よって、学外にアクセスするには各種アプリケーションゲートウェイを利用する必要があります。

プライベートネットワークについての詳しい話は 95 年度新歓と調布祭の百萬石の tree さんの記事と RFC を参照して下さい。

4 ネットマスク

IP アドレスとあわせて、ホストの所属するセグメントの範囲を示します。

MMA だと基本的に 255.255.255.0 ですが、最近 proxyARP をテストしている関係でインターフェイスによってはそうとも言い切れないので確認してください。

172.21.139.0/24 と言うような形式でセグメントそのものをあらわしたりもします。この場合、24bit のマスクなので 8bit であらわせる範囲のうち、172.21.139.1〜172.21.139.254 というような IP アドレスのホストを含むネットワークのセグメントということになります。また、 0 は ネットワークアドレス、255 はブロードキャストアドレスという特殊な意味をもつので普通はホストには使いません。

また最近は一般に IP アドレスが不足してきていることから、効率的にサブネットを運用するため VLSM(Variable Length Subnet Mask) 可変長サブネットを利用している事も多く 172.21.139.128/28 というようなかたちで 172.21.139.129〜172.21.139.142 を使うなんて事もよくあります。この場合は 128 がネットワークアドレスで、143 がブロードキャストアドレスになります。

ただし、旧い機器には 8bit 単位で区切ることを前提としているものもあるので注意が必要です。

5 ゲートウェイ (デフォルトルート) の IP アドレス

外のネットワークの窓口となっているホストを指定します。

厳密にはゲートウェイとデフォルトルートは別物なのですが、末端のネットワークでは基本的に外部のどこかと通信したい時は上のネットワーク通じているゲートウェイに放り投げて任せとけばいいので深く考える必要は無いです。

でも、まぁ、普段お目にかかれない BGP4 とか OSPF2 とかその辺も面白そうなんですけどね。その辺まで理解できたらとうして www.altavista.digital.com までパケットが行って、返事が帰って来るか納得できそうです。

[4] karesansui はボツになりました

6　DNS サーバの IP アドレス

人にやさしいドメインネームとコンピュータにやさいいIP アドレスを変換してくれるサービスを行っているサーバを指定します。

そのうち慣れて来ると人にやさしくない IP アドレスで大抵すませてしまうようになってしまうんですけどね。

4　おしまい

これぐらい知ってれば、ネットワークインストールやインストール後の設定はだいたいできると思います。あとは慣れの問題でしょう。

Web で電卓

南條 理裕
core@mma.club.uec.ac.jp
1997/11/14

1 まえおき

今回書くにあたって、書くに値する程のネタも私には残念ながら見当たらずに困っていたところ、そういえば、HTML で SSI とか最近やっているしな〜。とか思ってじゃあ CGI でもやってみようと安易に考えたのが間違いでした。

2 初心者の無謀な話

どのコンピューターにも必ず電卓のようなプログラムがあるわけではなく、jed にはもちろん標準では xcalc はないわけで、dc も使い方がわかりません。
そこで、前に授業で作った計算式を解くプログラムをホームページ上から動かせばどこにいても使えて便利かなと、思ったわけです。

そのプログラムとは、pascal で書かれたプログラムで、1+3+4*5 など、人間には分かる計算式はコンピュータには理解しにくいわけで、それを計算するために、面倒な手続きで 13+45*+のような後置記法に直して、それを単純な計算の手続きで解くという代物です。
ゆえにデータさえプログラムに送ってしまえば答えがでて来るからそれを表示すればよいと単純に思ったものの、
その為には

- FORM タグの入力をプログラムに送る必要がある
- そのまえに、hoge=1+2+%2B となっているのを解読しなくてはいけない
- 解読したデータを CGI が計算用プログラムに送る
- 結果をページに返さなくてはいけない

などの面倒なことがありました。
perl などが書けない私は即座に shell スクリプトに飛びつき、本に載っているのを参考にして打ち込んだのですが、私の設定がおかしいらしく、
CC (cc.uec.ac.jp) では、CGI は

```
<FORM ACTION=" http://raven.cc.uec.ac.jp/~ n9612124/docalc.sh"
METHOD="GET">
```

のように書いて実行すると netscape では保存のダイアログがでて来てしまい、実行されません。(mma では大丈夫でしたが。)
なぜ、cc にこだわるかと言えば、私は C 言語がろくに使えないので、プログラムは pascal で書かれたものを使用したからです。
要するに、jed でコンパイルしたプログラムを、ftp で mma のマシンに持って来るわけにもいかず、じゃあ mma でコンパイルすればいいと思って、p2c をコンパイルしてみましたが、設定がおかしく、どうもコンパイルされません。

そこで、送られて来るデータ QUERY_ STRING は SSI で実行した shell スクリプトが、拾ってプログラムに送るということにしました。

3 つまり、

つまり、データの入力フォームは、

```
----------------------------------------
<FORM ACTION="result.html" METHOD="GET">
<INPUT NAME="param" SIZE="39"><BR><P>
<INPUT TYPE="submit" VALUE="送信">
<INPUT TYPE="reset" VALUE="書き直す">
</FORM>
----------------------------------------
```

の用になっており、入力してボタンを押すと result.html が開きます。また、得られたデータは、環境変数の QUERY_ STRING に入っています。
次に結果を表示するページを次のようにします。

```
-------------------------------
計算結果は、
<FONT SIZE="6">
<!--#exec cmd="sh docalc"-->
</FONT>
です。
-------------------------------
```

docalc は shell スクリプトなのですが、その内容は、

```
-------------------------------------------------------
#decoder,calc のプログラムを絶対パスつきで変数に指定
DECODER='/home/edu/a-96j/n9612124/public_html/decoder'
CALC='/home/edu/a-96j/n9612124/public_html/calc'

#''='' の部分を FS にして''='' 以降の部分を取り出す。
PREINPUT=`echo $QUERY_STRING | awk -F= '{print($2)}'`
#データを解読する。
INPUT=`echo $PREINPUT | $DECODER`
#解読したデータを計算プログラムに入力
OUTPUT=`echo $INPUT | $CALC`
#結果を出力
echo "$INPUT = $OUTPUT"
-------------------------------------------------------
```

と、なっています。(実際には日本語のコメントは入っていません。動かなくなるので。)
decoder は Pascal で書かれたプログラムで、演算子と括弧をデコードし、空白を入力したときに入ってしまう+を取り除いたりするものです。
実際どのようになるかといえば、まず、ページ上から 1+2*4(3+4)-1 を入力します。
このデータは環境変数 QUERY_ STRING に param=1%2B2*4%283%2B4%29-1 のように入っているので、awk で、1%2B2*4%283%2B4%29-1 の部分だけ取り出したあと、それを 1+2*4(3+4)-1 に戻すために decoder に送ります。
decoder でデコードされた後、データを計算プログラム calc に送ります。
最後に echo で計算結果を元の計算式と一緒にページに戻してあげて終りです。
そうすると、ページには

```
-----------------------------------
計算結果が出ました。
計算結果は、 1+2*4(3+4)-1 = 56 です。
-----------------------------------
```

のように表示され、めでたしめでたしです。

4 あとがき

MMA の部員でありながらも私にはこのようなへなちょこなモノしか作れませんが許して下さい。ちなみに、このプログラムは http://raven.cc.uec.ac.jp/ñ9612124/calc.html から実行することができます。

各種入門書の書評

hasegawa

前書き

電通大に入って初めてBSD系のOSに触り、未だに修行中の身なので、技術力を必要とする物は書けません。そこで、最近読んだ本について印象に残った物の書評をかきます。なお、書いている人はまだまだ、初心者であるので、内容についての不慮は御了承ください。

プロフェショナルBSD

砂原秀樹、石井秀治、植原啓介、林周志 共著：アスキー：1994年

BSD-OS全般について解説しています。UNIXの動作の仕組みから、アプリケーションの利用についてまで書いています。特にUNIXの内部構造についてはかなりのページ数を割いて、説明しています。逆にアプリケーション(EmacsやMHなど)や、ツールなどの説明はあまり詳しくは行っておりませんが一通り説明されており、多くの情報を広く浅く説明すると言った感じです。著者によると、これからシステム管理者を目指そうという人向けの本だそうです。

バークレイUNIXとANSI C入門

J.Hodge 著：トッパン：1996年

図書館で見付けた本。定価が書いていないので、もしかしたら、何かの教科書かも知れない。第一印象はすごく教科書的な内容だという感じで、コンピュータを使う上での倫理やマナーの解説に重点をおくなど、コンピュータリテラシーの内容みたいである。コマンドラインでのBSDの各種アプリケーション(findとかgrep等)の使い方を、一通り解説している。また、C入門の部分ではC言語の解説より実際の開発を目的とした解説に、重点が置かれている。C言語自体の説明はあまり詳しくない。なお、X WindowSystemには、全く触れていない。

祝入門TCP/IP

?著：ソフトバンク：1996年

部室にあった黄色い表紙に黒いタイトル文字の派手な本。TCP/IPを中心にネットワークの基礎を解説しています。比喩や例えをつかって、分かりやすく説明しようとしています。空回りしている所も結構ありますが、基本的に分かりやすいです。

入門XWindow

松田晃一、暦本純一 著：アスキー：1993年

XWinsowSystem の基本的な動作原理、その使用方法について、解説している。発行自体が少し昔なのでウィンドウマネジャーの解説は twm だけです。

FreeBSD 徹底入門

細川達巳、三田吉郎 他 4 名著：翔泳社：1997 年

よくある CD 付き PC-UNIX 入門書。内容は洗練されていて PC-UNIX を導入、管理するにはどうすればよいのか、とりあえず本を読めば分かるように作られています。根本的なコマンドの解説はありませんが、PC-UNIX を導入しようとしている人にはそんな物必要ないでしょう。

LiNUX 入門キット

林雅人 著：秀和システム：1997 年

よくある CD 付き PC-UNIX インストール本。RedHat Linux が付属している。2 枚の CD が付属しているので、ソフトが豊富である。ただし、本の中身自体はかなり about なつくりで、HD の DOS 占有領域縮小して、新たに HD に空きを作るツール FIPS の使い方さえ、述べられていません。その他の主に設定に関する説明も、X WindowSystem がまともに動く環境を想定しているので、X WindowSystem がうごかなかったらお手上げです。よって、この本はまるまる 1 台ハードディスクが空いているか、FIPS 等をつかえて、自分の持っているビデオカード等の細かい仕様 (clock や chip-set など) がわかっている人向きです。でも、そういう人はパッケージ買った方が速いです。最近、Red Hat の 4.2 が入ったやつが出ましたが、基本的に中身が同じで進歩してない。徹底入門の方を買った方が無難。もし、Linux がいいっ、と言うなら、多分この本が、置いてある近くに、Slack Ware 付きの本があるので、そっちの方が良い。

C 言語入門

?著：アスキー：?年

名前の通り、C の入門書です。他の言語を学んでいなくても、理解できるレベルの内容です。C 言語をできるだけ平易に分かりやすく説明していて、他の言語を学んでいなくても、理解できるレベルの内容です。C を学び始めた人や。K&R(プログラム言語 C 共立出版) を読んで、理解できなかった人などに薦めます。また、標準入出力関数についての説明にやけに力が入っています。この本を読めば、とりあえず一通りの事ができるようになると思います。

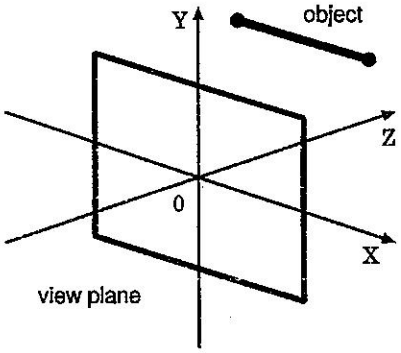


図 1: 視平面座標系

射影変換の話

佐藤 喬
ikidou@mma.club.uec.ac.jp

Abstract

ある次元空間の象をより低い次元空間に移すことを、一般的に射影 (projection) といいます。ここでは 3DCG で必要になってくる、3 次元から 2 次元への射影変換の話をします。

1 はじめに

3D ゲームといえば、昔はワイヤーフレームで精一杯というものでしたが、最近はマシンパワー、レタリング技術の向上により、ポリゴン表現で、グリグリ動いて見てると酔ってしまいそうなゲーム[1]がたくさんでています。

ところで、ゲームをしているとあたり前すぎて気にしないかもしれませんが、現在、我々が一般に使用するディスプレイは、2 次元の情報のみを表示でき、3 次元の情報を直接表示[2]することができません。そこで、3 次元の画像を表現するために一度 2 次元に射影変換する必要があるのです。

2 視平面座表系

では、実際にどのような操作をするのか考えていきましょう。まず、2 次元に変換される物体 (object) が存在する座標系 (coordinate system) — 視平面座標系 (view plane coordinate system) という — を考えます。座標系は左手座標系としましょう。中指と親指がそれぞれ X, Y 軸に、人差指は奥行きを表す Z 軸となります。$z = 0$ となる XY 平面を視平面 (view plane) とし、物体の射影先にします。

さて、おおまかな設定は上記のとおりとします。これはあくまで ikidou が頭の中で理解しやすくしたものなので、射影変換の内容を理解したら、自分の好みに変更しても O.K. です。

[1] SS のガンダム外伝、PS の攻殻機動隊等がよさげです。
[2] 3D ホログラフ (で名前良かったっけ?) などのようにという意味で。

3 射影変換

射影変換には 2 つの種類がありまして、1 つが透視射影 (perspective projection)、もう 1 つが平行射影 (parallel porjection) と呼ばれるものです。

1 透視変換

この透視変換は、小、中学校の美術でならった透視画法を計算で行うものです。つまり遠くは小さく、近くは大きく見えるということを利用するのです。透視変換の特徴は

- 物体は遠くになるほど小さくなり、無限遠方において数個の点に収束する
- 処理が少々大変になるが、遠近感が表現しやすい。
- 平行関係が変換後には保証されない。

といったところです。

まず、視平面を眺める仮想的な目を決めます。これを射影中心 (center of projection) といいます。では、視平面に射影していきましょう。視覚的に言えば、この作業は物体の頂点と射影中心を通る直線と、視平面との交点群を求めることにより実現します。

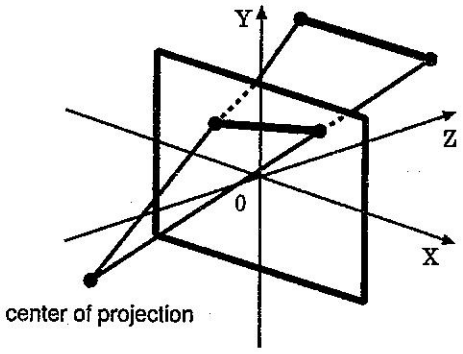


図 2: 透視変換

交点を求める計算過程は高校の学力があれば解けるハズ[3]です。

射影中心を $C(x_c, y_c, z_c)$、物体の頂点を $P(x_p, y_p, z_p)$、視平面上の射影された点を $V(x_v, y_v)$ とします。直線の式を媒介変数表示すると、

$$\begin{cases} x &= x_p + t(x_c - x_p) \\ y &= y_p + t(y_c - y_p) \\ z &= z_p + t(z_c - z_p) \end{cases}$$

[3] 代数幾何を忘れていた ikidou はまる 1 日悩みましたが...

ここで、視平面の式は $z=0$ なのだから、

$$t=-\frac{z_p}{z_c-z_p}$$

となり、これより Vを求めると以下のようになります。

$$\begin{cases} x_v &= x_p+\dfrac{x_c-x_p}{z_c-z_p}z_c \\ y_v &= y_p+\dfrac{y_c-y_p}{z_c-z_p}z_c \end{cases}$$

ちなみに

A 1 点透視

B 2 点透視

C 3 点透視

といった種類があります。

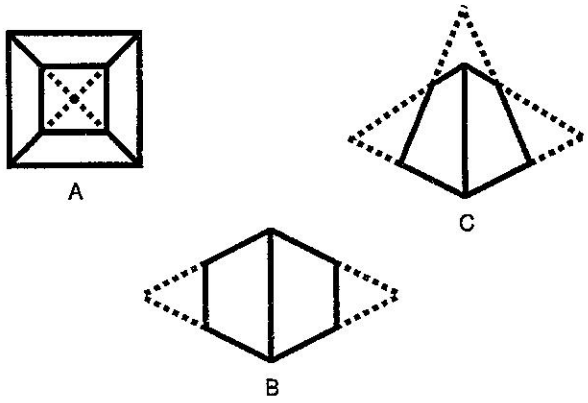


図 3: A:1 点透視 B:2 点透視 C:3 点透視

ところで、$z_p-z_c=0$ になったらヤバイんでないかい? と思うかもしれませんが、このような点はクリッピング作業により回避します。

2 平行射影

透視射影において射影中心が視平面に近付くと視野角がひろまり、ちょうど魚眼レンズをのぞいたようになり、遠くになると、遠近による変形が少なくなっていきます。さらに射影中心を無限遠方にもっていき、方向情報のみを残して射影することを平行射影とよび、この方向情報を射影方向 (direction of projection) といいます。

平行射影の特徴は

- 物体の遠近による変形はない。

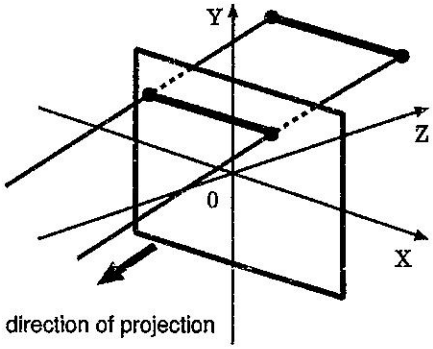


図 4: 平行射影

- 処理が簡単だが、遠近間が無い。
- 平行関係が変換後も保証される。

のような感じになります。

視平面への射影は、さきの透視射影とほぼ同じ、むしろ計算は簡単になっています。さきの射影中心 $C(x_c, y_c, z_c)$ を射影方向$\vec{d}(a, b, c)$ として、計算すると[4]以下のようになります。

$$\begin{cases} x_v &= x_p - \frac{a}{c} z_p \\ y_v &= y_p - \frac{b}{c} z_p \end{cases}$$

4 奥行きを考える

射影変換をほどこすと奥行きである z座標が欠落[5]してしまいます。しかし、この変換により欠落した奥行き情報が、必要になることがあるのです。例えば、ポリゴンの表裏判定でつかう法線ベクトルを知りたいときや、ポリゴンの重なりを判定するときなどです。そんなの変換前のを使えばいいじゃん、と考えるかもしれません。確かに平行射影の場合はそれですむのですが、透視射影になると平面が平面として変換されなくなってしまいます。

これを防ぐために以下の条件を満たす zの条件を考えます。

- 変換前 $z_1 \leq z_2 \leq z_3$なら変換後も $z'_1 \leq z'_2 \leq z'_3$の関係をたもつ。
- 平面は平面として変換される。

この条件を満たす zの値の変換方法は、理屈を完全に理解しきれていず、安直に書くのはいやなので、省略[6]してしまいます。

[4]過程は省略しちゃいます。
[5]$z = 0$になります。
[6]逃げちゃってますね。ほんとはここが一番大事。

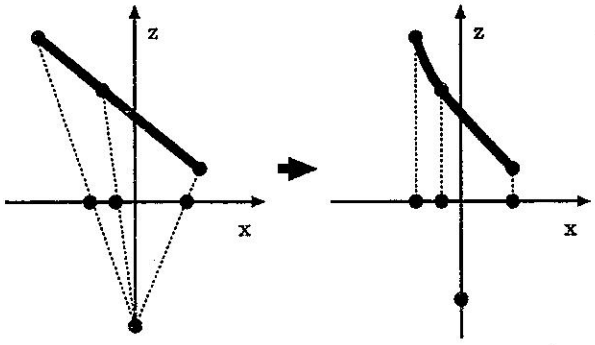


図 5: 平面として変換されない

5 最後に

3 次元画像はこのほかにも変換行列や座標系のネスト、光と色、クリッピング操作などと考えるとおもしろいことがたくさんあり、なかなか楽しめます。

参考文献

- 中島正幸, 川合慧:グラフィックスとマンマシンシステム, 岩波講座ソフトウェア科学 10, 岩波書店,1995
- 図書館にあった CG 本の一部分コピー (書名失念)

編集中の話

ikidou

OP-A 長官、各個人の原稿を別々にコンパイルする分はいいのですが、root file からまとめてコンパイルすると section 等がおかしくなります。

長官 なにっ、ちゃんと必要なカウンタの初期化をするマクロをつくったではないか!

OP-A はあ、確かに section 等のカウンタは初期化されているのですが、表示がアルファベットになってしまいます。

OP-B オーッ! ホントデース! デモ、ホワァイ?

OP-C へんね、そんなことするコマンドはつかってないのに...

長官 理由がわからん以上、みつけるしかない! 各自検証を開始してくれ。

全員 了解っ!

2 時間経過...

OP-A 長官、どうやら`\appendix`の後おかしくなっているようです。おそらくこれが表示の定義を変更しているのではないでしょうか?

長官 よし、では初期化のマクロに`\renewcommand`をいれて修正しろ。

OP-A 了解しました。

長官 くっ、しかしこの進行状況では調布祭までに完成せんぞ。

博士 ... 長官、もはや弾丸 X しかないんじゃないのか?

長官 だっ、弾丸 X! しかし...

OP-D 弾丸 X! 偽善`^H^H`勇者に流れる V [1] リキッドを刺激することで一時的に限界以上に力をあたえるあの...

OP-C まってください! ikidou は 1 週間ほど風邪をこじらせ、しかも寝違いをおこして弱っています。いま弾丸 X を使用すれば反動で体がもちません!

博士 ああ、できることならワシも使いたくはない。じゃが、こうでもせんことには百萬石ができんぞ。

OP-A 長官、ikidou が現実逃避モードにはいり、花組対戦コラムスを始めました。おそらくパズルモード 51—90 面制覇に挑戦するもようです。

長官 いかんっ! ここで猿になられたら終りだ、弾丸 X 使用承認っ!!

OP-C ちょっ長官っ!

長官 ikidou の中にねむる虚栄心にかけてみよう、彼はりっぱな偽善`^H^H`勇者だ!

百萬石を完成させるためにはなたれた弾丸 X。そして...

[1] V—vanity(虚栄心)

長官 なんとか百萬石は完成したようだな...

博士 ああ... じゃが...

OP-A 長官、ikidou の V ストーン反応がレーダーから消えました...

OP-C いやーーっ!!!

こうして百萬石は、自尊と虚栄の精神を肥やしにすることにより完成した。しかし生活パターンが夜型になってしまう、という手痛い犠牲を払ってしまった ikidou に明日はあるのか? 朝型を取り戻すのだ ikidou!

NEXT

もう一度徹夜をして朝型にする。 TEX、LATEX の勉強をもっとしとく。

↑これが、勝利のカギだっ!!!