

# ブラウザを利用した演習授業の一方法

—人文系における計算機支援学習への提言—

吉 村 誠

How to conduct the browser-used classwork:an approach  
- A Compute-assisted learning in humanities science

Makoto Yoshimura

キーワード：計算機支援、大学教育、授業実践、国文学

## 1. はじめに

最近の Internet 利用の普及はめざましいものがある。まだ転送量などシステム上で改善すべき問題はあるが、利用者は急増している。企業、大学、研究機関はもとより、初等、中等教育機関や個人においてもその利用は着実に増加の傾向にある。network の利用は、電子 mail やファイル転送など様々な利用形態があるが、最も多いのは Home Page による情報授受の形態であろう。netscape や、internet exploreなどを主としたブラウザと呼ばれるソフトでネットワークを通して相手の Home Page (HTML document) を読むことにより、情報の発信や受信が行われているというのが、現在の標準の形となりつつある。

Home Page による情報提供の流れは、一方的に静的に送信者側からの情報を流すのみならず、動画を取り込んだ動的なより視覚性を強調する表現方面と、受信者側がデータ入力することにより、求めている情報を引き出すような相互通信機能を持たせる方向に動きつつある。後者で我々の身近なところでは、図書館の蔵書検索システムや、yahoo などの Home Page 検索機能などがそれであろう。そこで注目されるのがこの相互通信機能である。この機能が以下に述べる演習授業に見られる具体的方法に応用出来るからである。

現在、筆者が担当している文学作品についての演習の授業形態は、あらかじめ指定された報告者が作品の解釈を基調報告することにより、他の学習者との討議を通してより高度な作品理解の追求を行うという形が通例である。

これを情報という形に置き換えると、報告者の作業は情報整理にあると言ってよい。演習授業において、報告者が必要な情報を取得し整理して、それを基本として自己の考えをまとめ、他人に伝えるという基本作業を行っていると考えたとするならば、一方で情報の取得、整理を目的として、近年著しく発達したコンピュータを効果的に利用するという事は可能である。従来はそれをデータベースやワープロなど、スタンドアロンの適宜利用する方法に限られていたように思われる。しかしその方法は、結果としては指導者と学習者間の個人的な情報授受に限られ、学習者全員の討議というようなネットワークの

構築は不可能であったと言ってよい。

学習者全員の討議という形態をとった情報交換は、計算機利用という点では電子掲示板などのプログラムで実現できる。しかしこれには、教師がかなり高度なプログラム作成の技術を持っていなければならない、学習者がその使い方への習得も必要になったりして、特殊性を伴ってしまう。またそれに必要な情報取得のためには、別のデータベース検索の方法学習も必要になり、それぞれが個々に独立した形で学習しなければならないという煩雑さが存在してしまう。そしてまたシステム構築の複雑さもともなう。パソコンを中心としたシステムを考えた場合、サーバーの設置が必要で、そうなるセキュリティ問題が生じてくる。

そうした中で、ブラウザを利用する形態での電子掲示板の作成や情報検索は、教師のプログラム作成も容易であり、操作性も一般的である。バージョンアップもあり、ブラウザ自身が過渡的であるかも知れないが、現時点ではそれを利用した操作性は他にも応用できる一般性を持っていると考えてよい。また検索に利用される原データは直接のアクセスを許さないで、セキュリティは保たれていると言える。

そこで本稿では、筆者が実践している国文学の演習授業の形態を情報という概念で置き換える試みの中で、ブラウザを利用する形態でのシステム構築の具体例を提示してみたい。

## 2. 「情報」としての演習の形態

演習における計算機支援は、「情報」そのものへの実践的理解を基礎とした上に構築されるものであって、計算機の操作性を学習するものではない。従ってワープロやデータベースなどの既存のソフトの習得は排除される。また「情報」といっても内容は多義にわたるので、ここでは筆者の専門とする「万葉集」という古典文学作品を「情報」という概念に置き換えた形で提示する。

「万葉集」の個々の歌を中心として、その文学的理解を集団討議的形式で学習する場合、まずテーマに基づいた研究史を把握し、問題点を整理した報告を中心に、学習者の意見の形成が基本となる。そして学習者同士で討議を行い、新たな疑問を発見したり、それへの解決方法を模索するという作業が必要になる。その過程においては、隣接諸科学の力、例えば国語学や歴史学が必要になったり、或いは方法論への自覚も伴うことになる。

方法的自覚は、残念ながら計算機で処理出来るものではない。また文学への理解も学習者個人に帰する問題であり、計算機支援による効果は期待できない。しかし研究史への把握や、語法などの調査においては、計算機による利用は従来の手段よりもはるかに効果的であると評価してよいものであろう。ただし残念ながらまだ完全な研究史情報が計算機の上において稼働しているわけではない。ここで述べることは理想であることを前提にした。ただし万葉集テキストそのものや、一部注釈書レベルの内容、論文目録は既に計算機上で稼働可能の状態である。特に万葉集テキストは、著作権問題をクリアし、筆者によって free text として一般的に network 上で公開、配付しているものである。

計算機を導入した場合は、次のような方法が考えられる。学習者は、課題となる歌の基礎調査を行う。これにはデータベースを利用する。そしてそれを掲示板に提示する。掲示板を見た別の学習者が意見や質問を書き込む。それに対して課題を提示した学習者が再度調査し回答する。以下再び疑問点や質問があれば、書き込んでいく。ということを繰り返

し行う。教師は適宜アドバイスや回答の不足分を補う形で書き込む。箇条書きでその作業を示すと、以下のようになろう。

1. 万葉集の歌作品の基礎調査

具体的作業：テキスト情報（本文異同、校合）

注釈（訓読、解釈）

論文による問題点解決（国語学、歴史学、万葉集）

データベースによる調査、まとめ 掲示板に提示する。

2. 他の学習者による疑問、意見などの書き込み

3. 疑問点の追求。再びデータベースを用いて調査する。

調査結果に基づいて考察。結果を提示する。

4. 質問者も独自に調査。論証を踏まえた上で討議する。

この形態は、元来口頭で行ってきた方法である。しかし従来と異なる点は、複数の学習者に対して、同時に課題を与え、課題別に討議を行うということが可能な点である。従来は単一的に行ってきた討議を同時進行的に進めることが出来る。また報告や討議は、授業時に限らなくともよく、いつでも参加することが出来る。特に自宅にパソコンがあり、network 利用が可能な状態であれば、自宅からでも討議に加わることが出来る。

具体例として、ここに10人の学習者からなる一つの授業をシュミレートしてみる。まず10人にそれぞれ課題を与える。学習者はそれぞれ与えられた課題に対して基礎調査を行う。まとめたものを課題別に準備された掲示板に記入する。学習者がお互いに課題別の掲示板を開いて、意見を書き込むという形態である。個人別にしなくともグループ別にすることも可能である。具体的には以下のようなイメージになるであろう。ブラウザ上で見える形で示してみる。ただし引用しているhtml documentのfile name及びC programのfile nameは、仮名である。

man\_sitsugi.html

-----  
万葉集演習課題

- |            |        |         |
|------------|--------|---------|
| 1. 卷1・18番歌 | 掲示板を見る | 意見を書き込む |
| 2. 卷1・21番歌 | 掲示板を見る | 意見を書き込む |
| 3. 卷1・25番歌 | 掲示板を見る | 意見を書き込む |
| 4. 卷1・29番歌 | 掲示板を見る | 意見を書き込む |

・  
・  
・

-----  
これに対して、例えば18番の歌を開いて、意見を書き込む。

man2\_mes.html

-----  
Wed Sep 3 22:10:21 JST 1997 吉田太郎 tarou@kokugo.yamaguchi-u.ac.jp

基本報告

卷1.0018

皇太子答御歌 [明日香宮御宇天皇諡曰天武天皇]

紫のほへる妹を憎くあらば人妻故に我れ恋ひめやも

(基本説明省略)

Wed Sep 3 22:10:47 JST 1997 山口花子 hanako@kokugo.yamaguchi-u.ac.jp

「故に」という語法は、現代語と意味が異なるように感じられるのですが、何故、解説のような理解が可能なのですか。

Wed Sep 3 22:11:27 JST 1997 国語二郎 jirou@kokugo.yamaguchi-u.ac.jp

皇太子とあるのは、「皇太弟」とあるのが本来的だと報告にありましたが、それならば、何故題詞はそのような記述になっているのでしょうか。

-----  
必要があれば、その場で調査する。用例調査は、万葉集テキスト検索を用いる。

Wed Sep 3 22:13:34 JST 1997 吉田太郎 tarou@kokugo.yamaguchi-u.ac.jp

山口花子さんへの回答

「故に」という語法は、万葉集中全部で16例あります。

具体的に用例を示します。

02/0122H01 大船の泊つる泊りのたゆたひに物思ひ瘦せぬ人の子故に  
02/0167H16 高知りまして 朝言に 御言問はさぬ 日月の 数多くなりぬれ そこ故に  
02/0194H06 そこ故に 慰めかねて けだしくも 逢ふやと思ひて  
02/0196H19 たゆたふ見れば 慰もる 心もあらず そこ故に 為むすべ知れや  
02/0200H01 ひさかたの天知らしぬる君故に日月も知らず恋ひわたるかも  
03/0305H01 かく故に見じと言ふものを楽浪の旧き都を見せつつもとな  
03/0372H04 思ひぞ我がする 逢はぬ子故に  
03/0411H01 我妹子がやどの橋いと近く植ゑてし故にならずはやまじ  
04/0599H01 朝霧のおほに相見し人故に命死ぬべく恋ひわたるかも  
07/1301H01 海神の手に巻き持てる玉故に磯の浦廻に潜きするかも  
07/1320H01 水底に沈く白玉誰が故に心尽して我が思はなくに  
07/1370H01 はなはだも降らぬ雨故にはたつみいたくな行きそ人の知るべく  
08/1576H01 この岡に小鹿踏み起しうかねらひかもかもすらく君故にこそ  
08/1629H05 一日一夜も 離り居て 嘆き恋ふらむ ここ思へば 胸こそ痛き そこ故に  
09/1772H01 後れ居て我れはや恋ひむ印南野の秋萩見つ去なむ子故に  
現代語と同じ「～だから」という原因理由の意味を持つ用例もありますが、それでは解釈上意味が通らないものもあります。それらのものから考えますと、この大海人皇子歌の「故に」の語法的解釈は、やはり諸注釈の述べるように、「～だからといって」という意味になるでしょう。

-----  
このような授業形態になる。最後にそれぞれの課題について、レポートをまとめさせる。レポートも、エディタで簡単に引用が可能である。

さらに地域的な調査が必要であるならば、外の Home Page にアクセスして、情報を取得することも効果的であろう。

### 3. システム構築の具体的説明

#### 3.1 CGIとは

ネットワークを通した Home Page 上で書き込みや検索出来るシステム構築を一般に CGI (Common Gateway Interface) と呼ばれる。基本原理は以下のようなものである。具体的に万葉集検索のファイル名で示す。

ブラウザ      www server  
                  ~/foo/man\_ser.html    (~/fooとは任意のディレクトリを示す)

検索文字列など諸条件をブラウザ上で入力

man\_ser.htmlに記述された変数にセットされる  
man\_ser.htmlに記述されたCGIプログラムが呼び出される  
変数がCGIプログラムに渡される

CGIプログラム    ~/cgi-bin/search.shで処理  
結果がブラウザに返される

以上のような手順になる。CGI プログラムは、主にシェル (Bシェル) または、perl あるいは、C言語で記述される。CGIシステムを構築するserverは、一般的にUNIX server が中心であるが、security や動作能力に注意を払えば、Macintosh または windows NT server 或いは Linux 等 PC unix server 上でも構築可能である。

CGI プログラムを置くディレクトリは、システム管理者によって作られる。それは、html ファイルを置くディレクトリがシステム管理者によって作られているのと同様である。ディレクトリの作り方は、それぞれの OS の取り扱い方になるので、解説は省略する。

テキスト検索の場合、注意すべきは文字コードの統一である。特にパソコンで作ったテキストをUNIXワークステーションに置く場合は、たいがいは SJIS コードになったままなので、ブラウザからの入力がEUCに自動変換されている場合は、正常に検索されない。UNIXワークステーションに置いたテキストは EUC コードに変換しておく必要がある。

だいたい CGI についての概説であるが、以下に具体的な CGI プログラムを示していく。

#### 3.2 授業参加者のみを対象とする掲示板へのアクセス許可

通常 www server に掲示板を置くと、URL を直接指定することによって掲示板に誰でもアクセス出来るようになる。その場合、公開授業のようになってしまっていて、授業としての特性が損なわれる恐れも起こる。もちろんビジターとして他者の参加を許すことも

network を使うことの利点ではあるが、とりあえず授業参加者のみを対象としたい場合は、最初に授業参加者であることを確認する必要がある。そこで以下に授業参加者確認のためのシステム構築を述べる。

授業参加者は全員が授業用の server への利用登録をしていることが原則となっているので、server の user id を入力することによって掲示板へ進めるようにすることが一番簡単な方法であろう。具体的には次のようなイメージになる。

confirm.html

---

## USER 確認

あなたの user id を入力して、確認ボタンを押して下さい。  
確認されると掲示板コーナーに移ります。

USER ID : hanako                    # 参加者の user id を入力する。

確認    取消                        # 確認ボタンを押す

# それぞれの場合によって異なる。

# ID が照合した場合

山口花子さん、ようこそ。ここから掲示板に入して下さい。

# ID が異なる場合

受講者でないか、入力を間違っています。

# ID の長さが異なる場合

ID は正しく入力して下さい。

---

最初の ID 確認もブラウザによる CGI 機能を用いる。以下のファイル構成になる。

confirm.html	確認用 HTML ファイル (フレーム分割)
confirm_1.html	確認用 HTML ファイル
confirm_2.html	確認用 HTML ファイル
confirm.sh	確認用 shell ファイル
confirm.c (confirm.exe)	確認用プログラム
user_id.dat	授業参加者登録ファイル

source は以下のようなものである。

confirm.html

---

```
<! confirm.html Ver 1.00 1997.10>
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>国文学演習掲示板</TITLE>
</HEAD>
<FRAMESET ROWS=20%,80%>
<FRAME SRC="confirm_1.html">
<FRAME SRC="confirm_2.html">
</FRAMESET>
</HTML>
```

-----

原則としてブラウザにおけるファイル名表示を隠す目的でフレーム分割を行っている。ただブラウザの viewer などアクセスファイル名を知ることが出来るが、一応表に出ない形にする。

confirm\_1.html

```
-----
<! confirm_1.html Ver 1.00 1997.10 >
<HTML>
<HEAD>
<TITLE>国文学演習掲示板</TITLE>
</HEAD>
<BODY>
<BODY BGCOLOR="#ffffff">
<P>
国文学演習の掲示板です。<BR>
下にあなたのUSER NAME を入れて下さい。<BR>
受講者であると認められると、掲示板を使用できます。<BR>
</P>
</BODY>
</HTML>
```

confirm\_2.html

```
-----
<! confirm_2.html Ver 1.00 1997.10 >
<HTML>
<HEAD>
<TITLE>国文学演習掲示板</TITLE>
</HEAD>
<BODY>
<BODY BGCOLOR="#ffffdd">
<P>
<FONT COLOR="#ff00ff">
<H2>USER 確認</H2></FONT>
あなたのuser id を入力して、確認ボタンを押して下さい。<BR><BR>
```

```

確認されると掲示板コーナーに移ります。<BR><BR>
<FORM METHOD="POST" ACTION="/cgi-bin/confirm.sh" >
<FONT COLOR=" #0000ff" >
<B>USER ID NAME<B></FONT>
  <INPUT TYPE=text NAME="userid" MAXLENGTH=16 SIZE=10><BR>
<BR>
<INPUT TYPE="submit" VALUE="確認" >
<INPUT TYPE="reset" VALUE="修正" >
</FORM>
</P>
</BODY>
</HTML>

```

---

```
confirm.sh
```

---

```
#!/bin/sh
```

```
echo "Content-type: text/html"
```

```
echo ""
```

```
if [" $REQUEST_METHOD" = "POST" ]; then
```

```
    eval `~/foo/cgi-bin/cgiparse -init`
```

```
fi
```

```
    eval `~/foo/cgi-bin/cgiparse -form | /usr/local/bin/nkf -e`
```

```
CONFIR="/~foo/user_id.dat"      # 利用登録者名簿の定義
```

```
KAKUNIN="/~foo/confirm"        # 確認プログラムの定義
```

```
USERID=$ FORM_userid           # ID変数を定義
```

```
$ KAKUNIN $ USERID $ CONFIR    # 確認プログラムを実行
```

---

```
confirm.c (ANSI規格に準じる)
```

---

```
/*
*****

```

```
confirm.c
```

```
M.Yoshimura 1997.10
```

```
*****

```

```
#include <stdio.h>
```



```

#include <string.h>

#define MAX 256

main( int argc, char *argv[] )
{
    FILE *fp;
    char buff[ MAX ];
    char *space;
    int count = 0;

    if( NULL == ( fp = fopen( argv[2], "r" )) ){
        fprintf( stderr, "Cannot open file¥n" );
        exit( 1 );
    }

    /* ID は半角5文字以上。長さが合わないとエラーにする */
    if( strlen( argv[1] ) < 5 ){
        printf( "IDは正確に入力して下さい¥n" );
        exit( 1 );
    }

    while( fgets( buff, MAX, fp ) != NULL ){

        /* user_id.datからuseridを検索する。ヒットすると、datファイル中の
        文字列 @以下を消して登録者名を表示し、掲示板URLにリンクする
        */

        if( NULL != strstr( buff, argv[1] )){
            if(( space = strchr( buff,'@')) != NULL );
                *space++ = '¥0';
                printf( "%sさん、ようこそ。¥n",buff );
                printf( "<A HREF=¥"http://www.server/~foo/man_si
tsugi.html¥" >ここ</A>" );
                printf( "から掲示板に入して下さい¥n" );
                count++;
            }
        }

        /* useridがuser_id.datにないと、エラー */

```

```
if( count == 0 ) printf(" 受講者でないか、入力を間違っています。¥n");
fclose( fp );
}
```

-----  
user\_id.dat  
-----

吉田太郎@tarou  
山口花子@hanako  
国語二郎@jirou

·  
·  
·

(名前及びIDは、仮称である)

### 3.3 掲示板の作成

学習者間の情報交換の場となる掲示板の仕様は、複数の課題に対してそれぞれに対応したものを必要分準備するところからはじまる。

ファイル名でその構成を示す。

課題一覧page        man\_sitsugi.html  
書き込みボード     man?\_iken.html    (それぞれの課題に応じて複数作成)  
メッセージボード   man?\_mes.html    (それぞれの課題に応じて複数作成)  
書き込み用シェル    man?\_zemi.sh      (それぞれの課題に応じて複数作成)

まず親となる掲示板は、次のようなhtmlファイルである。

man\_sitsugi.html

-----  
<! man\_sitsugi.html >  
<HTML><HEAD><TITLE>万葉集演習質疑応答</TITLE></HEAD>  
<BODY>  
<H1>万葉集演習課題</H1>  
<P><BR><BR>  
1.巻1・18番歌 <A HREF="man1\_keiji.html">掲示板を見る</A> <A HREF="man1\_iken.html">意見を書き込む</A><BR>  
2.巻1・21番歌 <A HREF="man2\_keiji.html">掲示板を見る</A> <A HREF="man2\_iken.html">意見を書き込む</A><BR>  
3.巻1・25番歌 <A HREF="man3\_keiji.html">掲示板を見る</A> <A HREF="man3\_iken.html">意見を書き込む</A><BR>  
4.巻1・29番歌 <A HREF="man4\_keiji.html">掲示板を見る</A> <A HREF="man4\_iken.html">意見を書き込む</A><BR>

-----  
そして書き込みファイルが必要分作成されなければならない。書き込みファイルの document sourceは、次のような仕様である。

man1\_iken.html

-----  
<! man1\_iken.html>  
<HTML><HEAD><TITLE>意見書き込み</TITLE></HEAD><BODY>  
<H2>意見書き込みボード</H2>  
<P>  
「1.巻1・18番歌」についての書き込みボードです。書き込み先を間違わないように注意してください。一旦書き込まれると消去出来ません。  
下の書き込みボタンで、書き込まれます。<BR>  
意見を見る場合は、<A HREF="man1\_mes.html">掲示板</A>をご覧ください。  
<BR><BR>  
<FORM METHOD="POST" ACTION="/cgi-bin/man1\_zemi.sh">  
<B>氏名</B>  
<INPUT TYPE=text NAME="name" MAXLENGTH=30 SIZE=20><BR>  
<B>E-mail address</B>  
<INPUT TYPE=text NAME="e\_mail" MAXLENGTH=40 SIZE=40><BR>  
<B>Message</B><BR>  
<TEXTAREA NAME="message" ROWS=20 COLS=80></TEXTAREA><BR><BR>  
<INPUT TYPE="submit" VALUE="書き込み">  
<INPUT TYPE="reset" VALUE="修正">  
</FORM>  
<A HREF="man\_keiji.html">課題一覧に戻る</A>  
</P></BODY></HTML>

-----  
具体的に書き込み作業を行う shell fileは以下のようなものである。

man1\_zemi.sh

-----  
#!/bin/sh  
  
echo "Content-type: text/html"  
echo ""  
  
if [" \$REQUEST\_METHOD" = "POST" ]; then  
    eval `~/foo/cgi-bin/cgiparse -init`  
fi

```

eval `~/foo/cgi-bin/cgiparse -form | /usr/local/bin/nkf -e`
# ~/foo は、任意のディレクトリ。以下同じ
# ファイル名を定義
IKEN="/~/foo/iken1_log"           # 書き込まれた内容を保存する
TUIKA="/~/foo/tuika"             # 戻り先を指定するリンク
IKEN_1="/~/foo/man1_mes.html"    # 原掲示板
IKEN_2="/~/foo/man1_mes.html"    # 戻り先が追加された掲示板
TALE="/~/foo/tale"               # 戻り先を指定するリンク
COPY='cp'                        # cpコマンド定義
DATE='date'                       # 日付変数定義

# 入力されたデータをデコードする
NAME=$FORM_name
ID=$FORM_e_mail
MESSAGE=$FORM_message

cat <<EOF >$IKEN                 # 入力メッセージ確認
$DATE $NAME $ID<BR>              # 日付、氏名、mail address
$MESSAGE<BR>                     # メッセージ本体
<HR>
EOF
cat $TUIKA                       # 戻り先指定ファイルを後に追加
cat $IKEN
cat $IKEN >>$IKEN_1              # 入力メッセージを掲示板に追加書き出し
cp $IKEN_1 $IKEN_2
cat $TALE >>$IKEN_2             # 戻り先指定ファイルを後に追加する

```

-----

tuikaとtaleのファイル内容は、以下のようなものである。

tuika

```

-----
<HTML><HEAD><TITLE>掲示板</TITLE></HEAD><BODY>
<H2>掲示板</H2>
<PRE>
<A HREF="/~/foo/man_sitsugi.html">一覧に戻る</A>
<A HREF="/~/foo/man1_iken.html">書き込みボードに戻る</A>
<A HREF="/~/foo/man1_mes.html">掲示板を見る</A>
以下のメッセージが書き込まれました<BR>
<HR>

```

-----

tale

-----

```
<A HREF="/~foo/man_sitsugi.html">一覧に戻る</A>
<A HREF="/~foo/man1_iken.html">書き込みボードに戻る</A>
</PRE><BODY></HTML>
```

---

### 3.4 万葉集の検索CGIの作成

万葉集の検索の場合は、grepなどの検索プログラムの利用がもっとも単純な方法であるが、実際には一首の歌に対して、原文、漢字交じり表記、仮名書き表記の三種類の表記が必要であり、構造化されたデータベース的な処理が求められる。構造化されたファイルを検索するには、CGI構築においてはperlが最も効果的であるが、本稿ではすでに実用化しているC言語による検索方法を中心に提示する。

通常はデータベース構造を持ったファイルの方が便利であるが、歌語による検索の場合は、プレーンテキストの方が便利な場合がある。そこでそれぞれのファイル間でリレーションしながら検索する方法を述べる。

対象とするファイルは、原文が入っているgenbun.txt、漢字仮名交じり訓読の入っているkundoku.txt、仮名訓読だけのkana.txt、それから歌の属性をまとめている事項ファイル(zikou.txt)の四種類のファイルであり、これらを検索対象ファイルや検索結果を出力するファイルに交互に切り替えるシステムを作る。

まず、ブラウザに表示するhtmlファイルは以下のようなものである。ファイル名をman\_ser.htmlとする。

man\_ser.html

---

```
<! man_ser.html 1996.06 Ver 1.00 >
<HTML>
<HEAD>
<TITLE>万葉集検索</TITLE></HEAD>
<BODY>
<BODY BACKGROUND="back073.gif" >
<FORM ACTION="/cgi-bin/search.sh" METHOD="GET" >
<P>
<FONT COLOR="#0000ff" >
<H3>万葉集検索</H3></FONT>
<BR>
manyo.txt Ver 2.01 R1.9 のtext fileを基にしています。<BR><BR>
<B>検索文字列</B>
<INPUT TYPE="text" NAME="name" SIZE=20 MAXLENGTH=30><BR>
<B>検索対象ファイル</B>
原文<INPUT TYPE="radio" NAME="man_file" VALUE="/~foo/genbun.txt" >
訓読<INPUT TYPE="radio" NAME="man_file" VALUE="/~foo/kundoku.txt" c
hecked>
```

```

仮名<INPUT TYPE=radio NAME="man_file" VALUE="/~foo/kana.txt" >
事項<INPUT TYPE=radio NAME="man_file" VALUE="/~foo/zikou.txt" ><
BR>
<B>出力ファイル</B>
原文<INPUT TYPE=radio NAME="out_file" VALUE="/~foo/genbun.txt" >
訓読<INPUT TYPE=radio NAME="out_file" VALUE="/~foo/kundoku.txt" c
hecked >
仮名<INPUT TYPE=radio NAME="out_file" VALUE="/~foo/kana.txt" ><B
R>
<BR>
<INPUT TYPE=submit NAME="run" VALUE="検索実行" >
<INPUT TYPE=reset VALUE="書き直し" >
</P></FORM>
</BODY>
</HTML>

```

-----

漢字仮名交じり訓読ファイルを入出力のdefaultとしている。事項ファイルは出力用には用いない。ここから呼び出すシェルファイルは、以下のようなものである。

search.sh

-----

```
#!/bin/sh
```

```
echo "Content-type: text/html"
```

```
echo ""
```

```
if [ "$REQUEST_METHOD" = "POST" ]; then
```

```
    eval `~/foo/cgi-bin/cgiparse -init`
```

```
fi
```

```
    eval `~/foo/cgi-bin/cgiparse -form | ~/foo/bin/nkf -e`
```

```
HEAD="/~foo/head"
```

```
NAME=$FORM_name
```

```
IN_FILE=$FORM_man_file
```

```
OUT_FILE=$FORM_out_file
```

```
cat << EOF
```

```
検索文字列 = $NAME<BR>
```

```
検索対象ファイル = $IN_FILE<BR>
```

```
出力ファイル = $OUT_FILE<BR>
```

しばらくお待ち下さい。数分以上かかる場合があります。<BR>  
検索文字列がない場合は、\$NAME not found というメッセージが出ます。<BR>  
検索中<BR>  
<HR>  
EOF  
\$HEAD \$NAME \$IN\_FILE \$OUT\_FILE

-----  
検索に用いるプログラムは以下のようなものである。プログラムの書式はANCI規格に準  
じる。

head.c

-----  
/\*\*\*\*\*  
                  head.c                  For search character in manyo text.  
  
  M.Yoshimura          1997.06.12  
\*\*\*\*\*/  
#include <stdio.h>  
#include <string.h>  
  
#define MAX 256  
  
void kensaku\_1( char arg\_1[], char arg\_2[] );  
void kensaku\_2( char arg\_1[], char arg\_2[], char arg\_3[] );  
void kensaku\_3( char arg\_1[], char arg\_2[], char arg\_3[] );  
  
void main( int argc, char \*argv[] )  
{  
  if( argv[2] == argv[3] )  kensaku\_1( argv[1], argv[2] );  
  if(( strcmp( argv[2], "~/foo/zikou.txt" ) == NULL )  
      /\* ~/fooは、任意のディレクトリを示す \*/  
      kensaku\_2( argv[1], argv[2], argv[3] );  
  else  kensaku\_3( argv[1], argv[2], argv[3]);  
  
}

void kensaku\_1( char arg\_1[], char arg\_2[] )  
{  
  FILE  \*fp;  
  char  buff[MAX];  
  int   x = 0;

```

if( NULL == ( fp = fopen( arg_2, "r" ) ) ) {
    fprintf( stderr, "Cannot open file¥n" );
    exit( 1 );
}
while( fgets( buff, MAX, fp ) != NULL ) {
    if( NULL != strstr( buff, arg_1 ) ) {
        printf( "%s<BR>", buff );
        x++;
    }
}
printf( "%d¥n", x );
if( x == 0 ) { printf( "%s not found<BR>", arg_1 ); }
fclose( fp );
}

```

```

void kensaku_2( char arg_1[], char arg_2[], char arg_3[] )
{
FILE   *fp_1, *fp_2;
char   buff_1[ MAX ], buff_2[ MAX ];
char   zikou[ 8 ];
int    x = 0;

```

```

if( NULL == ( fp_1 = fopen( arg_2, "r" ) ) ) {
    fprintf( stderr, "Cannot open file¥n" );
    exit(1);
}

```

```

if( NULL == ( fp_2 = fopen( arg_3, "r" ) ) ) {
    fprintf( stderr, "Cannot open file¥n" );
    exit(1);
}

```

```

while( fgets( buff_1, MAX, fp_1 ) != NULL ) {
    rewind( fp_2 );
    if( NULL != strstr( buff_1, arg_1 ) ) {
        strncpy( zikou, buff_1, 7 );
        zikou[7] = '¥0';
        while( fgets( buff_2, MAX, fp_2 ) != NULL ) {
            if( NULL != strstr( buff_2, zikou ) ) {
                printf( "%s<BR>", buff_2 );
                x++;
            }
        }
    }
}

```



```

        }
    }
    printf("<BR>");
}

if( x == 0 ) printf( "%s not found<BR>",arg_1 );
fclose(fp_1);
fclose(fp_2);
}

void kensaku_3( char arg_1[], char arg_2[], char arg_3[] )
{
FILE *fp_1, *fp_2;
char buff_1[MAX], buff_2[MAX];
char head[11];
int x = 0;

if( NULL == ( fp_1 = fopen( arg_2, "r" ) ) ){
    fprintf( stderr, "Cannot open file¥n" );
    exit( 1 );
}

if( NULL == ( fp_2 = fopen( arg_3, "r" ) ) ){
    fprintf( stderr, "Cannot open file¥n" );
    exit( 1 );
}

while( fgets( buff_1, MAX, fp_1 ) != NULL ){
    if( NULL != strstr( buff_1, arg_1 ) ){
        strncpy( head, buff_1, 10 );
        head[10] = '¥0';
        while( fgets( buff_2, MAX, fp_2 ) != NULL ){
            if( NULL != strstr( buff_2, head ) ){
                printf( "%s<BR>", buff_2 );
                break;
            }
        }
    }
}

if(x == 0 ) printf( "%s not found<BR>",arg_1 );
}

```

```
fclose( fp_1 );
fclose( fp_2 );
}
```

-----

検索文字列は1つとしているが、先ほどの grep を使ったものと同様、and 検索や or 検索が可能のように拡張してもよい。またここではシェルファイルと組み合わせているが、デコード部分をはじめから C で書くことも可能である。そうなると直接 C program の実行ファイルを呼び出す形になる。またここでは必要最低限の機能しか作っていないが、例えば画面の色を変えたり、文字の大きさを変えるなどのコーティングを行うとより見栄えがよくなるであろう。

同様の形で、他に論文目録、各注釈書などを準備する。

#### 4. 今後の問題とまとめ

上記の万葉集テキストの CGI 検索システムおよび一般公開用の電子掲示板は、本学部国語研究室の Home page 上で運営、実行している。URL は、<http://www.edu.yamaguchi-u.ac.jp/~kokugo> である。また上記の Home Page にて万葉集テキストそのものも配付している。ただし授業においては、授業用電子掲示板の利用はあるが、論文目録等の準備不足のために演習目的ではまだ実践するにはいたっていない。システム上の構築は容易であるが、データの入力に時間がかかるからである。

一方で学習者の環境も問題になる。コンピュータの操作については、特定の学習をする必要はないが、基本的な network への理解や文字入力の学習が前提となる。自宅からの接続の場合は、PPP 接続可能の設備がシステム側で必要になるし、或いは学習者があらかじめ自宅で internet 利用可能の環境にしなければならない。また、Home Page 上で実現するには、server の性能も問題になる。授業などで多数の学習者が一度にアクセスすると著しく速度が低下し、最悪の場合にはシステムがダウンすることがある。効率の問題も同時に考えていかなければならないであろう。

授業などは原則として非公開のものであるので、参加者確認の設定ファイルを設けたが、電子掲示板等のファイルアクセスを外部に広げるといふ方向で考えれば、network 利用授業の新しい視野にたつものとして評価出来ると考えている。

本稿では主にシステム構築を中心に述べたが、今後機会があれば、授業実践に基づいた評価を中心にその有効性について論じてみたいと思う。

(1997.12.01 記)