基礎演習 III テキスト
part 4: 自分のホームページを作ってみよう

三木　邦弘

平成 8 年 1 月 26 日

目 次

1　はじめに

ここでは、ワークステーション上に自分のホームページを作るためのやり方を説明します。現在学内のいくつかのワークステーションでは各利用者が自分のホームページを作成する事が可能になっています。そしてそれは学内だけでなく、学外(もちろん海外も含む)からも見ることができるようになっています。どのような内容の物を作るのも自由ですし、通常は関係者以外はURLを知らないので大丈夫ですが、他の人にも宣伝するのならば、それなりに恥ずかしくないホームページを作ってからにしてください。

簡単なページを作成するだけならば1時間もかかりません。高度な事を行おうとすると、それなりに学ばなければならないことが多くなりますが、むしろ内容を工夫することに勤める方が良いと感じています。

2　準備作業

自分で作って自分で見るだけならばどこのディレクトリーに作成しても良いのですが、後に出てくるcgi機能を使ったり、他の人からも見えるようにするためには、自分のホームディレクトリー(loginした時に居るディレクトリー)の下のwwwと言うディレクトリーの中に、関連するファイルを入れる必要があります。このディレクトリーは通常存在しないので作らなければなりません。また、このディレクトリーが他の人から見えるようにしなければなりません。以上の作業をするためには、次のようなコマンドを実行してください。

```
[miwako]% mkdir www△
[miwako]% chmod a+rx . www△
```

この作業は一回だけやれば十分です。[miwako]%はmiwakoさんの場合で、通常は自分のユーザ名が[ ]の中に表示されます。なお、menuプログラムを使用している方は、「その他」の中の「Unixのコマンド」を選択するとこのようなプロンプト(入力促進記号、この場合[miwako]%)が表示されてコマンドが入力できるようになります。最後にmenuへ戻る時には、exit△と入力します。

実際に色々な作業はこれでできた、wwwと言うディレクトリーで行いますので、loginした後では、必ず次のコマンドでそのディレクトリーに移っておいた方が良いでしょう。

```
[miwako]% cd www△
```

ファイルを作成したり、編集したりするとできたファイルは標準では他の人には見えないような設定になっています。作業が終了してさあ見てみようと言う前には必ず、

```
[miwako]% chmod a+r *△
```

を実行しておいて下さい。これを忘れると、自分は見えるが他の人からは見えない物になってしまいます。

3　ファイル名とURL

wwwの下にaaa.htmlと言うファイルを作成した場合のURLは、

http://cc01.center.sugiyama-u.ac.jp/~miwako/aaa.html

になります。miwakoの部分は各自のユーザー名になります。またcc01でないワークステーションを利用している方の場合は、メールアドレスの前にワークステーション名を付けたものになります。このURLならば海外からでもアクセスできますが、知らなければちょっと思い付かないでしょう。

index.html と言う名前のファイルを作ると、このファイルのURLは少し短くなって次のようになります。

```
http://cc01.center.sugiyama-u.ac.jp/~miwako
```

短いほうが伝えるのにも楽ですので、公開する場合にはindex.html と言うファイルを使うと良いでしょう。

aaa.htmlをテストをする際に毎度このURLを入力するのは大変です。自分のファイルならば単に、

```
aaa.html
```

でも可能ですし、同じワークステーションのmasamiの物ならば、

```
http://localhost/~masami/aaa.html
```

でアクセスできます。

4 HTML入門

WWWのサーバーはクライアント(利用者)からの要求に従ってデータを送ります。クライアントはもらったデータを表示するのですが、そのデータはHTML (HyperText Markup Language)と言う言語で記述されています。WWWは予め用意してあったデータを送るだけなので、自分のデータを公開したい場合には、自分のデータをHTMLで記述する必要があります。ここではその概要を説明します。

4.1 簡単な例

例えば次のような内容を、aaa.htmlと言うファイルに入力してみましょう。

```
<TITLE>簡単な例</TITLE>
<H1>これはレベル1の見出し</H1>
HTMLの世界へようこそ。
これは1番目の段落です。<P>
そしてこれは2番目の段落です。<P>
```

念のためにファイルの入力・編集方法を復習しておくと、

```
[miwako]%ng aaa.html△
```

で編集プログラムngを起動して、データの入力や編集を行った後で、C-x C-s で保存して、C-x C-c で終了します。そして次のようにしてそれを見ることができます。

```
[miwako]%lynx aaa.html△
```

lynxで見る場合には入力したものとあまり変わらないのではないかと言う印象を受けるでしょうが、次のようにしてMosaic等で見ると、見出しの所の字の大きさが大きくなっています。

```
[miwako]%Mosaic aaa.html△
```

HTMLは文章中に様々なマークアップタグ (markup tags) を挿入して様々な指示を行います。この例では、< >で囲まれた部分がそうです。< の次にはタグ名が続きます。これは大文字と小文字の区別は無いので、<TITLE>の代わりに<title>と書いても構いません。タグ名が／で始まっているのは有効範囲の終わりを示します。</XXXX>は<XXXX>の終わりを示しています。通常のタグは全て終わりのタグと対になって使われますが、例外も幾つかあります。段落の終わりを示す<P>等がその例です。

4.2 基本タグ

ここでは、先程の例にも出てきた基本的なタグについて説明します。

- 表題：文章の表題を示すものです。通常本文とは別の場所に表示されます。また索引などの見出しに使われることもありますので、文章の内容を的確に示すものが望まれます。タグの形式は次のようなものです。

  <TITLE>表題の文</TITLE>

- 見出し：HTMLは、1から6までの6つのレベルの見出しが可能で、レベル1が一番大きな見出しになります。見出しとして指定された文は独立した左詰めの行として表示されます。タグの形式は次のようなものです。ただし、yの所は実際は1〜6の数字になります。

  <Hy>見出しの文</Hy>

- 段落：何もタグの付いていない文章は、クライアント側の都合 (通常画面の幅) に合わせて詰め込まれます。段落として独立させたい場合には、段落の終わりに次のようなタグを付ける必要があります。

  文文文...文<P>

- 番号なしリスト：この説明文のような●が先頭に付いた箇条書きをするためには次のようなタグを使います。

  ```
  <UL>
  <LI>文章 1
  <LI>文章 2
  </UL>
  ```

  <LI>が●になる感じです。<LI>の部分は幾つでも構いません。また3重までの入れ子にすることも可能です。

- 番号付きリスト：先頭に1、2、3と数字が順番に付いた箇条書きをするためには次のようなタグを使います。

  ```
  <OL>
  <LI>文章 1
  <LI>文章 2
  </OL>
  ```

  今度は<LI>が数字になる感じです。<LI>の部分は幾つでも構いません。また3重までの入れ子にすることも可能です。

- 定義型リスト：言葉ではちょっと説明しがたいものですが、次のような形にしたいときにこれを用います。

電子計算機
　　コンピュータのこと。
コンピュータ
　　かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。

これは、次のような3種類のタグを使って記述します。

```
<DL>
<DT>電子計算機
<DD>コンピュータのこと。
<DT>コンピュータ
<DD>かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。
</DL>
```

- 引用文：引用などで通常の文章よりも右に凹んだ文章を記述したいときには、次のタグを使います。

```
<BLOCKQUOTE>
文文文...文
</BLOCKQUOTE>
```

- 整形済み文章：既に整形が終わっていると言う事で、次のタグで囲まれた文章は入力したままの形で表示されます。

```
<PRE>
                +- 食品栄養学科
生活科学部----+- 生活環境学科
                +- 生活社会科学科
</PRE>
```

この場合画面に表示されるのは、<PRE>のタグが無いだけで後は全く同じものです。

- 強制改行：段落を示すタグ<P>を使用すると段落の間に空行が入ります。それを避けたい場合には、次のようなタグを使います。

```
文文文...文<BR>
```

- 水平線：画面一杯の水平線を引くタグは次のようなものです。

```
<HR>
```

- コメント：文章の説明的なもので、表示されては困るものは次のようなタグを付けておきます。

```
<!--  文文...文  -->
```

4.3 文字の形式について

通常の文字はそのままですが、< > & "の4文字は特殊な意味を持つためにそのまま使えません。それぞれ次のような形で記述します。

< → &lt
> → &gt
& → &amp
" → &quot

次のようなタグを付けることにより論理的な意味付けを文字に与えることが可能です。異なる意味付けのものはクライアントで色や書体の違いとして表示されます。

<DFN>定義された語</DFN>	通常イタリックで表示される。
<EM>強調された語</EM>	通常イタリックで表示される。
<CITE>本等の表題</CITE>	通常イタリックで表示される。
<CODE>プログラムなど</CODE>	通常等幅文字で表示される。
<KBD>キーボードのキー</KBD>	通常等幅の太字で表示される。
<SAMP>コンピュータの状態</SAMP>	通常等幅文字で表示される。
<STRONG>強調された語</STRONG>	通常太字で表示される。
<VAR>変数など</VAR>	通常イタリックで表示される。

また直接次のように字体を指定することも可能です。

<B>太字</B>
<I>イタリック</I>
<TT>等幅文字</TT>

4.4 リンクの付け方

他の文章へのリンクや同じ文章内にリンクを張ることができます。クライアントはそこを指定するだけでそのリンクを張った先を見ることができます。このリンクが可能である点がハイパーテキストの由来だと言われています。

- 他文章へのリンク：これは次のような形式のタグを用います。

  <A HREF="URL">クリックされる文</A>

  URLの部分には実際にリンクする先の文章のURLが入ります。「クリックされる文」の所はlynxならば選択されたときに反転表示されたり、Mosaic等では色が異なる表示がなされます。リンク先が判るような文が入ります。実際は例えば次の様な形になります。

  <A HREF="http://www.sugiyama-u.ac.jp">椙山女学園大学のホームページ</A>
  <A HREF="betu.html">同じディレクトリーにあるbetu.htmlと言うファイル</A>

- 文書内へのリンク：予め次の様なタグ(アンカー)を入れておくと、そこへ行くリンクを張ることが可能です。

  文...文<A NAME="namae">文</A>文...文

  namaeは適当な語を使います。同じファイル中で同じ語は使えません。そしてリンクを張るときには次の様にします。

```
<A HREF="#namae">クリックされる文</A>
```

要するに先程指定した語の前に#を付けます。長めの文章で先頭の所に目次や索引を設けて、そこから後に続く文章の該当するところへリンクを張ると言う形でよく使われます。

この両者を同時に使う事も可能です。つまり他文書の中で予めアンカーを指定しておけば、リンクを張る側は、URLの後に#とアンカーで指定した語を書けば良いようになっています。

4.5 演習問題

以下のような超簡易CAIを作れ。

1. cai1.htmlと言うファイルを作成し、この中には問題文と答えの選択子(4つぐらい)を入れる。正解を選択したらcai2.htmlへ、間違った選択ならばcai3.htmlへ行くようにリンクを張る。
2. cai2.htmlと言うファイルを作成し、正解者に対するメッセージを入れる。また「次の問題」と言う所を作成して、他の人のcai1.htmlへリンクを張る。
3. cai3.htmlと言うファイルを作成し、誤答者に対するメッセージ(ヒントなど)を入れる。また「問題にもどる」と言う所を作成して、cai1.htmlへリンクを張る。
4. 確認の前に、chmod +r *△を忘れずに。他の人の問題に行くときに行き先のファイルが無かったり、この作業がされていないとエラーメッセージが表示される。
5. lynx cai1.html△と入力してちゃんと文章が表示されるか、リンクが正しく張れたかを確認せよ。

各ファイルでは、<TITLE>なども忘れずに入れること。

5 イメージの扱い

通常の端末ではイメージ(画像)を扱えないので、本体で作業を行わなければならない。

5.1 入力と編集

5.2 イメージファイルの指定の仕方

5.3 演習問題

前章で作成した超簡易CAIの各ファイルにそれぞれふさわしいイメージを追加せよ。

6 会話的なページ

これまでの知識で作成するページは全てあらかじめ作られたものを見せるのに過ぎない。利用者側からの入力を受けてこれに答えるようなものの作成法について説明する。

6.1 コンパイルの仕方

ワークステーション上でCによるプログラムを作成するには、次のような手順となる。

1. ソースプログラムの入ったファイルを作成する。拡張子は.cにする。
2. gcc ファイル名△でコンパイルする。何かメッセージが出たら、ソースプログラムの誤りがあるので、修正して再度コンパイルをする。
3. a.out△でコンパイルしたプログラムを実行することができる。利用者側からの入力をキーボードで入力し、画面に出てきたものが実際に利用者へ返されるものである。よって画面に出されたものは、HTMLで書かれたものになっていなければならない。
4. a.outを拡張子.cgiのファイル名に変更する。例えばtest.cgiにするならば、mv a.out test.cgi△と入力する。そして他の人にもこれを実行できるように、chmod a+rx test.cgi△と入力する。

実際の実行例を示すと、

1. ng test.c△と入力して、以下のプログラムを入力してから保存、終了する。

```
#include <stdio.h>

main(){
   puts("Content-type: text/html");
   puts("");
   puts("<H1>This is a test for cgi.</H1>");
}
```

2. gcc test.c△と入力してコンパイルする。
3. a.out△と入力して、以下の画面に以下のように出力されれば良い。

```
Content-type: text/html

<H1>This is a test for cgi.</H1>
```

4. mv a.out test.cgi△を入力し、次にchmod a+rx test.cgi△と入力する。
5. lynx http://localhost/~miwako/test.cgi△と入力して画面に、

```
   This is a test for cgi.
```

と表示されることを確認する。cgiプログラムを作成する際には以下のような点に注意すること。

- プログラムを修正した場合には再度コンパイルをし、名前を変え、実行できるように設定を変更しなければならないが、名前を変えるところで2回目からは、remove ファイル名?と確認を求められるので、y△と答えます。
- プログラムが別のファイルを参照するならば、そのファイルの設定も正しく設定すること。読み出すだけならば、chmod a+r、書き込みも行うならばchmod a+rwとする。後者の設定は全ての人にファイルの内容の変更を許すことになるのでやや危険な設定です。
- 出力の先頭には必ず “Content-type: text/html”の行と次の空行がなければならない。WWWのサーバーはこれを読んで以下に続く内容を扱うからです。

6.2 入力用ページの作り方

実際の処理を行うのはプログラムですが、通常はプログラムは利用者から何かデータをもらってからそれを処理します。そのもらうためのページの作るために用いるHTMLのコマンドについて説明します。

6.2.1 FORM: 形式とプログラムの指定

FORMを使って、入力されたデータを送る先のプログラムの指定とそのときのデータ形式を指定します。

```
<FORM method=POST action="URL">
 .....
</FORM>
```

methodとしてはPOST以外にGETと言うのも利用できますがここでは説明を省略します。URLの部分には実際に作成したプログラムのURLが入ります。

6.2.2 INPUT: ボタン入力

入力が全て終わって、プログラムを起動してくれと言うボタンの定義は次のようにします。

```
<INPUT type="submit" value="Send">
```

Sendの部分はボタンの上に書かれる文字列なので、任意のものが可能です。また似たような形ですが、利用者の入力したものを全て消して最初の状態に戻すボタンは次のように定義します。

```
<INPUT type="reset" value="Erase">
```

Eraseの部分はボタンの上に書かれる文字列なので、任意のものが可能です。
利用者が印を付ける形のボタン(チェックボックス)は次のようにします。

```
<INPUT type="checkbox" name="info">
```

この場合はボタンしか出てこないのでこの前後に文章を補って、何のボタンかを示す必要があります。infoはプログラムに結果を返すときの見出し名になります。info以外の適当なものでも構いません。
幾つかの選択の中で一つだけしか選べないボタン(ラジオボタン)には、次のようにします。

```
<INPUT type="radio" name="service" value="ip">
<INPUT type="radio" name="service" value="uucp">
<INPUT type="radio" name="service" value="ppp">
```

serviceとなっているところが同じもの同士が一つのグループとなり、この中では一つだけしか選択できなくなります。そしてこれはプログラムに返すときの見出し名ですので、service以外の適当なものでも構いません。ip、uucp、pppはそれぞれのボタンが選択されたときにプログラムに返される値です。これも適当な値で構いません。

6.2.3 INPUT: 短い文字列の入力

短い文字列の入力もINPUTで行えます。長い文章などの入力には次のTEXTAREAを用います。

```
<INPUT name="realname">
```

realnameはプログラムに結果を返すときの見出し名になりますので、適当なものでも構いません。
また、次のようにすると利用者が入力した文字が全て*印で置き換えられて表示されるので(もちろんプログラムには入力した文字がそのまま送られる。)パスワードの入力などに使えます。

```
<INPUT type="passwd" name="pass">
```

passの部分は適当なものでも構いません。

6.2.4 TEXTAREA: 長い文字列の入力

長い文章などを入力するにはTEXTAREAを使います。

```
<TEXTAREA name="bunsyo" rows=10 cols=50>見出し？</TEXTAREA>
```

この例では縦10行、横50文字の入力領域が確保されます。

6.2.5 入力されたデータの表現方法

以上の入力欄に入力されたデータは、FORMの所で指定されたプログラムに次のような形で渡されます。

```
名前=値&名前=値&...
```

名前は今まで説明したタグの中でname=で指定した内容です。そして値が実際に入力された内容になります。名前と値の間には=が、次の名前との間には& が入ります。このデータはプログラムへの標準入力として渡されますが、いくら多くのデータであっても1行です。

また値の中に含まれる記号類は全て%の後に16進数のコードとして表現されます。また空白は+になっています。これらの例を以下に示しますが、漢字などはクライアント側で使用している漢字コードの種類によって変わってきますので対応はかなり難しくなります。

今次のようなファイルを作成して実行すると、

```
<FORM method=POST action="http://localhost/~miwako/show.cgi">
<TEXTAREA NAME="name-1" ROWS=3 COLS=60></TEXTAREA><P>
<input NAME="name-2"><p>
<input type="submit" VALUE="Send">
<input type="reset" VALUE="Erase">
</FORM>
```

```
name-1
```

6.3 演習問題

1. 呼び出した回数を表示するcount.cgiを作成せよ。

   (例：http://cc01.center.sugiyama-u.ac.jp/~miki/count.cgi)

2. INPUT等のタグを用いて問題を作成せよ。そしてその結果をメールの形で先生に送るプログラムを作成せよ。

7 解答例

超簡易CAI

- cai1.htmlの内容の例は次の通りである。

```
<TITLE>三木先生の性格に関する問題</TITLE>
<H1>やさしい三木先生の性格は？</H1>

このテキストを作成した三木先生は、これまでにも「こんなやさしい問題は無
い」と言いながら訳の判らない問題を多数出して多くの<B>学生を泣かせて
</B>来ました。一体彼はどんな性格の持ち主なのでしょうか？
```

```
<UL>
<LI><A HREF="cai3.html">暗い暗いネクラである。</A>
<LI><A HREF="cai3.html">まじめな中年男性である。</A>
<LI><A HREF="cai2.html">結構チャランポランである。</A>
<LI><A HREF="cai3.html">やさしい人である。</A>
<UL>
```

- cai2.html の内容の例は次の通りである。

```
<TITLE>正解です</TITLE>

<H1>よくできました</H1>

昔はそうでもなかったのですが、最近は本人も嫌になるほどの
チャランポランです。<P>

<A HREF="http://cc01.center.sugiyama-u.ac.jp/~tuginohito/cai1.html">
次の問題に進む</A>
```

- cai3.html の内容の例は次の通りである。

```
<TITLE>まちがいです</TITLE>

<H1>残念でした外れです</H1>

<UL>
<LI>「暗い暗いネクラである。」そんなに暗くありません。
最近女子大生不信にはなっていますが。
<LI>「まじめな中年男性である。」どう考えたって「まじめ」ではありません。
<LI>「やさしい人である。」表題にそう書いてあるからって、そうとは限りません。
<UL>

<A HREF="cai1.html">問題にもどる</A>
```

回数を表示する

プログラムで使用する変数は、プログラムの実行を終わると全て消えてしまいます。ですから、回数はファイルで記憶しなければならないと言う点に気が付けば、簡単な課題です。

- count.c の内容は次のようになります。count と言うファイルに回数を入れています。

```
#include <stdio.h>

main(){
   FILE  *f;
   int   n;

   f=fopen("count","r");
   if (f==NULL)     /*  countが無かった場合    */
      n=1;
   else {           /*  countがあった場合      */
      fscanf(f,"%d",&n);
      fclose(f);
   }

   printf("Content-type: text/html\n\n"); /* \n\nで空行も出している。*/
```

```
    printf("<H1>あなたで%d人目です。</H1>\n",n);

    f=fopen("count","w");
    fprintf(f,"%d\n",n+1);
    fclose(f);
}
```

- countと言うファイルはプログラムを実行すると自動的に作られますが、そのままでは他の人から読めない状態になっているので、chmod a+rw count△を一度実行しておく必要があります。