生活社会科学科

専門演習Bテキスト

三木　邦弘

平成15年9月24日

目 次

1 会話的なWebページ

基礎演習で説明した方法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ません。JavaScriptを利用すれば利用者側からの入力を受けて、これに答えるようなものも作れますが、余りたいした物はできません。例えば図書検索などのページをJavaScriptで作成することも不可能ではありませんが、そのためにはページを表示する際に全ての図書のデータをブラウザに送らなければなりません。ここでは利用者がブラウザで入力した内容に応じて応答を返すプログラムの作成法について取り上げます。

1.1 プログラムの作成方法

利用者の入力に応じて異なる応答をするためには、それを実行するプログラムを作成する必要があります。このためにWWWのサーバーには設定したプログラムをサーバー上で起動し、利用者からの入力をそのプログラムに渡し、プログラムからの出力をまた利用者に送り返すと言うような機能があります。これをCGI (Common Gateway Interface)と呼んでいます。

このCGI機能を利用するにはプログラムに様々な条件を満たすことが必要です。

- プログラムの拡張子は.cgiにする。通常のプログラムはコンパイルするとa.outと言う名前になりますが、そのままでは使えません。
- 利用者側からの入力は、パラメタの形か標準入力の形でプログラムに渡されます。標準入力とは通常はキーボードからの入力ですので、テストはキーボードで入力することで可能です。
- プログラムの標準出力はそのまま利用者側に送られます。利用者側はそれが単なるテキストなのか、HTMLの記号が付加されたものなのか、画像データなのかわからないので、最初にその目印を送ります。(後の例で出てくるContent-type: text/htmlとその次の空行)通常標準出力は画面ですのでテストの際は画面に正しいものが表示されれば大丈夫です。
- プログラムの実行はサーバーが行います。そのためにファイル等の読み書きを行うプログラムはファイルの許可に注意する必要があります。

実際の簡単な実行例は次のようになります。

1. w3setupを実行する。loginしたあとに必ず1回だけ実行します。これを忘れるとWWWでアクセスできないところに以下のファイルができてしまいます。
2. testcgi.cと言うファイルに以下のプログラムを入力して保存、終了する。

```
#include <stdio.h>

main(){
   puts("Content-type: text/html");
   puts("");
   puts("<HTML>\n<Head>");
   puts("<Title>Test</Title>");
   puts("</Head>\n<Body>");
   puts("<H1>This is a test for cgi.</H1>");
   puts("</Body>\n</HTML>");
}
```

3. testcgi.cをコンパイルする。

4. プログラムを実行して、画面に次のように出力されれば良い。

```
Content-type: text/html

<HTML>
<Head>
<Title>Test</Title>
</Head>
<Body>
<H1>This is a test for cgi.</H1>
</Body>
</HTML>
```

5. a.outと言う名前をtest.cgiと言う名前に変更する。

```
cc12[miwako]% mv a.out test.cgi
```

6. URLとしてhttp://www.center.sugiyama-u.ac.jp/グループ名/登録名/test.cgi を指定すると画面に、

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

と表示されます。

cgiプログラムを作成する際には以下のような点に注意します。

- 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りませんが、名前を変えるところで2回目からは、「mv: ··· を上書きしてもよろしいですか (yes/no)?」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読み書きするならば、そのプログラムの設定を変更する必要があります。今そのプログラム名がtest.cgiならば、a.outをこの名前に変更した後で、

```
cc12[miwako]% chmod u+s test.cgi
```

を必ず行わないとテストでは動いても正しく動作しません。

- プログラムの出力が正しいHTMLのものになっているかどうかの確認は、名前の変更前ならば、

```
cc12[miwako]% a.out | jweblint -
```

で行えます。

- プログラムからページの内容を出力するのではなく、既存のページの内容を表示するには、「Content-Type:」の行の代わりに「Location: URL」と言う行と空行のみを出力します。URLの部分はもちろん表示させたいページのものをhttp:よりフルに指定します。またプログラムの途中でこれを出力すると、プログラムの実行が途中で止められてしまうことがあるので、プログラムの最後で出すようにします。

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. 呼び出すたびに昨年度作成したCAIの問題文のページ、正解のページ、間違いのページを順番に表示するiroiro.cgiを作成せよ。

3. 呼び出す回数をもとに毎回異なる運勢を表示するおみくじプログラムを作成せよ。なお運勢の内容は/homes01a/ss92a/mmiki/omikujiと言う名前のファイルの中にあるメッセージを利用せよ。メッセージの件数は7つで、1行につき一つのメッセージである。

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

1.2 定型的なプログラムの起動法

次に説明するFORMを利用すると様々な入力データをプログラムに渡すことが可能ですが、場合によっては、

- 単に指定のプログラムを起動すれば良い程度である。
- プログラムの起動を普通のリンクと同じような形式にしたい。
- 一つのページで様々なプログラムを起動したい。

のような要求が出ることもあります。そのような場合には、既に述べた<A href="URL"> ... </A>でURLで起動したいプログラムを指定する事により解決できます。(ただしプログラム名は前節同様に.cgiで終わっている必要がある。)

さらにプログラム名の後に次のような形でプログラムのパラメタを渡すことが可能です。

```
<A href="プログラムのURL?パラメタ1+パラメタ2+パラメタ3"> ... </A>
```

この場合プログラムに渡されるパラメタ1からパラメタ3は、プログラムのmain の先頭を次のようにしていると、argv[1]からargv[3]までにそれぞれ入ります。

```
main(int argc, char *argv[])
```

argcにはパラメタの数+1が入りますので、パラメタの数が色々変化するような場合にも対応できます。注意しなければならないのは、argv[]に入ったパラメタは文字列として扱われていることです。もし数値として扱いたい場合にはatoi()と言う関数を使用して数値に直す必要があります。

```
n=atoi(argv[1]);
```

これで1つ目のパラメタとして指定した数値がnに入ります。端末でまたテストをする場合はパラメタの部分は空白で区切ります。例えばhref="test.cgi?aaa+2222+ccc"をテストするときには、

```
cc12[miwako]% a.out aaa 2222 ccc
```

とします。

なおパラメタの中に空白や漢字等が含まれる場合には次に説明するコード化したものを指定します。

演習問題

1. index.htmの中で、「おみくじ」を選択すると前の演習問題で作成したomikuji.cgiが起動されるようにせよ。
2. パラメタとして与えられたファイル名をもとに、そのファイルの内容を表示するlist.cgiを作成せよ。
3. 前問で作成したプログラムの出力を見ると<>の部分が消えている。そのような事がないようにせよ。

   ヒント：ファイルの内容を出力する際に、一文字ずつ出すようにして、<や>の時は代わりに<や>を出すようにする。

1.3 プログラムの出力の取り込み

あちこちのホームページにカウンターが付いており、何人目のアクセスであるかどうかが判るようになっています。回数を数えるだけならばすでにCGIを利用して実現してみましたが、実際のページではその内容の全てをプログラムで出力するのは実用的ではありません。ページのごく一部にプログラムの出力やファイル等を取り込むSSI (Server Side Include) と言う機能があるのでそれを利用するのが普通です。本来SSI自体は様々なものを取り込む事ができるのですが、ここではそのごく一部のみを紹介します。

- SSIを利用したファイルの拡張子は**.shtml**である必要があります。中身は99%は通常の.htmのファイルと同じなのですが、次に述べる取り込みの指示が有効になります。もし拡張子が.htmのものにSSIの指示を入れても無視されますので注意してください。また拡張子が正しくてもサーバー上にないと動きません。
- コマンド (通常のプログラム) の実行結果を取り込みたいときには、

  ```
  <!--#exec cmd="実行したいコマンド" -->
  ```

  を取り込みたいところに挿入します。自作のプログラムの場合は「./プログラム名」にしないと動きません。
- CGIプログラムの実行結果を取り込みたいときには、

  ```
  <!--#exec cgi="実行したいCGIプログラム" -->
  ```

  を取り込みたいところに挿入します。自作のCGIプログラムの場合は「./CGIプログラム名」にしないと動きません。

どうしても拡張子が.htmのファイルでもSSIを使いたいときはそのやり方がありますがお勧めできません。と言うのはSSIが使えるファイルはサーバーがクライアントに送る際に必ずその内容を調べて必要があれば挿入を行わなければなりません。その結果サーバーに負担がかかるわけですが、.htmでもSSIと言うことになると全てのファイルに対して調べる必要が生じるので大変だからです。

コマンド (普通のプログラム) とCGIプログラムとの違いは最初にContent-type の行を送るかどうかが大きな違いです。ちょっと考えるとその分簡単で済む普通のプログラムだけでも良いように思えますが、Content-typeで以下続く内容が画像データであることを示して画像データを出力して表示すると言うことはCGIプログラムでしかできません。

自分のホームページ (index.htm) にSSIを使いたい場合は、これをindex.shtml に名前を変更すれば可能です。(つまりindex.htmはなくす) 他の人にURLを教えるときにファイル名を省略したもの (`http://www.center.sugiyama-u.ac.jp/`グループ名/登録名/) で伝えれば、これまで通り今度はindex.shtmlを見せるようになります。

演習問題

1. date と言うコマンドを使用すると現在の日時が表示される。これを test.shtml の中で呼び出すようにせよ。
2. cal と言うコマンドを使用すると今月のカレンダーが表示される。これを test.shtml の中で呼び出して正しく表示されるようにせよ。
3. test.shtml の中に乱数を使用して、これまで作成した画像の中から一つをランダムに表示するようにせよ。(selgif.c)

1.4 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。そのデータをもらうためのページの作るために用いる HTML のタグについて説明します。

1.4.1 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。JavaScript の際には name の指定が必要でしたが、CGI での使用の際は method と action の指定が必要です。JavaScript も併用する場合は name の指定もします。

```
<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、GET は POST と比べてやや扱いが簡単なものの、入力できるデータ量などに制限があるので、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。

</Form> までの部分に以下で説明するボタンや入力欄の定義が入ります。

1.4.2 Input: 起動ボタン、消去ボタン

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。

1.4.3 Input: チェックボックス、ラジオボタン

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">
```

この場合はチェックする所しか出てこないのでこの前後に文章を補って、何のチェック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名前になりますので、同じ Form の中で重複しなければ 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" checked>
<Input type="radio" name="service" value="uucp">
<Input type="radio" name="service" value="ppp">
```

name=で指定しているもの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名前になりますので、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。これも識別さえできれば適当な値で構いません。なお checked の付いている所が最初に選択された状態になります。

1.4.4 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名前になりますので、適当なものでも構いません。画面上では通常 20 文字程度の幅の入力欄になりますが、入力する内容は、それ以上になっても構いません。しかし入力の際に一部しか見えない状態になるので、場合によっては次のように size=として入力欄の幅を文字数で指定する方が良いでしょう。

```
<Input name="realname" size=100>
```

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。

1.4.5 Input: 定形データを送る

いくつかのページで同じプログラムを利用する場合、プログラム側からどのページから来たものか判断するための情報が必要となります。また WWW では、サーバーとクライアントは一つのページをやりとりする度に接続を切ってしまうので、プログラムが複数のページを生成するときに何番目までを送ったかを知るためにも使われます。

```
<Input type="hidden" name="address" value="miwako">
```

この場合、画面には入力のためのものは何も表示されません。そして後述のように、address=miwako と言うデータがプログラムに渡されます。

1.4.6 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。

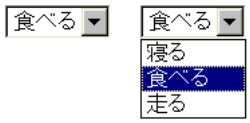
```
<Textarea name="bunsyo" rows=10 cols=50>内容</Textarea>
```

この例では縦 10 行、横 50 文字の入力領域が確保されます。「内容」の部分はこの入力用領域に予め入るものです。不要ならば省略しても構いません。

1.4.7 Select: 選択メニュー

項目を示してその中から一つを選んでもらう方式です。選択をしようとすると項目の一覧が表示されるので、多くの項目でも場所をとりません。

```
<Select name="selone">
   <Option>寝る
   <Option selected>食べる
   <Option value="run">走る
</Select>
```



この例では、「寝る」、「食べる」、「走る」の3つの中から一つ選ぶ事ができます。<Option>の所はいくつでも可能です。selectedは1つだけ指定可能で、この項目には最初から印が付きます。またvalueの指定をすると、プログラムに渡される値はこちらになります。(この例では「走る」の代わりに「run」が渡される。)

なお、複数選択可能な項目とするには、最初の行を次のようにします。

```
<Select name="selmul" multiple>
```

この場合にはselectedを複数指定しても構いません。複数選択したときには名前=値と言うものが複数繰り返されますので注意が必要です。

また、画面上で幾つかの項目が最初から表示されるようにしたい場合には、最初の行を次のようにします。

```
<Select name="selone" size="3">
```

とすれば、最初から項目が3つ表示されるようになります。項目が3 つ以上ある場合には、残りの項目を表示させるためのスクロールバーが自動的に表示されます。

1.4.8 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=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(例えばinput.htm)を作成して実行すると、

```
<HTML><Head><Title>Test form</Title></Head>
<Body>
<Form method=POST action="show.cgi">
これはTextAreaです。
<Textarea name="name-1" rows=3 cols=60>Sample Data</Textarea>
<P>これはInputです。
<Input name="name-2">
<p><Input type="submit" value="Send">
   <Input type="reset" value="Erase">
</Form>
</Body></HTML>
```

次の様な画面になります。

これを実行する前にshow.cgiを作成しておく必要があります。例えば、cgiプログラムに渡されたものをそのまま返すプログラムならば次の様になります。

```
#include <stdio.h>

main(){
   char line[300];                /*  十分な大きさの変数にすること      */

   gets(line);                    /*  FORMからの入力                     */
   puts("Content-type: text/html");
   puts("");
   puts("<HTML><Head><Title>Input data</Title></Head>");
   puts("<Body><Pre>");
   puts(line);                    /*  FORMからの入力をそのまま出力する */
   puts("</Pre></Body></HTML>");
}
```

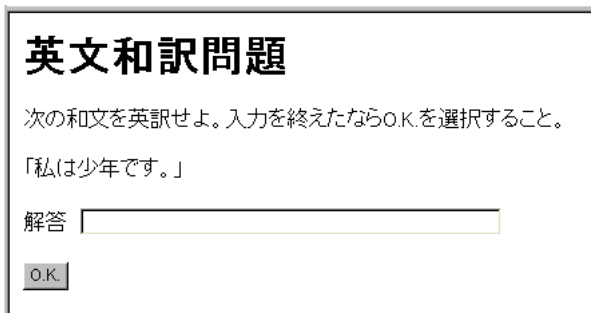
これをコンパイルして前述のinput.htmlを実行して適当なデータを入力してSendボタンを実行すると、

```
   name-1=abc%0def%0ghi&name-2=jkl
```

のような表示がされます。

演習問題

1. 和文英訳の問題のページを作成せよ。入力された英文が正解と一致するかどうかで判定を下す。(eigo.htm、eigo.c、参考：http://cc01.center.sugiyama-u.ac.jp/~miki/eigo.html)



2. 性格診断のページを作成せよ。(seikaku.htm、seikaku.c)
   - 答えのパターン数は50以上のこと。
   - 答えの判定にはstrstr()関数を利用するとよい。
   - 参考：http://cc01.center.sugiyama-u.ac.jp/~miki/seikaku.html

2 CGI中級編

ここではより高度？な会話的なページを作成する方法について説明します。

2.1 ページ内容の自動更新

通常のWebページは一旦表示されると利用者が何かしない限り動くことができません。例えば街の風景などの画像を定期的に自動更新していても、そのままでは利用者の方に表示された画面は変化しません。この問題を解決するには3 つほど方法があります。

- Javaを利用する。
- サーバーから継続的にデータを送る。
- ブラウザから自動的に更新の要求が出るようにする。

ですが、ここでさらにJavaまで勉強する元気はないでしょうし、サーバーから継続的にデータを送るのも少しややこしいので省略して、最後の例だけ示します。ブラウザから5秒毎に更新の要求を出させるには、Headタグと/Headタグの間に次のような指示を入れます。

```
<Meta HTTP-EQUIV="Refresh" Content="5">
```

またCGIの出力ではContent-typeの次に出力する方法もあります。

```
Content-type: text/html
Refresh: 5

<HTML>
......
</HTML>
```

Refresh:の次の行は空行でなければなりません。最後の5が秒数ですがあまり短い間隔で更新をすると、更新に必要な時間を越えてしまうこともあるので注意します。特に学外から見てもらう場合には、そのページの表示に思いがけないほどの時間がかかることもありえますから。

なお、これの応用として、次のようにすると5秒後に別のページを見せることも可能です。

```
<Meta HTTP-EQUIV="Refresh" Content="5;URL=別のページのURL">
```

別のWebサーバーに引っ越したときに、元のページにこの指定をして、元のページを見に来た人を自動的に新しい引っ越し先に飛ばすのに使われます。

演習問題

1. test.shtmlファイルに10秒毎に更新を行う指示を追加して確認せよ。
2. test.shtmlとaaa.htmlに指示を追加し、ほっておくと、それぞれ10秒毎に入れ替わるようにせよ。

2.2 送信前の確認

利用者が何らかの入力を行った後、それを元に処理を行う場合には、大抵利用者の入力間違いのチェックが必要になります。明らかに間違いとわかる入力をもとに処理を行っても良い結果が得られる訳がないからです。チェックの方式としては、

- サーバー側で行う
- ブラウザ側で行う

の2通りが考えられます。サーバーでチェックする場合は、データベースなどを用いて確認ができますので、例えば入力された氏名が本学の学生のものではない、と言うような判断もできます。ブラウザ側でこのようなことを行おうとすると学生全員の氏名をブラウザに送らなければなりません。ただサーバーに入力内容を送り、サーバーで処理して、その結果を受け取らないとエラーメッセージを出すことができません。よって反応はネットワークやサーバーの状態によってはかなり悪くなります。逆にブラウザ側では、利用者の入力忘れや、数値が異常に大きいとか少ない程度の簡単なチェックしかできません。しかしブラウザ側で処理をするので素早くエラーメッセージを表示することができ、ネットワークやサーバーに負担をかけません。実際は両者の手法によるチェックを併用することが一般的です。

ブラウザ側でチェックをする場合は、通常 JavaScript で行います。その時、利用者が入力の終了を知らせるための type が「submit」のボタンの定義を次のようなものに置き換えて、クリックするとチェック用の関数を呼び出すようにします。

```
<Input type="button" value="送信" onClick="check()">
```

もちろん「check()」は別途定義しなければなりません。if などを用いて入力欄の内容を確認し、問題があれば return を実行し、問題がなければ「document. フォーム名.submit()」を実行するとサーバーにフォームの内容が送られます。

2.3 ファイルの排他制御

Web サーバーでは通常 10 ぐらいの利用者からの要求に同時に答えることができます。と言っても厳密に言えば CPU は一つですので、細切れに少しずつ処理をしていくことになります。同時に来た要求が同じものに対することもあります。単純に HTML で書かれたファイルであれば問題は生じませんが、CGI プログラムの場合には同時に実行すると問題が生じることがあります。

ファイルを更新するプログラムの場合、通常まず現在のデータをファイルから読み込み、処理をし、その結果を書き込むと言うような形なります。よってプログラムが複数起動されると、同時に読み込み、ほぼ同時に結果を書き込もうとするので、先行した方の結果はすぐに上書きされて消えてしまいます。さらに細かく見ると、ファイルの書き込みの際には一旦ファイルの内容は消去されます。そして次に新しい内容を入れるのですが、その瞬間に読み込まれてしまうと、読み込んだプログラムには空の内容が渡ってしまいます。カウンターのプログラムでこれが生じるとせっかく蓄積した数千回の値がゼロにもどってしまいます。

このような事が生じないように、プログラムがファイルを触る前にファイルをロック (鍵をかける) し、処理して書き込みが終わるまで他のプログラムを待たせると言うことが行われます。これをファイルの排他制御と言いますが、実現するためには、

- OS の持つファイルのロック機能を利用する

- リンクファイルを使う
- シンボリックリンクを使う

などの方法があります。ここでは比較的容易に使える最後の方法を紹介します。まずプログラムの先頭で、

```
#include <unistd.h>
```

を入れて次に出てくるsymlink()関数が利用できるようになります。そしてファイルへのアクセスの直前に次のような記述を入れます。

```
while (symlink("bbb","aaa")!=0) sleep(1);
```

symlink()関数は、「bbb」というファイルへのシンボリックリンクを作成しようとします。できるシンボリックリンクの名前は「aaa」になります。シンボリックリンクは実在しないファイルに対しても設定することができますので、「bbb」は実在するファイル名でも良いし、そうでないファイル名でも構いません。一方「aaa」の方は存在しないファイル名でないとエラーになります。symlink()関数はエラーが生じるとゼロ以外の数を返すので、その場合はsleep(1)を実行することになります。これは1秒間プログラムの実行を停止する関数です。よって1秒後に再びsymlink()関数を実行することになります。

最初にこの部分を実行したプログラムは問題なく「aaa」を作成しこの後に書かれたファイルの処理を行います。遅れてきた同じプログラムは「aaa」が既に存在するために、symlink()関数がエラーになって寝て待つことになります。ファイル処理が全て終わったところで、次のunlink()関数を実行して「aaa」を消してやると、寝て待っていたプログラムがファイル処理にはいります。

```
unlink("aaa");
```

問題はなんらかのトラブルでプログラムが「aaa」を消さないうちに実行終了になってしまうと、それ以降プログラムはみんなsymlink()関数のところでひっかかって先へ行けなくなってしまうことです。

演習問題

カウンタープログラム(count.cgi)にファイルの排他制御の機能をもたせよ。

2.4 Cookieによる利用者の識別

WWWのサーバーと利用者の間は毎回接続されては切れると言う特徴があります。そのために同じ利用者が再び接続してきた場合に何か特別な待遇をしようとすると、それなりの工夫が必要となります。

その一つとしてtype="hidden"のinputタグを用いる方法があります。利用者に表示するページに必ずこのタグを使用して何らかの識別内容を設定しておき、再び利用者からアクセスがあったときには、設定しておいた内容で判断します。ただこのやり方では、直接利用者には設定した内容は見えませんが、ちょっと操作すれば見えてしまいます。よって偽造などがされやすいのと、inputタグを利用するので必ずFormを使用しなければならない等の問題があります。

ここではCookieと呼ばれるこれとは別のやり方を紹介します。これを利用すると、

- 利用者に設定した内容を見られない。
- 指定した期限まで設定した内容がブラウザ(パソコン)に残る。

- 設定したディレクトリーの中のファイルに対しての応答の場合には、常に設定した内容が返ってくる。

などの特徴があります。なおCookieの具体例とそのプログラムは、

http://cc01.center.sugiyama-u.ac.jp/~miki/Cookie/

を参照してください。

2.4.1 Cookieの設定

Cookieは通常のHTMLのファイルやCGIのプログラムで設定することができます。設定されたCookieは期限の指定があればパソコンのディスクに、期限が無ければパソコンのメモリーに記憶されます。後者の場合、利用者がブラウザのプログラムを終了すると消えてしまいます。Cookieの設定には次のような2つのやり方があります。

- Headと/Headタグの間に、

  ```
  <Meta HTTP-EQUIV="Set-Cookie" Content="設定内容">
  ```

  を入れる。これはHTMLのファイルでもCGIプログラムでも可能です。

- 「Content-type: text/html」や「Location: URL」を出力する前に、

  ```
  Set-Cookie: 設定内容
  ```

  と言う形で出力する。これはCGIプログラムでしか利用できませんが、Cookie で設定した内容が利用者から見えないので偽造の恐れが少なくなります。

どちらのやり方にしても「設定内容」は次のようなものになります。

```
変数名=値; expires=値; domain=値; path=値; secure
```

「変数名=値」以外は省略可能です。それぞれ、次のような意味があります。

- 変数名=値

  好きな変数名に任意の値を指定します。セミコロン(;)、カンマ(,)、空白や日本語を使用する際にはそれぞれ、%3B、%2C、%20のようにコード化する必要があります。

- expires=値

  クライアント側のディスクに記録されるCookieの有効期限を指定します。値は次のような形式で指定します。

  ```
  Fri, 31-Dec-1999 23:59:59 GMT
  ```

  この項目を省略した場合、Cookieはクライアント側のメモリーにのみ残ります。また過去の値を指定すると直ちにCookieを削除します。

- path=値

  Cookieを発行するページを指定します。このパスを含むページを開いた時に、ブラウザからサーバーへCookieが送信されます。通常、「http://WWWサーバー/ディレクトリ/ファイル」の「/ディレクトリ」の部分になります。省略した場合は今開いたファイルと同じディレクトリーにあるファイルにアクセスした場合にCookieが送られます。

- secure

  これを記述しておくと、Cookie情報が暗号化されて送信されるようになります。

2.4.2 Cookieの読み取り

Cookieの読み取りはCGIプログラムでなければできません。ブラウザが設定したCookieの値は環境変数と言うところに設定されるので、CGIプログラムではこれをgetenvと言う関数で読み取ります。そのためには以下のような手順をふみます。

1. プログラムの先頭にgetenvを使うために、

   ```
   #include <stdlib.h>
   ```

   を追加します。

2. 環境変数の値は文字列になります。これを参照するための変数として、次のような文字ポインター変数を宣言します。

   ```
   char *p;
   ```

3. getenv()を使用して「HTTP_COOKIE」と言う環境変数の値を取り込みます。

   ```
   p=getenv("HTTP_COOKIE");
   ```

4. Cookieがなかった場合には、pの値がNULLになります。そうでない場合は、例えば次のようにしてCookieの内容を表示することができます。

   ```
   printf("Cookie=%s\n",p);
   ```

演習問題

アクセスするたびに何回目かを指摘するCGIプログラムを作成せよ。ただしこの回数はその場限り、利用者ごとのものとする。(countc.cgi)

ヒント：まずCookieがあるかどうか確認し、なければ「初めてですね」と挨拶して内容が2のCookieを設定する。もしCookieがあり、その内容がnならば「n回目ですね」と表示した上で内容がn+1のCookieを設定する。文字列を数値に直すにはatoi()と言う関数が利用できる。

```
p="n=123"のとき、
   atoi(p)   → ゼロ
   atoi(p+2) → 123
   atoi(p+3) → 23
```

2.5 応用課題「みきっち」

数年前に「たまごっち」と言う携帯ゲーム？が流行しました。誕生から死ぬまで、えさをやったりウンチの片づけをしたり、色々世話の焼き方によって様々な成長をすると言うものです。年中ぴーぴー注文を付けてくるので、子供が学校に持ち込んだりして社会問題になったりもしました。これに似たものをWebで実現することを考えます。

基本的な筋書きは次のようになります。

- みきっちの状態としては、
  - 小型、中型、大型、死んでいる
  - やせ、正常、でぶ
  - ウンチがある、ない

  が見た目でもわかるものだが、ウンチが出るかどうかはえさを過去にやったかどうかによって決るので、このような外から見える状態だけではできない。よって内部では次のような変数を用いてみきっちの状態を表す。

変数名	値の範囲	変数の内容
day	0,1,2...	これまでの経過時間。
status	1,2,3	それぞれ「やせ」、「正常」、「でぶ」であることを示す。
level	1,2,3,4	それぞれ「小型」、「中型」、「大型」、「死亡」であることを示す。
food	0,1,2...	体内にあるエサの数。食べ過ぎると死ぬ。
weight	0,1,2...	体重。これによってstatusが変る。軽すぎも重過ぎも死ぬ。
age	0,1,2...	年齢。これによってlevelが変る。ある値まで来たら死ぬ。
unchi	0,1,2...	未処理のウンチの数。ある値を越えたら死ぬ。

- 「誕生させる」で開始する。開始時の状態は、小型、やせ、ウンチなし、age、day、foodはゼロ、体重は10とする。
- 開始以後できる操作とその結果状態変数に及ぼす影響は次の通り。

操作	操作の及ぼす影響
放置する (次の日)	day++、 weight−− やせならば1/6、正常ならば1/4、でぶならば1/2の確率でage++ foodが正ならばfood/10の確率でunchi++してfood−−とする。
えさをやる	food++、weightを10増やす。
ウンチを片づける	unchiが正ならばunchi−−。

  各操作に共通なものとして、
  - いつ死ぬか：「weightがゼロ以下」、「weightが100以上」、「ageが100以上」、「unchiが5個以上」
  - 体型の変化：weightが10以下で「正常→やせ」、weightが30以上で「やせ→正常」、weightが50以下で「でぶ→正常」、weightが70以上で「正常→でぶ」
  - 大きさの変化：ageが20以下で「小型」、ageが60以下で「中型」、それ以降は「大型」

とする。以下はこの「みきっち」を少しずつ作成していくことにする。

1. 全体のフレームの定義のファイル(mikichi.html)を作成する。画面を左右に30%と残りに分割する。左側にはメニューのファイル(mmenu.html)、右側にはaction.cgiが表示されるようにする。せっかくだからmenu.htmlにこのファイルへのリンクを追加する。

2. 左側のメニューのファイル(mmenu.html)にはとりあえず、「次の日」、「一覧表示」、「次の人へ」、「トップページに戻る」の4つのリンクを作る。それぞれ、自分のaction.cgiを右側に表示する、自分のichiran.htmlを右側に表示する、次の人のmikichi.htmlを画面全体に、自分のindex.htmlを画面全体に表示するようにする。

3. 最初の段階として「みきっち」を表示するだけのものを作成する。(action1.c)

   ファイル名は順番に変更していくが、コンパイルした後で名前を変更する際は常にaction.cgiにすること。

```
#include <stdio.h>

int day,status,level,food,weight,age,unchi;

void display(){
    今の変数にあった画像を含むページを表示する。
    変数の内容もついでに表示する。
}

main(){
   day=12;status=2;level=1;food=3;weight=15;age=22;unchi=4;
   display();
}
```

4. Cookieがなかった場合の関数を作成する。(action2.c)

```
void setCookie(){
   Cookieを設定する(複数の変数を設定するにはSet-Cookieの行を複数出力する)
}

void noCookie(){
   変数に初期値を設定をする
}

main(){
   noCookie();setCookie();display();
}
```

5. Cookieがあった場合に変数の読み込みを行う関数を作成する。(action3.c)

   複数の変数を設定してCookieを読み込むと次のような形になる。

```
   "x=1; y=2; z=3;"
```

   ここでsscanfと言う関数を使用すると一発で変数a、b、cに読み込める。

```
p=getenv("HTTP_COOKIE");
sscanf(p,"x=%d; y=%d; z=%d;",&a,&b,&c);
```

これを利用して以下を実現する。更新をする度にウンチが増えれば良い。

```
void getdata(){
   Cookieがなければ noCookie() を呼ぶ。
   そうでなければCookieから変数に値を読み込む。
}

main(){
   getdata();
   unchi++;
   setCookie();
   display();
}
```

6. 「みきっち」に対して行える操作として、「次の日」の他に「エサをやる」、「ウンチを捨てる」、「誕生させる」が考えられる。これに対してそれぞれ別のプログラムを作成する方法も考えられるが、内容の違いは変数の操作の部分に限られるので同じプログラムで対応することとする。同じプログラムで異なる動作をさせるために一番簡単な方法として、パラメタを使用する。

動作	パラメタ
次の日	パラメタ無し
エサをやる	1
ウンチを捨てる	2
誕生させる	3

よって呼び出す側は、

```
<A href="action.cgi?3">誕生させる</A>
```

となり、呼び出される側は、

```
main(int argc, char *argv[]){
   ...
   if (atoi(argv[1])==3) { 誕生させる処理 }
   ...
```

のような形となる。以上をもとに、mmenu.htmを修正し、プログラムは「次の日」以外に対応できるようにせよ。(action4.c)

7. パラメタがなかった場合や「次の日」の処理(nextDay())をプログラムに追加せよ。ただし死亡状態であればこの関数をmain()から呼び出さないこと。確率的な動作が必要な部分は、getpid()で得られた値をもとに実現せよ。(action5.c)

8. 「次の日」、「えさ」、「ウンチ」の操作の後に共通して行う部分をmain()に追加せよ。また死亡状態でない場合には、20秒毎に自動更新がかかるようにせよ。(action6.c)

9. 死亡したときにdayの値と日付をrecordと言うファイルに追加するようにせよ。またmmenu.htmに「飼育記録」と言う項目を追加してクリックすると右側にrecord.cgiが表示されるようにし、record.cgiではrecordと言うファイルの中身をtableタグを使用して表示するものとする。(action7.c、record.c)

   ヒント：日付を調べるにはtime()と言う関数を用いる。しかしこの関数はグリニッジ標準時の1970年1月1日午前0時からの秒数を返すだけなので、通常この結果を我々の扱いやすい形式に直す関数を利用する。その中でも一番簡単なのはctime()だが、使い方にちょっと難しい点があるので、次のサンプルプログラムを参考にせよ。

```
#include <stdio.h>
#include <time.h>              /* time()やctime()を使用する際に必要  */

main(){
   time_t t;                   /* time_tは秒数を入れるための変数の型 */

   t=time(NULL);
   printf("%s",ctime(&t));     /* ctime()は\n付きの文字列を返す。     */
}
```

   これをコンパイルして実行すると次のような感じの出力が得られます。

```
   Wed Dec 15 11:53:37 1999
```

2.6 応用課題「簡易チャット ルーム」

Webを利用したチャットと言うのが結構流行しているようです。要するに複数の利用者が書き込みをする伝言板のようなものです。以前過激な書き込みをして大学にクレームが付くと言う事件がありました。別に名乗っていなくても、どこのパソコンから書き込んだかはすぐわかりますので、調子に乗らないようにしてください。

この応用課題では簡易版と言うことで通常のものとちょっと違う形になっています。
(例が`http://cc01.center.sugiyama-u.ac.jp/~miki/chat.html` にあるので必要があれば参照せよ。)

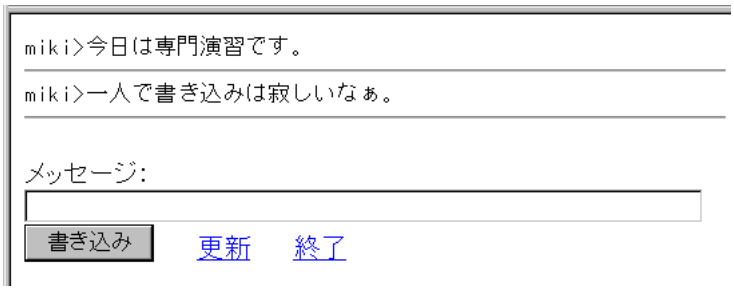
- chat.htm: 利用者受け付けページ。ここで利用者名を入力してもらう。「Start」をクリックしたらchat.cgiにstartと言うパラメタを付けて呼び出す。
- chat.cgi: パラメタの有無で動作が変ります。
  - パラメタ＝startの場合：利用者名をCookieとして設定してからチャットの画面を表示する。
  - パラメタ＝endの場合：Cookieを消去してからchat.htmを呼び出す。(「終了」のボタンに対応)
  - パラメタ＝inputの場合：入力された文をファイルに追加してからチャットの画面を表示する。(「書き込み」のボタンに対応)
  - パラメタ無しの場合：チャットの画面を表示する。(「更新」のボタンに対応)

  チャットの画面では、ページの上部にこれまでのやり取りを順番に並べ、一番下にメッセージの入力欄と「書き込み」、「更新」、「終了」のボタンやリンクがあるものとする。

以下の手順で作成してみよ。

1. chat.htm を作成せよ。名前の入力欄には namae と言う名前を付けること。また menu.htm に「簡易チャット」と言う項目を追加して、画面の右側に呼び出せるようにせよ。

2. 最後の入力欄と「書き込み」、「更新」、「終了」のボタンやリンクを表示するだけの chat.cgi を作成せよ。ただしこの表示の部分は display() と言う関数にすること。(chat.c 以下同様)



3. パラメタが無い場合は単に display() を呼ぶようにせよ。

4. パラメタが start の時に、namae の入力欄に入力された内容を Cookie として設定する関数 setCookie() を呼び、次に display() を呼ぶようにせよ。

5. 「終了」をクリックすると Cookie を消去した上で chat.htm に戻るようにせよ。

6. パラメタが input の時に、入力された内容を「/var/tmp/グループ名.登録名.chat」と言うファイルに追加する関数 input() を呼び、次に display() を呼ぶようにせよ。なおここで「/var/tmp/」で始まるファイル名にするのは、ここはシステムの一時ファイルの置き場になっており、ここに放置されたファイルは自動的に消去されるようになっているからである。書き込まれた内容をずっと保存した場合は別の場所にするが、その場合は別途ファイルを管理するものが必要となる。

   入力される内容は日本語であることは間違いない。すると入力された文字の大部分は%BB%B0%CC%DA のようにコード化されてしまう。これを戻すには例えば次のようにする。

```
for (i=8;line[i]!='\0';i++)
   switch (line[i]) {
      case '+' : fprintf(f," ");
                 break;
      case '%' : i++;
                 if (line[i]>='0' && line[i]<='9') j=(line[i]-'0')*16;
                 else                              j=(line[i]-'A'+10)*16;
                 i++;
                 if (line[i]>='0' && line[i]<='9') j=j+(line[i]-'0');
                 else                              j=j+(line[i]-'A'+10);
                 fprintf(f,"%c",j);
                 break;
      default  : fprintf(f,"%c",line[i]);
   }
```

7. 前述のファイルの内容を表示する部分をdisplay()に追加せよ。通常？のチャットでは最後に入力されたメッセージが最初(一番上)に出てくるが、これをしようとするとファイルから逆順にデータを取り出さなければならない。

8. このままでは更新や入力時にページの表示が先頭までもどってしまうので次のようなJavaScriptの記述を`</Body>`タグの直前に出るようにせよ。

```
<Script language="JavaScript">
<!--
   document.fo.message.focus();
//-->
</Script>
```

fo と言うのはFormに付けた名前である。Formのタグの中で`name="fo"`のような指定を追加する。messageと言うのは入力欄に付けた名前である。

演習問題

1. チャットの最初の画面を表示したときに名前の入力欄にカーソルが自動的に出るようにせよ。(chat.htm)
2. 最初に入れる名前が漢字でも良いようにせよ。(chat2.c)
3. 表示する前にファイルの行数を数えて、最後の50行のみ表示するようにせよ。(chat3.c)
4. ブラウザによっては%bb%b0%cc%daのようにコード化するもののある。これにも対応できるようにせよ。(chat4.c)

3　その他色々

3.1　パスワード付きページ

あるWebページを利用者が見ようとしたときに、パスワードを入力しなければ見ることができないように設定できます。これは大抵のWebサーバーに実装されているBASIC認証とよばれる機能を利用します。BASIC認証は利用者が入力したパスワードがそのままネットワークを流れるので安全ではないと言われますが、普通の用途に用いるのに問題というほどではありません。

パスワード付きにするためには、アクセスの際にパスワードが必要と言うことを示す設定ファイルとパスワードが入ったファイルを作成する必要があります。さらにこの設定ファイルがあるディレクトリー全体にその効果が及ぶので、これまで作成したページにまでパスワード保護が付かないように、新たにディレクトリーを作成し、その中に設定ファイルを入れる事にします。

ディレクトリーの作成

設定ファイルの作成

```
AuthUserFile xxxx/password
AuthGroupFile /dev/null
AuthName Test
AuthType Basic
<Limit GET POST>
    require valid-user
</Limit>
```

パスワードファイルの作成

パスワードファイルの作成を行います。通常このファイルは外部から見えない所に置きます。

```
cd ~
htpasswd -c password ユーザ名
```

ユーザ名は適当な英数字の名前です。こうしてhtpasswdを実行するとここで指定したユーザ名に対するパスワードの入力を促してくるので2回入力します。このとき入力した文字は画面に表示されませんので注意深く入れます。以下にその実行例を示します。

```
cc08[mmiki]%htpasswd -c password guest
Adding password for guest.
New password:                   ← ここでパスワードを入れる
Re-type new password:           ← ここで同じパスワードをもう一回入れる。
```

複数のユーザ名を登録することが可能です。その場合は、

```
cc08[mmiki]%htpasswd password ユーザ名
```

のように-cの無いものを繰り返します。

3.2 Makefile

ちょっとしたシステムでも多数のプログラムから構成されています。特にWeb 関連ではCGIのプログラムが多数必要となります。これは通常のプログラムでは最初から終わりまで一つのプログラムが面倒を見ることが多いのに対して、Webでは一つの画面表示ごとにCGIプログラムの実行が終了してしまうからです。

プログラムの開発段階では繰り返しコンパイルや設定変更をしなければなりません。また大きなプログラムになると複数のファイルに分割されている事が多く、また共通に使われているファイルが変更された場合には、多くのプログラムの再コンパイルが必要になります。

このような作業を簡単に、間違いなく実行できるように考えられたのがmakeコマンドで、このコマンドが作業内容を知るために参照するファイルがMakefile です。

makeコマンドの使い方

基本的な使い方は、単にmakeと打つだけです。

```
cc08[mmiki]%make
```

makeはMakefileの内容を見て、再コンパイルの必要があるものがあれば、Makefileの設定にしたがってコンパイルを行います。ファイルの作成日時を比較した結果何もする必要がなければ、何もメッセージを出さずに終了したり、場合によっては次のようなメッセージを出して終了します。

```
make: Nothing to be done for ‘ALL’.
```

とりあえず再コンパイルしたいものだけコンパイルをしたい場合は、パラメタとして次のような感じで指定します。

```
cc08[mmiki]%make action.cgi
```

Makefileの内容

Makefileの内容は例えば次のようなものです。

```
ALL: action.cgi

action.cgi: action3.c
        gcc -o action.cgi action3.c
        chmod u+s action.cgi
```

「:」の前にあるものが作られるべきファイルで、その後ろに並んでいるファイルは作るときに使用するファイルです。ALLはaction.cgiから作られ、action.cgiはaction3.cから作られるという記述になっています。複数のファイルから作られる場合は、空白で区切って並べます。1行が長くなりすぎたら、行の終わりに¥を入れ、次の行に続きを書くことができます。

「:」行の次に作るためのコマンドを書きます。この時行の先頭は必ずTabで空けます。ここを普通の空白にしてしまう間違いがよくあります。コマンドはいくつ書いても構いません。

make コマンドは何も指定がない場合には、最初の「:」のところを見ます。ALL というファイルと action.cgi を比べようとします。すると後の方に action.cgi の記述があるので先にそれを調べて action.cgi を最新の状態にしようとします。その後で ALL と比べるのですが、ALL というファイルは存在しません。そこで作ろうとするのですが、作るためのコマンドは何も記述されていないのでなにもしないままで終わります。

3.3 ファイルの操作

ファイルの内容を読み込んだり、ファイルに書き込んだり、追加したりする方法については既に述べましたが、ここではファイル自体を操作する方法について説明します。

ファイルの削除

ファイルの削除を行いたい場合は次の関数を利用します。

```
unlink("ファイル名");
```

この関数は、ファイルの削除ができた場合はゼロ、失敗した場合は–1 を返します。よって削除ができなかった場合の対処を if を使って記述することもできます。

```
if (unlink("ファイル名")==-1) {
   puts("ファイルの削除に失敗しました。");
```

ファイルの名前変更

ファイルの名前の変更を行いたい場合は次の関数を利用します。

```
rename("現在のファイル名","新しいファイル名");
```

この関数も、ファイル名の変更ができた場合はゼロ、失敗した場合は–1 を返します。良くある失敗は、指定した「新しいファイル名」のファイルが既に存在する場合です。このような場合はまず unlink() で現在あるファイルを削除してからファイル名の変更を行ってください。