

メディア情報応用 第4週

Advanced Topics in Media Informatics Week 04

金沢工業大学 情報フロンティア学部
メディア情報学科

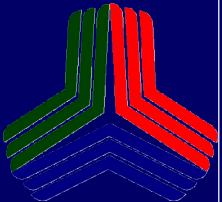
山岸 芳夫

Yoshio Yamagishi

Dept. of Media Informatics

Col. of Informatics and Human Communication

Kanazawa Inst. Tech.

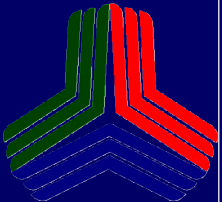


httpd の環境変数

Environment Variables of httpd

- 環境変数：そのプログラムの**実行環境のデータを格納している変数**。Linux の場合はコマンドラインで `set` や `env` コマンドを実行することで、現在コマンドライン実行環境で設定されている環境変数を見ることができる。C言語では**`getenv()`**関数を使うことで環境変数を取得できる

Environment Variables: Variables which contain **some data about execution environment**. In the Linux case, Env. Var. can be shown in command line by using “set” or “env” command. The “**`getenv()`**” function is used to refer to Env. Var. in a C language program.



httpd の環境変数

Environment Variables of httpd

- httpd の環境変数 : httpd の実行環境に置ける各種データを格納している。CGI や PHP などのWebプログラミングで利用可能

Environment Variables of httpd: Variables which contain some data of execution environment of httpd.

Web programs based on CGI or PHP can use Env. Var. of httpd.

代表的な環境変数

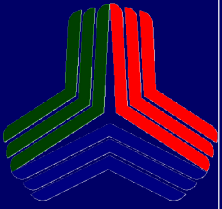
Typical Env. Var. of httpd

- **HTTP_HOST** : httpdが動作しているサーバのホスト名/Name of the host in which the httpd is running.
- **HTTP_USER_AGENT** : クライアントがそのサーバにアクセスしたときに使ったブラウザを表す文字列/Strings which is specific to browser which the client uses to access to the web server.
- **SCRIPT_NAME** : CGI自身のパス名/Path name of the CGI program
- **SERVER_NAME** : サーバ名 (通常は HTTP_HOST と同一) /Usually identical to the HTTP_HOST
- **SERVER_ADDR** : サーバのIPアドレス/IP address of the server.
- **SERVER_PORT** : サーバが使っているポート番号 (通常は 80) /Port number of the server (default value is 80)

代表的な環境変数

Typical Env. Var. of httpd

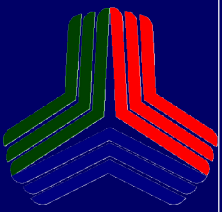
- **REMOTE_ADDR** : クライアントのIPアドレス/IP address of the client.
- **REQUEST_METHOD** : クライアントがサーバに要求したメソッド/Methods of which the client requests to the server.
- **QUERY_STRING** : GET メソッドを利用してデータをWebプログラムに送る場合の、送るデータ (**クエリ**) の文字列/Query strings which the client sends to web program running on the server with GET method



練習/Practice

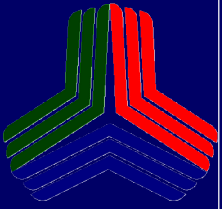
- Moodleの4週目のweek04.tar.gzをダウンロードして展開し、作成された week04/ ディレクトリの中にある **envcgi.c** をコンパイルして **env.cgi** という実行ファイルを作ってCGIとして実行してみよう。

Download “week04.tar.gz” from 4th topic of ATMI course of Moodle, uncompress them, enter to week04/ directory and then compile envcgi.c to make env.cgi executable. Copy env.cgi to /var/www/cgi-bin and then access <http://localhost/cgi-bin/env.cgi>



tar

- Tape ARchive の略。もともとはファイル/ディレクトリをテープにアーカイブ（バックアップ）するためのアプリケーションだが、現在は主に複数のファイル/ディレクトリを一つにまとめたアーカイブファイルの作成、展開に用いられる。
An abbreviation of Tape Archive. Originally tar is an application to archive files/directories to the tape, but currently tar is mainly used to create/extract an archive file which contains some files and/or directories.
- tar のオプション/option
 - “f” ...テープの代わりにファイルを用いる/Use file instead of tape
 - “v” ...処理の様子を表示する/show process verbosely
 - “c” ... アーカイブの作成/create archive
 - “t” ... アーカイブの閲覧/show list of contents of an archive
 - “x” ... アーカイブの展開/extract archive
 - “z” ... 圧縮オプション（データを圧縮するためアーカイブのサイズを小さく出来る）/compressing option



tar

- tar の使用例/Usage of tar
 - 圧縮アーカイブファイルの作成/create compressed archive
`tar zcvf (archive name) file file ...`
 - 圧縮アーカイブファイルの中身を見る(展開はしない) /list contents of compressed archive
`tar ztvf (archive name)`
 - 圧縮アーカイブファイルからの展開/uncompress and extract compressed archive
`tar zxvf (archive name)`

フォームの復習

“Form” Tags

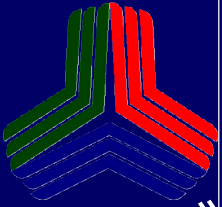
Webページから情報を入力する。<form>～</form>の間に設定
Input interface for web page. They are described between <form>
and </form> tags.

フォームの種類/Various Forms :

- **text**...一行の文字を入力/text(One line)
- **textarea**...複数行の文字を入力/text(Multiple lines)
- **button**...ボタン
- **checkbox**...チェックボックス（単一の項目を選択）/select one item
- **radio**...ラジオボタン（複数の項目を選択）/select multiple items
- **reset**...リセットボタン
- **submit**...送信ボタン
- **select**...メニュー形式による選択/pull down menu

データをCGIプログラムに入力

Data Input to CGI Program



- データの入力にはGETとPOSTの2種類のメソッドを用いることができる。まずGETメソッドを用いた入力方法を使ってみよう。


There are two methods (GET and POST) to input data to CGI programs. Firstly we use “GET” method.

- week04ディレクトリの `formtest.c` をコンパイルして `formtest.cgi` という名前の実行ファイルを作成する。

Compile `formtest.c` in `week04/` directory to build `formtest.cgi` executable.

データをCGIプログラムに入力

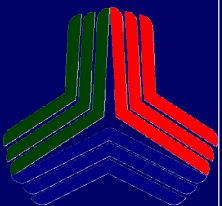
Data Input to CGI Program

- 
- week04 ディレクトリの中の **formtest.html** を **/var/www/html/** にコピーし、formtest.cgi は **/var/www/cgi-bin/** にコピーして、Firefox で **http://localhost/formtest.html** にアクセスしてみよう。フォームから入力されたデータが **QUERY_STRING** に出てくればOK。ただし、テキストフィールドにはアルファベットで入力すること！日本語だと勝手にエンコードされてわけが分からなくなる。

Copy formtest.html and fromtest.cgi to /var/www/html/ and /var/www/cgi-bin, respectively. Then, access **http://localhost/formtest.html** with your Firefox browser. Confirm that the data input from form is displayed as **QUERY_STRING** variable. Note that the characters to input to text field must be alphabet! Japanese character will be broken.

入力文字列の解釈

Format of QUERY_STRING



- QUERY_STRING の値は、GETメソッドの場合、CGI に渡るURLの” ?”以下に続く文字列の内容と同じである
The value of QUERY_STRING equals to the string following after “?” in the URL.
- 通常は、「(フォーム名) = (値) & (フォーム名) = (値) &...」という形式になっている。これらは一つの文字列に入っているので、データとして利用するためには、” &” で切り分け、さらに” =” でも切り分けなくてはならない！

Default format of QUERY_STRING is “(form name) = (value) & (form name) = (value) & ...” so you should split QUERY_STRING by '&' and also '=' when you use these data with your CGI program.

入力文字列の解釈

Format of QUERY_STRING



- 今の場合の” & ”、” = ” のような、いわゆる「区切り文字」は、通常デリミタ(delimiter)と呼ばれる。

Such letters like '&' or '=' which is used to split data are often called as “delimiters”.

文字列を切り分ける関数

Function to split String

- **split_str()** ...山岸オリジナルの文字列を切り分ける関数 / function written by Yamagishi to split string

- 使い方 / Usage

- split_str(元の文字列, 切り分けられた文字列, 区切り文字 (デリミタ), 最大文字数)

split_str(string, splitted string, delimiter, max_size)

- 例 / Example

```
char a[]="yamagishi&yoshio", b[10];  
split_str(a, b, '&', sizeof(b)/sizeof(char));
```

↓
b = "yamagishi" a = "yoshio"

URLをデコードする関数

Function to decode URL

- `url_decode()` ...山岸オリジナルのURLをデコードする関数 / function written by Yamagishi to decode URL
- 使い方 / Usage
 - `url_decode(デコードする文字列, デコードされた文字列, 最大文字数)`
`url_decode(string_to_decode, decoded_string, max_size)`
 - 例 / Example

```
char a[]="%E3%81%82%E3%81%84%E3%81%86";  
char b[20];  
url_decode(a, b, sizeof(b)/sizeof(char));
```

↓
b = “あいう”

URLをエンコードする関数

Function to encode URL

- `url_encode()` ...山岸オリジナルのURLをエンコードする関数/ function written by Yamagishi to encode URL

- 使い方 / Usage

- `url_encode(エンコードする文字列, エンコードされた文字列, 最大文字数)`

`url_decode(string_to_encode, encoded_string, max_size)`

- 例 / Example

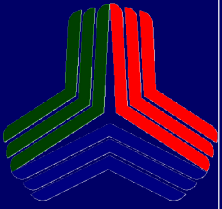
```
char a[]="あいう" ;
```

```
char b[20];
```

```
url_decode(a, b, sizeof(b)/sizeof(char));
```



```
b = "%E3%81%82%E3%81%84%E3%81%86"
```

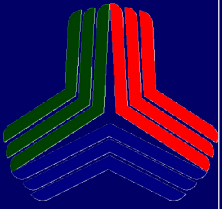
cgilib.c

- `split_str()`, `url_decode()`, `url_encode()`といった関数は、**cgilib.c** というファイルの中で定義されている。week04 ディレクトリの中にあるので、興味があったら中身を見てみよう。

`split_str()`, `url_decode()` and `url_encode()` are defined in the file **cgilib.c** in week04/ directory. You can check it out.

- 以降、これらの関数を含むソースファイルは全て **cgilib.c** と一緒にコンパイルすること。

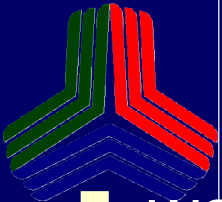
From now on, sample c-language source file should be compiled with **cgilib.c**



複数のソースのコンパイル

How to Compile Multiple Sources

- gcc (ソースファイル名) (ソースファイル名) -o (実行ファイル名)
gcc (source file 1) (source file 2) ... -o (executable)



練習 2 / Practice 2

- week04ディレクトリにある **formtest2.c** をcgilib.c と一緒にコンパイルして、**formtest2.cgi** という名前の実行ファイルを作って、**/var/www/cgi-bin** にコピーすること

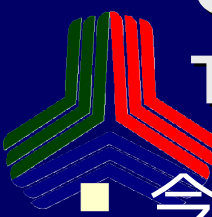
Compile formtest2.c with cgilib.c in week04/ directory to build formtest2.cgi executable. Then, copy formtest2.cgi to /var/www/cgi-bin/ directory.

- /var/www/html/formtest.html の9行目を書き換えて、form タグの action を **formtest2.cgi** にして、<http://localhost/formtest.html> にアクセスして、入力してみよう。今度は日本語もちゃんと表示される。

Rewrite form tag in 9th line of /var/www/html/formtest.html to 'action="formtest2.cgi"' and save. Then, access <http://localhost/formtest.html> and check whether Japanese letters input are shown correctly or not.

CGIの動作を切り換えるテクニック

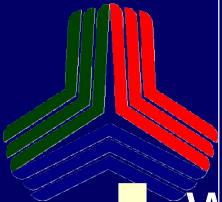
Tips to change the action of CGI

- 
- 今までは HTML ファイルから CGI を呼び出すようにしてきたが、CGI でHTMLを生成すれば、別にHTMLファイルを使わなくても CGI だけで全て完結することができる。

Our applications have called CGI from HTML file so far, but we can omit HTML file if CGI program generates HTML by itself.

- この場合、良く使われるテクニックは、クエリ (QUERY STRING) の有無やその値によってCGIの動作を切り換える方法である。これはPHPなど他の Web アプリケーション開発でも有効なので覚えておこう！

In this case, the methods to change action of CGI by QUERY_STRING are often used. This tips are also available with some other web application platform like PHP.



練習 3 / Practice 3

- week04ディレクトリにある **bmi.c** をcgilib.c と一緒にコンパイルして、**bmi.cgi** という名前の実行ファイルを作って、**/var/www/cgi-bin** にコピーすること。

Compile bmi.c in week04/ directory with cgilib.c to build bmi.cgi executable. Then, copy bmi.cgi to /var/www/cgi-bin/ directory.

- **http://localhost/cgi-bin/bmi.cgi** にアクセスして、ちゃんとBMIが計算できることを確かめよう。

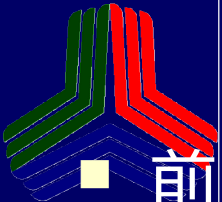
Access <http://localhost/cgi-bin/bmi.cgi> and make sure to calculate BMI



練習 3 / Practice 3

- 同じように **bmi2.c** もコンパイルして、**bmi2.cgi** という名前の実行ファイルを作って /var/www/cgi-bin にコピーし、アクセスしてみよう。これは名前が入力できる。bmi.c, bmi2.c のソースの違いを調べておこう。

Simulary, compile bmi2.c to build bmi2.cgi executable, copy executable to /var/www/cgi-bin and then access. You can input your name in bmi2.cgi. Check out the difference between bmi.c and bmi2.c.



POSTメソッド/POST Method

- 前回まで使ってきたGETメソッドは、URLにデータが全て表示されてしまうので、大量の、もしくは秘密のデータを送信するのには向いていない。

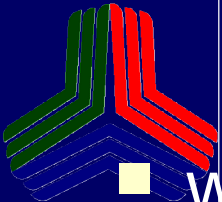
Although we have used GET method so far, since all of the input data is shown in URL, GET method is not favorable for the transmission of secret/huge data.

- POSTメソッドを利用すれば、URLにデータを表示させずに送信することができる。

You can transmit data without URL description when you use POST method.

- POSTメソッドでのCGIとWebサーバとのデータの送受信は、環境変数ではなく、標準入出力を用いる。

The data communication between CGI and Web server is not by environment variable but standard I/O.



練習4 / Practice 4

- week04ディレクトリにある `bmi-post.c` を `cgilib.c` と一緒にコンパイルして、`bmi-post.cgi` という名前の実行ファイルを作って、`/var/www/cgi-bin` にコピーすること。

Compile `bmi-post.c` in `week04/` directory with `cgilib.c` to build `bmi-post.cgi` executable. Then, copy `bmi-post.cgi` to `/var/www/cgi-bin/` directory.

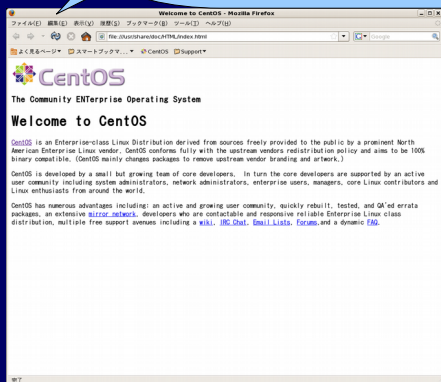
- <http://localhost/cgi-bin/bmi-post.cgi> にアクセスして、ちゃんとBMIが計算できることを確かめよう。 `bmi.cgi` との動作の違い、および `bmi.c`, `bmi-post.c` のソースの違いを調べておこう。

Access <http://localhost/cgi-bin/bmi-post.cgi> and make sure to calculate BMI. Check out the difference between `bmi.c` and `bmi-post.c`.

コラム:GETとPOST

- 今まではWebサーバに情報を送信するのにGETメソッドを使ってきたわけだが、良く考えてみると、GETは文字どおり、サーバから情報を受信するための命令である。それなのに、なぜ情報を送ることが出来るのだろうか？
- いくらGETが情報を受信する命令と言っても、サーバに「これこれこういうURLを送ってくれ」という要求は送信する必要がある。

サーバ、「〇〇」というURLを送ってくれ



ブラウザ



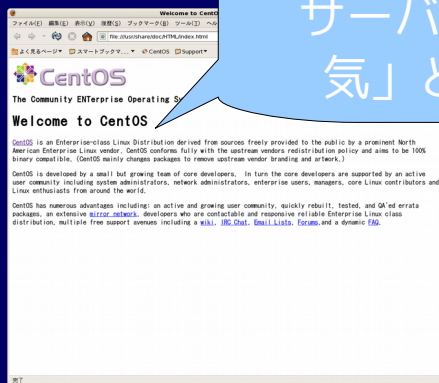
了解



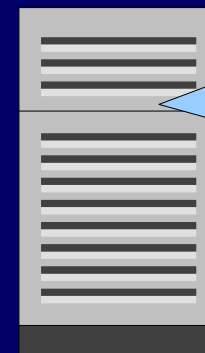
Webサーバ

コラム:GETとPOST

- GETで情報を送信する場合、例えば” GET ○○?xxx=yyy....” という形で、Webサーバに対する**要求の中にちゃっかり情報を取り混ぜて送ってしまっている**のである。（今の例でいえば、 ” ?” 以降の「xxx=yyy....」にあたる部分がサーバにちゃっかり送っている情報）
- このように、GETでの情報送信は少し裏技っぽいので、情報の送信にはPOST（文字どおり「送信」）メソッドを使うのが正道である。ただ、POSTだけでは不便な場合もあるので、ほとんどのWebアプリケーションではGETとPOSTを場合に応じて使い分けている。



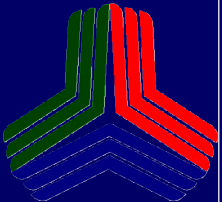
サーバ、「○○?今日はいいい天気」というURLを送ってくれ



なるほど、「今日はいいい天気」なのか

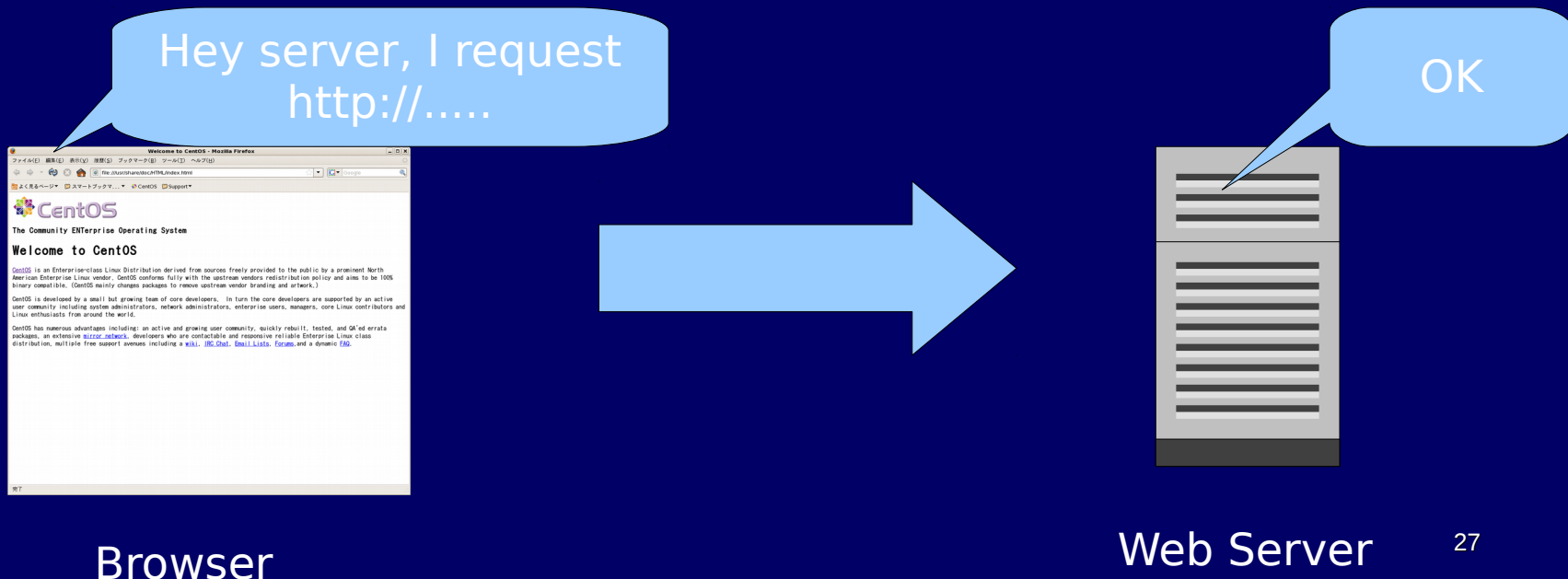
ブラウザ

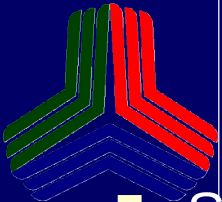
Webサーバ



Column:GET & POST

- We have used GET method to **transmit** data from browser to web server so far. However, basically GET is a method for browser to **receive** data from server. Why we can transmit data to server with GET?
- Although GET is a method to receive data, browser should **send a request** to receive contents of the URL.



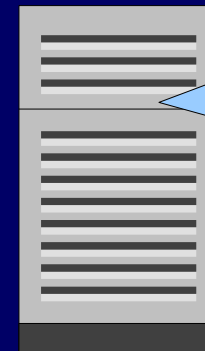


Column:GET & POST

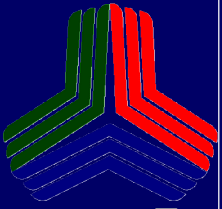
- So, browser can send data to server **embedding data into GET request** like "GET http://...?xxx=yyy...". In this case, "xxx=yyy" part following "?" delimiter in the URL is the data which browser sends to server.
- As shown above, the data transmission with GET method seems to be a little bit tricky. Basically, POST is major data transmission method. However, There are some data transmission cases which POST is not adaptable. So most web applications use both of GET and POST for data transmission.



Browser



Web Server



課題 Assignment

- BMIを求めるCGIのソースを流用して、フォームから入力した値に対して何らかの処理を行った結果を表示する CGI を作成せよ。

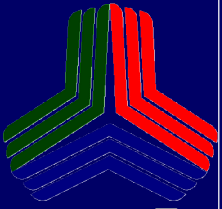
Modifying bmi.c in Practice 3, build CGI Program which processes some data entered from the form.

Ex :

- 任意の二つの整数を入力→最大公約数を表示
Input: Arbitrary 2 integers Output: Greatest Common Major of them
- 消費燃料と走行距離を入力→燃費を表示
Input: Burn-out fuel and travel distance Output: Mileage
- 平成の年数を入力→西暦に変換して表示
Input: Heisei year Output: A.D. year
- ...

- 上記の例のレベルのものができれば80点。さらに独自の工夫が凝らしてあれば最大+20点。

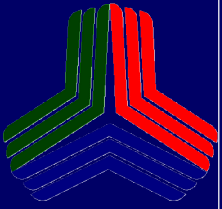
80% grade point will be granted if your program achieves to a same level as above examples. Maximum 20% additional point can be acquired if you arrange more with your original idea.



課題 Assignment

- 用意するファイルは以下の2つである。
You must build and upload below 2 files:
 - readme.txt (どういうプログラムなのかという説明と、工夫した点を書く。emacs で作成せよ must include the concept of the program and arrange points. Writing with Emacs is recommend)
 - CGI のC言語ソースファイル CGI source file
- これらのファイルを tar で（圧縮）アーカイブして、kadai04.tar.gz というファイルにまとめてMoodleにアップロードすること。

Compile them into a tar archive file “kadai04.tar.gz” and the upload to Moodle.



ありがちなミス

Frequent Mistakes

- CGIを実行すると、ブラウザにHTMLのソースが表示されてしまう
The browser does not show the contents but HTML source codes
 - **HTTPヘッダの記述ミス**。ちゃんと最初に「Content-type: text/html」と書かれていて、さらにその下に空行が一行入っているかどうか確認しよう
HTTP header is malformed. Make sure that the string "Content-type: text/html" is on the top of the header, and one empty line is inserted below the string.

ありがちなミス

Frequent Mistakes

- フォームから情報を入力しているのに、CGIに入力されていない
The input data from the form is not reflected in the CGI program
 - **フォームHTMLの記述ミス**。<form> タグの “action” が、正しい CGI プログラム名になっているか？<input> タグの “name” は正しいか？などをチェックしよう
Form HTML is misconfigured. Check that the CGI name described in the "form" tag and/or "name" in the "input" tag are really correct.

これらのミスは gcc のコンパイルでもエラーとして表示はされないなので、注意しよう。

Above mistakes are not compile errors so they can not be shown at the time of compiling.