Open Source Software Project Development

Dr. T.Y. Wong

Week 1

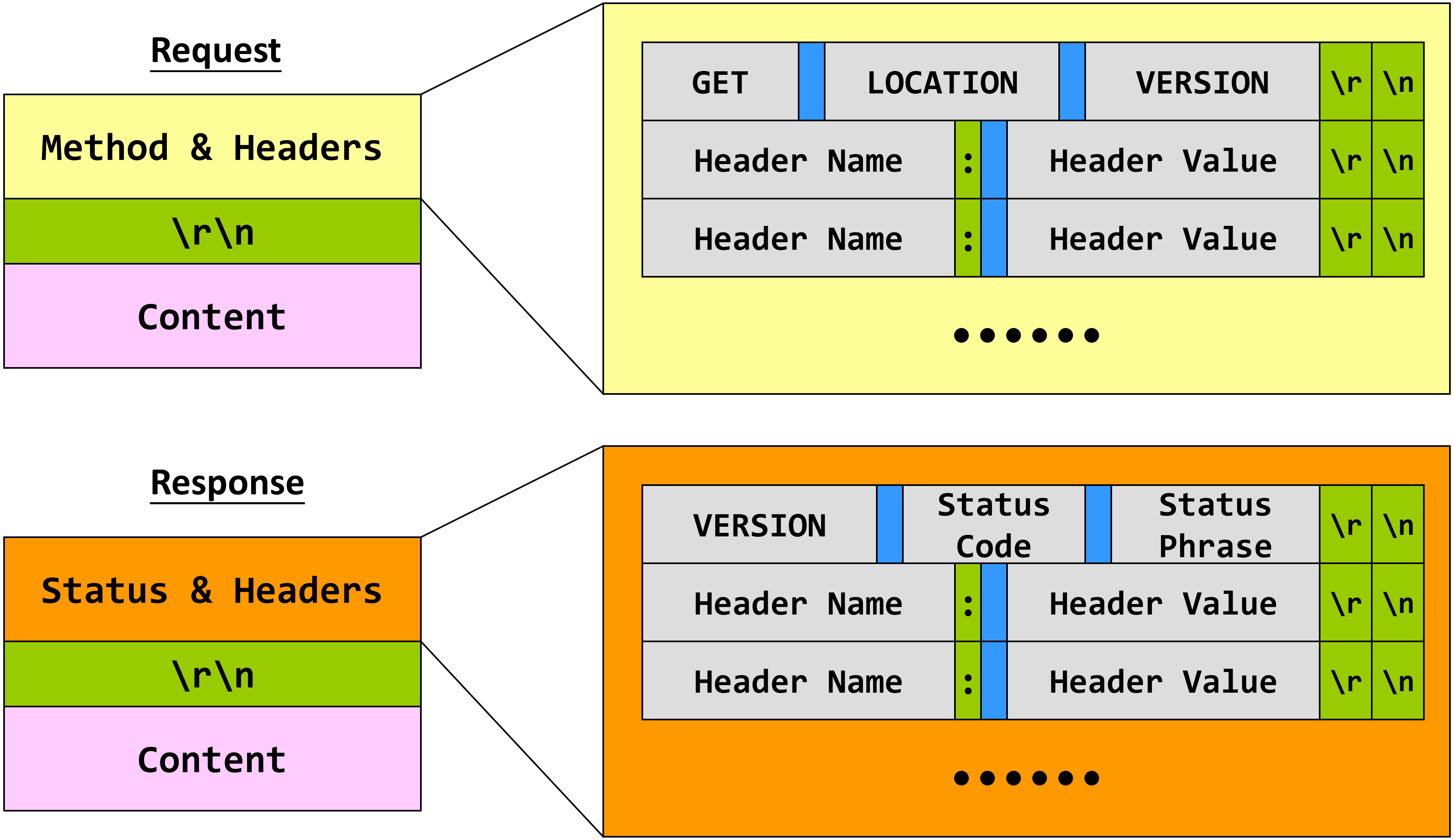
Introduction to Web-based Applications

- simple, yet basic, principles all lies in HTTP.

A Quick Review on HTTP

- for those who missed or forgot CSCI4430.

HTTP Basics



Request

Method & Headers

\r\n

Content

GET | LOCATION | VERSION \r \n

Header Name : Header Value \r \n

Header Name : Header Value \r \n

••••••

Response

Status & Headers

\r\n

Content

VERSION | Status Code | Status Phrase \r \n

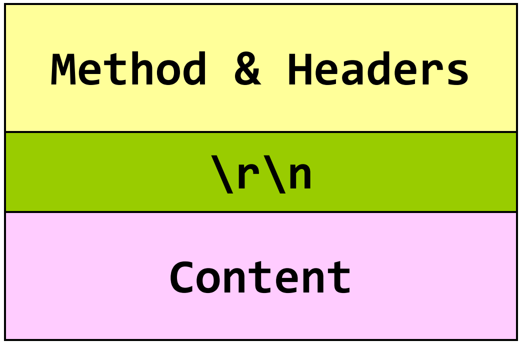
Header Name : Header Value \r \n

Header Name : Header Value \r \n

••••••

HTTP Basics – Request

Request



In a web application, a request will be used as follows:

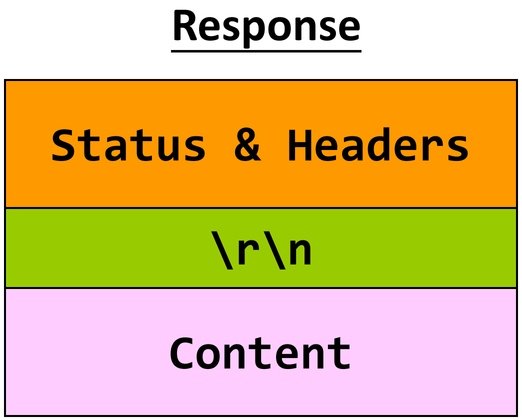
- Embedding **browser information** in the headers;

- Embedding **data input** by the user in either the headers or the content, depending on the 'method'.

Content depends on two things.

Content-Length	:		an integer	\r	\n	If it is 0, then, the content is empty.
Content-Type	:		mime types	\r	\n	The type of the content.

HTTP Basics – Response


Response

Status & Headers

\r\n

Content

In a web application, a response will be used as follows:

- Embedding the data returned in the content.
- Embedding the type of the data returned in the "**Content-Type**" header.

Content depends on two things.

Content-Length	:	an integer	\r	\n	If it is 0, then, the content is empty.
Content-Type	:	mime types	\r	\n	The type of the content.

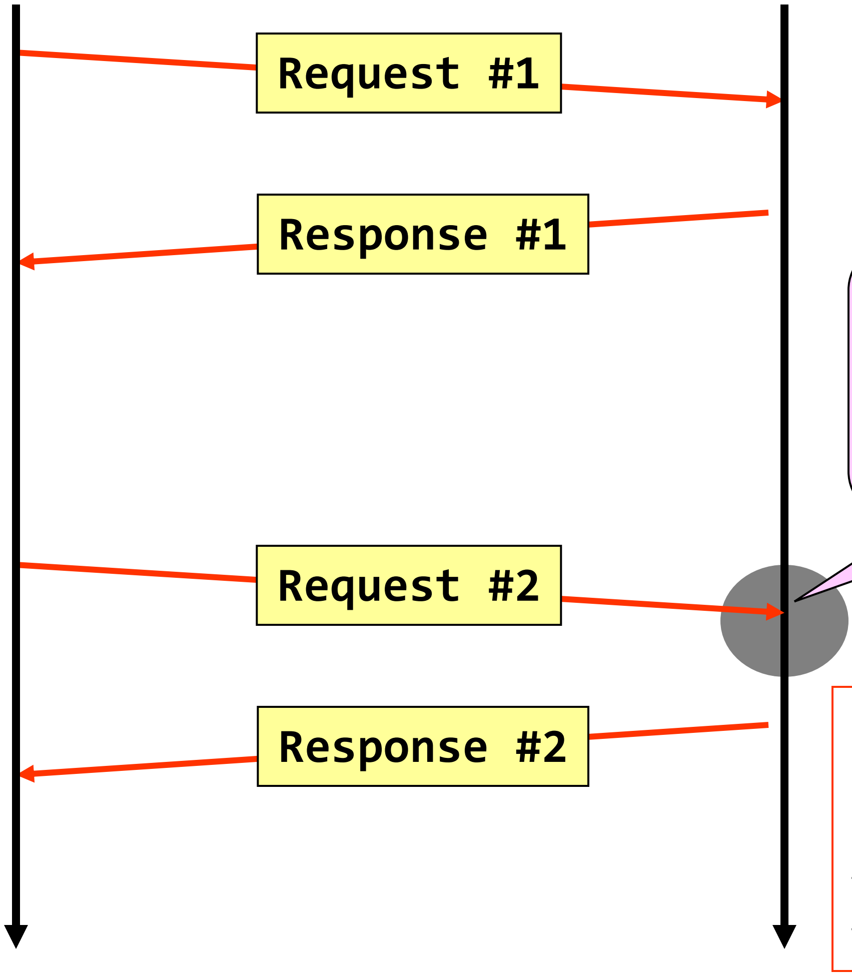
HTTP Basics – Stateless

Although the client may be implemented in the way that:

(1) it memorizes all requests and responses, and

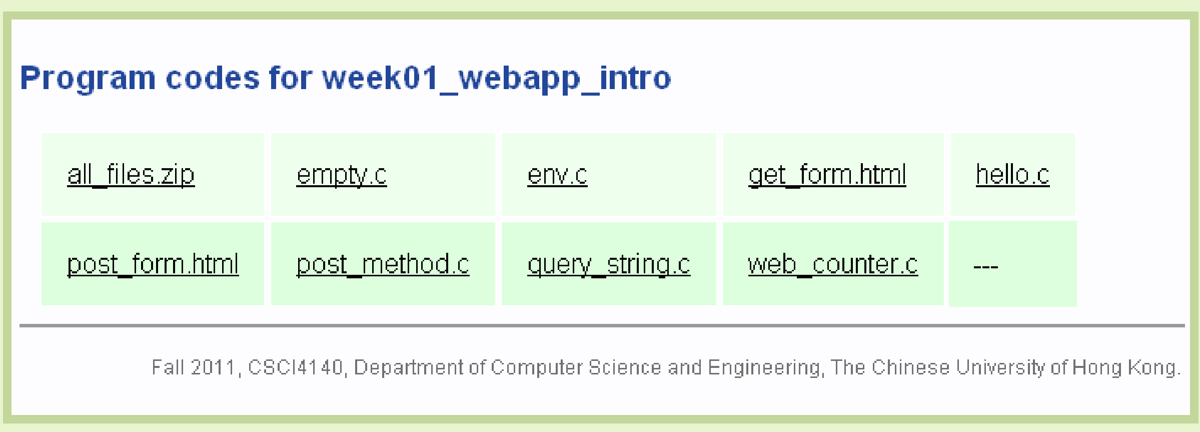
(2) the latest request depends on previous requests and responses,

But...the server is stupid (or stateless)...



I forgot what the contents of “Request #1” and “Response #1” are...

This problem triggers the needs of the **HTTP cookies** and the **session management**.



Program codes for week01_webapp_intro

all_files.zip	empty.c	env.c	get_form.html	hello.c
post_form.html	post_method.c	query_string.c	web_counter.c	---

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

All demonstration program will be provided in the following link.

`http://appsrv.cse.cuhk.edu.hk/~csci4140/cgi-bin/week01_webapp_intro/`

Web Application 101

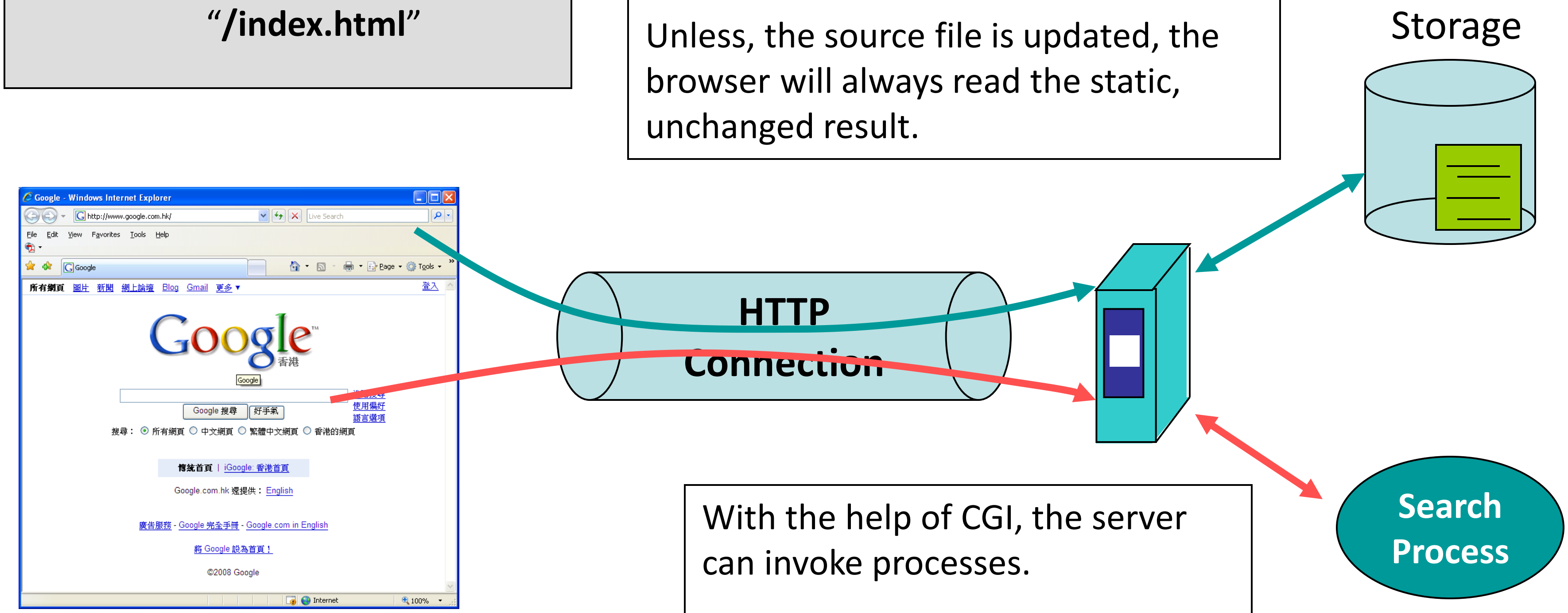
- CGI, common gateway interface

The “*almighty*” web server...

→ to retrieve the file “**/index.html**”

Traditionally, a web server is for transferring files...

Unless, the source file is updated, the browser will always read the static, unchanged result.



With the help of CGI, the server can invoke processes.

Processes can then create different response based on different users’ inputs.

→ when clicking on the button “Google Search”.

Installing a CGI program in dept a/c...

CSE Dept-only

Rule #1: All CGI programs must resides in "~/www/cgi-bin/".

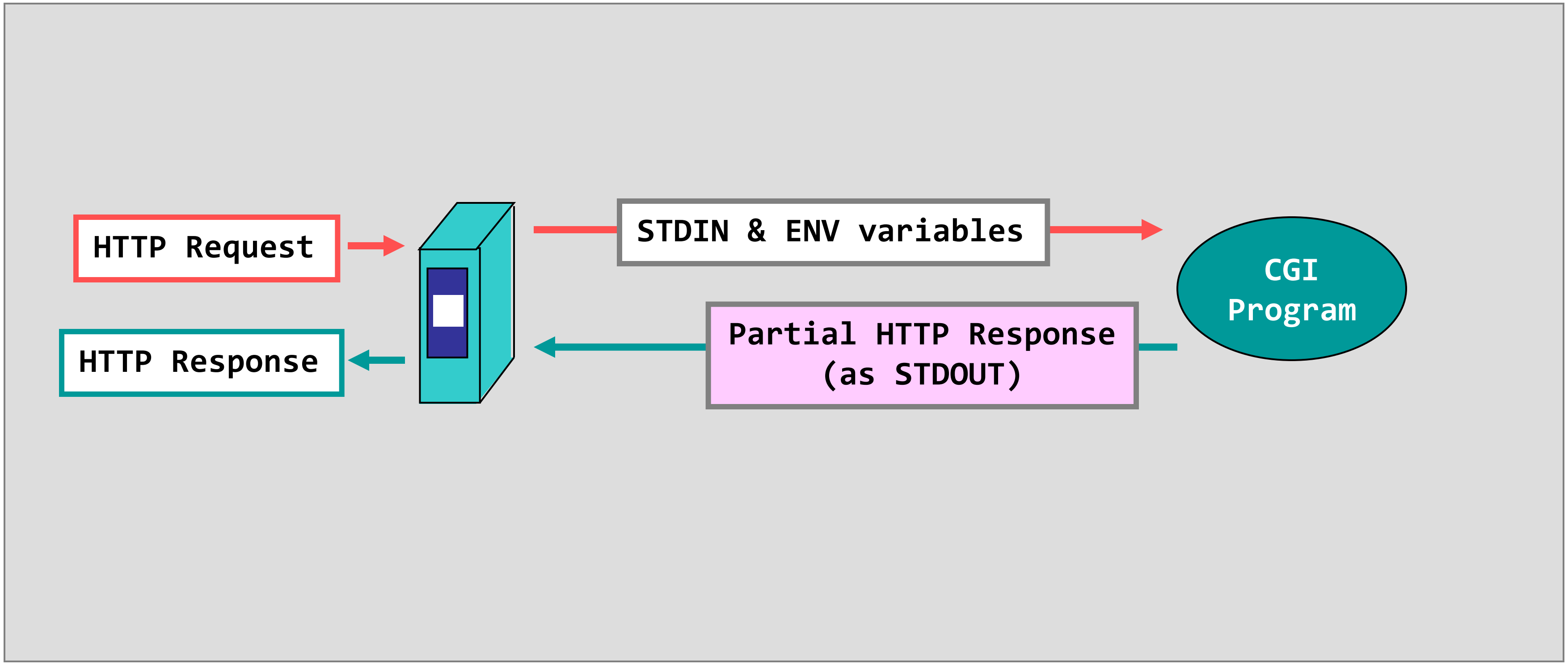
```
sparc15.cs.cuhk.hk:/uac/cact/csci4140/www> ls -ld cgi-bin
drwx--x--x   3 csci4140  cact         512 Sep 4 13:52 cgi-bin/
sparc15.cs.cuhk.hk:/uac/cact/csci4140/www>
```

Rule #2: All CGI programs must have the extension "cgi".

```
sparc15:/.../www/cgi-bin/.../script> ls -l hello.cgi
-rwx--x--x   1 csci4140  cact        6284 Sep 3 21:42 hello.cgi*
sparc15:/.../www/cgi-bin/.../script>
```

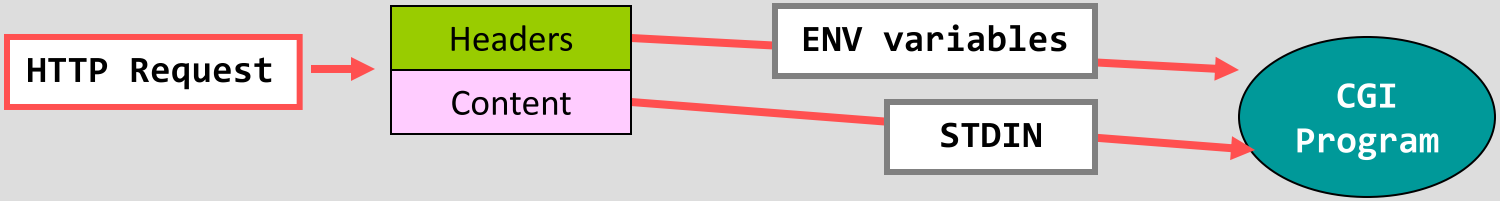
How CGI programs works?

- Let’s understand how the CGI program generates output.



How CGI programs works?

- The request side...

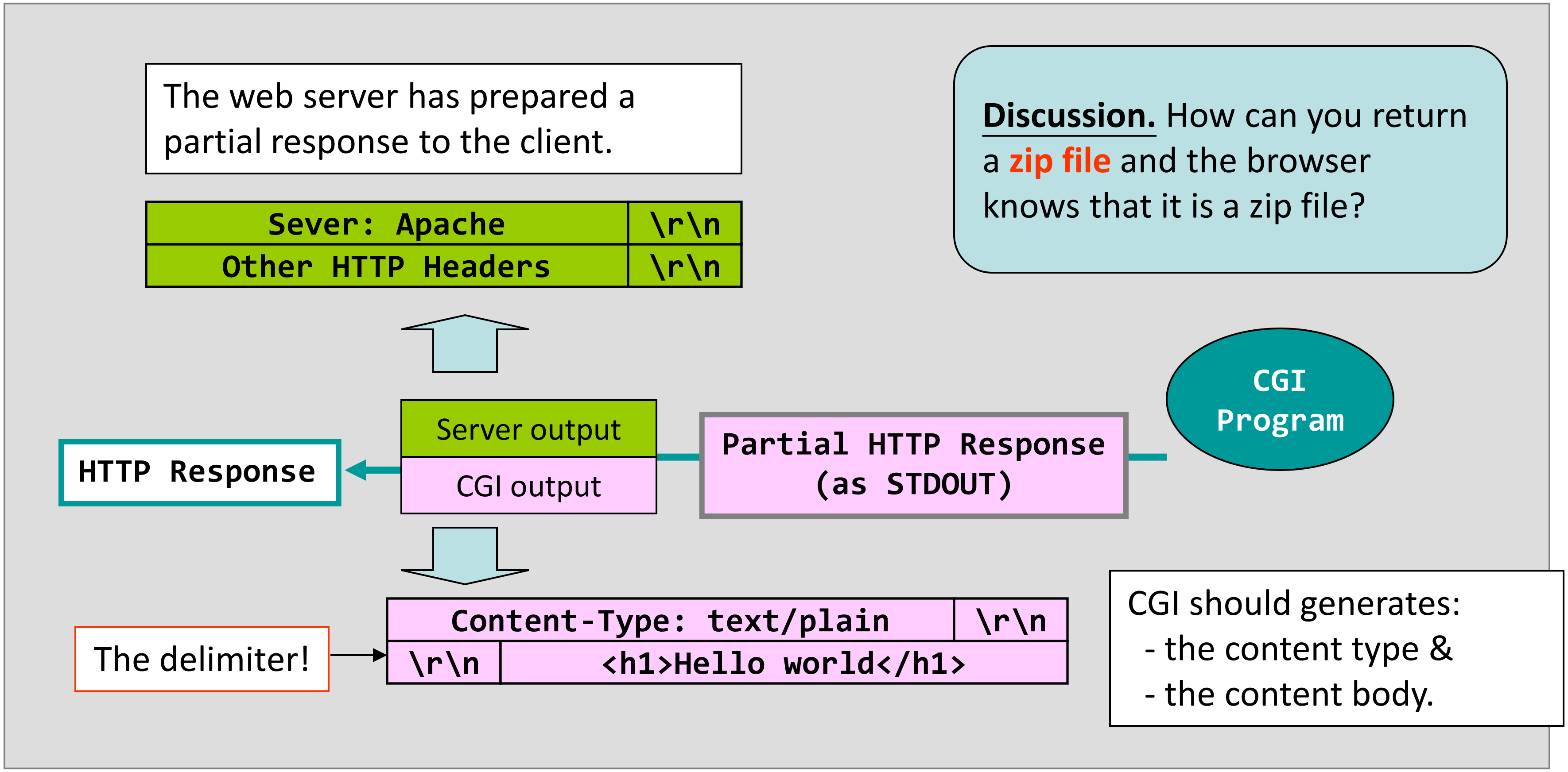


The browser has cheated you!
It leaks a lot of information to web servers through the headers in the request.

`[Example] “env.c”`

How CGI programs works?

- The response side…



The web server has prepared a partial response to the client.

```
Sever: Apache          \r\n
Other HTTP Headers     \r\n
```

Discussion. How can you return a **zip file** and the browser knows that it is a zip file?

CGI Program

Partial HTTP Response (as STDOUT)

Server output

CGI output

HTTP Response

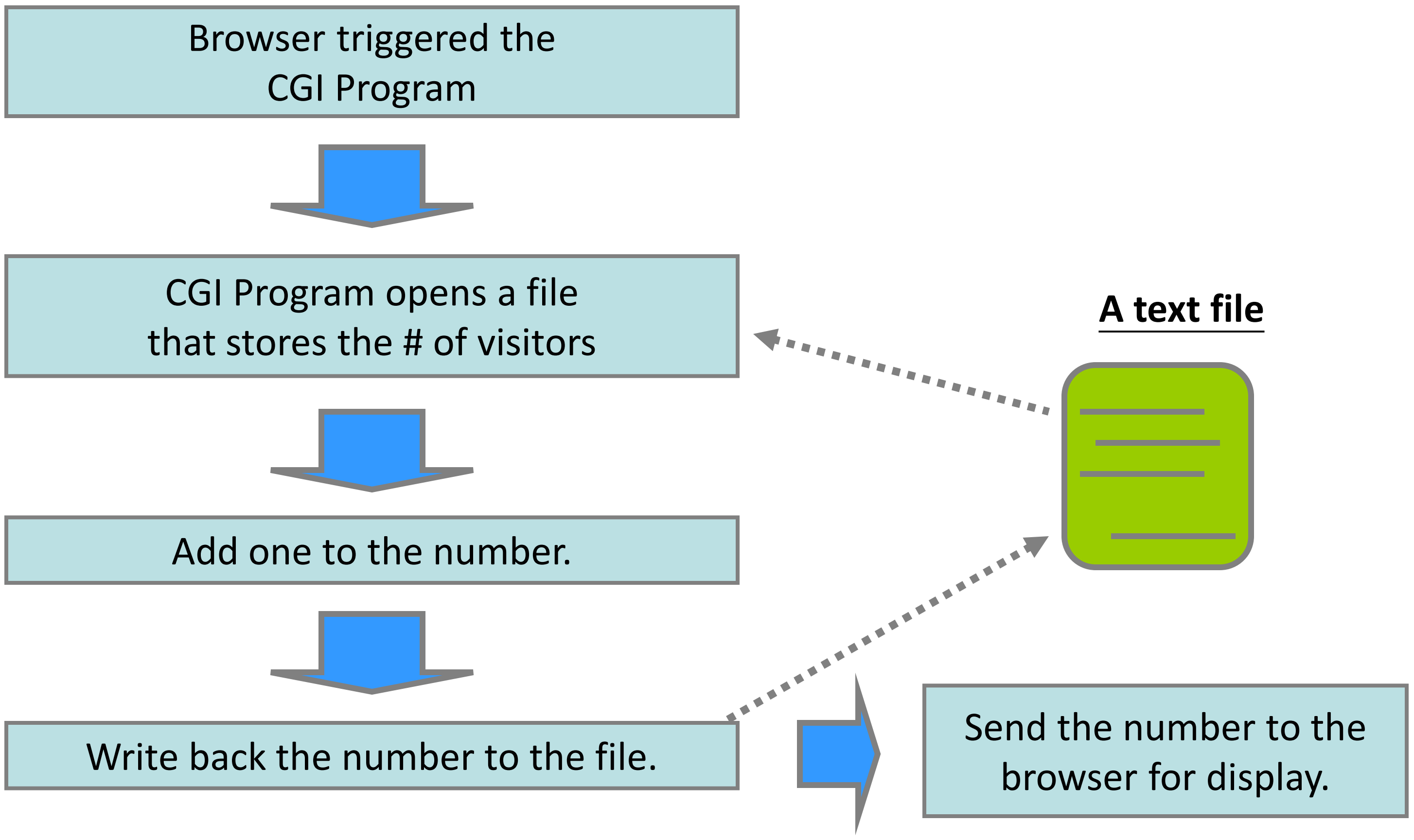
The delimiter!

```
Content-Type: text/plain    \r\n
\r\n    <h1>Hello world</h1>
```

CGI should generates:
- the content type &
- the content body.

`[Example] Try “empty.c” first, then try “hello.c”`

CGI program example: web counter


Browser triggered the CGI Program
CGI Program opens a file that stores the # of visitors
Add one to the number.
Write back the number to the file.
Send the number to the browser for display.
A text file

[Example] “web_counter.c”

Example: web counter - permission?

Common Problem

- The permission of the CGI program highly depends on setting of the web server!

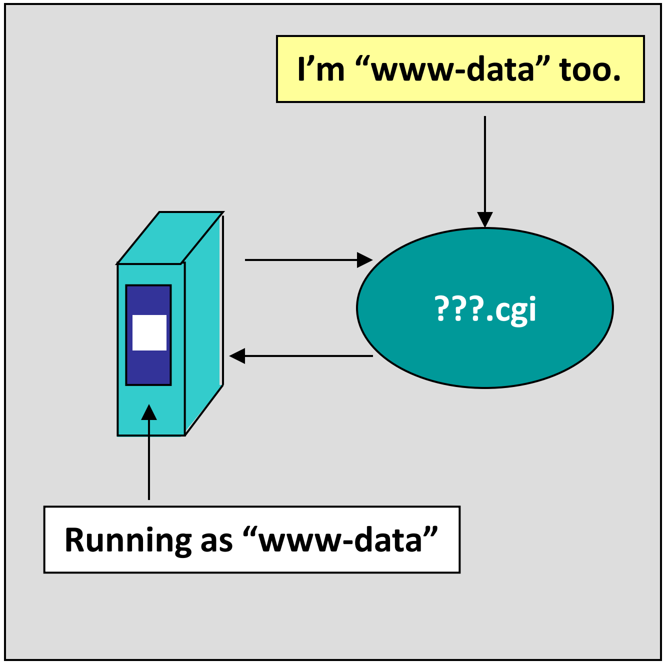
Security Concern

Supporting CGI programs is a very dangerous act… because the server will never know *what kind of programs will be running*.

So, giving CGI processes the root privilege is never a good idea.

Usually, the web server is running as an ordinary user account, e.g., “**nobody**”, “**www-data**”, so that there is no way for any CGI processes to get the root privilege.

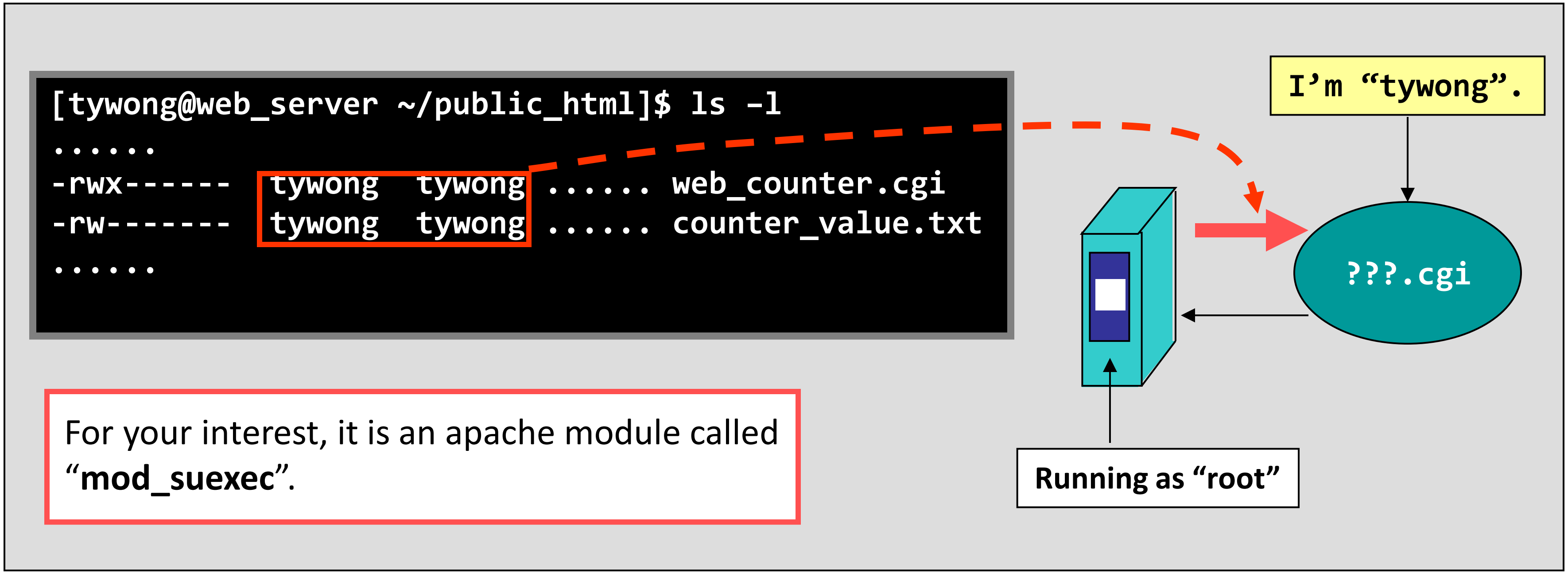
Change mode to “**777**” is not good enough neither...



Example: web counter - permission?

CSE Dept-only

- Some web servers' settings allow:
  - the server to run as "**`root`**", but
  - the server drops the root privilege of a CGI process after it was created.
  - "**`appsrv.cse.cuhk.edu.hk`**" is an example.



```
[tywong@web_server ~/public_html]$ ls -l
......
-rwx------  tywong  tywong ...... web_counter.cgi
-rw-------  tywong  tywong ...... counter_value.txt
......
```

For your interest, it is an apache module called "**mod_suexec**".

User Input?

- So, we now know how to invoke and execute CGI programs.

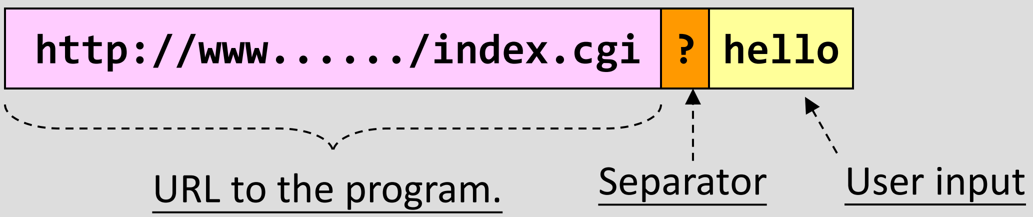
- With the help of tailor-made programs, we can fulfill many tasks.
  - E.g., form submission, web searching, file uploading, etc

- But, how can the user send inputs in the first place…
  - In terms of the HTTP protocol level, there are two methods to set user inputs.

Methods: GET & POST

- defining two kinds of tasks for us...

GET method



`http://www....../index.cgi` `?` `hello`

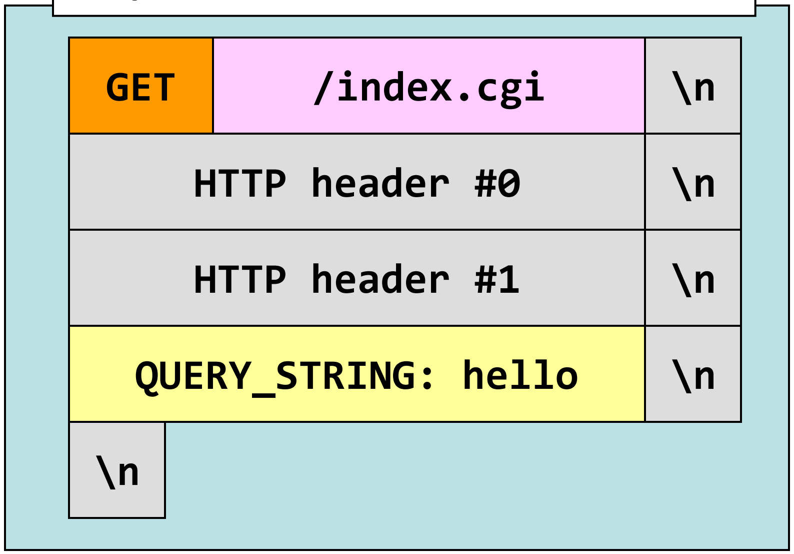
URL to the program.

Separator

User input

Note that there is a limit on the **length of the URL**. Different browsers have different limits.

Request sent from browser to server.



```
GET  /index.cgi          \n
HTTP header #0           \n
HTTP header #1           \n
QUERY_STRING: hello      \n
\n
```

The user input will be stored in the environment variable "**QUERY_STRING**".

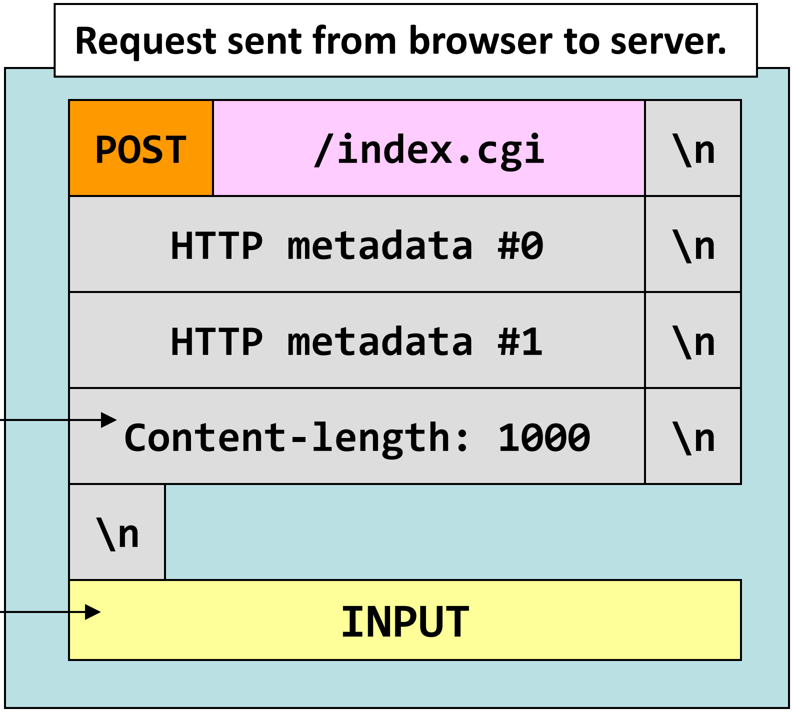
So, if the input is sent using GET method, the CGI program has to process the **environment variables**.

POST method

In this time, the “**QUERY_STRING**”. becomes useless.

What we need is the variable “**CONTENT_LENGTH**”: it defines the length of the input.

The input is converted into the data in the **standard input stream** of the CGI program



Request sent from browser to server.

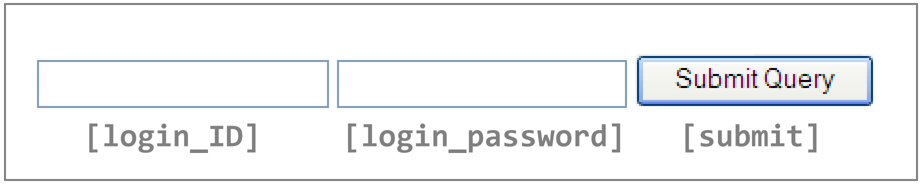
HTML Form and GET & POST Methods

Either GET or POST.

```
<html>
<form method=GET action=query_string.cgi>

<input type=input name="login_ID" />

<input type=password name="login_password" />

<input type=submit>

</form>
</html>
```

The "**method**" field defines which method the browser will use when the "submit" button is pressed.

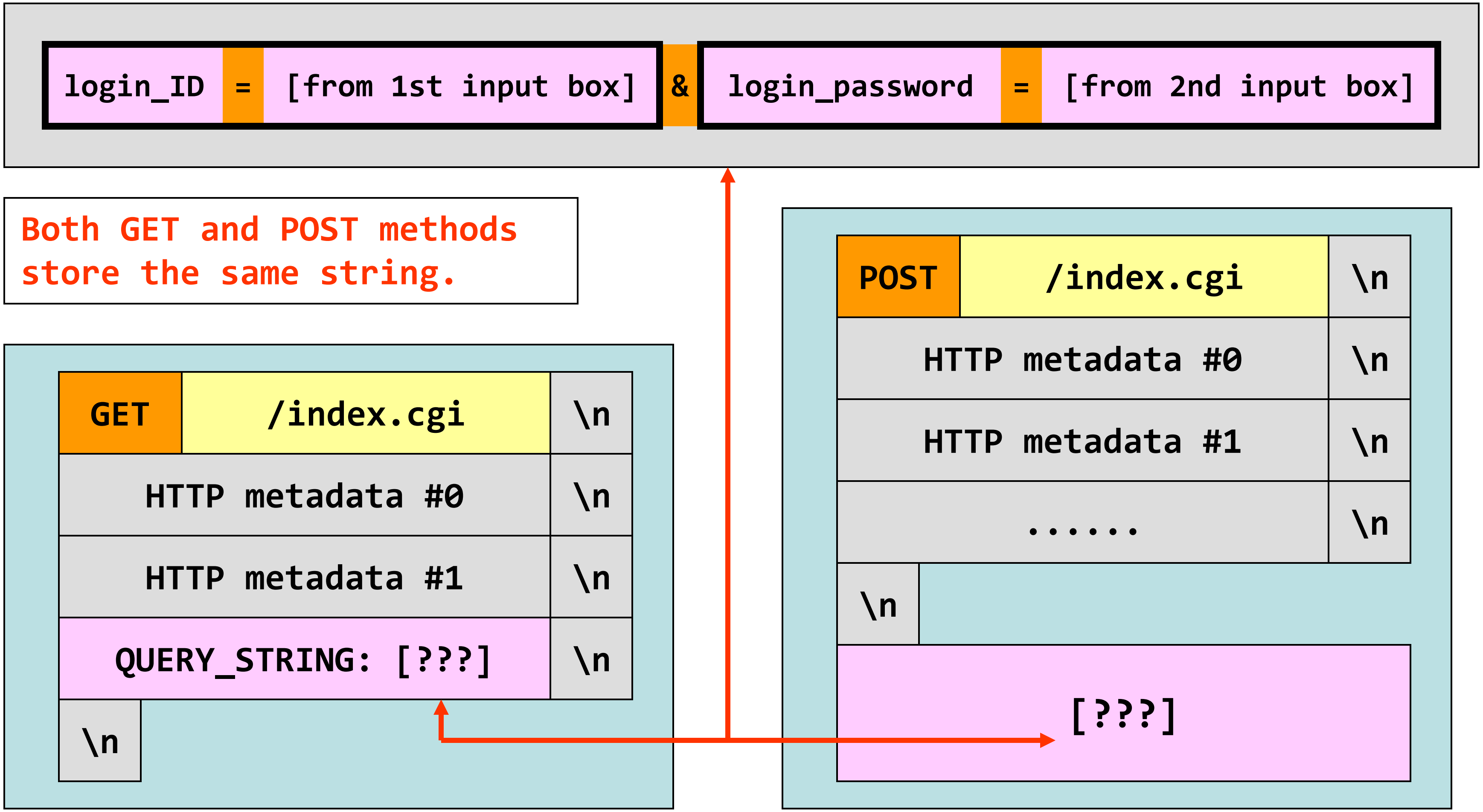
The "action" field defines the location to the CGI program.



When the "submit" button is hit, the browser will send the following string to the web server:

```
login_ID = [from 1st input box] & login_password = [from 2nd input box]
```

HTML Form and GET & POST Methods



login_ID = [from 1st input box] & login_password = [from 2nd input box]

Both GET and POST methods store the same string.

GET /index.cgi	\n
HTTP metadata #0	\n
HTTP metadata #1	\n
QUERY_STRING: [???]	\n
\n	

POST /index.cgi	\n
HTTP metadata #0	\n
HTTP metadata #1	\n
......	\n
\n	
[???]	

[Example] “query_string.c” + “get_form.html” & “post_method.c” + “post_form.html”

HTML Form and GET & POST Methods

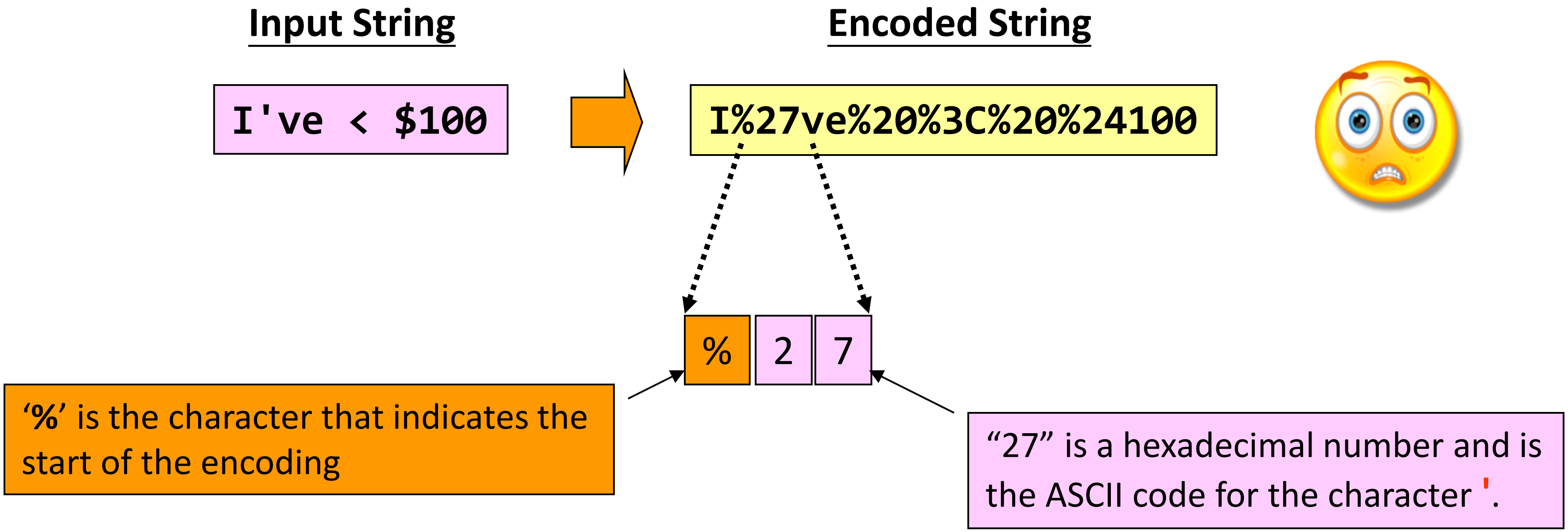
- After reading the code, you can find that:

GET Method	Post Method
Suitable for sending **small amount** of data. It is because of the limit on the length of the URL.	Suitable for sending **large amount** of data.
Not suitable for sending sensitive data.	**A must for sensitive data.** E.g., password transmission.

Usually used for:
(1) session management;
(2) search engine queries.
	Usually used for:
	(1) login interface;
	(2) file upload;
	(3) Form with **`<textarea>`**, e.g., wiki.

HTML Form and GET & POST Methods

- Point to note:
  - the browser will encode escape characters for URL into something else…



Input String

```
I've < $100
```

Encoded String

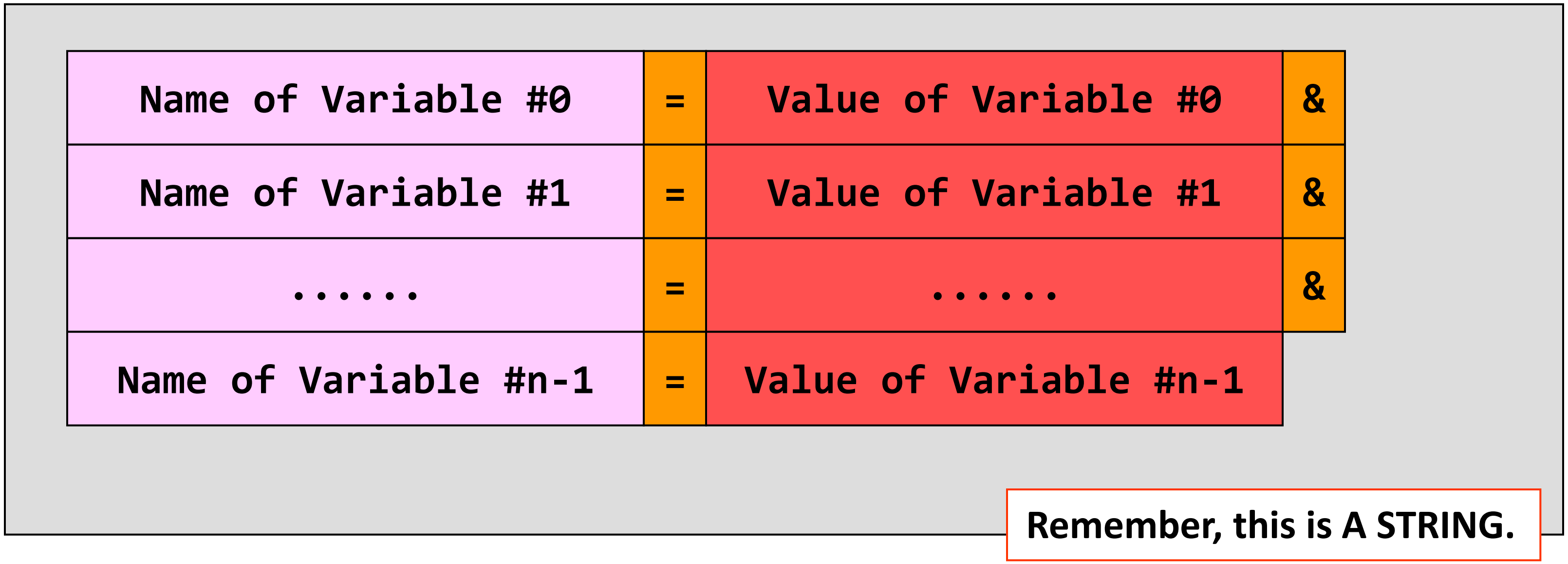
```
I%27ve%20%3C%20%24100
```

'%' is the character that indicates the start of the encoding

"27" is a hexadecimal number and is the ASCII code for the character '.

String Processing…

- Now, we have a query string in the following format:



Name of Variable #0	=	Value of Variable #0	&
Name of Variable #1	=	Value of Variable #1	&
……	=	……	&
Name of Variable #n-1	=	Value of Variable #n-1	

Remember, this is A STRING.

Writing C functions to process such a string is a tedious work.

Leaving C for good...

- In the next lecture, we’ll start learning Perl.
  - Why Perl?
  - Because of its strength in string processing.
  - We are going to write less code, but have a fairly strong program.

- As an example, we’ll use Perl to introduce **what a scripting language is**.