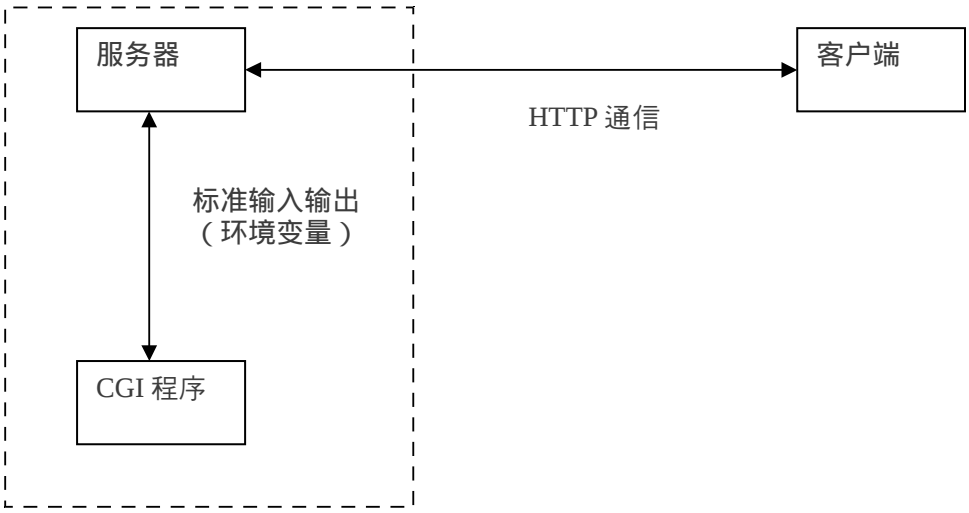
CGI

一．基本原理

CGI:通用网关接口（ Common Gateway Interface）是一个 Web 服务器主机提供信息服务的标准接口。通过 CGI 接口，Web 服务器就能够获取客户端提交的信息，转交给服务器端的 CGI 程序进行处理，最后返回结果给客户端。

组成 CGI 通信系统的是两部分：一部分是 html 页面，就是在用户端浏览器上显示的页面。另一部分则是运行在服务器上的 Cgi 程序。

它们之间的通讯方式如下图：



服务器和客户端之间的通信，是客户端的浏览器和服务器端的 http 服务器之间的 HTTP 通信，我们只需要知道浏览器请求执行服务器上哪个 CGI 程序就可以了，其他不必深究细节，因为这些过程不需要程序员去操作。

服务器和 CGI 程序之间的通讯才是我们关注的。一般情况下，服务器和 CGI 程序之间是通过标准输入输出来进行数据传递的，而这个过程需要环境变量的协作方可实现 。

1. 服务器将 **URL** 指向一个应用程序
2. 服务器为应用程序执行做准备
3. 应用程序执行，读取标准输入和有关环境变量
4. 应用程序进行标准输出

对于 Windows 系统而言，还可以通过 profile 文件进行数据传输（如 ini 文件），但在
这里不做研究。

环境变量在 CGI 中有着重要的地位!每个 CGI 程序只能处理一个用户请求，所以在激活一个 CGI 程序进程时也创建了属于该进程的环境变量。

二．环境变量

对于 CGI 程序来说，它继承了系统的环境变量。CGI 环境变量在 CGI 程序启动时初始化，在结束时销毁。

当一个 CGI 程序不是被 HTTP 服务器调用时，它的环境变量几乎是系统环境变量的复制。

当这个 CGI 程序被 HTTP 服务器调用时，它的环境变量就会多了以下关于 HTTP 服务器、客户端、CGI 传输过程等项目。

与请求相关的环境变量	**REQUEST_METHOD**	服务器与 CGI 程序之间的信息传输方式
	QUERY_STRING	采用 GET 时所传输的信息
	CONTENT_LENGTH	STDIO 中的有效信息长度
	CONTENT_TYPE	指示所传来的信息的 MIME 类型
	CONTENT_FILE	使用 Windows HTTPd/WinCGI 标准时，用来传送数据的文件名
	PATH_INFO	路径信息
	PATH_TRANSLATED	CGI 程序的完整路径名
	SCRIPT_NAME	所调用的 CGI 程序的名字
与服务器相关的环境变量	GATEWAY_INTERFACE	服务器所实现的 CGI 版本
	SERVER_NAME	服务器的 IP 或名字
	SERVER_PORT	主机的端口号
	SERVER_SOFTWARE	调用 CGI 程序的 HTTP 服务器的名称和版本号
与客户端相关的环境变量	REMOTE_ADDR	客户机的主机名
	REMOTE_HOST	客户机的 IP 地址
	ACCEPT	例出能被次请求接受的应答方式
	ACCEPT_ENCODING	列出客户机支持的编码方式
	ACCEPT_LANGUAGE	表明客户机可接受语言的 ISO 代码
	AUTORIZATION	表明被证实了的用户
	FORM	列出客户机的 EMAIL 地址
	IF_MODIFIED_SINGCE	当用 get 方式请求并且只有当文档比指定日期更早时才返回数据
	PRAGMA	设定将来要用到的服务器代理
	REFFERER	指出连接到当前文档的文档的 URL
	USER_AGENT	客户端浏览器的信息

CONTENT_TYPE:如 application/x-www-form-urlencoded，表示数据来自 HTML 表单，并且经过了 URL 编码。

ACCEPT:客户机所支持的 MIME 类型清单，内容如:i"age/gif,image/jpeg"

REQUEST_METHOD:它的值一般包括两种:POST 和 GET，但我们写 CGI 程序时，最后还要考虑其他的情况。

1．POST 方法

如果采用 POST 方法，那么客户端来的用户数据将存放在 CGI 进程的标准输入中，同时将用户数据的长度赋予环境变量中的 CONTENT_LENGTH。客户端用 POST 方式发送数据有一个相应的 MIME 类型（通用 Internet 邮件扩充服务：Multi-purpose Internet Mail Extensions）。目前，MIME 类型一般是：application/x-wwww-form-urlencoded，该类型表示数据来自 HTML 表单。该类型记录在环境变量 CONTENT_TYPE 中,CGI 程序应该检查该变量的值。

2．GET 方法

在该方法下，CGI 程序无法直接从服务器的标准输入中获取数据，因为服务器把它从标
准输入接收到得数据编码到环境变量 QUERY_STRING（或 PATH_INFO）。

GET 与 POST 的区别：采用 GET 方法提交 HTML 表单数据的时候，客户机将把这些数
据附加到由 ACTION 标记命名的 URL 的末尾，用一个包括把经过 URL 编码后的信息与 CGI 程序的名字分开：http://www.mycorp.com/hello.html?name=hgq$id=1，QUERY_STRING 的值为 name=hgq&id=1

有些程序员不愿意采用 GET 方法，因为在他们看来，把动态信息附加在 URL 的末尾有
违 URL 的出发点：URL 作为一种标准用语,一般是用作网络资源的唯一定位标示。

环境变量是一个保存用户信息的内存区。当客户端的用户通过浏览器发出 CGI 请求时，服务器就寻找本地的相应 CGI 程序并执行它。在执行 CGI 程序的同时，服务器把该用户的信息保存到环境变量里。接下来，CGI 程序的执行流程是这样的：查询与该 CGI 程序进程相应的环境变量：第一步是 request_method，如果是 POST，就从环境变量的 len，然后到该进程相应的标准输入取出 len 长的数据。如果是 GET，则用户数据就在环境变量的 QUERY_STRING 里。

3．POST 与 GET 的区别

以 GET 方式接收的数据是有长度限制，而用 POST 方式接收的数据是没有长度限制的。并且，以 GET 方式发送数据，可以通过 URL 的形式来发送，但 POST 方式发送的数据必须要通过 Form 才到发送。

三．CGI 程序实现步骤

1．从服务器获取数据

C 语言实现代码：

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

int get_inputs()
{
    int length;
    char *method;
    char *inputstring;

    method = getenv("REQUEST_METHOD"); //将返回结果赋予指针
    if(method == NULL)
       return 1;      //找不到环境变量 REQUEST_METHOD
    if(!strcmp(method, "POST"))  // POST 方法
    {
       length = atoi(getenv("CONTENT_LENGTH")); //结果是字符，需要转换
       if(length != 0)
       {
            inputstring = malloc(sizeof(char)*length + 1) //必须申请缓存，因为
                                                          stdin 是不带缓存的。
         fread(inputstring, sizeof(char), length, stdin); //从标准输入读取一定数据
        }
    }
    else if(!strcmp(method, "GET"))
    {
       Inputstring = getenv("QUERY_STRING");
       length = strlen(inputstring);
    }
    if(length == 0)
         return 0;
}
```

Perl 实现代码：

```
$method = $ENV{'REQUEST_METHOD'};
if($method eq 'POST')
{
   Read(STDIN, $input, $ENV{'CONTENT_LENGTH'});
}
if($method eq 'GET' || $method eq 'HEAD')
```

```
{
   $input = $ENV{‘QUERY_STRING’};
}
if($input eq “”)
{
     &print_form;
     exit;
}
```

PYTHON 代码实现

```
#!/usr/local/bin/python
import cgi
def main():
     form = cgi.FieldStorage()
```

Python 代码实现更简单，cgi.FieldStorage()返回一个字典，字典的每一个 key 就是变量名，key 对应的值就是变量名的值，更本无需用户再去进行数据解码！

获取环境变量的时候，如果先判断“REQUEST_METHOD”是否存在，程序会更健壮，否则在某些情况下可能会造成程序崩溃。因为假若 CGI 程序不是由服务器调用的，那么环境变量集里就没有与 CGI 相关的环境变量（如 REQUEST_METHOD，REMOTE_ADDR 等)添加进来，也就是说“getenv(“REQUEST_METHOD”)”将返回 NULL！

2．URL 编码

不管是 POST 还是 GET 方式，客户端浏览器发送给服务器的数据都不是原始的用户数据，而是经过 URL 编码的。此时，CGI 的环境变量 Content_type 将被设置，如 Content_type = application/x-www-form-urlencode 就表示服务器收到的是经过 URL 编码的包含有 HTML 表单变量数据。

编码的基本规则是：

变量之间用**“&”**分开；

变量与其对应值用**“=”**连接；

空格用“+”代替；

保留的控制字符则用**“%”**连接对应的 **16** 禁止 **ASCII** 码代替；

某些具有特殊意义的字符也用**“%”**接对应的 **16** 进制 **ASCII** 码代替；

空格是非法字符；

任意不可打印的 ASCII 控制字符均为非法字符。

例如，假设 3 个 HTML 表单变量 filename、e-mail 和 comments，它们的值对应分别为 hello、mike@hotmail.com 和 I’ll be there for you，则经过 URL 编码后应为：

```
   filename=hello&e-mail=hello@hotmail.com&comments=I
%27ll+be+there+for+you
```

所以，CGI 程序从标准输入或环境变量中获取客户端数据后，还需要进行解码。解码

的过程就是 URL 编码的逆变：根据“&”和“=”分离 HTML 表单变量，以及特殊字符的替换。

在解码方面，PYTHON 代码实现是最理想的，cgi.FieldStorage()函数在获取数据的同时就已自动进行代码转换了，无需程序员再进行额外的代码编写。Perl 其次，因为在一个现成的 Perl 库：cgi-lib.pl 中提供了 ReadParse 函数，用它来进行 URL 解码很简单：

```
require ‘cgi-lib.pl’;
&ReadParse(*input);
```

3．CGI 数据输出

CGI 程序如何将信息处理结果返回给客户端？这实际上是 CGI 格式化输出。

在 CGI 程序中的标准输出 stdout 是经过重定义了的，它并没有在服务器上产生任何的输出内容，而是被重定向到客户浏览器，这与它是由 C，还是 Perl 或 Python 实现无关。

所以，我们可以用打印来实现客户端新的 HTML 页面的生成。比如，C 的 printf 是向该进程的标准输出发送数据，Perl 和 Python 用 print 向该进程的标准输出发送数据。

（1） CGI 标题

CGI 的格式输出内容必须组织成标题/内容的形式。CGI 标准规定了 CGI 程序可以使用的三个 HTTP 标题。标题必须占据第一行输出！而且必须随后带有一个空行。

标题	描述
Content_type （内容类型）	设定随后输出数据所用的 MIME 类型
Location （地址）	设定输出为另外一个文档（URL）
Status （状态）	指定 HTTP 状态码

MIME：

向标准输出发送网页内容时要遵守 MIME 格式规则：

任意输出前面必须有一个用于定义 MIME 类型的输出内容（Content-type）行，而且随后还必须跟一个空行。如果遗漏了这一条，服务将会返回一个错误信息。（同样使用于其他标题）

例如 Perl 和 Python：

```
print “Content-type:text/html\n\n”;  //输出 HTML 格式的数据
print “<body>welcome<br>”
print “</body>”
```

C 语言：

```
printf( “Content-type:text/html\n\n”);
printf(“Welcome\n”);
```

MIME 类型以类型/子类型（type/subtype）的形式表示 。

其中 type 表示一下几种典型文件格式的一种：

Text、Audio、Video、Image、Application、Mutipart、Message

Subtype 则用来描述具体所用的数据格式。

Application/msword	微软的 Word 文件
Application/octet-stream	一种通用的二进制文件格式
Application/zip	Zip 压缩文件
Application/pdf	Pdf 文件
。。。。。。。。。。。。。。。。。。。。。。。。。。。	。。。。。。。。。。。。。。。。。。。。。。。。

Location：

使用 Location 标题，一个 CGI 可以使当前用户转而访问同一服务器上的另外一个程序，甚至可以访问另外一个 URL，但服务器对他们的处理方式不一样。

使用 Location 的格式为：Location:Filename/URL，例如：

```
print “Location:/test.html\n\n”;
```

这与直接链接到 test.html 的效果是一样的。

```
print “Location:http://www.chinaunix.com/\n\n”
```

由于该 URL 并不指向当前服务器，用户浏览器并不会直接链接到指定的 URL，而是给用户输出提示信息。

HTTP 状态码：

表示了请求的结果状态，是 CGI 程序通过服务器用来通知用户其请求是否成功执行的信息码，本文不做研究。

四．CGI 中的信号量和文件锁

因为 CGI 程序时公用的，而 WEB 服务器都支持多进程运行，因此可能会发生同时有多个用户访问同一个 CGI 程序的情况。比如，有 2 个用户几乎同时访问同一个 CGI 程序，服务器为他们创建了 2 个 CGI 程序进程，设为进程 A 和进程 B。假如进程 A 首先打开了某个文件，然后由于某种原因被挂起（一般是由于操作系统的进程调度）；而就在进程 A 被挂起的这段时间内，进程 B 完成了对文件的整个操作流程：打开，写入，关闭；进程 A 再继续往下执行，但进程 A 所操作的文件依旧是原来文件的就版本，此时进程 A 的操作结果将覆盖进程 B 的操作结果。

为了防止这种情况发生，需要用到文件锁或者信号量。

钥匙文件？

假如有多个不同的 HTML 可以调用同一个 CGI 程序，那么 CGI 程序如何区分它们呢？一个是通过隐含的 INPUT 标签。不过觉得这个比较麻烦，因为 CGI 必须经过一系列解码后才能找到这个隐含 INPUT 的变量和其值。

五．设置 HTTP 服务器以兼容 CGI

用 Perl 编写的 CGI 程序后缀为：.pl；Python 编写的 CGI 程序后缀为：.py；而 C 编写的 CGI 程序后缀为：.cgi，如果在 win 下编译出来的是.exe，最好将它重命名为.cgi。这些都是为了 HTTP 服务能够识别并调用它们。

当使用 appche httpd 服务器时，请编辑它的配置文件 httpd.conf 如下：

修改 AddHandler cgi-script 一句为 AddHandler cgi-script .cgi .py .pl

六 关于 CGI 的 C 语言库——cgihtml

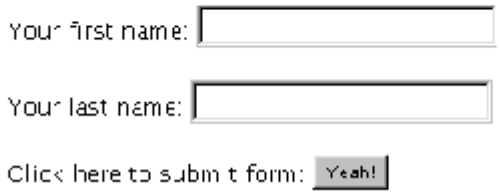
Cgihtml 是一个应用非常广泛的 C 语言编写的 CGI 库。它提供的功能函数如下：
Read_cgi_input()：获取并解析 HTML 表单输入，返回一个指向某结构体的指针
Cgi_val()：获取每个表单变量的值
Html_header():输出 HTML 标题栏
Html_begin():输出 HTML 文档的开始部分
H1():输出一行字符，字体为 H1
Html_end():输出 HTML 文档的结尾部分。
#include “cgi-lib.h”
#include “html-lib.h”
#include “string-lib.h”

六. 后话

有的人认为可以用 JavaScript 来代替 CGI 程序，这其实是一个概念上的错误。JavaScript 只能够在客户浏览器中运行，而 CGI 却是工作在服务器上的。他们所做的工作有一些交集，比如表单数据验证一类的，但是 JavaScript 是绝对无法取代 CGI 的。但可以这样说，如果一项工作即能够用 JavaScript 来做，又可以用 CGI 来做，那么绝对要使用 JavaScript，在执行的速度上，JavaScript 比 CGI 有着先天的优势。只有那些在客户端解决不了的问题，比如和某个远程数据库交互，这时就应该使用 CGI 了。

SSI:一种用来动态输出 HTML 文本的特殊程序。
网页里包含有某个变量，提交给服务器后，只有该变量改变。此时我们希望服务器不要把整个页面内容都发送过来，而只需要告诉客户端的浏览器，哪个变量的值便成什么样了，浏览器会自动更新。
SSI 在服务器端运行。
SSI 不需要外部接口，它不像 CGI 从标准输入接收信息。
你浏览你的 HTML 文档时看不到 SSI 标记，因为它已经被相应的程序输出所替代。
所有的 SSI 命令都是嵌入在普通的 HTML 注释行中的。当服务器无法解释 SSI 时，它将不解释并直接把文档传给浏览器，由于命令在注释中，故浏览器将忽略它们。而当服务器识别 SSI 时，它并不将该命令传给浏览器，相反，服务器将从上到下扫描 HTML 文档，执行每一个嵌入注释的命令，并将命令的执行结果代替原注释。
<! –注释文本-- >。服务器将根本不查看注释，除非已启动 SSI。
与纯注释不同的是，所有的 SSI 命令都是以#打头。
<! --#command tagname = “parameter”-- >,command 指出服务器做什么 tagname 指出参数类型，parameter 是该命令的用户定义值。
The current date is<! --#echo var = “DATE.LOCAL”-- >，服务器将向浏览器输出时间。

A typical HTML form




```
<form method="POST" action="http://host.com/cgi-bin/test.py">
   <p>Your first name: <input type="text" name="firstname">
   <p>Your last name: <input type="text" name="lastname">
   <p>Click here to submit form: <input type="submit" value="Yeah!">
   <input type="hidden" name="session" value="1f9a2">
</form>
```

A typical CGI script

```
#!/usr/local/bin/python
import cgi

def main():
    print "Content-type: text/html\n"
    form = cgi.FieldStorage()          # parse query
    if form.has_key("firstname") and form["firstname"].value != "":
        print "<h1>Hello", form["firstname"].value, "</h1>"
    else:
        print "<h1>Error! Please enter first name.</h1>"

main()
```

CGI script structure

- Check form fields
  - use cgi.FieldStorage class to parse query
    - takes care of decoding, handles GET and POST
    - "foo=ab+cd%21ef&bar=spam" -->
      {'foo': 'ab cd!ef', 'bar': 'spam'} # (well, actually, ...)
- Perform action
  - this is up to you!
  - database interfaces available
- Generate HTTP + HTML output
  - print statements are simplest
  - template solutions available