CGIC简明教程

CGIC简明教程

本系列的目的是演示如何使用C语言的CGI库“CGIC”完成Web开发的各种要求。

基础知识
* **1:** 使用**CGIC**的基本思路
* **2:** 获取**Get**请求字符串
* **3:** 反转义
* **4:** 获取请求中的参数值

进阶训练
* 用**CGIC**实现文件上传

CGIC简明教程**1**：使用**CGIC**的基本思路

C语言编程是一项复杂且容易出错的工作，所以在完成复杂任务时，一定要选择合适的库。对于用C语言编写CGI程序则更是如此。
CGIC是非常优秀的C语言CGI库函数。 其下载地址为：www.boutell.com/cgic/#obtain，现在的版本号是2.05。
本站从今天开始，将逐步介绍如何使用CGIC完成各种操作，也可以说是一个Tutorial。
（注：本系列涉及的编程环境都是Linux，Windows用户需要对用到的操作系统命令稍作修改）

本文纲要 ：
CGIC的安装、测试安装、使用CGIC的基本思路；
1) CGIC的下载安装

从上面提供的官方网址下载了CGIC库之后，解开压缩包，里面有大约10个文件，有用的是：
cgic.h：头文件；
cgic.c：CGIC的源代码文件；

cgictest.c：CGIC库的作者提供的一个CGI程序例子；
capture.c：用于调试CGI程序的工具；
Makefile：安装CGIC的脚本文件；
可以看到，整个库实际上就是cgic.c一个文件，可以说是非常的精炼。
我们可以把CGIC安装为操作系统的一个动态链接库，这样我们每次编译的时候，就不需要有cgic.c这个源文件了。
但是由于需要（以后将会看到），我们将修改cgic.c代码，所以我们不把它安装进系统。每次编译的时候，只要把cgic.c和cgic.h放到当前文件夹就好了。
2) 测试安装

在开始编写你自己的CGI程序之前，一定要先走通他的例子程序，免得后来程序出错的时候还不知道是配置有问题，还是你的程序代码有问题。
我们用他自带cgictest.c来实现自己的第一个C语言CGI程序。
你可以新建一个工作目录，用于存放你的CGI程序源代码，把cgic.h, cgic.c, cgictest.c三个文件拷贝到这个目录，然后建立一个Makefile文件，其内容为：

```
1. test.cgi:cgictest.c cgic.h cgic.c
2. gcc -wall cgictest.c cgic.c -o test.cgi
```

需要提醒的是，第二行开头一定是一个tab键（且仅有一个），不能使用空格。

保存好Makefile的内容之后，执行make命令：
make

我们看到，当前目录下应该多了一个test.cgi文件。

在你的网站根目录下建立一个cgi-bin目录（当然名字可以任意取，但作为习惯，一般叫做cgi-bin），然后在Apache的配置文件里赋予其执行CGI代码的权限，权限修改完之后要重启Apache。完成之后，把刚才生成的test.cgi放到cgi-bin目录中。此时我们可以在浏览器中输入以下地址进行访问：
http://127.0.0.1/cgi-bin/test.cgi

如果正常的话，应该看到一个网页被展示出来。这样，第一个C语言的CGI程序就运行起来了。如果浏览器报错，那么多半是配置Apache的时候有些操作没有正确完成。
3) 使用CGIC的基本思路

从cgic.c的代码可以看出，它定义了main函数，而在cgictest.c中定义了一个cgiMain函数。也就是说，对于使用CGIC编写的CGI程序，都是从cgic.c中的代码进入，在库函数完成了一系列必要的操作（比如解析参数、获取系统环境变量）之后，它才会调用你的代码（从你定义的cgiMain进入）。

另外一点就是，cgi程序输出HTML页面的方式都是使用printf把页面一行一行地打印出来，比如cgictest.c中的这一段代码：
fprintf(cgiOut, "<textarea NAME=/"address/" ROWS=4 COLS=40>/n");
fprintf(cgiOut, "Default contents go here. /n");
fprintf(cgiOut, "</textarea>/n");

上面这段代码的运行结果就是在页面上输出一个textarea。 第一个参数cgiOut实际上就是stdin，所以我们可以直接使用printf，而不必使用fprintf。不过在调试的时候会用到fprintf来重定向输出。
这种方式与Java Servlet非常类似，Servlet也是通过调用打印语句System.out.println(…)来输出一个页面。（不过后来Java推出了JSP来克服这种不便。）
但是与Servlet不同的地方在于，使用C语言的我们还要自己输出HTML头部（声明文档类型）：
cgiHeaderContentType("text/html");

这个语句的调用一定要在所有printf语句之前。而这个语句执行的任务实际上就是：
```
void cgiHeaderContentType(char *mimeType) {
   fprintf(cgiOut, "Content-type: %s/r/n/r/n", mimeType);
}
```

这个语句告诉浏览器，这次传来的数据是什么类型，是一个HTML文档，还是一个bin文件… 如果是个HTML文档，就通过浏览器窗口显示，如果是一个bin（二进制）文件，则打开下载窗口，让用户选择是否保存文件以及保存文件的路径。

理解了这几点之后，你就可以编写自己的CGIC程序了。新建一个文件test.c试试：
下载: test.c

```
1. #include <stdio.h>
2. #include "cgic.h"
3. #include <string.h>
4. #include <stdlib.h>
5. int cgiMain() {
6.    cgiHeaderContentType("text/html");
7.    fprintf(cgiOut, "<HTML><HEAD>/n");
8.    fprintf(cgiOut, "<TITLE>My First CGI</TITLE></HEAD>/n");
9.    fprintf(cgiOut, "<BODY><H1>Hello CGIC</H1></BODY>/n");
10.    fprintf(cgiOut, "</HTML>/n");
11.    return 0;
12. }
```

把Makefile文件中的cgictest.o都换成test.o，保存，再执行make命令即可。

把Makefile文件中的cgitest.c全部换称test.c，保存，再执行make命令即可。

此时通过浏览器访问，会在页面上看到一个大大的“Hello CGIC”。

CGIC简明教程**2**：获取**Get**请求字符串

Get请求就是我们在浏览器地址栏输入URL时发送请求的方式，或者我们在HTML中定义一个表单（form）时，把action属性设为“Get”时的工作方式；

Get请求字符串就是跟在URL后面以问号“?”开始的字符串，但不包括问号。比如这样的一个请求：
http://127.0.0.1/cgi-bin/out.cgi?ThisIsTheGetString

在上面这个URL中，“ThisIsTheGetString”就是Get请求字符串。

在进入我们自己编写的cgi代码之前，CGIC库已经事先把这个字符串取到了，我们可以在程序中直接获得，要做的仅仅是在你编写的cgiMain方法前面加入以下声明：
extern char *cgiQueryString;

现在给出一个简单的例子，这个例子跟上一篇的测试程序非常相似，只不过程序的输出是使用者输入的Get请求字符串。
下载: test.c

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

extern char *cgiQueryString;
int cgiMain() {
    cgiHeaderContentType("text/html");
    fprintf(cgiOut, "<HTML><HEAD>/n");
    fprintf(cgiOut, "<TITLE>My CGIC</TITLE></HEAD>/n");
    fprintf(cgiOut, "<BODY>");
    fprintf(cgiOut, "<H1>%s</H1>",cgiQueryString);
    fprintf(cgiOut, "</BODY>/n");
    fprintf(cgiOut, "</HTML>/n");
    return 0;
}
```

假设把这个程序编译成out.cgi（编译方法参见上一篇），并部署到Web服务器的cgi-bin目录下，当用户在浏览器地址栏输入本文开头给出的URL字符串时，浏览器页面上会显示：
ThisIsTheGetString

我们也可以编写一个用于测试的HTML页面：
下载: test.html

```
<html>
<head>
    <title>Test</title>
</head>
<body>
    <form action="cgi-bin/out.cgi" method="get">
        <input type="text" name="theText">
        <input type="submit" value="Continue &rarr;">
    </form>
</body>
</html>
```

文件的部署结构应该为：
|test.html
|—-cgi-bin/out.cgi

大家可以试试，通过浏览器访问http://127.0.0.1/test.html，在文本框内输入一些字符，并点击提交按钮，然后就可以看到cgi程序的执行结果：把在文本框输入的字符原样显示在浏览器上。

CGIC简明教程**3**：反转义

浏览器在发送Get请求时，会把请求字符串进行转义操作（英文术语为: escape）；比如，我们在地址栏输入（注意最后”it's me”中的空格）：
http://localhost/~Jack/cgi-bin/out.cgi?it's me

浏览器会把它转义为：
http://localhost/~Jack/cgi-bin/out.cgi?it's%20me

在上一篇最后给出的例子中，如果在文本框内输入
it's me

你会发现，浏览器最终发送的请求为
http://localhost/~Jack/cgi-bin/out.cgi?theText=it%27s+me

通过CGIC，我们可以把这些被转义后的字符还原为我们本来的输入，这个过程就叫“反转义”(Unescape)。
不过这个过程有点像hack他的代码。

整个过程分三个步骤：
1）打开cgic.c，找到这一行语句：
static cgiUnescapeResultType cgiUnescapeChars(char **sp, char *cp, int len);

注意，我们要找的只是这个函数声明，不是函数定义；

2）在这个函数声明语句的上方，你会看到一个结构体定义：

```
typedef enum {
    cgiUnescapeSuccess,
    cgiUnescapeMemory
} cgiUnescapeResultType;
```

把这几行语句复制到cgic.h文件中，并在这里把它注释掉；
同时还要删除在第一步中找到的函数声明语句中的“static”关键字。

3）我们现在就可以使用反转义函数cgiUnescapeChars了：
在你自己的代码（按照惯例，还是test.c）中，加入以下声明语句即可
extern cgiUnescapeResultType cgiUnescapeChars(char **sp, char *cp, int len);

接下来我们给出一段完整的test.c代码
下载: test.c

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

extern char *cgiQueryString;
extern cgiUnescapeResultType cgiUnescapeChars(char **sp, char *cp, int len);
```

```
8. int cgiMain() {
9.     char * buffer;

10.     cgiHeaderContentType("text/html");
11.     fprintf(cgiOut, "<HTML><HEAD>/n");
12.     fprintf(cgiOut, "<TITLE>My CGI</TITLE></HEAD>/n");
13.     fprintf(cgiOut, "<BODY>");
14.     cgiUnescapeChars(&buffer, cgiQueryString, strlen(cgiQueryString));
15.     fprintf(cgiOut, "<H1>%s</H1>",buffer);
16.     fprintf(cgiOut, "</BODY>/n");
17.     fprintf(cgiOut, "</HTML>/n");
18.     free(buffer);
19.     return 0;
20. }
```

值得注意的是，buffer的存储空间是cgiUnescapeChars帮你分配的，但最后要由你自己来释放（free），这一点千万不可忘记。

下面你可以结合上一篇给出的测试用html代码试试该cgi程序的运行结果，也可以直接在浏览器地址栏输入一些带有特殊符号的字符串。

最后讲一下为什么不得不用这种hacker的方式来完成该任务，而CGIC不显式提供？
CGIC的出发点是，我们平时只需要解析请求中的键值对，比如：”?q=nice&client=IE”，当我们在服务端查询“q”的值时，我们就能得到“nice”。CGIC有一族函数帮助我们完成这个任务，比如cgiFormString（以后会讲到）。在解析这种请求格式的时候，如果我们提供的参数值含有被转义的字符，那么CGIC就会在内部调用cgiUnescapeChars完成反转义。
但是，有时候我们会发送非常复杂的Get请求字符串，但并不是“键－值”对的格式。这就需要直接使用cgiUnescapeChars进行反转义了。
例如：假设我们有个服务端cgi程序chat.cgi，这是个网络聊天机器人（也许你可以开发自己的Web版MSN机器人、QQ机器人）。如果我们发送如下请求：
http://127.0.0.1/cgi-bin/chat.cgi?"this is a cgi user"

那么chat.cgi就会把“this is a cgi user”当做你对它说的话，经过处理，它会回复一段语句。为了方便，我们并没有写成“键－值”对的形式。这个时候被我们hack的cgiUnescapeChars就能派上用场了。

CGIC简明教程**4**：获取请求中的参数值

我们在提交一个表单(form)时，怎样把表单内的值提取出来呢？
比如下面这个表单：

```
<form action="cgi-bin/out.cgi" method="POST">
   <input type="text" name="name" />

   <input type="text" name="number" />
   <input type="submit" value="Submit" />
</form>
```

当out.cgi收到请求时，需要把输入框”name”和输入框”number”内的值提取出来。而且不管form中的action是GET还是POST，都要有效。

下面给出示例代码：
下载: test.c

```
1. #include <stdio.h>
2. #include "cgic.h"
3. #include <string.h>
4. #include <stdlib.h>
5.
6. int cgiMain() {
7.     char name[241];
```

```
8.  char number[241];
9.  cgiHeaderContentType("text/html");
10.  fprintf(cgiOut, "<HTML><HEAD>/n");
11.  fprintf(cgiOut, "<TITLE>My CGI</TITLE></HEAD>/n");
12.  fprintf(cgiOut, "<BODY>");
13.  cgiFormString("name", name, 241);
14.  cgiFormString("number", number, 241);
15.  fprintf(cgiOut, "<H1>%s</H1>",name);
16.  fprintf(cgiOut, "<H1>%s</H1>",number);
17.  fprintf(cgiOut, "</BODY>/n");
18.  fprintf(cgiOut, "</HTML>/n");
19.  return 0;
20. }
```

从上面的代码可以看出，第13行和第14行获取了输入框的值。

获取输入参数值在CGIC中其实有一族函数，cgiFormString是其中最常用的一个。
cgiFormStringNoNewlines用来去掉换行符（如果用户是在一个TextArea里输入字符的话）；
cgiFormStringSpaceNeeded用于测试输入值的长度，可以以此为依据，然后按需精确分配缓冲区。

用**C**语言库**(CGIC)**编写**CGI**，实现文件上传

用C语言编写cgi程序的话，多半会用到CGIC。 这是个非常流行的库，遇到文件上传之类的应用更是离不开它。官方页面及下载地址为：www.boutell.com/cgic/#obtain

不少网站都有文件上传的功能，本文展示如何用CGIC库编写文件上传的服务端程序，最后给出一段简单的HTML代码，供大家测试使用 。
下载: upload.c

```
//upload.c
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include"cgic.h"
#define BufferLen 1024
int cgiMain(void){
   cgiFilePtr file;
   int    targetFile;
   mode_t    mode;
   char name[128];
   char fileNameOnServer[64];
   char contentType[1024];
   char buffer[BufferLen];
   char *tmpStr=NULL;
   int size;
   int got,t;
   cgiHeaderContentType("text/html");
   //取得html页面中file元素的值，应该是文件在客户机上的路径名
   if (cgiFormFileName("file", name, sizeof(name)) !=cgiFormSuccess) {
      fprintf(stderr,"could not retrieve filename\n");
      goto FAIL;
   }
   cgiFormFileSize("file", &size);
   //取得文件类型，不过本例中并未使用
```

```
    cgiFormFileContentType("file", contentType, sizeof(contentType));
    //目前文件存在于系统临时文件夹中，通常为/tmp，通过该命令打开临时文件。临时文件的名字与用户文件的名字不同，所以不能通过路径/tmp/userfilename的方式获得文件
    if (cgiFormFileOpen("file", &file) != cgiFormSuccess) {

        fprintf(stderr,"could not open the file\n");
        goto FAIL;
    }
    t=-1;
    //从路径名解析出用户文件名
    while(1){
        tmpStr=strstr(name+t+1,"\\");
        if(NULL==tmpStr)
            tmpStr=strstr(name+t+1,"/");//if "\\" is not path separator, try "/"
        if(NULL!=tmpStr)
            t=(int)(tmpStr-name);
        else
            break;
    }
    strcpy(fileNameOnServer,name+t+1);
    mode=S_IRWXU|S_IRGRP|S_IROTH;
    //在当前目录下建立新的文件，第一个参数实际上是路径名，此处的含义是在cgi程序所在的目录（当前目录））建立新文件
    targetFile=open(fileNameOnServer,O_RDWR|O_CREAT|O_TRUNC|O_APPEND,mode);
    if(targetFile<0){
        fprintf(stderr,"could not create the new file,%s\n",fileNameOnServer);
        goto    FAIL;
    }
    //从系统临时文件中读出文件内容，并放到刚创建的目标文件中
    while (cgiFormFileRead(file, buffer, BufferLen, &got) ==cgiFormSuccess){
        if(got>0)
            write(targetFile,buffer,got);
    }
    cgiFormFileClose(file);
    close(targetFile);
    goto    END;
FAIL:
    fprintf(stderr,"Failed to upload");
    return 1;
END:
    printf("File \"%s\" has been uploaded",fileNameOnServer);
    return 0;
}
```

假设该文件存储为upload.c，则使用如下命令编辑：

```
gcc -Wall upload.c cgic.c -o upload.cgi
```

编译完成后把upload.cgi复制到你部署cgi程序的目录（通常命名为cgi-bin）。
正式部署时，请务必修改用open创建新文件那一行代码。把open的第一个参数设置为目标文件在服务器上存储的绝对路径，或者相对于cgi程序的相对路径。本例中，出于简单考虑，在cgi程序所在目录下创建新文件。

测试用HTML代码:
下载: upload.html

```
1. <form target="_blank" method="post" action="cgi-bin/upload.cgi">
2.     <input name="file" type="file" /> <input name="submit" type="submit" />
3. </form>
```

最后的文件目录结构为

/MyWebRoot
├—/upload.html
├—/cgi-bin
├——/upload.cgi
当然，你必须配置能够cgi-bin，并且程序要有权限在cgi-bin目录下创建文件（因为此例把文件上传到cgi-bin目录下）。

那么如何控制上传文件的大小呢？因为你有时会不允许用户上传太大的文件。
通过分析cgic.c的源代码，我们发现它定义了一个变量cgiContentLength，表示请求的长度。但我们需要首先判断这是一个上传文件的请求，然后才能根据cgiContentLength来检查用户是否要上传一个太大的文件。
cgic.c的main函数中进行了一系列if-else判断来检查请求的类型，首先确定这是一个post请求，然后确定数据的编码方式为 “multipart/form-data”，这个判断通过之后，就要开始准备接收数据了。所以我们要在接收数据开始之前使用 cgiContentLength判断大小，如果超过标准，就立即返回，不允许继续操作。
下面贴出修改后代码片段（直接修改cgic.c的源代码即可）：

```
else if (cgiStrEqNc(cgiContentType, "multipart/form-data")) {
#ifdef CGICDEBUG
        CGICDEBUGSTART
        fprintf(dout, "Calling PostMultipartInput\n");
        CGICDEBUGEND
#endif /* CGICDEBUG */
        //我的代码
        //UpSize:文件长度上限值，以byte为单位，UpSize是一个int变量，因为cgiContentLength的类型为int
        if(cgiContentLength>UpSize){
            cgiHeaderContentType("text/html");

            printf("File too large!\n");
            cgiFreeResources();
            return -1;
        }
        //我的代码结束
        if (cgiParsePostMultipartInput() != cgiParseSuccess) {
#ifdef CGICDEBUG
            CGICDEBUGSTART
            fprintf(dout, "PostMultipartInput failed\n");
            CGICDEBUGEND
#endif /* CGICDEBUG */
            cgiFreeResources();
            return -1;
        }
#ifdef CGICDEBUG
        CGICDEBUGSTART
        fprintf(dout, "PostMultipartInput succeeded\n");
        CGICDEBUGEND
#endif /* CGICDEBUG */
    }
  }
```

变量UpSize表示文件大小的上限。在cgic.c的main中找到相关代码，并修改成上面这样即可。你可以在cgic.c中定义UpSize，也可以在刚才完成的upload.c中定义，然后在cgic.c中用extern方式引用。

上一篇：CGIC库的移植
下一篇：C语言CGI上传文件

联动天下虚拟主机基础知识
硬笔书法简明教程
吉他入门简明教程
数据库管理系统(简明教程)

更多

您可能会喜欢

明


教你辨明传统本/轻薄本/上网本的区别


唐代摩尼教传播过程辨析


教你养肝明目操 还你明亮双眼


明 文徵明尺牍

无觅

发表评论：

您好，请 登录 或者 注册 后再进行评论

使用合作网站登录： 新浪微博 QQ 人人