TinyHttpd解析

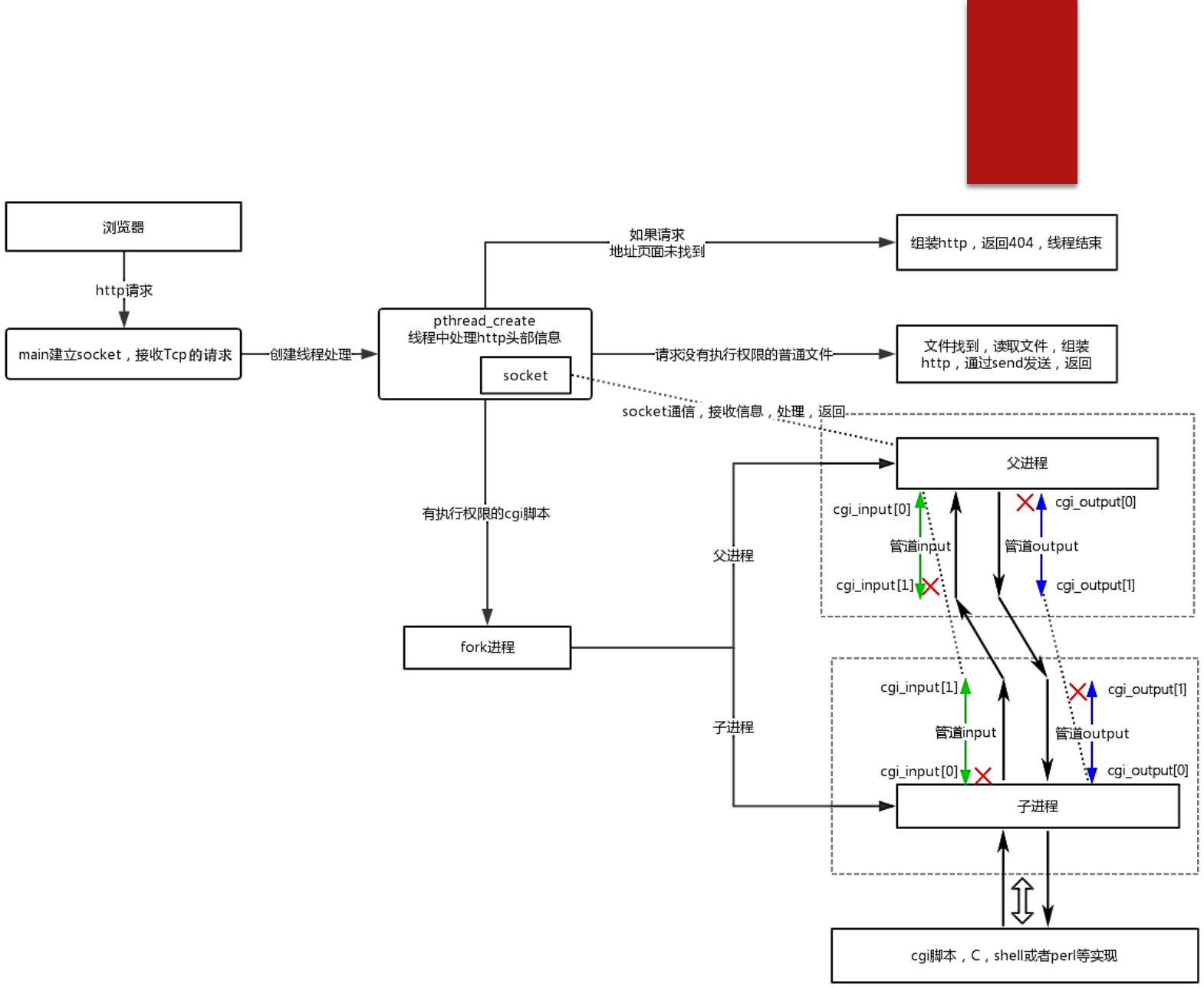
1.什么是Tinyhttpd?

- Tinyhttpd是由J. DavidBlackstone在1999年编写的一个超轻量型Http Server，使用C语言开发，全部代码只有502行(包括注释)，附带一个简单的Client，可以通过阅读这段代码理解一个Http Server的本质。

写在正式开始之前：

- 由于本程序是1999年在solaris上实现的，所以不能直接在Linux环境下编译运行，需要修改一些地方：
- 1. 第33行改为void *accept_request(void *);
  - 所以下面函数的实现也要修改下：
    - void *accept_request(void* tclient) {
    - int client = *(int *)tclient
- 2. 438行和483行的变量类型从int改为socklen_t
- 3. 497行改为if (pthread_create(&newthread , NULL, accept_request, (void*)&client_sock) != 0);
- 4. Makefile中编译的一行改为gcc -W -Wall -o httpd httpd.c -lpthread

先用一张图了解Tinyhttp是如何运作的：



预备知识——HTTP协议

什么是HTTP？

- HTTP协议（HyperText Transfer Protocol，超文本传输协议）是因特网上应用最为广泛的一种网络传输协议，所有的WWW文件都必须遵守这个标准。
- HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP的工作原理：

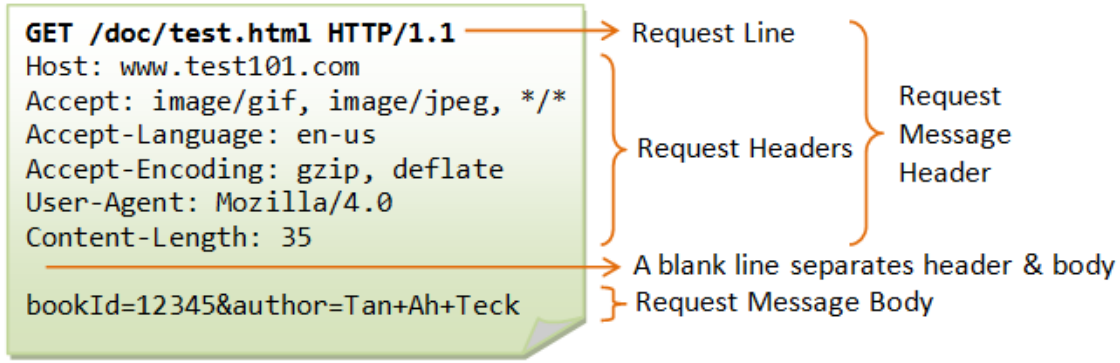
- HTTP协议工作于客户端-服务端架构上。浏览器作为HTTP客户端通过URL向HTTP服务端（即WEB服务器）发送所有请求。
- Web服务器根据接收到的请求后，向客户端发送响应信息。
- 常见的Web服务器有Apache服务器，以及这次我们要讲的Tinyhttpd等。

客户端和服务器端交互的过程：

- 1. 客户发起连接
- 2. 客户发送请求
- 3. 服务器响应请求
- 4. 服务器关闭连接

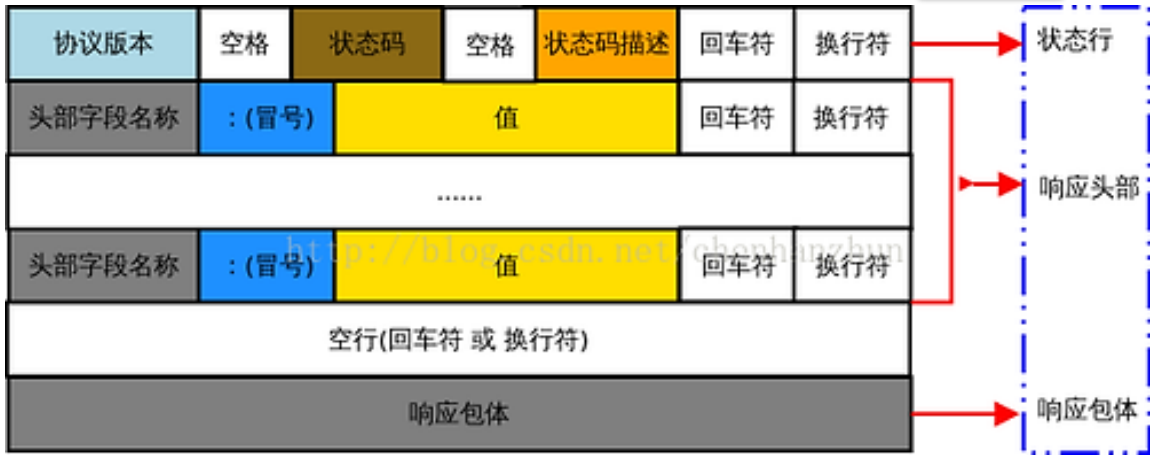
HTTP的请求消息：

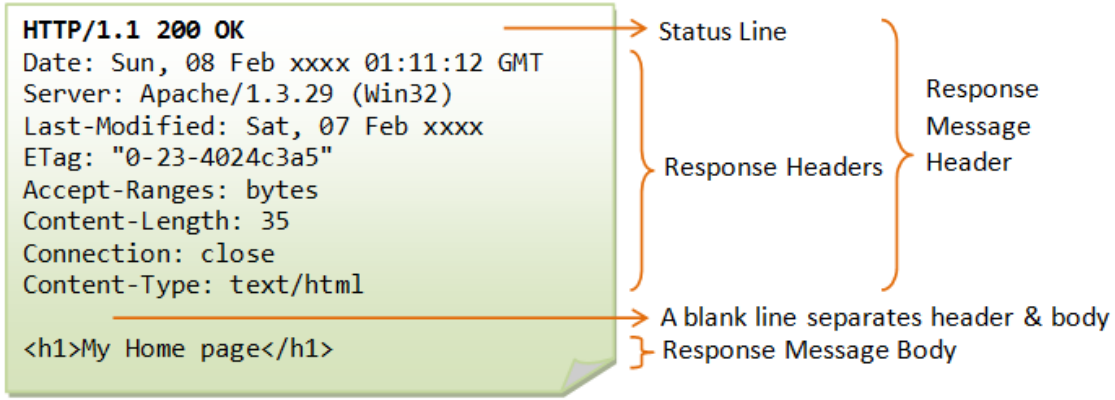
- 客户端发送一个HTTP请求到服务器的请求消息包括以下格式：
- 请求行（request line）
- 请求头部（header）
- 空行
- 请求数据



HTTP的响应信息：

- 在接收和解释请求消息后，服务器返回一个HTTP响应消息。HTTP响应也由四个部分组成：
- 状态行（Response Line）
- 消息报头
- 空行
- 响应正文





HTTP的请求方法：

- 根据HTTP标准，HTTP请求可以使用多种请求方法。
- HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。
- HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法
- 这里只详细介绍Tinyhttpd所支持的两种请求方法：
- GET： 请求URI所标识的页面信息，并返回实体主体。
- POST：向URI所标识的资源提交数据，进行处理请求（例如提交表单或者上传文件）。
- GET 方法通过请求URI得到资源。一般用于获取/查询资源信息。
- POST 方法用于向服务器提交新的内容。一般用于更新资源信息。

什么是URL？

- URL是Internet上用来描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上，采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL一般由三部组成：
  ①协议(或称为服务方式)
  ②存有该资源的主机IP地址(有时也包括端口号)
  ③主机资源的具体地址。如目录和文件名等

URL实例：

http://www.aspxfans.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name

- 1.协议部分：该URL的协议部分为“http：”，这代表网页使用的是HTTP协议。
- 2.域名部分：该URL的域名部分为“www.aspxfans.com”。
- 3.端口部分：该URL的端口为8080。
- 4.虚拟目录部分：从域名后的第一个“/”开始到最后一个“/”为止。该URL的虚拟目录部分为/news。
- 5.文件名部分：从域名后的最后一个“/”开始到“？”为止。该URL的文件名部分为/index.asp。
- 6.锚部分：从“#”开始到最后，都是锚部分。
- 7.参数部分：从“？”开始到“#”为止之间的部分为参数部分，又称搜索部分、查询部分。

HTTP的响应头信息：

- HTTP头部提供了关于请求，响应或者其他的发送实体的信息。
- HTTP请求头用于说明是谁在发送请求、请求源于何处等信息。但是在Tinyhttpd中，请求头中的信息被程序所忽略，这里不做介绍。
- HTTP响应头用于向客户端提供一些额外信息，比如谁在发送响应、响应者的功能等。这里只介绍Tinyhttpd中用到的两种响应头信息：
- Content-Length: 服务器通过这个头告诉浏览器，回送的数据的长度。
- Content-Type: 服务器通过这个头告诉浏览器，回送数据的类型。

HTTP状态码：

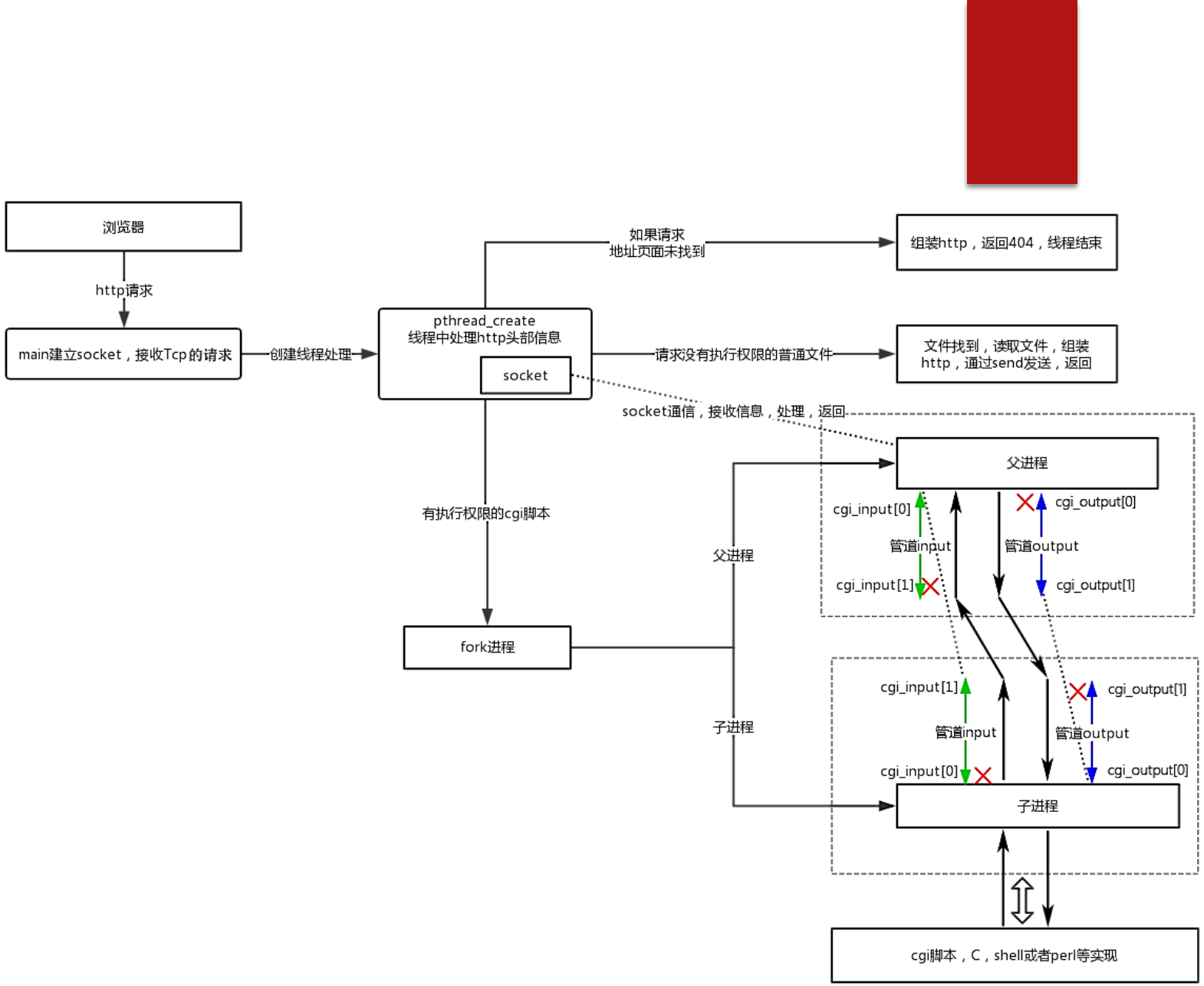
- 当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。
- 下面是Tinyhttpd中出现的HTTP状态码：
- 200 - 请求成功
- 400 -客户端请求的语法错误，服务器无法理解
- 404 - 请求的资源（网页等）不存在
- 500 - 服务器内部错误，无法完成请求
- 501 - 服务器不支持请求的功能，无法完成请求

正文——Tinyhttpd中代码的理解

主要函数简略说明：

- main　　主函数
- startup　　绑定监听套接字
- accept_request　　每次收到请求，创建一个线程来处理接受到的请求
- serve_file　　接读取文件返回给请求的http客户端
- execute_cgi　　执行cgi文件
- 建议源码阅读顺序： main -> startup -> accept_request -> serve_file -> execute_cgi，通晓主要工作流程后再仔细把每个函数的源码看一看。

重新体会一次Tinyhttp是如何运作的：



Tinyhttpd的工作流程（1）：

- （1） 服务器启动，在指定端口或随机选取端口绑定 httpd 服务。
- （2）收到一个 HTTP 请求时（其实就是 listen 的端口 accpet 的时候），派生一个线程运行 accept_request 函数。

Tinyhttpd的工作流程（2）：

- （3）取出 HTTP 请求中的 method (GET 或 POST) 和 url,。对于 GET 方法，如果有携带参数，则 query_string 指针指向 url 中 ？ 后面的 GET 参数。

- （4） 格式化 url 到 path 数组，表示浏览器请求的服务器文件路径，在 tinyhttpd 中服务器文件是在 htdocs 文件夹下。当 url 以 / 结尾，或 url 是个目录，则默认在 path 中加上 index.html，表示访问主页。

- （5）如果文件路径合法，对于无参数的 GET 请求，直接输出服务器文件到浏览器，跳到最后一步。其他情况（带参数 GET，POST 方式，url 为可执行文件），则调用 excute_cgi 函数执行 cgi 脚本。

Tinyhttpd的工作流程（3）：

- （6）如果是 GRT，读取整个 HTTP 请求并丢弃，如果是 POST 则找出 Content-Length. 把 HTTP 200 状态码写到套接字。

- （7） 建立两个管道 cgi_input 和 cgi_output，目的是实现父子进程间的通信，并 fork 一个子进程来执行 CGI 程序。

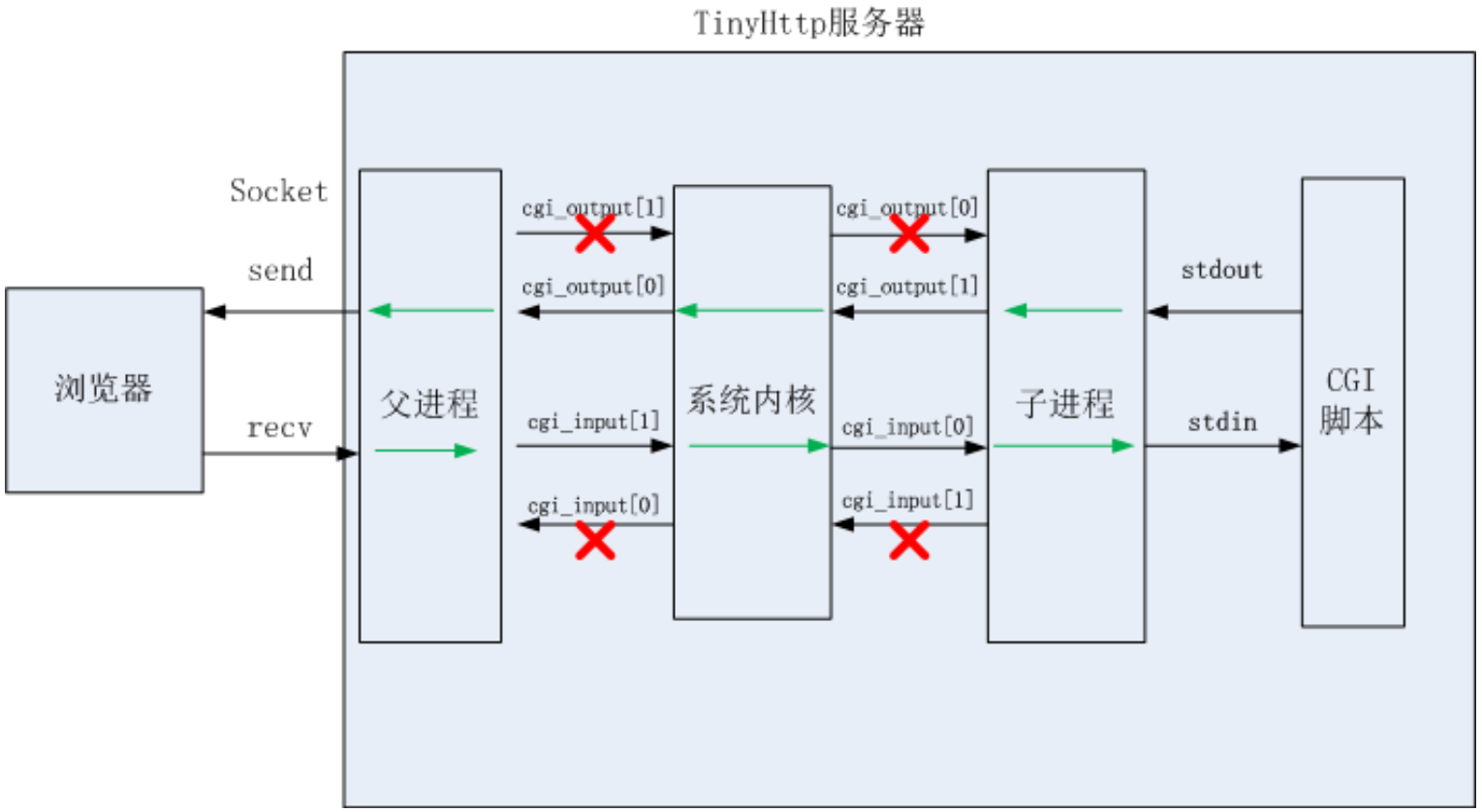
Tinyhttpd的工作流程（4）：

- （8） 在子进程中，把 stdout 重定向到 cgi_outputt 的写入端，把 stdin 重定向到 cgi_input 的读取端，关闭 cgi_input 的写入端 和 cgi_output 的读取端，设置 request_method 的环境变量，GET 的话设置 query_string 的环境变量，POST 的话设置 content_length 的环境变量，这些环境变量都是为了给 cgi 脚本调用，接着用 execl 运行 cgi 程序。

- （9） 在父进程中，关闭 cgi_input 的读取端 和 cgi_output 的写入端，如果 POST 的话，把 POST 数据写入 cgi_input，已被重定向到 stdin ，读取 cgi_output 的管道输出到客户端，该管道输入是 stdout。接着关闭所有管道，等待子进程结束。

- 这一部分比较复杂，详见后一页的图示。

图示：


TinyHttp服务器
Socket
send
recv
浏览器
父进程
cgi_output[1]
cgi_output[0]
cgi_input[1]
cgi_input[0]
系统内核
cgi_output[0]
cgi_output[1]
cgi_input[0]
cgi_input[1]
子进程
stdout
stdin
CGI
脚本

Tinyhttpd的工作流程（5）：

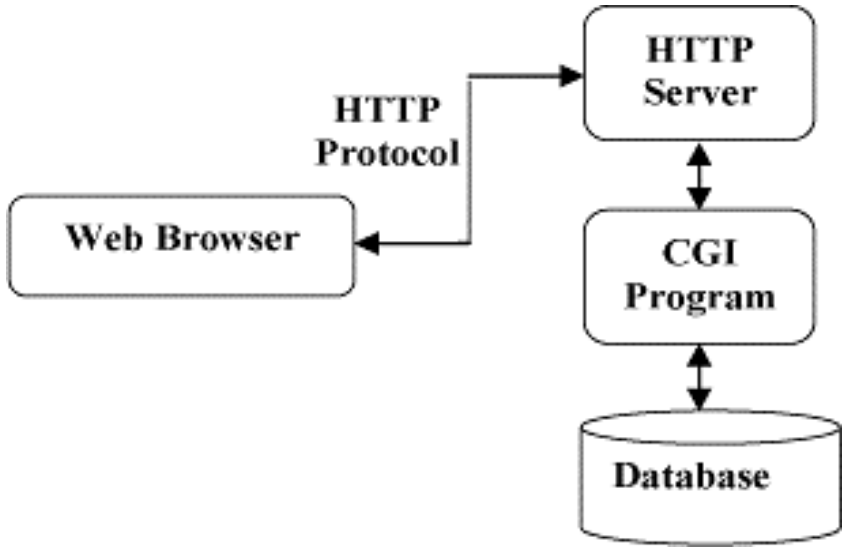
- （10） 关闭与浏览器的连接，完成了一次 HTTP 请求与回应，因为 HTTP 是无连接的。

拓展内容—— CGI

什么是CGI?

- 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右图：



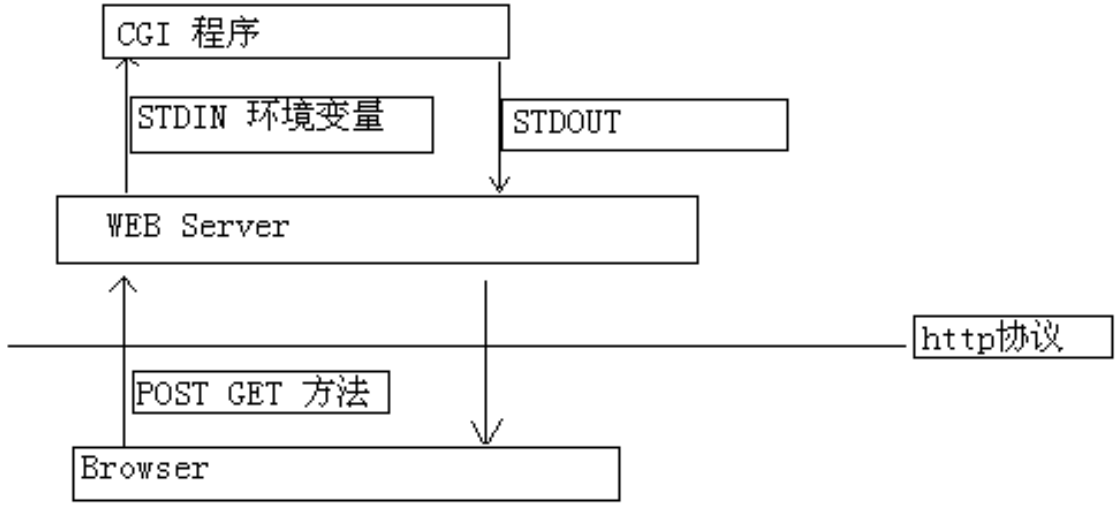
什么是CGI程序？

- CGI程序就是真正的被服务器（如Apache）调用的来处理用户发送过来的数据的程序。所谓CGI程序就是按照CGI接口规范编写的能够处理用户通过浏览器发送到服务器的数据的一个程序。

WEB服务器与CGI程序交互：

WEB服务器将根据CGI程序的类型决定数据向CGI程序的传送方式，一般来讲是通过标准输入/输出流和环境变量来与CGI程序间传递数据。 如右图所示：

对一个 CGI 程序，做的工作其实只有：从环境变量(environment variables)和标准输入(standard input)中读取数据、处理数据、向标准输出(standard output)输出数据。



CGI结构示意图

常见的环境变量：

REQUEST_METHOD	*前端页面数据请求方式：get/post*
QUERY_STRING	采用GET时所传输的信息
CONTENT_LENGTH	STDIO中的有效信息长度
SCRIPT_NAME	所调用的CGI程序的名字
SERVER_NAME	*服务器的IP或名字*
SERVER_PORT	主机的端口号

CGI的优点和不足之处：

- CGI的主要优点是它的简单性，语言无关性，Web服务器无关性以及广泛的可接受性。
- CGI是一种标准，并不限定语言。所以Java、PHP、Python都可以通过这种方式来生成动态网页。但是实际上这些动态语言却很少这样用。原因是CGI有一大硬伤。那就是每次CGI请求，服务器都要启动一个进程去执行这个CGI程序，这是一种颇具Unix特色的fork-and-execute。当用户请求量大的时候，这个fork-and-execute的操作会严重拖慢Server的进程。所以CGI在今天已经逐渐被FastCGI 等所取代。

用C语言编写一个简单的CGI程序：

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

int main() {
    char *data;
    data = getenv("QUERY_STRING");

    puts("Content-Type: text/html\r\n\r\n");
    puts(data);
    printf("Hello cgi!");
    return 0;
}
```