建议源码阅读顺序：

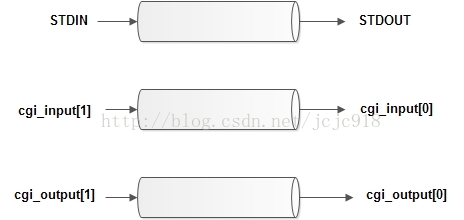
main -> startup -> accept\_request -> execute\_cgi

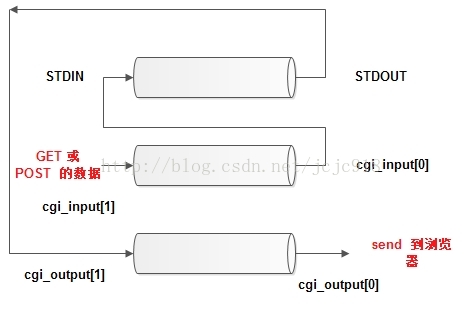
通晓主要工作流程后再仔细把每个函数的源码看一看。

**工作流程**

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。
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，即用 HTTP 格式写到套接字上，跳到（10）。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。
6. 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, 并 fork 一个进程。
8. 在子进程中，把 STDOUT 重定向到 cgi\_output 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。这一部分比较乱，见下图说明：





（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。