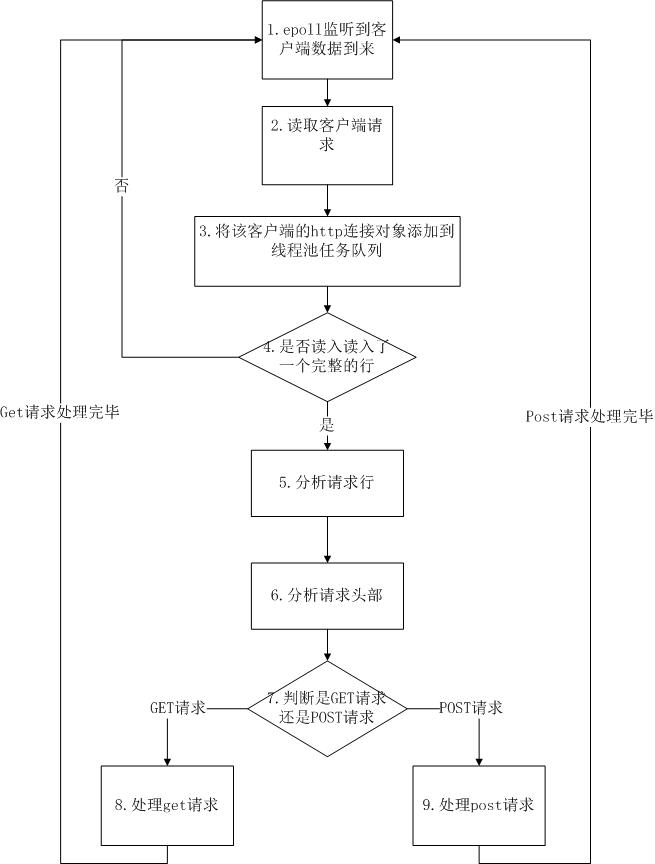
**客户端请求到来之后处理流程图**



**注解:**

1. Epoll监听到客户端数据到来

即客户端描述符就绪。

1. 读取客户端请求

调用http\_conn::read\_request()方法来读取客户端请求到m\_read\_buf缓冲区中

1. 将客户端连接的http\_conn对象添加到线程池任务队列

读取了客户端的请求后，将该http\_conn对象添加到线程池任务队列中，工作线程获取到任务对象后，然后开始调用任务对象的process()方法，这个借口是线程池规定任务类必须实现的，线程池的工作线程将调用它来执行任务。

1. 判断是否读入了一个完整的行

当前已进入http\_conn对象内的处理流程：

开始解析已读到的请求数据，判断是否读到一个完整的行，包括请求行和请求头部，以及空行。 如果没有读到，说明数据没有读完，则继续去监听客户端数据转到步骤1 。如果读到了一个完整的行，则继续分析。

1. 分析请求行

分析请求行，获取请求方法，URL，协议版本

1. 分析请求头部

分析请求头部，获取头部字段值，包括Connection、host以及Content-length等等。

1. 判断是GET请求还是POST请求

通过分析请求行，获取到的请求方法来判断是GET请求还是POST请求

1. 处理GET请求
2. 根据URL获取到请求文件名，然后根据服务器文件存储目录构造出真实路径。
3. 对文件的属性进行分析，获取其大小。
4. 将文件映射到内存空间，并记录映射地址。
5. 然后开始制作GET请求响应头部
6. 响应头部制作完毕后注册客户端套接字的可写事件EPOLLOUT，此时任务线程的任务就结束了。而真正的向客户端发送请求交给监听线程来完成
7. 处理POST请求

（1） 父进程创建两个管道：

fa\_To\_ch(父进程向子进程传递参数)

ch\_To\_fa(子进程向父进程传递结果)

（2） 创建子进程，关闭子进程对fa\_To\_ch管道的写端，关闭ch\_To\_fa的读端。

（3）将标准输入STDIN\_FILENO重定向到fa\_To\_ch管道的读端，这样子进程读标准输入即可获取到fa\_To\_ch中父进程传递给子进程的数据。

（4） 将标准输出STDOUY\_FILENO重定向到ch\_TO\_fa管道的写端，这样父进程就可以在ch\_To\_fa的读端获取到子进程的处理结果。

1. 用CGI程序替换进程空间，通过环境变量将POST请求方法和参数长度传递给CGI进程
2. 父进程通过读取ch\_To\_fa管道获取到CGI进程的处理结果，并转发给客户端套接字。