目录

[1. 从输入URL到页面加载发生了什么 2](#_Toc476735817)

[1.1 DNS解析 2](#_Toc476735818)

[1.2 TCP连接 2](#_Toc476735819)

[1.3 发送HTTP请求 2](#_Toc476735820)

[1.4 服务器处理请求并返回HTTP报文 2](#_Toc476735821)

[1.5 浏览器解析渲染页面 3](#_Toc476735822)

[1.6 连接结束 4](#_Toc476735823)

[2. dns劫持 4](#_Toc476735824)

[3. url 重定向 6](#_Toc476735825)

[3.1. 前端实现 6](#_Toc476735826)

[3.1.1 html 页面跳转方式 6](#_Toc476735827)

[3.1.2 JS 页面跳转方式 6](#_Toc476735828)

[3.2. 后端实现 7](#_Toc476735829)

[3.2.1 http消息 7](#_Toc476735830)

[3.2.2 http消息源码分析 8](#_Toc476735831)

[附录 12](#_Toc476735832)

[1 几个重要的结构体 12](#_Toc476735833)

[2 配置参数读入 14](#_Toc476735834)

# 1. 从输入URL到页面加载发生了什么

总体来说分为以下几个过程:

## 1.1 DNS解析

## 1.2 TCP连接

## 1.3 发送HTTP请求

它主要发生在客户端。发送HTTP请求的过程就是构建HTTP请求报文并通过TCP协议中发送到服务器指定端口(HTTP协议80/8080, HTTPS协议443)。HTTP请求报文是由三部分组成: 请求行, 请求报头和请求正文。

## 1.4 服务器处理请求并返回HTTP报文

后端从在固定的端口接收到TCP报文开始，这一部分对应于编程语言中的socket。它会对TCP连接进行处理，对HTTP协议进行解析，并按照报文格式进一步封装成HTTP Request对象，供上层使用。这一部分工作一般是由Web服务器去进行，HTTP响应报文也是由三部分组成: 状态码, 响应报头和响应报文。

HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

1\*\* 信息，服务器收到请求，需要请求者继续执行操作

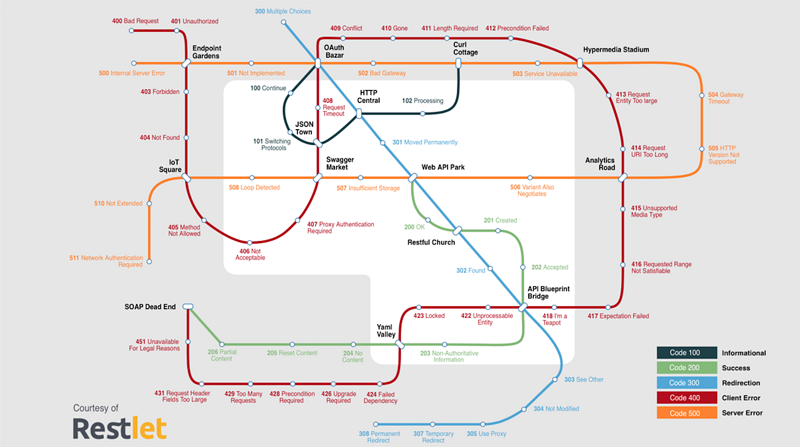
2\*\* 成功，操作被成功接收并处理

3\*\* 重定向，需要进一步的操作以完成请求

4\*\* 客户端错误，请求包含语法错误或无法完成请求

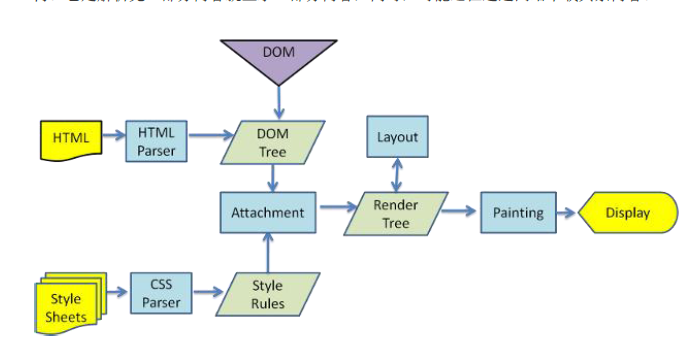
5\*\* 服务器错误，服务器在处理请求的过程中发生了错误

状态码查询：<http://www.runoob.com/http/http-status-codes.html>

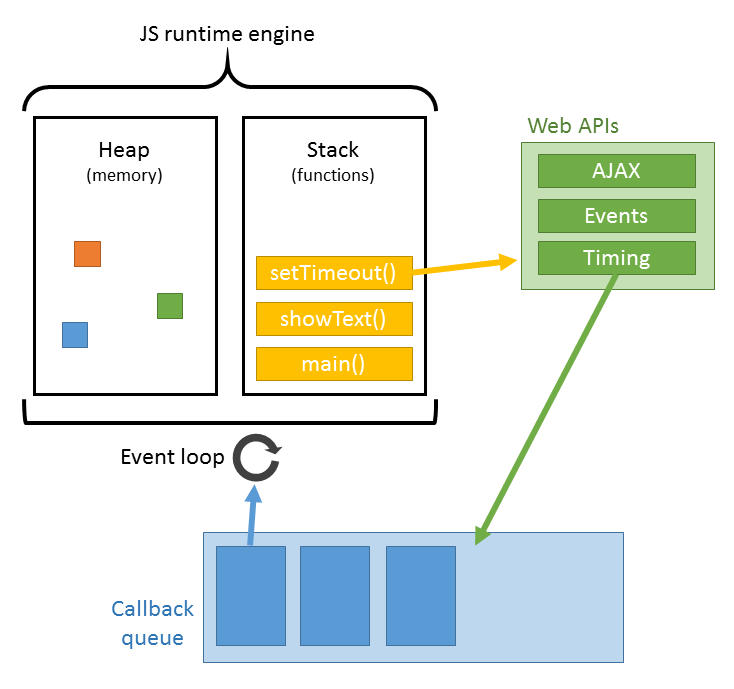


## 1.5 浏览器解析渲染页面

浏览器在收到HTML,CSS,JS文件后，它是如何把页面呈现到屏幕上的？下图对应的就是WebKit（一个开源的浏览器引擎）渲染的过程。



浏览器是一个边解析边渲染的过程。首先浏览器解析HTML文件构建DOM树，然后解析CSS文件构建渲染树，等到渲染树构建完成后，浏览器开始布局渲染树并将其绘制到屏幕上。



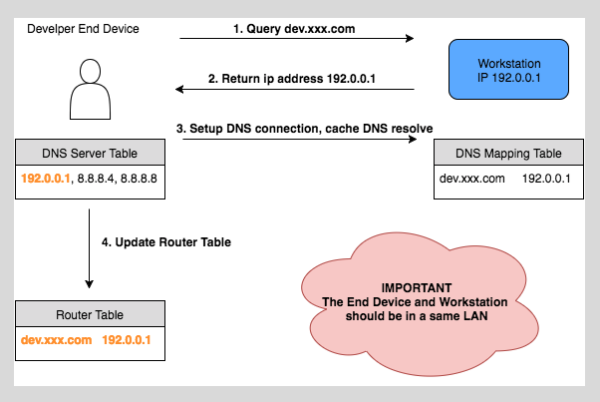
JS的解析是由浏览器中的JS解析引擎完成的。JS的执行机制就可以看做是一个主线程加上一个任务队列(task queue)。同步任务就是放在主线程上执行的任务，异步任务是放在任务队列中的任务。所有的同步任务在主线程上执行，形成一个执行栈;异步任务有了运行结果就会在任务队列中放置一个事件；脚本运行时先依次运行执行栈，然后会从任务队列里提取事件，运行任务队列中的任务，这个过程是不断重复的，所以又叫做事件循环(Event loop)。但是当文档加载过程中遇到JS文件，HTML文档会挂起渲染过程，不仅要等到文档中JS文件加载完毕还要等待解析执行完毕，才会继续HTML的渲染过程。原因是因为JS有可能修改DOM结构，这就意味着JS执行完成前，后续所有资源的下载是没有必要的。

## 1.6 连接结束

# 2. dns劫持

当用户输入一个URL时，想要能够访问我们的路由器管理界面首先就需要将改URL的DNS解析到路由器的web服务器地址上，这时候，我们需要dns劫持。Dns劫持中主要用到一个开源的软件-dnsmasq。

首先我们利用dnsmasq将自己的工作站配置为一个能够解析开发域名的server，解析的ip地址设置为工作站的ip地址。利用dnsmasq建立了一个dns mapping table，将[www.baidu.com](http://www.baidu.com)的域名解析为路由器管理界面地址192.168.2.1。 这时候，通过[www.baidu.com](http://www.baidu.com)访问时会转跳至192.168.2.1。

****

# 3. url 重定向

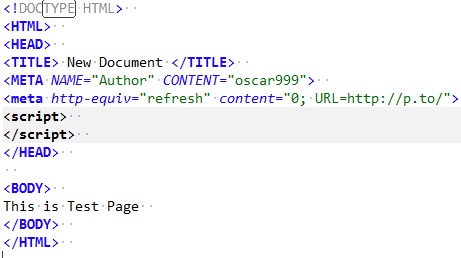
这时候我们已经可以成功的通过域名[www.baidu.com访问到路由器的web](http://www.baidu.com访问到路由器的web)服务器。

url重定向的实现可以在前端实现或是后端实现。这时候web服务器需要将指定的html返回给客户端，比如我们的快速向导页面或是首页页面。这就需要重新定向用户输入的url。

## 3.1. 前端实现

### 3.1.1 html 页面跳转方式

可以使用html 的meta 标签实现页面的跳转。



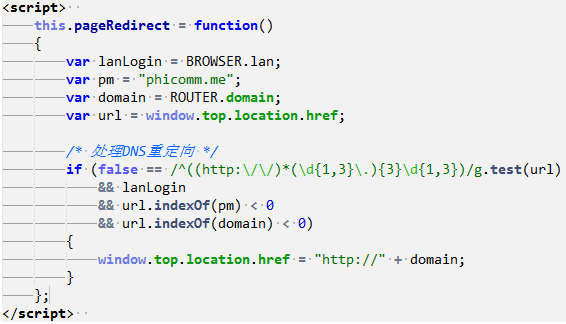
meta是html语言head区的一个辅助性标签。meta标签共有两个属性，它们分别是http-equiv属性和name属性，不同的属性又有不同的参数值，这些不同的参数值就实现了不同的网页功能。

http-equiv属性: 相当于http的文件头作用，它可以向浏览器传回一些有用的信息，以帮助正确和精确地显示网页内容，与之对应的属性值为content，content中的内容其实就是各个参数的变量值。

<meta http-equiv="参数"content="参数变量值">；

### 3.1.2 JS 页面跳转方式

1. 使用window.location.href = "newurl"



也可以用window.location = "newurl"

2. 使用 window.navigate

<script>

window.navigate("http://www.csdn.net");

</script>

## 3.2. 后端实现

后端实现主要是通过响应头中的http响应location字段，令客户端重定向至指定URL。

数据交互过程

：<http://172.17.200.153:8800/bbs/index.php?/topic/167-httpd%E7%AE%80%E4%BB%8B/>

### 3.2.1 http消息

HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

一个HTTP"客户端"是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。一个HTTP"服务器"同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。

HTTP协议的请求和响应都是一段按一定规则组织起来的文本，其请求的头部包括请求行（请求方式method、请求的路径path、协议版本protocol），请求头标（一系列key：value形式组织的文本行），空行（分隔请求头部与数据）和请求数据。

#### 1. 客户端请求

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）、空行和请求数据四个部分组成。



#### 2. 服务器响应消息

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。

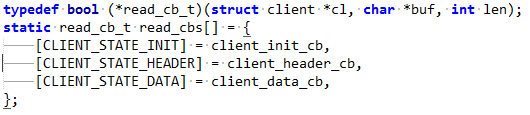


### 3.2.2 http消息源码分析

#### 1 客户端请求解析

客服端的请求处理其实就是将请求拆解，分解各个字段，提取出header中的信息。

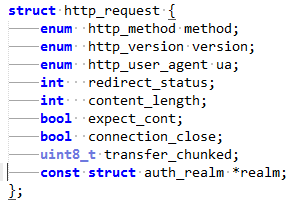
首先，uhttpd会将收到的请求存放在一个buffer中。在uhttpd中有一个状态机来处理http请求



这三个状态分别用来处理客服端请求中的请求行（request line）、请求头部（header）、请求数据。



Uhttpd中默认的需要获取的http请求包括以下字段：



##### **1. CLIENT\_STATE\_INIT**

状态-处理请求行，在Init状态中,调用static bool client\_init\_cb(struct client \*cl, char \*buf, int len)函数来method, url, version



获取成功后将状态变为CLIENT\_STATE\_HEADER

##### 2. CLIENT\_STATE\_HEADER

状态--处理请求头部，调用static bool client\_header\_cb(struct client \*cl, char \*buf, int len)函数来解析request header；

解析的方式就是static bool client\_header\_cb(struct client \*cl, char \*buf, int len)；函数中通过/r/n作为标志将buffer中的数据一行一行读入。然后将每一行数据通过“：”为标志存到结构体中传入static void client\_parse\_header(struct client \*cl, char \*data)；函数中来获取文件头。

这两个函数中将buffer中的http header按照字符串解析的方式取出有用信息，存放到client结构体中。当buffer中全部解析完成之后状态切换到CLIENT\_STATE\_DATA；

##### 3. CLIENT\_STATE\_DATA

处理请求数据，调用函数void client\_poll\_post\_data(struct client \*cl)

没看明白……大概是按照content-length取出数据。╮(╯\_╰)╭

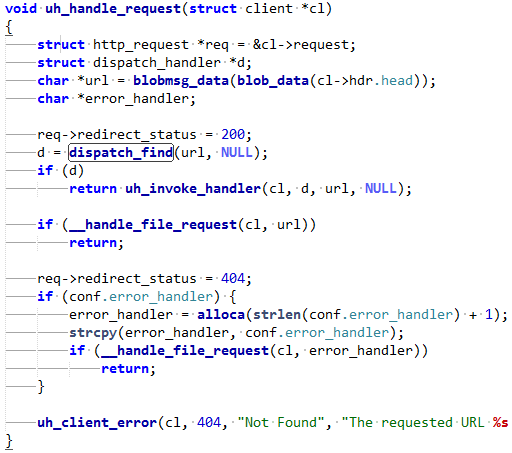
#### 2 服务器响应消息处理

Uhttpd在完成CLIENT\_STATE\_HEADER处理的时候会调用uh\_handle\_request(cl)函数来处理客户端的请求。



响应主要是处理url并返回状态码。响应的处理主要在file.c文件中进行处理。简单的就是讲url当做是相对于www文件夹的文件路径来查找文件。比如p.to/cgi-bin，其中“/cgi-bin”就会进入file.c文件处理。在www文件夹下寻找cgi-bin。

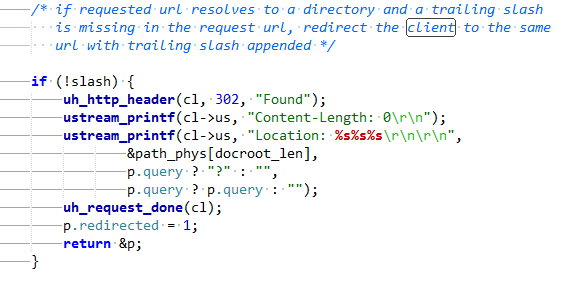
file.c文件的函数入口在void uh\_handle\_request(struct client \*cl)；



在这个函数中调用static bool \_\_handle\_file\_request(struct client \*cl, char \*url)来处理请求；

其中又调用了 static struct path\_info \*uh\_path\_lookup(struct client \*cl, const char \*url)函数来寻找路径。

其中，在uh\_path\_lookup（）函数中，当url访问的是一个目录，但是url中没有“/”的时候会转跳到302，将url加上“/”

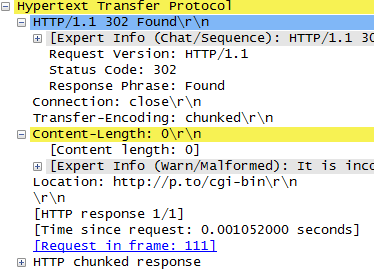


这里面的path\_phys[docroot\_len]为根目录，K2中就是www文件夹，默认将重定向到根文件夹中。

p.query ? "?" : "",

p.query ? p.query : "");用来提取query信息，也就是url中的查询信息。

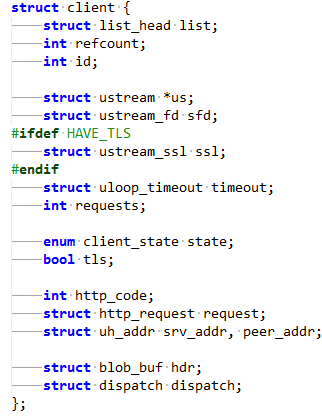
我们可以在这里通过location字段对url进行重定向。



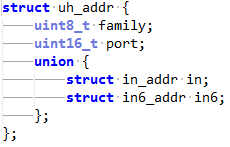
# 附录

## 1 几个重要的结构体

存放客户端数据的结构体client.



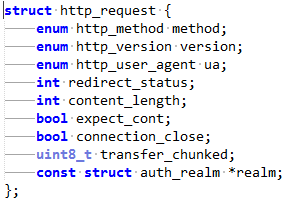
其中，uh\_addr结构体



可以用来表示一个32位的IPv4地址



得到local\_addr,就是我们的lan ip。

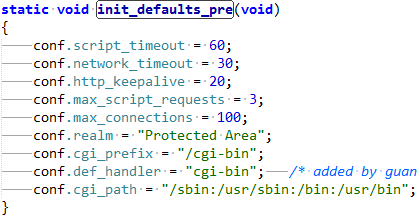


存放http请求和响应的字段

## 2 配置参数读入

Uhttpd的参数位于uhttpd.config文件中。在main.c的main函数中通过while循环读入配置参数；





函数中设定默认初始值。