**WEB基础原理**

## 一、域名概述

1、域名解析的作用：

主机数量增多时，IP地址不容易记忆，域名方便记忆。域名记忆更加直观。

2、hosts文件

早期通过hosts文件进行域名的解析，Linux系统中hosts文件存放路径为/etc/hosts，Windows系统中存放路径为C:\Windows\System32\drivers\etc\hosts内。

但后来随着Internet网上的网站发展迅速，一个小小的hosts文件以不足以存放，再加上主机名称数量庞大、主机维护困难等问题，出现了DNS域名解析服务

1. 加快用户查询速度
2. 便于管理服务器

3、DNS（Domain Name System）域名系统

（1）两大特点：分布式、层次性

（2）域名空间结构：根域、顶级域（国家/地区域名）、二级域

（3）完整域名格式：FQDN=主机名.DNS后缀，例：www.sohu.com.

## 二、网页基本术语概述

1、网页：纯文本格式文件，其编写语言为HTML，在用户的浏览器中被“翻译”成网页形式显示出来

2、网站：网站是由一个一个页面构成的，是多个网页的结合体

3、主页：打开网站后出现的第一个网页称为网站主页（或首页 index.html）

（例如下图为www.sohu.com网站首页）



4、域名：浏览网页时输入的网址（例如：www.sohu.com）

5、HTTP：用来传输网页的通信协议（超文本传输协议）

6、URL：是一种万维网寻址系统（统一资源定位符）

<http://www.baidu.com/jpg/1.jpg>

7、HTML：用来编写网页的超文本标记语言

8、超链接：将网站中不同网页链接起来的功能

9、发布(上线)：将制作好的网页上传到服务器供用户访问的过程

## 三、HTML（超文本标签语言）概述

1、HTML：Hyper Text Markup Language，编写网页的语言

2、浏览器：“解释和执行”HTML源码的工具

3、HTML文档的结构

头部部分

标题部分

主体部分

## 四、网页基本标签

### 1、头部部分

网页摘要信息的作用：有利于浏览器解析及搜索引擎的搜索

**<title>** 标题标签

**<meta>** 标签 （meta-information）针对搜索引擎和更新频度的描述和关键词

示例：

**<head>**

**<title>测试网页</title>**

**<meta name="keywords" content="云计算,Linux,网络服务,IT技术"/>**

**</head>**

### 2、主体部分

1.标题标签

**<h1>-<h6>** <h1>字号最大，<h6>字号最小

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**</body>**

2.行控制相关标签

**<p>** 段落标签

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**<p>这是一个段落</p>**

**</body>**

**<br />** 折行标签

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**<p>这是一个段落</p>**

**测试换行标签<br/>**

**</body>**

3.范围标签

**<span>** 范围标签，组合文档中的行内元素

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**<p>这是一个段落</p>**

**测试换行标签<br/>**

**<span>范围测试**

**范围测试</span> 不加入其他属性的设置，无变化**

**</body>**

4.图像标签

**<img>** 图像标签

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**<p>这是一个段落</p>**

**测试换行标签<br/>**

**<span>范围测试**

**范围测试</span>**

**<img src="E:\test.jpg"/>**

**</body>**

5.超链接标签

<a> 超链接标签

示例：

**<body>**

**<h1>一级标题</h1>**

**<h2>二级标题</h2>**

**<p>这是一个段落</p>**

**测试换行标签<br/>**

**<span>范围测试**

**范围测试</span>**

**<img src="linux.png" />**

**<a href="www.sohu.com">搜狐</a>**

**</body>**

6.特殊符号

&nbsp; 空格符号

&quot; “ 引号

&copy; © 公司

&gt; > 大于号

示例：

**&quot;Crushlinux.com&quot;&nbsp;&copy;版权所有**

**示例与展示：**

**<html>**

**<head>**

**<title>我的测试网页</title>**

**<meta name="keywords" content="云计算，LINUX，SHELL">**

**</head>**

**<body>**

**<h1>一级标题测试</h1>**

**<h3>三级标题测试</h3>**

**<p>这是一个段落的测试</p>**

**测试换行标签<br />**

**<span style="color:red">范围测试</span><br />**

**<img src="linux.png" /><br />**

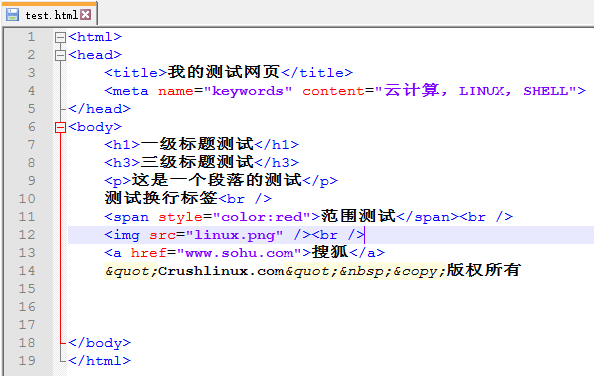
**<a href="www.sohu.com">搜狐</a>**

**&quot;Crushlinux.com&quot;&nbsp;&copy;版权所有**

**</body>**

**</html>**

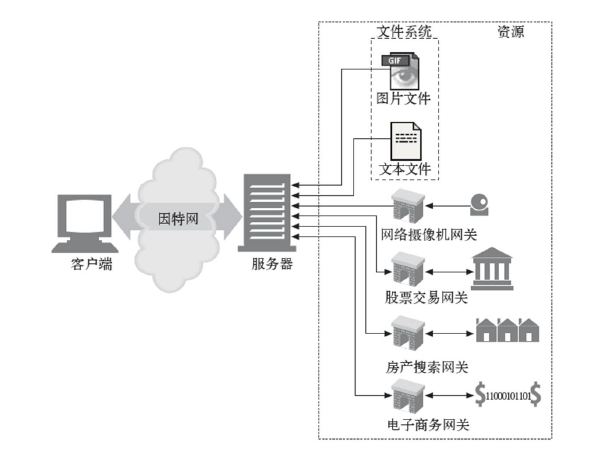
将文件后缀改为.html，用浏览器打开



## 五、Web概述

Web内容储存在Web服务器上，最简单的Web资源就是Web服务器文件系统中的静态文件，这些文件可以包含任意内容：文本文件、HTML文件、微软的Word文件、Adobe的Acrobat文件、JPEG图片文件、AVI电影文件。

资源不一定是静态文件，资源还可以是根据需要生成内容的软件程序。这些动态内容资源可以根据你的用户身份、所请求的信息或每天的不同时段来产生内容。



### 1、WEB1.0与WEB2.0

Web1.0是以编辑为特征，网站提供给用户的内容是编辑处理后结果，然后用户阅读网站提供的内容。这个过程是网站到用户的单向行为。

Web2.0更注重用户的交互作用，用户既是网站内容的消费者（浏览者），也是网站内容的制造者。Web2.0加强了网站与用户之间的互动，网站内容基于用户提供，网站的诸多功能也由用户参与建设，实现了网站与用户双向的交流与参与。

### 2、静态网页与动态网页

（1）静态网页

在网站设计中，纯粹HTML格式的网页通常被称为“静态网页”，静态网页是标准的HTML文件，扩展名是.htm、.html，可包含文本、图像、声音、FLASH动画、客户端脚本和ActiveX控件及JAVA小程序等。静态网页是网站建设的基础，早期的网站一般都是静态网页制作的。静态网页相对于动态网页而言，是没有后台数据库、是不含程序和可交互的网页。静态网页相对更新起来比较麻烦，适用于一般更新叫少的展示型网站（公司首页）。

每一个静态网页都有一个固定的URL，且URL以.htm、.html、.shtml等常见形式为后缀，而不含有“?”。网页内容一经发布到网站服务器上，无论是否有用户访问，每个静态网页都是存在在网站服务器上的，也就是说，静态网页是实实在在保存在服务器上的文件，每个网页都是一个独立的文件。

静态网页的内容相对稳定，因此容易被搜索引擎检索。

静态网页没有数据库的支持，在网站制作和维护方面工作量较大，因此当网站信息量很大时完全依靠静态网页制作方式比较困难。静态网页的交互性较差，在功能方面有较大的限制。页面浏览速度迅速，过程无需连接数据库，开启页面速度快于动态页面。

（2）动态网页



动态网页是与静态网页相对应的，网页URL的后缀不是.htm、.html、.shtml、xml等静态网页的常见形式，而是以.aspx、.asp、.jsp、.php、.perl、.cgi等形式为后缀，并且在动态网页网址中有一个标志性的符号——“?”

动态网页显示的内容可以随着时间、环境或者数据库操作的结果而发生改变。动态网页与网页上的各种动画、滚动字幕等视觉上的动态效果没有直接关系，只要是采用了动态网站技术生产的网页都可以称为“动态网页”。

动态网页是基本的html语法与Java、PHP等高级程序设计语言、数据库编程等多种技术的融合，以实现对网站内容和风格的高效、动态和交互式管理。

动态网页一般以数据库技术为基础，可以大大降低网站维护的工作量。采用动态网页技术的网站可以实现更多的功能，如用户注册、用户登录、在线调查、用户管理、订单管理等等。

动态网页实际上并不是独立存在于服务器上的网页文件，只有当用户请求时服务器才返回一个完整的网页。

## 六、HTTP协议分析

### 一、HTTP协议概述

每天都有数以亿万计的JPEG图片、HTML页面、文本文件、MPEG电影、WAV音频文件、JAVA小程序和其他资源在因特网上游弋。HTTP可以通过遍布全世界的Web服务器上将这些信息快迅速、便捷、可靠地搬移到人们桌面上的Web浏览器上去。

Web服务器所使用的是HTTP协议，因此经常会被称为HTTP服务器。HTTP服务器存储了因特网中的数据，如果HTTP客户端发出请求的话，它们会提供数据，客户端向服务器发送HTTP请求，服务器会在HTTP响应中回送所有请求的数据。

### 二、HTTP协议版本

http://www.sohu.com/login.jsp?u=xxxx&y=xxxxx

HTTP协议是互联网上应用最为广泛的一种网络协议，设计这个协议的目的是为了发布和接收Web服务器上的HTML页面。

**·HTTP 0.9**

已过时。只接受 GET 一种请求方法，没有在通讯中指定版本号，且不支持请求头。由于该版本不支持 POST 方法，所以客户端无法向服务器传递太多信息。

**·HTTP 1.0**

这是第一个在通讯中指定版本号的HTTP 协议版本，至今仍被广泛采用，特别是在代理服务器中。

**·HTTP 1.1**

当前版本。持久连接被默认采用，并能很好地配合代理服务器工作。还支持以管道方式同时发送多个请求，以便降低线路负载，提高传输速度。

**·HTTP 2.0**

HTTP 2.0即超文本传输协议 2.0，是下一代HTTP协议。是由互联网工程任务组（IETF）的Hypertext Transfer Protocol Bis (httpbis)工作小组进行开发。是自1999年http1.1发布后的首个更新。HTTP 2.0在2013年8月进行首次合作共事性测试。在开放互联网上HTTP 2.0将只用于https://网址，而 http://网址将继续使用HTTP/1，目的是在开放互联网上增加使用加密技术，以提供强有力的保护去遏制主动攻击。

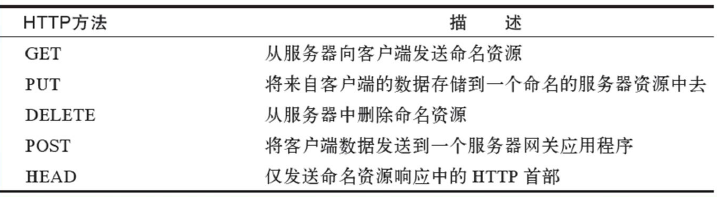
http->80

https->443

### 三、HTTP方法

HTTP支持几种不同的请求命令，这些命令被称为HTTP方法（HTTP method）每条HTTP请求报文会包含一个方法，告诉服务器要执行什么动作：

HTTP协议有多种获得Web资源的方法，常用的有两种：GET和POST



#### GET和POST方法

例如访问：http://www.test.com/a.php?Id=123就是一个GET请求，如果访问正常，我们会从服务器响应报文中获取200状态码。

假如此请求使用POST方法，那么我们会传递给a.php的Id参数依旧是123，但是浏览器的URL将不会显示后面的Id=123字样，因此表单类或者有用户名、密码等内容提交时建议使用POST方式。不管使用哪种方式，最终a.php获取到的值是一样的。

GET方法：从指定的服务器上获取数据 POST方法：提交数据给指定服务器处理

GET请求能被缓存 POST请求不能被缓存

GET请求会保存在浏览器的浏览记录里 POST请求不会保存在浏览器的浏览记录里

GET请求要有长度的限制 POST请求没有长度的限制

主要用于获取数据 查询的字符串不会显示在URL中，比较安全

查询的字符串会显示在URL中，不安全

### 四、HTTP状态码

当使用浏览器访问某一个URL，会根据请求URL返回状态码。

·通常整正常的状态码为2xx，3xx（如200）

·如果出现异常会返回4xx，5xx（如404）。

**2、成功（2字头）**

**200 OK**

请求已成功，请求所希望的响应头或数据体将随此响应返回。

**201 Created**

请求已经被实现，而且有一个新的资源已经依据请求的需要而建立，且其 URI 已经随Location 头信息返回。

**3、重定向（3字头）**

**301 Moved Permanently**

被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个 URI 之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。除非额外指定，否则这个响应也是可缓存的。

新的永久性的URI 应当在响应的 Location 域中返回。除非这是一个 HEAD 请求，否则响应的实体中应当包含指向新的 URI 的超链接及简短说明。

**302 Move temporarily**

请求的资源临时从不同的 URI响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况下，这个响应才是可缓存的。

上文有提及。

如果这不是一个 GET 或者 HEAD 请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。

**304:Not Modfied没有修改**

如果客户端发送了一个带条件的GET 请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变，则服务器应当返回这个304状态码。简单的表达就是：客户端已经执行了GET，但文件未变化。更改文件内容再访问返回代码就是200，缓存cookie值（ETAG字段）也就变了，使用浏览访问，用fiddler抓包可以查看到304代码，需要安装.net,最好用server版本并联网安装fiddler

**4、客户端请求错误（4字头）**

**401 Unauthorized：验证失败**

当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。客户端可以重复提交一个包含恰当的 Authorization 头信息的请求。如果当前请求已经包含了 Authorization 证书，那么401响应代表着服务器验证已经拒绝了那些证书，表示验证失败了。

在apache主配置文件中添加或修改如下配置

Require all denied

authname "保护的领域名称"

authtype basic

authuserfile /usr/local/httpd/conf/.123

require valid-user

启用了用户认证，但用户认证失败就是401错误

/usr/local/httpd/bin/htpasswd -c /usr/local/httpd/conf/.123 ycq 添加验证账户后就可以登陆了

**403 Forbidden：拒绝访问**

**将index.html文件改名，访问时就无法访问到主页（要关闭目录浏览设置，将indexes去掉），或者在主配置文件中设置网站根目录的访问控制，客户机地址不被允许时（**require all denied）**将是403代码**

服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。一般是客户机无权限禁止访问或者无主页无法访问

**404 Not Found：http://www.benet.com/abc 网站目录下无abc目录**

请求失败，请求所希望得到的资源未被在服务器上发现。请求的目录路径不存在

**5、服务器错误（5、6字头）**

**500 Internal Server Error：**

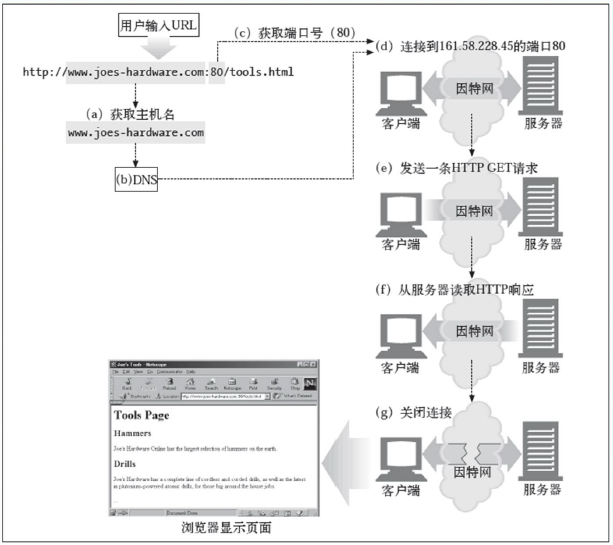
服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器端的源代码出现错误时出现，比如401错误，验证失败了，缓存里的信息还是原来的验证信息，这个时候服务器就报500错误，相当于不允许服务器处理（服务器自身可能没有问题），服务器无法完成这个请求。

**502 Bad Gateway**

作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

服务器连接正常时，将网卡连接断开，这个时候客户端访问就是502错误

### 五、HTTP协议请求流程分析



**六、HTTP响应消息的格式**

一个状态行、若干消息头和实体内容

**HTTP/1.1 200 OK //HTTP协议版本与状态码**

**Date: Thu, 27 Oct 2016 15:50:23 GMT //日期与时间**

**Server: Apache/2 //服务器版本信息**

**X-Powered-By: PHP/5.6.21 //动态服务器版本信息**

**Content-Encoding: gzip //传输压缩**

**Vary: Accept-Encoding,User-Agent**

**Keep-Alive: timeout=2, max=99 //长连接时间**

**Connection: Keep-Alive //支持长连接**

**Transfer-Encoding: chunked**

**Content-Type: text/html; charset=UTF-8 //媒体类型与字符集**

**可以用curl -I（小写i会显示内容代码）获取信息**

**[root@localhost ~]# curl -I http://www.benet.wang**

**HTTP/1.1 200 OK**

**Date: Thu, 27 Oct 2016 16:36:18 GMT**

**Server: Apache/2**

**X-Powered-By: PHP/5.6.21**

**Vary: Accept-Encoding,User-Agent**

**Content-Type: text/html; charset=UTF-8**

### 七、DNS解析流程

以客户端访问www.baidu.com.为例：

1. 查找浏览器DNS缓存及本机hosts文件
2. 查找LDNS(Local DNS)服务器（当地缓存服务器）北京：202.106.0.20
3. LDNS查找根域服务器找到.com服务器
4. LDNS查找.com顶级域服务器baidu.com服务器
5. LDNS请求baidu.com服务器
6. baidu.com服务器返回[www.baidu.com域名的IP解析记录给LDNS](http://www.baidu.com域名的IP解析记录给LDNS)
7. LDNS返回给客户机解析记录及缓存到本地

**[root@localhost ~]# dig +trace www.baidu.com**

**; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.30.rc1.el6 <<>> +trace www.baidu.com**

**;; global options: +cmd**

**. 146276 IN NS b.root-servers.net.**

**. 146276 IN NS g.root-servers.net.**

**. 146276 IN NS l.root-servers.net.**

**. 146276 IN NS d.root-servers.net.**

**. 146276 IN NS m.root-servers.net.**

**. 146276 IN NS k.root-servers.net.**

**. 146276 IN NS f.root-servers.net.**

**. 146276 IN NS h.root-servers.net.**

**. 146276 IN NS j.root-servers.net.**

**. 146276 IN NS e.root-servers.net.**

**. 146276 IN NS a.root-servers.net.**

**. 146276 IN NS i.root-servers.net.**

**. 146276 IN NS c.root-servers.net.**

**;; Received 508 bytes from 202.106.0.20#53(202.106.0.20) in 65 ms**

**#a-m共13台根域DNS服务器，b.root-servers.net.负责了本次解析**

**com. 172800 IN NS a.gtld-servers.net.**

**com. 172800 IN NS b.gtld-servers.net.**

**com. 172800 IN NS c.gtld-servers.net.**

**com. 172800 IN NS d.gtld-servers.net.**

**com. 172800 IN NS e.gtld-servers.net.**

**com. 172800 IN NS f.gtld-servers.net.**

**com. 172800 IN NS g.gtld-servers.net.**

**com. 172800 IN NS h.gtld-servers.net.**

**com. 172800 IN NS i.gtld-servers.net.**

**com. 172800 IN NS j.gtld-servers.net.**

**com. 172800 IN NS k.gtld-servers.net.**

**com. 172800 IN NS l.gtld-servers.net.**

**com. 172800 IN NS m.gtld-servers.net.**

**;; Received 491 bytes from 193.0.14.129#53(193.0.14.129) in 296 ms**

**#解析com顶级域的共有13台DNS服务器，a.gtld-servers.net.负责了本次解析**

**baidu.com. 172800 IN NS dns.baidu.com.**

**baidu.com. 172800 IN NS ns2.baidu.com.**

**baidu.com. 172800 IN NS ns3.baidu.com.**

**baidu.com. 172800 IN NS ns4.baidu.com.**

**baidu.com. 172800 IN NS ns7.baidu.com.**

**;; Received 201 bytes from 192.31.80.30#53(192.31.80.30) in 144 ms**

**#解析baidu.com的有5台服务器，dns.baidu.com. 负责了本次解析**

**www.baidu.com. 1200 IN CNAME www.a.shifen.com.**

**a.shifen.com. 1200 IN NS ns2.a.shifen.com.**

**a.shifen.com. 1200 IN NS ns1.a.shifen.com.**

**a.shifen.com. 1200 IN NS ns5.a.shifen.com.**

**a.shifen.com. 1200 IN NS ns3.a.shifen.com.**

**a.shifen.com. 1200 IN NS ns4.a.shifen.com.**

**;; Received 228 bytes from 119.75.219.82#53(119.75.219.82) in 186 ms**

## 七、日志切割

### Apache日志分割

随着网站访问量的增加，访问日志中的信息会越来越多，Apache默认访问日志（access\_log）单个文件会越来越大，日志文件体积越大，信息都在一个文件中，查看及分析信息会及不方便。

#### Apache 的日志轮询的方式：

第1种：利用apache自带的日志轮循程序rotatelogs

第2种：利用一个发展已经比较成熟的日志轮循工具cronolog

第3种：利用定时任务和shell脚本做日志切割

#### rotatelogs分割工具

**主配置文件中的使用方法**

ErrorLog "|rotatelogs命令的绝对路径 -l logs/error\_%Y%m%d\_%H%M%S.log 秒数"

CustomLog "| rotatelogs命令的绝对路径 -l logs/access\_%Y%m%d\_%H%M%S.log 秒数" combined

**虚拟主机配置文件中的使用方法**

ErrorLog "| rotatelogs命令的绝对路径-l logs/网站名-error\_%Y%m%d\_%H%M%S.log 秒数"

CustomLog "| rotatelogs命令的绝对路径-l logs/网站名-access\_%Y%m%d\_%H%M%S.log 秒数" combined

**操作案例：**

**[root@server01 logs]# sed -i '/httpd-vhosts/s/^#//' /usr/local/httpd/conf/httpd.conf**

**[root@localhost ~]# awk '/httpd-vhosts/{print NR,$0}' /usr/local/httpd/conf/httpd.conf**

**398 Include conf/extra/httpd-vhosts.conf**

**[root@localhost ~]# cat /usr/local/httpd/conf/extra/httpd-vhosts.conf**

**<VirtualHost 192.168.200.111>**

**DocumentRoot "/usr/local/httpd/htdocs/benet"**

**ServerName www.benet.com**

**ErrorLog "|/usr/local/httpd/bin/rotatelogs -l /usr/local/httpd/logs/benet.com-error\_%Y%m%d\_%H%M%S.log 10"**

**CustomLog "|/usr/local/httpd/bin/rotatelogs -l /usr/local/httpd/logs/benet.com-access\_%Y%m%d\_%H%M%S.log 10" combined**

**</VirtualHost>**

**// -l 选项表示localtime，调用本地时间的年月日时分秒**

**// 10 表示十秒切割一次（一天86400）**

**[root@localhost ~]# /usr/local/httpd/bin/httpd -t**

**Syntax OK**

**[root@localhost ~]# mkdir /usr/local/httpd/htdocs/benet**

**[root@localhost ~]# echo "www.benet-test.com" > /usr/local/httpd/htdocs/benet/index.html**

**[root@localhost ~]# /usr/local/httpd/bin/apachectl restart**

**浏览器访问:** [**http://192.168.200.111**](http://192.168.200.101)**测试：**

**[root@localhost ~]# ls /usr/local/httpd/logs/benet.com\***

**/usr/local/httpd/logs/benet.com-access\_20161028\_111440.log /usr/local/httpd/logs/benet.com-access\_20161028\_111820.log**

**/usr/local/httpd/logs/benet.com-access\_20161028\_111500.log**

**按天轮询:**

**CustomLog "|/usr/local/httpd/bin/rotatelogs -l /usr/local/httpd/logs/access\_%Y%m%d.log 86400" combined**

**按周轮询:**

**CustomLog "|/usr/local/httpd/bin/rotatelogs -l /usr/local/httpd/logs/access\_%w.log 604800 " combined**

**按小时轮询:**

**CustomLog "|/usr/local/httpd/bin/rotatelogs –l /usr/local/httpd/logs/access\_%Y%m%d%H.log 3600" combined**

#### cronolog分割工具

cronolog 是一个简单的过滤程序，读取日志文件条目从标准输入和输出的每个条目并写入指定的日志文件的文件名模板和当前的日期和时间当扩展文件名的变化。cronolog 旨在和一个Web服务器一起使用，如Apache，分割访问日志为每天或每月的日志。

cronolog主要和Web服务器配置使用，特别是Apache服务器，Apache 默认日志文件是不分割的，一个整文件既不易于管理，也不易于分析统计。安装cronolog后，可以将日志文件按时间分割，易于管理和分析。

时间域:

H 小时(00..23)

I 小时(01..12)

M 分钟(00..59)

S 秒 (00..60)

日期域:

d 当月中的天数 (01 .. 31)

j 当年中的天数 (001 .. 366)

m 月数 (01 .. 12)

U 当年中的星期数，以周日作为一周开始,其中第一周为首个含星期天的星期(00..53)

W 当年中的星期数，以星期一作为一周的开始,其中第一周为首个含星期天的星期(00..53)

w 工作日数(0 .. 6, 0表示星期天)

y 两位数的年份(00 .. 99)

Y 四位数的年份(1970 .. 2038)

**主配置文件中的使用方法**

ErrorLog "|cronolog命令的绝对路径 /usr/local/httpd/logs/error\_%Y%m%d%H%M%S.log "

CustomLog "|cronolog命令的绝对路径 /usr/local/httpd/logs/access\_%Y%m%d%H%M%S.log " combined

**虚拟主机配置文件中的使用方法**

ErrorLog "|cronolog命令的绝对路径 /usr/local/httpd/logs/网站名-error\_%Y%m%d%H%M%S.log "

CustomLog "|cronolog命令的绝对路径 /usr/local/httpd/logs/网站名-access\_%Y%m%d%H%M%S.log " combined

**操作案例：**

**下载cronlog软件:**

**[root@localhost ~]# wget** [**http://down1.chinaunix.net/distfiles/cronolog-1.6.2.tar.gz**](http://down1.chinaunix.net/distfiles/cronolog-1.6.2.tar.gz)

**[root@localhost ~]# tar xf cronolog-1.6.2.tar.gz -C /usr/src/**

**[root@localhost ~]# cd /usr/src/cronolog-1.6.2/**

**[root@localhost cronolog-1.6.2]# ./configure && make && make install**

**[root@localhost ~]# which cronolog**

**/usr/local/sbin/cronolog**

**[root@localhost ~]# awk '/httpd-vhosts/{print NR,$0}' /usr/local/httpd/conf/httpd.conf**

**398 Include conf/extra/httpd-vhosts.conf**

**[root@localhost ~]# cat /usr/local/httpd/conf/extra/httpd-vhosts.conf**

**<VirtualHost 192.168.200.111>**

**DocumentRoot "/usr/local/httpd/htdocs/benet"**

**ServerName www.benet.com**

**ErrorLog "|/usr/local/sbin/cronolog -l /usr/local/httpd/logs/benet.com-error\_%Y%m%d%H%M%S.log"**

**CustomLog "|/usr/local/sbin/cronolog /usr/local/httpd/logs/benet.com-access\_%Y%m%d%H%M%S.log" combined**

**</VirtualHost>**

**[root@localhost ~]# httpd -t**

**Syntax OK**

**[root@localhost ~]# mkdir /usr/local/httpd/htdocs/benet**

**[root@localhost ~]# echo "www.benet-test.com" > /usr/local/httpd/htdocs/benet/index.html**

**[root@localhost ~]# service httpd stop**

**[root@localhost ~]# service httpd start**

**浏览器访问:** [**http://192.168.200.111**](http://192.168.200.101)**测试：**

**[root@localhost ~]# ls /usr/local/httpd/logs/benet.com\***

**/usr/local/httpd/logs/benet.com-access\_20161028114221.log /usr/local/httpd/logs/benet.com-access\_20161028114222.log**

#### 日志合并

**可以将每个服务器每天的日志文件通过rsync下载到专门进行访问统计分析的服务器上进行合并。**

**合并多个服务器的日志文件，例如：log1 log2 log3并输出到log\_all中的方法是：**

**sort -m -t " " -k 4 -o log\_all access\_log1 access\_log2 access\_log3**

**-m 使用 merge优化算法**

**-k 4 表示根据时间进行排序**

**-o 表示将排序结果存放到指定的文件中**

**[root@localhost logs]# sort -m -t " " -k 4 -o apache\_log benet.com-access\_20161028121053.log benet.com-access\_20161028121054.log benet.com-access\_20161028121055.log**

**[root@localhost logs]# cat apache\_log**

**192.168.200.2 - - [28/Oct/2016:12:10:53 +0800] "GET / HTTP/1.1" 304 - "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)"**

**192.168.200.2 - - [28/Oct/2016:12:10:54 +0800] "GET / HTTP/1.1" 304 - "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)"**