第十章　Web 服务器 Apache 的配置

Apache 是世界排名第一的 Web 服务器，根据 Netcraft(www.netsraft.co.uk)所作的调查,世界上百分之五十以上的 Web 服务器在使用 Apache。

尤其是现在，使用 LAMP（Linux + Apache + MySQL + PHP）来搭建中等级别的应用已经是一种流行的方式，因此，掌握 Apache 的配置是系统工程师必备的技能之一。本章主要介绍 Apache 的配置以及如何构建 LAMP 平台。

1. HTTP 协议简介

超文本传送协议(Hypertext Transfer Protocol，HTTP)是万维网(World Wide Web，WWW，也简称为 Web)的基础。

HTTP 服务器（通常的 Web 服务器）与 HTTP 客户机（通常为网页浏览器）之间的会话如图 10-1 所示：

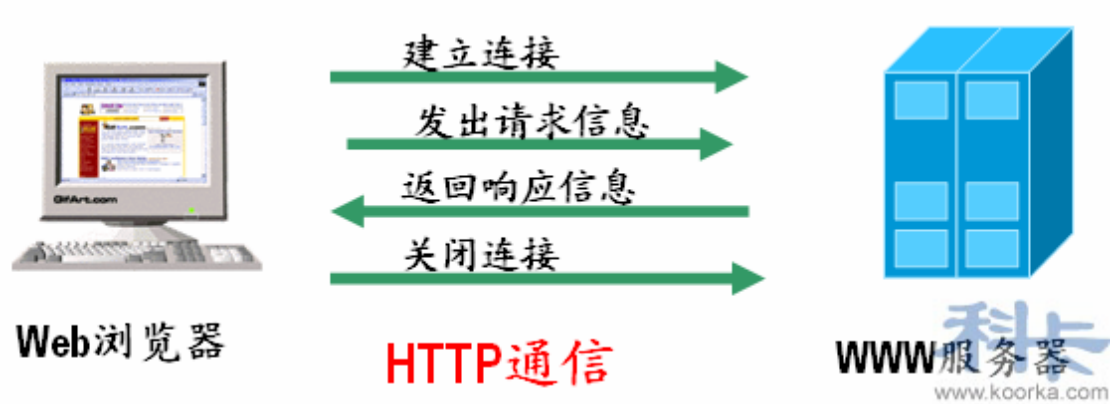

图 10-1

下面对这一过程进行详细分析。

1 客户机与服务器建立连接

与服务器建立连接，就是与 SOCKET 建立连接，因此要指定机器名称、资源名称和端口号，可以通过 URL 来提供这些信息。URL 的格式为：

HTTP：//<IP 地址>/[端口号]/[路径][?<查询信息>] ，例如：

http://www.koorka.com/index.php?op=ShowProductDetail&product_id=1

资源的缺省值是 INDEX 或 DEFAULT，端口号缺省为 80。

2 客户向服务器提出请求

请求信息包括希望返回的文件名和客户机信息。客户机信息以请求头发送给服务器，请求头包括 HTTP 方法和头字段。

HTTP 方法常用的有 GET、HEAD、POST，而 PUT、DELETE、LINK、UNLINK 方法许多 HTTP 服务器都不使用。

头字段（通常叫做 HTTP 头）包括：

。DATE：请求发送的日期和时间

。PARGMA：用于向服务器传输与实现无关的信息。这个字段还用于告诉代理服务器，要从实际服务器而不是从高速缓存取资源

。FORWARDED：可以用来追踪机器之间，而不是客户机和服务器的消息。这个字段可以用来追踪在代理服务器之间的传递路由。

。MESSAGE_ID：用于唯一地标识消息

。ACCEPT： 通知服务器客户所能接受的数据类型和尺寸。（*/*表示可以接受所有类型的数据。

。AOTHORIZATION：向服务器提供旁路安全保护和加密机制，若服务器不需要这个字段，则不提供这个字段

。FROM：当客户应用程序希望想服务器提供有关其电子邮件地址时使用

。IF-MODEFIED-SINCE 用于提供条件 GET。如果所请求的文档自从所指定的日期以来没有发生变化，则服务器应不发送该对象。如果所发送的日期格式不合法，或晚于服务器的日期，服务器会忽略该字段。

。BEFERRER：向服务器进行资源请求用到的对象

。MIME-VERTION：用于处理不同类型文件的 MIME 协议版本号

。USER-AGENT：有关发出请求的客户信息

3 服务器对请求作出应答

服务器收到一个请求，就会立刻解释请求中所用到的方法，并开始处理应答。服务器的应答消息也包含头字段形式的报文信息。

报文第一行是状态行，格式为：

<HTTP 版本号><状态代码><解释短语>

状态码是个三位数字码，分为四类：

。以 2 开头，表示请求被成功处理

。以 3 开头，表示请求被重定向

。以 4 开头，表示客户的请求有错

。以 5 开头，表示服务器不能满足请求

例如，如图 10-2 所示，就是访问一个不存在的网页或目录时返回的响应，注意浏览器的标题栏中的代码 404（图片的左上角）就是返回的状态码：

图 10-2

解释短语是对状态码的解释。例如 Not Found 是对 404 的解释。

响应报文除了返回状态行，还向客户返回几个头字段，如：

。DATE：服务器的时间

。LAST-MODIFIED：网页最后被修改的时间

。SERVER：服务器信息

。CONTENT _TYPE：数据类型

。RETRY_AFTER：服务器太忙时返回这个字段

。WWW_AUTHENTICATE：当服务器的安全机制要求客户发送某中授权信息时使用该字段

报文最后是实体信息，即客户请求得到的 HTTP 服务器上的资源内容。

4 关闭客户与服务器之间的连接

2. 使用 Apache 搭建电子商务网站

使用 LAMP（Linux + Apache + MySQL + PHP）来搭建中等级别的应用（尤其是电子商务）已经是一种流行的方式，因为全部是开源和免费的软件，所以成本非

常低廉。本节介绍平台的搭建。在搭建平台时，也可以直接使用 RPM 包来安装，但是由于使用 RPM 包在一些系统上不支持，因此我们使用更通用的方法：直接从源代码来安装。

2.1 APACHE 简介

1995 年 4 月，最早的 Apache(0.6.2 版)由 Apache Group 公布发行。Apache Group 是一个完全通过 Internet 进行运作的非盈利机构，由它来决定 Apache Web 服务器的标准发行版中应该包含哪些内容，准许任何人修改隐错，提供新的特征和将它移植到新的平台上，以及其它的工作。当新的代码被提交给 Apache Group 时，该团体审核它的具体内容，进行测试，如果认为满意，该代码就会被集成到 Apache 的主要发行版中。

Apache 的特性：

1) 几乎可以运行在所有的计算机平台上.

2) 支持最新的 HTTP/1.1 协议

3) 简单而且强有力的基于文件的配置(HTTPD.CONF)

4) 支持通用网关接口(CGI)

5) 支持虚拟主机.

6) 支持 HTTP 认证.

7) 集成 PERL.

8) 集成的代理服务器

9) 可以通过 WEB 浏览器监视服务器的状态，可以自定义日志.

10) 支持服务器端包含命令(SSI).

11) 支持安全 SOCKET 层(SSL).

12) 具有用户会话过程的跟踪能力.

13) 支持 FASTCGI

14) 支持 JAVA SERVLETS.

2.2 从源代码安装 Apache

通常，对于大多数电子商务网站而言，都会有在线支付系统。为安全起见，对于支付系统需要使用 https 协议来进行访问，也就是需要 SSL 的支持，因此，在开始安装 apache 软件之前，首先要安装 OpenSSL。

安装 OpenSSL 的步骤如下：

（1）获取源代码：

创建存放软件的目录：

mkdir /backup/software (可以存放在任何自己想存放的目录)

到 http://www.openssl.org/source/ 下载最新稳定版的源代码，放到/backup/software。

本案例中下载的是 openssl-0.9.8.tar.gz

（2）解压软件

tar -zxvf openssl-0.9.8.tar.gz

（3）进入源代码目录：

cd openssl-0.9.8

（4）配置编译选项：

./config --prefix=/mnt/software/openssl --shared

--shared 的含义是创建共享库文件，如果不加，在编译时只生成静态库文件。

（5）编译:

make

make test　#测试

（6）安装

make install

执行后将把相关的文件拷贝到/mnt/software/openssl 的对应目录下。

在安装完 OpenSSL 后，接下来就可以安装 Apache 了，安装 Apache 的步骤如下：

（1）获取源代码

创建存放软件的目录：

mkdir /backup/software (可以存放在任何自己想存放的目录)

到 http://www.apache.org/下载最新稳定版的源代码，放到/backup/software。

本案例中下载的是 httpd-2.0.59.tar.bz2

（2）解压软件包

tar –jxvf httpd-2.0.59.tar.bz2

cd httpd-2.0.59

（3）配置编译选项

Apache 是模块化的服务器，核心服务器中只包含了功能最常用的模块，而扩展功能由其他模块提供。设置过程中，你必须指定需要包含的模块。Apache 文档中有模块清单备查，其中状态为"Base"的模块会被默认地包含进核心服务器，如果不需要包含某个模块(比如 mod_userdir)，则必须明确地禁用它；其他状态的模块(比如 mod_expires)，也必须明确启用以使之包含进核心服务器。

Apache 有两种使用模块的方法，其一是永久性包含进核心；如果操作系统支持动态共享对象(DSO)，而且能为 autoconf 所检测，则模块还可以被动态编译。DSO 模块的存储是独立于核心的，可以被核心使用由 mod_so 模块提供的运行时刻配置指令包含或排除。如果编译中包含有任何动态模块，则 mod_so 模块会被自动包含进核心。如果希望核心能够装载 DSO，而不实际编译任何动态模块，需要明确指定 --enable-so。

在我们的电子商务网站中，一般情况下，核心模块功能我们全部启用；除此之外，我们还要启用 SSL 加密（mod_ssl）；为了使搜索引擎更容易收录我们的网页，需要将动态页面的 URL 重写为静态页面的 URL，需要 mod_rewrite；为了今后动态添加模块而不重新编译 apache（例如添加 PHP 的支持），需要启用 mod_so。基于上面的分析，我在配置编译选项时，推荐使用以下选项：

./configure --prefix=/mnt/software/apache2 --with-ssl=/mnt/software/openssl --enable-ssl --enable-so --enable-rewrite --enable-mime-magic --enable-mem-cache

每个项目及网站的情况不同，如果还需要支持其它的模块，请在编译时使用相应的选项。在工作或学习中，如有问题请及时到 http://www.koorka.com/进行答疑。

（4）编译并安装

make && make install

2.3 从源代码安装 PHP

PHP 是一种嵌入在 HTML 并由服务器解释的脚本语言。它可以用于管理动态内容、支持数据库、处理会话跟踪，甚至构建整个电子商务站点。它支持许多流行的数据库，包括 MySQL、PostgreSQL、Oracle、Sybase、Informix 和 Microsoft SQL Server。

要使 PHP 支持相关的功能那么在安装前需要安装相应的软件，例如要使 PHP 支持 MySQL 数据库，在编译 PHP 之前，必须首先安装 MySQL 数据库，关于 MySQL 数据库的安装，查阅第 9 章 MySQL 数据库的安装。

在安装完 Apache 和 MySQL 后，可以开始安装 PHP 了。安装 PHP 的步骤如下：

（1）获取源代码

创建存放软件的目录：

mkdir /backup/software (可以存放在任何自己想存放的目录)

到 http://www.php.net/下载最新稳定版的源代码，放到/backup/software。

本案例中下载的是 php-5.1.4.tar.bz2

（2）解压

tar –jxvf php-5.1.4.tar.bz2

cd php-5.1.4

（3）配置编译选项

刚才已经提到，打算使用 MySQL 来存储数据，因此必须要指名支持 MySQL 数据（--with-mysql），并指名 MySQL 数据的安装位置；如果需要处理 XML 数据，需要 --wiht-xml 和 –with-dom；如果需要使用 PHP 脚本来生成图片，需要使用 --with-gd。总之如果需要使用的功能的库文件不在系统路径内，必须要明确支持，如果不明确支持，那么配置脚本自动进行处理。集体需要使用什么功能，这取决于站点的 PHP 脚本使用的函数，大多数情况下，需要这样来配置编译选项：

./configure --prefix=/mnt/software/php51
--with-apxs2=/mnt/software/apache2/bin/apxs --with-mysql=/mnt/software/mysql
--with-mysql-sock=/mnt/software/mysql/tmp/mysql.sock --with-xml --with-dom
--with-mcrypt --with-iconv --with-gd --with-mime-magic
--with-openssl=/mnt/software/openssl --enable-ftp

其中--with-apxs2=/mnt/software/apache2/bin/apxs 选项的作用是：在安装时会修改 APACHE 的配置文件，加入 PHP 模块，同时将模块复制到 apache 的模块目录下。

（4）编译并安装

make

make install

（5）拷贝 php 的配置文件

cp php.ini-dist /mnt/software/php51/php/php.ini

如果在编译 php 时使用--with-config-file-path=[dir]来明确指明配置文件的存储位置，那么就拷贝到指定位置，否则就拷贝到：安装目录/php/

2.4 测试安装是否成功

要进行测试，修改/mnt/software/apache2/conf/httpd.conf：

（1）修改 DocumentRoot，的值，指定 web 页面的存储位置

DocumentRoot "/mnt/web"

（2）确保文件中已经存在并且启用下面的行

```
LoadModule php5_module        modules/libphp5.so
```

（3）在文件中添加下面的行：

```
AddType application/x-httpd-php .php
```

目的是使以 php 为扩展名的文件会使用 PHP 程序来解析 。

（4）创建一个测试文件/mnt/web/test.php，其内容如下：

```
<?php
phpinfo();
?>
```

（5）启动 apache 服务：

```
/mnt/software/apache2/bin/apachectl start
```

（6）在浏览器中访问刚才的页面，例如：http://localhost/test.php

如果成功返回 php 的相关信息，说明安装成功。

3. Apache 的配置

Apache 的配置由 httpd.conf 文件配置，因此下面的配置指令都是在 httpd.conf 文件中修改。

3.1 基本配置

如果要然服务器比较安全和正常地工作，新安装的 Apache 至少应该修改以下几项：

```
ServerRoot "/mnt/software/apache2"    # apache 软件安装的位置。其它指定的目录如果没有指定绝对路径，则目录是相对于该目录。
PidFile logs/httpd.pid      #第一个 httpd 进程(所有其他进程的父进程)的进程号文件位置。
Listen 80 #服务器监听的端口号。
ServerName www.koorka.com:80        #主站点名称（网站的主机名）。
```

```
ServerAdmin admin@koorka.com        #管理员的邮件地址。
DocumentRoot "/mnt/web/clusting"    #主站点的网页存储位置。

以下是对主站点的目录进行访问控制：
<Directory "/mnt/web/clusting">
Options FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>

DirectoryIndex index.html index.htm index.php   #主页文件的设置（本例将主页文件设置为：index.html,index.htm 和 index.php）
```

在上面这段目录属性配置中，主要有下面的选项：

Options：配置在特定目录使用哪些特性，常用的值和基本含义如下：

ExecCGI: 在该目录下允许执行 CGI 脚本。

FollowSymLinks: 在该目录下允许文件系统使用符号连接。

Indexes: 当用户访问该目录时，如果用户找不到 DirectoryIndex 指定的主页文件(例如 index.html),则返回该目录下的文件列表给用户。

SymLinksIfOwnerMatch: 当使用符号连接时，只有当符号连接的文件拥有者与实际文件的拥有者相同时才可以访问。

AllowOverride：允许存在于.htaccess 文件中的指令类型(.htaccess 文件名是可以改变的，其文件名由 AccessFileName 指令决定)：

None: 当 AllowOverride 被设置为 None 时。不搜索该目录下的.htaccess 文件(可以减小服务器开销)。

All: 在.htaccess 文件中可以使用所有的指令。

Order：控制在访问时 Allow 和 Deny 两个访问规则哪个优先：

Allow：允许访问的主机列表(可用域名或子网，例如：Allow from 192.168.0.0/16)。

Deny：拒绝访问的主机列表。

3.2 服务器的优化 (MPM: Multi-Processing Modules)

apache2 主要的优势就是对多处理器的支持更好，在编译时同过使用--with-mpm 选项来决定 apache2 的工作模式。如果知道当前的 apache2 使用什么工作机制，可以通过 httpd -l 命令列出 apache 的所有模块，就可以知道其工作方式：

prefork：如果 httpd -l 列出 prefork.c，则需要对下面的段进行配置：

```
<IfModule prefork.c>
StartServers 5 #启动 apache 时启动的 httpd 进程个数。
MinSpareServers 5 #服务器保持的最小空闲进程数。
MaxSpareServers 10 #服务器保持的最大空闲进程数。
MaxClients 150 #最大并发连接数。
MaxRequestsPerChild 1000 #每个子进程被请求服务多少次后被 kill 掉。0 表示不限制，推荐设置为 1000。
</IfModule>
```

在该工作模式下，服务器启动后起动 5 个 httpd 进程(加父进程共 6 个，通过 ps -ax|grep httpd 命令可以看到)。当有用户连接时，apache 会使用一个空闲进程为该连接服务，同时父进程会 fork 一个子进程。直到内存中的空闲进程达到 MaxSpareServers。该模式是为了兼容一些旧版本的程序。我缺省编译时的选项。

worker：如果 httpd -l 列出 worker.c，则需要对下面的段进行配置：

```
<IfModule worker.c>
StartServers 2 #启动 apache 时启动的 httpd 进程个数。
MaxClients 150 #最大并发连接数。
MinSpareThreads 25 #服务器保持的最小空闲线程数。
MaxSpareThreads 75 #服务器保持的最大空闲线程数。
ThreadsPerChild 25 #每个子进程的产生的线程数。
MaxRequestsPerChild 0 #每个子进程被请求服务多少次后被 kill 掉。0 表示不限制，推荐设置为 1000。
</IfModule>
```

该模式是由线程来监听客户的连接。当有新客户连接时，由其中的一个空闲线程接受连接。服务器在启动时启动两个进程，每个进程产生的线程数是固定的(ThreadsPerChild 决定)，因此启动时有 50 个线程。当 50 个线程不够用时，服务器自动 fork 一个进程，再产生 25 个线程。

perchild：如果 httpd -l 列出 perchild.c，则需要对下面的段进行配置：

```
<IfModule perchild.c>
NumServers 5 #服务器启动时启动的子进程数
```

StartThreads 5 #每个子进程启动时启动的线程数

MinSpareThreads 5 #内存中的最小空闲线程数

MaxSpareThreads 10 #最大空闲线程数

MaxThreadsPerChild 2000 #每个线程最多被请求多少次后退出。0 不受限制。

MaxRequestsPerChild 10000 #每个子进程服务多少次后被重新 fork。0 表示不受限制。

</IfModule>

该模式下，子进程的数量是固定的，线程数不受限制。当客户端连接到服务器时，又空闲的线程提供服务。 如果空闲线程数不够，子进程自动产生线程来为新的连接服务。该模式用于多站点服务器。

3.3 HTTP 返头回信息配置:

ServerTokens Prod #该参数设置 http 头部返回的 apache 版本信息，可用的值和含义如下：

Prod：仅软件名称，例如：apache

Major：包括主版本号，例如：apache/2

Minor：包括次版本号，例如：apache/2.0

Min：仅 apache 的完整版本号，例如：apache/2.0.54

OS：包括操作系统类型，例如：apache/2.0.54（Unix）

Full：包括 apache 支持的模块及模块版本号，例如：Apache/2.0.54 (Unix) mod_ssl/2.0.54 OpenSSL/0.9.7g

ServerSignature Off #在页面产生错误时是否出现服务器版本信息。推荐设置为 Off

3.4 持久性连接设置

下面几个选项主要配置持久连接的选项：

Timeout 300 #超时，发送或接收两个 TCP 包之间的时间间隔

KeepAlive On #开启持久性连接功能。即当客户端连接到服务器，下载完数据后仍然保持连接状态。

MaxKeepAliveRequests 100 #一个连接服务的最多请求次数。

KeepAliveTimeout 30 #持续连接多长时间，该连接没有再请求数据，则断开该连接。缺省为 15 秒。

3.5 别名设置与 URL 重定向

3.5.1 别名（虚拟目录）

对于不在 DocumentRoot 指定的目录内的页面，既可以使用符号连接，也可以使用别名。别名的设置如下：

```
Alias /download/ "/var/www/download/"    #访问时可以输入：http://www.custing.com/download/

<Directory "/var/www/download"> #对该目录进行访问控制设置
Options Indexes MultiViews
AllowOverride AuthConfig
Order allow,deny
Allow from all
</Directory>
```

3.5.2 AliasMatch

当使用 Alias 时，只能明确匹配，但是有时候我们需要动态匹配或模糊匹配，这就需要另一个命令 AliasMatch。

个指令与 Alias 等效， 但是它使用了标准的正则表达式，而不是简单的前缀匹配。如果此正则表达式与 URL-path 相匹配， 则服务器会用给定的字符串替换加了括弧的匹配，并视之为一个文件名。

例如：

```
AliasMatch ^/manual(?:/(?:de|en|fr|ja|ko|ru))?(/.*)?$ "/var/www/manual$1"
```

当使用/manual/de/howto/index.html 与表达式 ^/manual(?:/(?:de|en|fr|ja|ko|ru))?(/.*)?$ 相匹配，捕获的第一个子串为 /howto/index.html，因此路径为/var/www/manual/howto/index.html

在使用 AliasMatch 指令时，用到了正则表达式。关于正则表达式的更多信息，请参考 http://www.pcre.org/pcre.txt。下面介绍正则表达式中常用的几个字符的含义：

\ ：转义字符

^：用在开始，表示匹配的模式出现在匹配对象的开头。用在[ ]中表示排除字符类，但仅当其为第一个字符时有效。例如：“^pw”任何以 pw 开头的字符串。

$：表示匹配的模式出现在匹配对象的末尾 。例如："\.gif$"任何以".gif 结尾"的字符串

.：匹配除了换行符外的任意一个字符（默认情况下）

|: 开始一个多选一的分支

[ ]：字符类定义。例如："^[abc]" ，任何以 a、b、c 开头。"[^abc]\.html"， 除 abc.html 以外，匹配*.html

（ ）：子模式。子模式由圆括号定界，可以嵌套。 将多选一的分支局部化，例如：cat(aract|erpillar|)匹配了 "cat"，"cataract" 或 "caterpillar" 之一，没有圆括号的话将匹配 "cataract"，"erpillar" 或空字符串。 将子模式设定为捕获子模式，例如：如果将字符串 "the red king" 来和模式 the ((red|white) (king|queen))进行匹配，捕获的子串为 "red king"，"red" 以及 "king"，并被计为 1，2 和 3（第一次匹配外层括号）。 如果左括号后面跟着 "?:"，子模式不做任何捕获，并且在计算任何之后捕获的子模式时也不算在内。例如，如果用字符串 "the white queen" 去和模式 the ((?:red|white) (king|queen)) 匹配，捕获的子串是 "white queen" 和 "queen"，并被计为 1 和 2。

?：扩展 “(” 的含义，也是 0 或 1 数量限定符，以及数量限定符最小值

* ：匹配 0 个或多个的数量限定符

+ ：匹配 1 个或多个的数量限定符

{ }： 最少 / 最多数量限定。 a{2，4},a 字母最少 2 次，最多 4 次

3.6 CGI 设置

CGI（通用网关接口），也是 Web 后台程序常用的程序之一，通常用 C、C++等语言写成的 web 应用程序需要以 CGI 的方式来进行执行。

ScriptAlias /cgi-bin/ "/mnt/software/apache2/cgi-bin/" # 访问时可以：http://www.clusting.com/cgi-bin/ 。但是该目录下的 CGI 脚本文件要加可执行权限！

```
<Directory "/usr/local/apache2/cgi-bin"> #设置目录属性
AllowOverride None
Options None
Order allow,deny
Allow from all
</Directory>
```

3.7 个人主页的设置 (public_html)

这里的个人主页指的是系统用户的个人主页，通常有以下几种设置方法：

UserDir public_html (用户的主页存储在用户主目录下的 public_html 目录下 URL http://www.koorka.com/~bearzhang/file.html 将读取 /home/bearzhang/public_html/file.html 文件)

UserDir /var/html (URL http://www.koorka.com/~bearzhang/file.html 将读取 /var/html/bearzhang/file.html)

UserDir /var/www/*/docs URL http://www.koorka.com/~bearzhang/file.html 将读取 /var/www/bearzhang/docs/file.html)

如果主页存储在用户的主目录内，别忘了修改权限使其它用户拥有读取的权限：

```
chmod 755 /home/bearzhang  #使其它用户能够读取该文件。
```

3.8 日志的设置

日志记录了 Web 服务器的错误信息以及访问信息，是分析网站的流量和访问量的重要依据。

3.8.1 错误日志的设置

```
ErrorLog logs/error_log #日志的保存位置
LogLevel warn #日志的级别
```

显示的格式日下：

[Mon Oct 10 15:54:29 2006] [error] [client 192.168.10.22] access to /download/ failed, reason: user admin not allowed access

3.8.2 访问日志设置

日志的缺省格式有如下几种：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common #common 为日志格式名称
```

```
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

定义访问日志时使用下面的命令，其中 common 代表记录日志的格式：

```
CustomLog    logs/access_log    common
```

格式中的各个参数如下：

%h --客户端的 ip 地址或主机名

%l --The 这是由客户端 identd 判断的 RFC 1413 身份，输出中的符号 "-" 表示此处信息无效。

%u --由 HTTP 认证系统得到的访问该网页的客户名。有认证时才有效，输出中的符号 "-" 表示此处信息无效。

%t --服务器完成对请求的处理时的时间。

\"%r"\ --引号中是客户发出的包含了许多有用信息的请求内容。

%>s --这个是服务器返回给客户端的状态码。

%b --最后这项是返回给客户端的不包括响应头的字节数。

\"%{Referer}i\" --此项指明了该请求是从被哪个网页提交过来的。

\"%{User-Agent}i\" --此项是客户浏览器提供的浏览器识别信息。

下面是一段访问日志的实例：

192.168.10.22 - bearzhang [10/Oct/2006:16:53:06 +0800] "GET /download/ HTTP/1.1" 200 1228

192.168.10.22 - - [10/Oct/2006:16:53:06 +0800] "GET /icons/blank.gif HTTP/1.1" 304 -

192.168.10.22 - - [10/Oct/2006:16:53:06 +0800] "GET /icons/back.gif HTTP/1.1" 304 -

3.9 用户认证的配置

在 APACHE 中，用户的认证可以采用数据来存储用户认证信息，不过我们这里先介绍使用文本来存储用户的信息。例如我们要使用户在访问 /download/ 时需要认证才能访问，需要进行如下的配置：

(1)在配置文件 httpd.conf，需要有如下的配置：

```
AccessFileName .htaccess      #指名访问控制文件的文件名
.........
Alias /download/ "/var/www/download/"
<Directory "/var/www/download">
Options Indexes
AllowOverride AuthConfig
</Directory>
```

注意 AllowOverride AuthConfig 就是启用了用户认证

(2) 创建用户信息文件:

```
/usr/local/apache2/bin/htpasswd -c /var/httpuser/passwords bearzhang
```

(3)创建用户访问控制文件.htaccess，在本案例中为 /var/www/download/.htaccess 文件的内容如下

```
AuthType Basic        #基本认证
AuthName "Restricted Files"        #采用文件认证
AuthUserFile /var/httpuser/passwords        #用户信息存储位置
Require user bearzhang              #只有 bearzhang 用户可以访问
#Require valid-user #all valid user
```

3.10 虚拟主机的配置

(1)基于 IP 地址的虚拟主机配置

```
Listen 80
<VirtualHost 172.20.30.40>
DocumentRoot /www/example1
ServerName www.example1.com
</VirtualHost>
<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example2.org
</VirtualHost>
```

(2) 基于 IP 和多端口的虚拟主机配置

```
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
Listen 172.20.30.50:80
Listen 172.20.30.50:8080

<VirtualHost 172.20.30.40:80>
DocumentRoot /www/example1-80
ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
```

```
DocumentRoot /www/example1-8080
ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50:80>
DocumentRoot /www/example2-80
ServerName www.example1.org
</VirtualHost>

<VirtualHost 172.20.30.50:8080>
DocumentRoot /www/example2-8080
ServerName www.example2.org
</VirtualHost>
```

(3)单个 IP 地址的服务器上基于域名的虚拟主机配置：

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
DocumentRoot /www/example1
ServerName www.example1.com
ServerAlias example1.com. *.example1.com
# Other directives here
</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/example2
ServerName www.example2.org
# Other directives here
</VirtualHost>
```

(4)在多个 IP 地址的服务器上配置基于域名的虚拟主机：

```
Listen 80

# This is the "main" server running on 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver
```

```
# This is the other address
NameVirtualHost 172.20.30.50

<VirtualHost 172.20.30.50>
DocumentRoot /www/example1
ServerName www.example1.com
# Other directives here ...
</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example2.org
# Other directives here ...
</VirtualHost>
```

(5)在不同的端口上运行不同的站点(基于多端口的服务器上配置基于域名的虚拟主机)：

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
ServerName www.example1.com
DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example1.com
DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
ServerName www.example2.org
DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example2.org
DocumentRoot /www/otherdomain-8080
```

```
</VirtualHost>

(6)基于域名和基于 IP 的混合虚拟主机的配置:
Listen 80

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
DocumentRoot /www/example1
ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/example2
ServerName www.example2.org
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/example3
ServerName www.example3.net
</VirtualHost>
```

3.11 SSL 加密的配置

首先在配置之前先来了解一些基本概念：

证书的概念：首先要有一个根证书，然后用根证书来签发服务器证书和客户证书，一般理解：服务器证书和客户证书是平级关系。SSL 必须安装服务器证书来认证。 因此：在此环境中，至少必须有三个证书：根证书，服务器证书，客户端证书。在生成证书之前，一般会有一个私钥，同时用私钥生成证书请求，再利用证书服务器的根证来签发证书。

SSL 所使用的证书可以自己生成，也可以通过一个商业性 CA（如 Verisign 或 Thawte）签署证书。

签发证书的问题：如果使用的是商业证书，具体的签署方法请查看相关销售商的说明；如果是知己签发的证书，可以使用 openssl 自带的 CA.sh 脚本工具。

如果不为单独的客户端签发证书，客户端证书可以不用生成，客户端与服务器端使用相同的证书。

要启用 SSL 加密，需要下面几个步骤

(1) conf/ssl.conf 配置文件中的主要参数配置如下：

```
Listen 443
SSLPassPhraseDialog buildin
#SSLPassPhraseDialog exec:/path/to/program
SSLSessionCache dbm:/usr/local/apache2/logs/ssl_scache
SSLSessionCacheTimeout 300
SSLMutex file:/usr/local/apache2/logs/ssl_mutex

<VirtualHost _default_:443>

# General setup for the virtual host
DocumentRoot "/usr/local/apache2/htdocs"
ServerName www.example.com:443
ServerAdmin you@example.com
ErrorLog /usr/local/apache2/logs/error_log
TransferLog /usr/local/apache2/logs/access_log

SSLEngine on
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key
CustomLog /usr/local/apache2/logs/ssl_request_log "%t %h
%{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

(2) 创建和使用自签署的证书：

第一步：创建 RSA 私钥

```
/usr/local/openssl/bin/openssl genrsa -des3 -out
/usr/local/apache2/conf/ssl.key/server.key 1024
```

第二步：创建证书签署请求(CSR)

```
/usr/local/openssl/bin/openssl req -new -key
/usr/local/apache2/conf/ssl.key/server.key -out /usr/local/apache2/conf/ssl.key/server.csr
```

第三步：使用 RSA 私钥创建一个自签署的 X509 证书请求

```
/usr/local/openssl/bin/openssl req -x509 -days 365 -key
/usr/local/apache2/conf/ssl.key/server.key -in /usr/local/apache2/conf/ssl.key/server.csr
-out /usr/local/apache2/conf/ssl.crt/server.crt
```