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。   
主站点的配置(基本配置)   
  
(1) 基本配置:   
ServerRoot "/mnt/software/apache2" #你的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.clusting.com:80 #主站点名称（网站的主机名）。   
  
ServerAdmin admin@clusting.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>   
  
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。   
  
其它可用值和含义请参阅：http://www.clusting.com/Apache/ApacheManual/mod/core.html#options   
  
  
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。   
  
其他的可用值及含义(如：Options FileInfo AuthConfig Limit等)，请参看： http://www.clusting.com/Apache/ApacheManual/mod/core.html#AllowOverride   
  
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。   
  
更详细的用法可参看：http://www.clusting.com/Apache/ApacheManual/mod/mod\_access.html#order   
  
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）   
  
  
(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) 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   
  
  
(4) 持久性连接设置   
  
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。   
  
别名设置   
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>   
  
  
CGI设置   
  
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>   
  
  
个人主页的设置 (public\_html)   
  
UserDir public\_html (间用户的主页存储在用户主目录下的public\_html目录下 URL http://www.clusting.com/~bearzhang/file.html 将读取 /home/bearzhang/public\_html/file.html 文件)   
  
chmod 755 /home/bearzhang #使其它用户能够读取该文件。   
  
UserDir /var/html (the URL http://www.clusting.com/~bearzhang/file.html 将读取 /var/html/bearzhang/file.html)   
  
UserDir /var/www/\*/docs (the URL http://www.clusting.com/~bearzhang/file.html 将读取 /var/www/bearzhang/docs/file.html)   
  
日志的设置   
  
(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 2005] [error] [client 192.168.10.22] access to /download/ failed, reason: user admin not allowed access   
  
(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   
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/2005:16:53:06 +0800] "GET /download/ HTTP/1.1" 200 1228   
192.168.10.22 - - [10/Oct/2005:16:53:06 +0800] "GET /icons/blank.gif HTTP/1.1" 304 -   
192.168.10.22 - - [10/Oct/2005:16:53:06 +0800] "GET /icons/back.gif HTTP/1.1" 304 -   
  
各参数的详细解释，请参阅：http://www.clusting.com/Apache/ApacheManual/logs.html   
  
  
用户认证的配置   
(1)in the httpd.conf:   
AccessFileName .htaccess   
.........   
Alias /download/ "/var/www/download/"   
<Directory "/var/www/download">   
Options Indexes   
AllowOverride AuthConfig   
</Directory>   
(2) create a password file:   
/usr/local/apache2/bin/htpasswd -c /var/httpuser/passwords bearzhang   
  
(3)onfigure the server to request a password and tell the server which users are allowed access.   
vi /var/www/download/.htaccess:   
AuthType Basic   
AuthName "Restricted Files"   
AuthUserFile /var/httpuser/passwords   
Require user bearzhang   
#Require valid-user #all valid user   
  
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>   
  
  
  
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。   
(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：   
a.Create a RSA private key for your Apache server   
/usr/local/openssl/bin/openssl genrsa -des3 -out /usr/local/apache2/conf/ssl.key/server.key 1024   
  
b. Create a Certificate Signing Request (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   
  
c. Create a self-signed CA Certificate (X509 structure) with the RSA key of the CA   
/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   
  
/usr/local/openssl/bin/openssl genrsa 1024 -out server.key   
/usr/local/openssl/bin/openssl req -new -key server.key -out server.csr   
/usr/local/openssl/bin/openssl req -x509 -days 365 -key server.key -in server.csr -out server.crt   
  
(3) 创建自己的CA（认证证书），并使用该CA来签署服务器的证书。   
mkdir /CA   
cd /CA   
cp openssl-0.9.7g/apps/CA.sh /CA   
./CA.sh -newca   
openssl genrsa -des3 -out server.key 1024   
openssl req -new -key server.key -out server.csr   
cp server.csr newreq.pem   
./CA.sh -sign   
cp newcert.pem /usr/local/apache2/conf/ssl.crt/server.crt   
cp server.key /usr/local/apache2/conf/ssl.key/