正则表达式

1. 非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

字符	描述
\cx	匹配由 x 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

2. 特殊字符

字符	描述
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
()	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \(和 \)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。（前面星 0 多）
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。（前面加一多）
.	匹配除换行符 \n 之外的任何单字符。要匹配 .，请使用 \. 。（句单）
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。（前面问 01）
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。
|	指明两项之间的一个选择。要匹配 |，请使用 \\|。

3. 限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有*或+或?或{n}或{n,}或{n,m}共6种。

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配 n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中 n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

4. 定位符：

字符	描述
^	匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配。
$	匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与 \n 或 \r 之前的位置匹配。
\b	匹配一个字边界，即字与空格间的位置。 （小 b 边）
\B	非字边界匹配。 （大 B 非边）

注意：不能将限定符与定位符一起使用。由于在紧靠换行或者字边界的前面或后面不能有一个以上位置，因此不允许诸如 ^* 之类的表达式。边界字符说明例子:（限定不）

/\Bapt/

匹配 Chapter 中的字符串 apt，但不匹配 aptitude 中的字符串 apt。

/\bapt/

匹配 aptitude 中的字符串 apt，但不匹配Chapter 中的字符串 apt。

5. 其他字符：

字符	描述

x|y	匹配 x 或 y。例如，'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。
[xyz]	字符集合。匹配所包含的任意一个字符。例如， '[abc]' 可以匹配 "plain" 中的 'a'。
[^xyz]	负值字符集合。匹配未包含的任意字符。例如， '[^abc]' 可以匹配 "plain" 中的'p'、'l'、'i'、'n'。
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。
\num	匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。例如，'(.)\1' 匹配两个连续的相同字符（引用一次'.'匹配到的字符）。

第 2 章 企业级 CentOS6.6 操作系统安装

1. 用练习机安装时可设置/、/boot、swap 三个分区，这也是工作中常用的分区方法。如果是在工作环境中，对于数据库及存储的服务器还可以再分出一个/data 分区。（主、boot、swap、data）

2. 标准分区：就是硬盘分区，可以做裸设备，或建文件系统，受硬盘大小限制，可扩展性差。(标准硬盘)

   RAID：磁盘阵列，由很多价格便宜的磁盘，组成一个容量巨大的磁盘组，利用个别磁盘提供数据所产生加成效果提升整个磁盘系统效能。利用这项技术，将数据切割成许多区段，分别存放在各个硬盘上。操作系统的 RAID 功能性能差，冗余也受限于操作系统，因此企业很少使用。(磁盘、磁盘组)

   LVM：逻辑卷管理，它可以对设置好的分区大小进行动态调整，前提是所有的分区格式都需要事先做成 LVM 格式。性能和标准分区以及 RAID 相比还有一定的差距。（L 动态调整）

3. IDE 硬盘（ATA）：个人电脑使用的主流硬盘。（IDE 主流）

   SCSI 硬盘：系统占用率低，转速快，传输率高，但价格高，安装不便，主要应用于服务器。（S 传输高，价格高）

   SATA：和 IDE 类似，IDE 硬盘是并行的，而 SATA 是串行的。（IDE 并，SATA 串）

4. 磁盘分区命名：

(1) 系统的第一块 IDE 接口的硬盘称为/dev/hda，第二块 IDE 接口的硬盘称为/dev/hdb。（IDE、hd）

(2) 系统的第一块SCSI接口的硬盘称为/dev/sda，第二块SCSI接口的硬盘称为/dev/sdb。(SCSI、sd)

(3) 系统的第一块IDE接口硬盘的第1个分区称为/dev/hda1，系统的第一块IDE接口硬盘的第5个分区称为/dev/hda5。

(4) 系统的第二块SCSI接口硬盘的第1个分区称为/dev/sdb1，系统的第二块SCSI接口硬盘的第5个分区称为/dev/sdb5。

5. 门户网站一般的分区方案：假设服务器内存为16GB，硬盘为1TB。
   - /boot分区：100～200MB
   - swap分区：物理内存的1.5～2倍，如果内存大于16GB，可以配置为8～16GB。
   - /分区：80～200GB
   - 剩余空间不分，保留给使用的人根据业务中的具体问题进行划分。
6. 一般要求swap应该是1.5～2倍的物理内存大小。即使没有swap，依旧能够安装和运行Linux操作系统。
7. 安装系统时应该遵循最小化原则，自定义选择包组。如果安装时漏了部分包组，可以使用类似下面的命令补上包组：

```
yum groupinstall 包组名
```

使用类似下列命令查看具体安装的包组组件：

```
yum groupinfo “包组名”                    #注意，有双引号的
```

安装完成后使用下面这条指令查看安装的包情况：

```
yum grouplist hidden
```

8. 安装完成后，通过setup命令来设置网卡。
9. yum源文件有两个，都在/etc/yum.repos.d/目录下：（源、e、y、d）

- CentOS-Base.repo：yum 网络源的配置文件
- CentOS-Media.repo：yum 本地源的配置文件

更新源时，需要到网上下载相应的源文件，然后替换掉/etc/yum.repos.d/目录相应的源文件，然后执行：（源e、y、d）

```
yum clean all              #清除缓存
yum makecache              #生成缓存
```

10. 网卡的配置文件为/etc/sysconfig/network-scripts/ifcfg-eth0，其中 ONBOOT 要设置为 yes，才能保证下次开机启动时激活网卡设备。（网卡 e、s、n、i）

11. 两个 yum 更新命令的区别：

- yum -y update：升级所有包同时也升级软件和系统内核。
- yum -y upgrade：只升级所有包，不升级软件和系统内核。(upgrade 只包)

12. 使用 yum 更新系统：

```
yum update -y
```

13. 使用标准语法查看每个系统进程：

```
ps -ef
```

使用 BSD 语法查看每个系统进程

```
ps  aux
```

注意：ps aux 和 ps -aux 不一样，

14. 尽量不要使用/etc/init.d/network restart 重启网卡，因为这条命令会影响所有网卡，而应该使用 ifdown eth0 以及 ifup eth0 重启网卡。

15. linux 下各种包安装方法：

- deb 包：（dd）

```
dpkg -i <包名>
```

- rpm 包

```
rpm -i xxx.rpm
```

- run 包：

```
./<包名>
```

- bundle 包

```
./<包名>
```

16. /proc：虚拟的目录，是系统内存的映射。可直接访问这个目录来获取系统信息。（proc 系统）

第 3 章 CentOS6.6 连接管理及优化

1. 当前，在几乎所有的互联网企业环境中，最常用的提供 Linux 远程连接服务的工具就是 SSH 软件了，SSH 分为 SSH 客户端和 SSH 服务器端两部分。其中，SSH 服务器端包含的软件程序主要有 openssh 和 openssl，在 Linux 系统中可以用如下方法查询 SSH 服务器端工具的安装情况：

```
[root@www ～] # rpm -qa openssh openssl
openssl-1.0.1e-30.el6.x86_64
openssh-5.3p1-104.el6.x86_64
```

提示：openssh 是提供 SSH 服务的程序，openssl 是为 SSH 提供连接加密的程序。

2. rpm 是 RedHat Package Manget 的缩写，类似 windows 里面的“添加/删除程序”。RPM 共有 10 种基本的模式：它们是安装、查询、验证、删除等。(i 安 q 查 e 删)

- 安装模式：rpm －i
- 查询模式：rpm －q
- 校验模式：rpm －V 或－verify
- 删除模式：rpm －e

如：查询安装的 openssh 和 openssl 版本：

```
Rpm -qa openssh openssl
```

3. 开启和关闭防火墙：

```
[root@lan ~]# /etc/init.d/iptables start
[root@lan ~]# /etc/init.d/iptables stop
```

4. SELinux 是美国国家安全局对于强制访问控制的实现，大多数生产环境选择把它关闭，至于安全问题，可以通过其他手段解决。关闭方式如下：（SELinux、e、s、c）

(1) 修改配置文件，使关闭 SELinux 永久生效：　　（后取代前）

```
sed -i 's/SELINUX=enforcing/SELINUX=disable/' /etc/selinux/config
```

重启生效。

(2) 临时关闭：

```
setenforce 0
```

Setenforce：用于命令行管理 SELinux 的级别，后面的数字表示设置对应的级别。

Getenforce：查看 SELinux 当前的级别。

5. /etc/inittab 目录保存着当前的运行级别：（grep、inittab、运行级别）

```
[root@lan ~]# grep id /etc/inittab
# Individual runlevels are started by /etc/init/rc.conf
id:3:initdefault:
[root@lan ~]#
```

使用 runlevel 查看当前系统的运行级别，使用 init 切换运行级别。后接对应级别的数字。

6. 假设远程 Linux 服务器的 IP 地址为 10.0.0.7，现在要进行连接故障排查。SSH 远程连接故障排查示例：（ssh 故障检查、ping、telnet、iptables）

- 利用 ping 命令检查（客户端执行）
- 利用 telnet 或 nmap 命令检查（客户端执行）。具体命令为：

```
telnet 10.0.0.7 22
```

或者

```
nmap 10.0.0.7 -p 22 仅适合 Linux，需要安装该软件包后才能使用。
```

通过该命令可以查看连接服务器端 10.0.0.7 的 22 端口是不是处于开通状态，因为 SSH 服务默认开启的是 22 端口。

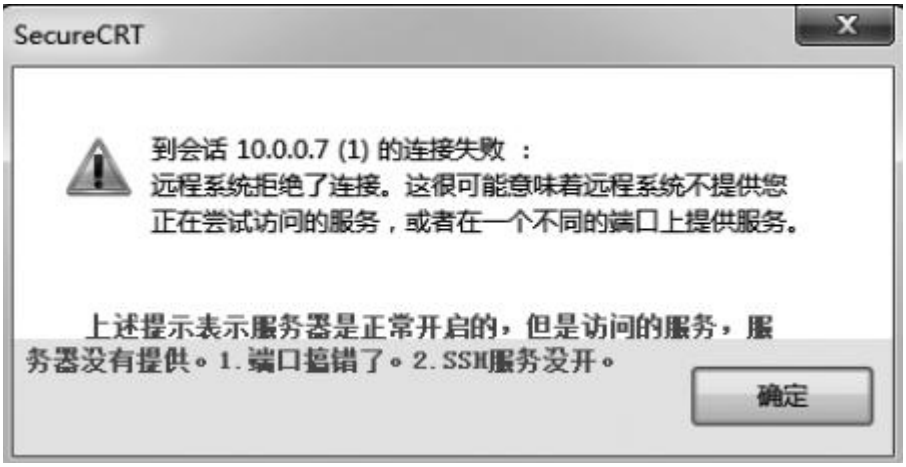
- 检查 iptables 等防火墙策略是否阻挡了连接（服务器端执行）。具体命令为：

```
/etc/init.d/iptables status
```

7. netstat 命令用于显示各种网络相关信息，如网络连接、路由表、接口状态、多播成员等等：

- -a (all)显示所有选项，默认不显示 LISTEN 相关
- -t (tcp)仅显示 tcp 相关选项
- -u (udp)仅显示 udp 相关选项
- -n 拒绝显示别名，能显示数字的全部转化成数字。
- -l 仅列出有在 Listen (监听) 的服務状态
- -p 显示建立相关链接的程序名
- -r 显示路由信息，路由表
- -e 显示扩展信息，例如 uid 等
- -s 按各个协议进行统计
- -c 每隔一个固定时间，执行该 netstat 命令。

8. 使用 SecureCRT 连接服务器时如果出现的是远程系统拒绝连接的提示，则要从 SSH 端口查找原因。(是否开启、默认端口、防火墙、配错端口)



- sshd 服务程序是否正确开启，确认的示例命令如下：

```
[root@www ~]# ps -ef|grep sshd|grep -v grep
root 364 1 0 Oct06 00：00：00 /usr/sbin/sshd <==有这一行表示服务正常
```

- sshd 服务的默认端口 22 是不是被更改了，确认的示例命令如下：（netstat -lntup、端口）

```
[root@www ~]# netstat -lntup|grep sshd
tcp 0 0 0.0.0.0：22 0.0.0.0：* LISTEN 364/sshd
tcp 0 0 ：：22 ：：* LISTEN 364/sshd 提示：22 为端口，如果被改就是其他数字了。
```

- 是否因开启了 iptables 防火墙而导致禁止连接，确认的示例代码如下：

```
[root@www ~]# /etc/init.d/iptables stop
iptables：Setting chains to policy ACCEPT：filter [ OK ]
iptables：Flushing firewall rules： [ OK ]
iptables：Unloading modules： [ OK ]

[root@www ~]# /etc/init.d/iptables sto
```

- 客户端 SecureCRT 是不是配错了连接的端口或 IP。

9. rz、sz 命令的安装方法：

- 第一种方法：安装系统时选包含 rz、sz 命令的包组，即

- 第二种方法：安装系统后通过执行 yum install lrzsz -y 或 yum groupinstall"Dial-up Networking Support" -y 命令来安装。

10. 上传内容时，执行 rz 命令，如果希望覆盖服务器上的同名内容上传，可加-y 参数。下载内容时，执行命令“sz filename”，如果希望覆盖本地的同名内容下载，则可输入“sz -y filename”命令，“sz -y”命令后面的“filename”为命令行 Linux 主机当前目录下的文件。（r 上 s 下）

注意：

- 只能上传和下载文件，不能上传和下载目录，如果是目录需要打包成文件再传。
- 执行 rz 命令按回车键后出现的窗口中，一定不要勾选最下方的“以 ASCII 方式上传文件”，否则会遇到问题，如图所示。

11. 除了 rz、sz 等传输文件命令外，还可以用 ftp、sftp（SSH 服务）等工具来传输文件。

12. 使用下列命令查看查看一下当前 Linux 系统的版本、内核等信息：

```
[root@www ~]# cat /etc/redhat-release
CentOS release 6.6 (Final) ←这是系统版本信息

[root@www ~]# uname -r
2.6.32-504.el6.x86_64 ←这是内核 kernel 的版本号

[root@www ~]# uname -m
x86_64 ←这表示为 64 位系统
```

13. uname 命令用于打印当前系统相关信息（内核版本号、硬件架构、主机名称和操作系统类型等）。

- -a 或—all：显示全部的信息；
- -m 或--machine：显示电脑类型；
- -n 或-nodename：显示在网络上的主机名称；
- -r 或--release：显示操作系统的发行编号；
- -s 或--sysname：显示操作系统名称；
- -v：显示操作系统的版本；
- -p 或--processor：输出处理器类型或"unknown"；
- -i 或--hardware-platform：输出硬件平台或"unknown"；
- -o 或--operating-system：输出操作系统名称；
- --help：显示帮助；

14. 根据老男孩多年的经验，企业环境新装 Linux 系统之后有必要保留的开机自启动服务有 5 个，具体如下：（ssh、日志、网络、定时、sysstat）

- sshd：远程连接 Linux 服务器时需要用到这个服务程序，所以必须要开启，否则 Linux 服务器就无法提供远程连接服务了。
- rsyslog：日志相关软件，这是操作系统提供的一种机制，系统的守护程序通常会使用 rsyslog 程序将各种信息写到各个系统日志文件中，在 CentOS 6 以前此服务的名字为 syslog。
- network：系统启动时，若想激活/关闭各个网络接口，则应（必须）考虑开启此服务。
- crond：该服务用于周期性地执行系统及用户配置的任务计划。有要周期性执行的任务时，就要开启，此服务几乎是生产场景必须要用的一个软件。
- sysstat：sysstat 是一个软件包，包含监测系统性能及效率的一组工具，这些工具对于我们收集系统性能数据很有帮助，比如 CPU 使用率、硬盘和网络吞吐数据等，对这些数据的收集和分析，有利于判断系统运行是否正常，所以它是提高系统运行效率、安全运行服务器的得力助手。

15. sysstat 软件包集成的主要工具为：（systat 软件包）

- iostat 工具提供 CPU 使用率及硬盘吞吐效率的数据。（iostat、CPU 使用率）
- mpstat 工具提供与单个或多个处理器相关的数据。（mpstat、处理器相关数据）
- sar 工具负责收集、报告并存储系统活跃的信息。 （sar、系统活跃信息）

16. 设置开机自启动服务的常见方法：

- 执行命令，然后手动选择处理的方法：执行 setup 命令→system service，然后在弹出的窗口中进行。（开机、setup、system service）

- 通过一行命令或Shell脚本进行设置。

17. 关闭iptables防火墙：

```
[root@www ~]# /etc/init.d/iptables stop
iptables: Setting chains to policy ACCEPT: filter  [ OK ]
iptables: Flushing firewall rules:  [ OK ]
iptables: Unloading modules:  [ OK ]

[root@www ~]# /etc/init.d/iptables stop#←重复执行确认已关闭

[root@www ~]# chkconfig iptables off#←关闭开机自启动命令，前面已经关闭这里就无需执行

[root@www ~]# chkconfig ——list|grep ipt
iptables 0: off 1: off 2: off 3: off 4: off 5: off 6: off
```

18. 最小化原则对Linux系统安全来说极其重要，即多一事不如少一事。具体包括如下几个方面：（安开操登双权限）

- 安装Linux系统最小化，即选包最小化，yum安装软件包也要最小化，无用的包不装。
- 开机自启动服务最小化，即无用无用的服务不开启。
- 操作命令最小化。
- 登录Linux用户最小化。平时没有特殊需求不登录root，用普通用户登录即可。
- 普通用户授权权限最小化，即只给用户必需的管理系统的命令。
- Linux系统文件及目录的权限设置最小化，禁止随意创建、更改、改、删除文件。

21. 为了系统安全，必须隐藏或更改SSH服务器的默认配置： （e、s、s）

```
[root@www ~]#cp /etc/ssh/sshd_config /etc/ssh/sshd_config.ori
#→更改配置前进行备份，是系统管理员的一个良好的习惯。

[root@www ~]#vi /etc/ssh/sshd_config #→编辑sshd_config
####by oldboy#2011-11-24##
Port 52113
PermitRootLogin no
PermitEmptyPasswords no
UseDNS noGSSAPIAuthentication no
####by oldboy#2011-11-24##
```

参数说明：

参　数	说　明
Port	指定 sshd 守护进程监听的端口号，默认为 22。默认在本机的所有网络接口上监听，也可以通过 ListenAddress 指定只在某个特定的接口上监听。 端口范围：0 ~ 65535，不能与已有的服务器端口冲突。 一般建议改为比 1024 大的端口
PermitEmptyPasswords	是否允许密码为空的用户远程登录。默认为“no”
PermitRootLogin	是否允许 root 登录。可用值如下：“yes”（默认）表示允许；“no”表示禁止；“without-password”表示禁止使用密码认证登录；“forced-commands-only”表示只有在指定了 command 选项的情况下才允许使用公钥认证登录，同时其他认证方法全部被禁止，这个值常用于做远程备份之类的事情
UseDNS	指定 sshd 是否应该对远程主机名进行反向解析，以检查此主机名是否与其 IP 地址真实对应。默认值为“yes”。 建议改成“no”，否则可能会导致 SSH 连接很慢
GSSAPIAuthentication no	解决 Linux 之间使用 SSH 远程连接慢的问题㊀

将以上信息更改后，保存退出。执行如下命令重启 sshd，使修改的配置生效：

```
[root@www ～] # /etc/init.d/sshd reload
 或者
[root@www ～] # /etc/init.d/sshd restart
```

提示：reload 为平滑重启，不影响正在 SSH 连接的其他用户，因此要好于 restart。（reload、平滑重启）

注意：1024 以下端口为系统保留端口，1024 到 65535 为用户可用端口。

22. su 和 su -的区别：su 是不更改环境变量的；而 su -是要更改环境变量的。也就是说 su 只是获得了 root 的权限，su -是切换到 root 并获得 root 的环境变量及执行权限。如：（横更）

```
[lan@www ～] #su - root
```

23. 为了安全及管理的方便，可将需要 root 权限的普通用户加入 sudo 管理，这样用户就可以通过自己的普通账户登录，利用 root 的权限来管理系统了，当然也就不需要有 root 账号及密码了。执行如下 visudo 命令，即可打开 sudo 的配置文件进行编辑：（visudo、普通账户可使用 root 权限）

```
[ root@wwwentos ~ ] #visudo #→相当于直接编辑/etc/sudoers，但用命令方式更安全，推荐
```

在/etc/sudoers 文件的大约第 98 行下面，添加需要提升为 root 权限的普通用户名及对应权限，格式如下：

visudo或者vi /etc/sudoers，在98行下面加入，也可以在其他位置加入。

root ALL=(ALL)　　ALL <==此行是98行

oldboy ALL=(ALL)　　/usr/sbin/useradd, /usr/sbin/userdel

例如，在 98 行下面添加下列代码，可使 oldboy 拥有 all 权限：

oldboy ALL=(ALL)　　NOPASSWD: ALL

24. cron 定时任务有两种方式：（用户级、-e）

(1)　用户级：

当前用户在命令行下输入:　　（-e）

```
crontab -e
```

该命令在/var/spool/cron/下创建一个以当前用户命名的文件，如/var/spool/cron/root，并打开，当前用户可以在该文件输入要定时的任务。

(2) 系统级：直接修改/etc/crontab，并将相应的任务文件放到/etc/相应的目录里，如：

```
[root@localhost ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly          //每小时执行/etc/cron.hourly 内的脚本

02 4 * * * root run-parts /etc/cron.daily           //每天执行/etc/cron.daily 内的脚本

22 4 * * 0 root run-parts /etc/cron.weekly          //每星期执行/etc/cron.weekly 内的脚本

42 4 1 * * root run-parts /etc/cron.monthly         //每月去执行/etc/cron.monthly 内的脚本
```

“run-parts”和前面的“22 4”等数字应该是固定的。

25.　相应的 crontab 命令：

- crontab -e：编辑当前用户的定时任务。
- crontab -l：显示当前用户的所有定时任务。
- crontab -r：删除当前用户的定时任务。

26. cron 文件语法与写法：（分、时、日、月、周、命）

```
Minute Hour Day Month Week command
分钟 小时 天 月 星期 命令
0-59 0-23 1-31 1-12 0-6 command
```

含义如下：

```
Minute：每个小时的第几分钟执行该任务
Hour：每天的第几个小时执行该任务
Day：每月的第几天执行该任务
Month：每年的第几个月执行该任务
DayOfWeek：每周的第几天执行该任务，0 表示周日
Command：指定要执行的程序 、脚本或命令
```

几个特殊字符：（星范围、/每、-到）

```
*代表取值范围内的数字
/代表”每“, 如*/5 表示每 5 个单位
-代表从某个数字到某个数字
,分开几个离散的数字
```

如：

- 25 8-11 * * * ls 每天 8-11 点的第 25 分钟执行 ls 命令。
- */15 * * * * ls 每 15 分钟执行一次 ls 命令，注意整个“*/15” 是 Minute。

27. linux 的时间同步服务为 ntp 服务。我们可以手动同步互联网时间到本地 linux 主机，命令如下：（ntp 服务、时间同步、/usr/sbin/ntpdate）

[root@www ~] # /usr/sbin/ntpdate time.nist.gov

利用定时任务 crond 把上述的命令每 5 分钟自动执行一次，命令如下：

[root@www ~] # echo '#time sync by oldboy at 2010-2-1' >>/var/spool/cron/root

[root@www ~] # echo '/5 * * * * /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1' >>/var/spool/cron/root*

[root@www ~] # crontab -l

#time sync by oldboy at 2010-2-1

**/5 * * * * /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1*

28. 如果机器数量大，那么最好在内网部署一个时间同步服务器 ntp server，然后让自动的网内服务器的时间都与 ntp server 同步就可以了。

29. 历史记录数及登录超时环境变量设置

- 设置账号闲置超时时间：（TMOUT、账号闲置超时时间）

```
[root@lan ~]# export TMOUT=10
[root@lan ~]# timed out waiting for input: auto-logout          #仅过 10 秒就显示超时
```

注意这种设置方法仅临时有效。

- 设置 linux 的命令行历史记录数：（HISTSIZE、命令行历史记录数）

```
[root@lan ~]# export HISTSIZE=10
[root@lan ~]# history
   137  echo $TMOUT
   138  export TMOUT=10
   139  history
   140  export HISSIZE=10
   141*
   142  export $HISSIZE=10
   143  export HISTIZE=10
   144  history
   145  export HISTSIZE=10
   146  history
```

这种方法也是临时有效。

要想上述两个命令永久有效，必须将命令写入配置文件/etc/profile 中。

30. export 命令用于设置或显示环境变量：（export、设置环境变量）

```
echo ‘export LC_ALL=C’ >> /etc/profile
```

设置环境变量，并将该语句输出到/etc/profile 文件中。

31. 查看 linux 服务器文件描述符设置的情况可以使用 ulimit -n 命令，文件描述符大小默认是 1024。（ulimit 文件描述符）

```
[ root@www ~ ] # ulimit -n

1024
```

对于高并发的业务 linux 服务器来说，这个默认设置值是不够的，需要调整。

32. 要锁定关键系统文件，必须对账号密码文件及启动文件加锁，防止被篡改。上锁命令如下：（锁定 chattr）

```
chattr +i /etc/passwd /etc/shadow /etc/group /etc/gshadow /etc/inittab
```

上锁后，所有的用户都不能对文件修改删除。解锁命令如下：（改名转移）

```
chattr -i /etc/passwd /etc/shadow /etc/group /etc/gshadow /etc/inittab
```

如果想更加安全，可以把 chattr 改名转移，防止被黑客利用。命令如下：

```
[ root@oldboylinux ~ ] # mv /usr/bin/chattr /usr/bin/oldboy1
```

```
14 password ——md5 $1$hoY96$dM9GlbjKLbi/GV8J9neOm1
```

```
#注意：password要加在splashimage和title之间，否则可能无法生效
```

33. 升级具有漏洞的软件版本：

(1) 首先查看相关软件的版本号，命令如下：

```
[ root@www ~ ] # rpm -qa openssl openssh bash
```

```
openssl-1.0.1e-30.el6.x86_64
```

```
bash-4.1.2-29.el6.x86_64
```

```
openssh-5.3p1-104.el6.x86_64
```

(2) 升级已知漏洞的软件版本到最新，命令如下：

```
[ root@www ~ ] # yum install openssl openssh bash -y
```

```
[ root@www ~ ] # rpm -qa openssl openssh bash
```

```
openssh-5.3p1-104.el6_6.1.x86_64
```

```
bash-4.1.2-29.el6.x86_64
```

```
openssl-1.0.1e-30.el6_6.5.x86_64
```

第 4 章 Web 服务基础

1. 从域名到 IP 的解释过程，称作 A 记录，即 Address Record。（从域名到 IP、A 记录）

状态代码	详细描述说明
500-Internal Server Error	内部服务器错误，服务器遇到了意料不到的情况，不能完成客户的请求。这是一个较为笼统的报错，一般为服务器的设置或内部程序问题导致。例如：SELinux 开启，而又没有为 HTTP 设置规则许可，客户端访问就是 500
502-Bad Gateway	坏的网关，一般是代理服务器请求后端服务时，后端服务不可用或没有完成响应网关服务器。这通常为反向代理服务器下面的节点出问题所致
503-Service Unavailable	服务当前不可用，可能是服务器超载或停机维护导致的，或者是反向代理服务器后面没有可以提供服务的节点
504-Gateway Timeout	网关超时，一般是网关代理服务器请求后端服务时，后端服务没有在特定的时间内完成处理请求。多数是服务器过载导致没有在指定的时间内返回数据给前端代理服务器

2. 通过 curl 命令（附带相关参数）在 linux 命令行下查看 HTTP 响应的数字状态码，命令如下：（curl 命令、状态码）

[root@oldboy ~] # curl -I www.etiantian.org

HTTP/1.1 200 OK← 200即为状态码

Server：nginx/1.6.2

Date：Mon，09 Mar 2015 05：42：25 GMT

Content-Type：text/html；charset=utf-8

Connection：keep-alive

Vary：Accept-Encoding

3. 互联网上的数据有很多不同的类型，WEB 服务器会把通过 WEB 传输的每个对象都打上 MIME 类型的数据格式标签。MIME 类型存在于 HTTP 响应报文的响应头部信息里，它是一种文本标记，表示一种主要的对象类型和一个特定的子类型，中间用一条斜杠来分隔。常见的 MIME 类型：　　（MIME 类型：一主类型一子类型）

MIME 类型	文件类型
text/html	html、htm、shtml 文本类型
text/css	css 文本类型
text/xml	xml 文本类型
image/gif	gif 图像类型
image/jpeg	jpeg、jpg 图像类型
application/javascript	js 文本类型
text/plain	txt 文本类型
application/json	json 文本类型
video/mp4	mp4 视频类型
video/quicktime	mov 视频类型
video/x-flv	flv 视频类型
video/x-ms-wmv	wmv 视频类型
video/x-msvideo	avi 视频类型

4. URI 中文翻译为统一资源标识符，用于标识其一互联网资源名称的字符串。互联网上所有可用的数据资源（如 HTML、图片、视频等）皆通过统一资源标识符进行定位。URL 是 URI 的子集。（URI、标识资源名称的字符串）

说明：根据定义，网络上图片的地址应该就是一种 URI。

5. 动态网页一般在会有标志性的符号— “?”,“&”，此外在大多数情况下后端都需要有数据库的支持。（动态?&）

6. 其他服务并发连接：

- QPS，即每秒查询率，用于衡量一个特定的查询服务器在规定时间内所处理流量多少的标准，如 DNS 系统及数据库服务。（Q 查）
- IOPS，即每秒进行读写操作的次数，多用于数据库等场合，衡量随机访问的性能。（IO 读写）

7. 常用来提供静态 WEB 服务的软件有如下三种：

- Apache：中小型服务的主流，WEB 服务器中的老大哥
- Nginx：大型网站 WEB 服务主流。
- lighttpd：不愠不火的 WEB 优秀软件，社区不活跃，静态解析效率很高。

8. 常用来提供动态服务的软件：

- PHP：大中小型网站都会使用。
- Tomcat：中小企业动态 web 服务主流，互联网 Java 容器主流。
- Resin：大型动态 WEB 服务主流，互联网 Java 容器主流。

- IIS

第 5 章 Nginx Web 服务应用

1. Nginx（“engine x”）是一个开源的，支持高性能、高并发的 WWW 服务和代理服务软件。当前流行的 Nginx Web 组合被称为 LNMP 或 LEMP（即 linux、Nginx、MySQL、PHP）。
2. Nginx 常用命令（以下例子以 Nginx 的安装路径为/usr/local 为例）：

(1) 启动 Nginx

```
/usr/local/nginx/sbin/nginx
```

(2) 停止 Nginx

```
/usr/local/nginx/sbin/nginx -s stop
/usr/local/nginx/sbin/nginx -s quit
```

(3) Nginx 重载配置

```
/usr/local/nginx/sbin/nginx -s reload
```

(4) 指定配置文件

```
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

(5) 查看 Nginx 版本

```
/usr/local/nginx/sbin/nginx -v
或者
/usr/local/nginx/sbin/nginx -V
```

(6) 检查配置文件是否正确

```
/usr/local/nginx/sbin/nginx -t
```

(7) 显示帮助信息

```
/usr/local/nginx/sbin/nginx -h
```

3. 所谓虚拟主机，在 WEB 服务里面就是一个独立的网站站点，这个站点对应独立的域名，具有独立的程序及资源目录，可以独立地对外提供服务供用户访问。这个独立的站点在配置里是由一定格式的标签段标记的，对于 Apache 软件来说，一个虚拟主机

的标签段通常被包含在<VirtualHost></VirtualHost>内，而 Nginx 软件则使用一个 server{}标签来标示一个虚拟主机。（apache、VirtualHost、Nginx、server）

- Apache：<VirtualHost></VirtualHost>
- Nginx：server{}

4. 常见的虚拟主机类型有如下几种：

- 基于域名的虚拟主机
- 基于端口的虚拟主机
- 基于 IP 的虚拟主机

5. diff 命令逐行比较两个文本文件,列出其不同之处。
6. 安装之后在 windows 下可能无法正常访问，但在 linux 下可以 curl 成功，这是防火墙的问题，做如下处理：（注意是/sbin/iptables）

```
[root@localhost html]# /sbin/iptables -I INPUT -p tcp --dport 80 -j ACCEPT
[root@localhost html]# /etc/init.d/iptables save
[root@localhost html]# /etc/init.d/iptables restart
```

相关参数：

- -I xxx：往 xxx 链里插入一条规则。（I 链）
- -p：用于匹配协议（这里的协议通常有 3 种，TCP/UDP/ICMP）（p 协议）
- -dport：指定端口。（dport 端口）
- -j：后接一个 ACCTION，指定如何进行处理（j 处理）

7. Nginx 的默认站点是 Nginx 安装目录/application/nginx 下的 html 目录，该目录由/application/nginx/conf/nginx.conf 设置：

[root@www ~] # grep html /application/nginx/conf/nginx.conf

root html; ← 这个就是默认的站点目录*html*，就是*/application/nginx/html*

index index.html index.htm; ←这是站点的首页文件

提示：此处的/application/nginx/是/application/nginx-1.6.3 的软链接。

8. Nginx 主要的目录和文件：

```
/application/nginx/
|-- client_body_temp
```

```
|-- conf        # Nginx 所有配置文件的目录，极其重要
|   |-- nginx.conf  # 这是 Nginx 默认的主配置文件
|-- html        # 编译安装时 Nginx 的默认站点目录
|   |-- 50x.html    # 错误页面优雅替代显示文件，例如：出现 502 错误时会调用此页面
|   |-- index.html  # 默认的首页文件。首页文件名字是在 nginx.conf 中事先定义好的。
|-- logs        # 这是 Nginx 默认的日志路径，包括错误日志及访问日志
|   `-- nginx.pid   # Nginx 的 pid 文件，Nginx 进程启动后，会把所有进程的 ID 号写到此文件
|-- sbin        # 这是 Nginx 命令的目录，如 Nginx 的启动命令 nginx
|   `-- nginx       # Nginx 的启动命令 nginx
```

9. 如果是配合动态服务（例如 PHP 服务），Nginx 软件还会用到扩展的 FastCGI 相关配置文件，这个配置是通过在 nginx.conf 主配置文件中嵌入 include 命令来实现的，不过默认情况是注释状态，不过默认情况是注释状态，不会生效。（动态 FastCGI）

10. Nginx 配置虚拟主机的步骤如下：（server、目录、重启）

1）增加一个完整的server标签段到结尾处。注意，要放在http的结束大括号前，也就是将server标签段放入http标签。

2）更改server_name及对应网页的root根目录，如果需要其他参数，可以增加或修改。

3）创建server_name域名对应网页的根目录，并且建立测试文件，如果没有index首页，访问会出现403错误。

4）检查Nginx配置文件语法，平滑重启Nginx服务，快速检查启动结果。

11. 使用 include 来使主配置包含虚拟主机子文件：Nginx 的主配置文件为 nginx.conf，主配置文件包含的所有虚拟主机的子配置文件会统一放入 extra 目录中，虚拟主机的配置文件按照网站的域名或功能取名，例如 www.conf、bbs.conf、blog.conf 等。incluce 语法如下：（虚拟主机、域名或功能取名）

include file | mask ;

示例如下：

```
include mime.types ;

include www.conf ;          #<==包含单个文件

include vhosts/*.conf ;     #<==包含vhosts下所有以conf结尾的文件
```

事实上，就是把 nginx.conf 里面的 server 标签（server 标签也要移动）放到一个文件里，并统一放到一个目录中。（server 标签放到一个文件里）

12. sed 命令行格式为：（n 显 i 修）（a 增 c 取 d 删 i 插 p 列）

```
sed [-nefri] ‘command’ 输入文本
```

常用选项（即中括号里面的内容）：

- -n：只有经过 sed 特殊处理的那一行(或者动作)才会被列出来。
- -i：直接修改读取的档案内容，而不在屏幕输出。

常用命令：（command 里面的内容）

- a ：新增， a 的后面可以接字串，而这些字串会在新的一行出现(目前的下一行)～
- c ：取代， c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行！
- d ：删除，因为是删除啊，所以 d 后面通常不接任何咚咚；
- i ：插入， i 的后面可以接字串，而这些字串会在新的一行出现(目前的上一行)；
- p ：列印，亦即将某个选择的资料印出。通常 p 会与参数 sed -n 一起运作～
- s ：取代，可以直接进行取代的工作！通常在后面加上 g 表示替换所有符合要求的内容。

相关示例：

```
nl /etc/passwd | sed '2d'           #只删除第 2 行
nl /etc/passwd | sed '3,$d'         #删除第 3 行到最后一行
```

在第二行后(亦即是加在第三行)加上『drink tea?』字样:

```
[root@www ~]# nl /etc/passwd | sed '2a drink tea'
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
drink tea
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
.....(后面省略).....
```

将字样加在第 2 行前：　　（a 后 i 前）

```
nl /etc/passwd | sed '2i drink tea'
```

假设我们有一文件名为 ab:

```
#删除某行
[root@localhost ruby] # sed '1d' ab          #删除第一行
[root@localhost ruby] # sed '$d' ab          #删除最后一行
[root@localhost ruby] # sed '1,2d' ab        #删除第一行到第二行
[root@localhost ruby] # sed '2,$d' ab        #删除第二行到最后一行

#显示某行
[root@localhost ruby] # sed -n '1p' ab        #显示第一行
```

```
[root@localhost ruby] # sed -n '$p' ab          #显示最后一行
[root@localhost ruby] # sed -n '1,2p' ab        #显示第一行到第二行
[root@localhost ruby] # sed -n '2,$p' ab        #显示第二行到最后一行

#使用模式进行查询
[root@localhost ruby] # sed -n '/ruby/p' ab    #查询包括关键字 ruby 所在所有行
[root@localhost ruby] # sed -n '/\$/p' ab       #查询包括关键字$所在所有行，使用反斜线\屏蔽特殊含义

#增加一行或多行字符串
[root@localhost ruby]# cat ab
Hello!
ruby is me,welcome to my blog.
end
[root@localhost ruby] # sed '1a drink tea' ab  #第一行后增加字符串"drink tea"
Hello!
drink tea
ruby is me,welcome to my blog.
end
[root@localhost ruby] # sed '1,3a drink tea' ab #第一行到第三行后增加字符串"drink tea"
Hello!
drink tea
ruby is me,welcome to my blog.
drink tea
end
drink tea
[root@localhost ruby] # sed '1a drink tea\nor coffee' ab   #第一行后增加多行，使用换行符\n
Hello!
drink tea
or coffee
ruby is me,welcome to my blog.
end

#代替一行或多行
[root@localhost ruby] # sed '1c Hi' ab              #第一行代替为 Hi
Hi
ruby is me,welcome to my blog.
end
[root@localhost ruby] # sed '1,2c Hi' ab            #第一行到第二行代替为 Hi
Hi
end

#替换一行中的某部分
格式：sed 's/要替换的字符串/新的字符串/g'  （要替换的字符串可以用正则表达式）
[root@localhost ruby] # sed -n '/ruby/p' ab | sed 's/ruby/bird/g'   #替换 ruby 为 bird
[root@localhost ruby] # sed -n '/ruby/p' ab | sed 's/ruby//g'       #删除 ruby

#插入
[root@localhost ruby] # sed -i '$a bye' ab        #在文件 ab 中最后一行直接输入"bye"
```

```
[root@localhost ruby]# cat ab
Hello!
ruby is me,welcome to my blog.
end
bye
```

13. 所谓的虚拟主机别名，就是为虚拟主机设置除了主域名以外的一个或多个域名名字，这样就能实现用户访问的多个域名对应同一个虚拟主机网站的功能。具体配置内容如下：（虚拟主机别名、指多个域名）

原始 www 虚拟主机配置（修改前 extra/www.conf 内容）	带别名虚拟主机配置（修改后 extra/www.conf 内容）

[root@www conf]# cat extra/www.conf
#www virtualhost by oldboy
server {
listen 80;
server_name www.etiantian.org;
location / {
root html/www;
index index.html index.htm;
}
}
	[root@www conf]# cat extra/www.conf
	#www virtualhost by oldboy
	server {
	listen 80;
	server_name www.etiantian.org etiantian.org;
	location / {
	root html/www;
	index index.html index.htm;
	}
	}
	# 提示：仅在 server_name 所在行的行尾增加了 etiantian.org 内容！

设置完配置文件之后，重新加载配置：

[root@www conf] # ../sbin/nginx -t

nginx： the configuration file /application/nginx-1.6.3/conf/nginx.conf syntax is ok

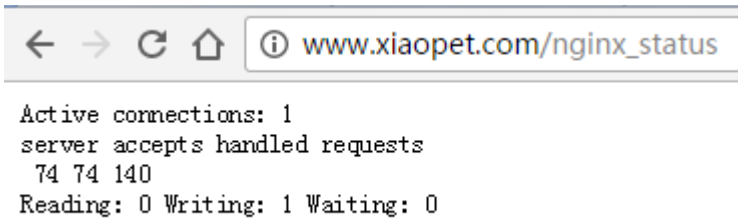
nginx： configuration file /application/nginx-1.6.3/conf/nginx.conf test is successful

[root@www conf] # ../sbin/nginx -s reload

14. Nginx 软件的功能模块中有一个 ngx_http_stub_status_module 模块（Nginx status），主要功能是记录 Nginx 的基本访问状态信息，让使用都了解 Nginx 的工作状态，例如连接数等信息。可以通过在 server 标签里增加下列代码来完成：（stub、access）

```
location /nginx_status {
    stub_status on;
    access_log off;
    allow 10.0.0.0/24;    #<==设置允许和禁止的 IP 段访问
    deny all;    #<==设置允许和禁止的 IP 段访问
}
```

然后输入 xxx.com/nginx_status 来获取状态信息：



上述状态信息具体含义如下：

- active connections：活跃的连接数量　　（活）
- server accepts handled requests：总共处理了 11989 个连接 , 成功创建 11989 次握手, 总共处理了 11991 个请求。
- reading：读取客户端的连接数　　（读）
- writing：响应数据到客户端的数量　　（响）
- waiting：Nginx 已经处理完正在等候下一次请求指令的驻留连接 （等）

15. 在 nginx.conf 中增加错误日志信息的参数是 error_log，语法格式如下：（错误日志、error_log）

```
error_log    file        level;
关键字          日志文件     错误日志级别
```

- error_log：关键字，不能改变
- file：日志文件可以指定任意存放日志的目录。
- level：错误日志级别常见的有
  debug、info、notice、warn、error、crit、alert、emerg，级别越高，记录的信息越少，生产场景一般是 warn、error、crit 这三个级别之一，注意不要配置 info 等较低级别，会带来巨大磁盘 I/O 消耗。

示例如下：

```
worker_processes 1;

error_log logs/error.log;      #<==非常简单，一般配置这一行即可。

events {
```

可以放置的标签段为：main、http、server、location。

16. Nginx 的访问日志由 ngx_http_log_module 模块负责。控制访问日志的参数有两个：（访问日志、access_log）

参　数	说　明
log_format	用来定义记录日志的格式（可以定义多种日志格式，取不同名字即可）
access_log	用来指定日志文件的路径及使用何种日志格式记录日志

- log_format 配置位置在 http 标签内，使用默认格式即可。
- access_log 示例如下

```
access_log  logs/access.log  main
```

后面的 main 表示为日志格式指定的标签，记录日志时通过这个 main 标签选择指定的格式。（main、指定的标签）

17. location 指令的作用就是根据用户请求的网站地址 URL 进行匹配，匹配成功即进行相关的操作。 location 使用的语法为：（location、匹配 URL）

```
Location [=|~|~*|^~]  /uri/ { … }          #注意，中括号和/之间是有个空格的
```

- = 开头表示精确匹配
- ~ 开头表示区分大小写　（区分）
- ~* 开头表示不区分大小写　（带星号不区分）
- ^~ 开头表示 uri 以某个常规字符串开头，理解为匹配 url 路径即可。nginx 不对 url 做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa 匹配到（注意是空格）。
- !~和!~*分别为区分大小写不匹配及不区分大小写不匹配的正则。（叹号不匹配）
- / 通用匹配，任何请求都会匹配到。

18. Nginx rewrite 的主要功能是实现 URL 地址重写，rewrite 规则需要 PCRE 软件的支持，即通过 Perl 兼容正则表达式进行规则匹配。（rewrite、URL 重写）

19. rewrite 指令语法：

```
rewrite regex replacement [flag]
```

其中 rewrite 是关键字，regex 是正则表达式，replacement 是重定向的目的地址，flag 是标记。示例：

```
rewrite ^/(.*)  http://www.etiantian.org/$1 permanent;
```

该规则跳转到 http://etiantian.org/$1。$1 取前面 regex 部分括号里的内容，结尾的 permanent 是永久 301 重定向标记。

20. regex 常用正则表达式字符：

字符	描　　述
\	将后面接着的字符标记为一个特殊字符或一个原义字符或一个向后引用。例如，“\n”匹配一个换行符，序列“\\”和“\$”则匹配“$”
^	匹配输入字符串的起始位置，如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配“\n”或“r”之后的位置
$	匹配输入字符串的结束位置，如果设置了 RegExp 对象的 Multiline 属性，$ 也匹配“\n”或“\r”之前的位置
*	匹配前面的字符零次或多次，例如，ol* 能匹配“o”及“olll”，* 等价于 {0,}
+	匹配前面的字符一次或多次，例如，“ol+”能匹配“ol”及“oll”，但不能匹配“o”，.+ 等价于 {1,}
?	匹配前面的字符零次或一次，例如，“do(es)?”可以匹配“do”或“does”中的“do”，.？等价于 {0,1} 当该字符紧跟在任何一个其他限制符（*,+?,{n},{n},{n,m}）的后面时，匹配模式是非贪婪模式的，非贪婪模式会尽可能少地匹配所搜索的字符串，而默认的贪婪模式则会尽可能多地匹配所搜索的字符串，例如，对于字符串“oooo”，“o+?”将匹配单个“o”，而“o+”将匹配所有“o”
.	匹配除“\n”之外的任何单个字符，要匹配包括“\n”在内的任何字符，请使用像“[.\n]”这样的模式
（pattern）	匹配括号内的 pattern，并可以在后面获取对应的匹配，常用 $0...$9 属性获取小括号中的匹配内容。要匹配圆括号字符，请使用“\(”或“\)”

- ~ 为区分大小写
- ~* 为不区分大小写
- !~和!~*分别为区分大小写不匹配及不区分大小写不匹配（感叹号非表示不匹配）
- -f 和!-f 用来判断是否存在文件
- -d 和!-d 用来判断是否存在目录
- -e 和!-e 用来判断是否存在文件或目录
- -x 和!-x 用来判断文件是否可执行

使用 rewrite 可以实现跳转。例如：

(1) 如果文件不存在，则跳转到首页：　（不存在、跳到首页）

```
If (!-f $request_filename){
      rewrite （.*）/index.php；
}
```

(2)　匹配任何查询，因为所有请求都已/ 开头。但是正则表达式规则和长的块规则将被优先和查询匹配。（匹配任何查询）

```
location = /
```

(3)　匹配任何已/images/开头的任何查询并且停止搜索。任何正则表达式将不会被测试。（匹配 images 开头的查询）

```
location ^~ /images/ {
```

(4)　匹配任何已.gif、.jpg 或 .jpeg 结尾的请求：（匹配结尾的请求）

```
location ~* .(gif|jpg|jpeg)$ {
```

$1-$9 存放着正则表达式中最近的 9 个正则表达式的匹配结果，这些结果按照子匹配的出现顺序依次排列。（$1、$9、最近的 9 个匹配结果）

21. Nginx 访问认证用于给网站设置访问权限，如网站后台。示例配置如下：

```
location / {

auth_basic "closed site";

auth_basic_user_file conf/htpasswd;

}
```

其中，有两个参数需要说明：

- auth_basic：用于设置登陆框的提示字符串。
- auth_basic_user_file：用于设置认证的密码文件。

auth_basic_user_file 参数后接认证密码文件，file 的内容如下：

```
# comment

name1:password1

name2:password2:comment

name3:password3
```

注意，密码是加密的。

使用 htpasswd 生成认证账号和密码示例：

- 首先安装 htpasswd

```
［root@www extra］# yum install httpd -y
［root@www extra］# which htpasswd -y
/usr/bin/htpasswd
```

- 然后执行以下命令：

[root@www extra] # htpasswd -bc /application/nginx/conf/htpasswd oldboy 123456

Adding password for user oldboy

[root@www extra] # chmod 400 /application/nginx/conf/htpasswd

[root@www extra] # chown nginx /application/nginx/conf/htpasswd

[root@www extra] # cat /application/nginx/conf/htpasswd

*oldboy: HWSXCFixaoiZ6 #<==*看到了吧，密码是加密的。

第 6 章 企业级 LNMP 环境应用实践

1. 当 LNMP 组合工作时，首先是用户通过浏览器输入域名请求 Nginx Web 服务，如果请求的是静态资源，则由 Nginx 解析返回给用户；如果是动态请求(.php 结尾)，那么 Nginx 就会把它通过 FastCGI 接口发送给 PHP 引擎服务进行解析，如果这个动态请求要读取数据库数据，那么 PHP 继续向后请求数据库，取得数据后最终通过 Nginx 服务把获取的数据返回给用户。（静态返回，动态解析）
2. /etc/init.d/目录下的脚本程序，一般称为服务，使用 chkconfig 来进行管理。服务一般可以使用下列命令来开启或者关闭开机自启动：　　（on 设 off 关）

chkconfig name on	设置开机自启动
chkconfig name off	关闭开机自启动

3. chkconfig 使用方法：

- chkconfig --list　　列出所有的系统服务
- chkconfig --add name　　增加名为 name 的服务
- chkconfig --del name　　删除 name 服务
- chkconfig --level name 2345 on　　把 name 服务设置为在运行级别为 2、3、4、5 的情况下都是 on（开启）的状态。

4. Linux 系统有 7 个运行级别(runlevel)

- 运行级别 0：系统停机状态，系统默认运行级别不能设为 0，否则不能正常启动
- 运行级别 1：单用户工作状态，root 权限，用于系统维护，禁止远程登陆

- 运行级别 2：多用户状态(没有 NFS)
- 运行级别 3：完全的多用户状态(有 NFS)，登陆后进入控制台命令行模式
- 运行级别 4：系统未使用，保留
- 运行级别 5：X11 控制台，登陆后进入图形 GUI 模式
- 运行级别 6：系统正常关闭并重启，默认运行级别不能设为 6，否则不能正常启动

在/etc/rc.d 下有 7 个名为 rcN.d 的目录，对应系统的 7 个运行级别。目录里的内容就是该运行级别运行的服务，实际上就是指向/etc/init.d/目录相关服务的软链接。对于以 K(Kill)开头的文件，系统将终止对应的服务；对于以 S(Start)开头的文件，系统将启动对应的服务。（rcN.d、软链接）

5. MySQL 的启动脚本为/application/mysql/bin/mysqld_safe
6. MySQL 安装方法有很多：

序号	MySQL 安装方式	特点说明
1	yum/rpm 包安装	特点是简单、速度快，但是没法定制安装，入门新手常用这种方式
2	二进制安装	解压软件，简单配置后就可以使用，不用安装，速度较快，专业 DBA 喜欢这种方式。软件名如：mysql-5.5.32-linux2.6-x86_64.tar.gz
3	源码编译安装	特点是可以定制安装，但是安装时间长，例如：字符集安装路径，等等。软件名如：mysql-5.5.32.tar.gz
4	源码软件结合 yum/rpm 安装	把源码软件制作成符合要求的 rpm，放到 yum 仓库里，然后通过 yum 来安装。结合了上面 1 和 3 的优点，即安装快速，可任意定制参数，但是安装者也需要具备更深能力。本书结尾有 rpm 定制包的内容介绍

7. 可以用以下命令修改用户组：（R 递归处理）

```
chown -R mysql.mysql /application/mysql/
```

格式为：

```
chown [-R] [用户名称.组名称] [文件或目录]
```

参数-R 表示递归处理。

8. 以下 grep 命令用于不显示包含 grep 的行：（grep -v、不显示 grep 的行）

```
grep -v grep
```

－v：不显示包含匹配文本的所有行。

示例如下：

```
        [root@lan ~]# ps -ef | grep php
root      2169     1  0 13:50 ?        00:00:00 php-fpm: master process
(/application/php5.3.27/etc/php-fpm.conf)
nginx     2170   2169  0 13:50 ?        00:00:00 php-fpm: pool www
nginx     2171   2169  0 13:50 ?        00:00:01 php-fpm: pool www
```

```
root      2249   2247  0 14:25 pts/1    00:00:00 grep php                    #grep -v grep 少这个
          [root@lan ~]# ps -ef | grep php | grep -v grep
root      2169      1  0 13:50 ?        00:00:00 php-fpm: master process
(/application/php5.3.27/etc/php-fpm.conf)
nginx     2170   2169  0 13:50 ?        00:00:00 php-fpm: pool www
nginx     2171   2169  0 13:50 ?        00:00:01 php-fpm: pool www
```

9. 安装完成后，默认 root 是无密码的，这个 root 密码必须设置，否则 root 会以匿名方式登陆，使用时会出错。使用 mysqladmin 命令为 MySQL 的 root 用户设置密码，命令如下：（默认无密码、mysqladmin 命令设置密码）

```
［root@www mysql］# mysqladmin -u root password 'oldboy123'        #<==更改默认密码。
```

10. 设置密码后使用 mysql 命令来登陆 mysql：（mysql、登陆）

```
［root@www mysql］# mysql -u root -p
```

11. grant 命令将某对象（表，视图，序列，函数过程语言，或者模式）上的特定权限给予一个用户或者多个用户或者一组用户。这些权限将增加到那些已经赋予的权限上，如果存在这些权限的话。（grant 命令、将特定权限授予特定非所有者用户）

```
grant 权限  on 数据库对象  to 用户
```

将数据库对象的指定权限赋予指定用户。

例如：

```
grant select on *.* to dba@'localhost' ;       #dba 可以查询  MySQL 中所有数据库中的表。
grant all    on *.* to dba@'localhost' ;       #dba 可以管理  MySQL 中的所有数据库
grant select, insert, update, delete on testdb.orders to dba@'localhost' ;     #用户 dba 可以查询、插入、更新、删除数据库 testdb.orders 表
```

其中，dba 是用户，localhost 是地址，非本地时可以填写 IP。

又如：

```
 grant all on wordpress.* to wordpress@'localhost' identified by '847339';
```

创建一个专用的 WordPress 数据库管理用户 wordpress，密码为 847339。格式为：

```
grant 权限  on 数据库对象  to 用户@‘地址’   identified by ‘密码’
```

12. CGI 的全称为“通用网关接口”，为 HTTP 服务器与其他机器上的程序服务通信交流的一种工具，CGI 程序必须运行在网络服务器上。传统 CGI 接口方式的主要缺点是性能较差，故而现在已经很少被使用了。FastCGI 是一个可伸缩地、高速地在 HTTP 服务器和动态脚本语言间通信的接口，主要优点是把动态语言和 HTTP 服务器分离开来。（CGI、通用网关接口、与其他机器通信交流）

13. devel 包主要是供开发用，一般会包括头文件、静态库甚至源码。（devel 包、供开发用）

14. PHP 安装准备：（安装各种库）

(1) 首先安装必备库：

```
yum install zlib-devel libxml2-devel libjpeg-devel libjpeg-turbo-devel libiconv-devel -y
yum install freetype-devel libpng-devel gd-devel libcurl-devel libxslt-devel -y
```

此时，会有 libiconv-devel 安装不上。因为默认源没有这个包。

(2) 下载安装安装不上的 libiconv-devel：

tar zxf libiconv-1.14.tar.gz

cd libiconv-1.14

./configure ——prefix=/usr/local/libiconv

make

make install

(3) 配置第三方 yum 源（下面几个包需要）：

wget -O /etc/yum.repos.d/epel.repo http : //mirrors.aliyun.com/repo/epel-6.repo

(4) 安装 libmcrypt：

```
yum -y install libmcrypt-devel
```

(5) 安装 mhash 加密扩展库：

```
yum -y install mhash
```

(6) 安装 mcrypt 加密扩展库：

```
yum -y install mcrypt
```

15. PHP 安装步骤（以安装到/application 为例）：（配置、编译、安装、软链接、php.ini、php-fpm.conf、启动、检查端口、local）

(1) 下载并解压，进入目录

(2) 配置

```
./configure --prefix=/application/php5.3.27 --with-mysql=/application/mysql --with-iconv=/usr/local/libiconv --with-freetype-dir --with-jpeg-dir --with-png-dir --with-zlib --with-libxml-dir=/usr --enable-xml --enable-rpath --enable-safe-mode --enable-bcmath --enable-shmop --enable-sysvsem -enable-inline-optimization --with-curl --with-curlwrappers  --enable-mbregex --enable-fpm --enable-mbstring --with-mcrypt --with-gd --enable-gd-native-ttf --with-openssl --with-mhash --enable-pcntl --enable-sockets --with-xmlrpc --enable-zip --enable-soap --enable-short-tags --enable-zend-multibyte --enable-static --with-xsl --with-fpm-user=nginx --with-fpm-group=nginx
```

(3) 编译 PHP

```
ln -s /application/mysql/lib/libmysqlclient.so.18 /usr/lib64
touch ext/phar/phar.phar          #在 php5.3.27 目录下执行
make
```

(4) 安装

```
make install
```

(5) 设置软链接以方便访问：

```
ln -s /application/php5.3.27 /application/php
```

(6) 拷贝 PHP 配置文件到 PHP 默认目录，并更改文件名称为 php.ini

```
cp php.ini-production /application/php/lib/php.ini
```

(7) 配置 PHP 服务的配置文件 php-fpm.conf

```
cp /application/php/etc/php-fpm.conf.default /application/php/etc/php-fpm.conf
```

(8) 启动 PHP 服务

```
/application/php/sbin/php-fpm
```

(9) 检查 PHP 服务启动端口的情况：

```
ps -ef | grep php-fpm
```

(10) 在虚拟主机 server 里面增加下面这个 local，并在相应的 html 目录里增加 PHP 文件，即可解释 PHP 文件：

```
        location ~ .*\.(php|php5)$ {        #  .*表示匹配除回车之外的任意字符
            root   html/www;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        include        fastcgi.conf;
    }
```

注意，书本上那个 PHP 程序有错，不应该是左边这样子，应该是带问号的右边这样的。

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

16. 启动 php 服务的命令是安装目录里 sbin 文件夹的 php-fpm 脚本（启动 php 服务、php-fpm）

```
/application/php/sbin/php-fpm
```

17. 使用下列命令关闭 php:　　（pkill）

```
pkill php-fpm
```

18. 几个应用的启动脚本：

- nginx：

```
［root@www tools］# /application/nginx/sbin/nginx
```

- Mysql:　　（由 support-files 的 mysql.server 修改而来）

```
cp application/mysql/support-files/mysql.server /etc/init.d/mysqld
chmod ＋x /etc/init.d/mysqld
/etc/init.d/mysqld start
```

- PHP：

```
/application/php/sbin/php-fpm
```

19. 创建一个 wordpress 用户专属数据库 wordpress：

(1) 创建一个数据库，名字为 wordpress （创建数据库）

```
mysql>create database wordpress;
```

(2) 创建一个专用的 WordPress blog 管理用户，命令如下：（创建用户并授权）

```
mysql>grant all on wordpress.* to wordpress@'localhost' identified by '123456';
```

(3) 刷新权限，使得创建用户生效　　（刷新权限）

```
mysql>flush privileges;
```

(4) 查看用户对应的权限

```
mysql>show grants for wordpress@'localhost';
```

第 7 章 PHP 服务器缓存加速优化实战

1. 当客户端请求一个 PHP 程序时，服务器的 PHP 引擎会解析该 PHP 程序，并将其编译为特定的操作码文件。默认情况下，这个操作码文件由 PHP 引擎执行后丢弃。而缓存加速软件的原理就是将操作码文件保存下来，并放到共享内存里，以便在指定时间内有相同的 PHP 程序请求访问时，不再需要重复解析编译，而是直接调用缓存中的 PHP 操作码文件。（PHP 缓存、缓存操作码文件放到共享内存）

2. PHP 缓存加速器软件常见的各类有 APC、eAccelerator、XCache、ZendOpcache 等，任选其一即可，没必要都安装上。老男孩喜欢用 XCache。（PHP 缓存、a、e、x、z）
3. phpize 是用来扩展 PHP 扩展模块的，通过 phpize 可以建立 PHP 的外挂模块。比如想在原来编译好的 PHP 中加入 Memcached 等扩展模块，可以使用 phpize 工具。(phpize 扩展)
4. eAccelerator 是一个免费的、开放源码的 PHP 加速、优化及缓存的扩展插件软件。安装命令如下：(phpize、配置、编译、安装)　　（多个 phpize）

```
    tar -xvf eaccelerator-0.9.6.1.tar.bz2
    cd eaccelerator-0.9.6.1
    /application/php/bin/phpize
    ./configure --enable-eaccelerator=shared --with-php-
config=/application/php/bin/php-config
    make
    make install
    ls /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/
#最后生成 eaccelerator.so           （l、p、e、n）
```

5. xchche 的安装方法和 eaccelerator 类似：

```
    tar -xvf xcache-3.2.0.tar.bz2
    cd xcache-3.2.0
    /application/php/bin/phpize
    ./configure --enable-xcache --with-php-config=/application/php/bin/php-
config
    make
    make install
    ls /application/php5.3.27/lib/php/extensions/no-debug-non-zts-20090626/
#最后生成 xcache.so
```

6. eAccelerator、XCache，从 PHP 5.5 开始，官方已经集成了新一代的缓存加速插件，其名字为 ZendOpcache，功能和前三者相似但又有少许不同，据官方说，ZendOpcache 缓存速度更快。PHP 5.5 以上版本，支持 ZendOpcache 很简单，只需在编译 PHP 5.5 时加上--enable-opcache 就行了。5.5 以前的版本，ZendOpcache 作为独立的软件存在。5.5 版本以前的 ZendOpcache 安装方法和之前的缓存软件类似。（官方 ZendOpcache）
7. Memcached 是一个开源的、支持高性能、高并发及分布式的内存缓存服务软件，用来存储小块的任意数据（字符串、对象）。这些数据可以是数据库调用、API 调用或者是页面渲染的结果。Memcached 服务分为服务器端和客户端两部分，其中服务器端软

件的名字形如Memcached-1.4.13.tar.gz，客户端软件的名字形如Memcache-2.27.tar.gz。（Memcached、存储小块的任意数据）

8. PDO扩展为PHP访问数据库定义了一个轻量级一致性的接口，它提供了一个数据访问抽象层，这样，无论使用的是什么数据库，都可以通过一致的函数执行查询并获取数据。(PDO扩展、一致的数据库接口)

9. ImageMagick是一套功能强大、稳定而且免费的工具集和开发包，可以用来读、写和处理超过89种基本格式的图片文件，包括流行的tiff、jpeg、gif、png、pdf，以及PhotoCD等。利用ImageMagick，可以根据Web应用程序的需要动态生成图片，还可以对一个（或一组）图片进行改变大小、旋转、锐化、减色或增加特效等操作，并将操作的结果以相同格式或其他格式保存。安装非常简单，不再叙述。（ImageMagick处理图片）

10. imagick插件工作需要ImageMagick软件的支持，所以，必须要先安装ImageMagick，否则会报错。imagick插件是一个可以供PHP调用ImageMagick功能的扩展模块。使用这个扩展可以使PHP具备和ImageMagick相同的功能。（imagick插件调用ImageMagick处理图片）

11. 安装了ImageMagick图像程序后，再安装PHP的扩展imagick插件，才能使用ImageMagick提供的api进行图片的创建与修改、压缩等操作，因为它们都集成在imagick这个PHP扩展中。（既要安装ImageMagick，也要安装imagick插件）

12. 修改文件时养成检查习惯和备份习惯。

13. 使用tmpfs优化eAccelerator缓存目录。tmpfs是一种基于内存的文件系统，通常使用tmpfs作为数据临时存储，比本地磁盘存储快很多，此方法适用于临时使用的各类缓存场景。本书将/tmp/eaccelerator挂载到tmpfs文件系统上，让访问缓存的数据更快。具体操作方法如下：（tmpfs优化eAccelerator缓存目录）

```
    mount -t tmpfs -o size=16m tmpfs /tmp/eaccelerator          #<==创建16M大小的tmpfs类型文件系统挂载到/tmp/eaccelerator
    tail -1 /etc/fstab #<==配置永久挂载，生产场景size可以根据实际情况调整
tmpfs /tmp/eaccelerator tmpfs size=16m 0 0
    umount /tmp/eaccelerator/
    mount -a #<==测试永久挂载
```

14. 生产环境插件的安装建议：

(1) 对于功能性插件，如果业务产品不需要使用，可以暂时不考虑安装，例如PDO_MYSQL\memcache\imagick等。如果不清楚是否需要，最好还是装上，有备无患。

(2)　　对于性能优化插件，eAccelerator、XCache、ZendOpcache、APC 可以安装任一种，在选择时最好能搭建相关环境进行压力测试，然后根据实际测试结果来选择，用数据说话很重要。

15.　下面是针对 PHP 加速器比较结果进行的总结。

- 通过测试得出，eAccelerator 在请求时间和内存占用综合方面是最好的。
- 通过测试得出，使用加速器比无加速器在请求时间快了 3 倍左右。

16.　phpize 是用来扩展 PHP 扩展模块的，通过 phpize 可以建立 PHP 的外挂模块。编译 PHP 后，其 bin 目录下会有 phpize 这个脚本文件。在编译要添加的扩展模块之前，执行以下 phpize 就可以了。（phpize、扩展模块）

17.　可用压力测试软件 webbench、loadruner 来测试各种缓存软件的性能。（压力测试软件、webbench、loadruner）

第 8 章 企业级 Nginx Web 服务优化实战

1. 尽量隐藏或消除 Web 服务对访问用户显示各类敏感信息（例如 Web 软件名称及版本号等信息），这样恶意用户就很难猜到他攻击的服务器所用的是否有特定漏洞的软件，或者是否有对应漏洞的某一特定版本，从而加强 web 服务的安全性。

(1) 编辑 nginx.conf 配置文件增加参数，实现隐藏 Nginx 版本号的方式如下：

```
http
{
.....
server_tokens off;
.....
}
```

server_tokens 参数放置在 http 标签内，作用是控制 http response header 内的 web 服务版本信息的显示，以及错误信息中 Web 服务版本信息的显示。（server_tokens、隐藏 nginx 版本号）

(2) 通过更改源码隐藏 Nginx 软件名及版本号

- 第一个要修改的文件为nginx-1.6.3/src/core/nginx.h

  [root@oldboy core] # sed -n '13 , 17p' nginx.h

  #define NGINX_VERSION "1.6.3" #<==修改为想要显示的版本号，如2.2.23

  #define NGINX_VER "nginx/" NGINX_VERSION

  #<==将nginx修改为想要修改的软件名称，如OWS

  #define NGINX_VAR "NGINX" #<==将nginx修改为想要修改的软件名称，

  如OWS (Oldboy Web Server)

  #define NGX_OLDPID_EXT ".oldbin"

- 第二个要修改的文件为nginx-1.6.3/src/http/ngx_http_header_filter_module.c的第46行:

[root@oldboy http] # grep -n 'Server: nginx' ngx_http_header_filter_module.c

49: static char ngx_http_server_string [ ] = "Server: nginx " CRLF; #<==修改本行结尾的nginx。

```
[ root@oldboy http ] # sed -n '21 , 30p' ngx_http_special_response.c

static u_char ngx_http_error_full_tail [ ] =

"<hr> <center> " NGINX_VER "</center> " CRLF#<==此行需要修改

"</body> " CRLF

"</html> " CRLF ;

static u_char ngx_http_error_tail [ ] =

"<hr> <center> Nginx</center> " CRLF#<==此行需要修改

"</body> " CRLF
```

- 要修改的第三个文件是 nginx-1.6.3/src/http/ngx_http_special_response.c，对外页面报错时，它会控制是否展示敏感信息。这里输出修改前的内容：

将其修改为：

```
[ root@oldboy http ] # sed -n '21 , 30p' ngx_http_special_response.c

static u_char ngx_http_error_full_tail [ ] =

"<hr> <center> " NGINX_VER " （ http : //oldboy.blog.51cto.com ）</center> " CRLF#<==此行是定义对外展示的内容

"</body> " CRLF

"</html> " CRLF ;

static u_char ngx_http_error_tail [ ] =

"<hr> <center> OWS</center> " CRLF #<==此行将对外展示的Nginx名字更改为OWS

"</body> " CRLF
```

(3) 修改后重新编译，使其生效。

2. 为了使 web 服务更安全，要尽量可能地改掉软件默认的所有配置，包括端口、用户等。（更改软件默认的所有端口、用户等）
3. 说明：本章几乎全部优化都是在 nginx.conf 文件中完成。

4. user 指令语法:

```
user user [group]
```

指定 Nginx Worker 进程运行用户，默认是 nobody 帐号。示例：（user 指定进程运行用户）

```
user www users;
```

5. 为 Nginx 服务建立新用户：（useradd、nologin、-M 不建立用户目录）

useradd nginx -s /sbin/nologin -M

#<==不需要有系统登录权限，应当禁止其登录能力，相当于apache里的用户

id nginx #<==检查用户

(1) 配置 Nginx 服务，让其使用刚建立的 nginx 用户：

- 第一种为直接更改配置文件参数，将默认的#user nobody;改为如下内容：

user nginx nginx;

注意，默认的#user nobody 应该不起作用的，修改时需要把#号去掉才能起作用。

- 第二种方法为直接在编译 Nginx 软件时指定编译有用户和组，命令如下：

./configure ——user=nginx ——group=nginx ——prefix=/application/nginx1.6.3 ——with-http_stub_status_module ——with-http_ssl_module

(4) 检查更改用户的效果：

[root@oldboy conf] # ps -ef|grep nginx|grep -v grep

root 1428 1 0 09: 56 00: 00: 00 nginx: master process /

application/nginx/sbin/nginx

nginx 1610 1428 0 10: 03 00: 00: 00 nginx: worker process

nginx 1611 1428 0 10: 03 00: 00: 00 nginx: worker process

6. 优化 Nginx 服务的 worker 进程个数

(1) 优化 Nginx 进程对应的 Nginx 服务的配置参数如下：

```
worker_processes 1;  #<==指定了Nginx要开启的进程数，结尾的数字就是进程的个数
```

worker_processes 参数的大小最高设置和网站的用户数量相关联。新配置不知道网站的用户数量时可以等于 CPU 的核数，且 worker 进程数要多一些。（worker_processes、应该用户数量关联）

(2) 查看 linux 服务器 CPU 总核数的方法：（CPU 信息在/proc/cpuinfo 文件里）

```
[ oldboy@oldboy ~ ] $ grep processor /proc/cpuinfo|wc -l

4 #<==表示为1颗CPU四核

[ root@oldboy ~ ] # grep -c processor /proc/cpuinfo

4 #<==表示为1颗CPU四核
```

grep 参数：

－c：只输出匹配行的计数。

注意，参数-l 是 wc 命令的参数

7. 默认情况下，Nginx 的多个进程有可能跑到某一个 CPU 或者 CPU 的某一核上，导致 Nginx 进程使用硬件的资源不均。这时可以绑定不同的 Nginx 进程到不同的 CPU 上：（work_cpu_affinity、绑定 nginx 进程到不同的 cpu）

```
worker_processes 4;

worker_cpu_affinity 0001 0010 0100 1000;

#<==          worker_cpu_affinity就是配置Nginx进程与CPU亲和力的参数，即把不同的进程分给不同的CPU处理。这里0001  0010  0100  1000是掩码，分别代表第1、2、3、4核CPU，由于worker_processes进程数为4，因此，上述配置会把每个进程分配一核CPU处理，默认情况下进程不会绑定任何CPU，参数位置为main段。
```

8. Nginx 的连接处理机制在不同的操作系统会采用不同的 I/O 模型，在 linux 下，Nginx 使用 epoll 的 I/O 多路复用模型，在 Freebsd 中使用 kqueue 的 I/O 多路复用模型。手工设置 I/O 模型的示例如下：

```
events

#<==events指令是设定Nginx的工作模式及连接数上限

{

use epoll;

#<==use是一个事件模块指令，用来指定Nginx的工作模式。Nginx支持的工作模式有select、poll、
kqueue、epoll、rtsig和/dev/poll。其中select和pol都是标准的工作模式，kqueue和epol是高效的工作
模式，不同的是epoll用在Linux平台上，而kqueue用在BSD系统中。对于Linux系统Linux 2.6+的内核，推
荐选择epoll工作模式，这是高性能高并发的设置

}
```

根据 Nginx 的官方文档建议，不指定事件处理模型，Nginx 会自动选择最佳的事件处理模型服务。（官方建议不指定事件处理模型）

9. events 区块是一个用来设置连接进程的区块，例如：设置 Nginx 的网络 I/O 模型，以及连接数等。（event、连接进程）

对于使用连接进程的方法，通常不需要进行任何设置，Nginx 会自动选择最有效的方法。

10. 调整 Nginx 单个进程允许的客户端最大连接数：（worker_connections、单个进程最大连接数）

```
events #<==events指令是设定Nginx的工作模式及连接数上限

{

worker_connections 20480;

#<==worker_connections也是个事件模块指令，用于定义Nginx每个进程的最大连接数，默认是
1024。最大客户端连接数由worker_processes和worker_connections决定，即Max_client=
worker_processes*worker_connections。 进程的最大连接数受Linux系统进程的最大打开文件数限制，在
执行操作系统命令 "ulimit -HSn 65535" 或配置相应文件后，worker_connections的设置才能生效
```

11. 调整配置 Nginx worker 进程的最大打开文件数，这个控制连接数的参数为 worker_rlimit_nofile。该参数的实际配置如下：（worker_rlimit_nofile、最大打开文件）

```
worker_rlimit_nofile 65535;

#<==最大打开文件数，可设置为系统优化后的ulimit -HSn的结果，在第3章中，调整系统文件描述符
和这个问题有相同之处
```

12. 优化服务器域名的散列表大小，尽可能使用确切的名字。例如，如果使用 nginx.org 和 www.nginx.org 来访问服务器是最频繁的，那么将它们明确地定义出来就更为有效，命令如下：

```
server {

listen 80;

server_name nginx.org www.nginx.org *.nginx.org;

.....

}
```

如果定义了大量的名字，或者定义了非常长的名字，那就需要在 HTTP 配置块中调整 server_names_hash_max_size 和 server_names_hash_bucket_size 的值。（域名太长要修改 server_names_hash_max_size 和 server_names_hash_bucket_size）

server_names_hash_max_size：设置存放域名的最大散列表大小。（存储域名的散列表大小）

server_names_hash_bucket_size：设置存放域名（server names）的最大散列表的存储桶（bucket）的大小。默认值依赖 CPU 的缓存行。 （存储桶大小）

13. Nginx 连接超时的参数设置：

(1) 设置参数：keepalive_timeout 60;

用于设置客户端连接保持会话的超时时间为 60 秒。超过这个时间，服务器会关闭该连接，此数值为参考值。

(2) 设置参数：tcp_nodelay no;

用于激活 tcp_nodelay 功能，tcp_nodelay 为 on 时立即发送数据，不延迟；为 off 时，当数据量过少时会延迟发送，等待下一个数据组合成一个大一点的数据包。（为 on 时立即发送，不延迟数据）

(3) 设置参数：client_header_timeout 15;（请求头超时时间）

用于设置读取客户端请求头数据的超时时间，单位是秒。

(4) 设置参数：client_body_timeout 15;

用于设置读取客户端请求主体的超时时间，默认值是 60。（请求体超时时间）

(5) 设置参数：send_timeout 25;

用于指定客户响应端的超时时间。这个超时仅限于两个连接活动之间的时间，如果超过这个时间，客户端没有任何活动，Nginx 将会关闭连接，默认是 60 秒。（客户端响应超时时间）

14. 调整上传文件的大小：（上传文件的大小）

```
client_max_body_size 8m
```

设置上传文件大小为 8M。

15. Nginx gzip 压缩模块提供了压缩文件内容的功能，用户请求的内容在发送到用户客户端之前，Nginx 服务器实施压缩，以节约网站出口带宽。纯文本内容最好进行压缩，图片、视频等文件尽量不要压缩。参数介绍及配置说明：（文件压图片不压）

```
gzip on;

#<==开启gzip压缩功能。

gzip_min_length 1k;

#<==设置允许压缩的页面最小字节数，页面字节数从header头的Content-Length中获取。默认值是0，表示不管页面多大都进行压缩。建议设置成大于1K，如果小于1K可能会越压越大。

gzip_buffers 4 16k;

#<==压缩缓冲区大小。表示申请4个单位为16K的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。

gzip_comp_level 2;

#<==压缩比率。用来指定gzip压缩比，1压缩比最小，处理速度最快；9压缩比最大，传输速度快，但处理最慢，也比较消耗CPU资源。

gzip_types text/plain application/x-javascript text/css application/xml;

#<==用来指定压缩的类型，"text/html"类型总是会被压缩，这个就是HTTP原理部分讲的媒体类型。

gzip_vary on;

#<==vary header支持。该选项可以让前端的缓存服务器缓存经过gzip压缩的页面，例如用Squid缓存经过Nginx压缩的数据
```

16. 当用户第一次访问网站时，会把页面存储到用户浏览器本地，这样用户第二次及以后继续访问该网站时，就会加载缓存的内容。Nginx expires 的功能就是允许通过 Nginx 配置文件控制 HTTP 的“Expires”和“Cache-Control”响应头部内容，告诉客户端浏览器是否缓存和缓存多久以内访问的内容。配置 Nginx expires 的功能，以 location 标签为例进行讲解，如果针对所有内容设置缓存，也可以不用 location。（expires、设置缓存时间）

(1) 根据文件扩展名进行判断，添加 expires 功能范例

- 根据文件扩展名进行判断，添加 expires 功能范例：（文件名判断）

```
location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
{
expires 3650d;
}
```

当用户访问网站 URL 结尾的文件扩展名为上述指定类型的图片时，设置缓存 3650 天，即 10 年。

- 范例 2

```
location ~ .*\.(js|css)$
{
expires 30d;
}
```

当用户访问网站 URL 结尾的文件扩展名为 js、css 类型的元素时，设置缓存 30 天。

(2) 根据 URI 中的路径进行判断，添加 expires 范例：（URI 中的路径判断）

```
## Add expires header according to URI(path or dir).
location ~ ^/(images|javascript|js|css|flash|media|static)/{
expires 360d;
}
```

访问网站URL中包含上述路径时，把访问的内容设置缓存360天。

(3) 单个文件添加expires功能范例

```
location ~(robots.txt){
expires 7d;
break;
}
```

给robots.txt机器人文件设置过期时间为7天，在这7天并不记录404错误日志。

17. 编写脚本实现自动切割、轮询，详细过程如下：

(1) 配置日志切割脚本，命令如下：

```
[root@oldboy ~]# mkdir /server/scripts/ -p
[root@oldboy ~]# cd /server/scripts/
[root@oldboy scripts]# vim cut_nginx_log.sh
cd /application/nginx/logs &&\
/bin/mv www_access.log www_access_$(date +%F -d -1day).log #<==将日志按日期改成前一天的名称
/application/nginx/sbin/nginx -s reload #<==重新加载nginx使得触发重新生成访问日志文件提示：实际上脚本的功能很简单，就是改名日志，然后加载nginx，重新生成文件记录日志
```

&&：只有左边的命令成功时才执行右边的命令。

说明：其实就是改日志名，然后重新加载nginx，重新生成日志文件。

(2) 将这段脚本保存入到 服务器的定时任务配置里，让此脚本在每天凌晨 0 点执行，就可以实现日志的每天分割功能，操作如下：

```
[ root@oldboy scripts ] # crontab -e #<==打开编辑功能后，加入如下内容

#cut nginx access log by oldboy at 201409

00 00 * * * /bin/sh /server/scripts/cut_nginx_log.sh >/dev/null 2>&1
```

18. 不记录不需要的访问日志，具体配置方法如下：（不记录不需要的日志、access_log off）

```
location ~ .*\.(js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$ {

access_log off;

}
```

不记录图片、JS 等文件的访问。

19. nginx 的配置文件中可以使用的内置变量以美元符$开始。常见内置变量如下：

(1) $request_filename 当前连接请求的文件路径

```
If (!-f $request_filename){
    rewrite  (.*) /index.php;
}
```

(2) $request_method：这个变量是客户端请求的动作，通常为 GET 或 POST。

(3) $host：请求中的主机头(Host)字段，如果请求中的主机头不可用或者空，则为处理请求的 server 名称(处理请求的 server 的 server_name 指令的值)。值为小写，不包含端口。

20. 根据扩展名限制程序和文件访问：

(1) 配置 Nginx，禁止解析指定目录下的指定程序：（禁止解析、deny all）

```
location ~ ^/images/.*\.(php|php5|sh|pl|py)$
{
 deny all;
}
location ~ ^/static/.*\.(php|php5|sh|pl|py)$
{
 deny all;
}
location ~* ^/data/(attachment|avatar)/.*\.(php|php5)$
{
deny all;
}
```

对上述目录的限制必须写在Nginx处理PHP服务配置的前面，如下：（必须在PHP服务配置的前面）

```
location ~ .*\.(php|php5)$
{
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
include fcgi.conf;
}
```

(2) Nginx 下配置禁止访问*.txt 和*.doc 文件，实际配置信息如下：

```
location ~* \.(txt|doc)$ {

if (-f $request_filename) {

root /data/www/www;

#rewrite .....可以重定向到某个URL

break;

}

}

location ~* \.(txt|doc)${

root /data/www/www;

deny all;

}
```

21. 禁止访问指定目录下的所有文件和目录。

(1) 配置禁止访问指定的单个或多个目录。禁止访问单个目录的命令如下：

```
location ~ ^/(static)/{
deny all;
}
location ~ ^/static{
deny all;
}
```

禁止访问多个目录的命令如下：

```
location ~ ^/(static|js){
deny all;
}
```

(2) 禁止访问目录并返回指定的 HTTP 状态码，命令如下：（返回状态码、return）

```
server{
listen 80;
server_name www.etiantian.org etiantian.org;
root /data0/www/www;
index index.html index.htm;
access_log /app/logs/www_access.log commonlog;
location /admin/{return 404;}
location /templates/{return 403;}
}
```

作用：禁止访问目录下的指定文件，或者禁止访问指定目录下的所有内容。

22. 限制网站来源 IP 访问。（allow 允许、deny 禁止）

(1) 禁止某目录让外界访问，但允许某 IP 访问该目录，且支持 PHP 解析，命令如下：

```
location ~ ^/oldboy/ {
allow 202.111.12.211;
deny all;
}
location ~ .*\.(php|php5)$ {
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
include fastcgi_params;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
}
```

(2) 限制指定 IP 或 IP 段访问，命令如下：（限制、使用 IP 段的形式）

```
location / {

deny 192.168.1.1;

allow 192.168.1.0/24;

allow 10.1.1.0/16;

deny all;

}
```

注意：

- deny 一定要加一个 IP，否则会直接跳转到 403，不再往下执行了，如果 403 默认页是在同一域名下，会造成死循环。
- 以 deny all;结尾，表示除了上面允许的，其他的都禁止。

23. 配置 Nginx，禁止非法域名解析访问企业网站，也就是相当于直接使用 IP 访问网站。（非法解析、使用 IP 访问）

(1) 让使用 IP 访问网站的用户，或者恶意解析域名的用户，收到 501 错误，命令如下：

```
server {

listen 80 default_server;

server_name _;

return 501;

}
```

说明：直接报 501 错误，用户体验上不是很好。

(2) 通过 301 跳转到主页，命令如下：

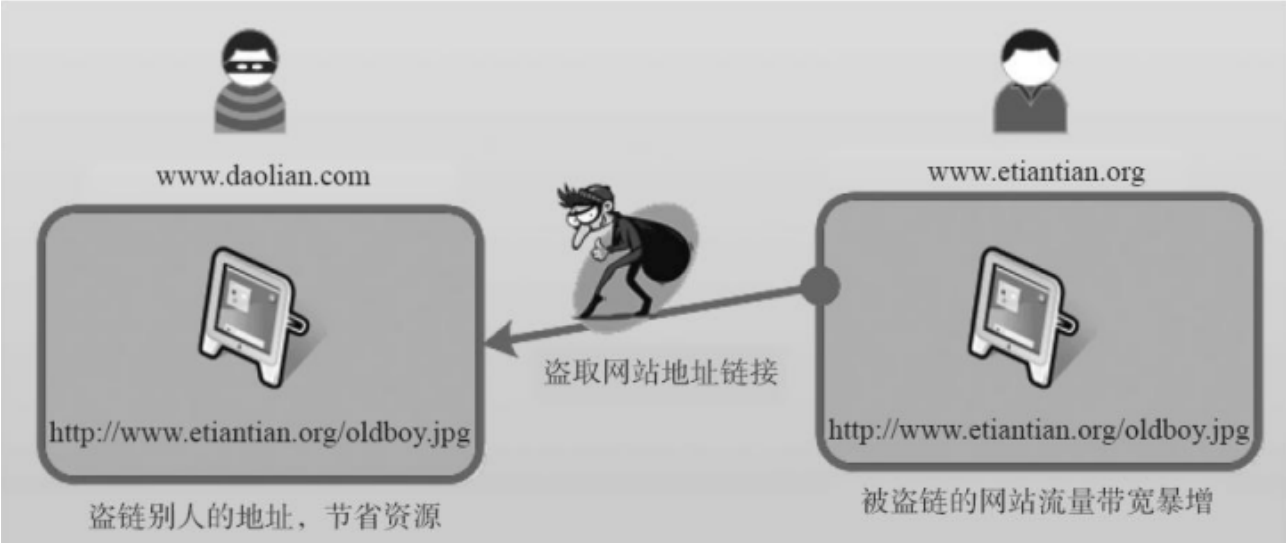
```
server {

listen 80 default_server;

server_name _ ;

rewrite ^(.*) http://blog.etiantian.org/$1 permanent;

}
```

(3) 发现某域名恶意解析到公司的服务器 IP，在 server 标签里添加以下代码即可，若有多个 server 则要多处添加。（恶意解析、$host）

```
if ( $host !~ ^www/.eduoldboy/.com$ ) {

rewrite ^(.*) http://www.eduoldboy.com$1 permanent;

}
```

上面代码的意思是如果 header 信息的 host 主机名字段非 www.eduoldboy.com，就 301 跳转到 www.eduoldboy.com。

24. 什么是资源盗链？简单地说，就是某些不法网站未经许可，通过其自身网站程序里非法调用其他网站的资源，然后在自己的网站上显示这些调用的资源，达到填充自身网站的效果。这一举动不仅浪费了调用资源网站的网络流量，还造成其他网站的带宽及服务压力吃紧。



25. 如何预防和发现盗链？（盗链、宽带监控、查看流量图、日志分析）

(1) 对 IDC 和 CDN 带宽做监控报警。

(2) 作为高级运维或运维经理，每天上班的重要任务，就是经常查看网站流量图，关注流量变化，关注异常流量。

(3) 对访问日志做分析，迅速定位异常流量，并且和公司市场推广等保持较好的沟通，以便调试带宽和服务器资源，确保网站正常的访问体验。

26. 常见防盗链原理：

(1) HTTP referer：在 HTTP 中，有一个表头字段叫 referer，使用 URL 格式来表示是哪里的链接用了当前的网页的资源。HTTP referer 是 header 的一部分，当浏览器向 Web 服务器发送请求时，一般会带上 referer，告诉服务器我是从哪个页面链接过来的，服务器借此获得一些用于处理的信息。Apache、Nginx、Lighttpd 三者都支持根据 HTTP referer 实现防盗链。

(2) cookie 防盗链：一些特殊的数据如流媒体并不向服务器提供 referer header，此时可以采用 cookie 技术，解决 Flash、Windows Media 视频等的防盗链问题。

(3) 通过加密变换访问路径实现防盗链：此种方法适合视频及下载类业务数据的网站。

(4) 如果不怕麻烦，有条件实现的话，推荐使用 NginxHttpAccessKeyModule 实现防盗链。

27. 指定错误页面：

```
error_page 404 /404.html;                #当出现 404 错误时，执行 404.html 页面
```

在 server、location 标签中执行。

也可以在后面接一个链接：

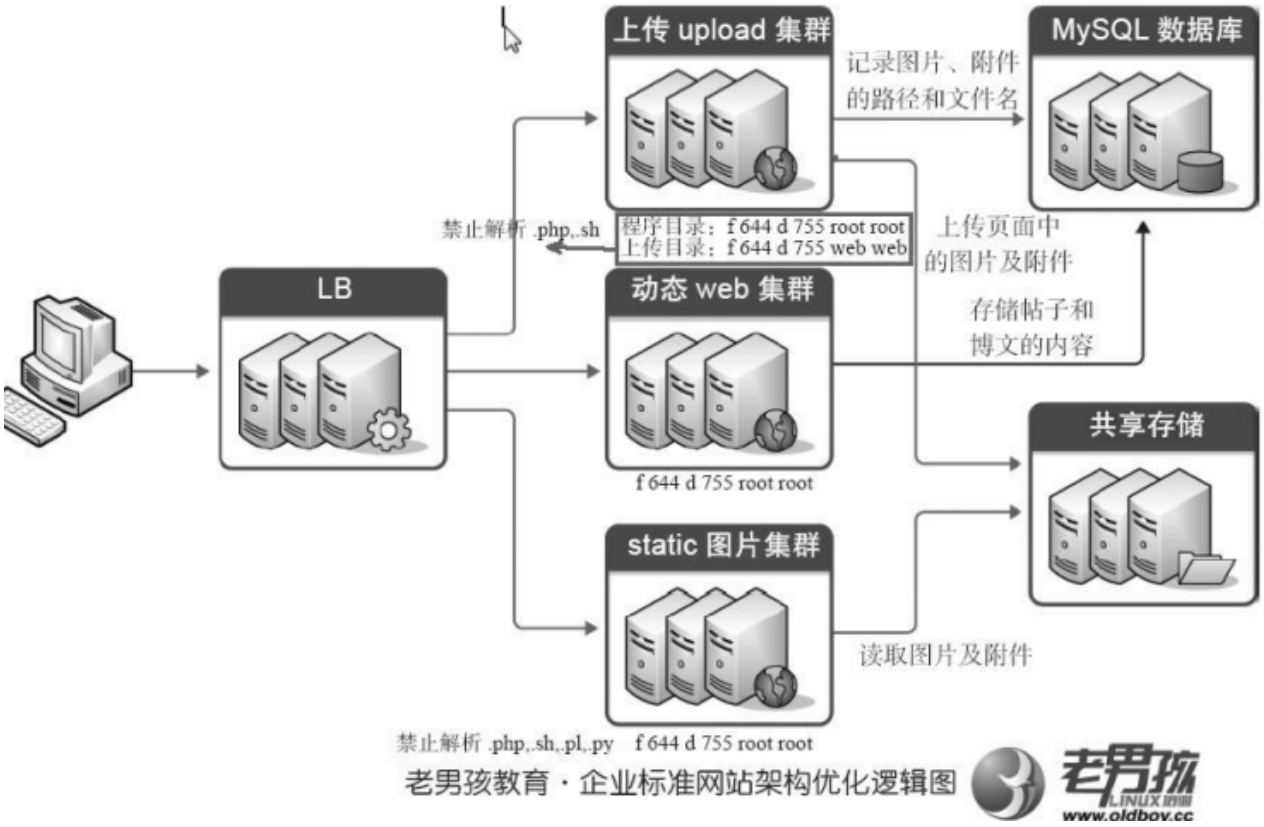
```
error_page 404 http：//oldboy.blog.51cto.com;
```

28. 为了保证网站不遭受木马入侵，所有站点目录的用户和组都应该为 root，所有目录权限是 755，所有文件权限是 644。但是，合理的网站用户上传的内容也会被拒之门外。那么如何让合法的用户可以上传文件，又不至于被黑客利用攻击呢？

- 如果是单机的 LNMP 环境，站点目录和文件属性设置如下：把用户上传资源的目录权限设置为 755，将用户和组设置为 Nginx 服务的用户，最后针对上传资源的目录做资源访问限制。（单机用户）
- 在比较好的网站业务架构中，应把资源文件，包括用户上传的图片、附近等服务和程序服务分离，最好把上传程序服务也分离出来，这样就可以从容地按照上文所述进行安全授权了。（分离）

29. 集群架构中不同于前面的Web业务的权限管理细化：（上传、动态Web、图片）

服务器角色	权限处理	安全系数
动态 Web 集群	目录权限 755，文件权限 644，所用的目录，以及文件用户和组都是 root。环境为 Nginx+PHP	文件不能被改，目录不能被写入，安全系数 10
static 图片集群	目录权限 755，文件权限 644，所用的目录，以及文件用户和组都是 root。环境为 Nginx	文件不能被改，目录不能被写入，安全系数 10
上传 upload 集群	目录权限 755，文件权限 644，所用的目录，以及文件用户和组都是 root。特别：用户上传的目录设置为 755，用户和组使用 Nginx 服务配置的用户	文件不能被改，目录不能被写入，但是用户上传的目录允许写入文件且需要通过 Nginx 的其他功能来禁止读文件，安全系数 8



30. Robots协议全称是“网络爬虫排除标准”，网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。

31. 可以通过Nginx限制HTTP请求的方法来达到提升服务器的安全目的，例如，让HTTP只能使用GET、HEAD和POST方法的配置如下：（限制HTTP请求、request_method）

```
#Only allow these request methods
if ( $request_method ! ~ ^( GET|HEAD|POST) $ ) {
return 501 ;
}
#Do not accept DELETE , SEARCH and other methods
```

32. CDN 的中文意思是内容分发网络。简单地讲，通过在现有的 Internet 中增加一层新的网络架构，将网站的内容发布到最接近用户的 Cache 服务器内，通过智能 DNS 负载均衡技术，判断用户的来源，让用户就近使用与服务器相同的线路的带宽访问 Cache 服务器，取得所需的内容。并不是所有的网站都可以一上来就能用 CDN，要加速的业务数据应该存在独立的域名，业务内容图片、附件、JS 等静态元素，这样的静态网站域名才可以使用 CDN。

33. nginx 分为 single 和 master 两种进程模型。master 模型分为一个 master 进程和 n 个 worker 进程的工作方式 。master 进程管理 worker 进程。（single 和 master 两种进程模型、一个 master 进程和 n 个 worker 进程）

34. 默认情况下，Nginx 的 Master 进程使用的是 root 用户。根据最小化分配权限原则，最好使用普通用户启动 Nginx，只要普通用户和 nginx 的所有者用户（非 root）同组即可管理 nginx，同时也避免了用户权限过大导致系统出问题。通过 Nginx 启动命令的-c 参数指定不同的 Nginx 配置文件，可以同时启动多个实例，并使用普通的用户运行服务。配置普通用户启动 Nginx 的过程如下。（-c 参数指定配置文件，可启动多个实例）

```
[root@www ~] # useradd inca
[root@www ~] # su - inca
[inca@www ~] $ mkdir conf logs www
[inca@www ~] $ cp /application/nginx/conf/mime.types ~/conf/  #在这里要将
[inca@www ~] $ echo inca >www/index.html
[inca@www ~] $ ps -ef|grep nginx|grep -v grep
[inca@www ~] $ /application/nginx/sbin/nginx -c /home/inca/conf/nginx.conf
```

基本思想：就是切换到相应的用户，复制 conf、log、www 目录到该用户根目录下，然后把~/conf/nginx.conf 文件中有目录的内容修改成用户目录，最后使用 nginx 的-c 参数接上该配置文件启动。

第 9 章 MySQL 数据库企业级应用实践

1. 访问及管理 MySQL 数据库的最常用标准化语言为 SQL 结构化查询语言。

2. MySQL 多实例就是在一台服务器上同时开启多个不同的服务器端口，同时运行多个 MySQL 服务器进程，这些服务进程通过不同的 socket 监听不同服务器端口来提供服务。这些 MySQL 多实例共用一套 MySQL 安装程序，使用不同的 my.cnf（也可以相同）配置文件、启动程序（也可以相同）和数据文件。（多实例、不同端口）

3. MySQL 多实例的生产应用场景：

- 资金紧张型公司的选择。
- 并发访问并不是特别大的业务。
- 门户网站应用 MySQL 多实例场景。

4. 为了让 MySQL 多实例之间彼此独立，要为每一个实例建立一个 my.cnf 配置文件和一个启动文件 MySQL，让它们分别对应自己的数据文件目录 data。（每一个实例有一个配置文件 my.cnf 和一个启动文件 MySQL）

5. MySQL 数据库支持单向、双向、链式级联、环状等不同业务场景的复制。在复制过程中，一台服务器充当主服务器（Master），接收来自用户的内容更新，而一个或多个其他的服务器充当从服务器（Slave），接收来自主服务器 binlog 文件的日志内容，解析出 SQL，重新更新到从服务器，使得主从服务器数据达到一致。如果设置了链式级联复制，那么，从服务器（Slave）本身除了充当从服务器外，也会同时充当其下面从服务器的主服务器。（主写从读）

6. 实现 MySQL 主从读写分离方案：

(1) 通过程序实现读写分离， 性能和效率最佳，推荐，缺点是需要开发人员对程序进行改造，使其对下层不透明，这种方式更容易开发和实现，适合互联网业务场景。

(2) 通过开源的软件实现读写分离。

(3) 大型门户独立开发 DAL 层综合软件。

7. 要实现 MySQL 主从复制，首先必须打开 Master 端的 binlog 记录功能，因为整个复制过程实际上就是 Slave 从 Master 端获取 binlog 日志，然后再在 Slave 上以相同顺序执行获取的 binlog 日志中记录的各种 SQL 操作。要打开 MySQL 的 binlog 记录功能，可通过在 MySQL 配置文件 my.cnf 中的 mysqld 模块 fulk“log-bin”参数实现。具体信息如下： （主从复制，打开 log-bin）

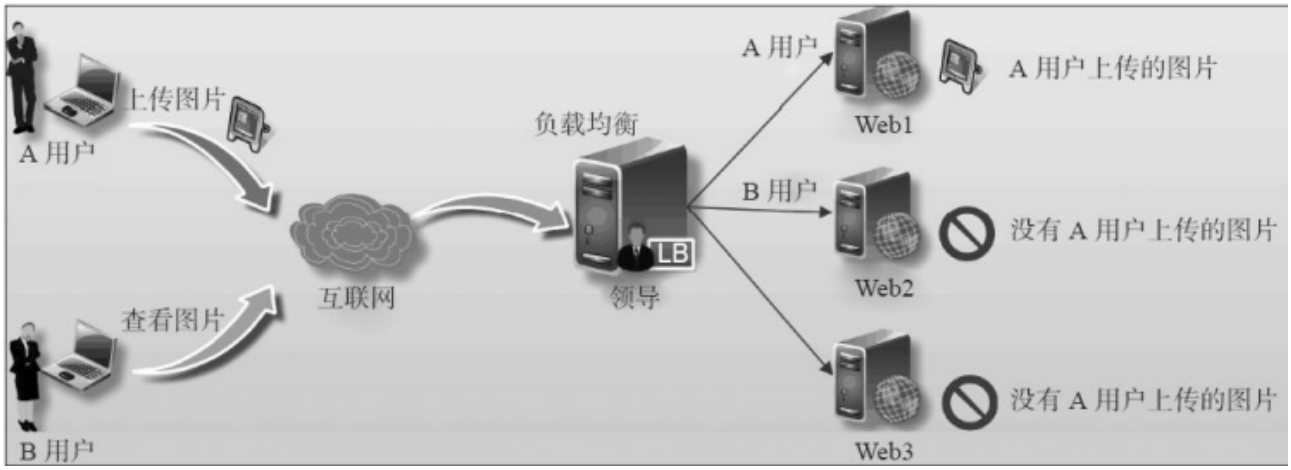
```
[ mysqld ]

log-bin = /data/3306/mysql-bin
```

8. 配置多实例权限时应该使用 700，不要使用 755，因为启动文件里有数据库管理员的密码。

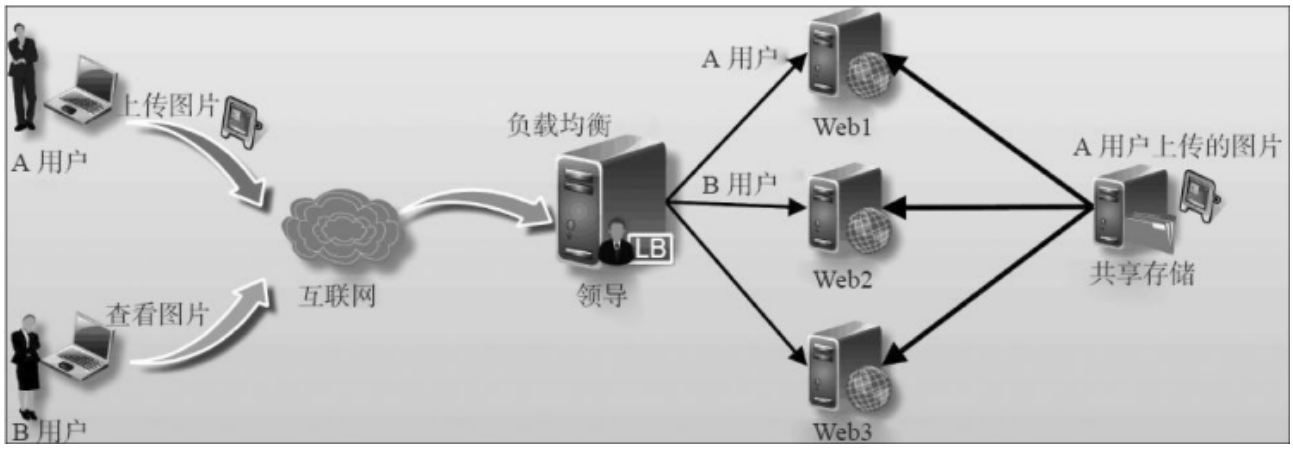
第 10 章 企业级 NFS 网络文件共享服务

1. NFS 在企业集群中的应用：

(1) 企业生产集群没有 NFS 共享存储访问示意图



(2) 企业生产集群有 NFS 共享存储访问示意图（NFS、共享存储）



2. NFS 在传输数据时使用的端口会随机选择，NFS 客户端通过 RPC 服务获取 NFS 服务器端使用的端口。NFS 的 RPC 服务，在 CentOS 5.x 下名称为 portmap，在 CentOS 6.x 下名称为 rpcbind。

3. 要部署 NFS 服务，需要安装下面的软件包：（nfs 两个软件包、nfs-utils 和 rpcbind）

- nfs-utils：NFS 服务的主程序，包括 rpc.nfsd、rpc.mountd 这两个 daemons 和相关的文档说明。

- rpcbind：RPC 服务主程序。RPC 服务最主要的功能就是指定每个 NFS 功能所对应的端口号，并且传递该信息给客户端，让客户端可以连接到正确的端口上去。（RPC、指定 NFS 所对应的端口号）

4. 使用 yum 来安装 NFS 软件包：

```
[root@lan data]# yum install nfs-utils rpcbind -y
[root@lan data]# rpm -qa nfs-utils rpcbind
```

RPC 默认安装为服务，即使用/etc/init.d/rpcbind start 启动

5. 检查 RPC 状态及启动 RPC：

```
[root@lan data]#  /etc/init.d/rpcbind status
[root@lan data]# /etc/init.d/rpcbind start
```

使用下列命令来查看注册的端口：（rpcinfo -p、查看注册的端口）

```
[root@lan data]# rpcinfo -p localhost
```

6. 要说明的是，在老男孩曾工作的多家大型企业里，大都是统一按照运维规范将服务的启动命令放到/etc/rc.local 文件里的，而不是用 chkconfig 管理的。
   把/etc/rc.local 文件作为本机的重要服务档案文件，所有服务的开机自启动都必须放入/etc/rc.local。这样规范的好处是，一旦管理此服务器的人员离职，或者业务迁移都可以通过/etc/rc.local 很容易地查看到服务器对应的相关服务，可以方便运维管理。下面是把启动命令放入到/etc/rc.local 文件中的配置信息，注意别忘了加上启动服务的注释。（将启动命令写入/etc/rc.local）

```
［root@nfs-server ～］# tail -3 /etc/rc.local
#start up nfs service by oldboy at 20150613
/etc/init.d/rpcbind start
/etc/init.d/nfs start
```

7. NFS 服务的默认配置文件路径为：/etc/exports，并且默认为空，需要自己添加内容。（NFS 默认配置文件、/etc/exports）

8. 配置文件/etc/exports 文件配置格式为：（export、目、地）

```
NFS 共享的目录  NFS 客户端地址 1（参 1，参 2.....）  客户端地址 2（参 1，参 2.....）
或者
NFS 共享的目录  NFS 客户端地址 1（参 1，参 2.....）
```

示例：

```
[root@lan data]# tail -2 /etc/exportfs
/data 10.0.0.24(rw,sync)      #在/etc/exports 中添加入这一句
```

9. NFS 服务器端设置：（启动、开机、用户组、配置、挂载）(启、开、用、配、测)

(1) 启动NFS服务，然后加入开机自启动：

```
[root@lan data]# /etc/init.d/rpcbind start
[root@lan data]# chkconfig rpcbind on
[root@lan data]# chkconfig nfs on
```

上述两个chkconfig命令也可以替换成在/etc/rc.local下输入以下命令：

```
/etc/init.d/rpcbind start
/etc/init.d/nfs start
```

chkconfig和rc.local二选一即可。

(2) 创建共享目录并修改目录的用户组：

```
[root@lan data]# mkdir -p /data
[root@lan data]# chown -R nfsnobody.nfsnobody /data
```

(3) 配置NFS服务的配置文件：

```
[root@lan data]# tail -2 /etc/exportfs
/data 10.0.0.24(rw,sync)      #在/etc/exports中添加入这一句
[root@lan data]# exportfs -rv
[root@lan data]# showmount -e localhost    #查看服务器本地挂载情况
```

每次修改/etc/exports后，需要执行/etc/init.d/nfs reload或exportfs -rv重新加载配置，但不需要重启NFS。（rv重新加载）

(4) 在客户端进行挂载测试（前提是客户端已经安装rpcbind）：

```
[root@lan data]# mount -t nfs 10.0.0.7:/data /mnt
```

注意：10.0.0.7是服务器IP

10. 使用showmount -e命令可以查看指定IP的共享目录：（showmount -e、查看指定IP的共享目录）

```
[root@lan data]# showmount -e 10.0.0.7     #查看10.0.0.7上的共享目录
```

11. 客户端获取共享目录要执行的操作：（客户端不用装NFS）

(1) 安装rpcbind：

```
[root@lan data]# yum install rpcbind -y
```

为了使用showmount等功能，所有客户端最好也要安装NFS软件，但不启动服务：

```
[root@lan data]# yum install nfs-utils -y
```

(2) 启动RPC服务（注意，无须启动NFS服务）

```
[root@lan data]# /etc/init.d/rpcbind start
[root@lan data]# showmount -e 10.0.0.7     #查看服务器上的共享目录
```

(3)　执行挂载：

```
[root@lan data]# mount -t nfs 10.0.0.7:/data /mnt     #执行挂载命令
```

(4)　将 rpcbind 服务和挂载加入开机自启动：

```
[root@lan data]# echo “/etc/init.d/rpcbind start” >> /etc/rc.local
[root@lan data]# echo “/bin/mount -t nfs 10.0.0.7:/data /mnt” >>
/etc/rc.local
```

12.　挂载命令：（挂载命令、mount -t nfs）

```
[root@lan data]# mount -t nfs 10.0.0.7:/data /mnt     #执行挂载命令
```

将目录 ip 的 data 目录加载到本地的/mnt 目录。

13.　当多个客户端访问服务器端的读写文件时，需要具有以下几个权限：（需要开放可写入权限、UID 65534 的 nfsnobody 用户）

- NFS 服务器/etc/exports 设置需要开放可写入的权限，即服务器端的共享权限。
- NFS 服务器实际要共享的 NFS 目录权限具有可写入 w 的权限，即服务器端本地目录的安全权限。
- 每台机器都对应存在和 NFS 默认配置 UID 的相同的 UID 65534 的 nfsnobody 用户（确保所有客户端的访问权限统一，否则每个机器需要同时建立相同的 UID 的用户，并覆盖 NFS 的默认用户配置）。

只有满足上述三个条件，多个 NFS 客户端才能具有查看、修改、删除其他任意 NFS 客户端上传文件的权限。

14.　NFS 客户端挂载的命令格式：

挂载命令	挂载的格式类型	NFS 服务器端提供的共享目录	NFS 客户端的挂载点
mount	-t nfs	10.0.0.7:/data	/mnt（必须存在）
完整挂载命令为：mount -t nfs 10.0.0.7:/data /mnt，此命令要在客户端执行			

15. 配置客户端 mount 挂载命令使挂载开机自动执行，这里有两种方法：（mount 挂载命令、开机自动执行、rc.local、/etc/fstab）

(1)　将挂载命令放在/etc/rc.local 里。缺点是偶尔开机挂载不上。

(2) 将挂载命令放在/etc/fstab 里。缺点是网络没启动时执行 fstab 会导致连不上 NFS 服务器端，无法实现开机挂载。

说明：程序启动时先加载/etc/fastab，该文件负责配置 linux 开机时自动挂载的分区。

16. showmount 命令一般用于从 NFS 客户端检查 NFS 服务器端共享目录的情况，常用参数说明如下：

短格式	长格式	用途及实例结果
-e	--exports	显示 NFS 服务器输出的目录列表： [root@nfs-client ~]# showmount -e 10.0.0.14 Export list for 10.0.0.14: /oldboy 10.0.0.0/24 [root@nfs-client ~]# showmount --exports 10.0.0.14 Export list for 10.0.0.14: /oldboy 10.0.0.0/24
-d	--directories	显示 NFS 服务器中提供共享的目录： [root@nfs-client ~]# showmount -d 10.0.0.14 Directories on 10.0.0.14: /oldboy
-a	--all	以 ip:/dir 格式显示 NFS 服务器的 IP 地址和可被挂载的目录： [root@nfs-client ~]# showmount -a 10.0.0.14 **All mount points on 10.0.0.14:** **10.0.0.7:/data**
更多命令帮助请执行 showmount --help 或 man showmount		

17. exportfs -rv 命令相当于/etc/init.d/nfs reload，该带参数的命令用于使新加载的配置生效，除此之外，通过 exportfs 命令，我们还可以管理当前 NFS 共享的文件系统目录列表。

18. nfs 服务器端需要对所有客户端权限统一，默认使用匿名用户 nfsnobody，但 centos5.5 系统的 NFS 有个 BUG，没有 UID 为 65534 的 nfsnobody 用户，更新 ntf-utils 版本可解决此问题。

19. 修改共享用户 UID：（-u、-g、anonuid、anongid）

(1) 建立用户组 zuma 并指定 GID 888，所有客户端也要执行同样的命令建立：

```
[root@lan data]# groupadd zuma -g 888
```

(2) 建立用户 zuma 并指定 UID 888，所有客户端也要执行同样的命令建立：

```
[root@lan data]# useradd zuma -u 888 -g 888
```

(3) 在/etc/exports 文件的参数中指定 UID 和 GID

```
[root@lan data]# echo  '/oldboy
10.0.0.0/24(rw,sync,all_squash,anonuid=888,anongid=888)'  >> /etc/exports
```

第 11 章 Nginx 反射代理与负载均衡应用实践

1. 集群就是指一组相互独立的计算机，利用调整通信网络组成一个圈套的计算服务系统，每个集群节点（指集群中的每台计算机）都是运行各自服务的独立服务器。这些服务器之间可以彼此通信，协同向用户提供应用程序、系统资源和数据，并以单一系统的模式加以管理。当用户客户机请求集群系统时，集群给用户的感觉就是一个单一独立的服务器，而实际上用户请求的是一组集群服务器。（集群、协同提供）
2. 只有当并发或总请求数量超过单台服务器的承受能力时，服务器集群才会体现出优势。
3. 计算机集群架构按功能和结构可以分成以下几类：（负、高、高、网）

- 负载均衡集群，简称 LBC 或者 LB。
- 高可用性集群，简称 HAC。
- 高性能计算集群，简称 HPC。
- 网格计算集群，很少使用。

负载均衡集群和高可用性集群是互联网行业常用的集群架构模式。

4. 负载均衡集群为企业提供了更为实用、性价比更高的系统架构解决方案。负载均衡集群可以把很多客户集中的访问请求负载压力尽可能平均地分摊在计算机集群中处理。这样的系统非常适合使用同一组应用程序为大量用户提供服务的模式。（负载平均）
5. 高可用性集群一般是指在集群中任意一个节点失效的情况下，该节点上的所有任务会自动转移到其他正常的节点上。（可用失效）
6. 高性能计算集群也称并行计算，通常，高性能计算集群涉及为集群开发的并行应用程序，以解决复杂的科学问题（天气预报、核反应模拟等）。
7. 互联网企业常用的开源集群软件
   有:Haproxy、Heartbeat、Keepalived、LVS、Nginx。　（H、H、K、L、N）

- Haproxy：是一个使用 C 语言编写的自由及开放源代码软件，其提供高可用性、负载均衡，以及基于 TCP 和 HTTP 的应用程序代理。　（高可用、负载）
- Heartbeat：Heartbeat 项目是 Linux-HA 工程的一个组成部分，它实现了一个高可用集群系统。　　（高可用）
- Keepalived：是 Linux 下一个轻量级别的高可用解决方案。　　（高可用）
- LVS：Linux Virtual Server 的简称，也就是 Linux 虚拟服务器，是一个由章文嵩博士发起的自由软件项目。通过 LVS 提供的负载均衡技术和 Linux 操作系统实现一个高

性能、高可用的服务器群集，它具有良好可靠性、可扩展性和可操作性。从而以低廉的成本实现最优的服务性能。　（高性能、高可用）

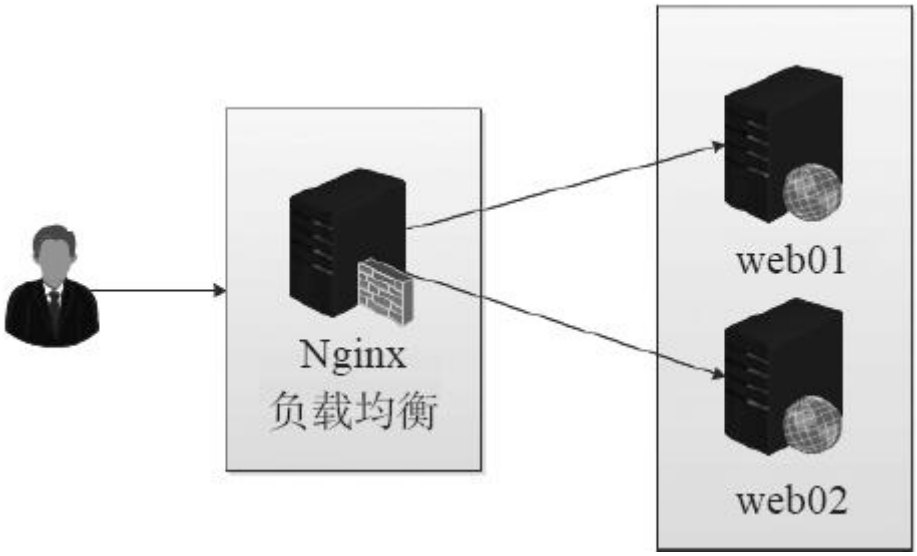
8. 传统的负载均衡是转发用户请求的数据包，而 Nginx 反向代理是接收用户的请求，然后重新发起请求却说请求其后面的节点。（传统转发、反向代理重发）（传转反重）

9. 实现 Nginx 负载均衡的组件主要有两个：（proxy 代理、upstream 负载均衡）

Nginx http 功能模块	模块说明
ngx_http_proxy_module	proxy 代理模块，用于把请求后抛给服务器节点或 upstream 服务器池
ngx_http_upstream_module	负载均衡模块，可以实现网站的负载均衡功能及节点的健康检查

upstream 模块使用 proxy 模块来实现负载均衡功能，而 proxy 模块代理方式中的其中一种。

10. 下图是 Nginx 负载均衡器的逻辑架构图，所有用户的请求统一发送给 Nginx 负载均衡器，然后由负载均衡器根据调试算法来请求 web01 和 web02：



11. ngx_http_upstream_module 模块允许 nginx 定义一组或多组节点服务器组，使用时可以通过 proxy_pass 代理方式把网站的请求发送到事先定义好的对应 Upstream 组的名字上，具体写法为“proxy_pass http://www_server_pools”，其中 www_server_pools 就是一个 Upstream 节点服务器组名字。

12. upstream 配置案例：（www_server_pools 为节点服务器组名字）

```
upstream www_server_pools{
    server 10.0.0.17:80 weight=5;
    server 10.0.0.18:80 weight=10;
}
```

说明：

- upstream 是关键字，必须要有，后面的 www_server_pools 为一个 upstream 集群组的名字，可以自己起名。
- server 关键字是固定的，后面可以接域名或 IP。可以不指定端口，默认端口为 80。weight 代表权重，默认值为 1，数值越大被分配的请求越多。（weight 越大，被分配的请求越多）
- server 后面如果接域名，需要内网有 DNS 服务器或者在负载均衡器的 hosts 文件做域名解析。

13. upstream 模块的内容应该放于 nginx.conf 配置的 http{}标签内，默认调试节点算法是 wrr(即权重轮询)。upstream 模块内部 server 标签部分参数说明：（upstream、应该放于 http 标签内）

upstream 模块内参数	参数说明
server 10.0.10.8:80	负载均衡后面的 RS 配置，可以是 IP 或域名，如果端口不写，默认是 80 端口。高并发场景下，IP 可换成域名，通过 DNS 做负载均衡
weight=1	代表服务器的权重，默认值是 1。权重数字越大表示接受的请求比例越大
max_fails=1	Nginx 尝试连接后端主机失败的次数，这个数值是配合 proxy_next_upstream、fastcgi_next_upstream 和 memcached_next_upstream 这三个参数来使用的，当 Nginx 接收后端服务器返回这三个参数定义的状态码时，会将这个请求转发给正常工作的后端服务器，例如 404、502、503。Max_fails 的默认值是 1；**企业场景下建议 2 ~ 3 次。如京东 1 次，蓝汛 10 次**，根据业务需求去配置
backup	热备配置（RS 节点的高可用），当前面激活的 RS 都失败后会自动启用热备 RS。这标志着这个服务器作为备份服务器，若主服务器全部宕机了，就会向它转发请求；注意，当负载调度算法为 ip_hash 时，后端服务器在负载均衡调度中的状态不能是 weight 和 backup
fail_timeout=10s	在 max_fails 定义的失败次数后，距离下次检查的间隔时间，默认是 10s；如果 max_fails 是 5，它就检测 5 次，如果 5 次都是 502。那么，它就会根据 fail_timeout 的值，等待 10s 再去检查，还是只检查一次，如果持续 502，在不重新加载 Nginx 配置的情况下，每隔 10s 都只检测一次。常规业务 2 ~ 3 秒比较合理，比如京东 3 秒，蓝汛 3 秒，可根据业务需求去配置
down	这标志着服务器永远不可用，这个参数可配合 ip_hash 使用

- weight：服务器权重。
- max_fails：连接失败的次数。　　（max_fails 次数）
- backup：备用服务器。　　（backup 备用）
- fail_timeout=10s：在 max_fails 定义的失败次数后，距离下次检查的间隔时间，默认 10 秒。　　（fail_timeout 间隔时间）

示例：

```
upstream www_server_pools{
    server 10.0.0.5;                #使用默认参数
    server 10.0.0.6:80 weight=10;
    server 10.0.0.7:80 backup;    #备份服务器，上面两个服务器不可用时，会启用这个
```

```
}
```

14. 调试算法一般分为两类：

- 第一类为静态调试算法，即负载均衡器根据自身设定的规则分配，不需要考虑后端节点服务器的情况。例如，rr、wrr、ip_hash 都属于静态调度算法。（静态不考虑）
- 第二类为动态调度算法，即负载均衡器会根据后端节点的当前状态来决定是否分发请求，如连接少的优先获得请求，响应时间短的优先获得请求。例如：least_conn、fair 等都属于动态调度算法。 （动态后端）

下面介绍常见的调度算法：

- rr 轮询（默认调度算法，静态调度算法）：按客户端请求顺序把客户端的请求逐一分配支不同的后端节点服务器。（rr 逐一）
- wrr 轮询：在 rr 轮询算法的基础上加上权重，即为权重轮询算法，当使用该算法时，权重和用户访问成正比，权重值越大，被转发的请求也就越多。（wrr 权重）

```
upstream www_server_pools{
    server 10.0.0.17:80 weight=5;
    server 10.0.0.18:80 weight=10;
}
```

- ip_hash（静态调度算法）：每个请求按客户端 IP 的 hash 结果分配，当新的请求到达时，先将其客户端 IP 通过哈希算法哈希出一个值，在随后的客户端请求中，客户 IP 的哈希值只要相同，就会分配到同一台服务器。

```
upstream www_server_pools{
    ip_hash;
    server 10.0.0.17:80;
    server 10.0.0.18:80;
}
```

- fair（动态调度算法）：根据后端节点服务器的响应时间来分配请求，响应时间短的优先分配。

```
upstream www_server_pools{
    fair;
    server 10.0.0.17:80 weight=5;
    server 10.0.0.18:80 weight=10;
}
```

- least_conn：根据后端节点的连接数来决定分配情况，哪个机器连接数少就分发。

- url_hash 算法：与 ip_hash 类似，这里是根据访问 URL 的 hash 结果来分配请求的，让每个 URL 定向到同一个后端服务器，后端服务器为缓存服务器时效果显著。示例配置如下：

```
upstream oldboy_lb{
      server squid1:3128;
      server squid2:3128;
      hash $request_uri;
      hash_method crc32;
}
```

- 一致性 hash 算法：用于代理后端业务为缓存服务的场景，通过将用户请求的 URI 或者指定字符串进行计算，然后调度到后端服务器上，此后任何用户查到同一个 URI 或者指定字符串都会被调度到这一台服务器上，因此后端的每个节点缓存的内容都是不同的。

15. proxy_pass 指令属于 ngx_http_proxy_module 模块，此模块可以将请求转发到另一台服务器，在实际的反向代理工作中，会通过 location 功能匹配指定的 URI，然后把接收到的符合匹配 URI 的请求通过 proxy_pass 抛给定义好的 upstream 节点池。下面是 proxy_pass 使用案例：　　（proxy_pass 转发）

- 将匹配 URI 为 name 的请求抛给 http://127.0.0.1/remote/:

```
location /name/{
      proxy_pass http://127.0.0.1/remote/;
}
```

- 将匹配URI为some/path的请求抛给http://127.0.0.1:

```
location /some/path/{
    proxy_pass http://127.0.0.1/;
}
```

- 将URI为name的请求应用指定的rewrite规则，然后抛给http://127.0.0.1：

```
location /name/{
    rewrite /name/ ([^/]+) /username=$1 break;
    proxy_pass http://127.0.0.1;
}
```

16. nginx的代理功能是通过http proxy模块来实现的，默认在安装nginx时已经安装了http proxy模块，因此可以直接使用http proxy模块。下面详细解释模块中每个选项代表的含义：

http proxy模块相关参数	说　明
proxy_set_header	设置http请求header项传给后端服务器节点，例如：可实现让代理后端的服务器节点获取访问客户端用户的真实IP地址
client_body_buffer_size	用于指定客户端请求主体缓冲区大小，此处如了解前文的http请求包的原理就好理解了
proxy_connect_timeout	表示反向代理与后端节点服务器连接的超时时间，即发起握手等候响应的超时时间

（续）

http proxy模块相关参数	说　明
proxy_send_timeout	表示代理后端服务器的数据回传时间，即在规定时间之内后端服务器必须传完所有的数据，否则，Nginx将断开这个连接
proxy_read_timeout	设置Nginx从代理的后端服务器获取信息的时间，表示连接建立成功后，Nginx等待后端服务器的响应时间，其实是Nginx已经进入后端的排队之中等候处理的时间
proxy_buffer_size	设置缓冲区大小，默认该缓冲区大小等于指令proxy_buffers设置的大小
proxy_buffers	设置缓冲区的数量和大小。Nginx从代理的后端服务器获取的响应信息，会放置到缓冲区
proxy_busy_buffers_size	用于设置系统很忙时可以使用的proxy_buffers大小，官方推荐的大小为proxy_buffers*2
proxy_temp_file_write_size	指定proxy缓存临时文件的大小

17. 淘宝技术团队开发了一个Tengine(Nginx的分支)模块nginx_upstream_check_module，用于提供主动式后端服务器健康检查。通过它可以检测后端realserver的健康状态，如果后端realserver不可用，则所有的请求就不会转发到该节点上。Tengine原生支持这个模块，而nginx则需要通过打补丁的方式将该模块添加到Nginx中。（Tengine健康检查）

18. 当Nginx接收到后端服务器返回proxy_next_upstream参数定义的状态码时，会将这个请求转发给正常工作的后端服务器，例如：500、502、503、504，此参数可以提升用户的访问体验，具体配置如下：

```
server{
      listen 80;
      server_name www.etiantian.org;
      location /{
            proxy_pass http://static_pools;
            proxy_next_upstream error timeout invalid_header http_500 http_502 http_503
http_504;
            include proxy.conf;
}
}
```

第12章 Keepalived高可用性集群应用实践

1. Keepalived起初是专为LVS负载均衡软件(LVS是一种负载均衡软件)设计的，用来管理并LVS集群系统中各个服务节点的状态，后来又加入了可以实现高可用的VRRP功能，因此，Keepalived除了能够管理LVS软件外，还可以作为其他服务（如Nginx、Haproxy等）的高可用解决方案软件。

2. Keepalived服务的三个重要功能：

- 管理LVS负载均衡软件。
- 实现对LVS集群节点健康检查功能，当某个节点服务器发生故障时，自动将失效节点从LVS的正常转发队列中清除出去。
- 作为系统网络服务的高可用功能。

3. Keepalived高可用功能实现的简单原理为，两台主机同时安装好Keepalived并启动服务，正常工作时，由角色为Master的主机获得所有资源并对用户提供服务，角色为Backup的主机作为热备。当Master主机失效或故障时，Backup主机自动接管Master的所有工作，待Master故障修复后，又会自动接管回它原来处理的工作。（热备接管）

4. Keepalived高可用服务对之间的故障切换转移是通过VRRP来实现的。

在Keepalived服务正常工作时，主Master节点会不断地向备节点发送（多播的方式）心跳消息，用以告诉备Backup节点自己还活着，当主Master节点发生故障时，就无法发送心跳消息，备节点也就因此无法继续检测到来自主Master节点的心跳了，于是调用自身的接管程序，接管主Master节点的IP资源及服务，而当主Master节点恢复时，备Backup节点又会释放主节点故障时自身接管的IP资源和服务，恢复到原来的备用角色。

5. VRRP中文名为虚拟路由冗余协议，VRRP的出现是为了解决静态路由的单点故障问题。在一组VRRP路由集群中，有多台物理VRRP路由器，但是这多台物理的机器并不是同

时工作的，而是由一台称为Master的机器负责路由工作，其他的机器都是Backup。Master角色并非一成不变的，VRRP会让每个VRRP路由参与竞选，最终获胜的就是Master。VRRP通过竞选机制实现虚拟路由器功能，所有协议报文都是通过IP多播包形式发送的。　　（VRRP竞选）

6. 什么是裂脑？由于某些原因，导致两台高可用服务器对在指定时间内，无法检测到对方的心跳消息，各自取得资源及服务的所有权，而此时的两台高可用服务器对都还活着并正常运行，这样就会导致同一个IP或服务在两端同时存在而发生冲突，最严重的是两台主机占用同一个VIP地址，当用户写入数据时可能会分别写入两端，这可能会导致服务器两端数据不一致或造成数据丢失，这种情况称为裂脑。　　（裂脑、无法检测）

7. 解决Keepalived裂脑的常见方案：

- 如果开启防火墙，一定要让心跳消息通过，一般通过允许IP段的形式解决。
- 可以拉一条以太网线或者串口线作为主被节点心跳线路的冗余。
- 开发监测程序通过监控软件监测裂脑。

8. Keepalived裂脑脚本检测思路：在备节点上执行脚本，如果可以ping通主节点并且备节点有VIP就报警，让人员介入检查是否裂脑。

第13章 企业级Memcached服务应用实践

1. Memcached是一套仅利用系统内存进行数据缓存的软件，常用于在动态Web集群系统后端、数据库前端等处，可临时缓存Web系统查询过的数据库数据，当用户请求查询数据时，由Memcached优先提供服务，从而减少Web系统直接请求数据库的次数，这极大地降低了后端数据库的压力，也因此提升了网站系统的性能。

2. Memcached服务分为服务器端和客户端，其中，服务器端软件的名字形如Memcached-1.4.24.tar.gz，客户端软件名字形如Memcache-2.25.tar.gz。（d服无客）

3. 互联网企业常见内存缓存服务软件：

软　件	类　型	主要作用	缓存的数据
Memcached	纯内存型	常用于缓存网站后端的各类数据，例如数据库中的数据	主要缓存用户重复请求的动态内容，例如：blog 的博文、BBS 的帖子等内容以及用户的 session 会话信息
Redis、Memcachedb	可持久化存储，即使用内存，也会使用磁盘存储	• 缓存后端数据库的查询数据 • 作为关系数据库的重要补充	作为缓存时，主要缓存用户重复请求的动态内容，例如：blog 的博文、BBS 的帖子等内容 作为数据库的有效补充时，例如：好友关注、粉丝统计、业务统计等功能可以用持久化存储

（续）

软　件	类　型	主要作用	缓存的数据
Squid、Nginx	内存或内存加磁盘缓存	主要用于缓存 Web 前端的服务内容	主要用于静态数据缓存，例如：图片、附件（压缩包）及 JS、CSS、html 静态代码等，此部分功能大多数企业会选择专业的 CDN 公司，如：网宿、蓝讯

4. Memcached 服务安装：

- 安装 libevent 及连接 Memcache 工具 nc。
- 使用 yum 安装 memcached，但 yum 安装的版本略低，高版本可使用源码编译安装。

5. 启动 Memcached：

```
[root@lan data]# Memcached -m 16m -p 11212 -d -u root -c 8192 -P
/var/run/11211.pid
```

参数说明：

命令参数	说　明
进程与连接设置	
-d	以守护进程（daemon）方式运行服务
-u	指定运行 Memcached 的用户，如果当前用户为 root，需要使用此参数指定用户
-l	指定 Memcached 进程监听的服务器 IP 地址，可以不设置此参数
-p（小写）	指定 Memcached 服务监听 TCP 端口号。默认为 11211
-P（大写）	设置保存 Memcached 的 pid 文件（$$），保存 PID 到指定文件
内存相关设置	
-m	指定 Memcached 服务可以缓存数据的最大内存，默认为 64MB
-M	Memcached 服务内存不够时禁止 LRU，如果内存满了会报错

命令参数	说　明
-n	为 key+value+flags 分配的最小内存空间，默认为 48 字节
-f	chunk size 增长因子，默认为 1.25
-L	启用大内存页，可以降低内存浪费，改进性能
并发连接设置	
-c	最大的并发连接数，默认是 1024
-t	线程数，默认 4。由于 Memcached 采用的是 NIO，所以太多线程作用不大
-R	每个 event 最大请求数，默认是 20
-C	禁用 CAS（可以禁止版本计数，减少开销）
调试参数	
-v	打印较少的 errors/warnings
-vv	打印非常多调试信息和错误输出到控制台，也打印客户端命令及响应
-vvv	打印极多的调试信息和错误输出，也打印内部状态转变

6. 使用 kill、pkill、killall 关闭实例：

```
[root@lan data]# killall memcached
或
[root@lan data]# pkill memcached
或
[root@lan data]# kill  ‘/var/run/11211.pid’
```

7. 可以使用”telnet ip port”的方式登陆到 Memcached，然后执行一些管理命令：

```
[root@lan data]# telnet 127.0.0.1 11211
```

8. 运维人员除了通过命令行管理 Memcached 以外，还可以使用很多第三方开源管理工具，如 memadmin。

9. Memcached 服务优化策略：

- 提高 Memcached 访问命中率是优化最关键的指标。
- 减少内存利用率，减少内存浪费。

10. Memcached 用于数据库内存缓存时存在一个问题，进程退出时，数据全丢，这样就需要将数据从数据库中读出来存到 Memcached 缓存里，然后前端才能开启对外访问。

11. Memcached 兼容持久化工具：

- MemcacheDB：是新浪网基于 Memcached 开发的一个开源项目。
- Tokyo Cabinet：读写速度快，在中小企业中应用很广。

第 14 章 企业级监控 Nagios 实践

1. 监控系统需要监控的数据有哪些？（资源、服务、设备、数据）

- 系统本地资源：负载、CPU、磁盘、内存等。
- 网络服务：端口、Web、DB、IDC 带宽网络流量。
- 其他设备：路由器、交换机等。
- 业务数据：用户登录失败次数、用户登录网站次数等。

2. Nagios 服务器端可以在 linux 系统和类 Unix 系统上运行，但目前无法在 Windows 上运行。Windows 可以作为被监控的主机运行 Nagios 客户端软件。

注意： Nagios 监控平台称为 Nagios 服务器端,而将远程被监控的服务器称为 Nagios 客户端。（平服被客）

3. Nagios 监控一般由一个程序（Nagios）、一个插件程序（Nagios-plugins 和一些可选的附加程序等组成）。Nagios 本身只是一个监控的平台而已，其具体的监控工作都是通过各类插件来实现的，也可以自己编写插件。因此，Nagios 主程序和 Nagios-plugins 插件都是 Nagios 服务器端必须要安装的程序组件。不过，一般 Nagios-plugins 也要安装于被监控端，用来获取相应的数据。（插件、程序）

4. 使用 yum 安装了 Nagios 之后，需要安装 Perl 插件程序，因此需要提前设置相关环境变量：

```
echo 'export LC_ALL=C'>>/etc/profile

tail -1 /etc/profile

source /etc/profile

echo $LC_ALL

cd ~
```

5. 在测试环境下为了调试方便，最好关掉 iptables 防火墙及 SELinux，如果是生产环境中，因为有外部 IP，所以在高度完毕后需要开启防火墙。一般允许服务通过的方法是整个局域网 IP 段都通过。

6. Nagios 可选的附加组件如下：

- NRPE 组件：

工作于被监控端，操作系统为 Linux/Unix 系统。用于在被监控的远程 linux/Unix 主机上执行脚本插件，获取数据回传给服务器端，以实现对这些主机资源和服务的监控。

- NSClient++组件：

相当于 Linux 下的 NRPE，用于被监控端为 Windows 系统的服务器。

- NDOUtils 组件（不推荐用）：

将 Nagios 的配置信息和各 event 产生的数据库存入数据库，以实现对这些数据的检索和处理，对于中小企业，不推荐使用，直接使用文件记录数据就很好。

- NSCA 组件：

相对于 NRPE，这个可以说是纯被动的监控组件，不推荐使用。让被监控的远程 linux/Unix 主机主动将监控到的信息发送到 Nagios 服务器，可以用在大规模分布式监控集群模式中，中小企业无需使用。

7. 根据监控行为，将 Nagios 的监控分为主动监控和被动监控（即 NRPE 半被动和 NSCA 全被动），下面先来看看什么是主动监控和半被动监控。（主动被动是针对服务器端来说的）

- 主动监控：把像 URL 监控一样由 Nagios 服务器端发出请求的主动探测监控方式，定义为主动监控，也就是说不需要在客户端安装任何插件。对于 Web 服务、数据库服务这种能对外提供服务的，一般用主动模式。

- 半被动监控：由 Nagios 服务器端通过 NRPE 插件定时去连接 client 的 NRPE 服务获取信息，并发回到 Nagios 服务器端的监控称之为半被动监控，这类监控通常是针对本地资源的，比如负载、内存、硬盘、虚拟内存、磁盘 I/O、温度等，而非系统对外提供的服务，只要安装了类似 NRPE 的插件方式的监控，都认为是半被动监控。

8. 如果不解决服务器的时间同步问题，很可能会导致 Nagios 整个服务配置异常甚至失败。

```
/usr/sbin/ntpdate pool.ntp.org

echo '#time sync by oldboy at 2015-06-26'>>/var/spool/cron/root

echo '*/5 * * * * /usr/sbin/ntpdate pool.ntp.org >/dev/null 2>&1'>>/var/spool/cron/root

crontab -l
```

9. 安装 Nagios：

(1) 服务器需要有 Web 界面展示监控效果，界面的展示主要使用 PHP 程序，因此，需要 LAMP 环境。特别强调：有些网友总想安装 LNMP 环境，这完全是自找麻烦，yum 安装的 LAMP 环境是配合 Nagios 服务器端展示界面的最佳环境。

```
yum install gcc glibc glibc-common -y #<==编译软件升级

yum install gd gd-devel -y #<==用于后面PNP出图的包

yum install mysql-server -y #<==非必须，如果有监控数据库，那么需要先安装

MySQL，否则，MySQL的相关插件不会被安装

yum install httpd php php-gd -y #<==Apache、PHP环境
```

上述所有软件包不需要在 Nagios 客户端安装。

(2) 创建 Nagios 服务器端需要的用户及组：

/usr/sbin/useradd nagios #<==这个地方最好创建家目录，否则，启动Nagios会提醒没家目录

/usr/sbin/groupadd nagcmd

/usr/sbin/usermod -a -G nagcmd nagios

/usr/sbin/usermod -a -G nagcmd apache

id -n -G nagios

id -n -G apache

groups nagios

groups apache

10. 启动 LAMP 环境的 HTTP 服务：

```
[root@lan data]#  /etc/init.d/httpd start
```

第 15 章 个人总结

1. 软件源码安装的相关步骤：　　（用、库、目、P）

(1)　创建相关用户组，该用户组用于软件运行及日志需要。（用户组）

(2)　安装软件需要的相关库。　　（库）

(3)　有记录日志需要则创建日志目录，并更改目录的用户组。　（目录）

(4)　如果是自定义安装目录，则需要修改 PATH 变量。　　（PATH）

(5)　./configure

(6)　make && make install

2. /etc/crontab 文件包括下面几行：

```
[root@localhost ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""HOME=/
# run-parts
```

```
51 * * * * root run-parts /etc/cron.hourly
24 7 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

第一行 SHELL 变量指定了系统要使用哪个 shell，这里是 bash，第二行 PATH 变量指定了系统执行命令的路径，第三行 MAILTO 变量指定了 crond 的任务执行信息将通过电子邮件发送给 root 用户，如果 MAILTO 变量的值为空，则表示不发送任务执行信息给用户，第四行的 HOME 变量指定了在执行命令或者脚本时使用的主目录。

3. crontab 文件的含义：

```
minute   hour   day   month   week   command
```

4. 查找文件：　　（find 查、locate 数据库、whereis 文、which 执）

- which　　查看可执行文件的位置

```
[root@redhat ~]# which passwd
/usr/bin/passwd
```

- whereis　查看文件的位置

```
[root@redhat ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
/usr/share/man/man5/passwd.5.g
```

将和 passwd 文件相关的文件都查找出来。

- locate　　配合数据库查看文件位置

```
[root@redhat ~]# locate passwd
```

- find　　实际搜寻硬盘查询文件名称

```
[root@redhat ~]# find / -name zgz
/home/zgz
/home/zgz/zgz
```

总结：用 whereis 和 locate 无法查找到需要的文件时，可以使用 find，但是 find 是在硬盘上遍历查找，因此非常消耗硬盘的资源，而且效率也非常低，因此建议大家优先使用 whereis 和 locate。 locate 是在数据库里查找，数据库大至每天更新一次。 whereis 可以找到可执行命令和 man page。find 就是根据条件查找文件。which 可以找到可执行文件和别名。

5. find 使用方法：

-amin n　　#查找系统中最后 N 分钟访问的文件

-atime n　　#查找系统中最后 n*24 小时访问的文件

-cmin n　　#查找系统中最后 N 分钟被改变状态的文件

-ctime n　　#查找系统中最后 n*24 小时被改变状态的文件

-empty　　#查找系统中空白的文件，或空白的文件目录，或目录中没有子目录的文件夹

-false　　#查找系统中总是错误的文件

-fstype type　#查找系统中存在于指定文件系统的文件，例如：ext2 .

-gid n　　#查找系统中文件数字组 ID 为 n 的文件

-group gname　#查找系统中文件属于 gnam 文件组，并且指定组和 ID 的文件。

```
find / -amin -10 # 查找在系统中最后 10 分钟访问的文件
```

6. 部分格式压缩与解压：

- .tar　　（x 解 c 打 vf）

解包：tar xvf FileName.tar

打包：tar cvf FileName.tar DirName

（注：tar 是打包，不是压缩！）

- .gz　　（gz、d 解）

解压：gzip -d FileName.gz

压缩：gzip FileName

- .tar.gz 和 .tgz　　（gz、z、x 解 c 打）

解压：tar zxvf FileName.tar.gz

压缩：tar zcvf FileName.tar.gz DirName

- .bz2

解压 1：bzip2 -d FileName.bz2

解压 2：bunzip2 FileName.bz2

压缩：　bzip2 -z FileName

- .tar.bz2　　（bz2 、j）

解压：tar jxvf FileName.tar.bz2

压缩：tar jcvf FileName.tar.bz2 DirName

注意：只要带 tar 的都是 x 解 c 打。

7. 更新 python 时可以网上的教程安装并更改 yum 的设置，但大多数教程并没有提及更新 python 后 pip 工具的更新，我们可以通过官方提供的工具来自动安装 setuptools 和 pip：

```
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
```

8. 本书出现的几个命令行工具参数：

- rpm -qa
- ps -ef
- netstat -lntup

vi 常见命令

1. 进入 vi 的命令

(1) vi filename :打开或新建文件，并将光标置于第一行首 （+、将光标置于）

(2) vi +n filename ： 打开文件，并将光标置于第 n 行首

(3) vi + filename ： 打开文件，并将光标置于最后一行首

(4) vi +/pattern filename：打开文件，并将光标置于第一个与 pattern 匹配的串处

(5) vi -r filename ： 在上次正用 vi 编辑时发生系统崩溃，恢复 filename （r、恢复）

(6) vi filename....filename ： 打开多个文件，依次进行编辑

2. 移动光标类命令

(1) w 或 W ： 光标右移一个字至字首 （w 右 b 左）

(2) b 或 B ： 光标左移一个字至字首

(3)) ： 光标移至句尾 （小括号句、大括号段）

(4) (： 光标移至句首

(5) }： 光标移至段落开头

(6) {： 光标移至段落结尾

(7) nG： 光标移至第 n 行首 （nG、移到第 n 行）

(8) n+： 光标下移 n 行

(9) n-： 光标上移 n 行

(10) n$： 光标移至第 n 行尾

(11) H：光标移至屏幕顶行 （H 顶、M 中间、L 最后）

(12) M：光标移至屏幕中间行

(13) L：光标移至屏幕最后行

(14) 0：（注意是数字零）光标移至当前行首

(15) $：光标移至当前行尾

3. 删除命令

(1) ndw 或 ndW：删除光标处开始及其后的 n-1 个字

(2) do：删至行首 （o 首$尾）

(3) d$：删至行尾

(4) ndd：删除当前行及其后 n-1 行

(5) x 或 X：删除一个字符，x 删除光标后的，而 X 删除光标前的

(6) Ctrl+u：删除输入方式下所输入的文本 （删除输入的文本）

4. 搜索及替换命令

(1) /pattern：从光标开始处向文件尾搜索 pattern

(2) ?pattern：从光标开始处向文件首搜索 pattern

(3) n：在同一方向重复上一次搜索命令 （n 重复）

(4) N：在反方向上重复上一次搜索命令

(5) :s/p1/p2/g：将当前行中所有 p1 均用 p2 替代

(6) :n1,n2s/p1/p2/g：将第 n1 至 n2 行中所有 p1 均用 p2 替代

(7) :g/p1/s//p2/g：将文件中所有 p1 均用 p2 替换

5. 最后行方式命令

(1) :n1,n2 co n3：将 n1 行到 n2 行之间的内容拷贝到第 n3 行下

(2) :n1,n2 m n3：将 n1 行到 n2 行之间的内容移至到第 n3 行下

(3) :n1,n2 d ：将 n1 行到 n2 行之间的内容删除

(4) :w ：保存当前文件

(5) :e filename：打开文件 filename 进行编辑

(6) :x：保存当前文件并退出

(7) :q：退出 vi

(8) :q!：不保存文件并退出 vi

(9) :!command：执行 shell 命令 command

6.