第一章　控制服务和守护进程

1.1 systemctl 控制服务

1、 systemd 介绍

许多年来，进程 ID 1 的 Linux 和 UNIX 系统一直使用 init 进程。这个过程是负责激活系统上的其他服务。系统在开机时与 system V 和 Linux 标准基地 （LSB） init 脚本开始频繁使用守护程序。由另一个服务经常使用的守护程序较少,如 initd 或 xinetd 的需求开始。

在红帽企业 Linux 7、 进程 ID 1 是 systemd，新的 init 系统。几个 systemd 所提供的新功能，包括：

- 并行化功能，提高系统的启动速度。
- 关于需求开始的守护程序不需要一个单独的服务。
- 自动服务依赖关系管理可以防止长时间超时，如网络不可用时不启动网络服务。
- 跟踪相关的进程 Linux 对照组一起使用的一种方法。

systemctl 和 systemd 单元

systemctl 命令用来管理不同类型的 systemd 对象，称之为单元。可用的单元类型的列表可以显示 systemctl -t 的帮助。

```
[root@localhost Desktop]# systemctl -t help
Available unit types:
service
socket
target
device
mount
automount
snapshot
timer
swap
path
slice
scope
```

服务状态

```
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2014-02-27 11:51:39 EST; 7h ago
 Main PID: 1073 (sshd)
   CGroup: /system.slice/sshd.service
           └─1073 /usr/sbin/sshd -D

Feb 27 11:51:39 server0.example.com systemd[1]: Started OpenSSH server daemon.
Feb 27 11:51:39 server0.example.com sshd[1073]: Could not load host key: /et...y
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on :: port 22.
Feb 27 11:53:21 server0.example.com sshd[1270]: error: Could not load host k...y
Feb 27 11:53:22 server0.example.com sshd[1270]: Accepted password for root f...2
Hint: Some lines were ellipsized, use -l to show in full.
```

Keyword:	Description:
loaded	Unit configuration file has been processed.
active (running)	Running with one or more continuing processes.
active (exited)	Successfully completed a one-time configuration.
active (waiting)	Running but waiting for an event.
inactive	Not running.
enabled	Will be started at boot time.
disabled	Will not be started at boot time.
static	Cannot be enabled, but may be started by an enabled unit automatically.

2、使用 systemctl 查看单元文件

- Query the state of all units to verify a system startup.

```
[root@serverX ~]# systemctl
```

- Query the state of only the service units.

```
[root@serverX ~]# systemctl --type=service
```

- Investigate any units which are in a failed or maintenance state. Optionally, add the **-l** option to show the full output.

```
[root@serverX ~]# systemctl status rngd.service -l
```

- The **status** argument may also be used to determine if a particular unit is active and show if the unit is enabled to start at boot time. Alternate commands can also easily show the active and enabled states:

```
[root@serverX ~]# systemctl is-active sshd
[root@serverX ~]# systemctl is-enabled sshd
```

- List the active state of all loaded units. Optionally, limit the type of unit. The **--all** option will add inactive units.

```
[root@serverX ~]# systemctl list-units --type=service
[root@serverX ~]# systemctl list-units --type=service --all
```

- View the enabled and disabled settings for all units. Optionally, limit the type of unit.

```
[root@serverX ~]# systemctl list-unit-files --type=service
```

- View only failed services.

```
[root@serverX ~]# systemctl --failed --type=service
```

3、启动和停止系统守护进程

- View the status of the **sshd** service.

```
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2014-02-27 11:51:39 EST; 7h ago
 Main PID: 1073 (sshd)
   CGroup: /system.slice/sshd.service
           └─1073 /usr/sbin/sshd -D
```

- Verify that the process is running.

```
[root@serverX ~]# ps -up 1073
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1073  0.1  0.0  82992  3612 ?        Ss   15:15   0:00 /usr/sbin/sshd -D
```

- Stop the service and verify the status.

```
[root@serverX ~]# systemctl stop sshd.service
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: inactive (dead) since Thu 2014-02-27 18:51:39 EST; 2s ago
 Main PID: 1073 (code=exited, status=0/SUCCESS)
```

- Start the service and view the status. The process ID has changed.

```
[root@serverX ~]# systemctl start sshd.service
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2014-02-27 18:52:39 EST; 2s ago
 Main PID: 1253 (sshd)
   CGroup: /system.slice/sshd.service
           └─1253 /usr/sbin/sshd -D
```

- Stop, then start, the service in a single command.

```
[root@serverX ~]# systemctl restart sshd.service
[root@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu 2014-02-27 18:54:39 EST; 2s ago
 Main PID: 1268 (sshd)
   CGroup: /system.slice/sshd.service
           └─1268 /usr/sbin/sshd -D
```

5、单元依赖与服务屏蔽

1） 单元的依赖

服务可开始作为其他服务的依赖。如果一个插座单元被启用，并且具有相同名称的服务单元，该服务会自动发出请求，网络上的插槽启用。通过路径单元在满足文件系统状态的服务，也可以触发。

该 systemctl list-dependencies 命令可以用来显示它必须与一个特定的单元一起启动等单位的一棵树。--reverse 选项此命令将显示哪些单元需要有指定的单元才能运行启动。

2） 屏蔽服务

一套系统可以安装一个特定的功能冲突的服务，如防火墙（iptables 和 firewalld）。为了防止管理员无意中启动一个服务，一个服务可能被屏蔽。创建蒙版在配置目录的链接，这样，如果该服务已启动，什么都不会发生。

```
[root@serverX ~]# systemctl mask network
ln -s '/dev/null' '/etc/systemd/system/network.service'
[root@serverX ~]# systemctl unmask network
rm '/etc/systemd/system/network.service'
```

6、单元依赖与服务屏蔽

服务在系统启动时，当链接在适当的 systemd 配置目录中创建。这些链接的创建和删除使用 systemctl 命令。

- View the status of a service.

```
[root@serverX ~]# systemctl status sshd.service
```

- Disable the service and verify the status. Note that disabling a service does not stop the service.

```
[root@serverX ~]# systemctl disable sshd.service
[root@serverX ~]# systemctl status sshd.service
```

- Enable the service and verify the status.

```
[root@serverX ~]# systemctl enable sshd.service
[root@serverX ~]# systemctl is-enabled sshd.service
```

1.2 控制引导过程

1、systemd targets

一个 systemd 目标是一组应该开始达到理想状态 systemd 单位。重要指标列于下表中

Target	Purpose
graphical.target	System supports multiple users, graphical and text-based logins.
multi-user.target	System supports multiple users, text-based logins only.
rescue.target	**sulogin** prompt, basic system initialization completed.
emergency.target	**sulogin** prompt, **initramfs** pivot complete and system root mounted on **/** read-only.

It is possible for a target to be a part of another target; for example, the **graphical.target** includes **multi-user.target**, which in turn depends on **basic.target** and others. These dependencies can be viewed from the command line with the following command:

```
[root@serverX ~]# systemctl list-dependencies graphical.target | grep target
```

An overview of all available targets can be viewed with:

```
[root@serverX ~]# systemctl list-units --type=target --all
```

An overview of all targets installed on disk can be viewed with:

```
[root@serverX ~]# systemctl list-unit-files --type=target --all
```

1）选择运行目标

在一个运行的系统，管理员可以选择使用 systemctl isolate 命令切换到不同的目标;例如，systemctl ioslate user.target。 不是所有的目标可以被分离。只有已 AllowIsolate=yes 在他们的单元文件设定的目标可以被分离;例如，graphical.target 目标可以被分离，但 cryptsetup.target 目标不能。

```
[root@localhost Desktop]# systemctl isolate
emergency.target   graphical.target   multi-user.target   rescue.target
[root@localhost Desktop]# systemctl isolate multi-user.target
```

2）设置一个默认的目标

当系统启动时，控制被传递下来的 initramfs 的到 systemd，systemd 将尝试激活 default.target 目标。通常情况下，default.target 目标将是一个符号链接（在/ etc/ systemd/系统/）要么 graphical.target 或多 user.target。

The **systemctl** tool provides two commands to manage the link: **get-default** and **set-default**.

```
[root@serverX ~]# systemctl get-default
multi-user.target
[root@serverX ~]# systemctl set-default graphical.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/graphical.target' '/etc/systemd/system/default.target'
[root@serverX ~]# systemctl get-default
graphical.target
```

3） 在引导时选择不同的目标

systemd.unit=：要选择不同的目标在引导时，一个特殊的选项，可以从引导装载程序被添加到内核命令行。例如，在系统启动到救援 shell，通过在交互式引导装载程序菜单中选择下列选项：

```
systemd.unit=rescue.target
```

使用选择不同的目标这种方法，使用下列程序红帽企业 Linux7 系统：

1. （重新）启动系统。
2. 按任意键中断引导装载程序菜单倒计时。
3. 将光标移动到入口启动。
4. 按 e 编辑当前的项目。
5. 将光标移动到与 linux16 开头的行。(内核命令行)
6. 追加 systemd.unit= desired.target。
7. 按 Ctrl+ X 来引导这些变化。

2 、 恢复 root 密码

在红帽企业 Linux7，可以从一个 initramfs 中运行的脚本暂停在某些点上，提供了一个 root shell，然后继续当那个 shell 退出。虽然这主要是用于调试，它也可以用来恢复丢失的 root 密码：

1. 重新启动系统。
2. 按任意键中断启动加载程序倒计时。
3. 将光标移动到需要被引导的条目。
4. 按 e 编辑选定的项目。
5. 将光标移动到内核命令行（找到 linux16 开头的行）。
6. 追加 rd.break（这将打破之前控制从 initramfs 的交给实际的系统）。
7. 按 Ctrl+ X 开机的变化。

To recover the **root** password from this point, use the following procedure:

1. Remount **/sysroot** as read-write.

```
switch_root:/# mount -oremount,rw /sysroot
```

2. Switch into a **chroot** jail, where **/sysroot** is treated as the root of the file system tree.

```
switch_root:/# chroot /sysroot
```

3. Set a new root password:

```
sh-4.2# passwd root
```

4. Make sure that all unlabeled files (including **/etc/shadow** at this point) get relabeled during boot.

```
sh-4.2# touch /.autorelabel
```

5. Type **exit** twice. The first will exit the **chroot** jail, and the second will exit the **initramfs** debug shell.

3、诊断和修复 systemd 启动问题

如果服务的启动过程中有问题，有提供给系统管理员，可以帮助调试和/或故障排除的几个工具： 早期使用 debug shell

通过运行 systemctl enable debug-shell.service，一个 root shell 会在引导过程的早期催生的 TTY9（CTRL + ALT + F9）。这个 shell 会自动以 root 身份登录，这样管理员可以使用一些其他的调试工具，而系统仍然启动。

应急救援目标

通过附加任何 systemd.unit= rescue.target 或 systemd.unit= emergency.target 从启动加载程序内核命令行，系统将产生进入一个特殊的救助，而不是正常启动或紧急 shell。这两种 shells 都需要 root 的密码。紧急目标保持根文件系统以只读方式挂载，而 rescue.target 等待 sysinit.target 先完成，使更多的系统将被初始化，例如，日志，文件系统等等这些 shells 将退出继续开机过程。

卡住作业

在启动过程中，systemd 滋生了许多作业机会。如果某些工作无法完成，他们将阻止其它作业的运行。要检查当前任务列表，管理员可以使用命令 systemctl list-jobs 列出工作。列为执行任何工作之前，必须完成列为等待的作业可继续进行。

第二章　管理 IPV6 网络

2.1 回顾 IPv4 的网络配置

1、 NetworkManager 的概述

在红帽企业 Linux7，网络接口的配置是通过 NetworkManager 系统守护进程来管理的。对于 NetworkManager 的：

- 设备是一个网络接口。
- 连接设置的集合，可以为设备进行配置。
- 一台设备只有一个连接是活动的。多个连接可能存在，由不同的设备使用或者允许被改变为在相同的设备的结构。
- 每个连接都有标识它的名称或 ID。
- 一个连接的永久配置存储在/etc/sysconfig/network-scripts 文件，脚本/为 ifcfg-name，其中 name 是连接的名称。如果需要该文件可以通过手动进行编辑。
- nmcli 实用程序可用于从 shell 提示符下创建和编辑连接文件。

2、 查看网络信息

The command **nmcli dev status** will show the status of all network devices:

```
[student@demo ~]$ nmcli dev status
DEVICE  TYPE      STATE         CONNECTION
eno1    ethernet  connected     eno1
eth0    ethernet  connected     static-eth0
eno2    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

The command **nmcli con show** will show a list of all connections. To list only the active connections, add the **--active** option.

```
[student@demo ~]$ nmcli con show
NAME         UUID                                  TYPE           DEVICE
eno2         ff9f7d69-db83-4fed-9f32-939f8b5f81cd  802-3-ethernet  --
static-eth0  72ca57a2-f780-40da-b146-99f71c431e2b  802-3-ethernet  eth0
eno1         87b53c56-1f5d-4a29-a869-8a7bdaf56dfa  802-3-ethernet  eno1
[root@demo ~]# nmcli con show --active
NAME         UUID                                  TYPE           DEVICE
static-eth0  72ca57a2-f780-40da-b146-99f71c431e2b  802-3-ethernet  eth0
eno1         87b53c56-1f5d-4a29-a869-8a7bdaf56dfa  802-3-ethernet  eno1
```

The **ip addr show** command displays the current configuration of network interfaces on the system. To list only a single interface, add the interface name as the last argument:

```
[student@demo ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,(1)UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    (2)link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    (3)inet 172.25.0.11/16 brd 172.25.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    (4)inet6 fe80::5054:ff:fe00:b/64 scope link
       valid_lft forever preferred_lft forever
```

3、 添加网络连接

该 nmcli con add 命令用来添加新的网络连接。

下面的命令将添加接口 eno2 一个新的连接，这将配置使用 DHCP IPv4 网络信息，将自动启动连接上。该配置将被保存在/ etc/sysconfig/network-scripts/ifcfg-eno2 因为 con-name 是 eno2。

```
[root@demo ~]# nmcli con add con-name eno2 type ethernet ifname eno2
```

下一个例子配置 eno2 接口，为静态 IPv4 地址，使用的 IPv4 地址和网络前缀 192.168.0.5/24 和默认网关 192.168.0.254，但仍 autoconnects 在启动和保存其配置到同一个文件中。

```
[root@demo ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
> ip4 192.168.0.5/24 gw4 192.168.0.254
```

3、 控制网络连接

nmcli con up name 命令将它绑定到网络接口上激活的连接名。注意，这个命令需要一个连接，而不是网络接口的名称。请记住，nmcli con show 可以用来列出所有可用的连接名称。

```
[root@demo ~]# nmcli con up static-eth0
```

该 nmcli dev 的断开设备的命令将断开网络接口设备。此命令可以简写 nmcli dev 的显示设备：

```
[root@demo ~]# nmcli dev dis eth0
```

4、 修改网络连接设置

NetworkManager 的连接有两种设置。其中静态连接的属性，是由管理员配置和存储在/ etc/sysconfig/network-scripts/ifcfg-*的配置文件组成的。也可能有活动的连接数据，该连接获得从 DHCP 服务器，哪些是不永久存储。

要列出一个连接的当前设置，运行 nmcli con show name 命令，其中 name 是连接的名称。小写的设置是静态属性，管理员可以更改设实例为临时使用的活动设置。

To set the IPv4 address to 192.0.2.2/24 and default gateway to 192.0.2.254 for the connection **static-eth0**:

```
[root@demo ~]# nmcli con mod static-eth0 ipv4.addresses "192.0.2.2/24 192.0.2.254"
```

一些设置可能有多个值。符号的设置名称的开始 - 特定值可以通过添加+或添加到列表从列表中要设置删除。

DNS 服务器 192.0.2.1 添加到域名服务器的静态连接的 eth0 使用的列表：

```
[root@demo ~]# nmcli con mod static-eth0 +ipv4.dns 192.0.2.1
```

默认情况下，nmcli con mod name 所做的更改会自动保存到/etc/sysconfig/network-scripts/ifcfg-name 文件。该文件也可以手动编辑一个文本编辑器。这样做之后，运行 nmcli con reload 使 NetworkManager 的读取配置更改。

因为 NetworkManager 的倾向于直接修改/ etc / resolv.conf 文件，直接编辑该文件可能会被覆盖。

要更改该文件中的设置，最好是设置 DNSn 和域指示在相关/ etc/sysconfig/network-scripts/ifcfg-*文件

Comparison of nm-settings and ifcfg-* Directives

nmcli con mod	ifcfg-* file	Effect
ipv4.method manual	**BOOTPROTO=none**	IPv4 addresses configured statically.
ipv4.method auto	**BOOTPROTO=dhcp**	Will look for configuration settings from a DHCPv4 server. If static addresses are also set, will not bring those up until we have information from DHCPv4.
ipv4.addresses "192.0.2.1/24 192.0.2.254"	**IPADDR0=192.0.2.1 PREFIX0=24 GATEWAY0=192.0.2.254**	Sets static IPv4 address, network prefix, and default gateway. If more than one is set for the connection, then instead of **0**, the **ifcfg-*** directives end with 1, 2, 3 and so on.
ipv4.dns 8.8.8.8	**DNS0=8.8.8.8**	Modify **/etc/resolv.conf** to use this **nameserver**.
ipv4.dns-search example.com	**DOMAIN=example.com**	Modify **/etc/resolv.conf** to use this domain in the **search** directive.
ipv4.ignore-auto-dns true	**PEERDNS=no**	Ignore DNS server information from the DHCP server.
connection.autoconnect yes	**ONBOOT=yes**	Automatically activate this connection at boot.
connection.id eth0	**NAME=eth0**	The name of this connection.
connection.interface-name eth0	**DEVICE=eth0**	The connection is bound to the network interface with this name.
802-3-ethernet.mac-address . . .	**HWADDR= . . .**	The connection is bound to the network interface with this MAC address.

5、 删除网络连接

使用 nmcli con del name 命令 删除指定的连接名称，断开设备并删除文件 /etc/sysconfig/network-scripts/ifcfg-name.

```
[root@localhost /]# nmcli connection add con-name eno3 type ethernet ifname eno3
Connection 'eno3' (391ec466-45e6-49b3-8268-72b0990667d7) successfully added.
[root@localhost /]# ls /etc/sysconfig/network-scripts/
ifcfg-Auto_Ethernet  ifdown-bnep  ifdown-post     ifdown-TeamPort  ifup-eth   ifup-plusb
ifcfg-eno16777736    ifdown-eth   ifdown-ppp      ifdown-tunnel    ifup-ippp  ifup-post
ifcfg-eno3           ifdown-ippp  ifdown-routes   ifup             ifup-ipv6  ifup-ppp
ifcfg-lo             ifdown-ipv6  ifdown-sit      ifup-aliases     ifup-isdn  ifup-routes
ifdown               ifdown-isdn  ifdown-Team     ifup-bnep        ifup-plip  ifup-sit
[root@localhost /]# cat /etc/sysconfig/network-scripts/ifcfg-eno3
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=eno3
UUID=391ec466-45e6-49b3-8268-72b0990667d7
DEVICE=eno3
ONBOOT=yes
[root@localhost /]# nmcli connection delete eno
eno16777736  eno3
[root@localhost /]# nmcli connection delete eno3
[root@localhost /]# ls /etc/sysconfig/network-scripts/
```

6、 修改系统的主机名

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@demo ~]# hostname
demo.example.com
```

A static host name may be specified in the **/etc/hostname** file. The **hostnamectl** command is used to modify t
and may be used to view the status of the system's fully qualified host name. If this file does not exist, the host nam
by a reverse DNS query once the interface has an IP address assigned.

```
[root@demo ~]# hostnamectl set-hostname demo.example.com
[root@demo ~]# hostnamectl status
   Static hostname: demo.example.com
         Icon name: computer
           Chassis: n/a
        Machine ID: 9f6fb63045a845d79e5e870b914c61c9
           Boot ID: aa6c3259825e4b8c92bd0f601089ddf7
    Virtualization: kvm
  Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
            Kernel: Linux 3.10.0-121.el7.x86_64
      Architecture: x86_64
[root@demo ~]# cat /etc/hostname
demo.example.com
```

静态主机名存储在/ etc /hostname。
上一版本的 Red Hat Enterprise Linux 的存储的主机名在/ etc/sysconfig /network 文件中。

7、 命令汇总

Summary of commands

The following table is a list of key commands discussed in this section.

Command	Purpose
`nmcli dev status`	Show the NetworkManager status of all network interfaces.
`nmcli con show`	List all connections.
`nmcli con show name`	List the current settings for the connection ***name***.
`nmcli con add con-name name ...`	Add a new connection named ***name***.
`nmcli con mod name ...`	Modify the connection ***name***.
`nmcli con reload`	Tell NetworkManager to reread the configuration files (useful after they have been edited by hand).
`nmcli con up name`	Activate the connection ***name***.
`nmcli dev dis dev`	Deactivate and disconnect the current connection on the network interface ***dev***.
`nmcli con del name`	Delete the connection ***name*** and its configuration file.
`ip addr show`	Show the current network interface address configuration.
`hostnamectl set-hostname ...`	Persistently set the host name on this system.

2.2 IPv6 的网络概念

1、IPv6 的概述

IPv6 的目的是作为替代 IPv4 的网络协议。它解决该主要问题是 IPv4 地址空间的枯竭。它还提供了许多增强功能和新功能对网络进行配置管理，并为未来的变化协议的支持。

IPv6 尚未广泛部署关键原因是核心协议中,没有一种简单的方法为具有 IPv6 地址与具有 IPv4 地址的系统进行通信。

目前最好的过渡计划是为所有主机配置 IPv4 和 IPv6 地址，这就是所谓的双堆栈配置，并在其上本课程将专注于方法。

2、解读 IPv6 地址

1） IPv6 地址

IPv6 地址是 128 位，每个段用四个十六进制数表示，用冒号分隔。每一个十六进制数代表 4 位 IPv6 地址，所以每个段代表 16 位的 IPv6 地址，共有 8 个段。

```
2001:0db8:0000:0010:0000:0000:0000:0001
```

为了更容易编写 IPv6 地址，以冒号分隔的段前导零可以省略。一段中四个零可以用一个零表示。

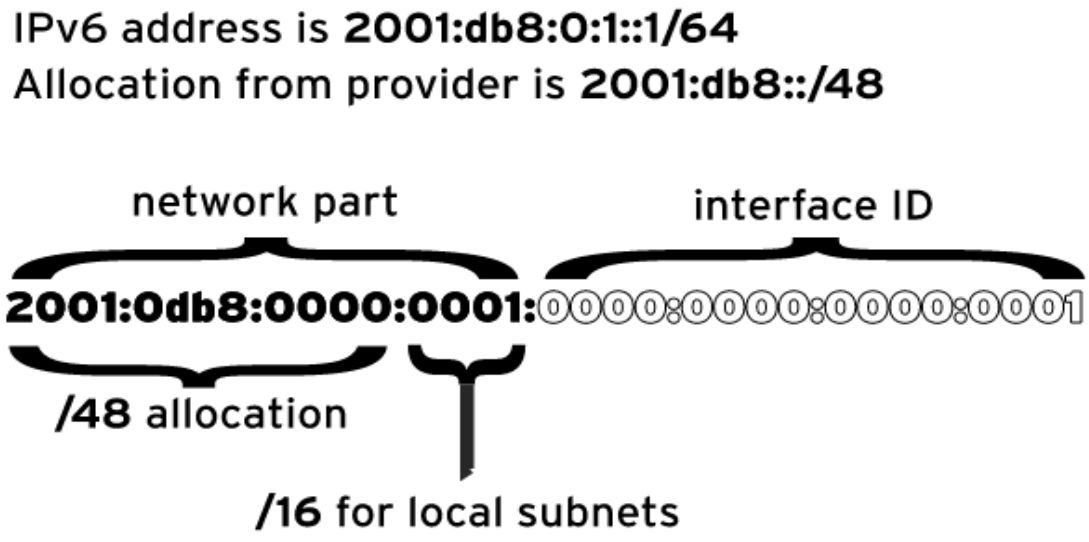
```
2001:db8:0:10:0:0:0:1
```

因为用零长串地址是共同的，一个或多个段的连续零可以用::表示，便::只能出现一次。

```
2001:db8:0:10::1
```

2） IPv6 子网

一个正常的单播地址被分为两个部分：网络前缀与接口标识。网络前缀标识子网。在同一子网中没有任何两个网络接口可以有相同的接口 ID;接口 ID 标识子网中的一个特定的接口。与 IPv4 不同，IPv6 有一个标准的子网掩码，这是用于几乎所有正常的地址，/64。在这种情况下，一半的地址是网络前缀和它的一半是接口 ID。这意味着，一个子网可以容纳尽可能多的主机是必要的。
典型地，网络提供者将一个较短的前缀分配给一个组织，例如/48。即留下 16 位的子网（高达 65536 子网）



3、IPv6 地址分配

IPv6 address allocation

Common IPv6 Addresses and Networks

IPv6 address or network	Purpose	Description
::1/128	localhost	The IPv6 equivalent to **127.0.0.1/8**, set on the loopback interface.
::	The unspecified address	The IPv6 equivalent to **0.0.0.0**. For a network service, this could indicate that it is listening on all configured IP addresses.
::/0	The default route (the IPv6 Internet)	The IPv6 equivalent to **0.0.0.0/0**. The default route in the routing table matches this network; the router for this network is where all traffic is sent for which there is not a better route.
2000::/3	Global unicast addresses	"Normal" IPv6 addresses are currently being allocated from this space by IANA. This is equivalent to all the networks ranging from **2000::/16** through **3fff::/16**.
fd00::/8	Unique local addresses (RFC 4193)	IPv6 has no direct equivalent of RFC 1918 private address space, although this is close. A site can use these to self-allocate a private routable IP address space inside the organization, but these networks cannot be used on the global Internet. The site must *randomly* select a **/48** from this space, but it can subnet the allocation into **/64** networks normally.
fe80::/64	Link-local addresses	Every IPv6 interface automatically configures a *link-local* address that only works on the local link on this network. This will be discussed in more detail later.
ff00::/8	Multicast	The IPv6 equivalent to **224.0.0.0/4**. Multicast is used to transmit to multiple hosts at the same time, and is particularly important in IPv6 because it has no broadcast addresses.

1） 链路本地地址

IPv6 的链路本地地址，是用来唯一的标示主机上特定的网络连接的不可路由的地址。系统中的每个网络接口自动配置为 FE80 ::链路本地地址。为了确保它是独一无二的，链路本地地址的接口 ID 是从网络接口的以太网硬件地址构造得来。通常的程序到 48 位 MAC 地址转换为 64 位接口 ID 是设置位的 MAC 地址，插入 FF：FE 它的两个中间字节之间。

网络前缀：fe80::/64

MAC 地址：00:11:22：aa：bb：cc

link-local 地址：ef80 ::211:22 ff：feaa：bbcc/64

同一网络主机可以通过链路本地地址通信。通信时，一个链路本地地址使用时必须在地址的末尾指定范围标识符。范围标识符包含%，其次是网络接口的名称。

例如，使用 ping6 来 ping 链路本地地址为 fe80::211:22 ff：feaa：bbcc 使用连接到 eth0 的网络接口的链路，正确的命令为：

```
[student@demo ~]$ ping6 fe80::211:22ff:feaa:bbcc%eth0
```

2） 多播

IPv6 的组播比在 IPv4 中起着更大的作用，因为在 IPv6 中没有广播地址。在 IPv6 的一个关键组播地址是 ff02::1，所有节点的链路本地地址。执行 ping 这个地址将流量发送到链路上的所有节点。链路范围的组播地址（起始 ff02 ::/8）需要与一个范围标识符指定的，就像一个链路本地地址。

```
[student@demo ~]$ ping6 ff02::1%eth0
PING ff02::1%eth0(ff02::1) 56 data bytes
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from fe80::200:aaff:fe33:2211: icmp_seq=1 ttl=64 time=102 ms (DUP!)
64 bytes from fe80::bcd:efff:fea1:b2c3: icmp_seq=1 ttl=64 time=103 ms (DUP!)
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=2 ttl=64 time=0.079 ms
...
```

2.3 IPv6 的网络配置

1、添加 IPv6 网络连接

nmcli con add 命令用来添加新的网络连接。
将增加该接口 eno2 一个新的连接，这将自动连接在启动时，使用的 DHCPv4 获取 IPv4 的网络信息。它也将获得 IPv6 的网络设置通过侦听本地链路上的路由器通告。

```
[root@demo ~]# nmcli con add con-name eno2 type ethernet ifname eno2
```

配置 eno2 接口，为静态地址，使用 IPv6 地址和网络前缀 2001：DB8：0:1:: C000：207/64，IPv6 缺省网关：2001：DB8：0:1::1，而 IPv4 地址和网络前缀 192.0.2.7/24 和默认 IPv4 网关的 192.0.2.1，保存其配置成/ etc/sysconfig/network-scripts/ifcfg-eno2。

```
[root@demo ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
> ip6 2001:db8:0:1::c000:207/64 gw6 2001:db8:0:1::1 ip4 192.0.2.7/24 gw4 192.0.2.1
```

2、修改 IPv6 网络连接设置

nmcli con show name 命令，其中 name 是连接的名称，可以用来查看 IPv6 相关的设置：

```
[root@demo ~]# nmcli con show static-eth0 | grep ipv6
ipv6.method:                            manual
ipv6.dns:                               2001:4860:4860::8888
ipv6.dns-search:                        example.com
ipv6.addresses:                         { ip = 2001:db8:0:1::7/64, gw = 2001:db8:0:1::1 }
ipv6.routes:
ipv6.ignore-auto-routes:                no
ipv6.ignore-auto-dns:                   no
ipv6.never-default:                     no
ipv6.may-fail:                          yes
ipv6.ip6-privacy:                       -1 (unknown)
ipv6.dhcp-hostname:                     --
[root@demo ~]#
```

同样，nmcli con mod name 可以用来调整如何连接设置 IPv6 地址。
修改 static-eth0 连接 IPv6 地址为 2001：DB8：0:1:: A00：1/64 和默认网关设置为 2001：DB8：0:1::1。

```
[root@demo ~]# nmcli con mod static-eth0 ipv6.address "2001:db8:0:1::a00:1/64 2001:db8:0:1::1"
```

在 static-eth0 连接上添加 DNS 服务器 2001:4860:4860::8888。

```
[root@demo ~]# nmcli con mod static-eth0 +ipv6.dns 2001:4860:4860::8888
```

比较命令与配置文件

Comparison of nm-settings and ifcfg-* Directives

nmcli con mod	ifcfg-* file	Effect
ipv6.method manual	IPV6_AUTOCONF=no	IPv6 addresses configured statically.
ipv6.method auto	IPV6_AUTOCONF=yes	Will configure network settings using SLAAC from router advertisements.
ipv6.method dhcp	IPV6_AUTOCONF=no DHCPV6C=yes	Will configure network settings by using DHCPv6, but not SLAAC.
ipv6.addresses "2001:db8::a/64 2001:db8::1"	IPV6ADDR=2001:db8::a/64 IPV6_DEFAULTGW=2001:db8::1	Sets static IPv4 address, network prefix, and default gateway. If more than one address is set for the connection, **IPV6_SECONDARIES** takes a double-quoted list of space-delimited address/prefix definitions.
ipv6.dns . . .	DNS0= . . .	Modify **/etc/resolv.conf** to use this **nameserver**. Exactly the same as IPv4.
ipv6.dns-search example.com	DOMAIN=example.com	Modify **/etc/resolv.conf** to use this domain in the **search** directive. Exactly the same as IPv4.
ipv6.ignore-auto-dns true	IPV6_PEERDNS=no	Ignore DNS server information from the DHCP server.
connection.autoconnect yes	ONBOOT=yes	Automatically activate this connection at boot.
connection.id eth0	NAME=eth0	The name of this connection.
connection.interface-name eth0	DEVICE=eth0	The connection is bound to the network interface with this name.
802-3-ethernet.mac-address . . .	HWADDR= . . .	The connection is bound to the network interface with this MAC address.

3、查看 IPv6 网络的信息

这两个 nmcli dev status 来显示所有网络设备和 nmcli con showr 来显示 NetworkManage 可用连接的列表中工作的状态。

ip addr show 命令仍然显示系统上的网络接口的当前配置。下面的示例调用了有关 IPv6 的一些项目。

```
[student@demo ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,(1)UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    (2)link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    (3)inet 192.0.2.2/24 brd 192.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    (4)inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global
       valid_lft forever preferred_lft forever
    (5)inet6 fe80::5054:ff:fe00:b/64 scope link
       valid_lft forever preferred_lft forever
```

(1) An active interface is **UP**.

(2) The **link/ether** line specifies the hardware (MAC) address of the device.

(3) The **inet** line shows an IPv4 address, its network prefix length, and scope.

(4) The **inet6** line shows an IPv6 address, its network prefix length, and scope. This address is of *global* scope normally used.

(5) This **inet6** line is for an address of *link* scope and can only be used for communication on the local Etherne

这个 ip -6 route show 命令用于显示系统中的 IPv6 路由表：

```
[root@demo ~]# ip -6 route show
unreachable ::/96 dev lo  metric 1024  error -101
unreachable ::ffff:0.0.0.0/96 dev lo  metric 1024  error -101
2001:db8:0:1::/64 dev eth0  proto kernel  metric 256
unreachable 2002:a00::/24 dev lo  metric 1024  error -101
unreachable 2002:7f00::/24 dev lo  metric 1024  error -101
unreachable 2002:a9fe::/32 dev lo  metric 1024  error -101
unreachable 2002:ac10::/28 dev lo  metric 1024  error -101
unreachable 2002:c0a8::/32 dev lo  metric 1024  error -101
unreachable 2002:e000::/19 dev lo  metric 1024  error -101
unreachable 3ffe:ffff::/32 dev lo  metric 1024  error -101
fe80::/64 dev eth0  proto kernel  metric 256
default via 2001:db8:0:1::ffff dev eth0  proto static  metric 1024
```

4、IPv6 的故障排除工具

1） ping6 测试连通性：

ping6 命令是平的红帽企业 Linux 的 IPv6 版本。

```
[root@demo ~]# ping6 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=18.4 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=0.180 ms
^C
--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.178/6.272/18.458/8.616 ms
[root@demo ~]#
```

ping 命令 FF02::1 可用于查找本地网络上的其他 IPv6 节点很有用。

```
[root@rhel7 ~]# ping6 ff02::1%eth1
PING ff02::1%eth1(ff02::1) 56 data bytes
64 bytes from fe80::78cf:7fff:fed2:f97b: icmp_seq=1 ttl=64 time=22.7 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=30.1 ms (DUP!)
64 bytes from fe80::78cf:7fff:fed2:f97b: icmp_seq=2 ttl=64 time=0.183 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=2 ttl=64 time=0.231 ms (DUP!)
^C
--- ff02::1%eth1 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.183/13.320/30.158/13.374 ms
[root@rhel7 ~]# ping6 -c 1 fe80::f482:dbff:fe25:6a9f%eth1
PING fe80::f482:dbff:fe25:6a9f%eth1(fe80::f482:dbff:fe25:6a9f) 56 data bytes
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=22.9 ms

--- fe80::f482:dbff:fe25:6a9f%eth1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.903/22.903/22.903/0.000 ms
```

Remember that IPv6 link-local addresses can be used by other hosts on the same link, just like normal addresses.

```
[student@demo ~]$ ssh fe80::f482:dbff:fe25:6a9f%eth1
student@fe80::f482:dbff:fe25:6a9f%eth1's password:
Last login: Thu Jun  5 15:20:10 2014 from demo.example.com
[student@server ~]$
```

2） 路由

tracepath6 和 traceroute -6 命令是相当于 tracepath 的和 traceroute

```
[root@demo ~]# tracepath6 2001:db8:0:2::451
1?: [LOCALHOST]                      0.091ms pmtu 1500
1:  2001:db8:0:1::ba                 0.214ms
2:  2001:db8:0:1::1                  0.512ms
3:  2001:db8:0:2::451                0.559ms reached
    Resume: pmtu 1500 hops 3 back 3
```

无论是 ss 命令或 netstat 命令可以显示网络套接字信息，他们采取几乎相同的选项。

```
[root@demo ~]# ss -A inet -n
Netid State      Recv-Q Send-Q       Local Address:Port          Peer Address:Port
tcp   ESTAB      0      0          192.168.122.98:22          192.168.122.1:35279
tcp   ESTAB      0      0        2001:db8:0:1::ba:22        2001:db8:0:1::1:40810
[root@demo ~]# netstat -46n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address        Foreign Address        State
tcp        0      0 192.168.122.98:22    192.168.122.1:35279    ESTABLISHED
tcp6       0      0 2001:db8:0:1::ba:22  2001:db8:0:1::1:40810  ESTABLISHED
```

Options for ss and netstat

Option	Description
-n	Show numbers instead of names for interfaces and ports.
-t	Show TCP sockets.
-u	Show UDP sockets.
-l	Show only listening sockets.
-a	Show all (listening and established) sockets.
-p	Show the process using the sockets.
-A inet	Display active connections (but not listening sockets) for the **inet** address family. That is, ignore local UNIX domain sockets. For **ss**, both IPv4 and IPv6 connections will be displayed. For **netstat**, only IPv4 connections will be displayed. (**netstat -A inet6** will display IPv6 connections, and **netstat -46** will display IPv4 and IPv6 at the same time.)

第三章　配置链路聚合和网桥

3.1 配置 network teaming

1、 network teaming(网络协作，网络群集)

简单来讲,Teaming 就是把同一台服务器上的多个物理网卡通过软件绑定成一个虚拟的网卡，也就是说，对于外部网络而言，这台服务器只有一个可见的网卡。对于任何应用程序，以及本服务器所在的网络，这台服务器只有一个网络链接或者说只有一个可以访问的 IP 地址。

之所以要利用 Teaming 技术，除了利用多网卡同时工作来提高网络速度以外，还有可以通过 Teaming 实现不同网卡之间的负载均衡（Load balancing）和网卡冗余（Fault tolerance）。

Team Driver 是红帽企业 7 的新项目， 提供了一套机制来将多个网络设备 （端口）绑定到数据链路层（OSI 第 2 层） 的单一逻辑接口上。 通常使用此机制来为链路增加最大带宽和提供冗余。Team Driver 在内核中只实现了必要的数据快速路径功能， 从而将大多数工作和逻辑转移到了用户空间后台程序中。 此策略相对于传统绑定具备多个优势， 比如， 在提供同等或更高性能的同时， 增加了稳定性并简化了调试和扩展。Rhel7 用很小内核驱动和一个叫 teamd 守护进程实现 network teaming,内核高效处理网络数据包，teamd 通过一个叫 runners 软件处理负载均衡或主备切换逻辑，teamd 有下列有效的 runners。

- Broadcast: 从所有端口传递包的一种简单 runner。
- Roundrobin:每个端口轮询传递一种简单 runner.
- Activebackup：监视链路变化并选择一个活跃端口传递数据的一种故障切换的 runner。
- Loadbalance:这种 runner 检查流量并用 hash 算法选择数据包传递的端口，来实现负载平衡。
- Lacp:实现 802.3ad 链路聚合控制协议，和 loadbalance 选择端口相同。

一个 teaming 接口由多个网络端口组成，当用 NetworkManager 控制 teaming 接口，特别是故障发生时：

- 启动一个 team 接口不会自动启动端口接口。
- 启动一个端口接口总是会启动 team 接口。
- 停止一个 team 接口也会停止一个端口接口。
- 一个 team 接口没有端口能启动一个静态的 IP 连接。
- 当启动一个 DHCP 连接，一个 team 没有端口就等待一个端口。

2、配置 network teams

1) 查看网络接口

```
[root@desktop10 Desktop]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
```

```
    link/ether 00:0c:29:ed:3e:56 brd ff:ff:ff:ff:ff:ff
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
    link/ether 00:0c:29:ed:3e:60 brd ff:ff:ff:ff:ff:ff
```

2）建立一个 active-backup 的 team 接口，连接名和接口名都为 team0。

```
[root@desktop10 Desktop]# nmcli con add type team con-name team0 ifname team0 config
'{"runner":{"name":"activebackup"}}'

[root@desktop10 Desktop]# nmcli mod team0 ipv4.adresses  '192.168.0.2/24'
[root@desktop10 Desktop]# nmcli mod team0 ipv4.method manual
```

其中 con-name ： team 连接名

Ifname：team 的接口名

Runner：broadcast，roundrobin，activebackup，loadbalance，lacp

3）为 team0 分配 ens33 和 ens37 接口。

```
[root@desktop10 Desktop]# nmcli con add type team-slave con-name team0-port1 ifname
ens33 master team0
[root@desktop10 Desktop]# nmcli con add type team-slave con-name team0-port2 ifname
ens37 master team0
```

4）查看 team 接口状态

```
[root@desktop10 Desktop]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens33
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
  ens37
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: ens33
[root@desktop10 Desktop]#
```

5）通过 team0 接口 Ping 局域网网关

```
[root@desktop10 Desktop]# ping -I team0 192.168.0.253
PING 192.168.0.253 (192.168.0.253) from 192.168.0.200 team0: 56(84) bytes of data.
64 bytes from 192.168.0.253: icmp_seq=1 ttl=64 time=0.613 ms
64 bytes from 192.168.0.253: icmp_seq=2 ttl=64 time=0.639 ms
64 bytes from 192.168.0.253: icmp_seq=3 ttl=64 time=0.657 ms
```

6) 打开另一终端，把 team0 活跃接口 ens33 断开，查看对 team0 影响及 ping 的结果

```
[root@desktop10 Desktop]# nmcli device disconnect ens33
[root@desktop10 Desktop]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens37
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: ens37
[root@desktop10 Desktop]#
[root@desktop10 Desktop]# ping -I team0 192.168.0.253
PING 192.168.0.253 (192.168.0.253) from 192.168.0.200 team0: 56(84) bytes of data.
64 bytes from 192.168.0.253: icmp_seq=1 ttl=64 time=0.613 ms
64 bytes from 192.168.0.253: icmp_seq=2 ttl=64 time=0.639 ms
64 bytes from 192.168.0.253: icmp_seq=3 ttl=64 time=0.657 ms
```

7) 启用 ens33,断开 ens37，查看对 team0 影响及 ping 的结果

```
[root@desktop10 Desktop]# nmcli con up team0-port1
[root@desktop10 Desktop]# nmcli device disconnect ens37
```

3.2 管理 network teaming

1、配置文件

```
 [root@desktop10 network-scripts]# cat ifcfg-team0
DEVICE=team0
TEAM_CONFIG="{\"runner\":{\"name\":\"broadcast\"}}"
DEVICETYPE=Team
BOOTPROTO=none
NAME=team0
ONBOOT=yes
IPADDR0=192.168.0.125
```

```
PREFIX0=24
```

```
[root@desktop10 network-scripts]# cat ifcfg-team0-port1
NAME=team0-port1
UUID=2cfc16e7-be55-44a2-9346-ebc0db95dd63
DEVICE=ens33
ONBOOT=yes
TEAM_MASTER=team0
DEVICETYPE=TeamPort
```

2、设置和调整 teams 配置

1）调整 JSON 配置

```
[root@desktop10 ~]# nmcli con mod team0 team.config JSON-configuration
```

JOSN-configuration:

- ‘{"runner":{"name":"activebackup"}}’
- ‘{"runner":{"name":"broadcast"}}’
- ‘{"runner":{"name":"roundrobin"}}’
- ‘{"runner":{"name":"loadbalance"}}’
- ‘{"runner":{"name":"lacp"}}’

```
[root@desktop10 ~]# nmcli connection modify team0 team.config
{ ‘"runner":{"name":"activebackup"}}’
```

3、network team 排错

Teamnl 和 teamctl 是两个有用排错命令。

1）显示 team0 中包的端口

```
[root@desktop10 ~]# teamnl team0 ports
 3: ens37: up 1000Mbit FD
 2: ens33: up 1000Mbit FD
```

2）显示前活跃端口

```
[root@desktop10 ~]# teamnl team0 getoption activeport
2
```

3） 设置 team0 活跃端口选项

```
[root@desktop10 ~]# teamnl team0 setoption activeport 3
[root@desktop10 ~]# teamnl team0 getoption activeport
3
```

4）用 teamdctl 显示 team0 当前的状态

```
[root@desktop10 ~]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens33
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
  ens37
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: ens33
[root@desktop10 ~]#
```

5）用 teamdctl 显示 team0 的 JSON 配置

```
[root@desktop10 ~]# teamdctl team0 config dump
{
    "device": "team0",
    "mcast_rejoin": {
        "count": 1
    },
    "notify_peers": {
        "count": 1
    },
    "ports": {
        "ens33": {
            "link_watch": {
                "name": "ethtool"
            }
        },
        "ens37": {
            "link_watch": {
                "name": "ethtool"
            }
        }
    },
    "runner": {
```

```
        "name": "activebackup"
    }
```

3.3 配置软件桥

1、软件桥

网桥基于 MAC 地址转发流量的二层设备。通过地址学习功能构建 MAC 表，来做转发决策。在 Linux 环境中软件桥能模拟硬件桥。软件桥最常见的应用是在虚拟化中一个或多个虚拟网卡共享一个物理网卡。

1、配置一个软件桥

```
[root@desktop10 ~]# nmcli con add type bridge con-name br0  ifname  br0
 Connection 'br0' (1cc41fc9-7bf0-4d01-b51e-994655fa15e3) successfully added.
[root@desktop10 ~]# nmcli con add type bridge-slave con-name br0-port1 ifname ens33
master br0
```

```
[root@desktop10 ~]# nmcli con mod br0 ipv4.addresses 192.168.0.2/24
[root@desktop10 ~]# nmcli con mod br0 ipv4.method static
```

注意：

NetworkManager 只能附加以太网接口到桥，不支持附加链路聚合接口，如 team 和 bound 接口，聚合接口必须手动创建配置文件,禁用 NetworkManager，启用 network 并用 ifup 和 ifdown 来管理桥和其他接口。

1）软桥配置文件

```
[root@desktop10 ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
STP=yes
BRIDGING_OPTS=priority=32768
TYPE=Bridge
BOOTPROTO=dhcp
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=br0
UUID=1cc41fc9-7bf0-4d01-b51e-994655fa15e3
```

```
ONBOOT=yes
[root@desktop10 ~]#
```

2）附加到桥的以太网接口的配置文件。

```
[root@desktop10 ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0-port1
TYPE=Ethernet
NAME=br0-port1
UUID=6f4cd149-a8df-4b6f-a4e2-57c2690d7816
DEVICE=ens33
ONBOOT=yes
BRIDGE=br0
[root@desktop10 ~]#
```

3）查看软件桥。

```
[root@desktop10 ~]# brctl show
bridge name     bridge id           STP enabled   interfaces
br0             8000.000000000000       yes             ens33
```

3.4 实验----配置链路聚合和桥

1、 配置 network teams

1) 查看网络接口

```
[root@desktop10 Desktop]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
    link/ether 00:0c:29:ed:3e:56 brd ff:ff:ff:ff:ff:ff
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
    link/ether 00:0c:29:ed:3e:60 brd ff:ff:ff:ff:ff:ff
```

2) 建立一个 active-backup 的 team 接口，连接名和接口名都为 team0。

```
[root@desktop10 Desktop]# nmcli con add type team con-name team0 ifname team0 config
'{"runner":{"name":"activebackup"}}'
```

3) 为 team0 分配 ens33 和 ens37 接口。

```
[root@desktop10 Desktop]# nmcli con add type team-slave con-name team0-port1 ifname
ens33 master team0
[root@desktop10 Desktop]# nmcli con add type team-slave con-name team0-port2 ifname
```

```
ens37 master team0
```

4）查看 team 接口状态

```
[root@desktop10 Desktop]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  ens33
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
  ens37
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: ens33
[root@desktop10 Desktop]#
```

2、创建一个 br0 桥

```
[root@localhost~]# nmcli con add type bridge con-name br0 ifname br0
Connection 'br0' (bfd49099-54c9-41fd-b4fb-2161691609cb) successfully added.
[root@localhost ~]# nmcli con mod br0 ipv4.addresses 10.11.21.238/23
[root@localhost ~]# nmcli con mod br0 ipv4.method manual
[root@localhost network-scripts]#
```

```
[root@localhost network-scripts]# cat ifcfg-br0
DEVICE=br0
STP=yes
TYPE=Bridge
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=br0
UUID=bfd49099-54c9-41fd-b4fb-2161691609cb
```

```
ONBOOT=yes
IPADDR0=10.11.21.238
PREFIX0=23
BRIDGING_OPTS=priority=32768
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
[root@localhost network-scripts]#
```

3、停止并禁用 NetworkManager 服务

```
[root@desktop10 ~]# nmcli device disconnect team0
[root@desktop10 ~]# nmcli device disconnect br0

[root@desktop10 ~]# systemctl stop NetworkManager.service
[root@desktop10 ~]# systemctl disable NetworkManager
[root@desktop10 ~]#
```

4、把 team0 附加到桥 br0，去掉 team0 和关联接口上 IP 设置 。

```
[root@desktop10 ~]# vim /etc/sysconfig/network-scripts/ifcfg-team0
[root@desktop10 ~]# cat !$
cat /etc/sysconfig/network-scripts/ifcfg-team0
DEVICE=team0
TEAM_CONFIG="{\"runner\":{\"name\":\"activebackup\"}}"
DEVICETYPE=Team
BOOTPROTO=none
NAME=team0
UUID=273a26e4-a37d-444a-9235-b0513d7f696e
ONBOOT=yes
BRIDGE=br0
[root@desktop10 ~]#
```

5、重启 network 服务，启动 br0 和重新启动 team0 。

```
[root@desktop10 ~]# systemctl restart network
```

6、用 ping 命令测试局域网的网关，开另一终端断开接口测试连通性。

```
[root@desktop10 ~]# ping -I br0 192.168.0.253
PING 192.168.0.253 (192.168.0.253) from 192.168.0.3 br0: 56(84) bytes of data.
64 bytes from 192.168.0.253: icmp_seq=1 ttl=64 time=0.768 ms
64 bytes from 192.168.0.253: icmp_seq=2 ttl=64 time=0.964 ms
```

开另一终端分别断开和启用两个物理接口，查看连通性。

```
[root@desktop10 ~]# ifdown ens33
[root@desktop10 ~]# teamdctl team0 s
setup:
  runner: activebackup
ports:
  ens33
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: ens37

[root@desktop10 ~]# ifup ens33
[root@desktop10 ~]# ifdown ens37
[root@desktop10 ~]# teamdctl team0 s
[root@desktop10 ~]# ifup ens37

[root@desktop10 ~]# ping -I br0 192.168.0.253
PING 192.168.0.253 (192.168.0.253) from 192.168.0.3 br0: 56(84) bytes of data.
64 bytes from 192.168.0.253: icmp_seq=156 ttl=64 time=0.768 ms
64 bytes from 192.168.0.253: icmp_seq=157 ttl=64 time=0.964 ms
```

第四章　网络端口安全

4.1 管理 firewalld

1、 firewalld 概述

Firewalld 是 rhel 7 管理主机级别防火墙默认方法。firewalld 是 rhel 7 的一大特性，最大的好处有两个：支持动态更新，不用重启服务；第二个就是加入了防火墙的“zone”概念。

FirewallD 提供了支持网络/防火墙区域(zone)定义网络链接以及接口安全等级的动态防火墙管理工具。它支持 IPv4, IPv6 防火墙设置以及以太网桥接，并且拥有运行时配置和永久配置选项。它也支持允许服务或者应用程序直接添加防火墙规则的接口。 以前的 system-config-firewall/lokkit 防火墙模型是静态的，每次修改都要求防火墙完全重启。这个过程包括内核 netfilter 防火墙模块的卸载和新配置所需模块的装载等。而模块的卸载将会破坏状态防火墙和确立的连接。

相反，firewall daemon 动态管理防火墙，不需要重启整个防火墙便可应用更改。因而也就没有必要重载所有内核防火墙模块了。不过，要使用 firewall daemon 就要求防火墙的所有变更都要通过该守护进程来实现，以确保守护进程中的状态和内核里的防火墙是一致的。另外，firewall daemon 无法解析由 ip*tables 和 ebtables 命令行工具添加的防火墙规则。

Firewalld 和 iptables、ip6tables、ebtables 有冲突，决定使用 firewalld 应禁用其他，一个好建议是将他们 mask。

```
[root@server1010 ~]# for SERVICE in iptables ip6tables ebtables;do
> systemctl mask ${SERVICE}.service
> done
ln -s '/dev/null' '/etc/systemd/system/iptables.service'
ln -s '/dev/null' '/etc/systemd/system/ip6tables.service'
ln -s '/dev/null' '/etc/systemd/system/ebtables.service'
[root@server1010 ~]#
```

Firewalld 把入站流量分到不同的区域，每一个区域有自己的规则。一个入站连接用哪个区域规则采用第一条规则相匹配的逻辑。

- 如果一个入站包源 IP 地址匹配一个区域的 source 规则，则这个包通过这个区域路由。
- 如果一个包的入站接口匹配一个区域的 filter，则这包通过这个区域路由。
- 否则用默认的区域，默认区域不是一个单独的区域，是由其他区域定义的。在没有更改的情况下系统默认区域是 public。

预定义的区域

Firewalld 定义许多预定义区域，默认区域是 public，接口分配到 public。下表列出安装的预定义区域，系统管理员也可以自定义区域。默认情况下，所有区域允许系统初始化一部分入站流量和所有出站流量。

Default configuration of firewalld zones

Zone name	Default configuration
trusted	Allow all incoming traffic.
home	Reject incoming traffic unless related to outgoing traffic or matching the **ssh**, **mdns**, **ipp-client**, **samba-client**, or **dhcpv6-client** pre-defined services.
internal	Reject incoming traffic unless related to outgoing traffic or matching the **ssh**, **mdns**, **ipp-client**, **samba-client**, or **dhcpv6-client** pre-defined services (same as the **home** zone to start with).
work	Reject incoming traffic unless related to outgoing traffic or matching the **ssh**, **ipp-client**, or **dhcpv6-client** pre-defined services.
public	Reject incoming traffic unless related to outgoing traffic or matching the **ssh** or **dhcpv6-client** pre-defined services. *The default zone for newly-added network interfaces.*
external	Reject incoming traffic unless related to outgoing traffic or matching the **ssh** pre-defined service. Outgoing IPv4 traffic forwarded through this zone is *masqueraded* to look like it originated from the IPv4 address of the outgoing network interface.
dmz	Reject incoming traffic unless related to outgoing traffic or matching the **ssh** pre-defined service.
block	Reject all incoming traffic unless related to outgoing traffic.
drop	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

由 firewalld 提供的区域按照从不信任到信任的顺序排序。

- 丢弃：任何流入网络的包都被丢弃，不作出任何响应。只允许流出的网络连接。
- 阻塞：任何进入的网络连接都被拒绝，并返回 IPv4 的 icmp-host-prohibited 报文或者 IPv6 的 icmp6-adm-prohibited 报文。只允许由该系统初始化的网络连接。
- 公开：用以可以公开的部分。你认为网络中其他的计算机不可信并且可能伤害你的计算机。只允许选中的连接接入。
- 外部：用在路由器等启用伪装的外部网络。你认为网络中其他的计算机不可信并且可能伤害你的计算机。只允许选中的连接接入。
- 隔离区（dmz）：用以允许隔离区中的电脑有限地被外界网络访问。只接受被选中的连接。
- 工作：用在工作网络。你信任网络中的大多数计算机不会影响你的计算机。只接受被选中的连接。
- 家庭：用在家庭网络。你信任网络中的大多数计算机不会影响你的计算机。只接受被选中的连接。
- 内部：用在内部网络。你信任网络中的大多数计算机不会影响你的计算机。只接受被选中的连接。

- 受信任的：允许所有网络连接。

2、 管理 firewalld

有三种主要方式为系统管理员与 firewalld 进行交互：

- 通过直接编辑配置文件/etc/firewalld/
- 通过使用图形化的防火墙配置工具
- 通过使用防火墙 cmd 命令行

1）配置防火墙工具 firewall-cmd

firewall-cmd commands	Explanation
--get-default-zone	Query the current default zone.
--set-default-zone=<ZONE>	Set the default zone. This changes both the runtime and the permanent configuration.
--get-zones	List all available zones.
--get-active-zones	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
--add-source=<CIDR> [--zone=<ZONE>]	Route all traffic coming from the IP address or network/netmask <CIDR> to the specified zone. If no --zone= option is provided, the default zone will be used.
--remove-source=<CIDR> [--zone=<ZONE>]	Remove the rule routing all traffic coming from the IP address or network/netmask <CIDR> from the specified zone. If no --zone= option is provided, the default zone will be used.
--add-interface=<INTERFACE> [--zone=<ZONE>]	Route all traffic coming from <INTERFACE> to the specified zone. If no --zone= option is provided, the default zone will be used.
--change-interface=<INTERFACE> [--zone=<ZONE>]	Associate the interface with <ZONE> instead of its current zone. If no --zone= option is provided, the default zone will be used.

--list-all [--zone=<*ZONE*>]	List all configured interfaces, sources, services, and ports for **<*ZONE*>**. If no **--zone=** option is provided, the default zone will be used.
--list-all-zones	Retrieve all information for all zones. (Interfaces, sources, ports, services, etc.)
--add-service=<*SERVICE*> [--zone=<*ZONE*>]	Allow traffic to **<*SERVICE*>**. If no **--zone=** option is provided, the default zone will be used.
--add-port=<*PORT/PROTOCOL*> [--zone=<*ZONE*>]	Allow traffic to the **<*PORT/PROTOCOL*>** port(s). If no **--zone=** option is provided, the default zone will be used.
--remove-service=<*SERVICE*> [--zone=<*ZONE*>]	Remove **<*SERVICE*>** from the allowed list for the zone. If no **--zone=** option is provided, the default zone will be used.
--remove-port=<*PORT/PROTOCOL*> [--zone=<*ZONE*>]	Remove the **<*PORT/PROTOCOL*>** port(s) from the allowed list for the zone. If no **--zone=** option is provided, the default zone will be used.
--reload	Drop the runtime configuration and apply the persistent configuration.

说明：

- --permanent:永久配置
- --zone:设置哪个区域，省略是用默认区域

The following examples show the default zone being set to **dmz**, all traffic coming from the **192.168.0.0/24** network being assigned to the **internal** zone, and the network ports for **mysql** being opened on the **internal** zone.

```
[root@serverX ~]# firewall-cmd --set-default-zone=dmz
[root@serverX ~]# firewall-cmd --permanent --zone=internal --add-source=192.168.0.0/24
[root@serverX ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@serverX ~]# firewall-cmd --reload
```

3、 实验---配置 firewalld

实验要求：在 serverx.example.com 上设置 firewalld 启动并开机自动启用，设置所有没有指定连接用 dmz 区域。设置源 IP 为 172.25.10.0/24 的包路由到 work 区域。设置 work 区域允许 https，过滤 http。

1）安装 httpd 和 mod_ssl

2）启动 httpd 服务

3）建立测试网页文件/var/www/html/index.html

4）设置 firewalld dmz 为默认区域。

```
[root@server10 ~]# firewall-cmd --set-default-zone=dmz
```

5） 配置 firewalld 源地址是 172.25.10.0/24 的包路由到 work 区域

```
[root@server10 ~]# firewall-cmd --permanent --zone=work
--add-source=172.25.10.0/24
```

6） 允许 work 区域的入站 https 流量

```
[root@server10 ~]# firewall-cmd --permanent --zone=work  --add-service=https
```

7）应用 firewalld 设置

```
[root@server10 ~]# firewall-cmd --reload
```

8）检查运行的 firewall 配置

```
[root@server10 ~]# firewall-cmd --get-default-zone
dmz
[root@server10 ~]# firewall-cmd --get-active-zones
dmz
  interfaces: br0 ens33 ens37
work
  sources: 172.25.10.0/24
[root@server10 ~]# firewall-cmd --zone=work --list-all
work
  interfaces:
  sources: 172.25.10.0/24
  services: dhcpv6-client https ipp-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

9）在 server10X 上验证

```
[root@desktop10 ~]# curl http://serverx.example.com        // failed

[root@desktop10 ~]# curl -k https://serverx.example.com      //sucess
```

-k 选项跳过证书。

4.2 管理 firewalld 富语言(rich)规则

1、 富语言规则概念

Firewalld 除了提供普通的区域和服务语法外，管理员能通过两个选项添加防火墙规则：direct 和 rich。

1) Direct

Direct 规则允许管理员插入手工编写的 iptables 规则到一个区域用 firewalld 管理。虽然强大,通过其他手段暴露不暴露内核 netfilter 子系统的功能,这些规则很难管理。dircet 规则的灵活性也低于标准的规则和富语言的规则。配置 direct 规则没有在本课程中涵盖。

除非明确插入一个区域由 firewall 管理，direct 规则在任何 firewalld 规则之前将被解析。

2) rich

Rich 给管理员一个基本防火墙语法没有覆盖自定义规则一种表法语言。例如：只允许单一 IP 连接到一个服务。

Rich 规则能用做基本 allow/deny 规则，也能用做配置 syslog 和 auditd 日志，端口转发，伪装、速率限制。

rich 基本的语法由下列块表示

```
Rule
[source]
[destination]
Service | port | protocol | icmp-block | masquerade | forward-port
[log]
[audit]
[accept | reject | drop]
```

3) 规则排序

多个规则加到一个区域，排序的的规则对防火墙行为起很大作用。所有区域内其本规序的排序是相同的。

- 这个区域端口转发和伪装规则
- 这个区域日专规则
- 这个区域任何允许规则
- 这个区域任何拒绝的规则

在所有情况下，按第一条匹配的规则处理。如果一个包在一个区域没有任何匹配的规则，则使用这个区域的默认规则，除 trused 区域外，其他区域默认是拒绝。匹配日志规则后还要续继匹配。

4) 测试与调试

为了便于测试与调试，几乎所有规则都可以添加一个运行配置的超时时间。用

--timeout=

2、 富语言规则配置

Firewall-cmd 有四个选项为富规则，所有这些选项都可以与普通--permanent，--zone 选项组合。

- --add-rich-rule= ‘<RULE>’

  添加一个规则到一个指定区域或默认区域。

- --remove-rich-rule= ‘<RULE>’

  删除一个规则在一个指定区域或默认区域。

- --query-rich-rule= ‘<RULE>’

  查询一个规则，查到返回 0，否则返回 1.

- --list-rich-rules

  列出所有富规则。

富规则配置举例

1） 在 work 区域中拒绝 192.168.0.11 的所有流量。

```
[root@desktop10 ~]# firewall-cmd --permanent --zone=work --add-rich-rule='rule family=ipv4 source address=192.168.0.11/32 reject'

[root@desktop10 ~]# firewall-cmd --reload
success
[root@desktop10 ~]# firewall-cmd --list-rich-rules  --zone=work
rule family="ipv4" source address="192.168.0.11/32" reject
[root@desktop10 ~]#
```

注：当用 address 选项用 source 或 destination 时，family 选项要设置 ipv4 或 ipv6.

2） 在默认区域中允许每分钟两个新的 ftp 连接。

```
t@desktop10 ~]# firewall-cmd --add-rich-rule='rule service name=ftp limit value=2/m accept'
```

3） 在默认区域中 drop 入站的所有 IPSce esp 协议的包。

```
[root@desktop10 ~]# firewall-cmd --permanent --add-rich-rule='rule protocol value=esp drop'
```

3） 在 work 区域中允许源 IP 为 192.168.1.0/24，tcp 端口 7900-7905 的所有包。

```
[root@desktop10 ~]# firewall-cmd --permanent --zone=work --add-rich-rule='rule family=ipv4 source address=192.168.1.0/24 port port=7900-7905 protocol=tcp accept'
```

2、 富语言规则日志

当调试或监视防火墙，记录允许或拒绝的连接是有用的。完成日志记录有两种方法：一是由 syslog 日志，二是发送消息到内核 audit 子系统，由 auditd 管理。

在所有情部分下，日志都能被限速，确定日志文件和磁盘空间不被填满。

配置举例

1）在 work 区域允许新的 SSH 连接，日志新的连接到 syslog，日志级别 notice，并限制每分钟最大 3 条消息。

```
[root@desktop10 ~]# firewall-cmd --permanent --zone=work --add-rich-rule='rule service name="ssh" log prefix="ssh" level="notice" limit value="3/m" accept'
```

2） 在默认区域中在下个 5 分钟内拒绝新的 IPv6 子网 2001::/64 连接的 DNS 请求，并日志到 audit，每小最多一条消息。

```
[root@desktop10 ~]# firewall-cmd --add-rich-rule='rule family=ipv6 source address="2001::/64" service name="dns" audit limit value="1/h" reject' --timeout=300
```

4.3 伪装和端口转发

1、 NAT

Firewalld 支持两种 NAT：masquerading 和 port forwarding。基本配置都能通过普通规则配置，富规则能完成高级配置。两者都可以修改一个包的源或目的地址。

2、Masquerading

Masquerading 是一个统发送包时源地址不用发送者的地址，而是用全局地址。当回复数据包到达时，再把目标地址修改为源主机地址并发送包，通常用做内网和互联网连接的边缘网络，Masquerading 是 NAT 的一种。IPv6 不支持。

配置举例

1）在一个区域中用普通 firewall-cmd 命令配置 masquerading。

```
[root@desktop10 ~]# firewall-cmd --permanent --zone=<zone> --add-masquerade
```

2）配置 masquerading 客户端控制，用富规则。

```
[root@desktop10 ~]# firewall-cmd --permanent --zone=<zone> --add-rich-rule='rule
family=ipv4 source address="192.168.1.0/24" masquerade
```

3、Port forwarding

另一种 NAT 是 port forwarding，到一个端口的流量可以转发到同一台机中不同端口，或转发另一台机的某个端口。这种机制常用作隐藏内部服务器或提供访问某一服务的一个替代端口。

配置举例

1）在一个区域中用普通 firewall-cmd 命令配置。

```
[root@desktop10~]# firewall-cmd --permanent --zone=<zone>
--add-forward-port=<port>:proto=<protocol>:toport=<port>:toaddr=<ip>
```

在防火墙的 public 区域中把端口为 513/tcp 流量转发到 192.168.0.254 的 132/tcp 端口。

```
[root@desktop10~]# firewall-cmd --permanent --zone=public
--add-forward-port=513:proto=tcp:toport=132:toaddr=192.168.0.254
```

2）在一个区域中用富规则 firewall-cmd 命令配置。

在防火墙的 work 区域中把源 IP 为 192.168.0.0/26 端口为 80/tcp 流量转发到 8080/tcp 端口。

```
[root@desktop10~]# firewall-cmd --permanent --zone=work --add-rich-rule=' rule
family=ipv4 source address=192.168.0.0/26 forward-port port 80 protocol=tcp
to-port=8080'
```

4.4 管理 SELinux 端口标签

1、SELinux port labeling

SELinux 不仅对文件和进程进行标签，网络流量也有严格的策略，SELinux 一种控制流量的方法对网络端口进行标签。例如，targetd 策略对 tcp/22 关联 ssh_port_t。

任何时候一个进程监听一个端口，SELinux 检查标签，看进程是否允许绑定这个端口标签，它能够从一个已知端口去阻止一个欺骗服务。

2、管理 SELinux port labeling

当管理员决定在一个非标准端口上运行一个服务，很可能需要更新 SELinux 端口标签。有些情况，targetd 策略已做标签，例如：tcp/8080 已标签关联 http_port_t。

1）查看端口标签。

```
[root@desktop10~]# semanage port -l
```

2）修改端口标签。

例：允许 gopher 服务监听 71/tcp 端口。

```
[root@desktop10~]# semanage -a -t gopher_port_t -p tcp 71
```

Target 策略有大量的端口类型，每个服务的 SELinux 类型、布尔值和端口类型能用下列 man 命令中找到。

```
[root@desktop10~]# yum -y install selinux-policy-devel
[root@desktop10~]# mandb
[root@desktop10~]# man -k  _selinux
```

3）删除端口标签。

```
[root@desktop10~]# semanage -d -t gopher_port_t -p tcp 71
```

4）修改端口标签。

```
[root@desktop10~]# semanage -m -t http_port_t -p tcp 71
```

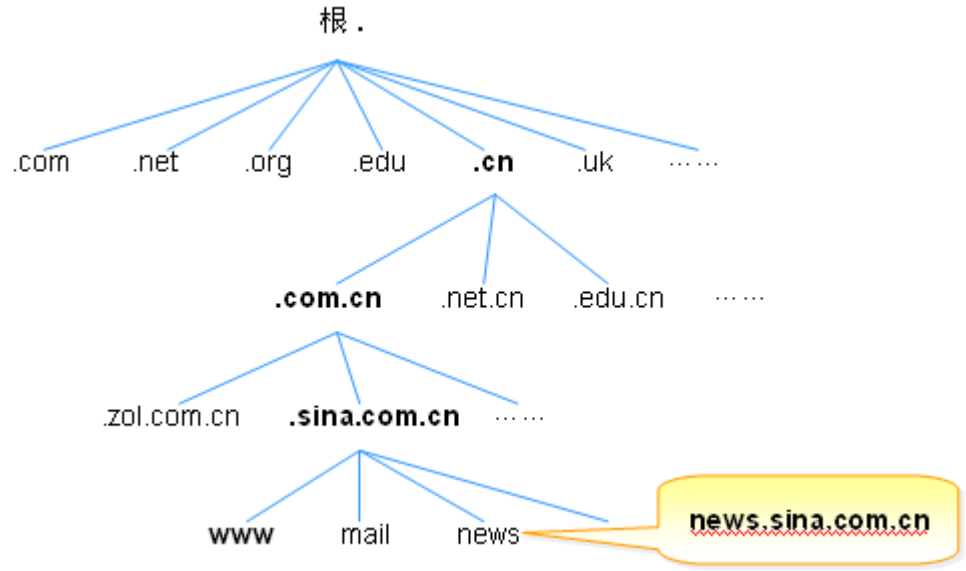
第五章　配置 DNS 服务

5.1 DNS 概念

在 Internet 中使用 IP 地址来确定计算机的地址，这种以数字表示的 IP 地址不容易记忆。为了便于对网络地址的管理和分配，人们采用了域名系统，引入了域名的概念。通过为每台主机建立 IP 地址与域名之间的映射关系，用户可以避开难记的 IP 地址，而使用域名来唯一地标识网络中的计算机。DNS（Domain Name System，域名系统），DNS 的功能是将局域网和

互联网中的域名翻译成 IP 地址。

5.1.1 域名空间结构



1. 根域

根（root）域就是“.”（点号），它由 Internet 名字注册授权机构管理该机构把域名空间各部分的管理责任分配给连接到 Internet 的各个组织。

2. 顶级域

DNS 根域的下一级是顶级域，由 Internet 名字授权机构管理，共有三种类型的顶级域。

- 组织域，采用三个字符的代号，表示 DNS 域中所包含的组织的主要功能或活动。如表 2-1 所示。

表 2-1 组织域

顶级域名	说　　明
gov	政府部门
com	商业部门
edu	教育部门
org	民间团体组织
net	网络服务机构
mil	军事部门

- 国家或地区域，采用两个字符的国家或地区代号。如表 2-2 所示。
- 反向域，这是一个特殊域，名称为 in-addr.arpa，用于将 IP 地址映射到名称。

表 2-2　国家或地区域

顶级域名	国别/地区

cn	中国
jp	日本
uk	英国
au	澳大利亚
hk	中国香港
…	…

3. 二级域

二级域是注册到个人、组织或公司的名称。这些名称基于相应的顶级域，如“Microsoft.com”，就是基于顶级域“.com”。二级域下可以包括主机和子域，如“Microsoft.com”可包含如“ftp.microsoft.com”这样的主机，也可以包含如“sale.microsoft.com”这样的子域，而该子域还可以包含如“printerl.sale.microsoft.com”这样的主机。

4. 主机名

主机名处于域名空间结构中的最底层，主机名和前面讲的域名（DNS 后缀）结合构成 FQDN（Full Qualified Domain Name，完全合格的域名），主机名是 FQDN 最左端的部分。例如，“aaa.bbb.com”中的“aaa”是主机名，“bbb.com”被称为 DNS 后缀。DNS 后缀最右边的“.”代表根域，因为根域是域名结构的最顶层，所以实际应用中可以将最右边的“.”省略，简写成“aaa.bbb.com”。

包含主机名及其所在域名的完整地址又称为 FQDN（Full Qualified Domain Name 完全限定域名）地址，或称为全域名。例如新浪网站服务器的地址 www.sina.com.cn，其中“www”表示服务器的主机名（大多数的网站服务器都使用该名称），sina.com.cn 表示该主机所属的 DNS 域。该地址中涉及多个不同 的 DNS 域及其他服务器

- .根域服务器，是所有主机域名解析的源头，地址中最后的通常.被省略。
- .cn 域服务器，负责所有以 cn 结尾的域名解析，.cn 域是处于根域之下的顶级域。
- .com.cn 域服务器，负责所有以.com.cn 结尾的域名解析，.com.cn 域是.cn 域的子域。
- .sina.com.cn 域服务器，由新浪公司负责维护，主要提供域.sina.com.cn 中所有主机的域名解析，如 www.sina.com.cn、mail.sina.com.cn 等，.sina.com.cn 域是.com.cn 域的子域。

从以上的 DNS 层.次结构中可以看出，对于 Internet 中每个主机域名的解析，并不需要涉及及多的 DNS 服务器就可以完成。通常客户端主机中只需要指定 1~3 个 DNS 服务器地址，就可以通过过递归或迭代的查询方式获知要访问的域名对应的 IP 地址。

客户机解析 www.baidu.com，DNS 查询过程。

PC——本地缓存——本机 hosts 文件——首选 DNS——根 DNS.——.com——baidu.com

5.1.2 DNS 系统的作用

DNS 服务器的主要作用就是维护一个主机域名与 IP 地址的对应关系数据库，在需要的时候为客户端网络程序提供以下两个方面的地址解析功能。

1、 正向解析：将主机的名称（域名）解析为对应的 IP 地址。域名的正向解析是 DNS 服务器最基本的功能，也是最常用的功能。

2、 反向解析：将主机的 IP 地址解析为对应的域名。域名的反向解析不是很常用，却是 DNS 服务器不可缺少的功能，一些特殊的网络应用（如邮件服务）可能会根据域名的反向解析结果实施访问控制策略，若缺少对应的反向解析记录有可能造成部分网络服务的不可用。

5.1.3 DNS 系统的类型

DNS 服务器按照配置和实现功能的不同，包括多种不同的类型。同一台服务器相对于不同的区域来说，也拥有不同的身份。常见的 DNS 服务器类型如下：

- 缓存域名服务器，或称为“唯高速缓存服务器”，其主要功能是提供域名解析记录的缓存。
- 主域名服务器，是特定 DNS 区域的官方服务器，对于某个指定域，主域名服务器是唯一存在的，其管理的域名解析记录具权威性。主域名服务器需要在本地设置所管理区域的地址数据库文件。
- 从域名服务器，或称为辅助域名服务器，其主要功能是提供备份，通常与主域名服务器同时提供服务，对于客户端来说，从域名服务器提供与主域名服务器完全相同的功能。

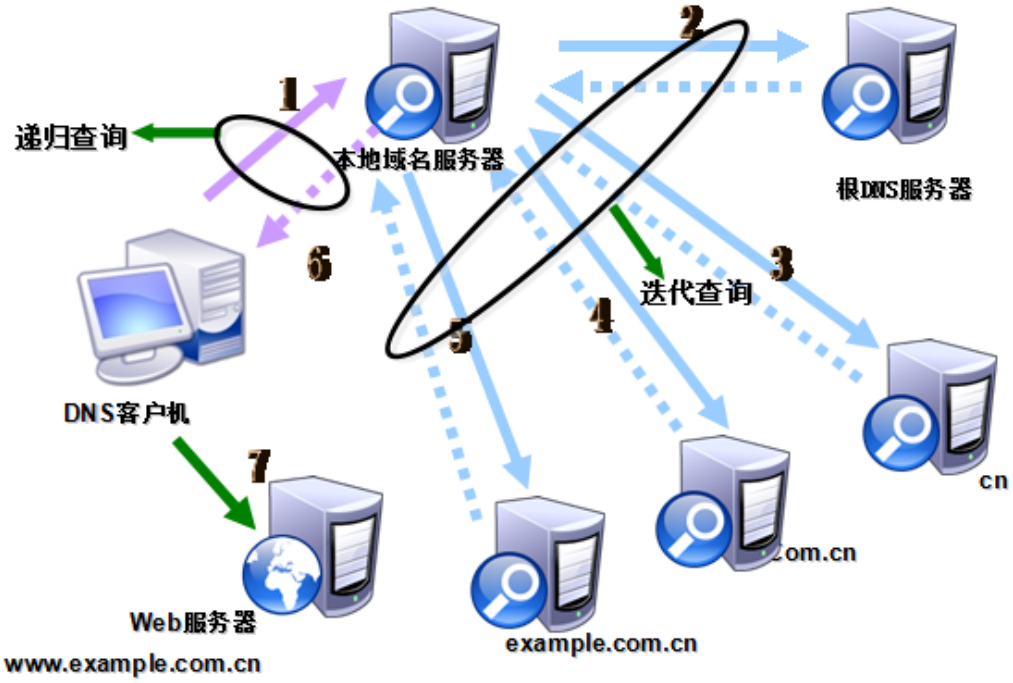
5.1.3 DNS 资源记录

资源记录	说　明
SOA（起始授权机构）	指定该区域的权威名称服务器
NS（名称服务器）	表示某区域的权威服务器和SOA中指定的该区域的主服务器和辅助服务器
A（主机）	列出了区域中FQDN（完全合格的域名）到IP地址的映射
PTR（指针）	相对于A资源记录，PTR记录把IP地址映射到FQDN
MX	邮件交换器记录，向指定邮件交换主机提供消息路由
SRV（服务）	列出了哪些服务器正在提供特定的服务
CNAME（别名）	将多个名字映射到同一台计算机，便于用户访问

注意：

AAAA　　FQDN 到 IPv6 地址的映射

5.1.3 DNS 名称解析过程



5.2 配置缓存的 DNS 服务器

1、安装 unbound

```
[root@server10 ~]#  yum install -y unbound
```

2、启动服务，并设置开机自动启服服务

```
[root@server10 ~]# systemctl start unbound
[root@server10 ~]# systemctl enable unbound.service
```

3、修改配置文件/etc/unbound/unbound.conf

```
[root@server10 ~]# vim /etc/unbound/unbound.conf
 38          interface: 0.0.0.0                    //在所有接口监听 DNS 请求
 39          interface: ::0

184          access-control:  0.0.0.0/0  allow      //允许所有客户端查询

373          domain-insecure: "example.com"          //绕开签名确认

549 forward-zone:                                   //定义转发
```

```
550         name: "."                                   //"."表示转发所有
551         forward-addr: 8.8.8.8                       //转发到 DNS 服务器地址
```

4、检查配置文件/etc/unbound/unbound.conf 语法是否正确

```
[root@server10 ~]# unbound-checkconf
```

5、重启 unbound.service

```
[root@server10 ~]#  systemctl restart unbound.service
```

6、配置防火墙允许 DNS 流量

```
[root@server10 ~]# firewall-cmd --permanent --add-service=dns
[root@server10 ~]# firewall-cmd --reload
```

7、确认的缓存服务器工作并检查 cache。

1）查看 cache 内容

```
[root@server10 ~]# unbound-control dump_cache
```

2）在客户机上(windows 或 linux)用 nslookup 查询

```
[root@desktop~]# nslookup
> server 10.11.21.235                          //缓存 DNS 服务 IP
Default Server:  [10.11.21.235]
Address:  10.11.21.235

> www.sina.com.cn
Server:  [10.11.21.235]
Address:  10.11.21.235

Non-authoritative answer:
Name:    ara.sina.com.cn
Addresses:  58.63.236.45, 58.63.236.46, 58.63.236.48, 58.63.236.49
          183.60.187.38, 183.60.187.39, 183.60.187.40, 183.60.187.41, 183.60.18
.42
          183.60.187.43, 58.63.236.37, 58.63.236.38, 58.63.236.39, 58.63.236.40
          58.63.236.43, 58.63.236.44
Aliases:  www.sina.com.cn, jupiter.sina.com.cn

>
```

在 Linux 客户机用 dig 命令查询。

```
[root@desktop~]# dig @10.11.21.235 A www.sina.com.cn
.......
www.sina.com.cn.                                3446    IN  CNAME
                                                jupiter.sina.com.cn.
jupiter.sina.com.cn.                            3401    IN  CNAME
                                                ara.sina.com.cn.
ara.sina.com.cn.                                41  IN  A   183.60.187.44
ara.sina.com.cn.                                41  IN  A   183.60.187.45
ara.sina.com.cn.                                41  IN  A   183.60.187.46
a.................
```

在 Linux 客户机用 host 命令查询。

```
[root@desktop~]# host -v -t A www.sina.com.cn
```

3）再一次查看 cache 内容

```
[root@server10 ~]# unbound-control dump_cache |grep sina
```

4）清理 cache 记录

```
[root@server10 ~]#unbound-control flush www.sina.com.cn
ok
```

5.3 配置本地的 DNS 服务器

1、修改配置文件/etc/unbound/unbound.conf

```
[root@server10 ~]# vim /etc/unbound/unbound.conf
         local-zone: "example.com." static
        local-data: "example.com.  86400 IN NS server10.example.com."
        local-data:    "example.com.    86400    IN    SOA    server10.example.com.
root.server10.com.  3 3600 1200 604800 86400"
       local-data: "server10.example.com.  86400 IN A 10.11.21.235"
        local-data: "www.example.com.  86400 IN A 10.11.21.235"
        local-data: "mail.example.com.  86400 IN A 10.11.21.235"
        local-data: "example.com.  86400 IN MX 10 mail.example.com."


        local-zone: "21.11.10.in-addr.arpa." static
        local-data: "21.11.10.in-addr.arpa.  86400 IN NS server10.example.com."
       local-data:  "21.11.10.in-addr.arpa.  86400  IN  SOA  server10.example.com.
root.server10.com.  3 3600 1200 604800 86400"
        local-data: "235.21.11.10.in-addr.arp 86400 IN PTR www.example.com."
```

注解：SOA 记录

86400 IN SOA server10.example.com. root.server10.com.

TTL 权威 DNS 服务器 邮件地址

3 3600 1200 604800 86400

序列号 从 DNS 刷新时间 从 DNS 重试延时 从 DNS 失效时间 无效解析记录缓存时间

2、检查配置文件/etc/unbound/unbound.conf 语法是否正确

```
[root@server10 ~]# unbound-checkconf
```

3、重启 unbound.service

```
[root@server10 ~]#  systemctl restart unbound.service
```

4、在客户机上(windows 或 linux)用 nslookup 查询

```
[root@server10 conf.d]# nslookup
> server 10.11.21.235
Default server: 10.11.21.235
Address: 10.11.21.235#53
> www.example.com
Server:                                    10.11.21.235
Address:                                   10.11.21.235#53

Name:                                      www.example.com
Address: 10.11.21.235
> mail.example.com
Server:                                    10.11.21.235
Address:                                   10.11.21.235#53

Name:                                      mail.example.com
Address: 10.11.21.235
>
```

在 Linux 客户机用 dig 命令查询。

```
[root@desktop~]# dig @10.11.21.235 A www.example.com
.......

;www.example.com.                                IN  A

```

```
;; ANSWER SECTION:
www.example.com.                        86400   IN   A    10.11.21.235
```

在 Linux 客户机用 host 命令查询。

```
[root@desktop~]# host -v -t A www.example.com
[root@desktop~]# host -v -t MX example.com
[root@desktop~]# host -v -t SOAwww.example.com
```

5.4 DNS 排错

DNS 服务器排错首先在服务器检查服务有没启动，再检查防火墙规则有没有允许 DNS 服务。如果都没问题，按下面方法排错。

1、查看名称解析的先后顺序

```
[root@server10 ~]#  cat /etc/nsswitch.conf
```

2、检查 hosts 文件

```
[root@server10 ~]# getent hosts www.example.com
[root@server10 ~]# gethostip www.example.com
```

3、客户机与服务器连接性问题

```
[root@server10 ~]# dig A  www.example.com
 ;connection timed out,no servers could be reached.
```

排错思路：

1) 客户机与服务网络连接问题。
2）服务器上 DNS 服务是否启动，并在相应 IP 监听，默认只在 127.0.0.1 上监听。
3) 服务器上 unbound.conf 配置文件中 forward-zone 是否设置。
3} 防火墙是否配置正确规则。
4）客户机上网卡属性设置 DNS 服务 IP 是否正确。

Dig 命令可以指定服务器查询

```
[root@server10 ~]# dig @服务器 IP  A  www.example.com
```

4、根据 DNS 响应代码排错

```
[root@server10 ~]# dig A  www.example.com

......
 ;; ->>HEADER<<- opcode :QUERY ,status: NOERROR,id 30523
......
```

注解：

DNS 服务器返回代码：

- NOERROR：没有错误
- SERVFAIL：DNS 服务器遇到问题
- NXDOMAIN：查找的名称在这个区域不存在
- REFUSED：由于策略限制 DNS 服务器拒绝客户机的查询请求

第六章　管理邮件传输

6.1 电子邮件系统概述

在人们的日常工作和生活中，发送和收取邮件已经成为相互之间沟通信息的常用方式，使用电子邮件系统为人们的生活带来了相当多的便利。

6.1.1　邮件服务器的角色及使用的协议

- MTA（Mail Transfer Agent，邮件传输代理）：一般被子称为邮件服务器软件。MTA 软件负责接收客户端软件发送的邮件，并将邮件传输给其他的 MTA 程序，是电子邮件系统中的核心部分。Exchange 和 Sendmail、Postfix 等服务器软件都属于 MTA。
- MUA（Mail User Agent，邮件用户代理）：一般被称为邮件客户端软件。MUA 软件的功能是为用户提供发送、接收和管理电子邮件的界面。
- MDA（Mail Delivery Agent，邮件分发代理）：MDA 软件负责在服务器中将邮件分发到用户的邮箱目录。

发送和接收邮件是电子邮件系统的两个基本功能。在电子邮件系统中，发送和接收邮件分别使用不同的网络协议。

- SMTP：主要用于发送和传输邮件，TCP 端口 25。
- POP：主要用于从邮件服务器中收取邮件，TPC 端口号为 110。

- IMAP4：主要用于从邮件服务器中收取邮件，TPC 端口号为 143。

6.1.2 常用的邮件服务器软件

1、商业邮件系统

- Exchange：Windows 系统中最著名的邮件服务器软件，也是微软公司的重量级产品，可以与活动目录等应用很好的结合在一起。使用 Windows 服务器平台构建电子邮件系统时，Exchange 服务器自然首选。
- Note/Domino：由 IBM 公司出品的商业电子邮件和办公协作软件产品，功能丰富、强大。集成性较好且提供跨系统平台的支持，给用户提供了广泛的选择。多应用于一些高校、政府部门、银行等较大的企业单位。

2、开源邮件系统

- Sendmail：对于运行在 Linux/unix 环境中的邮件服务器，Sendmail 无疑是资格最老的，目前仍然有许多企业的电子邮件系统是使用进行搭建的。运行的稳定性较好，但安全性欠佳。
- Qmail：另一款运行在 Linux/unix 环境中的邮件服务器，比 Sendmail 具有更好的执行效率，且配置、管理更加方便，很多商用电子邮件系统采用 Qmail 作为服务器
- Postfix：同样是运行在 Linux/unix 环境中的邮件服务器，由 Wietse 负责开发，其目的是为 Sendmail 提供一个更好的替代产品，在投递效率、稳定性、服务性能及安全性方面都有相当出色的表现。

14.2 Postfix 邮件服务器基础

1、Postfix 的相关目录

- /etc/postfix：该目录中包括 postfix 服务的主配置文件、各类脚本、查询表等。
- /usr/libexec/postfix：该目录中包括 postfix 服务的各个服务器程序文件。
- /var/spool/postfix：该目录中包括服务的邮件队列相关的子目录

  其中，每个队列子目录用于保存不同的邮件，例如：
  - Imcoming（传入）：刚接收到的邮件
  - Active（活动）：正在投递的邮件
  - Deferred（推迟）以前投递失败的邮件
  - Hold（约束）：被阻止发送的邮件
  - Corrupt（错误）不可读或不可分析的邮件
- /usr/sbin 该目录中包括服务的管理工具程序，这些程序文件名以 post 开头。其中，主要的几个程序文件及其作用于下。
  - postalias：用于构造、修改和查询别名表
  - postconf：用于显示和编辑 main.cf 配置文件夹
  - postfix：用于启动、停止 postfix，要求有 root 用户权限
  - postmap：用于构造、修改或者查询查询表
  - postqueue：用于管理邮件队列，一般用户使用
  - postsuper：用于管理邮件队列，要求有 root 用户权限

2、Postfix 的配置文件

Postfix 系统最主要的配置文件包括：/etc/postfix/main.cf 和/etc/postfix/master.cf。前者是 postfix 服务的配置文件，后者是 master 程序的配置文件。

在主配置文件 main.cf 中，可以设置的配置参数有三百多个。大部分的配置参数都被自动设置了默认值，如果在 main.cf 文件中没有对应的设置，那么服务器将使用默认值来启动及运行。大多数时候，只需要设置少数的几个配置参数，就可以满足一般邮件服务器的要求。

Postfix 系统提供的 postconf 工具可以用来辅助配置，不带任何选项的命令将打印出当前服务所使用的配置参数，而添加“-n”选项时将只列出不同于默认值的配置参数。

例、使用 postfix 工具查看当前服务所使用的配置参数。

```
[root@zx postfix-2.4.6]# postconf
```

例、使用 postconf 工具简化 main.cf 文件，只保留与默认配置不同的参数，提高易读性。

```
[root@zx postfix-2.4.6]# cd /etc/postfix
[root@zx postfix]# postconf -n > main2.cf
[root@zx postfix]# mv main.cf main.cf.bak
[root@zx postfix]# mv main2.cf main.cf
```

3、Postfix 的日志文件

Postfix 系统的日志文件位于“/var/log/maillog”，该文件中记录了服务器的运行状态信息（启动、出错及与其他 SMTP 服务器的会话信息等）。在安装调试邮件系统及日常维护的过程中，经常会使用带“-f”选项的 tail 命令来观察日志内容的实时更新。

例、使用 tail -f 命令跟踪/var/log/maillog 日志文件的内容变化。

```
[root@zx postfix]# tail -f /var/log/maillog
```

14.3 Postfix 邮件服务器配置

1、检查 postfix 否安装，没有安装则安装

```
[root@zx~]# rpm -qa |grep postfix
[root@zx~]# yum install -y postfix
[root@zx~]# systemctl start postfix
[root@zx~]#  systemctl enable postfix
```

2、编辑 vim /etc/postfix/main.cf 文件，调整 postfix 的基本运行参数。

```
113 #inet_interfaces = all
114 #inet_interfaces = $myhostname
115 #inet_interfaces = $myhostname, localhost
116 inet_interfaces = localhost
117
```

```
113 inet_interfaces = all     //去掉注释
116 #inet_interface = localhost    //注释掉
 重启服务
```

注解：

```
[root@zx ~]# vi /etc/postfix/main.cf
inet_interfaces = all                //服务器监听的 IP，缺省为 localhost
myhostname = mail.abc.com                  //服务器使用的主机名
mydomain = abc.com                          //服务器使用的邮件域
myorigin = $mydomain                         //外发邮件是发件人地址中邮件域名
mydestination = $mydomain, $myhostname       //可以接收的邮件地址的域名
home_mailbox = Mailbox                      //邮件存储位置
```

在上述配置中，将参数 mydestination 的值设置为$mydomain, $myhostname 后，则发送到 xxx@abc.com 和 xxx@mail.abc.com 的邮件都可以被服务器接收。各邮箱用户的邮件将被投递到各宿主目录下的 Maildir 子目录中。

用户的邮箱空间用于保存各自的电子邮件内容，在服务器中，支持如下两种最常见的邮箱存储方式（当指定的存储位置最后一个字符为“/”时，自动使用 Maildir 存储方式）

- Mailbox：将同一用户的所有邮件内容存储在同一个文件中，通常对应为目录“/var/spool/mail”中以用户名命名的文件。这种存储方式相对比较古老，在邮件数量度较多时查询和管理的效率最低。
- Maildir：这种方式使用目录结构来存储用户的邮件内容，每一个用户对应有一个文件夹，每一封邮件作为一个独立的文件保存。与 Mailbox 存储方式相比，此方式的存取速度和效率更好，而且对于管理邮件内容也更加方便。大多数较新的邮件服务器中，都采用了此邮件存储方式。

3、防火墙上允许 SMPT 服务，重启 postfix 服务。

```
[root@zx~]# firewall-cmd  --permanent  --add-service=smtp
[root@zx~]# firewall-cmd  --reload
[root@zx~]# systemctl restart postfix
```

3、添加邮件用户的帐号。

postfix 服务器默认使用本机中的系统用户作为邮件帐号。因此只需要添加 Linux 用户帐号即可。测试时，可以添加两个邮件帐号 xiaoqi 和 lisi，并为其设置密码。

```
[root@zx ~]# groupadd mailusers
[root@zx ~]# useradd -g mailusers -s /sbin/nologin xiaoqi
[root@zx ~]# useradd -g mailusers -s /sbin/nologin lisi
[root@zx ~]# passwd xiaoqi
[root@zx ~]# passwd lisi
```

4、测试邮件收发。

```
[root@zx ~]# echo this is a test email. | mail -s test lisi@mail.abc.com
[root@zx ~]# mail -u lisi
```

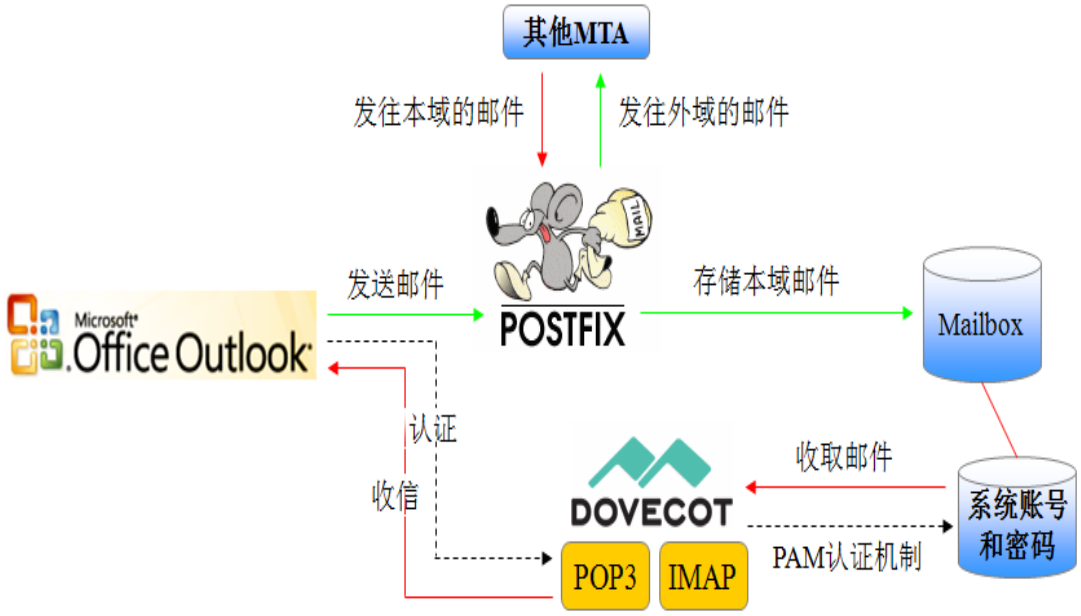
5、配置一个邮件别名到你的 MTA 中,如邮件发送到 admin 实际上是 xiaoqi 在接收邮件的。

```
 [root@zx ~]#  vim /etc/aliase
        admin:  xiaoqi
```

```
[root@zx ~]#newaliase                    （更新一下数据库）
```

6.4 基于 postfix 构建简单电子邮件系统

从本节开始，将以上一节中编译安装的系统为基础，学习如何构建一个完整的电子邮件系统。在整个案例中，将使用到 SMTP 服务器、POP3/IMAP 服务器、发信认证、Web 邮件界面等多种机制，如下图所示



在该电子邮件系统架构中，服务器所使用的相关配置环境如下。

- IP 地址：192.168.1.10/24。
- 邮件域：@abc.com
- 主机名：mail.abc.com
- 邮件帐号：使用本地系统用户
- POP3/IMAP 服务器软件：使用编译安装的软件包
- AMP 平台：使用编译安装的 Apache、MySQL 、PHP，对应的安装目录分别为 /usr/local/apach2、/usr/local/mysql、/usr/local/php5。
- 域名服务器：使用预先架设的 DNS 服务器，已做好 abc.com 域的解析设置，并为该域添加 192.168.1.10 的 MX 邮件交换记录。

6.4.1 构建 Dovecot 服务器

在整个电子邮件系统中，服务器作为一个 MTA 软件，也就是 SMTP 服务器，主要提供邮件发送和投递功能。若需要为 MUA 客户端软件提供收取邮件的功能，还得安装实现 POP3 或 IMAP4 协议的服务器程序。本节将学习通过编译安装的方式构建 Dovecot 收信服务器。

Dovecot 是一个安全性能较好的 POP3/IMAP 服务器软件，响应速度快而且扩展性好。Dovecot 也默认使用的系统用户，并通过 PAM 方式进行身份认证，通过认证的用户才可从邮箱中收取邮件。

1、rpm 安装 dovecot 软件包

```
[root@zx ~]# yum  -y install dovecot
```

2、配置 Dovecot 的运行参数

（1） 配置 dovecot.conf 文件

```
 [root@zx ~]# vi /etc/dovecot/dovecot.conf
protocols = pop3 imap                                    //支持邮局协议
login_trusted_networks = 192.168.1.0/24                 //指定允许登录的网段地址
```

（2） 编辑/etc/dovecot/conf.d 目录下 10-auth.conf 文件。

```
[root@zx ~]# vi /etc/dovecot/conf.d/10-auth.conf
disable_plaintext_auth = no                          //允许明文密码认证
```

（3） 编辑/etc/dovecot/conf.d 目录下 10-mail.conf 文件。

```
[root@zx ~]# vi /etc/dovecot/conf.d/10-mail.conf
  mail_location = mbox:~/mail:INBOX=/var/mail/%u          //指定邮箱位置及格式
```

（4） 创建相关目录。

```
[root@zx ~]# mkdir -p /home/lisi/mail/.imap/INBOX
[root@zx ~]# chown -R lisi:lisi /home/lisi
```

新建的用户自动创建目录

```
[root@zx ~]# vim /etc/skel/.bash_profile

  if [ ! -d ~/mail/.imap/INBOX ]; then
        mkdir -p  ~/mail/.imap/INBOX
  fi
```

3、启动 Dovecot 服务，并验证其监听的 TCP 端口（110 143）

Dovecot 服务的启动程序位于/usr/local/sbin/目录中，使用“-c”选项可以指定所使用的配置文件的位置。如果使用默认配置文件，则直接执行“dovecot”命令即可。

```
[root@zx ~]# systemctl start dovecot
[root@zx ~]# systemctl enable dovecot

[root@zx ~]# netstat -nutpl | grep dovecot
```

3、防火墙上允许 pop 和 imap 服务。

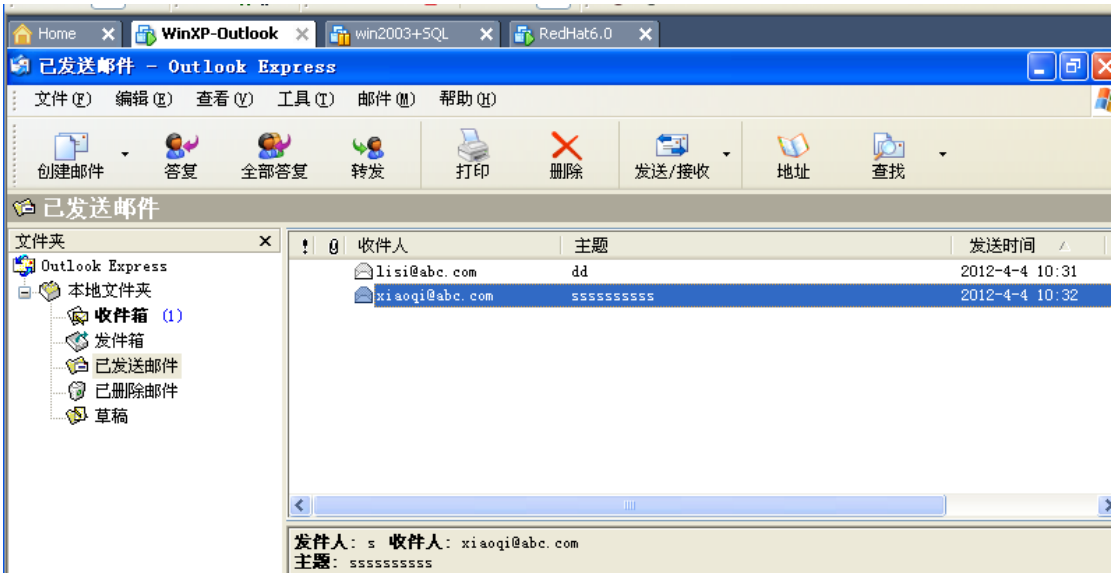
```
[root@zx~]# firewall-cmd  --permanent  --add-service=imaps
[root@zx~]# firewall-cmd  --permanent  --add-service=pop3s
[root@zx~]# firewall-cmd  --reload
```

6.4.2 使用 Outlook Express 等邮件客户端

在前一部分内容中，先后介绍了使用 telnet 命令进行发信、收信的测试过程，这种方式相对更加直接和有效。但是，对于绝大多数的普通电子邮件用户来说，图形界面的邮件客户端软件（MUA）要更受欢迎。本小节以 Windows 系统中自带的 Outlook Express 邮件客户端软件为例，来验证邮件服务器的发信、收信功能。

1、启动 Outlook Express 邮件客户端程序

在系统中，通过点击“开始”菜单“程序”，即可顺利打开程序。

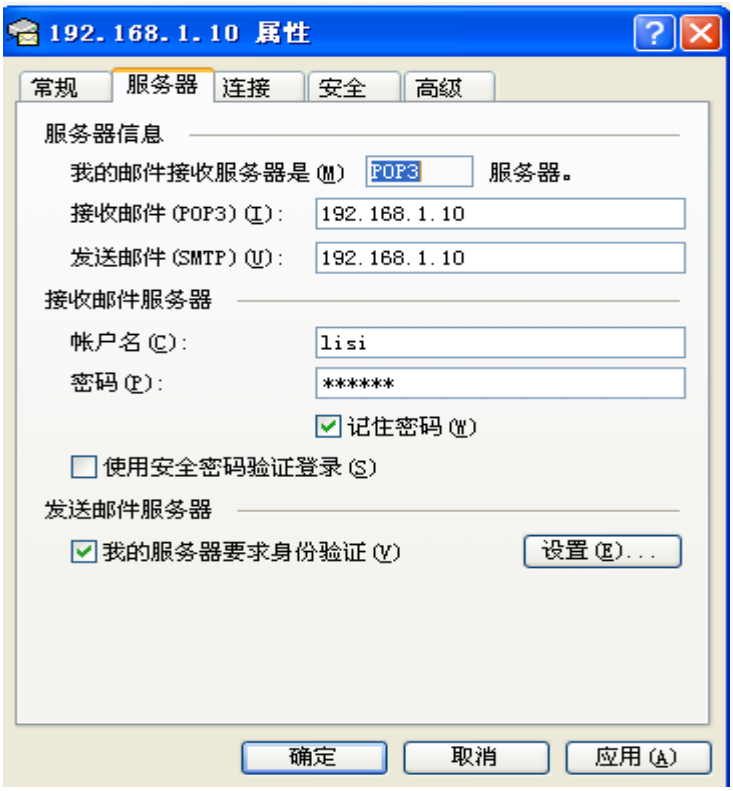


2、设置用户使用邮件帐号

在使用邮件客户端软件进行发信、收信之前，需要先设置一个邮件帐号，要设置的内容包括：邮件帐号名称、登录口令、SMTP 服务器的地址和 POP3 服务器的地址。例如：

- 邮件帐号名称：xiaoqi@abc.com
- SMTP 服务器地址：mail.abc.com
- POP3 服务器地址：mail.abc.com

选择“工具”菜单中的“帐户”，在出现的“Internet 帐户”对话框中选择“邮件”选项卡，单击“添加”按钮创建一个新的“邮件”用户帐号，根据向导提示完成用户 xiaoqi 的邮件帐号配置。使用前应确认帐号属性设置无误。帐号的密码也可以先不设置当接收或发送邮件时会提示用户进行输入。



3、验证邮件服务器的发信、收信功能。

6.5 Webmail 邮件

Webmail 指的是提供给邮件用户发信、收信使用的网页操作界面。通过访问邮件系统提供的 Web 界面，可以和邮件客户端软件实现类似的邮件管理功能。使用 Webmail 不需要预先设置邮件帐号的属性，使用更加简单、便捷，因此 在 Internet 中得到了广泛的应用。例如，新浪、网易、搜狐、Gmail 等站点都提供了通过浏览器访问邮箱的网页平台。

Postfix 邮件服务器支持使用多种 Webmail 系统，例如 OpenWebmail、Extmail、SquirreMail 等，本小节将使用较为流行的 SquirreMail 系统提供 WebMail 邮件界面。SquirreMail 可以与 Postfix 服务器、Dovecot 服务器很好的协作，面向邮件用户提供对邮件的发送、接收和管理操作。而普通邮件用户只需要访问 SquirrMail 的网页界面，就可以获得较完整的电子邮件服务。

从 RHEL6 开始未自带 SquirrelMail 了,用户可以从 http://www.squirrelmail.org 官方网站或其它网站下载 SquirrelMail 软件包和多国语言包。

1、yum 安装 http、php

```
[root@server yum.repos.d]# yum -y install httpd
[root@server yum.repos.d]# yum -y install php
```

2、依次安装程序包，中文语言包

```
[root@zx ~]#  tar zxvf squirrelmail-webmail-1.4.22.tar.gz -C /var/www/html/
[root@zx ~]# cd /var/www/html
[root@zx htdocs]# mv squirrelmail-1.4.22  webmail
```

```
[root@zx ~]# tar zxvf all_locales-1.4.18-20090526.tar.gz -C /var/www/html/webmail/
```

3、创建及调整数据目录、附件目录

```
[root@zx ~]# cd /var/www/html/webmail/

[root@zx webmail]# mkdir  attach
[root@zx webmail]# chown -R apache:apache attach/ data/
[root@zx webmail]# chmod 730 attach/
```

4、建立配置

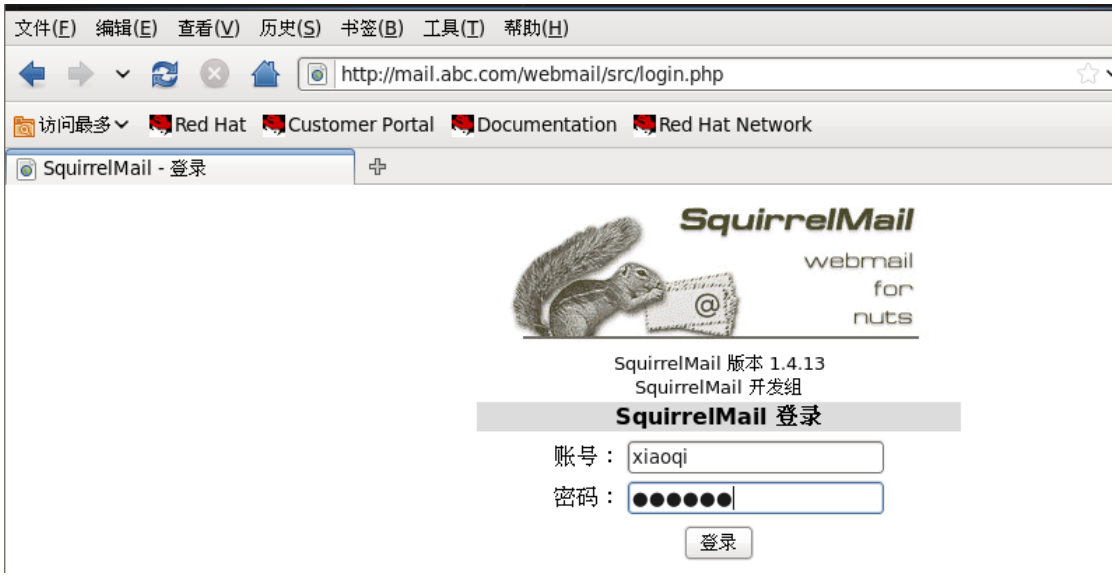
```
[root@zx webmail]# cp config/config_default.php  config/config.php
[root@zx webmail]# vi config/config.php
......
$domain = 'example.com';                                    //118 行:本地邮件域
$imap_server_type = 'dovecot' ;                             //231 行:使用 IMAP 的类型
$data_dir = '/var/www/html/webmail/data/';                  //499 行:数据目录的位置
$attachment_dir = '/var/www/html/webmail/attach';           //517 行:附件目录的位置
$squirrelmail_default_language = 'zh_CN';                   //1012 行:界面语言
$default_charset = 'zh_CN.UTF-8';                           //1027 行:默认字符集
```

5、修改 httpd 配置

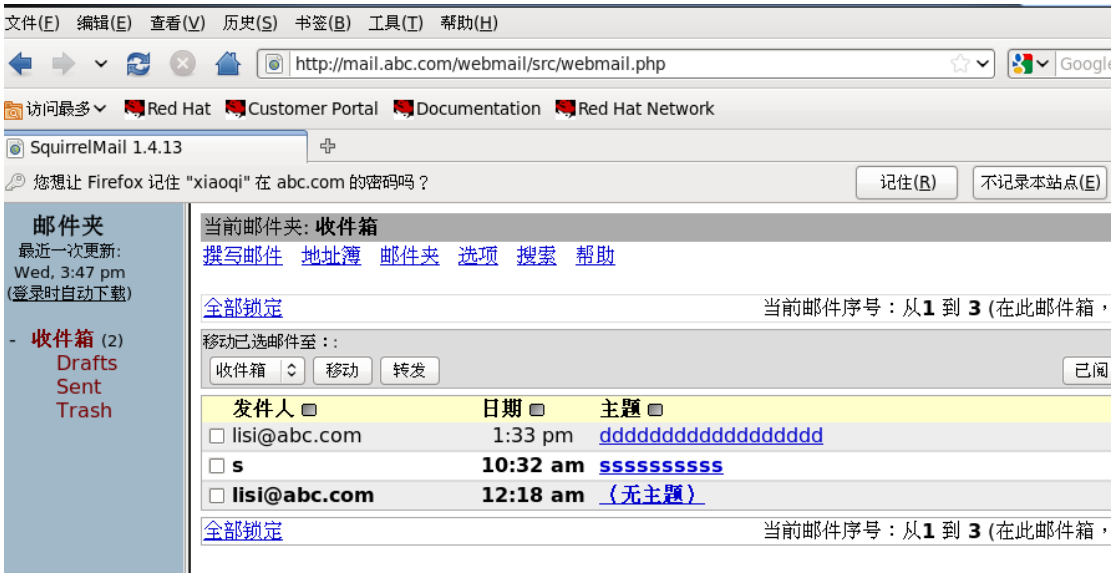
```
[root@zx ~]# vim  /etc/httpd/conf/httpd.conf
......
ServerName  mail.example.com:80
Alias  /webmail  "/var/www/html/webmail/"        /定义虚拟目录的别名和物理路径

[root@zx ~]# setsebool  -P  httpd_can_network_connect  on
[root@server webmail]# chcon -Rt httpd_sys_rw_content_t data
[root@server webmail]# chcon -Rt httpd_sys_rw_content_t attach
[root@zx ~]# firewall-cmd --permanent --add-service=http
[root@zx ~]# systemctl start httpd
[root@zx ~]# systemctl enable httpd
```

6、在浏览器中登录系统

访问 http://mail.example.com/webmail/，在登录页面中输入邮件用户帐号名及密码进行验证。



验证通过后可进入用户的邮箱管理界面，在该界面中可以完成发信、收信等电子邮件服务的基本操作。



➢

第七章　ISCSI

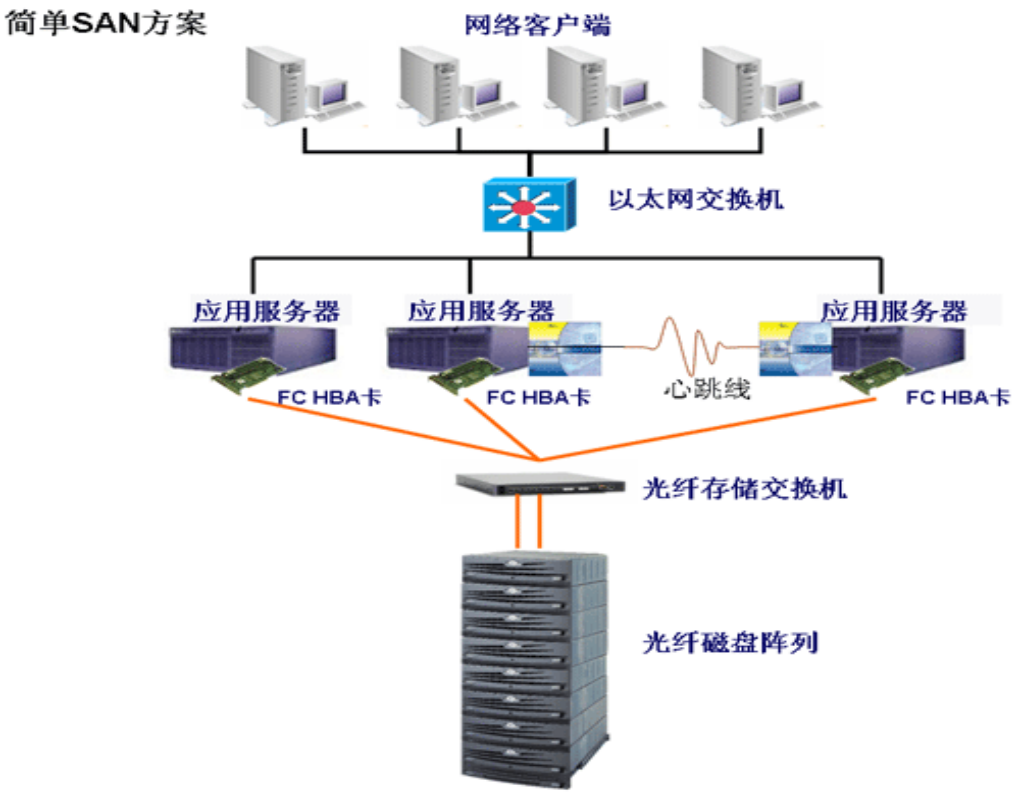
7.1 存储技术简介

1、SAN 的概念

SAN（Storage Area Network）存储区域网络，是一种高速的、专门用于存储操作的网络，通常独立于计算机局域网（LAN）。SAN 将主机和存储设备连接在一起，能够为其上的任意一台主机和任意一台存储设备提供专用的通信通道。SAN 将存储设备从服务器中独立出来，实现了服务器层次上的存储资源共享。SAN 将通道技术和网络技术引入存储环境中，提供了一种新型的网络存储解决方案，能够同时满足吞吐率、可用性、可靠性、可扩展性和可管理性等方面的要求。

1） FC-SAN

FC-SAN 通常 SAN 由磁盘阵列（RAID）连接光纤通道（Fibre Channel）组成（为了区别于 IP SAN，通常 SAN 也称为 FC-SAN）。SAN 和服务器和客户机的数据通信通过 SCSI 命令而非 TCP/IP，数据处理是“块级”（block level）。

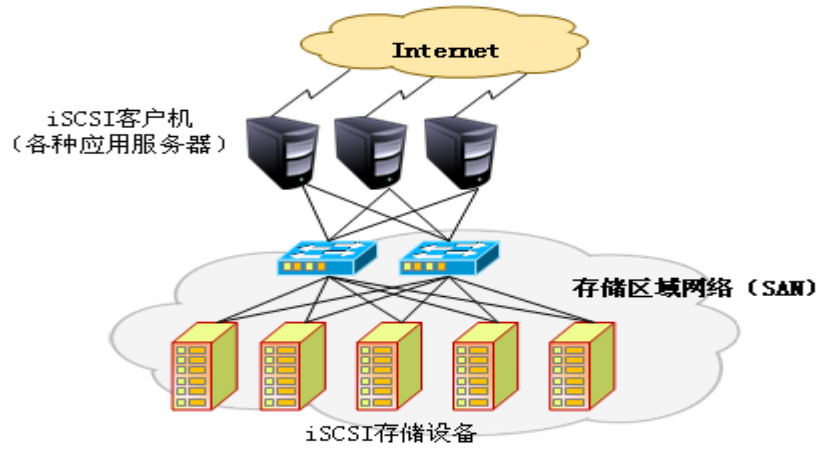


2） IP-SAN

我们简单来讲，IP-SAN（IP 存储）的通信通道是使用 IP 通道，而不是光纤通道，把服务器与存储设备连接起来的技术，除了标准已获通过的 iSCSI，还有 FCIP、iFCP 等正在制定的标准。而 iSCSI 发展最快，已经成了 IP 存储一个有力的代表。

iSCSI（互联网小型计算机系统接口）是一种在 internet 协议网络上，特别是以太网上进行数据块传输的标准。简单地说，iSCSI 可以实现在 IP 网络上运行 SCSI 协议，使其能够在诸如高速千兆以太网上进行路由选择，实现了 SCSI 和 TCP/IP 协议的连接。

iSCSI 是基于 IP 协议的技术标准，该技术允许用户通过 TCP/IP 网络来构建存储区域网（SAN）。而在 iSCSI 技术出现之前，构建存储区域网的唯一技术是利用光纤通道，但是其架构需要高昂的建设成本，远非一般企业所能够承受。iSCSI 技术的出现对于以局域网为网络环境的用户来说，它只需要不多的投资，就可以方便、快捷地对信息和数据进行交互式传输和管理。相对于以往的网络接入存储，iSCSI 的出现解决了开放性、容量、传输速度、兼容性、安全性等问题，其优越的性能使其自发布之日始便受到市场的关注与青睐。



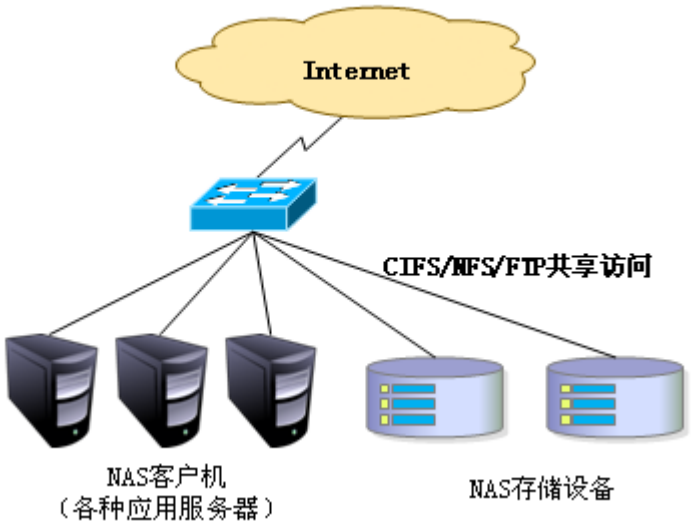
iSCSI 的技术优势：

- iSCSI 的基础是传统的以太网和 internet，同时能大大减少总体拥有成本。
- IP 网络的带宽发展相当迅速，1Gbps、10G 以太网早已大量占据市场，100Gbps 以太网也已整装待发。
- 在技术实施方面，iSCSI 以稳健、有效的 IP 及以太网架构为骨干，使忍受性大大增加。
- 简单的管理和布署，不需要投入培训，就可以轻松拥有专业的 iSCSI 人才。
- iSCSI 是基于 IP 协议的技术标准，它实现了 SCSI 和 TCP/IP 协议的连接，只需要不多的投资，就可以方便、快捷地对信息和数据进行交互式传输及管理。

- 完全解决数据远程复制及灾难恢复的难题。安全性方面，iSCSI 已内建支持 IPSEC 的机制，并且在芯片层面执行有关指令，确保安全性。

2、NAS（Network Attachmnet Storage，网络附加储存）

NAS 存储设备相当于一台独立的服务器，设备本身带有操作系统，也拥有网络接口，因此不依赖于其他服务器。在 NAS 设备中为客户机中分配存储空间时，通常采用共享文件夹的方式进行发布。存储的数据以文件的形式进行传输，采用 CIFS 或者 NFS（Network File System,网络文件系统）等协议，当然也可以支持 FTP 访问，如图所示。



3、DAS

DAS（Direct Attached Storage）直接附加存储，直接附加存储是指将存储设备通过总线（SCSI、PCI、IDE 等）接口直接连接到一台服务器上使用。DAS 购置成本低，配置简单，因此对于小型企业很有吸引力。

DAS 存在问题：

- 服务器本身容易成为系统瓶颈；
- 服务器发生故障，数据不可访问；
- 对于存在多个服务器的系统来说，设备分散，不便管理。同时多台服务器使用 DAS 时，存储空间不能在服务器之间动态分配，可能造成相当的资源浪费；
- 数据备份操作复杂。

FC-SAN，IP-SAN，NAS，DAS 区别：

	DAS	NAS	FC-SAN	IP-SAN
成本	低	较低	高	较高
数据传输速度	快	慢	极快	较快
扩展性	无扩展性	较低	易于扩展	最易扩展
服务器访问存储方式	直接访问存储数据块	以文件方式访问	直接访问存储数据块	直接访问存储数据块
服务器系统性能开销	低	较低	低	较高
安全性	高	低	高	低
是否集中管理存储	否	是	是	是
备份效率	低	较低	高	较高
网络传输协议	无	TCP/IP	Fibre Channel	TCP/IP

7.2 ISCSI 组件

1、 target：

ISCSI 存储资源，从 ISCSI 服务器连接。Target 必须分配一个唯一名字叫 IQN，一个 tartget 提供一个或多个块设备名叫 LUN。一个 ISCSI SERVER 能同时提供多个 target。

2、 initiator

ISCSI 客户端，一般是软件，必须分配一个唯一名字叫 IQN。

3、 LUN

Target 块设备的逻辑单元号。

4、 IQN

一个唯一名称用于标识 tartget 和 initiator。一般格式为：

Iqn.yyyy-mm.com.example:string

5、 node

用 IQN 标识的任一 target 或 initiator。

6、 ACL

用节点 IQN 设置控制 initiator 访问权限。

7、discovery

查询一个 target Server 列出配置 target。

8、 login

initiator 连上 target 或 LUN 开始使用块设备。

9、 portal

Target 监听的 IP 和端口，默认在所有 IP 的 3260/tcp 端口监听。

10、 TPG

Target Portal Group。用于设置 portal 和 ACL。

7.2 ISCSI target(服务端)配置

1、在服务器端添加硬盘，并创建一个新分区或逻辑卷（以下新分区为例/dev/sdb1）

2、安装 targetcli 软件包并启动 target 服务

```
[root@zx ~]# yum -y install targetcli
[root@zx ~]# systemctl start target
[root@zx ~]# systemctl enable target
```

3、设置防火墙允许 3260/tcp 端口。

```
[root@zx ~]# firewall-cmd --permanent --add-port=3260/tcp
[root@zx ~]# firewall-cmd --reload
```

3、运行 targetcli 进入交互模式

1）进入交互模式。

```
[root@zx ~]# targetcli
/> help
  - bookmarks action [bookmark]
  - cd [path]
  - clearconfig [confirm]
  - get [group] [parameter...]
  - ls [path] [depth]
  - pwd
  - refresh
  - restoreconfig [savefile] [clear_existing]
  - saveconfig [savefile]
  - sessions [action] [sid]
  - set [group] [parameter=value...]
  - status
/> ls
o- / ..................................................................... [...]
  o- backstores ........................................................... [...]
  | o- block ..................................... [Storage Objects: 0]
  | o- fileio .................................... [Storage Objects: 0]
  | o- pscsi ..................................... [Storage Objects: 0]
  | o- ramdisk ................................... [Storage Objects: 0]
  o- iscsi ......................................................... [Targets: 0]
```

2） 建立一个块存储 server10.disk1，使用上面新加硬盘/dev/sdb1。

```
//backstores> block/ create server10.disk1 /dev/sdb1
Created block storage object server10.disk1 using /dev/sdb1.
```

3) 建立一个名称 IQN 为 iqn.2014-11.com.example:remotedisk1。

```
/backstores> /iscsi create iqn.2014-11.com.example:remotedisk1
Created target iqn.2014-11.com.example:remotedisk1.
Created TPG 1.
```

4) 创建 ACL 允许 iqn.2014-11.com.example:desktop10 客户端连接。

```
/> /iscsi/iqn.2014-11.com.example:remotedisk1/tpg1/acls/ create
 iqn.2014-11.com.example:desktop10
Created Node ACL for iqn.2014-11.com.example:desktop10
```

5) 创建 LUN

```
/> /iscsi/iqn.2014-11.com.example:remotedisk1/tpg1/luns create
 /backstores/block/server10.disk1
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2014-11.com.example:desktop10
```

6) 配置 target 监听 IP 和端口（默认在所有接口 IP 的 3260/tcp 监听）。

```
/> /iscsi/iqn.2014-11.com.example:remotedisk1/tpg1/portals create 10.11.21.238
Using default IP port 3260
Created network portal 10.11.21.238:3260.
```

7) 确认配置并保存退出。

```
/> ls
o- / ......................................................................... [...]
  o- backstores .............................................................. [...]
  | o- block .................................... [Storage Objects: 1]
  | | o- server10.disk1 ..... [/dev/sdb1 (2.0GiB) write-thru activated]
  | o- fileio ................................... [Storage Objects: 0]
  | o- pscsi .................................... [Storage Objects: 0]
  | o- ramdisk .................................. [Storage Objects: 0]
  o- iscsi ............................................... [Targets: 1]
  | o- iqn.2014-11.com.example:remotedisk1 ..... [TPGs: 1]
  |   o- tpg1 ..................................... [no-gen-acls, no-auth]
  |     o- acls ..................................... [ACLs: 1]
  |     | o- iqn.2014-11.com.example:desktop10 ..... [Mapped LUNs: 1]
  |     |   o- mapped_lun0 ..... [lun0 block/server10.disk1 (rw)]
  |     o- luns ..................................... [LUNs: 1]
  |     | o- lun0 ..... [block/server10.disk1 (/dev/sdb1)]
  |     o- portals ..................................... [Portals: 1]
```

```
  |       o- 10.11.21.238:3260 ..................................... [OK]
  o- loopback ..................................................... [Targets: 0]
/> exit
```

7.3 ISCSI initiator(客户端)配置

1、yum 安装 iscsi-initiator-utils

```
[root@desktop10 ~]# yum -y install iscsi-initiator-utils
[root@sesktop10 ~]# systemctl start iscsi
```

2、修改 vim /etc/iscsi/initiatorname.iscsi

```
[root@esktop10~]#  vim /etc/iscsi/initiatorname.iscsi
Initiationame=iqn.2014-11.com.example:desktop10
```

3、发现并连接 iSCSI 设备

```
[root@esktop10~]#iscsiadm -m discovery -t st -p 10.11.21.238
10.11.21.238:3260,1 iqn.2014-11.com.example:remotedisk1
```

其中选项“-m discovery”表示发现/查找，选项“-t sendtargets”表示发布的 target(可缩写为“-t st”),选项“-p ip:port”用来指定服务器的 IP 地址、服务的监听端口。

查找结果中列出了对方发布的、本机可用的 iSCSI 对象名称。下一步就是根据此名称连接到指定的 iSCSI 存储对象，使用“-m node”选项表示管理目标为节点，选项“-l”或者“--login”表示连接/登录。

```
[root@localhost ~]# iscsiadm -m node -T iqn.2014-11.com.example:remotedisk1
--login
```

4、使用 iSCSI 存储设备

1）为磁盘规划分区

```
[root@localhost ~]# fdisk -l
......
Disk /dev/sdb doesn't contain a valid partition table  //新的 iSCSI 设备无分区表
```

```
[root@localhost ~]# fdisk /dev/sdb
 //省略分区过程
[root@localhost ~]# partprobe /dev/sdb
root@localhost ~]# fdisk -l /dev/sdb
......
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1017     4193060   83  Linux
```

2）创建文件系统（格式化）

```
[root@localhost ~]# mkfs -t xfs /dev/sdb1
```

3）挂载文件系统

```
[root@localhost ~]# mkdir /opt/data

[root@localhost ~]# blkid /dev/sdb1
/dev/sdb1: UUID="93846775-458c-405d-bd46-2de7712dfeb6" TYPE="xfs"

[root@localhost ~]# vi /etc/fstab
……
UUID=93846775-458c-405d-bd46-2de7712dfeb6  /opt/data xfs  defaults,_netdev  0 0
```

第八章　基于文件存储(一)----NFS

8.1 NFS 简介

NFS-Network FileSystem 的缩写，NFS 是由 Sun 开发并发展起来的一项用于在不同机器、不同操作系统之间通过网络互相分享各自的文件。NFS 是一种基于 TCP/IP 传输的网络文件系统协议。通过使用 NFS 协议，客户机可以像访问本地目录一样访问远程服务器中的共享资源。对于大多数负载均衡群集来说，使用 NFS 协议来共享数据存储是比较常见的做法，NFS 也是 NAS 存储设备必然支持的一种协议。

1、NFS exports

服务器端的设定都是在/etc/exports 这个文件中进行设定的，设定格式如下：

共享目录 主机名称 1 或者 IP1(参数 1，参数 2） 主机名称 2 或者 IP2（参数 3，参数 4）

上面这个格式表示，同一个目录分享给两个不同的主机，但提供给这两台主机的权限和参数是不同的，所以分别设定两个主机得到的权限。

可以设定的参数主要有以下这些：

- rw：可读写的权限；
- ro：只读的权限；
- no_root_squash:登入到 NFS 主机的用户如果是 ROOT 用户,他就拥有 ROOT 的权限，此参数很不安全，建议不要使用。
- root_squash：登入到 NFS 主机的用户如果是 ROOT 用户,他权限为 nfsnobody；
- all_squash：不管登陆 NFS 主机的用户是什么都会被重新设定为 nfsnobody。

- sync：资料同步写入存储器中。
- async：资料会先暂时存放在内存中，不会直接写入硬盘。
- insecure 允许从这台机器过来的非授权访问。
- anonuid：将登入 NFS 主机的用户都设定成指定的 user id,此 ID 必须存在于/etc/passwd 中。
- anongid：同 anonuid ，但是 group ID 就是了

例如可以编辑/etc/exports 为：

```
/tmp            *(rw,no_root_squash)
/home/public    192.168.0.*(rw)     *(ro)
/home/test      192.168.0.100(rw)
/home/linux     *.example.com(rw,all_squash,anonuid=40,anongid=40)
```

2、exportfs 命令：

如果我们在启动了 NFS 之后又修改了/etc/exports，是不是还要重新启动 nfs 呢？这个时候我们就可以用 exportfs 命令来使改动立刻生效，该命令格式如下：

exportfs [-aruv]

-a ：全部 mount 或者 unmount /etc/exports 中的内容

-r ：重新 mount /etc/exports 中分享出来的目录

-u ：umount 目录

-v ：在 export 的時候，将详细的信息输出到屏幕上。

8.2 NFS 服务器的配置案例

一、实验环境

服务器 IP 地址为 192.168.1.2 要实现 1 网段的所有客户端都能操作其 /share 文件夹

二、实验步聚

（一）服务端设置

1、启动 nfs-server 服务

```
[root@localhost ~]# systemctl  start   nfs-server
[root@localhost ~]# systemctl  enable  nfs-server
```

2、创建共享目录/share

```
[root@localhost ~]# mkdir -m 1777  /share
```

3、修改 NFS 服务配置文件/etc/exports

```
[root@localhost ~]# vim /etc/exports
/share    192.168.1.0(rw)

[root@localhost ~]# exportfs -r
```

4、配置防火墙

```
[root@localhost ~]# firewall-cmd --permanent --add-service=nfs
[root@localhost ~]# firewall-cmd --reload
```

（二）客户端配置

1）安装 rpcbind,nfs-utils 软件包，并启动 rpcbind 服务

```
[root@localhost ~]# yum -y install rpcbind nfs-utils
```

NFSv2 和 NFSv3 客户机也可以使用 showmount 查看 NFS 服务器端共享了哪些目录，查询格式为“showmount –e 服务器地址”。

```
[root@localhost ~]# showmount -e 192.168.1.2
```

2）手动挂载 NFS 共享目录

```
[root@localhost ~]# mount  192.168.1.2:/share  /var/www/html
```

3）fstab 自动挂载设置

```
[root@localhost ~]# vi /etc/fstab
192.168.1.2:/share   nfs  /var/www/html  defaults,_netdev  0  0
```

防火墙设置：

由于 nfs 服务需要开启 mountd,nfs,nlockmgr,portmapper,rquotad 这 5 个服务，需要将这 5 个服务的端口加到 iptables 里面

而 nfs 和 portmapper 两个服务是固定端口的，nfs 为 2049，portmapper 为 111。其他的 3 个服务是用的随机端口，那就需要

先把这 3 个服务的端口设置成固定的。

5、查看当前这 5 个服务的端口并记录下来 用 rpcinfo -p

这里显示 nfs 2049, portmapper 111, 将剩下的三个服务的端口随便选择一个记录下来

6、将这 3 个服务的端口设置为固定端口

vim /etc/services

在文件的最后一行添加：

```
mountd  976/tcp
mountd  976/udp
rquotad  966/tcp
rquotad  966/udp
nlockmgr 33993/tcp
nlockmgr 33993/udp
```

保存并退出。

7、重启下 nfs 服务。 service nfs restart

8、在防火墙中开放这 5 个端口

第八章 基于文件存储之二---samba

8.1 Samba 服务基础

在 Windows 网络环境中，用户可以通过“网上邻居”找到其他主机并访问其中的共享资源，主机之间进行文件和打印机共享时是通过微软公司自己的 SMB/CIFS 网络协议实现的。SMB(Server Message Block，服务消息块)和 CIFS（Common Internet File System，通用互联网文件系统）协议是微软的私有协议，在 Samba 项目出现之前，并不能直接与 Linux/Unix 系统进行通信。

Samba 是著名的开源软件项目在 Linux/Unix 系统中实现了 SMB/CIFS 网络协议，因此使得跨平台的文件共享变得更加容易。在部署 Windows、Linux、Unix 混合平台的企业环境时，使用 Samba 可以很好的解决不同系统之间的文件互访问题。

Samba 服务器主要提供以下两个服务程序：

- smbd 为客户机提供服务器中共享资源（目录和文件等）的访问。
- nmbd 提供基于 NetBIOS 主机名称的解晰，为 Windows 网络中的主机进行名称解析。
- mbd 服务程序监听 TCP 协议的 139 端口（SMB）、445 端口（CIFS），nmbd 服务程序监听 UDP 协议的 137-138 端口（NetBIOS）。
- 通过/etc/init.d/smb 脚本文件可以控制 Samba 服务的启动与终止，无需单独运行 smbd 或 nmbd。

8.2 Samba 服务器的配置

1、安装 Samba

```
[root@zx~]# yum install samba
```

2、准备共享目录

```
[root@zx~]# mkdir /share
[root@zx~]# semanage fcontext -a -t samba_share_t  '/share(/.*)?'
[root@zx~]# restorecon -vvRF /share
```

注意：

Samba 文件标签也可设为 public_content_t 和 public_content_rw_t 。允许读写的 public_content_rw_t 还要设置布尔值 smbd_anon_write.

3、Samba 主配置文件

Samba 服务的配置文件位于/etc/samba/smb.conf。

```
[root@zx Packages]# grep -v "^#" /etc/samba/smb.conf | grep -v "^;" | grep -v "^$"
[global]
    workgroup = MYGROUP
    server string = Samba Server Version %v
    log file = /var/log/samba/log.%m
    max log size = 50
    security = user
    passdb backend = tdbsam
    load printers = yes
    cups options = raw
[homes]
    comment = Home Directories
    browseable = no
    writable = yes
[printers]
    comment = All Printers
    path = /var/spool/samba
    browseable = no
    guest ok = no
    writable = no
    printable = yes /var/lib/dhcpd/dhcpd.leases
```

从以上内容可以看出，smb.conf 配置文件默认包括以下三部分内容。

- [global]全局设置：这部分配置项的内容对整个 Samba 服务器有效。
- [home]用户目录共享设置：设置对应 Samba 用户宿主目录的默认共享，即当用户访问

服务器中与自己用户名同名的共享文件夹时，默认会映射到自己的宿主目录。

- [printers]共享设置：如果需要共享打印机则在这部分配置。

下面列出 smb.conf 文件中常见的一些配置项及其含义说明。

配置项	说　　明
workgroup	设置服务器所在工作组名称，例名“WORKGROUP”
server string	设置服务器的说明文字，用于描述 Samba 服务器
security	设置服务器的安全级别，可以设置以下四个中的一个：share（可匿名访问）、user（需本服务器验认用户名及密码）、server（由另外一台服务器验证）、domain（由 Windows 域控验证）
log file	设置 Samba 服务器的日志文件，默认设置为 /var/log/samba/%m.log，表示日志文件保存到/var/log/samba/目录中，按每个客户端建一个日志文件，%m 变量表示客户端的 IP 或主机名。
max log size	设置日志文件最在容量，默认为 50KB
comment	设置对共享目录的注释、说明信息
path	设置共享目录在服务器中的文件夹的路径
browerable	设置该共享目录在“网上邻居”中是否可见，设为 no 时相当于隐藏共享目录
guest ok	设置是否所有人都可以访问共享目录，与 public 配置项作用相同
writable	设置该共享目录是否可写，与 read only 作用相反
valid users	设置可以访问的用户或组

1）添加用户授权设置

在 smb.conf 主配置文件中，共享目录的用户授权设置主要由 valid users、write list 配置项指定，同时要取消公开访问的设置（即设为 public=no）。

需要授权多个用户时以空格或逗号分隔，注意不要使用帐户别名，而应使用实际的 Samba 用户名。授权一个用户组时使用“@组名”的形式，组内的每个用户都需要有对应的 Samba 用户。

若需要设置 Samba 用户在共享目录中建立的子目录、文件的默认权限，可以使用 directory mask、create mask 配置项。

例、为 movie 共享添加用户授权，只允许 tom 或 root 组的用户访问，root 用户有写入权限。通过访问共享建立的文件夹的默认权限为 744，文件的默认权限为 600。

```
[root@zx~]# vi /etc/samba/smb.conf
[global]
    workgroup =  WORKGROUP
    security = user                    //将安全级别修改为 user,本服务器验证
[movie]                                 //在文件末尾添加以下内容
    comment = Public share with movie files        //注释
    path = /var/public/movies                      //共享目录对应文件夹
    public = no                                    //所有非公开
    read only = no
    valid users = tom , @root                 //合法用户为 tom 和 root 组内的用户
    write list = root                         //root 用户有写入的权限
```

```
    directory mask = 0700                    //建立文件夹的默认权限
    create mask = 0600                       //建立文件的默认权限
```

控制 Samba 用户访问共享目录的读取、写入权限时，注意要满足一个前提条件，即与该 Samba 用户同名的系统用户对服务器中的共享目录也必须有相应的读取、写入权限。

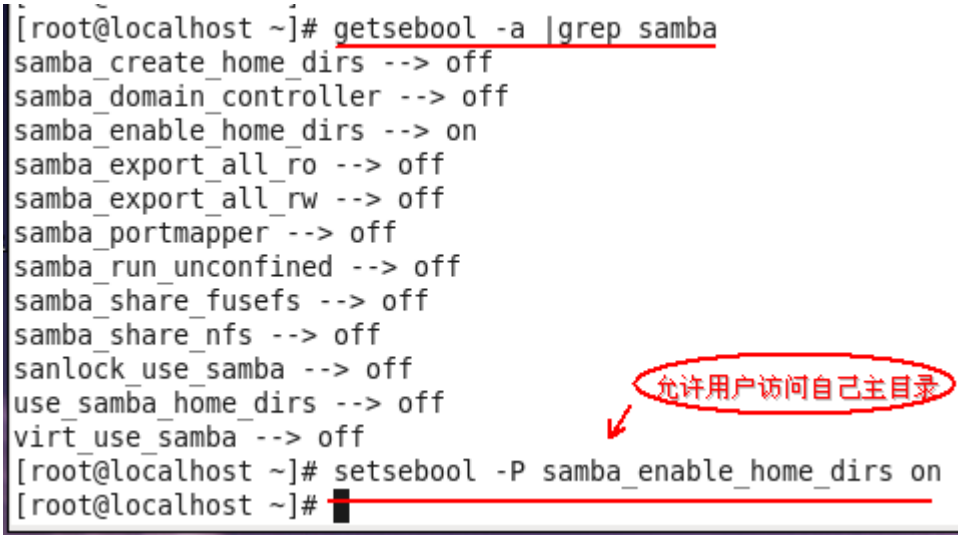
2）添加客户端地址授权设置

在 smb.conf 文件中，使用 host allow 配置项可以设置仅允许访问共享的客户地址，使用 host deny 配置项可以设置仅拒绝访问共享的客户机地址，两者选其一。配置参数可以选用主机名或 IP 地址、网络地址的形式，多个地址之间可以用空格或逗号分隔。

例、仅允许 192.168.2.0/24 、172.16.0.0/16 网段内的客户机访问 Samba 服务器。

```
[root@zx~]# vi /etc/samba/smb.conf
[global]
    Host allow = 192.168.2.   172.16.
```

3）允许用户访问自己的主目录，



Samba 服务器提供了一个配置文件的检查工具，testparm 程序，使用 testparm 工具可以对 smb.conf 配置文件的正确性进行检查，如果发现有错误将会进行提醒。

4、建立 samba 用户

为了区别于系统用户，通常将用于访问共享的用户称为 Samba 用户, Samba 服务器使用独立的帐号数据库文件,但是在建立 Samba 用户帐号时需要确保有对应的系统用户帐号存在，Samba 用户的密码可以与系统用户的密码不同。

使用带 –a 选项的 smbpass 命令可以添加 Samba 用户帐号，命令中需要指定要添加的用户名作为参数。第一次执行该命令时会自动创建帐户数据库文件（默认位于

/var/lib/samba/private 目录，文件名为 passdb.tdb）。

例、为 Samba 服务添加一个名为 tom 的用户帐号，并将 root 用户添加为 Samba 用户。

```
[root@zx~]# useradd tom                  //先添加系统用户
[root@zx~]# smbpasswd -a tom             //添加 tom 为 Samba 用户
[root@zx~]# smbpasswd -a root            //添加 root 为 Samba 用户
```

smbpasswd 命令除了可以添加 samba 用户帐号外，还可以结合不同的命令选项完成不同的帐号维护工作。

- -h ：显示命令的帮助信息
- -a：添加指定的 Samba 用户帐号
- -d：禁用指定的用户帐号
- -e：启用指定的用户帐号
- -x：删除指定的用户帐号
- 不使用任何命令选项时可以用于修改 Samba 用户的密码。

5、启动 samba 服务

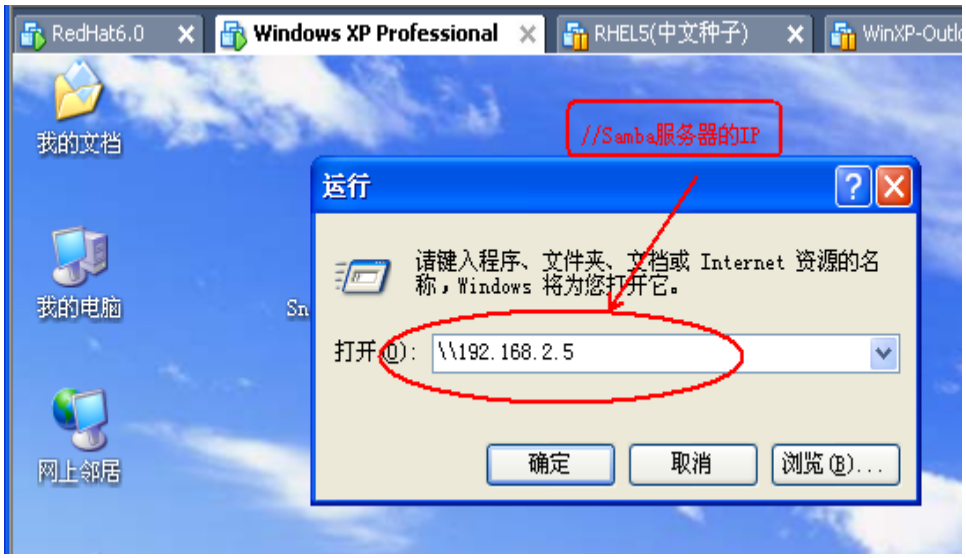
```
[root@zx~]# systemctl start smb nmb
[root@zx~]# systemctl enable smb nmb
[root@zx~]# firewall-cmd --permantent --add-service=samba
```

```
[root@zx~]# netstat -anptu | grep mbd
```

从上述结果可以看到，smbd 服务程序监听 TCP 协议的 139 端口（SMB）、445 端口（CIFS），nmbd 服务程序监听 UDP 协议的 137-138 端口（NetBIOS）。

8.2 在客户端访问 Samba 文件共享

1、使用 Windows 客户端访问文件共享服务



2、使用 Linux 客户端访问文件共享服务

1）使用 smbclient 工具登录到服务器

smbclient 是 Samba 服务器的字符模式客户端工具，使用的形式与 ftp 命令非常类似，可以登录到 samba 服务器，进行上传、下载等操作。

使用带 –l 选项的 smbclient 命令可以显示指定 Samba 服务器中的共享资源列表，未指定用户帐号时将以匿名登录的方式（提示输入密码时直接回车即可）查询服务器资源。

例、查看 Samba 服务器 192.168.2.5 中的共享资源列表。

```
[root@zx~]# smbclient  -l 192.168.2.5
```

访问共享资源时，添加 //192.168.2.5/movie 形式的参数指定 Samba 服务器的地址及共享目录名。如果访问的是带用户验证的共享目录，还需添加 –U 选项指定使用的 Samba 用户帐号，在登录过程中会提示用户输入口令。

例、以仿 ftp 客户端的方式登录访问 Samba 服务器中的共享目录。

```
[root@zx~]# smbclient  //192.168.2.5/movie -U tom
```

成功登录 Samba 服务器以后，会出现 smb:/> 的命令提示符，此时可能输入相应的命令进行查看目录，下载、上传文件等操作。smbclient 环境中的命令与 ftp 命令交互环境中的类似，例如 ls 列目录、pwd 查看当前目录、get 和 mget 用于下载文件、put 和 mput 用于上传文件……，可以使用“？”或 help 命令查看各种交互命令的在线帮助信息。

2）使用命令将共享目录挂载到本地

利用 smbclient 客户端工具可以非常方便的登录 smbclient 服务器，但是只有将文件下载到本地以后才能查看文件内容。若使用 mount 命令将共享资源挂载到本地，则无需下载，从挂载点目录中即可直接查看共享文件的内容，使用更加便捷。

用 mount 命令挂载 Samba 服务器共享资源时，同样要使用 //192.168.2.5/movie 形式的参数指定 Samba 服务器的地址及共享目录名。如果访问的是带用户验证的共享目录，还需添加 –o username= 选项指定使用的 samba 用户帐号，在登录过程中会提示用户输入口令。

例、将服务器具 192.168.2.5 中的共享目录挂载到本地的文件夹 /media/smbdir 中，以 tom 用户帐号进行验证。

```
[root@zx~]# mkdir -p /media/smbdir
[root@zx~]# mount -o username=tom //192.168.2.5/share /media/smbdir
Password:
```

如果服务器没有给该共享用户可写权限，则无法在载点创建目录或文件。

例、测试在挂载点目录中的读写权限。

```
[root@zx~]# cd /media/smbdir
[root@zx~]# mkdir dir1 ; touch file1
mkdir:无法创建目录 dir1: 权限不够
touch:无法触碰 file1: 权限不够
```

使用有可写权限的共享用户（如 root）重新挂载共享资源，才能在挂载点目录中写入。

例、以 Samba 用户 root 重新挂载共享资源，并测试在挂载点目录中的读写权限。

```
[root@zx~]# cd
[root@zx~]# umount /media/smbdir/
[root@zx~]# mount -o username=root //192.168.2.5/movie /media/smbdir
[root@zx~]# cd /media/smbdir
[root@zx~]# mkdir dir1 ; touch file1
```

8.3 实验案例一 Samba 服务器配置

一、实验环境

在公司局域网络中实现文件资源共享的 Samba 服务，以便于 windows 主机和 Linux 主机能互访共享资源。

二、需求描述

1、在 Linux 服务器上建一个需要验证的共享文件夹

三、实验步骤

1、安装 Samba

```
[root@zx~]# yum install samba samba-client
```

3、准备共享目录

```
[root@zx~]# mkdir /share
[root@zx~]# semanage fcontext -a -t samba_share_t  '/share(/.*)?'
[root@zx~]# restorecon -vvRF /share
```

3、Samba 主配置文件

```
[root@zx ~]#   vi /etc/samba/smb.conf
   [global]
```

```
      workgroup = mygroup
      security = user
     host allow = 192.168.2.   172.16.

  [share]
    comment = public share
    path = /share
    public = yes
    browseable = yes


[root@zx ~]# testparm
```

4、建立 samba 用户

```
[root@zx~]# useradd tom                  //先添加系统用户
[root@zx~]# smbpasswd -a tom             //添加 tom 为 Samba 用户
[root@zx~]# smbpasswd -a root            //添加 root 为 Samba 用户
```

5、启动 samba 服务

```
[root@zx~]# systemctl start smb nmb
[root@zx~]# systemctl enable smb nmb
[root@zx~]# firewall-cmd --permantent --add-service=samba
```

```
[root@zx~]# netstat -anptu | grep mbd
```

6、在 Windows 客户端验证

1）以 tom 或 player 用户只有读的权限。

如果 windows 客户端出现同用户多次连接的错误，删除访问摘要缓存,或者注销系统。

C:/ net use * /del

2）如果要能写文件，按如下修改，重启服务。

```
[share]
   comment = public share
   path = /share
   public = yes
   read only = no
```

7、在 Linux 客户端上验证

把 Samba 服务上的共享目录挂载到本地目录/media/smbdir

```
# yum -y install cifs-utils
# mkdir –p /media/smbdir
# mount –o username=tom    //192.168.2.5/share    /media/smbdir
```

8.4 Samba 多用户挂载

默认情况下，在客户端 samba 挂载后，挂载的证书决定挂载后的权限。一个新的 multiuser 选项可以分离每个用户的访问的证书。

1、安装 cifs-utils

```
[root@desktop10~]# yum install cifs-utils
```

2、创建挂载目录

```
[root@desktop10~]# mkdir /mnt/share
```

3、建立 tom 一个证书文件

```
[root@desktop10~]# vim /root/smb-multiuser.txt
```

```
username=tom
password=123456
```

4、编辑/etc/fstab 文件，挂载时用 multiuser 选项

```
[root@desktop10~]# vim /etc/fstab
//192.168.2.5/share /mnt/share cifs
credentials=/root/smb-multiuser.txt,multiuser,sec=ntlmssp  0 0

[root@desktop10~]# mount /mnt/share
```

5、切换 tom 用户，访问共享

```
[root@desktop10~]# su - tom
[tom@desktop10~]$  touch /mnt/share/test.txt
 Touch:cannot toutch:test.txt :permission denied
[tom@desktop10~]$  cifscreds add 192.168.2.5
 Password:123456
 [tom@desktop10~]$  touch /mnt/share/test.txt
```

第八章　基于文件存储(三)----FTP

在上一章中，大家已经学习了如何构建 Samba 文件共享服务，由于 Samba 文件共享通常要用到局域网协议 NetBIOS 的名称解析功能，因此多数时候只在内部网络中使用，本章将学习在局域网和广域网都适用的别一种文件服务器——FTP 文件传输服务器。

8.1　FTP 服务概述

FTP（File Transfer Protocol，文件传输协议）是 Internet 上使用非常广泛的一种协议，

它建立在传输层 TCP 协议之上。利用 FTP 可以给用户提供上传和下载文件的服务。

FTP 服务器就是在互联网或局域网中提供 FTP 服务的计算机，它可以是专用服务器，也可以是个人计算机。启动 FTP 服务后，用户可以连接到服务器下载文件，如果权限允许，用户也可以把文件上传到 FTP 服务器。

FTP（File Transfer Protocol 文件传输协议）是典型的 C/S 结构的应用层网络协议，需要有相应的客户端和服务器软件才能进行文件传输。对于 FTP 服务，可以从以下几个方面进行了解。

1、FTP 登录方式

FTP 服务器可以有两种登录方式，一种是匿名登录，另一种是使用授权账户登录。

- 匿名登录：一般匿名登录只能下载 FTP 服务器的文件，而且传输速度相对要慢一些。当然，这需要在 FTP 服务器上进行设置。针对这类用户，在 FTP 服务器上需要加以限制，不宜开启过高的权限，带宽应尽可能小。
- 授权账户登录：需要管理员将账户与密码告诉用户。管理员对这些账户进行设置，例如他们能访问到哪些资源，下载与上传速度如何等。

2、FTP 连接及传输模式

FTP 服务器默认使用 TCP 协议的 20、21 端口与客户端进行通信，21 端口用于建立控制连接，并传输 FTP 控制命令，20 端口用于建立数据连接，并传输文件数据。

根据 FTP 服务器在建立数据连接过程中的主、被动关系，FTP 数据连接分为主动模式和被动模式，二者主要区别于下。

- 主动模式：这种模式下服务器主动发起数据连接，首先由客户端向服务器的 21 端口建立 FTP 控制连接，当需要传输数据时，客户端以 PORT 命令告知服务器“我打开了某个端口，你过来连接我”，于是服务器从 20 端口向客户端的该端口发送请求并建立数据连接。
- 被动模式：这种模式下的服务器被动等待数据连接，如果客户机所在的网络的防火墙禁止主动模式连接，通常会使用被动模式。首先由客户端口向服务端的 21 端口建立 FTP 控制连接，当需要传输数据时，服务器以 PASV 命令告知客户端“我打开了某个端口，你过来连接我”，于是客户端向服务器的该端口（非 20）发送请求并建立数据连接。

客户端与服务器建立好数据连接以后，这可以根据从控制连接中发送的 FTP 命令，上传或下载文件了。在传输文件时，根据是否进行字符转换，分为文本模式和二进制模式

- 文本模式：又称为 ASCII，这种模式在传输文件时使用 ASCII 字符序列，一般只用于纯文本文件传输。
- 二进制模式：又称为 Binary 模式，这种模式不会转换文件中的字符序列，更适合传输程序、图片等文件。

使用二进制模式比文本模式更有效率，大多数 FTP 客户端工具可以根据文件类型自动选择文件传输模式，而无需用户手工指定。

3、FTP 用户类型

使用 FTP 客户端软件访问服务器时，通常要用到一类特殊的用户帐号，其用户名为 ftp 或 anonymous，提供任意密码（包括空密码）都可以通过服务器的验证，这样的用户称为“匿名用户”。匿名用户一般用于提供公共文件的下载，如一些提供免费软件，学习资料下载的站点。

除了不需要密码验证的匿名用户以外，许多 FTP 服务器软件可以直接使用服务器的系统用户账号进行验证，这些用户通常被称为“本地用户”在 RHEL6 系统中，匿名用户出有对应的本地用户帐号 ftp，但对于 vsftpd 服务来说，本地用户指的是除了匿名用户以外的其他系统用户。

有些 FTP 服务器软件还可以维护一份独立的用户数据库文件，而不是直接使用本地用户帐号，这些位于独立数据库文件中 FTP 用户帐号，通常被称为“虚拟用户”。使用 FTP 虚拟用户可以提供更好的安全性。

4、FTP 服务器软件的种类

在 Windows 操作系统中，微软自带的 IIS 服务器可以实现简单的 FTP 服务器功能，而是较为流行的 FTP 服务器软件。在 Linux 操作系统中可以选择使用 Wu-ftpd、Proftpd 和 vsftpd 等多个 FTP 服务器软件，这些 FTP 服务器都有各自的特点。

Wu-ftpd 服务器是早期较为流行的 FTP 服务器程序之一，功能强大且稳定性也较出色。但是服务器的组织较为零散，安全性比 Proftpd 和 vsftpd 要差一点。

Proftpd 服务器在 Wu-ftp 的基础上进行了改进，配置起来也相对比较容易，执行效率和安全性方面出得到了很大的提高。

vsftpd 服务器在设计之初主要基于安全属于进行改进，这也是软件名称的由来——Very Secure FTP Daemon。除了安全性外，vsftp 在速度和稳定性方面的表现也很突出。根据 ftp.redhat.com 服务器反映的数据，vsftpd 可以支持 150000 个用户并发连接。

本章将选用 vsftp 为例，讲解 FTP 服务器的构建过程

8.2 vsftpd 服务基础

vsftpd 服务的配置文件默认位于/etc/vsftpd 文件夹中，主要包括用户控制列表文件(ftpusers、user_list)和主配置文件（vsftpd.comf），下面对这两类配置文件分别进行介绍。

1、用户控制列表文件 ftpusers 和 user_list

Ftpusers 和 user_list 文件中均包含一份 FTP 用户名的列表，两个文件虽然都用于 FTP 用户的控制，但是具体作用存在一些差别。

- ftpusers 文件：该文件中包含的用户帐户将被禁止登录 vsftpd 服务器，不管该用户是否在文件中 user_list 出现。通常将 root、bin、daemon 等特殊用户列在该文件中，禁止用于登录 FTP 服务。
- user_list 文件：该文件中包含的用户帐户可能被禁止登录，也可能被允许登录，具体在主配置文件 vsftpd.conf 中决定。当存在 userlist_enable=YES 的配置项时，user_list 文件生效，如果配置 userlist_deny=YES，则仅禁止列表中的用户帐户登录，如果配置 userlist_deny=NO，则仅允许列表中的用户账户登录。

ftpusers 文件为 vsftpd 服务器提供了一份用于禁止登录的 FTP 用户列表，user_list 而文件提供了一份可灵活控制的 FTP 用户列表。二者相互结合，为 FTP 用户控制提供了良好的基础。

2、主配置文件 vsftpd.conf

在 vsftpd 的主配置文件夹中，配置行采用“配置项=参数”的格式。下面列出 vsftpd.conf 文件中常见的一些配置项以及其含义说明。

作用范围	配置项及示例	含义说明

匿名用户	Anonymous_enable=YES	是否允许匿名访问
	Anon_umask=022	设置匿名用户上传文件默认权限掩码
	Anon_root=/var/ftp	设置匿名用户 FTP 根目录（缺省/var/ftp）
	Anon_upload_enable=YES	是否允许匿名用户上传文件
	Anon_mkdir_write_enable=YES	是否允许匿名用户有创建目录的写入权限
	Anon_other_write_enable=YES	是否允许匿名用户有其他写入权限，如对文件改名、覆盖及删除文件
	Anon_max_rate=0	限制匿名用户最大传输速率，0 为不限制，单位为字节
本地用户	Local_enable=YES	是否允许本地系统用户访问
	Local_umask=022	设置本地用户上传文件默认权限掩码
	Local_root=/var/ftp	设置本地用户 FTP 根目录（缺省为用户宿主目录）
	Chroot_local_user=YES	是否将 FTP 本地用户禁锢在宿主目录中
	Local_max_rate=0	限制本地用户最大传输速率，0 为不限制，单位为字节
全局配置	Listen=YES	是否以独立运行的方式监听服务
	Listen_port=21	设置监听 FTP 服务的端口号
	Write_enable=YES	启用任何形式的写入权限（上传、删除文件等）都需开启此项
	Download_enable=YES	是否允许下载文件（建议仅浏览、上传的 FTP 服务器时可以将其设为 NO）
	Dirmessage_enable=YES	用户切换进入目录时，显示.message 文件内容
	Xferlog _enable=YES	启用 xferlog 日志，默认记录到/var/log/xferlog 文件
	Xferlog_std_format=YES	启用标准 xferlog 日志格式
	Connect_from_port_20=YES	允许服务主动服模式
	Pasv_enable=YES	允许被动模式连接
	Pasv_min_port=24500	设置被动模式的服务器最小端口号
	Pasv_max_port=24600	设置被动模式的服务器最大端口号
	Pam_service_name=vsftpd	设置用户认认的 PAM 文件位置(/etc/pam.d/目录中对应的文件名）
	Userlist_enable=YES	是否启用 user_list 用户列表文件
	Userlist_deny=YES	是否禁止 user_list 文件中用户登录
	Max_clients=0	最多允许多少个客户端同时连接（0 为无限制）
	Max_per_ip=0	对来相同 IP 的客户端，最多允许多少个并发连接（0 为无限制）
	Tcp_wrappers=YES	是否启用 TCP_Wrappers 主机访问控制

8.3 vsftpd 服务器的配置案例

实验案例一 vsftpd 服务器配置

一、实验环境

随着公司业务发展要求，需向面向 Internet 搭建一台 FTP 文件服务器，以提供文件上传和下载，选择在 RHEL7 操作系统中构建 vsftpd 服务器来实现。

二、实验步骤

（一）建立可匿名访问的 vsfptde 服务器实验步骤

1、 安装 RHEL7 系统盘中自带 vsftpd 软件包。

2、调整匿名上传目录权限，并准备下载测试文件。

```
[root@zx ~]# chown ftp /var/ftp/pub
[root@zx ~]#
[root@zx ~]# cp /etc/vsftpd/vsftpd.conf /var/ftp/
[root@zx ~]#
[root@zx ~]# ls -l /var/ftp
总用量 16
-rw-r--r--. 1 root root 2602  2月  18 22:10 ftpconfig.tar.bz2
drwxr-xr-x. 2 ftp  root 4096  2月  18 22:25 pub
-rw-------. 1 root root 4491  2月  18 22:41 vsftpd.conf
[root@zx ~]#
```

3、 修改/etc/vsftp/vsftpd.conf 配置文件，开放匿名用户访问，上传许可。

```
[root@zx~]# vi /etc/vsftpd/vsftpd.conf
anonymous_enable=YES                    //允许匿名访问
local_enable=NO                         //不允许本地用户访问
write_enable=YES                        //开放写权限
anon_umask=022                          //设置匿名用户上传文件默认权限掩码
anon_upload_enable=YES                  //是否允许匿名用户上传文件
anon_mkdir_write_enable=YES             //是否允许匿名用户有创建目录的写入权限
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=NO
tcp_wrappers=YES
```

如果希望匿名用户在上传目录中能够进行覆盖、删除重命名文件等写入操作，还可以添加的“anon_other_write_enable=YES”配置项。此项配置可能带来安全性问题，应谨慎使用。

4、重新启动 vsftpd 服务。

5、在另一台 Linux 做为客户端访问 FTP 服务器，（192.168.2.5）测试下载及上传功能。

```
[root@srv1 ~]# ftp 192.168.2.5
Connected to 192.168.2.5.
220 (vsFTPd 2.2.2)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.2.5:root): ftp                    //匿名用户
331 Please specify the password.
Password:                                       //可以不输密码
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,2,5,94,90).
150 Here comes the directory listing.
-rw-r--r--    1 0        0            2602 Feb 18 14:10 ftpconfig.tar
drwxr-xr-x    2 14       0            4096 Feb 18 14:25 pub
-rw-------    1 0        0            4491 Feb 18 14:41 vsftpd.conf
226 Directory send OK.
```

测试上传下载功能

```
ftp> get vsftpd.conf                //测试下载
local: vsftpd.conf remote: vsftpd.conf
227 Entering Passive Mode (192,168,2,5,23,177).
150 Opening BINARY mode data connection for vsftpd.conf (4491
226 Transfer complete.
4491 bytes received in 0.00084 seconds (5.2e+03 Kbytes/s)
ftp> !ls                        //查看下载文件
anaconda-ks.cfg  dsniff.services  install.log.syslog  scsrun.
Desktop          install.log      scsconfig.log       vsftpd.
ftp> put install.log            //测试上传，失败了，什么原因?
local: install.log remote: install.log
227 Entering Passive Mode (192,168,2,5,82,234).
553 Could not create file.
ftp> cd pub                 //因为匿名用户只有在pub目录还有写的权限
250 Directory successfully changed.
ftp> put install.log                //上传成功。
local: install.log remote: install.log
227 Entering Passive Mode (192,168,2,5,139,6).
150 Ok to send data.
226 Transfer complete.
39444 bytes sent in 0.17 seconds (2.3e+02 Kbytes/s)
ftp>
```

如果只是下载文件，也要以使用 wget 工具，指定服务器地址及文件路径即可进行下载。

```
[root@srv1 ~]# wget ftp://192.168.2.5/vsftpd.conf
--22:56:28--  ftp://192.168.2.5/vsftpd.conf
           => `vsftpd.conf.1'
Connecting to 192.168.2.5:21... 已连接。
正在以 anonymous 登录 ... 登录成功!
==> SYST ... 完成。    ==> PWD ... 完成。
==> TYPE I ... 完成。  ==> 不需要 CWD。
==> SIZE vsftpd.conf ... 4491
==> PASV ... 完成。    ==> RETR vsftpd.conf ... 完成。
长度: 4491 (4.4K)

100%[=======================================>] 4,49

22:56:28 (30.8 MB/s) - `vsftpd.conf.1' saved [4491]

[root@srv1 ~]#
```

（二）建立本地用户访问的 vsfptd 服务器实验步骤

1、添加 FTP 测试用户（即本地用户），并准备下载测试文件。

```
[root@zx ~]# useradd yaya
[root@zx ~]# passwd yaya
更改用户 yaya 的密码 。
新的 密码：
无效的密码： 过短
无效的密码： 过于简单
重新输入新的 密码：
passwd： 所有的身份验证令牌已经成功更新。
```

准备下载测试文件：

```
[root@zx ~]# ls -lh /etc/*.conf > /home/yaya/config.list
[root@zx ~]#
```

2、修改 vsftpd.conf 配置文件，开放本地用户。

```
[root@zx~]# vi /etc/vsftpd/vsftpd.conf
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
chroot_local_user=YES
max_clients=20
max_per_ip=2
local_max_rate=102400
pasv_enable=YES
pasv_min_port=24500
pasv_max_port=24600
dirmessage_enable=YES
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
```

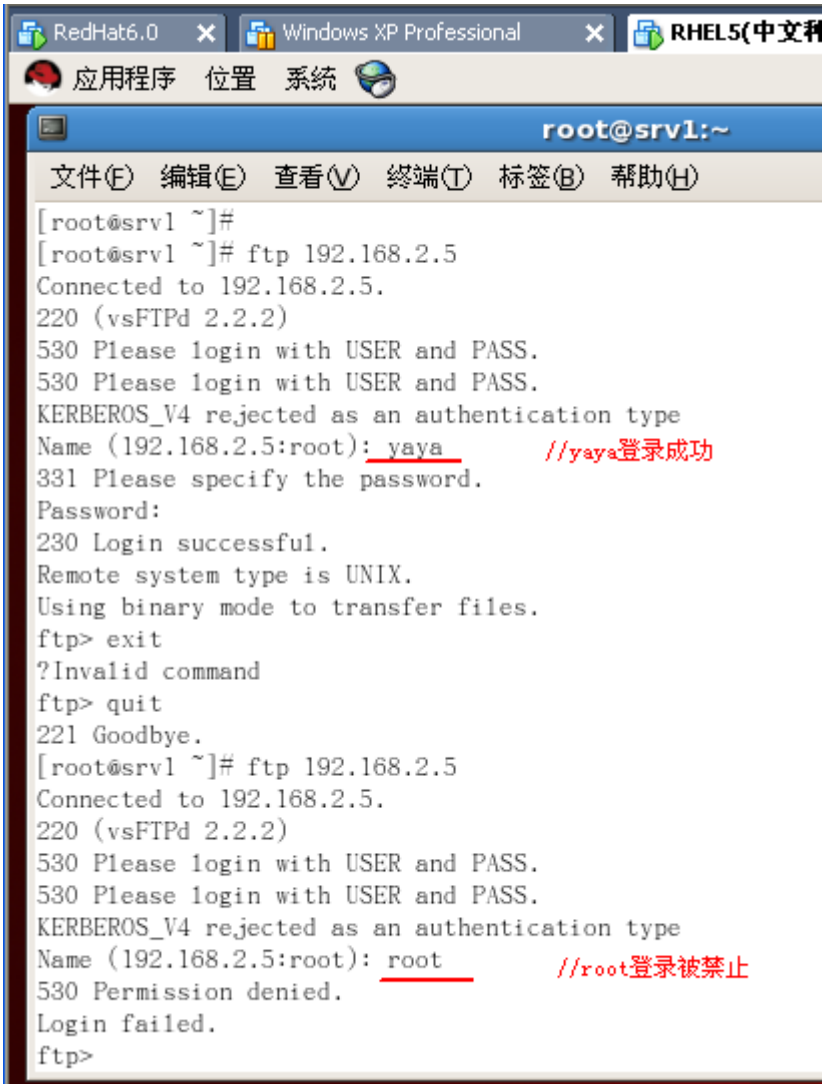
```
listen=YES
#listen_ipv6=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

结合 user_list 文件设置仅允许本地用户 yaya 访问服务器。

```
[root@zx~]# vi /etc/vsftpd/user_list
 yaya
[root@zx~]# vi /etc/vsftpd/vsftpd.conf
userlist_enable=YES
userlist_deny=NO
```

3、重新启动服务

4、在客户端访问 FTP 服务器（192.168.2.5），测试下载及上传功能。



根据步骤(2)中的配置，在客户端必须使用服务器中的本地用户账号进行登录验证，使

用匿名账号将无法登录该 FTP 服务器。配置了用户列表控制，则只有 yaya 用户可以访问并下载、上传文件。用户 yaya 将被禁锢在宿目录中，使用 ftp 命令登录该 FTP 服务器后，执行 pwd 将看到当前目录为“/”，无法切换到其他系统目录。

如果使用 wget 命令测试下载，需要在命令行指定用于登录 FTP 服务的用户账号及密码，进行下载。

第九章　配置 MariaDB 数据库

9.1　安装 MariaDB

1、安装 MariaDB.

```
[root@zx~]# yum -y gruopinstall mariadb mariadb-client
```

2、启动 MariaDB 服务

```
root@zx~]# systemctl  start  mariadb
[root@zx~]# systemctl enable mariadb
[root@zx~]# ss -ntpul |grep mysql
tcp    LISTEN     0      50                        *:3306                     *:*
users:(("mysqld",4738,13))

[root@zx~]# firewall-cmd --permanent --add-service=mysql
[root@zx~]# firewall-cmd --reload
```

3、提高 MariaDB 安装安全性

安装后默认 root 账号没有密码，直接登录。

```
[root@zx ~]# mysql -u root
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]> use test;
```

```
Database changed
MariaDB [test]>
```

用 mysql_secure_installation 提高安全性。

```
[root@zx ~]# mysql_secure_installation
/usr/bin/mysql_secure_installation:行 379: find_mysql_client: 未找到命令
Enter current password for root (enter for none):
OK, successfully used password, moving on...


Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!

Remove anonymous users? [Y/n] y
 ... Success!

Disallow root login remotely? [Y/n] y
 ... Success!

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reload privilege tables now? [Y/n] y
 ... Success!
Thanks for using MariaDB!
```

```
[root@zx ~]# mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

[root@zx ~]# mysql -u root -p
Enter password:
MariaDB [(none)]>
```

9.2 操作 MariaDB

1、建立 rhce 数据库

```
[root@zx ~]# mysql -u root -p
Enter password:
MariaDB [(none)]> create database rhce;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> use rhce;
Database changed
```

2、建立 porducts 表

```
MariaDB [rhce]> create table products (
    -> prod_id int not null,
    -> prod_name char(40) not null,
    -> prod_price decimal(8,2),
    -> primary key (prod_id))
    -> ;
Query OK, 0 rows affected (0.05 sec)

MariaDB [rhce]> show tables;
+---------------+
| Tables_in_rhce |
+---------------+
| products       |
+---------------+
MariaDB [rhce]> describe products;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| prod_id    | int(11)      | NO   | PRI | NULL    |       |
| prod_name  | char(40)     | NO   |     | NULL    |       |
| prod_price | decimal(8,2) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

9.3 结构化查询语言

1、插入数据

```
MariaDB [rhce]> insert into products values(1,'a',100);
Query OK, 1 row affected (0.01 sec)
```

```
MariaDB [rhce]> insert into products values(2,'b',200);
Query OK, 1 row affected (0.00 sec)

MariaDB [rhce]> insert into products values(3,'c',300);
Query OK, 1 row affected (0.00 sec)
```

2、查询数据

```
MariaDB [rhce]> select * from products;
+---------+-----------+------------+
| prod_id | prod_name | prod_price |
+---------+-----------+------------+
|       1 | a         |     100.00 |
|       2 | b         |     200.00 |
|       3 | c         |     300.00 |
+---------+-----------+------------+
3 rows in set (0.00 sec)

MariaDB [rhce]> select * from products where prod_price >=200;
+---------+-----------+------------+
| prod_id | prod_name | prod_price |
+---------+-----------+------------+
|       2 | b         |     200.00 |
|       3 | c         |     300.00 |
+---------+-----------+------------+
2 rows in set (0.01 sec)

MariaDB [rhce]> select * from products where prod_price between 100 and 200;
+---------+-----------+------------+
| prod_id | prod_name | prod_price |
+---------+-----------+------------+
|       1 | a         |     100.00 |
|       2 | b         |     200.00 |
+---------+-----------+------------+
2 rows in set (0.00 sec)
```

3、更新数据

例：把 2 号产品价格修改为 210.

```
MariaDB [rhce]> update products set prod_price=210 where prod_id=2;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4、删除数据

例：把 2 号产品删除.

```
MariaDB [rhce]> delete from products where prod_id=2;
Query OK, 1 row affected (0.00 sec)
```

9.4 管理数据库用户和权限

1、新建数据库用户

例 ：新建一个数据库用户名为 liweijie,口令为 redhat.

```
MariaDB [rhce]> create user liweijie@localhost identified by 'redhat';
Query OK, 0 rows affected (0.01 sec)

MariaDB [rhce]> use mysql;
MariaDB [mysql]> select host,user,password from user;
+-----------+----------+-------------------------------------------+
| host      | user     | password                                  |
+-----------+----------+-------------------------------------------+
| localhost | root     | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| 127.0.0.1 | root     | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| ::1       | root     | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| localhost | liweijie | *84BB5DF4823DA319BBF86C99624479A198E6EEE9 |
+-----------+----------+-------------------------------------------+
4 rows in set (0.00 sec)
```

liweijie@localhost :据库用户只能从 localhost 连接.

liweijie@192.168.1.5 :据库用户只能从 192.168.1.5 连接.

liweijie@192.168.1.% :据库用户只能从 192.168.1.0 网络连接.

liweijie@% :据库用户只能从任何主机连接.

liweijie@l2001:: 1 :据库用户只能从 2001::1 IPv6 地址连接.

2、为新建账号授予权限

新建用的用户可以登录 DB，但没有权限。

```
[root@zx ~]# mysql -u liweijie -p
```

```
Enter password:
MariaDB [(none)]> create database mydb;
ERROR 1044 (42000): Access denied for user 'liweijie'@'localhost' to database 'mydb'
MariaDB [(none)]>
MariaDB [(none)]> use rhce
ERROR 1044 (42000): Access denied for user 'liweijie'@'localhost' to database 'rhce'
```

为新建的用户授权

```
[root@zx ~]# mysql -u root -p
Enter password:

MariaDB [(none)]> grant select,update,delete,insert on rhce.products to liweijie@localhost ;
Query OK, 0 rows affected (0.00 sec)
```

9.5 备份数据库

1、执行逻辑备份数据库

例 ： 备份 rhce 数据库

```
[root@zx ~]# mkdir /backup
[root@zx ~]# mysqldump -u root -p rhce > /backup/rhce.dump
Enter password:
```

例 ： 备份所有数据库

```
[root@zx ~]# mkdir /backup
[root@zx ~]# mysqldump -u root -p --all-databases > /backup/mariadb.dump
Enter password:
[root@zx ~]#
```

2、执行物理备份数据库（以 LVM 为例）

(1) 确定数据库文件存储位置

```
[root@zx ~]# mysqladmin variables -u root -p |grep datadir
Enter password:
| datadir                                       | /var/lib/mysql/
```

(2) 确定数据库文件存储在哪个逻辑卷

```
[root@zx ~]#  df /var/lib/mysql
```

(3) 确定逻辑卷所在卷组有快照的空间

```
[root@zx ~]#  vgdisplay vg0 |grep free
```

(4) 连接到数据库，刷新表，保存到硬盘，并锁定数据库或停止服务

```
[root@zx ~]# mysql -u root -p
Enter password:
MariaDB [(none)]> flush tables with read lock;
```

(5) 不要关闭上面会话，新开一个终端，建立 LVM 快照。维持锁定直到 LVM 快照完成。

```
[root@zx ~]# lvcreate -L 20G -s -n mariadb-backup /dev/vg0/mariadb
```

(6) 回到数据库会话，解锁或启动服务。

```
MariaDB [(none)]> unlock tables;
```

(7) 把快照挂载到一个任意目录,可以复制到其他地方备份。

```
[root@zx ~]# mkdir /mnt/mariadb
[root@zx ~]# mount /dev/vg0/mariadb  /mnt/mariadb
```

(8) 备份后可以删除快照。

```
[root@zx ~]# umount /mnt/mariadb
[root@zx ~]# lvremove /dev/vg0/mariadb-backup
```

9.5 恢复数据库

1、执行逻辑备份的恢复

```
[root@zx ~]#  mysql -u root -p rhce < /backup/mariadb.dump
```

2、执行物理备份的恢复

(1) 停止数据库服务

```
[root@zx ~]# systemctl  stop mariadb
```

(2) 确定数据库文件存储位置

```
[root@zx ~]# mysqladmin variables -u root -p |grep datadir
Enter password:
| datadir                                       | /var/lib/mysql/
```

(3) 删除/var/lib/mysql 目录内容

```
[root@zx ~]#  rm -rf /var/lib/mysql/*
```

(4) 把备份的复制到/var/lib/mysql/目录下。

第十章　Apache

10.1　基本的 Apache httpd 配置

1、安装 Apache.

```
[root@zx~]# yum -y install httpd
```

安装好 httpd 软件包之后，与 Apache 服务器相关的主要目录和文件如下

- /etc/httpd：服务程序的根目录
- /etc/httpd/conf/httpd.conf：服务器的主配置文件
- /var/www/html/：网页文档的默认根目录（网站根目录）
- /etc/init.d/httpd：服务的控制脚本文件
- /usr/sbin/httpd：服务的主要执行程序
- /var/log/httpd/access_log：访问日志文件
- /var/log/httpd/error_log：错误日志文件

2、修改配置文件 /etc/httpd/conf/httpd.conf

```
[root@zx~]# vim /etc/httpd/conf/httpd.conf

      ServerName www.example.com

[root@zx~]# httpd -t
```

3、建立测试网站

```
[root@zx~]# vim /var/www/html/
[root@zx~]# vim index.html
  THis is my Homepage!!
```

4、检查 Selinux 上下文

```
 [root@zx~]#  chcon  -Rt httpd_sys_content_t  /var/www/html/index.html
或
 [root@zx~]#  restorecon  -R /var/www/html/
如果网站文档存放在在其他目录，如放在/opt/www 目录，则需要
```

```
[root@zx~]# semanage fcontext -a -t httpd_sys_content_t  '/opt/www(/.*)?'
```

5、启动服务.

```
[root@zx~]# systemctl start httpd
[root@zx~]# systemctl enable httpd
[root@zx~]# firewall-cmd --permanent --add-service=http --add-service=https
[root@zx~]# firewall-cmd --reload
```

6、测试.

```
[root@zx~]# elinks --dump 127.0.0.1
[root@zx~]# curl  127.0.0.1
```

读懂 httpd.conf 配置文件

（1）httpd.conf 中的全局配置

Httpd.conf 配置文件中包括有相当数量的全局配置项,这些配置项不包括在任何区域中,决定了 Apache 服务器的全局参数。以下是 httpd.conf 文件中比较常用的全局配置项内容。

```
[root@zx ~]# cd etc/httpd/conf/
[root@zx conf]# cp httpd.conf  httpd.conf.bak
[root@zx conf]# grep -v "#" httpd.conf.bak | grep -v "^$" > httpd.conf
```

```
ServerRoot  "/etc/httpd"
Listen 80
User daemon
Group daemon

ServerAdmin webmaster@abc.com

ServerName www.abc.com

DocumentRoot  "/var/www/html"
DirectoryIndex  index.html index.php
Errorlog logs/error_log
LogLevel warn
Customlog logs /access_log common
PidFile logs /httpd.pid
Timeout  300
KeepAlive  On
MaxKeepAliveReauests  100
KeepAliveTimeout 15
Include conf /extra /httpd-vhosts.conf
```

在上述设置行中，各项配置的含义如下。

- ServerRoot：用于设置 httpd 服务器的根目录，该目录中包括了运行站点必需的目录和

文件。httpd.conf 配置文件中，如果设置的目录或文件不使用绝对路径，都认为是在服务器根目录下面。

- Listen：用于设置服务器监听的网络端口号，默认为 80。
- User：用于设置运行进程时的用户身份。
- Group：用于设置运行进程的组身份。
- ServerAdmin：用于设置服务器管理员的 E-Mail 地址，可以通过此地址及时联系服务器管理员
- ServerName :用于设置服务器的完整主机名
- DocumentRoot：用于设置网页文档根目录在系统中的实际路径。配置项比较容易和 ServerRoot 混淆，需要格外注意。
- DircectoryIndex:用于设置网站的默认索引页（首页），可以设置多个文件，以空格分开，默认的首页文件名为 index.html。
- ErrorLog：用于设置错误日志文件的路径和文件名，默认值为
- LogLevel：用于设置记录日志的级别，默认为（警告）
- CustomLog：用于设置服务器中访问日志文件的路径和格式类型。
- PidFile：用于设置保存服务器程序进程号（PID）的文件，默认设置为，目录位于遥服务器根目录中。
- Timeout：用于设置服务器与浏览器之间网络连接的超时秒数，默认设置为 300 秒。
- KeepAlive：用于设置是否使用保持连接功能。设置为 Off 时表示不使用，客户机的每次连接只能从服务器请求返回一个文件，传输的效率比较低；当设置为 On 时，客户机与服务器建立一次连接后可以请求传输多个文件，将提高服务器传输文件的效率。
- MaxKeepAliveRequest：用于设置客户端每次连接允许请求响应的最大文件数，默认设置为 100 个。当 Keeplive 设置为 on 时才生效。
- KeepAliveTimeout：用于设置保持连接的超时数秒，当客户机的两次相邻请求超过该设置值时需要重新进行连接请求，默认设置值为 15 秒。
- Include ：用于包含另一个配置文件的内容，可以将实现一些特殊功能的配置单独放到一个文件里,再使用配置项包含到主配置文件中来,便于独立维护。

以上配置项是文件中最主要的全局配置项，还有很多其他的配置项，在此不一一列举，如果需要使用可以查看服务器中的相关手册文档。

（2）httpd.conf 中的区域设置

除了全局设置项外，文件中的大多数配置都是包括在区域中的。区域设置使用一对组合标记，限定了配置项的作用范围，例如，配置文件中常用的目录区域的形成如下。

```
<Directory />
   Option FollowSymlinks
   AllowOverride None
   Order deny ,allow
   Deny from all
</Directory>
```

如何禁止 Apache 显示目录列表呢？

要禁止 Apache 显示目录结构列表，只需将 Option 中的 Indexes 去掉即可。

```
<Directory "D:/Apa/blabla">
 Options Indexes FollowSymLinks #---------->Options FollowSymLinks
```

```
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

10.2 配置虚拟 web 主机

1、构建基于域名的虚拟主机（最常使用方法）

本案例以实现两个虚拟主机 www.abc.com 和 www.123.com 为例，其对应的服务器 IP 地址为 192.168.1.10，构建过程参考如下。

(1) 确定服务器的主机名、IP 地址等到参数

向域名注册机构申请 Web 站点域名，使得所有的用户在访问 www.abc.com 和 www.123.com 域名时，指向的 IP 地址对应为 192.168.1.10。在实验过程中也可以自行搭建 DNS 服务器.。也可以修改 /etc/hosts 文件添加两条 IP 到名称的记录。

```
[root@zx ~]# cat /etc/hosts
127.0.0.1   localhost localhost
::1         localhost localhost
192.168.1.10 www.abc.com www
192.168.1.10 www.123.com www
[root@zx ~]#
```

将服务器的 IP 地址设置为 192.168.1.10，主机名设置为虚拟站点域名中的一个。

(2) 分别准备两个虚拟站点的网页文件。

在服务器的网页根目录下建立两个文件夹，并分别建立两个测试网页。

```
[root@zx ~] cd /var/www/html/
[root@zx htdocs] mkdir abc 123
[root@zx htdocs] echo “www.abc.com” > abc/index.html
[root@zx htdocs] echo “www.123.com” > 123/index.html
```

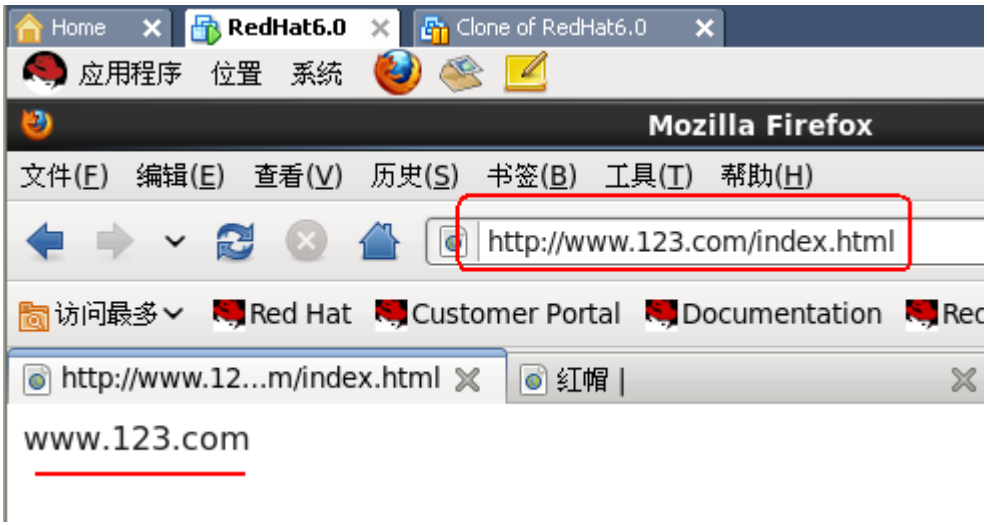
(3) vim /etc/httpd/conf.d/www.conf，添加虚拟主机配置

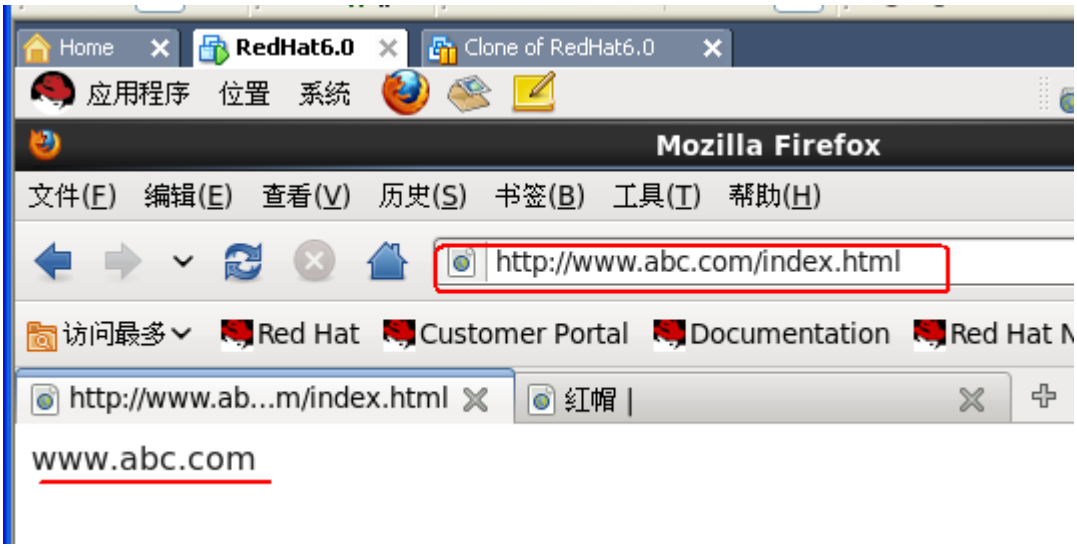
```
[root@zx ~] vim /etc/httpd/conf.d/www.conf
```

```
……
NameVirtualHost 192.168.1.10
<VirtualHost 192.168.1.10>
    DocumentRoot /var/www/html/abc
    ServerName www.abc.com
    ServerAdmin  root@example.com
    ErrorLog "logs/abc_error_log"
    CustomLog "logs/abc_access_log" combined

</VirtualHost>
<VirtualHost 192.168.1.10>
    DocumentRoot /var/www/html/123
    ServerName www.123.com
</VirtualHost>
```

(4) 重新启动 httpd 服务

(5)在客户机浏览器中访问虚拟站点





2、虚拟主机排错

1）名称解析

首先要确保客户机能够正确解析这两个虚拟主机的域名，并能够连接到该服务器。如果在实验中没有搭建可用的 DNS 服务器，也可以通过修改客户机的 hosts 文件来完成域名解析,如果是 Linux 客户机，则修改/etc/hosts 文件，如果是 Windows 客户机，则修改 c:/windows/system32/drivers/etc/hosts 文件，添加如下域名 IP——IP 地址映射记录即可。

2) 目录在 httpd.conf 中权限设置

```
<Directory "/opt/www">
    AllowOverride None
    Require all granted
</Directory>
```

2）httpd 服务不能启动，查日志

```
Journalctl   -xn
Journalctl   UNIT=httpd.service
```

例：如果把 www.123.com 网站目录建在 /home/html/ 目录下，发现只有 www.abc.com 可访问，www.123.com 不能访问了，为什么？

1）查日志，访问被拒绝了，原因是 SELinux

```
[root@instructor httpd]# tail -3  /var/log/httpd/error_log
[Thu May 01 01:34:48 2014] [notice] Apache/2.2.15 (Unix) DAV/2 configured -- resuming normal o
perations
[Thu May 01 01:35:12 2014] [error] [client 192.168.0.5] (13)Permission denied: access to /inde
x.html denied
```

2) 设置 SELinux 上下文

```
[root@instructor ~]# ll -Z  /home/html/
-rw-r--r--. root root unconfined_u:object_r:home_root_t:s0 index.html
[root@instructor ~]#
[root@instructor ~]# chcon -Rt httpd_sys_content_t /home/html
[root@instructor ~]# ll -Z  /home/html/
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 index.html
[root@instructor ~]#
```

10.3 基于 SSL 加密的 WEB 服务

一、需求

1. 安装的 httpd 服务器，添加 HTTPS 协议支持以提高安全性。
2. 当客户机通过 HTTP 方式访问站点时，能够自动跳转为 HTTPS 方式访问。

二、https 是什么？

HTTPS 指的是 Hyper Text Transfer Protocol Secure，安全超文本传输协议。HTTPS 实际上使用了 SSL（安全套接字层）作为 HTTP 应用层的子层，针对明文传输的 HTTP 通信流进行加密，从而避免敏感信息被捕获或窃听，因此 HTTPS 协议在网上银行、安全邮箱等 Web 访问场合比较常见。

三、在上面配置虚拟主机 www.abc.com 中启用 SSL 配置步骤

1. 确认系统中已安装有 openssl 软件包，用来为服务器生成证书

```
[root@localhost ~]# rpm -qa | grep openssl
openssl-1.0.1e-34.el7.x86_64
```

2. 在 httpd 软件包时添加了 ssl 支持选项

```
[root@localhost]# yum -y install mod_ssl
```

3. 生成 KEY 密钥文件和签发 CRT 证书

```
[root@localhost ~]# /etc/pki/tls/certs/
[root@localhost conf]# openssl genrsa -out server.key 1024          //生成服务器密钥文件
Generating RSA private key, 1024 bit long modulus
.....................................................................................++++++
.........++++++
e is 65537 (0x10001)
[root@localhost conf]# chmod 600 server.key

[root@localhost conf]# openssl req -new -key server.key -out server.csr
                                                                    //生成服务器证书文件
-----
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:Guangdong
Locality Name (eg, city) [Newbury]:Guangzhou
Organization Name (eg, company) [My Company Ltd]:easthome
Organizational Unit Name (eg, section) []:Rhce
Common Name (eg, your name or your server's hostname) []:www.abc.com
Email Address []:root@abc.com
```

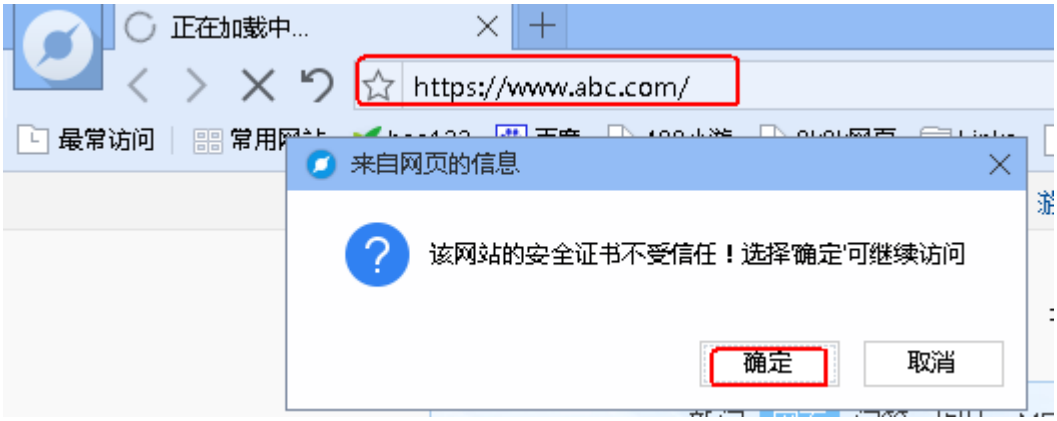
```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

[root@localhost conf]# ls -l server.key server.csr
-rw-r--r-- 1 root root 700 12-06 19:52 server.csr
-rw------- 1 root root 887 12-06 19:46 server.key
[root@localhost ~]# openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt
                                                    //签署服务器证书
Signature ok
Getting Private key
```

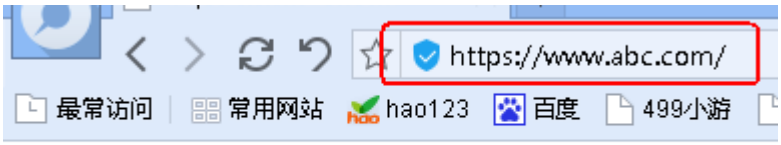
4. 修改虚拟主机配置文件

```
[root@localhost # vim /etc/httpd/conf.d/www.conf
<VirtualHost 10.11.21.239:443>
    DocumentRoot /var/www/html/abc
    ServerName www.abc.com
    SSLEngine on
    SSLProtocol all -SSLv2
    SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
    SSLCertificateFile          /etc/pki/tls/certs/server.crt
    SSLCertificateKeyFile       /etc/pki/tls/certs/server.key
  ServerAdmin   root@example.com
  ErrorLog "logs/abc_error_log"
  CustomLog "logs/abc_access_log" combined
</VirtualHost>
```

5. 在客户机浏览器中访问测试当访问 https://IP 或域名

```
[root@localhost]#   curl -k   https://www.abc.com
```





www.abc.com

6、如不能访问检查配置文件、防火墙是否允许了 https 服务和 SELinux 的相关设置。

第十一章　BASH 脚本（一）

常见的 Shell 变量的类型包括环境变量、预定义变量、位置变量、用户自定义变量。本节将分别学习这四种 Shell 变量的使用。

11. 1、 Shell 的环境变量

通过 set 命令可以查看系统中所有 Shell 变量（包括环境变量和用户自定义变量），由于内容输出较多，建议使用 less 命令分页查看。

常用环境变量：

PATH	决定了 shell 将到哪些目录中寻找命令或程序
HOME	当前用户主目录
HISTSIZE	历史记录数
LOGNAME	当前用户的登录名
USER	当前用户名

UID	当前用名的 UID
HOSTNAME	指主机的名称
SHELL	前用户 Shell 类型
LANGUGE	语言相关的环境变量，多语言可以修改此环境变量
MAIL	当前用户的邮件存放目录
PS1	基本提示符，对于 root 用户是#，对于普通用户是$
PS2	附属提示符，默认是“>”

例：以分号分隔，显示当前的用户的用户名、宿主目录、登录 Shell。

```
[root@zx ~]# echo "$USER;$HOME;$SHELL"
teacher;/root;/bin/bash
[root@zx ~]#
```

例：查看当前命令的搜索路径，并将/opt/bin 目录添加到现有搜索路径中去，从而可以直接执行此目录中的命令。

```
[root@zx ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bi
:/root/bin:/root/bin
[root@zx ~]# PATH="/opt/bin:$PATH"
[root@zx ~]# PATH
bash: PATH: command not found
[root@zx ~]# echo $PATH
/opt/bin:/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/us
/bin:/bin:/root/bin:/root/bin
```

环境变量的配置文件

用户可在当前的 Shell 环境中直接为环境变量赋值，但需要长期变更所使用的某个环境变量时，可以修改配置文件。

在 Linux 系统中，用户环境变量的设置工作习惯上在 /etc/profile 文件及宿主目录中 .bash_profile 文件中进行，前者称为全局配置文件（对所有用户起作用），后者称为用户配置文件（允许覆盖全局配置）。

例：在当前用户环境中，将用于限制历史命令的环境变量 HISTSIZE 的值改为 100。

```
[root@zx ~]# echo $HISTSIZE
1000
[root@zx ~]# export HISTSIZE=100
[root@zx ~]# echo $HISTSIZE
100
[root@zx ~]#
```

例：编辑“~/.bash_profile”文件，修改 PATH 的设置，以便用户在下次登录后能够使用服务/opt/bin 目录作为默认的搜索路径。

```
#  vi  /root/.bash_profiel
```

PATH=$PATH:$HOME/bin:/opt/bin

Export PATH

11.2 Shell 位置变量

为了在使用 Linux 程序时，方便通过命令行给程序提供操作参数，Bash 引入了位置变量的概念。在执行 Shell 命令操用时，除了输入的第一字段（命令名或脚本程序名）以外，其余的字符串参数按照从在到右的顺序依次赋组位置变量。

需要引用这些位置变量的值时，采用$n 的格式，其中 n 是参数位置的序号（从 1-9）。例如当执行 servive network restart 命令时，service 脚本程序的第 1 个位置参数“$1”表示，对应的值为 network，第 2 个位置参数用“$2”表示，对应的值为 restart。

$0 代表所执行命令或脚本程序的名称，虽然$0 与位置变量的格式相同，但是属于预定义变量而不是位置变量。在编写脚本程序时，经常会使用位置变量来接收用户所指定的命令参数。

11.3 预定义变量

预定义变量是由 Bash 程序预先定义好的一些特殊变量，用户只能使用预定义变量，而不能创建新的预定义变量，或者直接为预定义变量赋值。所有的预定义变量都是由$符号和别一个符号组成的，较常用的预定义变量包括以下这些。

- $#：表示命令行中位置参数的数量
- $*：表示所有位置参数的内容
- $?：表示命令执行后返回的状态，用于检查上一个命令的执行是否正确。在 Linix 中，命令退出状态为 0 表示命令执行正确，任何非 0 值的表示命令执行错误。
- $$：表示当前进程的进程号。
- $!：表示后台运行是最后一个进程的进程号。
- $0：表示当前执行进程的进程名。

预定义变量通常使用在脚本程序中，在命令行界面中的应用并不多见（尽管也可以使用）

11.4、 用户自定义变量

1、定义新的变量

变量名=变量值

例：新建立一个名为“DAY”的变量，初始内容设置为“Sunday”。

DAY=Sunday

2、查看和引用变量的值

通过在变量名称前添加前导符号$，可以引用一个变量的内容。若需要在终端中输出变量的内容，可以使用 echo 的命令，echo 命令同时也用于显示用户指定的其他字符串内容。

例、查看变量 DAY 的内容，比较使用符号$与不使用符号$的区别。

```
[root@zx ~]# DAY=Sunday                    //赋值
[root@zx ~]# echo DAY                    //错误的引用
DAY
[root@zx ~]# echo $DAY                  //正确引用DAY变量的内容
Sunday
[root@zx ~]#
```

当变量名称容易和紧跟其后的其他字符相混淆时，需要添加大括号“{}”将其包围起来，否则将无法确定正确的变量名称。当未指定变量或该变量未设置时，命令将显示空行。

例、在变量 DAY 的内容后紧跟“zhongxin”字符串并一起显示。

```
[root@zx ~]# echo $DAYzhongxin      //错误引用

[root@zx ~]# echo ${DAY}zhongxin//正确引用变量DAY内容
Sundayzhongxin
[root@zx ~]#
```

3、为变量赋值的常用方法。

在符号“=”后边直接指定变量内容是为变量赋值的最基本的方法，除此之外，管理员通常还会使用到其他的一些赋值勤操作，从而使变量内容的获取更加灵活多变，以便适应于各种复杂的系统管理任务。常用的几种变量赋值操作包括双引号、单引号、后撇号、 read 命令。

1）双引号（“）

使用双引号时，允许在双引号的范围内使用$符号来引用其他变量的值（变量引用）。在简单的赋值操作中，双引号有时可以省略。

例、确认变量 DAY 的内容，并使用双引号为 TODAY 变量赋值。

```
[root@zx ~]# echo $DAY
Sunday
[root@zx ~]# TODAY="Today is $DAY"
[root@zx ~]# echo $TODAY
Today is Sunday
[root@zx ~]#
```

2）单引号（‘）

使用单引号时，将不允许在单引号的范围内引用其他变量的值，$符号或者其他任何符号将作为普通字符看待。

例、确认变量 DAY 的内容，并使用单引号为 TODAY 变量赋值。

```
[root@zx ~]# echo $DAY
Sunday
[root@zx ~]# TODAY='Today is $DAY'
[root@zx ~]# echo $TODAY
Today is $DAY
[root@zx ~]#
```

3）反撇号（`）

使用反撇号时，允许将执行特定命令的输出结果赋给变量（命令替换），反撇号内包含的字串必须是能够执行的命令，执行后会输出结果替换该命令字串。新的可替代的语法是$( )。

例、统计当前登录到本地终端中的用户数量，并将结果保存到变量中。

```
[root@zx ~]# UserNum=`w | grep "tty" | wc -l`
[root@zx ~]# echo $UserNum
1
[root@zx ~]#
```

```
# UserNum=$(w | grep” tty”  | wc -l)
```

例、用一行命令找出安装了 fdisk 程序的软件包名称（需要先确定 fdisk 程序的文件位置）。

```
[root@zx ~]# rpm -qf `which fdisk`
util-linux-ng-2.17.2-6.el6.i686
[root@zx ~]#
```

例、通过 find 命令找出系统中用户 hackli 留下的文件或目录，并使用 rm 命令将其删除。

```
[root@zx ~]# rm -rf `find / -user hackli`
```

在需要嵌套使用命令替换操作时，反撇号将力所不能及，这时可以将反撇号用$()来代替。

例、还是使用一行命令，将上一个例子的输出结果保存到变量 FdiskPKG 中。

```
[root@zx ~]# FdiskPKG=$(rpm -qf $(which fdisk))
[root@zx ~]# echo $FdiskPKG
util-linux-ng-2.17.2-6.el6.i686
[root@zx ~]#
```

4）read 命令

除了上述赋值操作外，还可以使用的 Bash 内置命令 read 来给变量赋值。read 命令可以从终端（键盘）读取输入，实现简单的交互过程。将从标准输入读入一行内容，并以空格为分隔符，将读入的各字段分别赋值给指定列表中的变量（多余的内容赋值给最后一个变量）。若指定的变量只有一个，则将整行内容赋值给该变量。

例、从键盘输入一整行数据，赋值给变量 HELO，并确认 HELO 变量的内容。

```
[root@zx ~]# read HELO
Good Morning,teacher!
[root@zx ~]# echo $HELO
Good Morning,teacher!
[root@zx ~]#
```

例、从键盘输入一整行数据，依次赋值给变量 G1、G2，并确认 G1、G2 变量的内容。

```
[root@zx ~]# read G1 G2
Good Morning,teacher
[root@zx ~]# echo $G1
Good
[root@zx ~]# echo $G2
Morning,teacher
[root@zx ~]#
```

为了使交互式操作的界面更加友好，提高易用性，命令可以结合“-p”选项来设置提示信息，用于告知用户应该输入的内容等相关事项。

例、从键盘读出一个数值变量，并给出相应的提示信息。

```
[root@zx ~]# read -p "Please input a number:" YourNum
Please input a number:254
[root@zx ~]# echo $YourNum
254
[root@zx ~]#
```

4、设置变量的作用范围

对于用户自行定义的变量，默认情况下只能在当前的 Shell 环境中使用，因此称为局部变量。局部变量在新开启的子 Shell 环境中是无效的（无法引用定义的变量）。

例、在当前 Shell 环境中定义一个变量 FILESVR，开启一个新的 Shell 子进程，处于子 Shell 环境中时将无法使用变量 FILESVR 的内容。

```
[root@zx ~]# FILESVR=filesvr.zx.com
[root@zx ~]# tcsh
[root@zx ~]# echo $FILESVR
FILESVR: Undefined variable.
[root@zx ~]# exit
exit
[root@zx ~]# echo $FILESVR
filesvr.zx.com
[root@zx ~]#
```

为了使用户定义的变量在所有的子 Shell 环境中能够继续使用，减少重复设置工作，可以使用 export 命令将指定的变量设置为“全局变量”。export 命令可以同时使用多个变量名作为参数（不需要使用$符号），变量之间以空格分隔。

例、确认变量 FILESVR 的内容，并将其设置为全局变量，在子 Shell、当前 Shell 环境中进行验证。

```
[root@zx ~]# echo $FILESVR
filesvr.zx.com
[root@zx ~]# export FILESVR
[root@zx ~]# tcsh
[root@zx ~]# echo $FILESVR   //在子Shell中也可以使用该变量
filesvr.zx.com
[root@zx ~]# exti
exti: Command not found.
[root@zx ~]# exit
exit
[root@zx ~]# echo $FILESVR //返回原Shell环境中仍然能使用该变量
filesvr.zx.com
[root@zx ~]#
```

export 命令还可以在输出变量的同时对指定名称的变量进行赋值（创建），这样在使用 export 命令之前就不需要单独为变量进行赋值了。

例、定义两个变量 MONTH YEAR，并将其设置为全局变量。

```
[root@zx ~]# export MONTH=May YEAR=2012
[root@zx ~]# echo $MONTH $YEAR
May 2012
[root@zx ~]#
```

5、清除自定义变量

当用户不再需要使用自定义变量时，可以使用 unset 命令对已定义的用户变量进行清除，指定一个或多个变量名称作为参数即可（以空格分隔）

例、清除已设置的变量 DAY、MONTH、YEAR。

```
[root@zx ~]# unset DAY MONTH YEAR
[root@zx ~]# echo $DAY $MONTH $YEAR          //已清除

[root@zx ~]#
```

6、数值变量的运算

Bash 程序并不适合进行强大的数字运算（如小数、指数等），一般只适合进行简单的整数运算。对 Shell 变量进行数值运算时，更多的时候是用于脚本程序的过程控制（循环次数等，下一章将会讲解）。对整数数值变量进行算术运算时，可以使用 expr 表达式命令，格式如下。

Expr　变量 1　运算符　变量 2　[运算符 变量 3] …..

或

$[变量 1　运算符　变量 2]

其中，变量 1、变量 2……对应为需要计算的数值变量（需要用$符号引用），常用的几种运算符如下。

- “+”：加法运算

- “-”：减法运算
- “*”：乘法运算，注意不能仅使用“*”符号，否则将被当成文件通配符
- “/”：除法运算
- “%”：求模运算，又称为取模运算，即计算数值相除后的余数。

例、设置变量 X、Y 的值分别为 34、12，依次计算变量 X、Y 的加减乘除及取模运算结果。

```
[root@zx ~]# X=34
[root@zx ~]# Y=12
[root@zx ~]# expr $X + $Y
46
[root@zx ~]# expr $X - $Y
22
[root@zx ~]# expr $X \* $Y
408
[root@zx ~]# expr $X / $Y
2
[root@zx ~]# expr $X % $Y
10
[root@zx ~]#
```

echo $[x+y]

例、计算变量 X 的值与数值“123”的和，并将计算机结果重新赋值给变量

```
[root@zx ~]# echo $X
34
[root@zx ~]# X=`expr $X + 123`
[root@zx ~]# echo $X
157
[root@zx ~]#
```

第十二章　BASH 脚本（二）

12.1　Shell 脚本的概念

Bash 程序不仅可以作为用户 Linux 管理系统命令的命令操作环境，同时也可以作为一种优秀的脚本程序语言。凡是使用 Shell 编程语言编写的程序文件都可以称为脚本，通俗一点的说，只要将一些命令行按顺序保存到一个文本文件中，并给予这个文件可执行权限，那么这个文件就可以称为 Shell 脚本。当然，Shell 脚本是为了完成一定的管理任务才创建的，因此 Shell 脚本文件中各条命令并不是杂乱无章随便放置的，这就需要用户来进行组织和设计了。

写编译型的高级编程语言（如 C/C++、Java 等）不同，Shell 脚本程序是属于解释执行的，并不需要进行特别编译，而只需要有相应的命令解释器即可。长期以来，Shell 脚本在

Linux 系统中得到了广泛的使用，并承担着重要的角色。

12.2 编写 Shell 脚本文件

1、建立包含可执行语句的文本文件

例、使用编辑器 vi 编写一个简单的脚本文件 repboot.sh，用于报告当前系统中目录所占有的空间大小，并列出其中内核文件的属性信息。

```
#!/bin/bash
echo "Useage of /boot:"
du -sh /boot
echo "The mode of kernel file:"
ls -lh /boot/vmlinuz-*
~
~
```

在上述脚本文件中，依次设置了四条可执行命令，两条 echo 语句用于显示相应的提示信息，而 du、ls 命令分别用于完成查看目录空间、显示文件属性任务。

2、为脚本文件添加可执行权限

编写并保存好脚本文件后，需要执行该程序才能看到操作结果。但是刚建立的脚本文件通常不具有可执行属性，因此还得使用命令为文件添加“X”权限。

例、为上一步编写的脚本文件添加可执行权限。

```
[root@zx ~]# ls -l repboot.sh
-rw-r--r--. 1 root root 104  1月  31 23:44 repboot.sh
[root@zx ~]# chmod a+x repboot.sh
[root@zx ~]# ls -l repboot.sh
-rwxr-xr-x. 1 root root 104  1月  31 23:44 repboot.sh
[root@zx ~]#
```

将脚本文件设置了可执行属性后就可以通过执行脚本文件来验证程序的正确性了。

12.3 执行脚本

在命令行环境中可以有多种方式执行脚本，下面分别进行介绍。

1、直接执行带“ X”权限的脚本文件

为脚本文件设置了可执行属性后，在命令千可以直接通过脚本文件的路径执行脚本程序，这也是最常用的一种方式。

例、执行当前目录下的 repboot.sh 脚本程序文件。

```
[root@zx ~]# ./repboot.sh
Useage of /boot:
31M     /boot
The mode of kernel file:
-rwxr-xr-x. 1 root root 3.5M  9月   1 2010 /boot/vmlinuz-2.6.32-71.el6.i686
[root@zx ~]#
```

由于脚本文件存放在当前目录，而不是存放在用户的命令搜索路径，因此执行脚本文件时，需要在文件之前加入路径，明确指出需要执行当前目录下的脚本文件，这种方法也是出

于对系统安全性的考虑。

2、使用解释器程序执行脚本

这种方式可以将脚本文件作为指 Shell 定解释器程序（如 bash 等）的参数，由解释器程序负责读取脚本文件中的内容并执行，因此并不需要脚本文件具有可执行的属性。此方法通常只在脚本的调试阶段使用。

例、使用解释器程序手动加载执行脚本文件中的语句。

```
[root@zx ~]# bash repboot.sh
Useage of /boot:
31M     /boot
The mode of kernel file:
-rwxr-xr-x. 1 root root 3.5M  9月   1 2010 /boot/vmlinuz-2.6.32-71.el6.i686
[root@zx ~]#
```

12.4 Shell 脚本应用实例

本小节主要通过两个编写 Shell 管理脚本的具体实例，来展示普通 Shell 脚本的编写应用。

1、Shell 脚本案例 1

由于公司的文件服务器空间有限，需要完成一项定期任务，即在每周五下班前（17:30）检查公共共享目录 /var/ftp/pub 中的内容，并将其中所有子目录及文件的详细列表和当时的时间信息追加保存到 /var/log/pubdir.log 日志文件中，然后清空该目录中的内容。

根据上述需求描述，推荐的操作步骤如下。

（1）编写脚本文件，并添加执行权限。

```
[root@zx ~]# cat /opt/ftpclean.sh
#!/bin/bash
date >> /var/log/pubdir.log
ls -lhR /var/ftp/pub >> /var/log/pubdir.log
rm -rf /var/ftp/pub/*
[root@zx ~]#
[root@zx ~]# chmod a+x /opt/ftpclean.sh
[root@zx ~]#
```

（2）设置计划任务并确认服务已经启动。

```
[root@zx ~]# service crond status
crond (pid  1989) 正在运行...
[root@zx ~]# chkconfig --level 35 crond on
[root@zx ~]# crontab -l
30      17      *       *       5       /opt/ftpclean.sh
```

2、脚本案例 2

公司内网开发服务器中的数据库目录位于 /var/lib/mysql ，根据数据安全管理要求，至少每隔三天要做一次完整备份，备份前需要统计该目录占用的总空间大小，并将备分日期、目录空间大小等信息保存到临时文件 /tmp/dbinfo.txt 中，然后使用 tar 命令将 dbinfo.txt 文件随数据库目录一起备份到 /opt/dbbak 目录中，备份包文件名中要求体现当天的日期。

根据上述需求描述，推荐的操步骤如下。

（1）创建保存备份文件的目录

```
[root@zx ~]# mkdir /opt/dbbak
[root@zx ~]#
```

（2）编写脚本文件，并添加执行权限

```
#!/bin/bash
DAY=`date +%Y%m%d`
SIZE=`du -sh /var/lib/mysql`
echo "Date:$DAY" >> /tmp/dbinfo.txt
echo "Date Size:$SIZE" >> /tmp/dbinfo.txt
tar zcvf /opt/dbbak/mysqlbak-${DAY}.tar.gz /var/lib/mysql /tmp/dbinfo.txt
rm -f /tmp/dbinfo.txt
~
```

```
[root@zx ~]# chmod a+x /opt/dbbak.sh
[root@zx ~]#
```

（3）设置计划任务并确认服务已经启动

```
[root@zx ~]# crontab -l
55      23      */3     *       *       /opt/dbbak.sh

[root@zx ~]#
```

12.5 Shell 脚本结构控制语句

Shell 作为一种解释型脚本编程语言，同样包含顺序、分支、循环这些基本的程序控制结构。通过使用分支、循环等控制语句，可以编写出应用更加灵活、功能更强大的 Shell 管理脚本。

12.5.1 使用 if 条件语句

在 Shell 脚本中使用 if 语句的好处是：可以根据特定的条件来决定是否执行某项操作，当满足不同条件执行不同的操作。本小节分别从条作测试操作、if 语句结构、应用实例三个方面来讲解 if 语句在 Shell 脚本的应用。

1、条件测试操作

需要在 Shell 脚本中有选择的性的执行任务时，首先面临的问题就是，如何设置命令执行的条件？

在 Shell 环境中，可以根据命令执行后的返回状态值来判断该命令是否成功执行，当返回值为 0 时表示成功执行，否则（非 0 值）表示执行失败。用于特定条件表达示的测试时，可以使用 Liunx 系统中提供的专用工具——test 命令。

使用 test 测试命令时，可以有以下两种形式。

Test　条件表达式

或者

[条件表达式]

这两种方式的作用完全相同，但通常后一种方式更为常用，也更贴近编程习惯。需要注意的是，方括号“[”或者“]与条件表达示语句之间至少需要一个空格进行分隔。

根据需要判断的条件内容不同，条件操作也不同，最常用的条件操作主要包括文件状态

测试、比较整数值大小、比较字符串，以及同时判断多个条件时的逻辑关系，下面将分别进行讲解。在后续内容中，主要采用方括号的测试形式，不再对两种形式进行对比。

1）测试文件状态

文件状态测试是指根据给定的路径名称，判断该名称对就的是文件还是目录，或者判断文件是否可读、可写、可执行等。根据所判断的状态不同，在条件表达式中需要使用不同的操作选项。

- -d：测试是否为目录
- -e：测试目录或文件是否存在
- -f：测试是否为文件
- -r：测试当前用户是否有权限读取
- -w：测试当前用户是否有权限写入
- -x：测试当前用户是否可执行该文件
- -L：测试是否为符号连接文件

执行条件测试操作以后，通过预定义变量$?可以获得测试命令的返回状态值，从而能够判断该条件是否成立（返回 0 值表示条件成立，非 0 值表示条件不成立）但通过这种方式查看测试结果会比较繁琐。

例、先后测试/etc、/etc/hosts 是否是目录，并通过$?变量查看返回状态值，据此判断测试结果。

```
[root@zx ~]# [ -d /etc ]
[root@zx ~]# echo $?
0                          返回值为0，表示上一次测试的条件成立。
[root@zx ~]# [ -d /etc/hosts ]
[root@zx ~]# echo $?
1                          返回值非0，表示上一次测试的条件不成立。
[root@zx ~]#
```

虽然通过查看$?变量的值也可以判断条件测试结果，但是操作比较繁琐，输出结果也不是很直观。为了更便于查找看条件测试操作的结果，可以结合命令分隔符号&&和命令 echo 一起使用，当条件成立时直接输出“YES“其中符号&&表示”而且“的关系，只有当前边的命令执行成功后，才会执行后面的命令，否测后边的命令将会被忽略。

例、测试 /media/cdrom 及其父目录是否存在，如果存在就显示“YES”，否则不输出任何信息。

```
[root@zx ~]# [ -e /media/cdrom ] && echo "YES"
[root@zx ~]#                                  //无输出表示目录不存在
[root@zx ~]# [ -e /media ] && echo "YES"
YES                                           //显示YES表示该目录存在
[root@zx ~]#
```

例：使用 teacher 用户登录，并测试是否对 /etc/passwd 文件有读、写权限，如果有则显示“YES”。

```
[teacher@zx root]$ [ -r /etc/passwd ] && echo  "YES"          //有读的权限
YES
[teacher@zx root]$ [ -w /etc/passwd ] && echo  "YES"
[teacher@zx root]$                                            //无写的权限
[teacher@zx root]$
```

需要注意的是测试读取、写入权限时，尽量不要以 root 用户的身份进行操作，否则可以会看不到预期的效果（因为 root 是超级管理员，许多文件权限都不受限制）

2) 整数值比较

整数值比较是指根据给定的两个整数值，判断第一个数是否大于、等于、小于……第 2 个数，可以使用的操作选项如下：

- -eq：第一个数等于第二个数
- -ne：第一个数不等于第二个数
- -gt：第一个数大于第二个数
- -lt：第一个数小于第二个数
- -le：第一个数小于或等到于第二个数
- -ge：第一个数大于或等于第二个数

整数值比较的测试操作在 Shell 脚本编写中的应用较多，例如，用于判断磁盘使用率，登录用户数量是否超标以及用于控制脚本语句的循环次数等。

例、测试当前登录到系统中的用户数量是否小于或等于 10，是则输出“YES”。

```
[root@zx ~]# who | wc -l
3
[root@zx ~]# [ `who | wc -l` -le 10 ] && echo "YES"
YES
[root@zx ~]#
```

例、提取出/boot 分区的磁盘使用率，并判断是否超过 95%（为了便于理解，操作步骤已适当进行分解）。

```
[root@zx ~]# df -hT | grep "/boot"
/dev/sda1      ext4    194M   36M  148M  20% /boot
[root@zx ~]#
[root@zx ~]# BootUsage=`df -hT | grep "/boot" | awk '{print $6}' | cut -d "%" -f 1`
[root@zx ~]#
[root@zx ~]# echo $BootUsage
20
[root@zx ~]# [ $BootUsage -gt 95 ] && echo "YES"                    //无输出表示未超标。
[root@zx ~]#
```

3) 字符串比较

字符串比较可以用于检查用户输入，例如，在提供交互式操作时，判断用户输入的选择项是否与指定的变量内容相匹配。“=”、“！=”操作选项分别表示匹配、不匹配，“-z”操作选项用于检查字符串是否为空。其中“!”符号用于取反，表示相反的意思。

例：提示用户输入一个文件路径，并判断是否是 /etc/inittab ,如果是则显示“YES”。

```
[root@zx ~]# read -p "location:" FilePath
location:/etc/inittab                                    //从键盘输入
[root@zx ~]# [ $FilePath="/etc/inittab" ] && echo "YES"
YES
[root@zx ~]#
```

例、若当前环境变量 LANG 的内容不是“en.US”，则输出 LANG 变量的值，否则无输出。

```
[root@zx ~]# [ $LNAG!="en.US" ] && echo $LANG
zh_CN.UTF-8
[root@zx ~]#
```

例、使用 touch 命令建立一个新文件 test.txt，测试其内容是否为空，向文件中写入内容后，

再次进行测试。

```
[root@zx ~]# touch test.txt
[root@zx ~]# [ -z `cat test.txt` ] && echo "YES"
YES                                          //表示此文件为空文件。
[root@zx ~]# echo "wellcome" > test.txt
[root@zx ~]# [ -z `cat test.txt` ] && echo "YES"
[root@zx ~]#                                 //无输出表示此文件非空。
```

4) 逻辑测试

逻辑测试是指同时使用的两个（或多个）条件表达式之间的关系。用户可以同时测试多个条件，根据这些条件是否同时成立或者只要有其中一个条件成立等情况，来决定采取何种操作。逻辑测试可以使用的操作选项如下。

- &&：逻辑与，表示前后两个表达式都成立时整个测试结果才为真，否则结果为假。在使用 test 命令形式进行测试时，此选项可改为“-a”。
- ||：逻辑或，表示前后两个条件至少有一个成立时整个测试结果才为真，否测结果为假。在使用命令形式进行测试时，此选项可改为“-o”
- !：逻辑否，表示当前指定的条件表达式不成立时，整个测试命令的结果为真。

在上述逻辑测试的操作选项中，“&&”和“||”和通常也用于间隔不同的命令操作，其作用是相似的（前面已经接触过&&符号的应用）。同时使用多个逻辑运算操作时，一般按时从左到右的顺序进行测试。

例、测试当前的用户是否是 teacher，若不是则提示 “Not teacher”。

```
[root@zx ~]# echo $USER
root
[root@zx ~]# [ $USER = "teacher" ] || echo user is  ${USER} ,Not teacher
user is root ,Not teacher
[root@zx ~]#
[root@zx ~]#
```

例、只要 /etc/rc.d/rc.local 或者 /etc/init.d/rc.local 中有一个是文件，则显示“YES”，否则无任何输出。

```
[root@zx ~]# [ -f /etc/rc.d/rc.local ] || [ -f /etc/init.d/rc.local ] && echo "YES"
YES
[root@zx ~]#
```

例、测试 /etc/profile 文件是否有可以执行权限，若确实没有可执行权限，则提示 No x mode 信息。

```
[root@zx ~]# [ ! -x "/etc/profile" ] && echo "No x mode"
No x mode
[root@zx ~]#
```

例、若当前的用户是 root 且使用的 Shell 程序是 /bash/bash，则显示“YES”否则无任何输出。

```
[root@zx ~]# echo $USER $SHELL
root /bin/bash
[root@zx ~]# [ $USER = "root" ] && [ $SHELL = '/bin/bash' ] && echo "YES"
YES
[root@zx ~]#
```

2、if 语句的结构

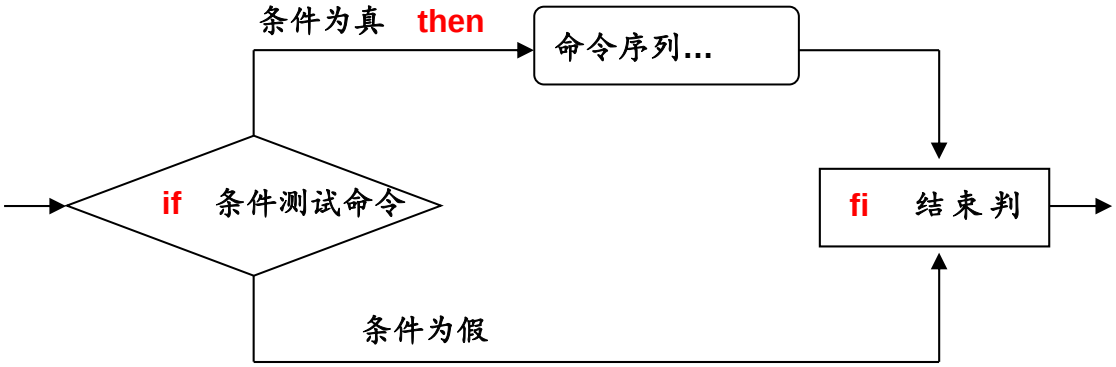
上一小节中学习了常用的一些条件测试操作，实际上使用和&&和||逻辑测试也可以完成简单的判断并执行相应的操作，但是当需要选择执行的命令语句较多时，再使用这种方式将使命令行语句显得很复杂，难以阅读。而使用 if 语句，则可以更好的体现有选择性执行的程序结构，使得层次分明，清晰易懂。
语句的选择结构是由易到难可以分为三种类型，分别适用于不同的各种场合。

1) 单分支的语句

单分支的 if 语句是最简单的选择结构，这种结构只判断指定的条件，当条件成立时执行相应的操作，否则不做任何操作。单分支使用的语句格式如下。

```
if    条件测试命令
then
    命令序列
fi
```

在上述语句中，首先通过 if 判断条件测试命令的返回状态值是否为 0（条件成立）如果是，则执行后面的一条或多条可执行语句（命令序列）一直到为 fi 表示结束；如果条件测试命令在返回状态值不为 0（条件不成立），则直接去执行后面的语句（如图 12-1 所示）。



如图 12-1　单分支的 if 语句结构

在上述格式中，then 关键字可以与 if 写在一行中，其间通过分号“;”进行分隔即可，例如，使用 if 条件测试命令：then 的形式。在 Shell 环境中，分号“;”也可以作为多条命令的分隔符，使用分号分隔的命令将按照顺序依次被执行。

2）双分支的语句

双分支的 if 语句使用了两路命令操作，在“条件成立”、“条件不成立”时分别执行不同的命令序列。双分支使用的语句格式如下。

```
if    条件测试命令
then
    命令序列 1
else
    命令序列 2
fi
```

在上述语句中，首先通过判断条件测试命令的返回状态值是否为 0（条件成立），如果是，则执行后面的一条或多条可执行语句（命令序列 1），然后跳转至结束判断；如果条件测试命令的返回状态不为 0（条件不成立），则执行 else 后面的语句，一直到 fi 表示结束（如图 12-2 所示）

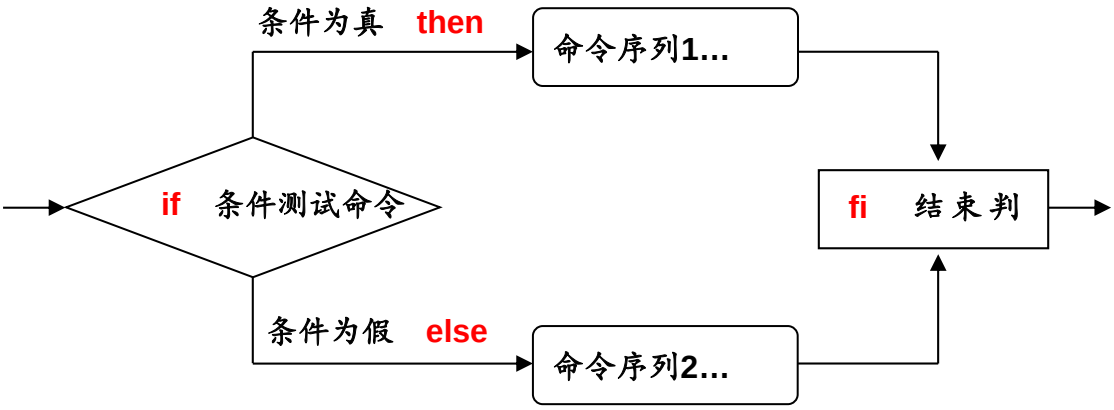


图 12-2　双分支的 if 语句结构

3）多分支的语句

由于 if 语句可以根据条件测试命令的两种状态分别进行操作，所以能够嵌套使用，进行多次判断（例如，首先判断某学生的得分是否及格，若及格则再次判断是否高于 90 分……）多重支使用的语句格式如下。

```
if    条件测试命令 1
 then
    命令序列 1
elif    件测试命令 2
then
    命令序列 2
else
    命令序列 3
fi
```

上面的语法格式中只嵌套了一个 elif 语句，实际上语句中可以嵌套多个 elif 语句，if 语句的嵌套在编写 shell 脚本时并不常用，因为多重嵌套容易使程序结构变得复杂。需要使用多重分支程序结构时，更多的是使用 case 语句来实现（将在后面小节中讲解）。

使用多分支的语句结构时，会依次对多个条件进行测试，一旦条件不成立时立即退出选择结构，否则将执行相应的命令序列后再跳转至 fi，结束判断（如图示 12-3 所示）。

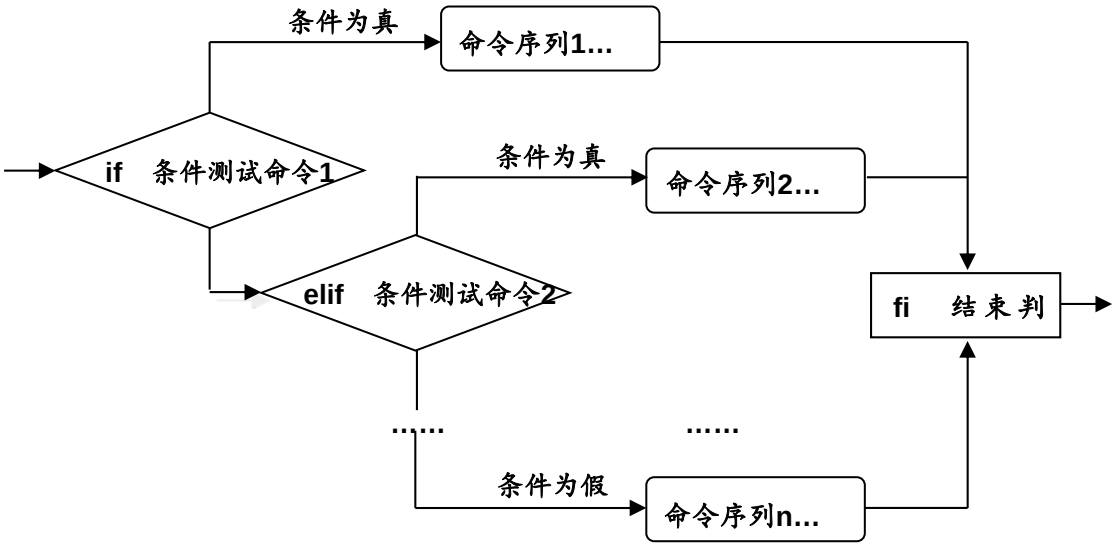


图 12-3　多分支的 if 语句结构

3、if 语句应用系列

if 语句是在 Shell 脚本中使用最多的一种程序结构，本小节将通过脚本实例的方式来展示 if 选择结构的具体用法。

例、创建脚本 chklog.sh，检查文件 /var/log/messages 是否存在，若存在则统计文件内容的行数并输出，否则不做任何操作（合理使用变量，可以提高编写效率）。

```
[root@zx ~]# vi chklog.sh
[root@zx ~]# cat chklog.sh
#!/bin/bash
LogFile="/var/log/messages"
if [ -f $LogFile ]
then
   wc -l $LogFile
fi
[root@zx ~]# ./chklog.sh
2820 /var/log/messages
[root@zx ~]#
```

例、创建脚本 mkbak.sh，提示用户指定备份目录的路径，若目录已存在则显示提示信息后跳过，否则显示相应提示信息后创建该目录。

```
[root@zx ~]# vi mkbak.sh
[root@zx ~]# cat mkbak.sh
#!/bin/bash
read -p "What is your backup directory:" BakDir
if [ -d $BakDir ]
then
   echo "$BakDir already exist."
else
  echo "$BakDir is not exist,will make it."
  mkdir $BakDir
fi
[root@zx ~]#
```

```
[root@zx ~]# chmod a+x mkbak.sh
[root@zx ~]# ./mkbak.sh
What is your backup directory:/opt/backup          //从键盘输入
/opt/backup is not exist,will make it.
[root@zx ~]# ls -ld /opt/backup
drwxr-xr-x. 2 root root 4096  2月   3 17:45 /opt/backup
[root@zx ~]#
```

例、创建 chkuser.sh 脚本，统计当前登录到系统中的用户数量，并判断是否超过三个，若是则显示实际数量并给出警告信息，否则列出登录的用户账号名称用所在终端。

```
[root@zx ~]# vi chkuser.sh
[root@zx ~]# cat chkuser.sh
#!/bin/bash
UserNum = `who | wc -l`
if [$UserNum -gt 3 ] ;then
  echo "ALert,too many login users (Total:$UserNum)."
else
  echo "Login users:"
  who | awk '{print $1,$2}'
fi
[root@zx ~]# ./chkuser.sh
./chkuser.sh: line 2: UserNum: command not found
./chkuser.sh: line 3: [: -gt: unary operator expected
Login users:
root tty1
root pts/0
root pts/1
[root@zx ~]#
```

例、创建脚本 chpostfix.sh，检查 postfix 进程是否已经存在，若已经存在则输出 postfix service is running.；否则检查是否存在 /etc/rc.d/init.d/postfix 可执行脚本，存在则启动 postfix 服务，否则提示 no postfix script file。

```
[root@zx ~]# cat chkpostfix.sh
#!/bin/bash
service  postfix status &> /dev/null
if [ $? -eq 0 ] ; then
  echo "postfix service is runnning."
elif [ -e "/etc/rc.d/init.d/postfix" ] ;then
  service postfix start
else
  echo "no postfix script file."
fi
[root@zx ~]#
```

例、编写脚本 chkmailsrv.sh 每隔五分钟监测一次 postfix 服务进程的运行状态，若发现进程已终止，则在 /var/log/messages 文件中追加写入日志信息（包括当时时间），并重启 postfix 服务，否则不进行任何操作。

```
[root@zx ~]# cat chkmailsrv.sh
#!/bin/bash
/sbin/service postfix status &> /dev/null
if [ $? -ne 0 ] ; then
  echo "At time:`date`:postfix server is down." >> /var/log/messages
 /sbin/service postfix restart
fi
[root@zx ~]# crontab -l
*/5     *       *       *       *       /root/chkmailsrv.sh
[root@zx ~]#
```

例：.创建一个 shell 脚本 /root/program：

--当你输入“kernel”参数时，此 shell 脚本就返回“user“参数

--当你输入”user“参数时，此 shell 脚本就返回”kernel”

--当此脚本不输入任何参数或者输入的参数是按照要求输入的话，那么就会输出标准错误“usage:/root/program kernel|user”。

(1).vim /root/program

```
#!/bin/bash
  if [ $1 = "user" ]
 then
     echo "kernel"
 elif
     [ $1  = "kernel" ]
 then
     echo "user"
 else
      echo "useage:/root/program kernel|user."
fi
```

12.5.2 使用 for 循环语句

在脚本中使用 for 循环语句时，可以为变量设置一个取值列表，每次读取列表中不同的变量值并执行相关命令操作，变量值用完以后则退出循环。Shell 中的 for 语句不需要执行条件判断，其使用变量的取值来自于预先设置的列表。

1、语句的结构

循环的语句格式如下。

```
for  变量名  in  取值列表
do
命令序列
done
```

在上述语句中，使用 in 关键字为用户自定义变量设置了一个取值列表（以空格分隔的多个值），for 语句第一次执行时首先将列表中的第一个取值赋给该变量，然后执行 do 后边的命令序列；然后 再将列表中的第二个取值赋给该变量，然后执行 do 后边的命令序列……如此循环，直到取值列表中的所有值都已经用完，最后跳至 done 语句，表示结束循环（如图 12-4 所示）。

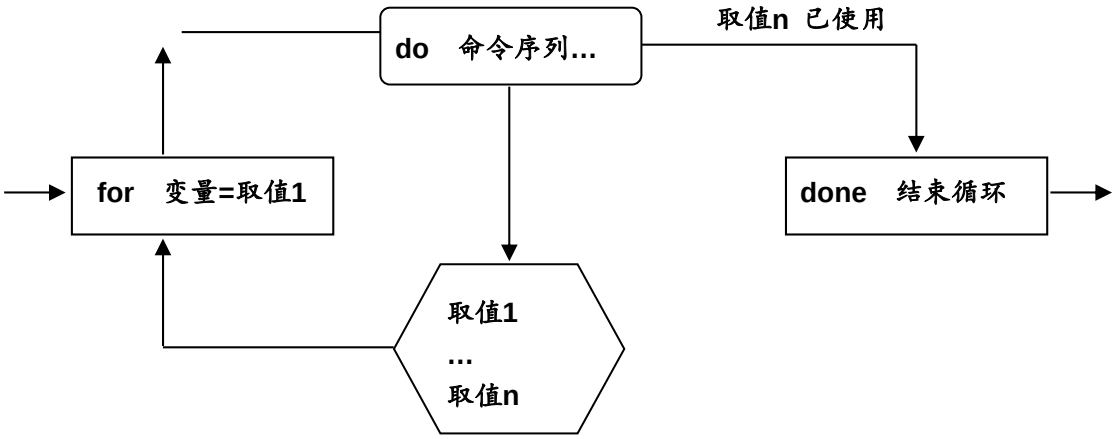


图 12-4　　for 循环语句的结构

2、for 语句应用示例

for 语句可以用于对一些特定列表值的处理（如过滤搜索到的文件），本小节将通过脚本实例的方式来展示 for 循环语句结构的具体方法。

例、依次输出三条文字信息，包括一天中的“Morning”、“Noon”、“Evening”字串。

```
[root@zx ~]# cat showday.sh
#!/bin/bash
for TM in "Morning" "Noon" "Evening"
do
  echo "The $TM of the day."
done
[root@zx ~]# chmod a+x showday.sh
[root@zx ~]# ./showday.sh
The Morning of the day.
The Noon of the day.
The Evening of the day.
[root@zx ~]#
```

例、对于使用/bin/bash 作为登录 Shell 的系统用户，检查他们在 /opt 目录中拥有的子目录或文件数量，如果超过 100 个则列出具体数值及对应的用户帐号。

```
[root@zx ~]# cat chkfileown.sh
#!/bin/bash
DIR="/opt"
LMT=2
ValidUsers=`grep "/bin/bash" /etc/passwd | cut -d ":" -f 1`
for UserName in $ValidUsers
 do
   Num=`find $DIR -user $UserName | wc -l`
   if [ $Num -gt $LMT ] ; then
     echo "$UserName have $Num files."
   fi
done
[root@zx ~]# ./chkfileown.sh
root have 5 files.
[root@zx ~]#
```

例、计算 /etc 目录中所有 *.conf 的形式的配置文件所占用的总空间大小。

```
[root@zx ~]# cat confsize.sh
#!/bin/bash
SizeNums=$(ls -l $(find /etc -type f -a -name *.conf) | awk '{print $5}')
Total=0                                                    //提取各文件的大小值。
for i in $SizeNums
do
  Total=`expr $Total + $i`                                 //计算每个文件的大小之和
done
echo "Total size of conf files:$Total bytes."
[root@zx ~]# chmod a+x confsize.sh
[root@zx ~]# ./confsize.sh
Total size of conf files:559508 bytes.
[root@zx ~]#
```

12.5.3 使用 while 循环语句

在 Shell 脚本中使用 while 循环语句时，将可以根据特定的条件重复执行一个命令列表，直到该条件不再满足时为止。除非有特别需要，否则在脚本程序中应该避免出现无限循环执行命令的情况，因为若无法跳出循环的话，后边的命令操作将无法执行。为了控制循环次数，

通常会在执行的命令序列中包含修改测试条件的语句，当循环达到一定次数后，测试条件将不再成立，从而可以结束循环。

1、 while 语句的结构

循环的语句格式如下。

```
while   条件测试命令
do
  命令序列
done
```

在上述语句中，首先通过 while 判断条件测试命令的返回状态值是否为 0（条件成立），如果是则执行 do 后边的命令序列，然后返回 while 再次进行条件测试并判断返回状态值，如果条件仍然成立，则继续执行 do 后边的命令序列，然后返回到 while 重复条件测试……如此循环，直到所测试的条件不成立时，跳转到 done 语句，表示结束循环。(如图 12-5 所示)。

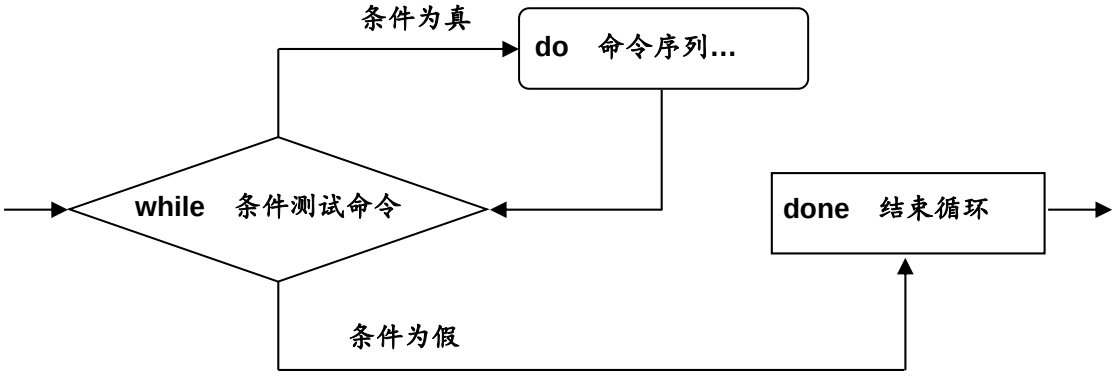


图 12-5　　while 循环语句的结构

使用 while 循环语句时，有几代人上特殊的条件测试返回值，即 true（真）、false（假）。使用 true 作为测试条件时，条件将永远成立，循环体现内的语句将不会被执行。这两个特殊值也可以用在 if 语句的条件测试中。

2、 while 语句应用实例

while 语句可以用于需要重复操作的循环系统管理任务，并能够通过设置循环条件来灵活实现各种管理任务。本小节将通过脚本实例的方式来展示循环结构的具体用法。

例、由用户从键盘输入一个大于 1 的整数（如 50），并计算从 1 到该数之间各整数的和。

```
[root@zx ~]# cat sumint.sh
#!/bin/bash
read -p "Input a number (>1):" UP     //读取数据保存到UP变量
i=1                                   //从整数1开始计数
Sum=0                                 //Sum变量用于保存各数这和
while [ $i -le $UP ]          //加数小于用户指定的上限时执行循环
do
 Sum=`expr $Sum + $i`             //将整数i值累加给Sum变量
 i=`expr $i + 1`                  //将 i  加1
done
echo "The sum of 1-$up is:$Sum"          输出求和结果
[root@zx ~]#
```

例、批量添加 20 个系统用户帐号，用户名称依次为 stu1、stu2、stu3…..，各用户的初始密码均设置为“123456”。

```
[root@zx ~]# cat add20users.sh
#!/bin/bash
i=1
while [ $i -le 20 ]
do
useradd stu$i
echo "123456" | passwd --stdin stu$i &> /dev/null
  i=`expr $i + 1`
done
[root@zx ~]#
```

例、编写一个批量删除用户的脚本程序，将上列中添加的 20 个用户删 除。

```
#!/bin/bash
i=1
while [ $i -le 20 ]
do
userdel -r stu$i
  i=`expr $i + 1`
done
[root@zx ~]#
```

12.5.4 其他控制语句

学会条件测试操作以及 if、for、while 语句的使用以后，编写一般的系统管理任务脚本已经基本可以胜任了。当然，这其中需要大家将之前学习过的各种命令、管道、重定向等操作融会贯通，才能编写出更有效的脚本程序。实际应用中的脚本程序灵活多变，即使是完成同一项任务，也可能有许多种不同的实现方式，大家应该勤加练习，在实践过程中，慢慢去领略。

本节中将简单介绍几个其他的控制语句，以便在编写脚本程序的过程中可以有更多的选择。主要包括 case 分支语句、until 循环、shift 移位以及 break 和 continue 循环中断语句。

1、 case 语句

case 语句适用于需要进行多重分支的应用情况，前面使用的多分支 if 语句，通常都

可以修改为 case 语句，这样程序结构将会更加清晰。
分支语句的格式如下。

Case　变量值
　模式 1）
命令序列 1
　;;
　模式 2）
命令序列 2
　;;
　…….
　*）
　默认执行的命令的序列
　Esac

在上述语句中，将使用 case 后边的“变量值”与模式 1、模式 2、……等进行逐一比较（各模式中为用户预设的固定值）直到找到与之相匹配的值，然后执行该模式下的命令序列，当遇到双分号后跳至 esac，表示结束分支。如果一直不到与之相匹配的值，则执行最后一个模式*后的命令序列，直到遇到 esac 后结束分支。(如图 12-6)。

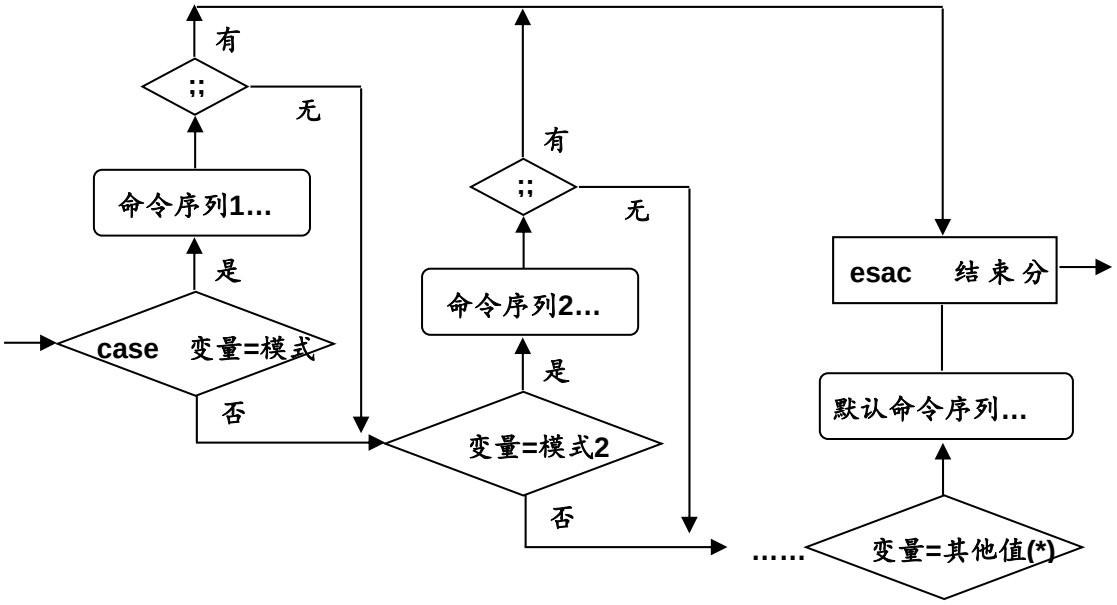


图 12-6　　case 分支语句结构

case 语句的结构特点

- case 行尾必须为单词 in，每一模式必须以右括号“)”结束。
- 双分号;；表示命令序列的结束。
- 匹配模式中可以使用方括号表示一个连续的范围，例如[0—9]；使用竖杠符号表示或，例如“A|B”（A 或者 B）。
- 最后的*表示默认模式，当使用前面的各种模式均无法匹配该变量时，将执行*后的命令序列。

例、由用户从键盘输入一个字符，并判断该字符是否为字母、数字或者其他字符，并输出相应的提示信息。

```
[root@zx ~]# cat hitkey.sh
#!/bin/bash
read -p "Press some key,then press Retrun:" KEY
case "$KEY" in
 [a-z]|[A-Z])
   echo "It is a letter."
;;
 [0-9])
  echo "It is a digit."
;;
 *)
 echo "It is function keys."
esac
[root@zx ~]#
```

在 Linux 系统中很多脚本文件都使用了 case 分支语句，其中 /etc/rc.d/init.d 目录下的各种系统服务脚本最为典型。有兴趣的同学可以通过这些脚本来学习 case 语句的使用。例如在 crond 服务脚本的 case 语句结构中，用于匹配的变量为$1，也就是用户输入的第一个参数。当用户执行 /etc/rc.d/init.d/crond start 命令时，$1 的值就是 start，因此脚本程序会执行 start) 模式后边的命令序列，用于启动 crond 服务。

例：.创建一个 shell 脚本 /root/program：

--当你输入“kernel”参数时，此 shell 脚本就返回“user“参数

--当你输入”user“参数时，此 shell 脚本就返回”kernel”

--当此脚本不输入任何参数或者输入的参数是按照要求输入的话，那么就会输出标准错误“usage:/root/program kernel|user”。

```
(1).vim /root/program
   #!/bin/bash
   case $1 in
      kernel)
        echo user
        ;;
     user)
        echo kernel
       ;;
      *)
        echo "usage:/root/program kernel|user."
      ;;
   esac
```

2、 until 循环

until 循环语句与 while 循环语句的结构非常类似，同样是根据特定的条件决定是否执行循环体中的命令序列，只不过 while 循环语句是当测试条件成立时执行，而 until 循环是当测试条件成立时结束循环，循环语句的格式如下。

until　条件测试命令
do
命令序列
done

until 的含义是“直到……为止”，因此同样会首先执行条件测试并判断其返回值，若条件不成立则执行循环，一直到该测试条件成立时为止，即退出循环.。（如图 12-7）

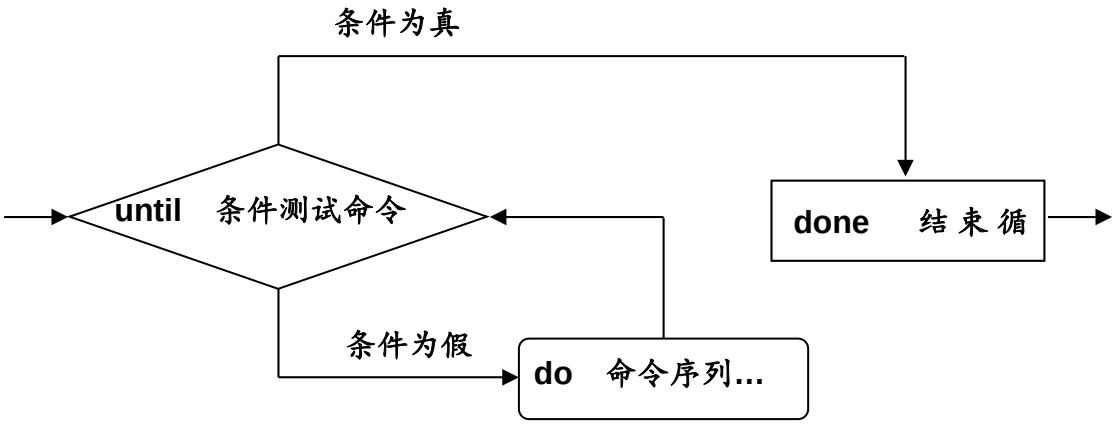


图 12-7　until 循环语句的结构

大多数时候，通过将测试条件反置，while 和 until 语句可以互相交换。

3、shift 语句

shift 实际上是 Bash 中一个特殊的内部命令，但是在命令行较少时用，而更多的是用在 Shell 脚本程序中，执行 shift 命令后，位置变量中($1~$9)的命令行参数会依次向左传递。例如：

若当前脚本程序获得的位置变量如下。

$1=file1、$2=file2、$3=file3、$4=file4

则执行一次 shift 命令操作后（丢弃最左边的一个值），各位置变量的值将变为如下所示。

$1=file2、$2=file3、$3=file4

再次 shift 执行命令操作后，各位置变量的值将变为如下所示.

$1=file3、$2=file4

合理利用 shift 语句，可以在位置参数的数量不固定的情况灵活实现程序的功能。

例、编写一个程序，计算多个整数值的和，需要计算的各个数值由用户在执行脚本时作为命令行参数给出。

```
[root@zx ~]# cat sumer.sh
#!/bin/bash
Result=0
while [ $# -gt 0 ]
do
  Result=`expr $Result + $1`
  shift
done
echo "The sum is:$Result"

[root@zx ~]# ./sumer.sh 12 23
The sum is:35
[root@zx ~]# ./sumer.sh 12 23 34
The sum is:69
[root@zx ~]#
```

4、循环控制语句

在使用 for、while 或 until 循环语句以及 case 分支语句的过程中，当满足特定条件时可能会需要中断循环体的执行，或者需要直接转到开关重新判断测试条件（而不再执行后边的命令序列）。这时候，可以使用 break 和 continue 语句对执行流程进行控制，这两个语句在与其他大部分编程语言中的含义是类似的。

1) break 命令

break 即中断的意思，用于跳出当前据的循环体，但是并不退出程序。

例、循环提示用户输入字符串，并将每次输入的内容保存到临时文件 /tmp/input.txt 中，当用户输入“END”字符串时退出循环体，并统计出 input.txt 文件中的行数、单词数、字节数等信息，统计完后删除临时文件。

```
[root@zx ~]# cat inputbrk.sh
#!/bin/bash
while true
do
    read -p "Input a string:" STR
    echo $STR >> /tmp/input.txt
    if [ $STR = "END" ] ; then
      break
    fi
done
wc /tmp/input.txt
rm -f /tmp/input.txt

[root@zx ~]# ./inputbrk.sh
Input a string:my
Input a string:name
Input a string:is
Input a string:lwj
Input a string:END
 5  5 19 /tmp/input.txt
[root@zx ~]#
```

2) continue 命令

continue 即继续的意思，用于暂停本次循环，跳转至循环语句的顶部重新测试条件。本次执行过程中 continue 后的命令序列将被忽略。

例、删除系统中的 stu1~stu20 各用户帐号，但是 stu8、stu18 除外。

```
[root@zx ~]# cat delsome.sh
#!/bin/bash
i=1
while [ $i -le 20 ]
do
  if [ $i -eq 8 ] || [ $i -eq 18 ] ;then
  let i++
  continue
 fi
   userdel -r stu$i
   let i++
done
[root@zx ~]#
```

12.6 监控系统脚本例子

例：监控系统硬盘、内存、CPU、登录用户数，超出警戒值则告警。

```
# /scripts/sys-warning.sh
#监控系统硬盘、内存、CPU、登录用户数，超出警戒值则告警。

#!/bin/bash
 #提取系统日期和时间
  date=`date`

#提取本服务器的 IP 地址信息
IP=`ifconfig eth0 | grep "inet addr" | cut -f 2 -d ":" | cut -f 1 -d " "`

# 1、监控硬盘分区，当使用超过 85%的时候发告警：(以下以 boot 分区为例)

#提取 boot 分区（/dev/sda1）已用的百份比值（只取整数部分）
disk_boot=`df -h | grep /boot | awk '{print $5}' | cut -f 1 -d "%"`

#设置剩余硬盘容量的告警值为 85%，如果当前硬盘使用超过 85%，立即告警
 if (($disk_boot> 85)); then
      echo "$date,$IP 服务器 boot 分区 使用率已经超过 85%，请及时处理。" >>
/opt/warning.log
      echo "$date,$IP 服务器 boot 分区 使用率已经超过 85%，请及时处理。"|mail -s
"$IP 服务器硬盘告警" test@163.com
fi

# 2、监控系统 cpu 的情况，当使用超过 80%的时候发告警：
```

```
#取当前空闲 cpu 百份比值（只取整数部分）
cpu_idle=`top -b -n 1 | grep Cpu | awk '{print $5}' | cut -f 1 -d "."`

#设置空闲 cpu 的告警值为 20%，如果当前 cpu 使用超过 80%（即剩余小于 20%），立即告警
if (($cpu_idle < 20)); then
   echo "$date,$IP 服务器 cpu 剩余$cpu_idle%，使用率超过 80%，请及时处理。"  >> /opt/warning.log
    echo "$date,$IP 服务器 cpu 剩余$cpu_idle%，使用率已经超过 80%，请及时处理。" | mail -s "$IP 服务器 CPU 告警"  test@163.com
fi

 # 3、监控系统交换分区 swap 的情况，当使用超过 80%的时候发告：

#系统分配的交换分区总量
swap_total=`free -m | grep Swap | awk '{print  $2}'`

#当前剩余的交换分区 free 大小
swap_free=`free -m | grep Swap | awk '{print  $4}'`

#当前已使用的交换分区 used 大小
swap_used=`free -m | grep Swap | awk '{print  $3}'`

if (($swap_used != 0)); then
#如果交换分区已被使用,则计算当前剩余交换分区 free 所占总量的百分比,用小数来表示,要在小数点前面补一个整数位 0
   swap_per=0`echo "scale=2;$swap_free/$swap_total" | bc`

#设置交换分区的告警值为 20%(即使用超过 80%的时候告警)。
   swap_warn=0.20

#当前剩余交换分区百分比与告警值进行比较（当大于告警值(即剩余 20%以上)时会返回 1，小于(即剩余不足 20%)时会返回 0 ）
   swap_now=`expr $swap_per \> $swap_warn`

#如果当前交换分区使用超过 80%（即剩余小于 20%，上面的返回值等于 0），立即发邮件告警
  if (($swap_now == 0)); then
    echo "$date,$IP 服务器 swap 分区只剩下 $swap_free M 未使用，剩余不足 20%，使用率已经超过 80%，请及时处理。" >> /opt/warning.log
    echo "$date,$IP 服务器 swap 分区只剩下 $swap_free M 未使用，剩余不足 20%，使用率已经超过 80%，请及时处理。" | mail -s "$IP 服务器内存告警"  test@163.com
  fi
 fi
```

```
# 4、监控系统用户登录的情况，当用户数超过 3 个的时候发告警邮件：

#取当前用户登录数（只取数值部分）
users=`uptime | awk '{print $4}'`

#设置登录用户数的告警值为 3 个，如果当前用户数超过 3 个，立即发邮件告警
if (($users >= 3)); then
     echo "$date,$IP 服务器用户数已经达到$users 个，请及时处理。" >>
/opt/warning.log
      echo "$date,$IP 服务器用户数已经达到$users 个，请及时处理。" | mail -s "$IP
服务器用户数告警" test@163.com
fi
```

二、加入任务计划：每十分钟检测一次。

```
  # chmod a+x /scripts/sys-warning.sh

 # crontab -e
     */10 * * * *  /scripts/sys-warning.sh
     # service crond restart
```

第十三章　配置 BASH 环境

13.1 Shell 的环境变量

这里所说的环境变量是指用户登录后 Linux 系统预先设好的一类 Shell 变量，其功能是设置用户的 Shell 工作环境，包括用户的宿主目录、命令查找路径、用户当前的目录、登录的终端等。在实际使用过程中，环境变量并没有严格的区分定义，用户自己设置的变量也可以作为环境变量。

环境变量的名称比较固定，通常使用大写字母、数字和其他字符组成，而不使用小写字母。环境变量的值一般由 Linux 系统自行维护，会随着用户状态的改变而改变，用户可以通过读取环境变量来了解自己的当前状态。

1、查看环境变量

通过 set 命令可以查看系统中所有 Shell 变量（包括环境变量和用户自定义变量），由于内容输出较多，建议使用 less 命令分页查看。

常用环境变量：

PATH	决定了 shell 将到哪些目录中寻找命令或程序

HOME	当前用户主目录
HISTSIZE	历史记录数
LOGNAME	当前用户的登录名
USER	当前用户名
UID	当前用名的 UID
HOSTNAME	指主机的名称
SHELL	前用户 Shell 类型
LANGUGE	语言相关的环境变量，多语言可以修改此环境变量
MAIL	当前用户的邮件存放目录
PS1	基本提示符，对于 root 用户是#，对于普通用户是$
PS2	附属提示符，默认是“>”

例：以分号分隔，显示当前的用户的用户名、宿主目录、登录 Shell。

```
[root@zx ~]# echo "$USER;$HOME;$SHELL"
teacher;/root;/bin/bash
[root@zx ~]#
```

例：查看当前命令的搜索路径，并将/opt/bin 目录添加到现有搜索路径中去，从而可以直接执行此目录中的命令。

```
[root@zx ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bi
:/root/bin:/root/bin
[root@zx ~]# PATH="/opt/bin:$PATH"
[root@zx ~]# PATH
bash: PATH: command not found
[root@zx ~]# echo $PATH
/opt/bin:/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/us
/bin:/bin:/root/bin:/root/bin
```

2、环境变量的配置文件

用户可在当前的 Shell 环境中直接为环境变量赋值，但需要长期变更所使用的某个环境变量时，可以修改配置文件。

在 Linux 系统中，用户环境变量的设置工作习惯上在 /etc/profile 文件及宿主目录中 .bash_profile 文件中进行，前者称为全局配置文件（对所有用户起作用），后者称为用户配置文件（允许覆盖全局配置）。

例：在当前用户环境中，将用于限制历史命令的环境变量 HISTSIZE 的值改为 100。

```
[root@zx ~]# echo $HISTSIZE
1000
[root@zx ~]# export HISTSIZE=100
[root@zx ~]# echo $HISTSIZE
100
[root@zx ~]#
```

例：编辑“~/.bash_profile”文件，修改 PATH 的设置，以便用户在下次登录后能够使用服务/opt/bin 目录作为默认的搜索路径。

```
#  vi  /root/.bash_profiel
   PATH=$PATH:$HOME/bin:/opt/bin
    Export  PATH
```

注意：登录的几个脚本文件

/etc/profile

/etc/bashrc

~/.bash_profile

~/.bashrc

~/.bash_logout

每一个文件都有特殊的功用并对登陆和交互环境有不同的影响。

/etc/profile 和 ~/.bash_profile 是在启动 一个交互登陆 shell 的时候被调用。

/etc/bashrc 和 ~/.bashrc 是在一个交互的非登陆 shell 启动的时候被调用。

~/.bash_logout 在用户注销登陆的时候 被读取

一 个交互的登陆 shell 会在 /bin/login 成功登陆之后运行。一个交互的非登陆 shell 是通过命令行来运行的，如[prompt]$/bin/bash。一般一个非交互的 shell 出现在运行 shell 脚本的时候 。之所以叫非交互的 shell，是因为它不在命令行上等待输入而只是执行脚本程序。

3、命令别名

Bash 中的命令别名功能可以将频繁使用的复杂命令定义为简短的别名，当用户需要执行该条复杂命令时，只需要使用别名即可完成对应的操作，降低了操作复杂性，提高了输入较率，别名功能主要通过这两个命令进行设置和取消。

1) 查看已设置的别名

直接执行 alias 命令可以查看当前用户环境中已设置的所有命令别名，若只需要查看待定的命令别名，可以使用别名名称作为参数。

在系统中默认情况下已经定义好了几个别名，例如，ll 是 ls -l 命令的别名，这样用户在需要以长格式显示文件信息时，就不需要每次都输入 ls -l 命令，直接输入 ll 命令即可。

2) 设置命令别名

用户可以根据自己的使用习惯定义特定的命令别名,具体格式可参考 alias 命令的输出结果，基本的形式为：alias 别名=’实际命令’

3) 取消已设置的命令别名

需要取消系统中定义的命令别名时，可以使用 unalias 命令。使用别名名称作为参数可以删除特定的命令别名，结合-a 选项使用时将删除所有已定义的命令别名。

使用alias命令定义新的别名或使用unalias命令取消已定义的别名都只在用户的当前Shell 环境中有效,下次启动后将恢复初始的别名设置。因此对于用户经常使用的命令别名，应该将相应的别名设置命令添加到宿主目录的 .bashrc 文件中。

第十四章 Docker

14.1 Docker 介绍

1、docker 是什么？

Docker 的英文本意是“搬运工”，在程序员的世界里，Docker 搬运的是集装箱（Container），集装箱里装的是任意类型的 App，开发者通过 Docker 可以将 App 变成一种标准化的、可移植的、自管理的组件，可以在任何主流系统中开发、调试和运行。最重要的是，它不依赖于任何语言、框架或系统。

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。几乎没有性能开销,可以很容易地在机器和数据中心中运行。

docker 容器可以在几乎所有的环境中运行，物理机、虚拟机、公有云、私有云、个人电脑、服务器等等。 docker 容器兼容很多平台，这样就可以把一个应用程序从一个平台迁移到另外一个。

为什么要使用容器？例如：想用 MySQL,那就要装 MySQL,可能要再装一堆依赖库，根据你的操作系统平台和版本进行设置，有时候还要从源代码编译报出一堆莫名其妙的错误,可不是这么好装。而且万一你机器挂了，所有的东西都要重新来，可能还要把配置在重新弄一遍。但是有了容器，你就相当于有了一个可以运行起来的虚拟机，只要你能运行容器，MySQL 的配置就全省了。而且一旦你想换台机器，直接把这个容器端起来，再放到另一个机器就好了。硬件，操作系统，运行环境什么的都不需要考虑了。

在公司中的一个很大的用途就是可以保证线下的开发环境、测试环境和线上的生产环境一致。经常碰到这样的事情，开发把东西做好了给测试去测，一般会给一坨代码和一个介绍上线步骤的上线单。结果代码在测试机跑不起来，开发就跑来跑去看问题……，若果利用容器的话，那么开发直接在容器里开发，提测的时候把整个容器给测试，测好了把改动改在容器里再上线就好了。通过容器，整个开发、测试和生产环境可以保持高度的一致。

此外容器也和 VM 一样具有着一定的隔离性，各个容器之间的数据和内存空间相互隔离，可以保证一定的安全性。

2、Docker 的体系结构

docker 使用 C/S 架构，docker daemon 作为 server 端接受 client 的请求，并处理（创建、运行、分发容器），他们可以运行在一个机器上，也通过 sockerts 或者 RESTful API 通信。

Docker daemon 一般在宿主主机后台运行，用户使用 client 而直接跟 daemon 交互。Docker client 以系统做 bin 命令的形式存在，用户用 docker 命令来跟 docker daemon 交互。

1）Docker 的内部组件

- Docker images

docker images 就是一个只读的模板。比如：一个 image 可以包含一个 ubuntu 的操作系统，里面安装了 apache 或者你需要的应用程序。images 可以用来创建 docker containers，docker 提供了一个很简单的机制来创建 images 或者更新现有的 images,你甚至可以直接从其他人那里下载一个已经做好的 images。

- Docker registries

Docker registries 也叫 docker 仓库，它有公有仓库和私有仓库 2 种形式，他们都可以用来让你上传和下载 images。公有的仓库也叫 Docker Hub。它提供了一个巨大的 image 库可以让你下载，你也可以在自己的局域网内建一个自己的私有仓库。

- Docker containers

Docker containers 也叫 docker 容器，容器是从 image 镜像创建的。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、安全的平台。

3、Docker 和虚拟机区别

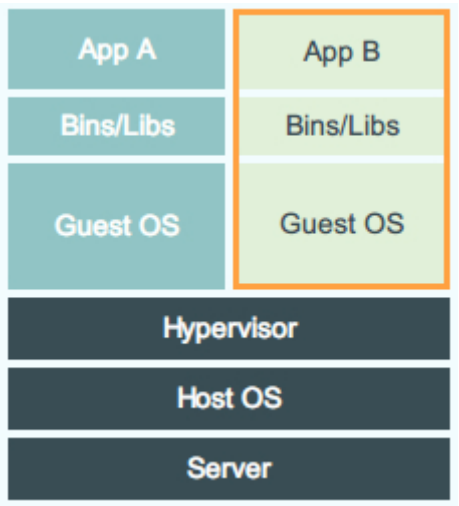
Docker 容器相对于虚拟机有以下几个优点：

- 容器建立、删除和移动比虚拟机快速
- 容器不需要支持整个操用系统，只需要内核实时运行的应用。
- 启动速度快，容器通常在一秒内可以启动
- 资源利用率高，一台普通 PC 可以跑上千个容器

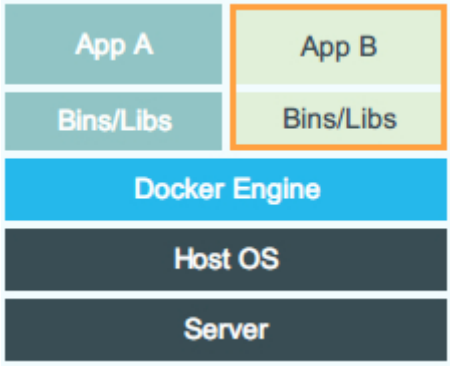
虚拟机相对于 Docker 容器有以下几个优点：

- 每个虚拟机运行自己内核和完整操作系统，相互隔离更强。
- 虚拟机能运行时可以在线从一个 hypervisor 节点迁移到另一节点，容器从一台机迁移到另一台机前必须停止。

性能开销小，虚拟机通常需要额外的 CPU 和内存来完成 OS 的功能，这一部分占据了额外的资源。为啥相似的功能在性能上会有如此巨大的差距呢，其实这和他们的设计的理念是相关的。VM 的 Hypervisor 需要实现对硬件的虚拟化，并且还要搭载自己的操作系统，自然在启动速度和资源利用率以及性能上有比较大的开销虚拟机 的设计图如下：



而 Docker 的设计图是这样的：



Docker 几乎就没有什么虚拟化的东西，并且直接复用了 Host 主机的 OS，在 Docker Engine 层面实现了调度和隔离。 Docker 的容器利用了 LXC，管理利用 namespaces 来做权限的控制和隔离，cgroups 来进行资源的配置，并且还通过 aufs 来进一步提高文件系统的资源利用率。

14.2 使用 Docker

1、安装启动 Docker(以下在 centos 7.0 上实验)

```
[root@zx ~]#  yum install docker
[root@zx ~]#  systemctl stop firewalld
[root@zx ~]#  systemctl disable firewalld
[root@zx ~]#  setenforce 0
[root@zx ~]#  set[root@zx ~]#  systemctl start docker; systemctl enable docker
```

2、搜索可用的镜像 （比如搜索 centos 并且支持 php 的）

```
[root@zx ~# docker search centos6/php
NAME                                      DESCRIPTION     STARS       OFFICIAL
AUTOMATED
sergeyzh/centos6-php-fpm                                  0                   [OK]
carbonsphere/docker-centos6-php-nginx                     0                   [OK]
melin/centos6-php                                         0
jinqiao/centos6devel-nginx-php5                           0
jinqiao/centos6-nginx-php5                                0
jinqiao/centos6mini-nginx-php5                            0
arkii/centos65-apache2065-php5317                         0
arkii/centos65-httpd2065-php5317                          0
```

3、下载可用的镜像

```
[root@zx ~]# docker pull arkii/centos65-apache2065-php5317
Pulling repository arkii/centos65-apache2065-php5317
ee9a24bdc164: Download complete
```

```
539c0211cd76: Download complete
.....
```

4、查看下载镜像

```
[root@zx ~]#docker images
REPOSITORY                                    TAG                         IMAGE ID
CREATED             VIRTUAL SIZE
arkii/centos65-apache2065-php5317    latest              ee9a24bdc164         4
months ago        996.1 MB
```

5、运行 docker 镜像

```
[root@zx ~]#docker run -i  -t arkii/centos65-apache2065-php5317 /bin/bash
bash-4.1#
bash-4.1# cat /etc/redhat-release
CentOS release 6.5 (Final)

bash-4.1# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
8: eth0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 7e:f2:ed:9a:7b:bc brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.4/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::7cf2:edff:fe9a:7bbc/64 scope link
       valid_lft forever preferred_lft forever
bash-4.1#
```

6、查看正在运行和已退出的容器

```
[root@zx~]#  docker ps -a
CONTAINER ID            IMAGE                                          COMMAND
CREATED             STATUS                          PORTS                NAMES
b3f0ef986790          ribozz/ubuntu-nginx-php:latest    "/bin/bash"           25
minutes      ago                                      Up       25       minutes
trusting_franklin
447ea2dd331d          ribozz/ubuntu-nginx-php:latest    "/bin/bash"           40
```

```
minutes ago      Exited (0) 26 minutes ago   prickly_sinoussi
启动已退出的容器：
[root@zx~# docker start -ai  prickly_sinoussi
```

7、虚拟机和物理机数据的共享

```
[root@zx~]#  mkdir /share
[root@zx~]#  touch /share/a.txt

[root@zx~]# docker images
REPOSITORY                                    TAG                         IMAGE ID
CREATED             VIRTUAL SIZE
arkii/centos65-apache2065-php5317    latest                ee9a24bdc164         4
months ago        996.1 MB

[root@~]# docker run -v /share:/share -i -t arkii/centos65-apache2065-php5317  /bin/bash
bash-4.1# cd /share
bash-4.1# ls
a.txt
bash-4.1#
```

8、下面我们把 docker 虚拟机中的 80 端口 映射到我们物理机的 80 端口

```
[root@zx~]#  docker run -p 10.11.20.11:80:80 -d -i -t ribozz/ubuntu-nginx-php
/bin/bash
03eeee8c4f05b5bcd1132dffa0a6ef19d90c5309b495306aeed7735159850e68
[root@localhost ~]#
```

9、保存虚机机的修改

```
[root@zx~]#  docker ps
CONTAINER ID            IMAGE                                          COMMAND
CREATED             STATUS              PORTS                      NAMES
03eeee8c4f05          ribozz/ubuntu-nginx-php:latest    "/bin/bash"          3
minutes ago       Up 3 minutes        10.11.20.11:80->80/tcp   naughty_nobel
447ea2dd331d          ribozz/ubuntu-nginx-php:latest    "/bin/bash"          56
minutes ago      Up 8 minutes                                 prickly_sinoussi
//把 03eeee8c4f05 提交为新镜像
[root@zx ~]# docker commit 03eeee8c4f05 liweijie/ubuntu-nginx-php
[root@zx~]#  docker run -i -t liweijie/ubuntu-nginx-php /bin/bash
//启动新镜像
[root@zx~]# docker run -i -t liweijie/ubuntu-nginx-php /bin/bash
```

14.3 建立 Docker images

创建镜像有很多方法，官方的推荐是 pull 一个。 这里推荐一个办法就是从一个文件系统 import 一个镜像，可以使用 opvz 的模板来创建（openvz 可以说是容器虚拟化的先锋）

openvz 的模板下载地址如下：

http://openvz.org/Download/templates/precreated

下载完之后，比如：下载了一个 **centos-7-x86_64-minimal.tar.gz** 的镜像

使用以下命令

```
[root@zx~]# systemctl start docker
[root@zx~]# cat centos-7-x86_64-minimal.tar.gz |docker import - liweijie/centos
5cd008f49aee720aff209366483438a264e775d5dfc95deb5571fc5d97858f6c

[root@zx ~]# docker images
REPOSITORY              TAG                         IMAGE ID                CREATED
VIRTUAL SIZE
liweijie                latest                  5cd008f49aee          5 minutes ago
414.7 MB
```

14.4 提交本地的镜像到云上

1、在私有仓库上传、下载、搜索 images

1) 创建私有仓库

```
[root@zx~]# yum -y install docker-registry
[root@zx~]# systemctl start docker-registry
[root@zx~]# systemctl enabel docker-registry

[root@zx~]# firewall-cmd --permanent --add-port=5000/tcp
```

2) 私有仓库上传、下载、搜索 images

```
[root@zx ~]# docker images
REPOSITORY              TAG                         IMAGE ID                CREATED
VIRTUAL SIZE
ubuntu-14.04          latest                  3ba09d0c76f4          15 minutes ago
415.7 MB
liweijie                latest                  5cd008f49aee          51 minutes ago
414.7 MB
//使用 docker tag 将 ubuntu-14.4 这个 image 标记为 10.11.21.235:5000/test
[root@zx ~]# docker tag ubuntu-14.4  10.11.21.235:5000/test

//使用 docker push 上传我们标记的新 image
[root@zx~]# docker push 10.11.21.235:5000/test
```

```
//搜索本地仓库
[root@zx~]# curl http://10.11.21.235:5000/v1/search

//现在我们到另外一台机器上下载这个 images
[root@opnvz ~]# docker pull 10.11.21.235:5000/test
```

1、在公有仓库上传、下载、搜索 images

1)先要到 www.docker.com 上注册账户

2）在 rhel7 中登录你注册的账户

```
[root@zx~]#  docker login index.docker.io
Username: liweijie
Password:
Email: wjli296@sina.com
Login Succeeded
```

（3）上传你自己的镜像到云上

```
[root@zx~]#  docker ps
[root@zx ~]# docker commit 1ad15487827f liweijie/test
[root@zx~]#  docker push liweijie/test
```

我注册的用户名为 liweijie 所以我的仓库名也就是 liweijie，后面的 test 是我的镜像名

等上传成功后，你就可以在 www.docker.com 中登录你的账户，就可以看到你上传的镜像了

在别的电脑执行 docker pull liweijie/test

就可以很方便的把你上传的镜像下载下来使用

rhce-rhel7-254 综合练习题

完成下列测试题

1、在 ServerX 上有两个网络接口，名为 slave1 和 slave2。这两个接口都必须属于一个名叫 team1 的 team 设备。

Team 配置为 activebackup 的 runner,并配置以下 IP 地址能 ping 通。

192.168.0.100/24

Fd00:ba5e:badd:x::1/64

解答：

1）创建一个 team 连接名和接口名都为 team1

```
[root@localhost ~]# nmcli con add type team con-name team1 ifname team1 config '{"runner":{"name":"activebckup"}}'
Connection 'team1' (0c0fe169-5c87-461d-9ff2-ab8aaac62c7c) successfully added.
```

2）team1 配置 IP

```
[root@localhost ~]# nmcli con mod team1 ipv4.addresses 192.168.0.100/24
[root@localhost ~]# nmcli con mod team1 ipv6.addresses fd00:ba5e:ba11:10::1/64
[root@localhost ~]# nmcli con mod team1 ipv4.method static
[root@localhost ~]# nmcli con mod team1 ipv6.method static
```

3）把两个物理接口加入 team1

```
[root@server10 ~]# nmcli con add type team-slave con-name team1-slave1 ifname ens33 master team1
[root@server10 ~]# nmcli con add type team-slave con-name team1-slave2 ifname ens37 master team1
```

2、在 ServerX 上 172.25.X.0/24 配置一 DNS 缓存服务器,DNS 查询转发到 172.25.254.254,在 example.com 域绕开 DNSSEC 验证。

```
[root@server10 ~]# yum -y install unbound
[root@server10 ~]# vim /etc/unbound/unbound.conf
        interface: 0.0.0.0
        interface: ::0

        access-control: 172.25.10.0/24 allow

        domain-insecure: "example.com"

        forward-zone:
```

```
            name: "."
            forward-addr: 172.25.254.254

[root@server10 ~]# systemctl start unbound
[root@server10 ~]# systemctl enable unbound

[root@server10 ~]# firewall-cmd --permanent --add-service=dns
[root@server10 ~]# firewall-cmd --reload
```

3、在 ServerX 上配置能从 172.25.X.0/24 子网通过 SMTP 接收邮件消息，不是发往 @serverX.example.com 和@localhost.localdoamin 的消息转发到 desktopX.exmaple.com 智能主机。

```
[root@server10 ~]# yum -y install postfix
[root@server10 ~]# postconf -e 'inet_interface = all'
[root@server10 ~]# postconf -e 'mynetwork = 172.25.10.0/24 127.0.0.0/8'
[root@server10 ~]# postconf -e 'mydestination =
server10.exmaple.com,localhost.local.domain,localhost'
[root@server10 ~]# postconf -e 'relayhost = desktop10.example.com'

[root@server10 ~]# systemctl start postfix
[root@server10 ~]# systemctl enable postfix
[root@server10 ~]# firewall-cmd --permanent --add-service=postfix
[root@server10 ~]# firewall-cmd --reload
```

4、在 serverX 的第二块硬盘上建立一个 512M 分区，用 ISCSI 共享名为 iqn.2014-11-11.com.example:serverX.zoidberg，只允许客户端名这 iqn.2014-11.com.example:desktopX 连接使用。

```
[root@server10 ~]# fdisk /dev/sdb
命令(输入 m 获取帮助)：n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
分区号 (1-4，默认 1)：
起始 扇区 (2048-41943039，默认为 2048)：
将使用默认值 2048
Last 扇区, +扇区 or +size{K,M,G} (2048-41943039，默认为 41943039)：+512M
分区 1 已设置为 Linux 类型，大小设为 512 MiB

命令(输入 m 获取帮助)：w

[root@server10 ~]# yum -y install targetcli
[root@server10 ~]# targetcli
```

```
/> /backstores/block create server10.zoidbreg /dev/sdb1
Created block storage object server10.zoidbreg using /dev/sdb1.
/> /iscsi create iqn.2014-11.com.example:server10.zoidbreg
Created target iqn.2014-11.com.example:server10.zoidbreg.
Created TPG 1.
/>       /iscsi/iqn.2014-11.com.example:server10.zoidbreg/tpg1/acls       create
iqn.2014-11.com.example:desktop10
Created Node ACL for iqn.2014-11.com.example:desktop10
/>       /iscsi/iqn.2014-11.com.example:server10.zoidbreg/tpg1/luns       create
/backstores/block/server10.zoidbreg
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2014-11.com.example:desktop10
/>     /iscsi/iqn.2014-11.com.example:server10.zoidbreg/tpg1/portals      create
172.25.10.1
Using default IP port 3260
Created network portal 172.25.10.1:3260.
/> exit

[root@server10 ~]# firewall-cmd --permanent --add-port=3260/tcp
[root@server10 ~]# firewall-cmd --reload
```

5、在 serverX 配置一个 NFSv4 共享/exports/hyphotad，用 krbs 验证，keytab 保存在 http://classroom.example.com/pub/kdytabs/serverX.keytab. 共享目录对客户端有读写权限，并用 kerberos 验证和加密，这个目录权限设置为 1777.

```
[root@server10 ~]# mkdir -m 1777 -p /exprots/hytpotad
[root@server10 ~]# vim /etc/exports
    /exprots/hytpotad   172.25.10.0/24(sec=krb5p,rw)

[root@server10 ~]# wget -o /etc/krb5.kdytab http://classroom.example.com/pub/kdytabs/serverX.keytab

[root@server10 ~]# systemctl start nfs-secure-server
[root@server10 ~]# systemctl enable nfs-secure-server
[root@server10 ~]# firewall-cmd --permanent --add-service=nfs
[root@server10 ~]# firewall-cmd --permanent --add-service=rpcbind
[root@server10 ~]# firewall-cmd --permanent --add-service=mounted
[root@server10 ~]# firewall-cmd --reload
```

6、在 serverX 上用 smb 共享一个新目录/exports/bigbang，这个目录的所属组为 bigbang，要求这个目录中建立的新文件能自动设置属组为 bigbang，bigbang 组中用户在外能访问这个共享。

Samba 唯一的用户 penny 属组为 bigbnag，Samba 用户 penny 的密码为 1234，能读写共享目录。

```
[root@server10 ~]# groupadd bigbang
```

```
[root@server10 ~]# useradd -s /sbin/nologin -G bigbang penny

[root@server10 ~]# mkdir -m 2777 -p /exprots/bigbang
[root@server10 ~]# chgrp bigbang /exprots/bigbang

[root@server10 ~]# semanage fcontext -a -t samba_share-t  '/exports/bigbang(/.*)?'
[root@server10 ~]# restorecon -RFv  /exprots/bigbang

[root@server10 ~]# yum -y install samba samba-client

[root@server10 ~]# vim /etc/samba/smb.conf
    [bigbang]
       path = /exprots/bigbang
       valid users = @bigbang
       write list =  @bigbang

[root@server10 ~]# smbpasswd -a penny
   New SMB password:
   Retype new SMB password:

[root@server10 ~]# systemctl start smb
[root@server10 ~]# systemctl enable smb
[root@server10 ~]# firewall-cmd --permanent --add-service=samba
[root@server10 ~]# firewall-cmd --reload
```

7、在 serverX 在 444/tcp 端口发布两个受 TLS 保护的两个域名 wwwX.example.com，webappX.example.com。两个网站文档分别要求存储在/srv/wwwX/www 和/srv/webappX/www.

1) 用 https://wwwX.example.com 能响应

This is wwwX

用 https://webappX.example.com 访问能响应：

执行/home/student/myapp.php

2)你需要的证书和钥密文件分别存储在以下位置：

http://classroom/example.com/pub/example-ca.crt

http://classroom/example.com/pub/tls/certs/wwwX.crt

http://classroom/example.com/pub/tls/certs/webappX.crt

http://classroom/example.com/pub/tls/private/wwwX.key

http://classroom/example.com/pub/tls/private/wwwX.key

3) PHP 应用需要数据库的支持，一个完整备份（mysqldump）存储 /home/student/mydb.mysqldump.

解答：

1）安装需要的包

```
[root@server10 ~]# yum insall httpd mod_ssl mariadb-server mariadb-client  php php-mysql
```

2）启动 mariadb,并恢复数据库

```
[root@server10 ~]# systemclt start mariadb
[root@server10 ~]# systemclt enable mariadb
[root@server10 ~]# mysql < /home/student/mydb.mysqldump
```

3）建立网站目录和文档

```
[root@server10 ~]# mkdir -p /srv/{www,webapp}X/www
[root@server10 ~]# echo “this is wwwX” > /srv/wwwX/www/index.html
[root@server10 ~]# cp /home/student/myaap.php /srv/webappX/www/index.php
[root@server10 ~]#  semanage fcontext -a -t httpd_sys_content_t  ‘/srv(/.*)?’
[root@server10 ~]#  restorecon  -RFv /srv
```

4）下载所需的证书

```
[root@server10 ~]# cd /etc/pki/tls/certs
[root@server10 ~]# wget http://classroom/example.com/pub/example-ca.crt
[root@server10 ~]#  wget http://classroom/example.com/pub/tls/certs/wwwX.crt
[root@server10 ~]#  wget http://classroom/example.com/pub/tls/certs/webappX.crt
[root@server10 ~]# cd /etc/pki/tls/privte
[root@server10 ~]#  wget http://classroom/example.com/pub/tls/private/wwwX.key
[root@server10 ~]#  wget http://classroom/example.com/pub/tls/private/webappX.key
[root@server10 ~]#  chmod 0400 w*key
```

5）配置 SElinux 允许 httpd 监听 444/tcp

```
[root@server10 ~]# semanage port -a -t http_port_t -p tcp 444
```

6）配置 httpd 监听 444/tcp

```
[root@server10 ~]# vim /etc/httpd/conf/httpd.conf
  listen  444  https
```

7）配置第一个虚拟主机 wwwX.exmaple.com

```
[root@server10 ~]# vim /etc/httpd/conf.d/wwwX.conf
   <VirtualHost *:444>
      DocumentRoot /srv/wwwX/www
      ServerName wwwX.example.com
      SSLEngine on
      SSLProtocol all -SSLv2
      SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
      SSLHonorCipherOrder on
```

```
    SSLCertificateFile       /etc/pki/tls/certs/wwwX.crt
    SSLCertificateKeyFile    /etc/pki/tls/private/wwwX.key
    SSLCertificateChainFile    /etc/pki/tls/certs/exmaple-ca.crt
</VirtualHost>

  <Directory "/srv/wwwX/www">
      Require all granted
  </Directory>
```

8）配置第二个虚拟主机 webappX.exmaple.com

```
[root@server10 ~]# vim /etc/httpd/conf.d/webappX.conf
  <VirtualHost *:444>
     DocumentRoot /srv/webappX/www
     ServerName webappX.example.com
     SSLEngine on
     SSLProtocol all -SSLv2
     SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
     SSLHonorCipherOrder on
    SSLCertificateFile       /etc/pki/tls/certs/webappX.crt
    SSLCertificateKeyFile    /etc/pki/tls/private/webappX.key
    SSLCertificateChainFile    /etc/pki/tls/certs/exmaple-ca.crt
</VirtualHost>

  <Directory "/srv/webappX/www">
      Require all granted
  </Directory>
```

9）启动 httpd 服务

```
[root@server10 ~]# systemctl start httpd
[root@server10 ~]# systemctl enable httpd
```

10）在防火墙上开放 444/tcp 端口

```
[root@server10 ~]# firewall-cmd  --permanent --add-port=444/tcp
[root@server10 ~]# firewall-cmd --reload
```

7、在 desktopX 上建立一脚本，/home/student/bin/myusers。

1)当运行脚本时带 userlist 参数,返回/etc/passwd 中所有登录 shell 不是/sbin/nologin 的用户，只显示用户名，并按字符排序。

2） 当运行脚本时带 userinfo 参数时，需要第二个用户名参数，正常返回第二个参数给出的用户登录的 shell 。如果第二个用户名不存在，则提示“invalid user”，如果第二个参数为空，则显示错误消息“Please specify a username”,并返回 132 代码。

3）如果脚本不带任何参数，则显示以下内容：

myusers userlist

myusers userinfo <USERNAME>

解答：

```
[root@desktop10 ~]# mkdir /home/student/bin
[root@server10 ~]#  vim /home/student/myusers
  #!/bin/bash
   if [ $# -eq 0 ] ;then
      echo  "$(basename $0) userlist"
      echo  "$(basename $0) userinfo <USERNAME>"
   fi

  case $1 in
     userlist)
       grep -v  ':/sbin/nologin' /etc/passwd | cut -d: -f1 | sort
     ;;
    userinfo)
     if  [ "$2" == "" ];then
        echo "Please specify a username"
        exit 132
     fi
    if ! getent passwd $2 &> /dev/null;then
       echo "invalid user"
       exit
    fi

     getent passwd $2 | cut -d: -f7
     ;;

*)
   exit
 ;;
esac
~
```

李　伟　阶

QQ：17533203

整理于 2014 年 11 月