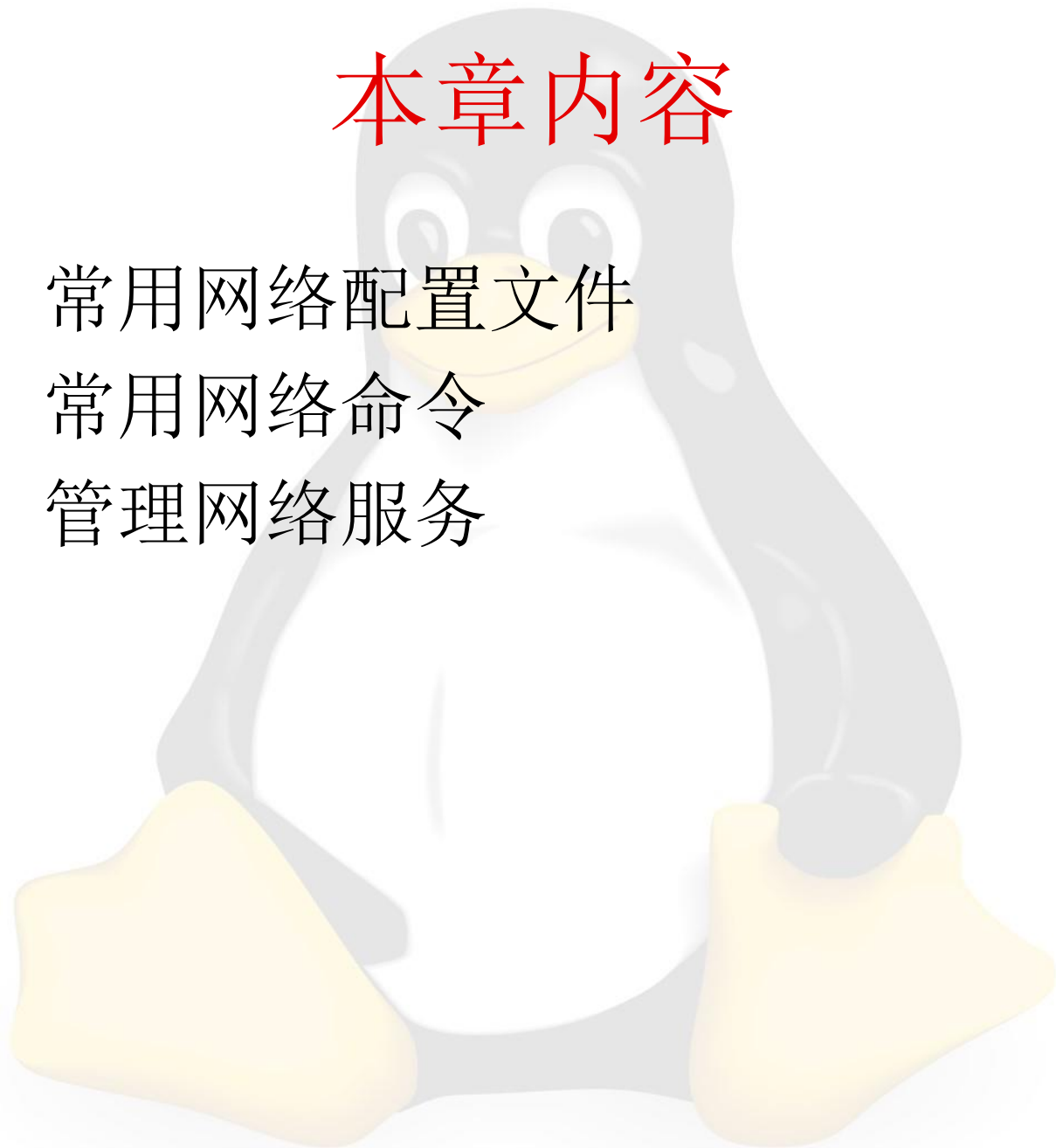


第12章 Linux网络基本配置



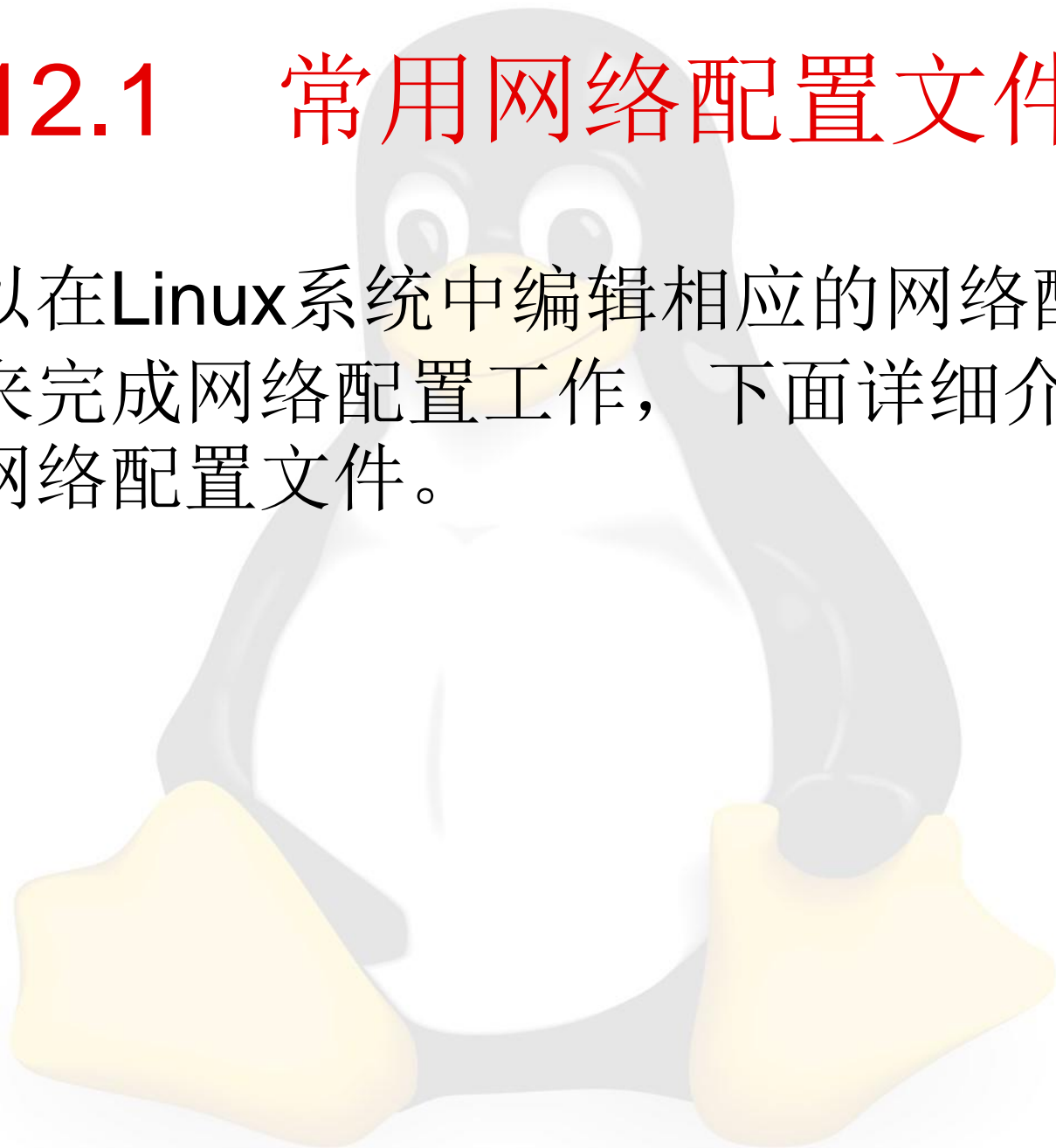
本章内容

- 12.1 常用网络配置文件
- 12.2 常用网络命令
- 12.3 管理网络服务



12.1 常用网络配置文件

- 可以在Linux系统中编辑相应的网络配置文件来完成网络配置工作，下面详细介绍这些网络配置文件。



/etc/sysconfig/network-scripts/ifcfg-eno16777736文件

- 在Linux统中，系统网络设备的配置文件保存在/etc/sysconfig/network-scripts目录下，其中文件ifcfg-eno16777736包含一块网卡的配置信息，文件ifcfg-lo包含回路IP地址信息。

/etc/resolv.conf文件

- /etc/resolv.conf文件是由域名解析器（resolver，一个根据主机名解析IP地址的库）使用的配置文件。



/etc/hosts文件

- 当计算机启动时，在可以查询**DNS**以前，计算机需要查询一些主机名到**IP**地址的匹配。这些匹配信息存放在**/etc/hosts**文件中。在没有域名服务器的情况下，系统上的所有网络程序都通过查询该文件来解析对应于某个主机名的**IP**地址。

/etc/services 文件

- /etc/services 文件定义了Linux系统中所有服务的名称、协议类型、服务的端口等信息。



12.2 常用网络命令

- 在Linux系统中提供了大量的网络命令用于网络配置、网络测试以及网络诊断，如 traceroute、ifconfig、ping、netstat、arp以及tcpdump等。

traceroute

- 使用**traceroute**命令可以显示数据包到目标主机之间的路径。**traceroute**命令使用户可以追踪网络数据包的路由途径，预设**IPv4**数据包大小是**60**字节，用户可以另外设置。

命令语法：

traceroute [选项] [主机名|IP地址] [数据包大小]

【例12.1】 跟踪从本地计算机到www.163.com网站的路径。

```
[root@rhel ~]# traceroute www.163.com
```

```
traceroute to www.163.com (220.181.28.50), 30 hops max, 60 byte packets
```


```
 1  192.168.0.1 (192.168.0.1)  1.036 ms  0.922 ms  0.829 ms
 2  58.41.132.1 (58.41.132.1)  2.054 ms  1.990 ms  2.195 ms
 3  222.72.255.153 (222.72.255.153)  1.698 ms  1.664 ms  1.658 ms
 4  218.1.4.29 (218.1.4.29)  1.933 ms  1.877 ms  1.971 ms
 5  218.1.0.198 (218.1.0.198)  2.672 ms  2.398 ms  2.503 ms
 6  202.101.63.194 (202.101.63.194)  2.472 ms  2.467 ms  2.400 ms
 7  (202.97.34.61)  23.957 ms  23.979 ms  23.876 ms
 8  (218.30.25.45)  24.117 ms  24.373 ms  24.134 ms
 9  (218.30.25.238)  24.764 ms  24.601 ms  24.557 ms
10  (220.181.16.138)  29.313 ms  37.606 ms  38.297 ms
11  (220.181.17.58)  27.850 ms  35.903 ms  36.844 ms
12  (220.181.28.50)  24.424 ms  20.385 ms  57.837 ms
```

ifconfig

- 使用ifconfig命令可以显示和配置网络接口，比如设置IP地址、MAC地址、激活或关闭网络接口。

命令语法：

ifconfig [接口] [选项| IP地址]



【例12.2】 配置网卡eno16777736的IP地址， 同时激活该设备。

```
[root@rhel ~]# ifconfig eno16777736 192.168.0.100 netmask 255.255.255.0 up
```

【例12.3】 配置网卡eno16777736别名设备eno16777736:1的IP地址。

```
[root@rhel ~]# ifconfig eno16777736:1 192.168.0.3
```

【例12.4】 激活网卡eno16777736:1设备。

```
[root@rhel ~]# ifconfig eno16777736:1 up
```

【例12.5】 查看网卡eno16777736网络接口的配置。

```
[root@rhel ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu
1500
    inet 192.168.0.100 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::20c:29ff:fe75:d61e prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:75:d6:1e txqueuelen 1000 (Ethernet)
    RX packets 5711 bytes 489075 (477.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2625 bytes 312104 (304.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0x2000
//查看该网卡IP地址是192.168.0.100,MAC地址是00:0c:29:75:d6:1e
```

【例12.6】 查看所有的网卡网络接口配置。

```
[root@rhel ~]# ifconfig
```

```
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu  
1500
```

```
inet 192.168.0.100 netmask 255.255.255.0 broadcast 192.168.0.255
```

```
inet6 fe80::20c:29ff:fe75:d61e prefixlen 64 scopeid 0x20<link>
```

```
ether 00:0c:29:75:d6:1e txqueuelen 1000 (Ethernet)
```

```
RX packets 5660 bytes 484973 (473.6 KiB)
```

```
RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 2596 bytes 307846 (300.6 KiB)
```

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
device interrupt 19 base 0x2000
```

```
eno16777736:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu  
1500
```

```
inet 192.168.0.3 netmask 255.255.255.0 broadcast 192.168.0.255
```

```
ether 00:0c:29:75:d6:1e txqueuelen 1000 (Ethernet)
```

```
device interrupt 19 base 0x2000
```



lo: flags=73<UP,LOOPBACK,RUNNING> mtu 16436

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 0 (Local Loopback)

RX packets 94 bytes 8664 (8.4 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 94 bytes 8664 (8.4 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255


ether 2a:e1:6e:43:1b:99 txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0



【例12.8】 禁用网卡eno16777736:1设备。
[root@rhel ~]# ifconfig eno16777736:1 down

【例12.9】 更改网卡eno16777736的硬件MAC地址为00:0C:29:18:2E:3D。
[root@rhel ~]# ifconfig eno16777736 hw ether 00:0C:29:18:2E:3D

ping

- 使用ping命令用来测试与目标计算机之间的连通性。执行ping命令会使用ICMP传输协议发出要求回应的信息，如果远程主机的网络功能没有问题，就会回应该信息，因而得知该主机是否运作正常。

命令语法：

ping [选项] [目标]



【例12.10】 测试与网站www.ak.com的连通性。

```
[root@rhel ~]# ping www.ak.com
```

```
PING www.ak.com(191.161.1.28) 56(84) bytes of data.
```

```
64 bytes from 191.161.1.28: icmp_req=0 ttl=64 time=1.49 ms
```

```
64 bytes from 191.161.1.28: icmp_req=1 ttl=64 time=0.873 ms
```

```
64 bytes from 191.161.1.28: icmp_req=2 ttl=64 time=1.00 ms
```

```
64 bytes from 191.161.1.28: icmp_req=3 ttl=64 time=0.440 ms
```

```
.....
```

```
//在Linux系统中用该命令会不间断地返回ICMP数据包，要停止测试按[Ctrl+c]键
```

【例12.11】 测试与192.168.0.222计算机的连通性，每次发送的ICMP数据包大小为128字节。

```
[root@rhel ~]# ping -s 128 192.168.0.222
PING 192.168.0.111 (192.168.0.222) 128(156) bytes of data.
136 bytes from 192.168.0.222: icmp_req=1 ttl=64 time=1.38 ms
136 bytes from 192.168.0.222: icmp_req=2 ttl=64 time=0.645 ms
136 bytes from 192.168.0.222: icmp_req=3 ttl=64 time=0.426 ms
136 bytes from 192.168.0.222: icmp_req=4 ttl=64 time=0.358 ms
136 bytes from 192.168.0.222: icmp_req=5 ttl=64 time=0.401 ms
```

.....

【例12.12】 测试与192.168.0.5计算机的连通性，要求返回4个ICMP数据包。

```
[root@rhel ~]# ping -c 4 192.168.0.5
PING 192.168.0.111 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_req=1 ttl=64 time=0.527 ms
64 bytes from 192.168.0.5: icmp_req=2 ttl=64 time=0.352 ms
64 bytes from 192.168.0.5: icmp_req=3 ttl=64 time=0.429 ms
64 bytes from 192.168.0.5: icmp_req=4 ttl=64 time=0.340 ms

--- 192.168.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.340/0.412/0.527/0.074 ms
```

netstat

- 使用**netstat**命令用来显示网络状态的信息，得知整个Linux系统的网络情况，比如网络连接、路由表、接口统计、伪装连接和组播成员。

命令语法：

netstat [选项] [延迟]

【例12.13】 显示内核路由表信息。

```
[root@rhel ~]# netstat -r
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irrt	Iface
default	192.168.0.1	0.0.0.0	UG	0 0	0	eno16777736	
192.168.0.0	*	255.255.255.0	U	0 0	0	eno16777736	
192.168.122.0	*	255.255.255.0	U	0 0	0	virbr0	

【例12.14】 显示端口号为22的连接情况。

```
[root@rhel ~]# netstat -antu |grep 22
```

```
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
```

【例12.15】 显示UDP传输协议的连接状态。

```
[root@rhel ~]# netstat -u
```

Active Internet connections (w/o servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address
udp	0	0	192.168.0.222:filenet-pa	192.168.0.100:domain
	ESTABLISHED			
udp	0	0	192.168.0.222:filenet-cm	192.168.0.100:domain
	ESTABLISHED			
udp	0	0	192.168.0.222:filenet-re	192.168.0.100:domain
	ESTABLISHED			
udp	0	0	192.168.0.222:32805	192.168.0.100:domain
	ESTABLISHED			



arp

- 使用**arp**命令可以用来增加、删除和显示**ARP**缓存条目。

命令语法：

arp [选项] [IP地址][MAC地址]

【例12.16】 查看系统arp缓存。

```
[root@rhel ~]# arp
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.5	ether	00:50:56:c0:00:01	C		

【例12.17】 添加一个IP地址和MAC地址的对应记录。

```
[root@rhel ~]# arp -s 192.168.0.99 00:60:08:27:CE:B2
```

```
[root@rhel ~]# arp
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.5	ether	00:50:56:c0:00:01	C		eno16777736
192.168.0.99	ether	00:60:08:27:ce:b2	CM		

//可以看到刚才添加的静态ARP记录

【例12.18】 删除一个IP地址和MAC地址的对应缓存记录。

```
[root@rhel ~]# arp -d 192.168.0.99
```

```
[root@rhel ~]# arp
```

Address	Hwtype	HWaddress	Flags	Mask	Iface
192.168.0.5	ether	00:50:56:C0:00:01	C		eno16777736

tcpdump

- 是Linux系统中强大的网络数据采集分析工具之一，可以将网络中传送的数据包的头完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来筛选信息。作为互联网上经典的系统管理员必备工具，tcpdump以其强大的功能，灵活的截取策略，成为每个高级的系统管理员分析网络，排查问题等所必备的工具之一。

命令语法：

tcpdump [选项] [表达式]

【例12.19】 使用指定的网络接口eno16777736读取数据链路层的数据包头。

```
[root@rhel ~]# tcpdump -i eno16777736
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eno16777736, link-type EN10MB (Ethernet), capture size 65535
bytes
10:40:01.324515 IP rhel.ssh > 192.168.0.5.49585: Flags [P.], seq
    1401759532                                :1401759648, ack
    1000464709, win 190, length 116
10:40:01.325282 IP rhel.ssh > 192.168.0.5.49585: Flags [P.], seq 116:168, a
    ck 1, win 190, length 52
10:40:01.325621 IP 192.168.0.5.49585 > rhel.ssh: Flags [.], ack 168, win 16
    370, length 0
10:40:01.325856 IP rhel.ssh > 192.168.0.5.49585: Flags [P.], seq 168:284, a
    ck 1, win 190, length 116
10:40:01.326319 IP rhel.ssh > 192.168.0.5.49585: Flags [P.], seq 284:336, a
    ck 1, win 190, length 52
```

.....

12.3 管理网络服务

- RHEL 7系统使用**systemd**，它提供更优秀的框架以表示系统服务间的依赖关系，并实现系统初始化时服务的并行启动，同时达到降低**Shell**的系统开销的效果，最终代替现在常用的**System V**。

systemd简介

- 在RHEL7之前，服务管理工作是由System V通过 `/etc/rc.d/init.d` 目录下的Shell脚本来执行的，通过这些脚本允许管理员控制服务的状态。在RHEL 7中，这些脚本被服务单元文件替换。在systemd中，服务、设备、挂载等资源统一被称为单元，所以systemd中有许多单元类型，服务单元文件的扩展名是.service，同Shell脚本的功能相似。比如有查看、启动、停止、重启、启用或者禁止服务的参数。
- 一个单元的配置文件可以描述系统服务（.service）、挂载点（.mount）、sockets（.sockets）、系统设备（.device）、交换分区（.swap）、文件路径（.path）、启动目标（.target）、由systemd管理的计时器（.timer）等。

systemd单元文件放置位置

- `/usr/lib/systemd/system`: systemd默认单元文件安装目录;
- `/etc/systemd/system`: 系统管理员创建和管理的单元目录, 优先级最高。

systemctl命令语法

- 使用 **systemctl** 控制单元时，通常需要使用单元文件的全名，包括扩展名（比如 **sshd.service**）。如果没有指定扩展名，**systemctl** 默认把扩展名当作 **.service**。

命令语法：

systemctl [选项] [单元命令|单元文件命令]

【例12.20】 启动named服务。

```
[root@rhel ~]# systemctl start named.service
```



【例12.21】查看named服务当前状态。

```
[root@rhel ~]# systemctl status named.service
```

```
named.service - Berkeley Internet Name Domain (DNS)
```

```
Loaded: loaded (/usr/lib/systemd/system/named.service; disabled)
```

```
Active: active (running) since Thu, 30 May 2013 09:31:38 +0800; 1min 1s ago
```

```
Process: 3073 ExecStop=/bin/sh -c /usr/sbin/rndc stop > /dev/null 2>&1  
|| /bin/kill -TERM $MAINPID (code=exited, status=0/SUCCESS)
```

```
Process: 3083 ExecStart=/usr/sbin/named -u named $OPTIONS  
(code=exited, status=0/SUCCESS)
```

```
Process: 3081 ExecStartPre=/usr/sbin/named-checkconf -z  
/etc/named.conf (code=exited,  
status=0/SUCCESS)
```

```
Main PID: 3084 (named)
```

```
CGroup: name=systemd:/system/named.service
```

```
└─ 3084 /usr/sbin/named -u named
```




【例12.22】停止named服务。

```
[root@rhel ~]# systemctl stop named.service
```

【例12.23】重新启动named服务。

```
[root@rhel ~]# systemctl restart named.service
```

【例12.24】重新加载named服务配置文件。

```
[root@rhel isolinux]# systemctl reload named.service
```

【例12.25】设置named服务开机自动启动。

```
[root@rhel ~]# systemctl enable named.service
```

```
ln -s '/usr/lib/systemd/system/named.service' '/etc/systemd/system/multi-user.target.wants/named.service'
```

【例12.26】查询named服务是否开机自动启动。

```
[root@rhel isolinux]# systemctl is-enabled named.service  
enabled
```

【例12.27】停止named服务开机自动启动。

```
[root@rhel ~]# systemctl disable named.service
```

```
[root@rhel ~]# systemctl is-enabled named.service  
disabled
```

【例12.28】查看所有已启动的服务。

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

UNIT	LOAD	ACTIVE	SUB	JOB DESCRIPTION
abrt-ccpp.service	loaded	active	exited	Install ABRT coredump hook
abrt-oops.service	loaded	active	running	ABRT kernel log watcher
abrt-vmcore.service	loaded	active	exited	Harvest vmcores for ABRT
abrt.service	loaded	active	running	ABRT Automated Bug Reportin
accounts-daemon.service	loaded	active	running	Accounts Service
acpid.service	loaded	active	running	ACPI Event Daemon
atd.service	loaded	active	running	Job spooling tools
auditd.service	loaded	active	running	Security Auditing Service
avahi-daemon.service	loaded	active	running	Avahi mDNS/DNS-SD Stack
bluetooth.service	loaded	active	running	Bluetooth Manager
colord-sane.service	loaded	active	running	Daemon for monitoring attac
.....				(省略)