服务控制

Systemd 的简介和特点

Systemd 是 Linux 系统中最新的初始化系统（init），它主要的设计目标是克服 sysvinit 固有的缺点，提高系统的启动速度。systemd和ubuntu的upstart是竞争对手，从15.04版本开始，Ubuntu也已经采systemd作为其标准的系统初始化系统。

Systemd 的基本概念

单元（unit）的概念

系统初始化需要做的事情非常多。需要启动后台服务，比如启动 SSHD 服务；需要做配置工作，比如挂载文件系统。这个过程中的每一步都被 systemd 抽象为一个配置单元，即 unit。可以认为一个服务是一个配置单元；一个挂载点是一个配置单元；一个交换分区的配置是一个配置单元；等等

- Service unit：系统服务，最常见的类型
- Target unit：多个 Unit 构成的一个组，执行环境类型
- Device Unit：硬件设备
- Mount Unit：文件系统的挂载点
- Automount Unit：自动挂载点
- Path Unit：侦测特定文件或目录类型的
- Scope Unit：不是由 Systemd 启动的外部进程
- Slice Unit：进程组
- Snapshot Unit：Systemd 快照，可以切回某个快照
- Socket Unit：进程间通信的 socket服务
- Swap Unit：swap 文件
- Timer Unit：循环执行的服务

Target 和运行级别

systemd 用目标（target）替代了运行级别的概念，提供了更大的灵活性，如您可以继承一个已有的目标，并添加其它服务，来创建自己的目标。下表列举了 systemd 下的目标和常见 runlevel 的对应关系：

Sysvinit 运行级别	Systemd 目标	备注
0	runlevel0.target, poweroff.target	关闭系统。
1, s, single	runlevel1.target, rescue.target	单用户模式。
2, 4	runlevel2.target, runlevel4.target, multi-user.target	用户定义/域特定运行级别。默认等同于 3。
3	runlevel3.target, multi-user.target	多用户，非图形化。用户可以通过多个控制台或网络登录。
5	runlevel5.target, graphical.target	多用户，图形化。通常为所有运行级别 3 的服务外加图形化登录。
6	runlevel6.target, reboot.target	重启
emergency	emergency.target	紧急 Shell

RHEL7监视和控制systemd的主要命令是systemctl，该命令可以用于查看系统状态和管理系统及服务。

systemctl是 Systemd 的主命令，用于管理系统。

```
# 重启系统
$ sudo systemctl reboot
# 关闭系统，切断电源
$ sudo systemctl poweroff
# CPU停止工作
$ sudo systemctl halt
# 暂停系统
$ sudo systemctl suspend
# 让系统进入冬眠状态
$ sudo systemctl hibernate
# 让系统进入交互式休眠状态
$ sudo systemctl hybrid-sleep
# 启动进入救援状态（单用户状态）
$ sudo systemctl rescue
```

systemd-analyze命令用于查看启动耗时。

```
# 查看启动耗时
$ systemd-analyze
```

```
# 查看每个服务的启动耗时
$ systemd-analyze blame
# 显示瀑布状的启动过程流
$ systemd-analyze critical-chain
# 显示指定服务的启动流
$ systemd-analyze critical-chain atd.service
```

systemctl list-units命令可以查看当前系统的所有 Unit 。

```
# 列出正在运行的 Unit
$ systemctl list-units
# 列出所有Unit，包括没有找到配置文件的或者启动失败的
$ systemctl list-units --all
# 列出所有没有运行的 Unit
$ systemctl list-units --all --state=inactive
# 列出所有加载失败的 Unit
$ systemctl list-units --failed
# 列出所有正在运行的、类型为 service 的 Unit
$ systemctl list-units --type=service
```

systemctl最常用的命令如下：

```
systemctl start <单元>      #立即启动单元
systemctl stop <单元>       #立即停止单元
systemctl restart <单元>    #重启单元
systemctl reload <单元>     #重新读取单元配置
systemctl status <单元>     #输出单元运行状态
systemctl is-enabled <单元>   #检查单元是否配置为自动启动
systemctl enable <单元>     #开机自动启动单元
systemctl disable <单元>     #取消开机自动激活单元
systemctl is-active <单元>    #查看单元是不是正在运行
systemctl is-failed <单元>      #查看单元是否处于启动失败状态
systemctl kill <单元>          #杀死单元的所有子进程
systemctl daemon-reload     #重新载入 systemd，扫描新的或有变动的单元
```

RHEL7的服务systemctl脚本存放在：/usr/lib/systemd/目录，有系统（system）和用户（user）之分，像需要开机不登陆就能运行的程序，存在系统服务里，即：/usr/lib/systemd/system目录下。每一个服务以.service结尾，一般会分为3部分：[Unit]、[Service]和[Install]，每个部分内部是一些等号连接的键值对，注意，键值对的等号两侧不能有空格。

```
[Unit]
Description=OpenSSH server daemon
After=syslog.target network.target auditd.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStartPre=/usr/sbin/sshd-keygen
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

[Unit]区块通常是配置文件的第一个区块，用来定义 Unit 的元数据，以及配置与其他 Unit 的关系。它的主要字段如下。

- Description：简短描述
- Documentation：文档地址
- Requires：当前 Unit 依赖的其他 Unit，如果它们没有运行，当前 Unit 会启动失败
- Wants：与当前 Unit 配合的其他 Unit，如果它们没有运行，当前 Unit 不会启动失败
- BindsTo：与Requires类似，它指定的 Unit 如果退出，会导致当前 Unit 停止运行
- Before：如果该字段指定的 Unit 也要启动，那么必须在当前 Unit 之后启动
- After：如果该字段指定的 Unit 也要启动，那么必须在当前 Unit 之前启动
- Conflicts：这里指定的 Unit 不能与当前 Unit 同时运行
- Condition...：当前 Unit 运行必须满足的条件，否则不会运行
- Assert...：当前 Unit 运行必须满足的条件，否则会报启动失败

[Service]部分是服务的关键，是服务的一些具体运行参数的设置，只有 Service 类型的 Unit 才有这个区块。它的主要字段如下。

- Type：定义启动时的进程行为。它有以下几种值。
    - Type=simple：默认值，执行ExecStart指定的命令，启动主进程
    - Type=forking：以 fork 方式从父进程创建子进程，创建后父进程会立即退出
    - Type=oneshot：一次性进程，Systemd 会等当前服务退出，再继续往下执行
    - Type=dbus：当前服务通过D-Bus启动

    - Type=notify：当前服务启动完毕，会通知Systemd，再继续往下执行
    - Type=idle：若有其他任务执行完毕，当前服务才会运行
- ExecStart：启动当前服务的命令
- ExecStartPre：启动当前服务之前执行的命令
- ExecStartPost：启动当前服务之后执行的命令
- ExecReload：重启当前服务时执行的命令
- ExecStop：停止当前服务时执行的命令
- ExecStopPost：停止当前服务之后执行的命令
- RestartSec：自动重启当前服务间隔的秒数
- Restart：定义何种情况 Systemd 会自动重启当前服务，可能的值包括 always（总是重启）、on-success、on-failure、on-abnormal、on-abort、on-watchdog
- TimeoutSec：定义 Systemd 停止当前服务之前等待的秒数
- Environment：指定环境变量

[Install]通常是配置文件的最后一个区块，用来定义如何启动，以及是否开机启动。它的主要字段如下。

- WantedBy：它的值是一个或多个 Target，当前 Unit 激活时（enable）符号链接会放入/etc/systemd/system目录下面以 Target 名 + .wants后缀构成的子目录中
- RequiredBy：它的值是一个或多个 Target，当前 Unit 激活时，符号链接会放入/etc/systemd/system目录下面以 Target 名 + .required后缀构成的子目录中
- Alias：当前 Unit 可用于启动的别名
- Also：当前 Unit 激活（enable）时，会被同时激活的其他 Unit

注意: 如果服务没有Install段落，一般意味着应该通过其它服务自动调用它们

systemd提供更优秀的框架以表示系统服务间的依赖关系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，systemd的目标是尽可能启动更少进程；尽可能将更多进程并行启动。

所有可用的单元文件存放在 /usr/lib/systemd/system/ 和 /etc/systemd/system/ 目录（后者优先级更高）

使用单元：

一个单元可以是系统服务（.service）、挂载点（.mount）、sockets（.sockets）。

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（例如 sshd.service）。但是有些单元可以在systemctl中使用简写方式。如果无扩展名，

systemctl 默认把扩展名当作 .service。例如sshd和sshd.service 是等价的。
挂载点会自动转化为相应的 .mount 单元。例如 /home 等价于 home.mount。
设备会自动转化为相应的 .device 单元，所以 /dev/sda2 等价于 dev-sda2.device。
例如：
systemctl is-enabled iptables.service
systemctl is-enabled servicename.service #查询服务是否开机启动
systemctl enable *.service #开机运行服务
systemctl disable *.service #取消开机运行
systemctl start *.service #启动服务
systemctl stop *.service #停止服务
systemctl restart *.service #重启服务
systemctl reload *.service #重新加载服务配置文件
systemctl status *.service #查询服务运行状态
systemctl mask *.service #禁用指定服务
systemctl unmask *.service #激活指用服务
注：*代表某个服务的名字，如http的服务名为httpd
案例：
查看服务的状况

```
[root@srv1 ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Wed 2015-04-29 22:16:04 CST; 1h 45min ago
 Main PID: 1853 (sshd)
   CGroup: /system.slice/sshd.service
           ¥1853 /usr/sbin/sshd -D
```

停止 sshd服务，查看状态

```
[root@srv1 ~]# systemctl stop sshd.service
[root@srv1 ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: inactive (dead) since Thu 2015-04-30 00:06:29 CST; 8s ago
 Main PID: 1853 (code=exited, status=0/SUCCESS)
```

状态表示为dead服务已经停止。
disable 服务，禁止开机自动运行

```
[root@srv1 ~]# systemctl disable sshd.service
rm '/etc/systemd/system/multi-user.target.wants/sshd.service'
[root@srv1 ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: inactive (dead)
```

开机自动加载，并启动该服务

```
[root@srv1 ~]# systemctl enable sshd
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-user.target.wants/sshd.servi
ce'
[root@srv1 ~]# systemctl start sshd
```

通过mask来禁用该服务，这样一旦服务终止，则无法再启动，必须通过unmask解除禁用才能使用systemctl start sshd 来运行服务

```
[root@srv1 ~]# systemctl mask sshd
ln -s '/dev/null' '/etc/systemd/system/sshd.service'
[root@srv1 ~]# systemctl status sshd
sshd.service
   Loaded: masked (/dev/null)
   Active: active (running) since Thu 2015-04-30 00:10:46 CST; 1min 25s ago
 Main PID: 5786 (sshd)
   CGroup: /system.slice/sshd.service
           ¥ì5786 /usr/sbin/sshd -D
```

除了上面提到的基本功能，systemctl还可以查询当前加载的模块单元，注意后缀为service的才是我们需要管理的服务

```
[root@srv1 ~]# systemctl list-unit-files  ¦ grep sshd
anaconda-sshd.service                     static
sshd-keygen.service                       static
sshd.service                              enabled
sshd@.service                             static
sshd.socket                               disabled
```

可以通过 type来过滤掉其他类型的单元

systemctl list-unit-files –type service

Target

启动计算机的时候，需要启动大量的 Unit。如果每一次启动，都要一一写明本次启动需要哪些 Unit，显然非常不方便。Systemd 的解决方案就是 Target。

简单说，Target 就是一个 Unit 组，包含许多相关的 Unit 。启动某个 Target 的时候，Systemd 就会启动里面所有的 Unit。从这个意义上说，Target 这个概念类似于"状态点"，启动某个 Target 就好比启动到某种状态。

传统的init启动模式里面，有 RunLevel 的概念，跟 Target 的作用很类似。不同的是，RunLevel 是互斥的，不可能多个RunLevel同时启动，但是多个Target可以同时启动。

查看当前系统的所有 Target

$ systemctl list-unit-files --type=target

查看一个 Target 包含的所有 Unit

$ systemctl list-dependencies multi-user.target

```
# 查看启动时的默认 Target
$ systemctl get-default
# 设置启动时的默认 Target
$ sudo systemctl set-default multi-user.target
# 切换 Target 时，默认不关闭前一个 Target 启动的进程，
# systemctl isolate 命令改变这种行为，
# 关闭前一个 Target 里面所有不属于后一个 Target 的进程
$ sudo systemctl isolate multi-user.target
```

它与init进程的主要差别如下。

（1）默认的 RunLevel（在/etc/inittab文件设置）现在被默认的 Target 取代，位置是/etc/systemd/system/default.target，通常符号链接到graphical.target（图形界面）或者multi-user.target（多用户命令行）。

（2）启动脚本的位置，以前是/etc/init.d目录，符号链接到不同的 RunLevel 目录 （比如/etc/rc3.d、/etc/rc5.d等），现在则存放在/usr/lib/systemd/system和/etc/systemd/system目录。

（3）配置文件的位置，以前init进程的配置文件是/etc/inittab，各种服务的配置文件存放在/etc/sysconfig目录。现在的配置文件主要存放在/usr/lib/systemd目录，在/etc/systemd目录里面的修改可以覆盖原始设置。

日志管理

Systemd 统一管理所有 Unit 的启动日志。带来的好处就是，可以只用journalctl一个命令，查看所有日志（内核日志和应用日志）。日志的配置文件是/etc/systemd/journald.conf。

journalctl功能强大，用法非常多。

```
# 查看所有日志（默认情况下 ，只保存本次启动的日志）
$ sudo journalctl
# 查看内核日志（不显示应用日志）
$ sudo journalctl -k
# 查看系统本次启动的日志
$ sudo journalctl -b
$ sudo journalctl -b -0
# 查看上一次启动的日志（需更改设置）
$ sudo journalctl -b -1
# 查看指定时间的日志
$ sudo journalctl --since="2012-10-30 18:17:16"
```

```
$ sudo journalctl --since "20 min ago"
$ sudo journalctl --since yesterday
$ sudo journalctl --since "2015-01-10" --until "2015-01-11 03:00"
$ sudo journalctl --since 09:00 --until "1 hour ago"
# 显示尾部的最新10行日志
$ sudo journalctl -n
# 显示尾部指定行数的日志
$ sudo journalctl -n 20
# 实时滚动显示最新日志
$ sudo journalctl -f
# 查看指定服务的日志
$ sudo journalctl /usr/lib/systemd/systemd
# 查看指定进程的日志
$ sudo journalctl _PID=1
# 查看某个路径的脚本的日志
$ sudo journalctl /usr/bin/bash
# 查看指定用户的日志
$ sudo journalctl _UID=33 --since today
# 查看某个 Unit 的日志
$ sudo journalctl -u nginx.service
$ sudo journalctl -u nginx.service --since today
# 实时滚动显示某个 Unit 的最新日志
$ sudo journalctl -u nginx.service -f
# 合并显示多个 Unit 的日志
$ journalctl -u nginx.service -u php-fpm.service --since today
# 查看指定优先级（及其以上级别）的日志，共有8级
# 0: emerg
# 1: alert
# 2: crit
# 3: err
# 4: warning
# 5: notice
# 6: info
# 7: debug
$ sudo journalctl -p err -b
```

```
# 日志默认分页输出，--no-pager 改为正常的标准输出
$ sudo journalctl --no-pager
# 以 JSON 格式（单行）输出
$ sudo journalctl -b -u nginx.service -o json
# 以 JSON 格式（多行）输出，可读性更好
$ sudo journalctl -b -u nginx.service
 -o json-pretty
# 显示日志占据的硬盘空间
$ sudo journalctl --disk-usage
# 指定日志文件占据的最大空间
$ sudo journalctl --vacuum-size=1G
# 指定日志文件保存多久
$ sudo journalctl --vacuum-time=1years
```