Systemd 入门教程：命令篇

Systemd 是 Linux 系统工具，用来启动守护进程，已成为大多数发行版的标准配置。

# Systemd由来

历史上，Linux 的启动一直采用init进程。下面的命令用来启动服务:

$ sudo /etc/init.d/apache2 start

# 或者

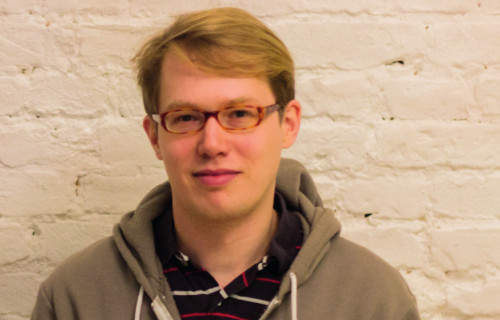
$ service apache2 start

这种方法有两个缺点：

* 启动时间长。init进程是串行启动，只有前一个进程启动完，才会启动下一个进程。
* 启动脚本复杂。init进程只是执行启动脚本，不管其他事情。脚本需要自己处理各种情况，这往往使得脚本变得很长。

# Systemd 概述

Systemd 就是为了解决这些问题而诞生的。它的设计目标是，为系统的启动和管理提供一套完整的解决方案。根据 Linux 惯例，字母d是守护进程（daemon）的缩写。 Systemd 这个名字的含义，就是它要守护整个系统。

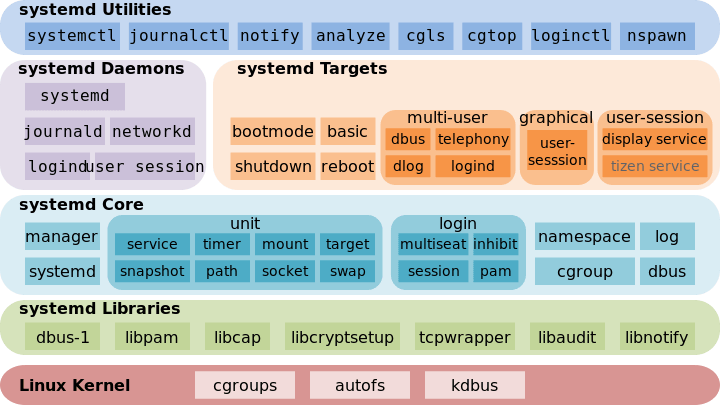


（上图为 Systemd 作者 Lennart Poettering）

使用了 Systemd，就不需要再用init了。Systemd 取代了initd，成为系统的第一个进程（PID 等于 1），其他进程都是它的子进程。

$ systemctl --version

上面的命令查看 Systemd 的版本。Systemd 的优点是功能强大，使用方便，缺点是体系庞大，非常复杂。事实上，现在还有很多人反对使用 Systemd，理由就是它过于复杂，与操作系统的其他部分强耦合，违反”keep simple, keep stupid”的Unix 哲学。



（上图为 Systemd 架构图）

# 系统管理

Systemd 并不是一个命令，而是一组命令，涉及到系统管理的方方面面。

## systemctl

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

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 |

## hostnamectl

hostnamectl命令用于查看当前主机的信息。

|  |
| --- |
| # 显示当前主机的信息  $ hostnamectl  # 设置主机名。  $ sudo hostnamectl set-hostname rhel7 |

## localectl

localectl命令用于查看本地化设置。

|  |
| --- |
| # 查看本地化设置  $ localectl  # 设置本地化参数。  $ sudo localectl set-locale LANG=en\_GB.utf8  $ sudo localectl set-keymap en\_GB |

**3.5 timedatectl**

timedatectl命令用于查看当前时区设置。

|  |
| --- |
| # 查看当前时区设置  $ timedatectl  # 显示所有可用的时区  $ timedatectl list-timezones  # 设置当前时区  $ sudo timedatectl set-timezone America/New\_York  $ sudo timedatectl set-time YYYY-MM-DD  $ sudo timedatectl set-time HH:MM:SS |

## loginctl

loginctl命令用于查看当前登录的用户。

|  |
| --- |
| # 列出当前session  $ loginctl list-sessions  # 列出当前登录用户  $ loginctl list-users  # 列出显示指定用户的信息  $ loginctl show-user ruanyf |

# Unit

## Unit含义

Systemd 可以管理所有系统资源。不同的资源统称为 Unit（单位）。

Unit 一共分成12种：

* Service unit：系统服务
* Target unit：多个 Unit 构成的一个组
* 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：文件或路径
* 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
* Swap Unit：swap 文件
* Timer Unit：定时器

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 |

**4.2 Unit 的状态**

systemctl status命令用于查看系统状态和单个 Unit 的状态。

|  |
| --- |
| # 显示系统状态  $ systemctl status  # 显示单个 Unit 的状态  $ sysystemctl status bluetooth.service  # 显示远程主机的某个 Unit 的状态  $ systemctl -H root@rhel7.example.com status httpd.service |

除了status命令，systemctl还提供了三个查询状态的简单方法，主要供脚本内部的判断语句使用：

|  |
| --- |
| # 显示某个 Unit 是否正在运行  $ systemctl is-active application.service  # 显示某个 Unit 是否处于启动失败状态  $ systemctl is-failed application.service  # 显示某个 Unit 服务是否建立了启动链接  $ systemctl is-enabled application.service |

## Unit 管理

对于用户来说，最常用的是下面这些命令，用于启动和停止 Unit（主要是 service）。

|  |
| --- |
| # 立即启动一个服务  $ sudo systemctl start apache.service  # 立即停止一个服务  $ sudo systemctl stop apache.service  # 重启一个服务  $ sudo systemctl restart apache.service  # 杀死一个服务的所有子进程  $ sudo systemctl kill apache.service  # 重新加载一个服务的配置文件  $ sudo systemctl reload apache.service  # 重载所有修改过的配置文件  $ sudo systemctl daemon-reload  # 显示某个 Unit 的所有底层参数  $ systemctl show httpd.service  # 显示某个 Unit 的指定属性的值  $ systemctl show -p CPUShares httpd.service  # 设置某个 Unit 的指定属性  $ sudo systemctl set-property httpd.service CPUShares=500 |

## 依赖关系

Unit 之间存在依赖关系：A 依赖于 B，就意味着 Systemd 在启动 A 的时候，同时会去启动 B。

systemctl list-dependencies命令列出一个 Unit 的所有依赖。

$ systemctl list-dependencies nginx.service

上面命令的输出结果之中，有些依赖是 Target 类型（详见下文），默认不会展开显示。如果要展开 Target，就需要使用--all参数。

$ systemctl list-dependencies --all nginx.service

# Unit 的配置文件

## 概述

每一个 Unit 都有一个配置文件，告诉 Systemd 怎么启动这个 Unit 。

Systemd 默认从目录/etc/systemd/system/读取配置文件。但是，里面存放的大部分文件都是符号链接，指向目录/usr/lib/systemd/system/，真正的配置文件存放在那个目录。

systemctl enable命令用于在上面两个目录之间，建立符号链接关系。

$ sudo systemctl enable clamd@scan.service

# 等同于

$ sudo ln -s '/usr/lib/systemd/system/clamd@scan.service' '

/etc/systemd/system/multi-user.target.wants/clamd@scan.service'

如果配置文件里面设置了开机启动，systemctl enable命令相当于激活开机启动。与之对应的，systemctl disable命令用于在两个目录之间，撤销符号链接关系，相当于撤销开机启动。

$ sudo systemctl disable clamd@scan.service

配置文件的后缀名，就是该 Unit 的种类，比如sshd.socket。如果省略，Systemd 默认后缀名为.service，所以sshd会被理解成sshd.service。

## 配置文件的状态

systemctl list-unit-files命令用于列出所有配置文件。

|  |
| --- |
| # 列出所有配置文件  $ systemctl list-unit-files  # 列出指定类型的配置文件  $ systemctl list-unit-files --type=service |

这个命令会输出一个列表：

$ systemctl list-unit-files

UNIT FILE STATE

chronyd.service enabled

clamd@.service static

clamd@scan.service disabled

这个列表显示每个配置文件的状态，一共有四种。

* enabled：已建立启动链接
* disabled：没建立启动链接
* static：该配置文件没有[Install]部分（无法执行），只能作为其他配置文件的依赖
* masked：该配置文件被禁止建立启动链接

注意，从配置文件的状态无法看出，该 Unit 是否正在运行。这必须执行前面提到的systemctl status命令。

$ systemctl status bluetooth.service

一旦修改配置文件，就要让 SystemD 重新加载配置文件，然后重新启动，否则修改不会生效。

$ sudo systemctl daemon-reload

$ sudo systemctl restart httpd.service

## 配置文件的格式

配置文件就是普通的文本文件，可以用文本编辑器打开。

systemctl cat命令可以查看配置文件的内容。

$ systemctl cat atd.service

[Unit]

Description=ATD daemon

[Service]

Type=forking

ExecStart=/usr/bin/atd

[Install]

WantedBy=multi-user.target

从上面的输出可以看到，配置文件分成几个区块。每个区块的第一行，是用方括号表示的区别名，比如[Unit]。注意，配置文件的区块名和字段名，都是大小写敏感的。

每个区块内部是一些等号连接的键值对。

[Section]

Directive1=value

Directive2=value

. . .

注意，键值对的等号两侧不能有空格。

## 配置文件的区块

[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

[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

[Service]区块用来 Service 的配置，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：停止当其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

# 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 |

Target 与 传统 RunLevel 的对应关系如下。

Traditional runlevel New target name Symbolically linked to...

Runlevel 0 | runlevel0.target -> poweroff.target

Runlevel 1 | runlevel1.target -> rescue.target

Runlevel 2 | runlevel2.target -> multi-user.target

Runlevel 3 | runlevel3.target -> multi-user.target

Runlevel 4 | runlevel4.target -> multi-user.target

Runlevel 5 | runlevel5.target -> graphical.target

Runlevel 6 | runlevel6.target -> reboot.target

它与init进程的主要差别如下：

* **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）。
* **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/lib/systemd/system和/etc/systemd/system目录。
* **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/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.serviceqq  -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 |