systemctl 中文手册

译者：金步国

其他作品

本文译者十分愿意与他人分享劳动成果，如果你对我的其他翻译作品或者技术文章有兴趣，可以在如下位置查看现有的作品集：

- 金步国作品集 [http://www.jinbuguo.com/]

联系方式

由于译者水平有限，因此不能保证译文内容准确无误。如果你发现了译文中的错误(哪怕是错别字也好)，请来信指出，任何提高译文质量的建议我都将虚心接纳。

- Email(QQ)：70171448在QQ邮箱

名称

systemctl - 控制 systemd 系统与服务管理器

概览

systemctl [OPTIONS...] COMMAND [NAME...]

描述

systemctl 可以用于检查和控制"systemd"系统与服务管理器的状态。参见 systemd(1) 手册页。

选项(OPTIONS)

systemctl 能够识别下列选项：

-t, --type=
参数必须是一个逗号(,)分隔的单元类型列表，例如"service,socket"。
当列出单元时，若使用了此选项，将只列出符合给定类型的单元，否则将列出所有类型的单元。
但有一个特例：如果参数中包含"help"，那么将列出所有可用的单元类型，然后退出。

--state=
参数必须是一个逗号(,)分隔的单元状态列表(只有 LOAD, SUB, ACTIVE 三大类)。
当列出单元时，若使用了此选项，将只列出符合给定状态的单元。
例如，使用 --state=failed 将只列出处于失败(failed)状态的单元。
此外，作为特例，使用 --state=help 将会导致程序在列出所有可用值后退出。

-p, --property=
参数必须是一个逗号分隔的属性名称列表，例如"MainPID,LogLevel"。
当使用 show 命令显示单元(unit)/任务(job)/管理器(manager)的属性时，将仅显示在参数中列出的属性。
如果未指定此选项，将显示全部的属性。
如果多次指定此选项，则相当于将这些选项的参数用逗号连接起来，合并放到同一个选项中。

不带参数的"systemctl show"命令将会显示管理器(manager)自身的所有属性(参见 systemd-system.conf(5) 手册)。

不同类型的单元拥有不同的属性集，指定任意一个单元(即使并不存在)，都可以查看此类单元的所有属性。
类似的，即使指定了一个不存在的任务(job)，也能查看任务的所有属性。
每种单元可拥有的属性集分散在 systemd.unit(5) 手册页以及此种单元专属的手册页中。

-a, --all
当列出单元时，表示列出所有已加载的单元，而不管其状态如何(包括处于停止状态的单元)。
当显示单元(unit)/任务(job)/管理器(manager)的属性时，表示显示所有属性，而不管它们是否已被设置。
如果想要列出所有已安装的单元，请使用 list-unit-files 命令。

-r, --recursive
当列出单元时，同时也显示本地容器的单元(在这些单元名称前加上容器名称前缀，并以":"作为分隔符)。

--reverse
在使用 list-dependencies 命令时，显示单元之间的反向依赖关系。
也就是显示 WantedBy=, RequiredBy=, PartOf=, BoundBy= 系列的依赖(而不是 Wants= 系列)。

--after
在使用 list-dependencies 命令时，显示在时间顺序上早于指定单元的那些单元，也就是递归的列出 After= 中的单元。

注意，每个 After= 依赖都会自动生成一个对应的 Before= 依赖。单元间在时间上的先后顺序既可以被显式的明确设定，
也可以由其他指令隐式的自动生成(例如 WantedBy= 或 RequiresMountsFor= 指令)。
无论是隐式自动生成的先后顺序，还是显式明确设定的先后顺序，都会被 list-dependencies 命令显示出来。

--before
在使用 list-dependencies 命令时，显示在时间顺序上晚于指定单元的那些单元，也就是递归的列出 Before= 中的单元。

-l, --full
在 status, list-units, list-jobs, list-timers 命令的输出中，
显示完整的单元名称、进程树项目、日志输出、单元描述，也就是不省略或截短它们。

--show-types
在显示 socket 时，同时显示其类型。

--job-mode=
当向作业队列中加入新任务(job)时，该选项设定如何处理队列中已有的任务。可以设为下列值之一：
"fail", "replace", "replace-irreversibly", "isolate", "ignore-dependencies", "ignore-requirements", "flush"
仅在使用 isolate 命令时，默认值为"isolate"且不能更改，否则默认值为"replace"。

"fail"表示如果新加入的任务与队列中已有的任务冲突，那么该命令将失败。
所谓"冲突"的含义是：导致某个队列中已有的启动操作转变为停止操作，或者相反。

"replace"表示将队列中冲突的任务按需替换为新加入的任务。

"replace-irreversibly"与"replace"类似，不同之处在于将新加入的任务同时标记为"不可撤销"，
从而确保此任务不会在未来被撤消(即使未来与其他新加入的任务发生冲突也不会被撤消)。
注意，这个"不可撤销"的任务，依然可以使用 cancel 命令显式的撤消。

"isolate"仅可用于启动操作，表示在该单元启动之后，所有其他单元都会被停止。
当使用 isolate 命令的时候，这是默认值，且不能设为其他值。

"flush"表示撤消所有既有的任务队列，然后加入新的任务。

"ignore-dependencies"表示忽略新任务的所有单元依赖(包括先后顺序依赖)，立即执行要求的操作。
如果成功，那么所有被依赖的单元以及先后顺序都将被忽略。主要用于调试目的，切勿用于常规目的。

"ignore-requirements"类似于"ignore-dependencies"，但是仅忽略必须的依赖，而依然遵守时间上的先后顺序依赖。

--fail
这是 --job-mode=fail 的快捷方式。
当与 kill 命令一起使用时，如果没有任何单元被杀死，那么将会导致报错。

-i, --ignore-inhibitors
当关闭或休眠系统时，忽略 inhibitor 锁。应用程序可以利用此锁防止某些重要操作(例如刻录光盘)被关机或休眠打断。
任何用户都可以获取这种锁，但是特权用户可以撤消或者忽略这种锁。
正常情况下，关机与休眠动作会因这种锁的存在而失败(无论该动作是否由特权用户发起)，同时所有已激活的锁也都会被显示出来。
但如果使用了此选项，这种锁将被忽略，关机或休眠将会照常执行，同时也不再显示这些已激活的锁。

-q, --quiet
安静模式，也就是禁止输出任何信息到标准输出。但是这并不适用于输出信息是唯一结果的命令(例如 show)。
注意，显示在标准错误上的出错信息永远不会被屏蔽。

--no-block
默认为阻塞模式，也就是任务经过校验、排入任务队列之后，systemctl 必须一直等到单元启动完成才算执行结束。
使用此选项之后，将变为无阻塞模式，也就是任务排入队列之后，即算 systemctl 执行结束(不必等待单元启动完成)。

--user
与当前调用用户的用户服务管理器(用户实例)通信，而不是默认的系统服务管理器(系统实例)。

--system
与系统服务管理器(系统实例)通信，这是默认值。

--no-wall
在执行 halt, poweroff, reboot 动作前，不发送警告消息。

--global
与 enable 和 disable 命令连用，表示在全局用户单元配置目录[通常是"/etc/systemd/user"]上操作，
从而全局的启用或禁用一个单元，这会影响到所有未来登入的用户。

--no-reload
与 enable 和 disable 命令连用，表示在完成操作之后不重新加载配置文件(默认会隐含的重新加载)。

--no-ask-password
与 start, reload, restart, try-restart, reload-or-restart, reload-or-try-restart, isolate 命令连用，表示禁止询问密码。

单元在启动时可能要求输入密码(例如解锁加密过的文件系统或证书)。
除非从终端使用此选项调用 systemctl 命令，否则 systemctl 将会在终端上向用户询问所需的密码。
当使用此选项以后，必须通过其他方法提供密码(例如通过图形化的密码代理程序)，否则此单元可能会启动失败。
使用此选项还会导致无法在验证用户身份时，从终端输入用户账户的密码。

--kill-who=
与 kill 命令连用，表示向哪个进程发送信号(--signal=)。
可以设为"main"(仅杀死主进程)或"control"(仅杀死控制进程)或默认值"all"(杀死全部进程)。
所谓"主进程"是指定义了单元生存期的进程。所谓"控制进程"是指用于改变单元状态的进程。
例如，所有根据 ExecStartPre=, ExecStop=, ExecReload= 的设置而启动的进程都是控制进程。
注意，对于一个单元来说，同一时刻只能存在一个控制进程，因为同一时刻只能存在一个状态变化的动作。
对于 Type=forking 类型的服务来说，根据 ExecStart= 的设置而启动的初始进程就是一个控制进程，
而此进程随后派生出来的、作为守护进程运行的那个进程，则是该单元的主进程(如果它可以被检测到的话)。
但对于其他类型的服务来说，情况则又有所不同：根据 ExecStart= 的设置而启动的初始进程始终是该服务的主进程。
一个服务单元可以包含以下进程：零个或一个主进程，零个或一个控制进程，任意数量(可以是零个)的其他进程。
注意，不是所有类型的单元都含有上述三种进程。
例如，对于 mount 类型的单元来说，就仅有控制进程(/usr/bin/mount 与 /usr/bin/umount)，而没有主进程。

-s, --signal=
与 kill 命令连用，表示向目标进程发送哪个信号。必须是诸如 SIGTERM(默认值), SIGINT, SIGSTOP 之类的众所周知的信号。

-f, --force
当与 enable 命令连用时，表示覆盖一切现存的有冲突的符号链接。

当与 halt, poweroff, reboot, kexec 命令连用时，表示直接强制执行选定的操作，而不先停止所有单元。
不过，所有进程都将被强制杀死，并且所有文件系统都将被卸载或者以只读模式重新挂载。
这可以算是一种野蛮但还算相对比较安全的快速关机或重启的方法。
如果连续使用 --force 选项两次，那么将既不杀死进程，也不卸载文件系统，而是直接强制关机或重启。
这将会导致数据丢失，文件系统不一致等不良后果。

--message=
当与 halt, poweroff, reboot, kexec 命令一起使用时，设置一个解释为什么进行该操作的字符串。
此字符串将与默认的关机消息一起记录到日志中。

--now
当与 enable 命令连用时，表示同时还要启动该单元。当与 disable 或 mask 命令连用时，表示同时还要停止该单元。
注意，仅在 enable 或 disable/mask 操作确实成功之后，才会进一步执行启动或停止操作。

--root=
与 enable, disable, is-enabled 等相关命令连用，用于设定寻找单元文件时的根目录。

--runtime
当与 enable, disable, edit 等相关命令连用时，表示仅作临时性变更，从而让这些变更在重启后丢失。
这意味着所做的变更将会保存在 /run 目录下(立即生效但重启后该目录的内容将全部丢失)，而不是保存在 /etc 目录下。

类似的，当与 set-property 命令连用时，所做变更亦是临时性的，这些变更在重启后亦会丢失。

--preset-mode=
与 preset 或 preset-all 命令连用，可设为下列值之一：
默认值"full"表示完全按照预设规则启用与禁用各单元。
"enable-only"表示仅按照预设规则启用各单元。
"disable-only"表示仅按照预设规则禁用各单元。

-n, --lines=
与 status 命令连用，控制日志的显示行数(从最新的一行开始计算)。必须设为一个正整数，默认值是"10"。

-o, --output=
与 status 命令连用，控制日志的显示格式。默认值为"short"，详见 journalctl(1) 手册。

--firmware-setup
与 reboot 命令连用，要求系统主板的UEFI固件重启到安装模式。仅支持某些以UEFI模式启动的主板。

--plain
与 list-dependencies, list-units, list-machines 命令连用，让输出从默认的树形变为列表型。

-H, --host=
通过SSH协议在远程主机上操作。可以设为一个主机名(hostname)，或者 username@hostname 格式。
hostname 后面可以加上容器名(以":"分隔)，也就是形如 hostname:container 的格式，表示直接连接到指定主机的指定容器内。
可以通过 machinectl -H *HOST* 命令列出远程主机上的容器名称。

-M, --machine=
在指定的本地容器内执行操作。

--no-pager
不将输出内容通过管道传递给分页程序(一般是 less 程序)。
也就是，即使输出内容已经长于一屏而必须滚屏，也全部一次性输出所有内容。

--no-legend
不打印列标题。也就是不显示列头和列尾的提示性标题。

-h, --help
输出简短的帮助信息后退出。

--version

输出简短的版本信息后退出。

单元命令(COMMAND)

模式(PATTERN)参数的语法与文件名匹配语法类似：用"*"代表任意数量的字符，用"?"代表单个字符，用"[]"指定字符范围。
如果给出了模式(PATTERN)参数，那么表示该命令仅作用于其名称与至少一个模式相匹配的单元。
systemctl 能够识别下列单元命令：

list-units [PATTERN...]
列出已知的单元(可以用 -t 选项限制单元的类型)。这是默认的命令。

list-sockets [PATTERN...]
列出套接字(socket)单元，并按照监听地址排序。
该命令的输出大致像下面这样子：

```
LISTEN           UNIT                         ACTIVATES
/dev/initctl     systemd-initctl.socket       systemd-initctl.service
...
[::]:22          sshd.socket                  sshd.service
kobject-uevent 1 systemd-udevd-kernel.socket systemd-udevd.service

5 sockets listed.
```

注意：因为监听地址中有可能包含空格，所以不适合对这个命令的输出使用程序进行分析。
参见 --show-types, --all, --state= 选项。

list-timers [PATTERN...]
列出定时器(timer)单元，并按照下次执行时间排序。
参见 --all, --state= 选项。

start PATTERN...
启动(activate)指定的已加载单元(未加载的单元不会被启动)。
如果某个单元未被启动，又没有处于失败(failed)状态，通常是因为它没有被加载，所以根本没有被模式匹配到。
此外，对于从模板文件中实例化而来的单元，因为 systemd 会在其尚未启动前忽略它们，
所以，在这个命令中使用包含通配符的模式并没有多少实际意义。

stop PATTERN...
停止(deactivate)指定的单元。

reload PATTERN...
要求指定的单元重新加载它们的配置。
注意，这里所说的"配置"是服务进程专属的配置(例如 httpd.conf 之类)，而不是 systemd 的"单元配置文件"。
如果你想重新重新加载 systemd 的"单元配置文件"，应该使用 daemon-reload 命令。
以 Apache 为例，该命令会导致重新加载 httpd.conf 文件，而不是 apache.service 文件，
所以不要将此命令与 daemon-reload 命令混淆。

restart PATTERN...
重新启动指定的单元。对于尚未启动的单元，则启动它们。

try-restart PATTERN...
重新启动指定的已启动单元。注意，对于尚未启动的单元，不做任何操作。
出于兼容 Red Hat 系统的原因，该命令等价于 condrestart 命令。

reload-or-restart PATTERN...
首先尝试重新加载指定单元的进程专属的配置，对于那些失败的单元，再继续尝试重新启动它们。
对于尚未启动的单元，则启动它们。

reload-or-try-restart PATTERN...
首先尝试重新加载指定单元的进程专属的配置，对于那些失败的单元，再继续尝试重新启动它们。
对于尚未启动的单元，则不做任何动作。
出于兼容SysV初始化脚本的原因，该命令等价于 force-reload 命令。

isolate NAME
启动指定的单元以及该单元的所有依赖单元，同时停止所有其他单元。如果没有给出单元的后缀名，那么视为以".target"作为后缀名。
这类似于传统上切换SysV运行级的概念。
该命令将会立即停止所有在新的目标单元中不需要的进程，这其中有可能包含当前正在运行的图形环境以及正在使用的终端。
注意，该命令仅可用于 AllowIsolate=true 的单元。参见 systemd.unit(5) 手册页。

kill PATTERN...
向指定单元的 --kill-who= 进程发送 --signal= 信号。

is-active PATTERN...
检查指定的单元中，是否有处于活动(active)状态的单元。
如果存在至少一个处于活动(active)状态的单元，那么返回"0"值，否则返回非零值。
除非同时使用了 --quiet 选项，否则，此命令还会在标准输出上打印所指定的单元的状态。

is-failed PATTERN...
检查指定的单元中，是否有处于失败(failed)状态的单元。
如果存在至少一个处于失败(failed)状态的单元，那么返回"0"值，否则返回非零值。
除非同时使用了 --quiet 选项，否则，此命令还会在标准输出上打印所指定的单元的状态。

status [PATTERN...|PID...]

如果指定了单元，则显示指定单元的运行时状态信息，以及这些单元最近的日志数据。
如果指定了PID，则显示指定PID所属单元的运行时状态信息，以及这些单元最近的日志数据。
如果未指定任何单元或PID，则显示整个系统的状态信息，
此时若与 --all 连用，则同时显示所有单元(可以用 -t 限定单元类型)的状态信息。

此命令的目的是生成人类可读的输出，不应当用于程序分析(应该使用 show 命令)。
默认只输出10行日志，除非使用了 --lines 与 --full 选项，否则超长的部分将会被省略号替代。
此外，journalctl --unit=NAME 或 journalctl --user-unit=NAME 也会对消息使用类似的省略号替代。

show [PATTERN...|JOB...]
显示指定单元或任务的属性及其值。如果没有指定任何单元或任务，则显示systemd自身的属性及其值。
单元用其名称表示，而任务则用其id表示。默认不显示属性值为空的属性，除非使用了 --all 选项。
可以使用 --property= 选项指定仅显示特定的属性。
此命令的目的是为了用于程序分析，不适合于人类的阅读(应该使用 status 命令)。

cat PATTERN...
显示指定单元的单元文件内容。在显示每个单元文件内容之前，会附加一行单元文件的绝对路径。

set-property NAME ASSIGNMENT...
在运行时设置特定的单元属性的值。主要用于改变单元的资源控制属性值而无需直接修改单元文件。
并非所有属性都可以在运行时被修改，但大多数用于资源控制的属性(参见 systemd.resource-control(5))可以。
所作修改会立即生效，并将永久保存在磁盘上，以确保永远有效。
但是如果使用了 --runtime 选项，那么此设置仅临时有效，下次重启此单元后，将会恢复到原有的设置。
设置语法与单元文件中的写法相同，例如：

```
systemctl set-property foobar.service CPUShares=777
```

注意，此命令可以同时改变多个属性值，只需依次将各个设置用空格分隔即可。
与单元文件的规定相同，设置一个空列表表示清空当前已存在的列表。

help PATTERN...|PID...
显示指定单元的手册页(若存在)。指定PID表示显示该进程所属单元的手册页(若存在)。

reset-failed [PATTERN...]
重置指定单元的失败(failed)状态。如果未指定任何单元，则重置所有单元的失败(failed)状态。
当某个单元因为某种原因操作失败(例如退出状态码不为零或进程被强制杀死或超时)，
将会自动进入失败(failed)状态，退出状态码与导致故障的原因将被记录到日志以方便日后排查。

list-dependencies [NAME]
显示单元的依赖关系。
也就是显示 Requires=, RequiresOverridable=, Requisite=, RequisiteOverridable=, Wants=, BindsTo= 所形成的依赖关系。
如果没有明确指定单元的名称，则表示显示 default.target 的依赖关系树。

默认情况下，仅以递归方式显示 target 单元的依赖关系树，而对于其他类型的单元，仅显示一层依赖关系(不递归)。
但是如果使用了 --all 选项，那么将对所有类型的单元都强制递归的显示完整的依赖关系树。
还可以使用 --reverse, --after, --before 选项指定仅显示特定类型的依赖关系。

单元文件命令(COMMAND)

模式(PATTERN)参数的语法与文件名匹配语法类似：用"*"代表任意数量的字符，用"?"代表单个字符，用"[]"指定字符范围。
如果给出了模式(PATTERN)参数，那么表示该命令仅作用于其名称与至少一个模式相匹配的单元。
systemctl 能够识别下列单元文件命令：

list-unit-files [PATTERN...]
列出已安装的单元文件及其启用状态(相当于同时使用了 is-enabled 命令)。

enable NAME...
启用指定的单元文件或单元实例。
这将会按照单元文件中"[Install]"小节的指示，在单元配置目录中创建指向所列单元文件的软链接。
创建完软连接之后，systemd 将会重新加载自身的配置(相当于执行 daemon-reload 命令)以确保所做的变更立即生效。
除非使用了 --now 选项(相当于额外再执行 start 命令)，否则启用单元文件并不会导致该单元被启动。
注意，对于单元实例，软链接自身的文件名是实例化之后的名称，但是所指向的目标文件则是该单元的模板文件。

除非使用了 --quiet 选项，否则此命令会打印出所执行的动作。

注意，此命令仅会按照单元文件中"[Install]"小节预设的名称创建软链接，并且是维护单元配置目录的首选方法。
不过，系统管理员亦可以手动修改单元配置目录中的内容，特别是在需要创建不同于默认软链接名称的时候。
此时，系统管理员必须在修改完成后手动执行 daemon-reload 命令，以确保所做的变更立即生效。

不要将 enable 命令与 start 命令混淆，它们是相互独立的命令：可以启动一个尚未启用的单元，也可以启用一个尚未启动的单元。
enable 命令只是设置单元的启动钩子(通过创建软链接)，例如在系统启动时或者某个硬件插入时。
而 start 命令则是执行单元的具体动作，例如对于服务单元来说是启动守护进程，而对于套接字单元来说则是绑定套接字，等等。

若与 --user 选项连用，则表示变更仅作用于用户实例，否则默认作用于系统实例(相当于使用 --system 选项)。
若与 --runtime 选项连用，则表示仅作临时性变更(重启后所有变更都将丢失)，否则默认为永久性变更。
若与 --global 选项连用，则表示变更作用于所有用户(在全局用户配置目录上操作)，否则默认仅作用于当前用户。
注意，当与 --runtime 选项连用时，systemd 守护进程不会重新加载配置。

不可将此命令应用于已被 mask 命令屏蔽的单元，否则将会导致错误。

disable NAME...

禁用指定的单元文件。这将会从单元配置目录中删除所有指向所列单元文件的软链接(包括手动创建的软链接)。
删除完软连接之后，systemd 将会重新加载自身的配置(相当于执行 daemon-reload 命令)以确保所做的变更立即生效。
除非使用了 --now 选项(相当于额外再执行 stop 命令)，否则禁用单元文件并不会导致该单元被停止。

除非使用了 --quiet 选项，否则此命令会打印出所执行的动作。
有关 --system, --user, --runtime, --global 选项的影响，参见上面对 enable 命令的解释。

reenable NAME...
重新启用指定的单元文件。这相当于先使用 disable 命令之后再使用 enable 命令。
通常用于按照单元文件中"[Install]"小节的指示重置软链接名称。

preset NAME...
按照预设文件(*.preset)的指示，重置指定的单元文件。
其效果等价于按照预设规则，对列出的单元文件依次使用 disable 或 enable 命令。
可以使用 --preset-mode= 选项控制如何参照预设文件。
有关预设文件的更多说明，详见 systemd.preset(5) 手册与 Preset 文档。

preset-all
按照预设文件(*.preset)的指示，重置全部的单元文件。
可以使用 --preset-mode= 选项控制如何参照预设文件。

is-enabled NAME...
检查是否有至少一个指定的单元文件已经被启用。
如果有至少一个单元文件已经被启用，那么返回"0"，否则返回非零。
除非使用了 --quiet 选项，否则此命令会打印出指定的单元文件的当前启用状态：

返回值	NAME	含义
0	"enabled"	已经通过 /etc/systemd/system/ 下的 .requires/ 或 .wants/ 目录中的软链接被永久启用
0	"enabled-runtime"	已经通过 /run/systemd/system/ 下的 .requires/ 或 .wants/ 目录中的软链接被临时启用
大于零	"linked"	虽然单元文件尚不在搜索目录中，但指向此单元文件的软连接已经存在于 /etc/systemd/system/ 目录中
大于零	"linked-runtime"	虽然单元文件尚不在搜索目录中，但指向此单元文件的软连接已经存在于 /run/systemd/system/ 目录中
大于零	"masked"	已经被 /etc/systemd/system/ 目录永久屏蔽，因此 start 操作会失败
大于零	"masked-runtime"	已经被 /run/systemd/systemd/ 目录临时屏蔽，因此 start 操作会失败
0	"static"	尚未被启用，且"[Install]"小节中没有可用于 enable 命令的选项
0	"indirect"	尚未被启用，但"[Install]"小节中 Also= 选项的值列表非空(其中的某些单元可能已被启用)
大于零	"disabled"	尚未被启用，但"[Install]"小节中存在可用于 enable 命令的选项
大于零	"bad"	单元文件不正确或者出现其他错误。is-enabled 不会返回此状态，而是会显示一条出错信息。list-unit-files 命令有可能会显示此单元。

mask NAME...
屏蔽指定的单元文件。也就是将这些单元文件指向 /dev/null 文件，以从根本上确保这些单元无法被启动。
这比 disable 命令禁用的更彻底，可以通杀一切启动方法(包括手动启动)，所以应该谨慎使用该命令。
若与 --runtime 选项连用，则表示仅作临时性屏蔽(重启后屏蔽将失效)，否则默认为永久性屏蔽。
除非使用了 --now 选项(相当于额外再执行 stop 命令)，否则仅仅屏蔽单元文件并不会导致该单元被停止。

unmask NAME...
解除对指定的单元文件的屏蔽，这是 mask 命令的反动作。

link FILENAME...
将不在标准单元文件搜索路径中的单元文件(通过软链接)连接到标准搜索目录中去。FILENAME 参数必须是单元文件的绝对路径。
该命令的结果可以通过 disable 命令撤消。通过该命令，可以让一个不在标准单元文件搜索路径中的单元文件，
也可以像位于标准搜索目录中的常规单元文件一样，被 start, stop, ... 等各种命令操作。

add-wants TARGET NAME..., add-requires TARGET NAME...
将指定的单元文件(NAME...)作为 "Wants=" 或 "Requires=" 依赖，添加到 TARGET 单元中。
有关 --system, --user, --runtime, --global 选项的影响，参见上面对 enable 命令的解释。

edit NAME...
调用文本编辑器(参见下面的"环境变量"小节)编辑指定的单元文件。
注意，编辑器实际操作的只是临时文件，仅当编辑器正常退出时，临时文件的内容才会被合并到实际的单元文件中。
注意，如果在编辑器退出时，临时文件的内容为空，则表示取消编辑动作(而不是合并空文件)。

若使用 --full 选项，则表示使用临时文件的内容替换原单元文件的内容，否则默认为将临时文件的内容附加到原单元文件的末尾。
有关 --system(默认值), --user, --runtime, --global 选项的影响，参见上面对 enable 命令的解释。
编辑动作完成之后，systemd 将会重新加载自身的配置(相当于执行 daemon-reload 命令)以确保所做的变更立即生效。

注意，该命令不可用于编辑远程机器上的单元文件。
注意，不可以在编辑 /etc 中的单元时使用 --runtime 选项，因为 /etc 中单元的优先级高于 /run 中单元的优先级。

get-default
返回默认的启动目标。这将返回 default.target 软链接所指向的实际单元名称。

set-default NAME
设置默认的启动目标。这会将 default.target 软链接指向 NAME 单元。

机器命令(COMMAND)

list-machines [PATTERN...]
列出主机和所有运行中的本地容器，以及它们的状态。
如果给出了模式(PATTERN)参数，那么仅显示其名称与至少一个模式匹配的容器。

任务(job)命令(COMMAND)

list-jobs [PATTERN...]
列出正在运行中的任务。如果给出了模式(PATTERN)参数，那么仅显示所属单元名称与至少一个模式匹配的任务。

cancel JOB...
根据给定的任务ID撤消任务。如果没有给出任务ID，则表示撤消所有尚未执行的任务。

环境变量命令(COMMAND)

show-environment
显示所有 systemd 环境变量及其值。显示格式正好符合shell脚本语法，可以直接用于shell脚本中。
这些环境变量会被传递给所有由 systemd 派生的进程。

set-environment VARIABLE=VALUE...
设置指定的 systemd 环境变量。

unset-environment VARIABLE...
撤消指定的 systemd 环境变量。如果仅仅指定了变量名，那么表示无条件的撤消该变量(无论其值是什么)。
如果以 VARIABLE=VALUE 格式同时给出了变量值，那么表示仅当 VARIABLE 的值恰好等于 VALUE 时，才撤消 VARIABLE 变量。

import-environment [VARIABLE...]
导入指定的客户端环境变量。如果未指定任何参数，则表示导入全部的客户端环境变量。

systemd 生命周期命令(COMMAND)

daemon-reload
重新加载 systemd 守护进程的配置。具体是指：重新运行所有的生成器(systemd.generator(7))，重新加载所有单元文件，重建整个依赖关系树
在重新加载过程中，所有由 systemd 代为监听的用户套接字都始终保持可访问状态。
不要将此命令与 reload 命令混淆。

daemon-reexec
重新执行 systemd 守护进程。具体是指：首先序列化 systemd 状态，接着重新执行 systemd 守护进程并反序列化原有状态。
此命令仅供调试和 systemd 升级使用。有时候也作为重量级版本的 daemon-reload 命令使用。
在重新执行过程中，所有由 systemd 代为监听的用户套接字都始终保持可访问状态。

系统命令(COMMAND)

is-system-running
检查当前系统是否处于正常运行状态(running)，若是则返回"0"，否则返回大于零的正整数。
所谓正常运行状态是指：系统完成了全部的启动操作，整个系统已经处于完全可用的状态，
特别是没有处于启动/关闭/维护状态，并且没有任何单元处于失败(failed)状态。
除非使用了 --quiet 选项，否则此命令会在标准输出上打印出系统的当前状态，如下表所示：

状态	含义
initializing	启动的早期阶段。也就是尚未到达 basic.target/rescue.target/emergency.target 之前的阶段。
starting	启动的晚期阶段。也就是任务队列首次达到空闲之前的阶段，或者进入了某个救援 target 中。
running	完成了全部的启动操作，整个系统已经处于完全可用的状态，并且没有任何单元处于失败(failed)状态。
degraded	完成了全部的启动操作，系统已经可用，但是某些单元处于失败(failed)状态。
maintenance	进入了 rescue.target/emergency.target 状态中。
stopping	系统正处于关闭过程中。
offline	整个系统已经处于完全可用的状态，但init进程(PID=1)不是systemd
unknown	由于资源不足或未知原因，无法检测系统的当前状态

default
进入默认模式。差不多相当于执行"systemctl isolate default.target"命令。

rescue
进入救援模式。差不多相当于执行"systemctl isolate rescue.target"命令。
但同时会向所有用户显示一条警告信息。

emergency
进入维护模式。差不多相当于执行"systemctl isolate emergency.target"命令。
但同时会向所有用户显示一条警告信息。

halt
关闭系统，但不切断电源。差不多相当于执行"systemctl start halt.target --job-mode=replace-irreversibly"命令。
但同时会向所有用户显示一条警告信息。
若使用了 --force 选项，则跳过服务的正常关闭步骤而直接杀死所有进程，强制卸载所有文件系统(或只读挂载)，并立即关闭系统。
若使用了两次 --force 选项，则跳过杀死进程和卸载文件系统的步骤，并立即关闭系统，这会造成数据丢失或损坏。

poweroff
关闭系统，同时切断电源。差不多相当于执行"systemctl start poweroff.target --job-mode=replace-irreversibly"命令。
但同时会向所有用户显示一条警告信息。
若使用了 --force 选项，则跳过服务的正常关闭步骤而直接杀死所有进程，强制卸载所有文件系统(或只读挂载)，并立即关闭系统。
若使用了两次 --force 选项，则跳过杀死进程和卸载文件系统的步骤，并立即关闭系统，这会造成数据丢失或损坏。

reboot [arg]
关闭系统，然后重新启动。差不多相当于执行"systemctl start reboot.target --job-mode=replace-irreversibly"命令。
但同时会向所有用户显示一条警告信息。
若使用了 --force 选项，则跳过服务的正常关闭步骤而直接杀死所有进程，强制卸载所有文件系统(或只读挂载)，并立即关闭系统。
若使用了两次 --force 选项，则跳过杀死进程和卸载文件系统的步骤，并立即关闭系统，这会造成数据丢失或损坏。

若给出了可选的 arg 参数，那么将会被作为可选参数传递给 reboot(2) 系统调用。其取值范围依赖于特定的硬件平台。
例如，"recovery"有可能表示触发系统恢复动作，而"fota"有可能表示"firmware over the air"。

kexec
关闭系统，并通过内核的 kexec 接口重新启动。差不多相当于执行"systemctl start kexec.target --job-mode=replace-irreversibly"命令。
但同时会向所有用户显示一条警告信息。
若使用了 --force 选项，则跳过服务的正常关闭步骤而直接杀死所有进程，强制卸载所有文件系统(或只读挂载)，并立即关闭系统。

exit [EXIT_CODE]
让 systemd 按照指定的退出码(必须是整数)退出。若未指定 EXIT_CODE 则退出码为零。
仅可用于用户实例(也就是以 --user 选项启动的实例)或容器(相当于 poweroff)，否则会导致失败。

switch-root ROOT [INIT]
将系统的根目录切换到 ROOT 目录并执行新的 INIT 程序(PID=1)。此命令仅应该在初始内存盘("initrd")中使用。
如果未指定 INIT 参数，那么表示自动在 ROOT 目录下搜索 systemd 二进制程序，并用作 INIT 程序，
同时"initrd"中 systemd 的状态将会传递给新的 systemd 进程，从而允许在新系统中对原"initrd"中的各种服务状态进行内省。

suspend
休眠到内存。相当于执行"systemctl isolate suspend.target"命令。

hibernate
休眠到硬盘。相当于执行"systemctl isolate hibernate.target"命令。

hybrid-sleep
进入混合休眠模式。也就是同时休眠到内存和硬盘。相当于执行"systemctl isolate hybrid-sleep.target"命令。

参数语法

单元命令的参数可能是一个单独的单元名称(NAME)，也可能是多个单元模式(PATTERN...)。

对于第一种情况，如果省略单元后缀，那么默认以".service"为后缀，除非那个命令只能用于某种特定类型的单元。
例如，"systemctl start sshd"命令等价于"systemctl start sshd.service"命令，
而"systemctl isolate default"命令等价于"systemctl isolate default.target"命令，因为 isolate 命令只能用于target单元。

注意，设备文件路径(绝对路径)会自动转化为device单元名称，其他路径(绝对路径)会自动转化为mount单元名称。例如，如下命令
```
  systemctl status /dev/sda
  systemctl status /home
```
分别等价于
```
  systemctl status dev-sda.device
  systemctl status home.mount
```

对于第二种情况，可以在模式中使用shell风格的匹配符，对所有已加载的单元进行名称匹配。
如果没有使用匹配符并且省略了单元后缀，那么处理方式与第一种情况完全相同。
这就意味着：如果没有使用匹配符，那么该模式就等价于一个单独的单元名称(NAME)，只表示一个明确的单元。
如果使用了匹配符，那么该模式就可以匹配任意数量的单元(包括零个)。

模式使用fnmatch(3)语法，也就是可以使用shell风格的 "*", "?", "[]" 匹配符(详见glob(7))。
模式将基于所有已加载的单元进行匹配，如果某个模式未能匹配到任何单元，那么将会被悄无声息的忽略掉。
例如，"systemctl stop sshd@*.service"命令将会停止所有 sshd@.service 的实例单元。

单元文件命令的 NAME 参数必须是一个单元文件的完整名称或绝对路径。例如：
```
  systemctl enable foo.service
```
或
```
  systemctl link /path/to/foo.service
```

退出状态

成功返回"0"，否则返回非零。

环境变量

$SYSTEMD_EDITOR
编辑单元文件时所使用的编辑器，会覆盖 $EDITOR 与 $VISUAL 的值。
如果 $SYSTEMD_EDITOR, $EDITOR, $VISUAL 都不存在或无法使用，那么将会依次尝试使用 nano(1), vim(1), vi(1) 编辑器。

$SYSTEMD_PAGER
在未指定 --no-pager 选项时所使用的分页程序，会覆盖 $PAGER 的值。
将此变量设为空或"cat"等价于使用 --no-pager 选项。

$SYSTEMD_LESS

设置传递给 less(1) 工具的默认选项("FRSXMK")。

参见

systemd(1), journalctl(1), loginctl(1), machinectl(1), systemd.unit(5), systemd.resource-control(5), systemd.special(7), wall(1), systemd.preset(5), systemd.generator(7), glob(7)

systemctl(1) systemd-228 systemctl(1)