systemd-run 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

手册索引 · 指令索引 systemd-241

名称

systemd-run — 在临时 scope 或 service 单元中运行命令

大纲

systemd-run [OPTIONS...] *COMMAND* [ARGS...]

systemd-run [OPTIONS...] [PATH OPTIONS...] {*COMMAND*} [ARGS...]

systemd-run [OPTIONS...] [SOCKET OPTIONS...] {*COMMAND*} [ARGS...]

systemd-run [OPTIONS...] [TIMER OPTIONS...] {*COMMAND*} [ARGS...]

描述

systemd-run 可以创建并启动一个临时的 .service 或 .scope 单元，并在其中运行 *COMMAND* 命令。 此外，**systemd-run** 还可以创建并启动一个临时的 .path, .socket, .timer 单元以实现在特定的条件下启动(用于运行 *COMMAND* 命令的)临时 .service 单元。

如果将某个命令作为临时服务单元来运行，那么它将像普通的服务单元一样接受 systemd 的管理，并且与其他单元一样，也会在 **systemctl list-units** 的输出中被显示出来。 该命令将会运行在一个干净的、独立的执行环境中，并以 systemd 进程作为其父进程。 **systemd-run** 将会以异步模式在后台启动临时服务并在命令开始执行之后返回(除非明确使用了 --no-block 或 --wait 选项，详见后文)。

如果将某个命令作为临时 scope 单元来运行(通过使用 --scope 选项)， 那么该命令将被作为 **systemd-run** 的子进程来运行，因此，该命令将会继承 **systemd-run** 的执行环境。 但同时需要注意的是，该命令的进程依然将像普通的服务单元一样接受 systemd 的管理， 并且与其他单元一样，也会在 **systemctl list-units** 的输出中被显示出来。 在这种情况下，**systemd-run** 将会以同步模式运行指定的命令，并等待命令执行完成之后才会返回。

如果使用了 path, socket, timer 选项(--*-property=)来运行命令， 那么将会在创建临时 service 单元的同时创建一个关联的临时 path, socket, timer 单元。 只有临时 path, socket, timer 单元会被立即启动，而临时 service 单元将会被 path, socket, timer 单元启动(触发)。 如果在使用了 --unit= 选项的同时又省略了 *COMMAND* 参数， 那么 **systemd-run** 将只创建一个专门用于启动指定单元的临时 .path, .socket, .timer 单元。

默认情况下，**systemd-run** 创建 simple 类型的临时服务(详见 systemd.service(5) 手册)。 因为 simple 类型的服务会让 **systemd-run** 在完成 fork() 之后、 未调用 execve() 之前，即认为临时服务已经启动成功(可能实际上无法启动指定的命令)， 所以可以考虑使用 exec 类型的临时服务(--property=Type=exec) 以确保 **systemd-run** 仅在指定的命令行确实已成功启动时， 才能成功返回。

选项(OPTIONS)

可以识别的选项如下:

--no-ask-password

在执行特权操作时 不向用户索要密码。

--scope

创建一个临时 .scope 单元(而不是默认的 .service 单元)来执行命令。

--unit=

明确指定单元的名称（而不是自动生成）

--property=, -p

为临时 scope 或 service 单元设置一个属性。 此选项接受的值的格式与 systemctl(1) 的 **set-property** 命令相同。

--description=

为临时 service, scope, path, socket, timer 单元设置一个描述性字符串。 默认值是 *COMMAND* 参数自身。参见 systemd.unit(5) 的 Description= 选项。

--slice=

将临时 .service 或 .scope 单元 放入指定的 slice 单元中(而不是默认的 system.slice 单元)。

-r, --remain-after-exit

在服务进程结束之后，继续保持服务的存在，直到被明确的停止(stop)。 这有助于在服务进程运行结束之后收集服务单元的运行时信息。参见 systemd.service(5) 的 RemainAfterExit= 选项。

--send-sighup

在终止临时 scope 或 service 单元时，紧跟 SIGTERM 信号之后再立即发送一个 SIGHUP 信号。 这有助于明确通知 shell 类进程连接已被切断。参见 systemd.kill(5) 的 SendSIGHUP= 选项。

--service-type=

设置临时 service 单元的类型， 默认值是 simple 。参见 systemd.service(5) 的 Type= 选项。此选项不能与 --scope 一起使用。

--uid=, --gid=

以指定的 UID/GID 身份运行服务进程。参见 systemd.exec(5) 的 User= 与 Group= 选项。

--nice=

以指定的谦让级别(nice)运行服务进程。参见 systemd.exec(5) 的 Nice= 选项。

--working-directory=

以指定的工作目录运行服务进程。参见 WorkingDirectory= 选项(systemd.exec(5))。

--same-dir, -d

类似于 --working-directory= 选项，但是使用用户当前的工作目录 运行服务进程。

-E *NAME*=*VALUE*, --setenv=*NAME*=*VALUE*

给服务进程传递一个环境变量。可以多次使用此选项以传递多个环境变量。参见 systemd.exec(5) 的 Environment= 选项。

--pty, -t

通过一个伪终端设备，将临时服务单元的标准输入、标准输出、标准错误连接到运行 **systemd-run** 命令的 TTY 上， 从而允许运行例如 shell 之类需要与用户交互的命令。

注意，如果想要在本地容器或者宿主机上打开一个新的交互式登录会话，那么应该使用 machinectl(1) 的 **shell** 命令。

参见下文以了解如何与 --pipe 一起使用。

--pipe, -P

让临时服务单元直接从 **systemd-run** 命令自身继承标准输入、标准输出、标准错误。 这样就允许在命令行管道中使用 **systemd-run** 命令。 注意，因为服务进程在此模式下不能控制 TTY， 所以不可将此模式用于运行例如 shell 之类需要与用户交互的命令(应该使用 --pty 选项)。

如果同时使用 --pipe 与 --pty 选项， 那么将会自动检测应该使用哪种模式：如果临时服务单元的标准输入、标准输出、标准错误已经连接到了某个 TTY 上， 那么使用 --pty 模式，否则使用 --pipe 模式。

使用此选项之后， **systemd-run** 接收到的文件描述符将原封不动的传递给临时服务进程。 如果临时服务拥有的权限与 **systemd-run** 不同，那么临时服务有可能无权打开传递过来的文件描述符。 如果被调用的进程是一个 shell 脚本， 并且使用类似 **echo "hello" > /dev/stderr** 的方式向标准错误输出消息， 那么可能会导致故障(有可能无权打开 stderr 文件描述符)。解决方案是使用类似 **echo "hello" >&2** 的方式进行替代。

--shell, -S

一个等价于 "--pty --same-dir --wait --collect --service-type=exec $SHELL" 的快捷方式。 也就是在当前工作目录中、在一个临时服务的上下文中，开启一个交互式 shell 来运行指定的命令。

--quiet, -q

安静模式，也就是不显示额外的运行时信息。 经常与 --pty 一起使用， 以略过关于如何结束TTY连接的消息。

--on-active=, --on-boot=, --on-startup=, --on-unit-active=, --on-unit-inactive=

为了在特定的时间运行指定的命令，定义一个与临时服务单元相关联的临时 timer 单元。详见 systemd.timer(5) 手册 OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec=, OnUnitInactiveSec= 的解释。 这些选项都是对应 **--timer-property=** 各种属性的快捷方式。 这些选项不能与 --scope 或 --pty 一起使用。

--on-calendar=

为了在特定的时间运行指定的命令，定义一个与临时服务单元相关联的临时 timer 单元。详见 systemd.timer(5) 手册 OnCalendar= 的解释。 此选项是 **--timer-property=OnCalendar=** 的快捷方式。 此选项不能与 --scope 或 --pty 一起使用。

--path-property=, --socket-property=, --timer-property=

为临时 path, socket, timer 单元设置一个属性。这些选项与 --property= 类似，但是其目标是临时 path, socket, timer 单元(而不是用来运行指定命令的临时服务单元)。 这些选项

接受的值与 systemctl(1) 的 **set-property** 命令相同。这些选项不能与 --scope or --pty 一起使用。

--no-block

不同步等待临时单元完成启动操作。 如果未使用此选项，那么 **systemd-run** 将会在临时单元完成启动操作之后才返回。 使用此选项之后，**systemd-run** 将会立即返回，并以异步方式检查临时单元是否完成了启动操作。 此选项不可与 --wait 一起使用。

--wait

同步等待临时服务单元直到其终止。使用此选项之后， **systemd-run** 将会一直等待临时服务单元(包括所依赖的单元)完成启动， 并且一直等待到临时服务单元最终终止(一般是指定的命令运行结束)， 最后显示临时服务单元的运行时简明统计信息(总运行时长、CPU使用量(如果设置了 --property=CPUAccounting=1 的话))，并返回被执行命令的退出状态。 可以使用 --quiet 选项略过最后的简明统计信息。此选项不可与 --no-block, --scope 以及 path, socket, timer 选项(--*-property=)一起使用。

-G, --collect

即使临时单元执行失败(failed)，也在结束后从内存中卸载它。 如果没有设置此选项，所有运行失败的(failed)的单元都会保留在内存中，直到用户明确使用 **systemctl reset-failed** 或等价命令重置了它们的失败状态。 另一方面，运行成功的单元总是会被立即卸载。使用此选项之后，单元的"垃圾回收"策略将更加激进， 无论单元是否执行成功，都会在执行结束后被卸载。此选项是 **--property=CollectMode=inactive-or-failed** 的快捷方式，详见 CollectMode= 选项(参见 systemd.unit(5) 手册)。

--user

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

--system

与系统服务管理器(systemd 系统实例)通信， 这是默认值。

-H, --host=

操作指定的远程主机。可以仅指定一个主机名(hostname)， 也可以使用 "username@hostname" 格式。 hostname 后面还可以加上 SSH监听端口(以冒号":"分隔)与容器名(以正斜线"/"分隔)， 也就是形如 "hostname:port/container" 的格式， 以表示直接连接到指定主机的指定容器内。 操作将通过SSH协议进行，以确保安全。 可以通过 **machinectl -H *HOST*** 命令列出远程主机上的所有容器名称。IPv6地址必须放在方括号([])内。

-M, --machine=

在本地容器内执行操作。 必须明确指定容器的名称。

-h, --help

显示简短的帮助信息并退出。

--version

显示简短的版本信息并退出。

所有 *COMMAND* 之后的部分都将被当作被调用命令的命令行参数(ARGS)。 如果想要以临时 service 单元的方式运行特定的命令， 那么 *COMMAND* 参数必须是一个可执行文件的绝对路径。

退出状态

返回值为 0 表示成功， 非零返回值表示失败代码。

例子

例 1. 记录 systemd 传递给服务单元的环境变量

```
# systemd-run env
Running as unit: run-19945.service
# journalctl -u run-19945.service
Sep 08 07:37:21 bupkis systemd[1]: Starting /usr/bin/env...
Sep 08 07:37:21 bupkis systemd[1]: Started /usr/bin/env.
Sep 08 07:37:21 bupkis env[19948]: PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
Sep 08 07:37:21 bupkis env[19948]: LANG=en_US.UTF-8
Sep 08 07:37:21 bupkis env[19948]: BOOT_IMAGE=/vmlinuz-3.11.0-0.rc5.git6.2.fc20.x86_64
```

例 2. 限制一个命令可以使用的资源

```
# systemd-run -p BlockIOWeight=10 updatedb
```

运行 updatedb(8) 工具，但是将其块设备 I/O 权重降低到"10"。参见 systemd.resource-control(5) 手册对 BlockIOWeight= 的说明。

例 3. 在特定的时间运行命令

下面的命令将会在30秒之后 touch 指定的文件。

```
# date; systemd-run --on-active=30 --timer-property=AccuracySec=100ms /bin/touch /tmp/foo
Mon Dec  8 20:44:24 KST 2014
Running as unit: run-71.timer
Will run service as unit: run-71.service
# journalctl -b -u run-71.timer
-- Logs begin at Fri 2014-12-05 19:09:21 KST, end at Mon 2014-12-08 20:44:54 KST. --
Dec 08 20:44:38 container systemd[1]: Starting /bin/touch /tmp/foo.
Dec 08 20:44:38 container systemd[1]: Started /bin/touch /tmp/foo.
# journalctl -b -u run-71.service
-- Logs begin at Fri 2014-12-05 19:09:21 KST, end at Mon 2014-12-08 20:44:54 KST. --
Dec 08 20:44:48 container systemd[1]: Starting /bin/touch /tmp/foo...
Dec 08 20:44:48 container systemd[1]: Started /bin/touch /tmp/foo.
```

例 4. 允许被运行的命令访问终端

以临时服务的方式运行 /bin/bash 命令， 并将其标准输入、标准输出、标准错误连接到当前的 TTY 设备上：

```
# systemd-run -t --send-sighup /bin/bash
```

例 5. 在一个临时的用户单元中运行 screen 工具

```
$ systemd-run --scope --user screen
Running scope as unit run-r14b0047ab6df45bfb45e7786cc839e76.scope.

$ screen -ls
There is a screen on:
        492..laptop     (Detached)
1 Socket in /var/run/screen/S-fatima.
```

这将把 **screen** 作为 **systemd --user** 进程(由 user@.service 的某个实例启动)的子进程运行，并将其封装在一个临时 scope 单元中。之所以使用 systemd.scope(5) 单元而不是 systemd.service(5) 单元，是因为 **screen** 会在脱离终端后退出，进而导致 service 单元被终结。 将 **screen** 放在用户单元中运行， 可以避免它成为会话 scope 单元的一部分。 因为，如果默认设置 KillUserProcesses=yes 存在于 logind.conf(5) 之中，那么当用户退出会话时，会话 scope 单元 将会被终结。

user@.service 会在用户首次登录时自动启动， 并且只要仍然存在至少一个用户登录会话，它就始终保持运行。 仅当用户退出最后一个登录会话之后， user@.service 及其所有子服务才会停止运行。 只要没有为用户开启"逗留"特性，那么这就是默认行为。 如果开启了"逗留"特性，那么 user@.service 将会在系统启动时自动启动(即使用户并未登录)， 并且不会随着用户退出所有登录会话而停止。

开启"逗留"特性之后，用户就可以在未登录的情况下运行进程了。 例如，为了确保 **screen** 在用户登出之后也能继续保持运行， 甚至是会话 scope 单元已经被终止的情况下也能继续保持运行， 在默认配置情况下，用户可以使用如下命令为自己开启"逗留"特性：

```
$ loginctl enable-linger
```

参见

systemd(1), systemctl(1), systemd.unit(5), systemd.service(5), systemd.scope(5), systemd.slice(5), systemd.exec(5), systemd.resource-control(5), systemd.timer(5), systemd-mount(1), machinectl(1)