

# systemd (简体中文)

From ArchWiki

翻译状态：本文是英文页面 Systemd 的翻译，最后翻译时间：2015-06-11，[点击这里 \(https://wiki.archlinux.org/index.php?title=Systemd&diff=0&oldid=377056\)](https://wiki.archlinux.org/index.php?title=Systemd&diff=0&oldid=377056)可以查看翻译后英文页面的改动。

摘自项目主页 (<http://freedesktop.org/wiki/Software/systemd>):

**systemd** 是 Linux 下的一款系统和服务管理器，兼容 SysV 和 LSB 的启动脚本。systemd 的特性有：支持并行化任务；同时采用 socket 式与 D-Bus 总线式激活服务；按需启动守护进程 (daemon)；利用 Linux 的 cgroups 监视进程；支持快照和系统恢复；维护挂载点和自动挂载点；各服务间基于依赖关系进行精密控制。

## Contents

- 1 systemd 基本工具
  - 1.1 分析系统状态
  - 1.2 使用单元
  - 1.3 电源管理
- 2 编写单元文件
  - 2.1 处理依赖关系
  - 2.2 服务类型
  - 2.3 修改现存单元文件
- 3 目标 (target)
  - 3.1 获取当前目标
  - 3.2 创建新目标
  - 3.3 目标表
  - 3.4 切换启动级别/目标
  - 3.5 修改默认启动级别/目标
- 4 临时文件
- 5 定时器
- 6 日志
  - 6.1 过滤输出

## 相关文章

[systemd/User \(简体中文\)](#)

[Systemd FAQ \(简体中文\)](#)

[init Rosetta](#)

[Daemons#List of deamons](#)

[udev \(简体中文\)](#)

[Improve Boot Performance \(简体中文\)](#)

[Allow users to shutdown](#)

- 6.2 日志大小限制
- 6.3 配合 syslog 使用
- 6.4 Forward journald to /dev/tty12
- 7 疑难解答
  - 7.1 Investigating systemd errors
  - 7.2 诊断启动问题
  - 7.3 Diagnosing problems with a specific service
  - 7.4 关机/重启十分缓慢
  - 7.5 短时进程无日志记录
  - 7.6 禁止在程序崩溃时转储内存
  - 7.7 Error message on reboot or shutdown
    - 7.7.1 cgroup : option or name mismatch, new: 0x0 "", old: 0x4 "systemd"
    - 7.7.2 watchdog watchdog0: watchdog did not stop!
  - 7.8 Boot time increasing over time
- 8 相关资源

## systemd 基本工具

检视和控制systemd的主要命令是 `systemctl`。该命令可用于查看系统状态和管理系统及服务。详见 `man 1 systemctl`。

### 小贴士:

- 在 `systemctl` 参数中添加 `-H <用户名>@<主机名>` 可以实现对其他机器的远程控制。该过程使用 SSH连接。
- `systemadm` 是 `systemd` 的官方图形前端。官方软件仓库 提供了稳定版本 `systemd-ui` (<https://www.archlinux.org/packages/?name=systemd-ui>)，AUR 中的软件包 `systemd-ui-git` (<https://aur.archlinux.org/packages/systemd-ui-git/>)<sup>AUR</sup><sup>[broken link: archived in aur-mirror]</sup> (<http://pkgbuild.com/git/aur-mirror.git/tree/systemd-ui-git>) 提供了开发版本。

## 分析系统状态

输出激活的单元：

```
$ systemctl
```

以下命令等效：

```
$ systemctl list-units
```

输出运行失败的单元：

```
$ systemctl --failed
```

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）。查看所有已安装服务：

```
$ systemctl list-unit-files
```

## 使用单元

一个单元配置文件可以描述如下内容之一：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`）。详情参阅 `man 5 systemd.unit`。

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 `netcfg` 和 `netcfg.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`。

**Note:** 有一些单元的名称包含一个 `@` 标记，(e.g. `name@string.service`): 这意味着它是模板单元 `name@.service` 的一个实例 (<http://0pointer.de/blog/projects/instances.html>)。 `string` 被称作实例标识符，在 `systemctl` 调用模板单元时，会将其当作一个参数传给模板单元，模板单元会使用这个传入的参数代替模板中的 `%I` 指示符。

在实例化之前，`systemd` 会先检查 `name@string.suffix` 文件是否存在（如果存在，应该就是直接使用这个文件，而不是模板实例化了）。大多数情况下，包换 `@` 标记都意味着这个文件是模板。如果一个模板单元没有实例化就调用，该调用会返回失败，因为模板单元中的 `%I` 指示符没有被替换。

### Tip:

- 下面的大部分命令都可以跟多个单元名，详细信息参见 `man systemctl`。
- 从 `systemd 220` 版本 (<https://github.com/systemd/systemd/blob/master/NEWS#L323-L326>) 开始，`systemctl` 命令在 `enable`、`disable` 和 `mask` 子命令中增加了

--now 选项，可以实现激活的同时启动服务，取消激活的同时停止服务。

- 一个软件包可能会提供多个不同的单元。如果你已经安装了软件包，可以通过 `pacman -Qql package | grep systemd` 命令检查这个软件包提供了哪些单元。

立即激活单元：

```
# systemctl start <单元>
```

立即停止单元：

```
# systemctl stop <单元>
```

重启单元：

```
# systemctl restart <单元>
```

重新加载配置：

```
# systemctl reload <单元>
```

输出单元运行状态：

```
$ systemctl status <单元>
```

检查单元是否配置为自动启动：

```
$ systemctl is-enabled <单元>
```

开机自动激活单元：

```
# systemctl enable <单元>
```

取消开机自动激活单元：

```
# systemctl disable <单元>
```

禁用一个单元(禁用后，间接启动也是不可能的)：

```
# systemctl mask <单元>
```

取消禁用一个单元：

```
# systemctl unmask <单元>
```

显示单元的手册页（必须由单元文件提供）：

```
# systemctl help <单元>
```

重新载入 systemd，扫描新的或有变动的单元：

```
# systemctl daemon-reload
```

## 电源管理

安装 polkit 后才可以一般用户身份使用电源管理。

如果你正登录在一个本地的 systemd-logind 用户会话，且当前没有其它活动的会话，那么以下命令无需root权限即可执行。否则（例如，当前有另一个用户登录在某个tty），systemd 将会自动请求输入root密码。

重启：

```
$ systemctl reboot
```

退出系统并停止电源：

```
$ systemctl poweroff
```

待机：

```
$ systemctl suspend
```

休眠：

```
$ systemctl hibernate
```

混合休眠模式（同时休眠到硬盘并待机）：

```
$ systemctl hybrid-sleep
```

## 编写单元文件

systemd 单元文件 (<http://www.freedesktop.org/software/systemd/man/systemd.unit.html>)的语法来源于 XDG桌面入口配置文件 .desktop 文件，最初的源头则是 Microsoft Windows的 .ini 文件。单元文件可以从两个地方加载，优先级从低到高分别是：

- /usr/lib/systemd/system/：软件包安装的单元
- /etc/systemd/system/：系统管理员安装的单元

**Note:** 当 systemd 运行在用户模式下时，使用的加载路径是完全不同的。

单元文件的语法，可以参考系统已经安装的单元，也可以参考 `man systemd.service` 中的 EXAMPLES 章节 (<http://www.freedesktop.org/software/systemd/man/systemd.service.html#Examples>)。

小贴士: 以 # 开头的注释可能也能用在 unit-files 中, 但是只能在新行中使用。不要在 *systemd* 的参数后面使用行末注释，否则 unit 将会启动失败。

## 处理依赖关系

使用systemd时，可通过正确编写单元配置文件来解决其依赖关系。典型的情况是，单元 A 要求单元 B 在 A 启动之前运行。在此情况下，向单元 A 配置文件中的 [Unit] 段添加 `Requires=B` 和 `After=B` 即可。若此依赖关系是可选的，可添加 `Wants=B` 和 `After=B`。请注意 `Wants=` 和 `Requires=` 并不意味着 `After=`，即如果 `After=` 选项没有制定，这两个单元将被并行启动。

依赖关系通常被用在服务 (service) 而不是目标 (target) 上。例如，`network.target` 一般会被某个配置网络接口的服务引入，所以，将自定义的单元排在该服务之后即可，因为 `network.target` 已经启动。

## 服务类型

编写自定义的 service 文件时，可以选择几种不同的服务启动方式。启动方式可通过配置文件 [Service] 段中的 `Type=` 参数进行设置。

- `Type=simple` (默认值)：systemd认为该服务将立即启动。服务进程不会fork。如果该服务要启动其他服务，不要使用此类型启动，除非该服务是socket激活型。
- `Type=forking`：systemd认为当该服务进程fork，且父进程退出后服务启动成功。对于常规的守护进程 (daemon)，除非你确定此启动方式无法满足需求，使用此类型启动即可。使用此启动类型应同时指定 `PIDFile=`，以便systemd能够跟踪服务的主进程。

- `Type=oneshot` : 这一选项适用于只执行一项任务、随后立即退出的服务。可能需要同时设置 `RemainAfterExit=yes` 使得 `systemd` 在服务进程退出之后仍然认为服务处于激活状态。
- `Type=notify` : 与 `Type=simple` 相同, 但约定服务会在就绪后向 `systemd` 发送一个信号。这一通知的实现由 `libsystemd-daemon.so` 提供。
- `Type=dbus` : 若以此方式启动, 当指定的 `BusName` 出现在 `DBus` 系统总线上时, `systemd` 认为服务就绪。
- `Type=idle` : `systemd` 会等待所有任务(`Jobs`)处理完成后, 才开始执行 `idle` 类型的单元。除此之外, 其他行为和 `Type=simple` 类似。

`type` 的更多解释可以参考 `systemd.service(5)` (<http://www.freedesktop.org/software/systemd/man/systemd.service.html#Type=>)。

## 修改现存单元文件

要更改由软件包提供的单元文件, 先创建名为 `/etc/systemd/system/<单元名>.d/` 的目录 (如 `/etc/systemd/system/httpd.service.d/` ), 然后放入 `*.conf` 文件, 其中可以添加或重置参数。这里设置的参数优先级高于原来的单元文件。例如, 如果想添加一个额外的依赖, 创建这么一个文件即可:

```
/etc/systemd/system/<unit>.d/customdependency.conf

[Unit]
Requires=<新依赖>
After=<新依赖>
```

As another example, in order to replace the `ExecStart` directive for a unit that is not of type `oneshot`, create the following file:

```
/etc/systemd/system/unit.d/customexec.conf

[Service]
ExecStart=
ExecStart=new command
```

想知道为什么修改 `ExecStart` 前必须将其置空, 参见 ([1] ([https://bugzilla.redhat.com/show\\_bug.cgi?id=756787#c9](https://bugzilla.redhat.com/show_bug.cgi?id=756787#c9))).

下面是自动重启服务的一个例子:

```
/etc/systemd/system/unit.d/restart.conf

[Service]
Restart=always
RestartSec=30
```

然后运行以下命令使更改生效：

```
# systemctl daemon-reload
# systemctl restart <单元>
```

此外，把旧的单元文件从 `/usr/lib/systemd/system/` 复制到 `/etc/systemd/system/`，然后进行修改，也可以达到同样效果。在 `/etc/systemd/system/` 目录中的单元文件的优先级总是高于 `/usr/lib/systemd/system/` 目录中的同名单元文件。注意，当 `/usr/lib/` 中的单元文件因软件包升级变更时，`/etc/` 中自定义的单元文件不会同步更新。此外，你还得执行 `systemctl reenable <unit>`，手动重新启用该单元。因此，建议使用前面一种利用 `*.conf` 的方法。

小贴士：用 `systemd-delta` 命令来查看哪些单元文件被覆盖、哪些被修改。系统维护的时候需要及时了解哪些单元已经有了更新。

可从 官方仓库 安装 `vim-systemd` (<https://www.archlinux.org/packages/?name=vim-systemd>) 软件包，可以使 unit 配置文件在 Vim 下支持语法高亮。

## 目标 (target)

启动级别 (runlevel) 是一个旧的概念。现在，systemd 引入了一个和启动级别功能相似又不同的概念——目标 (target)。不像数字表示的启动级别，每个目标都有名字和独特的功能，并且能同时启用多个。一些目标继承其他目标的服务，并启动新服务。systemd 提供了一些模仿 sysvinit 启动级别的目标，仍可以使用旧的 `telinit` 启动级别 命令切换。

### 获取当前目标

不要使用 `runlevel` 命令了：

```
$ systemctl list-units --type=target
```

### 创建新目标

在 Fedora 中，启动级别 0、1、3、5、6 都被赋予特定用途，并且都对应一个 systemd 的目标。然而，没有什么很好的移植用户定义的启动级别 (2、4) 的方法。要实现类似功能，可以以原有的启动级别为基础，创建一个新的目标 `/etc/systemd/system/<新目标>`（可以参考 `/usr/lib/systemd/system/graphical.target`），创建 `/etc/systemd/system/<新目标>.wants` 目录，向其中加入额外服务的链接（指向 `/usr/lib/systemd/system/` 中的单元文件）。

### 目标表



SysV 启动级别	Systemd 目标	注释
0	runlevel0.target, poweroff.target	中断系统 (halt)
1, s, single	runlevel1.target, rescue.target	单用户模式
2, 4	runlevel2.target, runlevel4.target, multi-user.target	用户自定义启动级别，通常识别为级别3。
3	runlevel3.target, multi-user.target	多用户，无图形界面。用户可以通过终端或网络登录。
5	runlevel5.target, graphical.target	多用户，图形界面。继承级别3的服务，并启动图形界面服务。
6	runlevel6.target, reboot.target	重启
emergency	emergency.target	急救模式 (Emergency shell)

## 切换启动级别/目标

systemd 中，启动级别通过“目标单元”访问。通过如下命令切换：

```
# systemctl isolate graphical.target
```

该命令对下次启动无影响。等价于 `telinit 3` 或 `telinit 5`。

## 修改默认启动级别/目标

开机启动进的目标是 `default.target`，默认链接到 `graphical.target`（大致相当于原来的启动级别5）。可以通过内核参数更改默认启动级别：

- `systemd.unit=multi-user.target`（大致相当于级别3）
- `systemd.unit=rescue.target`（大致相当于级别1）

另一个方法是修改 `default.target`。可以通过 `systemctl` 修改它：

```
# systemctl set-default multi-user.target
```

要覆盖已经设置的 `default.target`，请使用 `force`：

```
# systemctl set-default -f multi-user.target
```

可以在 `systemctl` 的输出中看到命令执行的效果：链接 `/etc/systemd/system/default.target` 被

创建，指向新的默认启动级别。

## 临时文件

`/usr/lib/tmpfiles.d/` 和 `/etc/tmpfiles.d/` 中的文件描述了 `systemd-tmpfiles` 如何创建、清理、删除临时文件和目录，这些文件和目录通常存放在 `/run` 和 `/tmp` 中。配置文件名称为 `/etc/tmpfiles.d/<program>.conf`。此处的配置能覆盖 `/usr/lib/tmpfiles.d/` 目录中的同名配置。

临时文件通常和服务文件同时提供，以生成守护进程需要的文件和目录。例如 Samba 服务需要目录 `/run/samba` 存在并设置正确的权限位，就象这样：

```
/usr/lib/tmpfiles.d/samba.conf
D /run/samba 0755 root root
```

此外，临时文件还可以用来在开机时向特定文件写入某些内容。比如，要禁止系统从USB设备唤醒，利用旧的 `/etc/rc.local` 可以用 `echo USBE > /proc/acpi/wakeup`，而现在可以这么做：

```
/etc/tmpfiles.d/disable-usb-wake.conf
w /proc/acpi/wakeup - - - USBE
```

详情参见 `systemd-tmpfiles(8)` 和 `man 5 tmpfiles.d`。

**注意：**该方法不能向 `/sys` 中的配置文件添加参数，因为 `systemd-tmpfiles-setup` 有可能在相关模块加载前运行。这种情况下，需要首先通过 `modinfo <模块名>` 确认需要的参数，并在 `/etc/modprobe.d` 下的一个文件中设置改参数。另外，还可以使用 `udev` 规则，在设备就绪时设置相应属性。

## 定时器

定时器是以 `.timer` 为后缀的配置文件，记录由system的里面由时间触发的动作, 定时器可以替代 `cron` 的大部分功能。详情参阅 `systemd/Timers (简体中文)`。

## 日志

`systemd` 提供了自己日志系统 (logging system)，称为 `journal`。使用 `systemd` 日志，无需额外安装日志服务 (`syslog`)。读取日志的命令：

```
# journalctl
```

默认情况下（当 `Storage=` 在文件 `/etc/systemd/journald.conf` 中被设置为 `auto`），日志记录将被写入 `/var/log/journal/`。该目录是 `systemd` (<https://www.archlinux.org/packages/?name=systemd>) 软件包的一部分。若被删除，`systemd` 不会自动创建它，直到下次升级软件包时重建该目录。如果该目录缺失，`systemd` 会将日志记录写入 `/run/systemd/journal`。这意味着，系统重启后日志将丢失。

**Tip:** 如果 `/var/log/journal/` 位于 `btrfs` 文件系统，应该考虑对这个目录禁用写入时复制，方法参阅 [Btrfs#Copy-On-Write \(CoW\)](#)。

## 过滤输出

`journalctl` 可以根据特定字段过滤输出。如果过滤的字段比较多，需要较长时间才能显示出来。

示例：

显示本次启动后的所有日志：

```
# journalctl -b
```

不过，一般大家更关心的不是本次启动后的日志，而是上次启动时的（例如，刚刚系统崩溃了）。可以使用 `-b` 参数：

- `journalctl -b -0` 显示本次启动的信息
- `journalctl -b -1` 显示上次启动的信息
- `journalctl -b -2` 显示上上次启动的信息 `journalctl -b -2`

- Show all messages from date (and optional time):

```
# journalctl --since="2012-10-30 18:17:16"
```

- Show all messages since 20 minutes ago:

```
# journalctl --since "20 min ago"
```

- 显示最新信息

```
# journalctl -f
```

- 显示特定程序的所有消息:

```
# journalctl /usr/lib/systemd/systemd
```

- 显示特定进程的所有消息:

```
# journalctl _PID=1
```

- 显示指定单元的所有消息：

```
# journalctl -u netcfg
```

- Show kernel ring buffer:

```
# journalctl -k
```

- Show auth.log equivalent by filtering on syslog facility:

```
# journalctl -f -l SYSLOG_FACILITY=10
```

详情参阅 `man journalctl`、`man systemd.journal-fields`，以及 Lennert 的这篇博文 (<http://0pointer.de/blog/projects/journalctl.html>)。

## 日志大小限制

如果按上面的操作保留日志的话，默认日志最大限制为所在文件系统容量的 10%，即：如果 `/var/log/journal` 储存在 50GiB 的根分区中，那么日志最多存储 5GiB 数据。可以修改 `/etc/systemd/journald.conf` 中的 `SystemMaxUse` 来指定该最大限制。如限制日志最大 50MiB：

```
SystemMaxUse=50M
```

详情参见 `man journald.conf`。

## 配合 **syslog** 使用

`systemd` 提供了 `socket /run/systemd/journal/syslog`，以兼容传统日志服务。所有系统信息都会被传入。要使传统日志服务工作，需要让服务链接该 `socket`，而非 `/dev/log`（官方说明 (<http://lwn.net/Articles/474968/>)）。Arch 软件仓库中的 `syslog-ng` (<https://www.archlinux.org/packages/?name=syslog-ng>) 已经包含了需要的配置。

As of *systemd* 216 the default `journald.conf` for forwarding to the socket is `no`. This means you will need to set the option `ForwardToSyslog=yes` in `/etc/systemd/journald.conf` to actually use *syslog-ng* with *journald*. See *Syslog-ng#Overview* for details.

If you use `rsyslog` (<https://www.archlinux.org/packages/?name=rsyslog>) instead, it is not necessary to change the option because `rsyslog` pulls the messages from the journal by itself (<http://lists.freedesktop.org/archives/systemd-devel/2014-August/022295.html#journald>).

设置开机启动 syslog-ng :

```
# systemctl enable syslog-ng
```

这里 (<http://0pointer.de/blog/projects/>)有一份很不错的 `journalctl` 指南。

## Forward journald to /dev/tty12

In `/etc/systemd/journald.conf` enable the following:

```
ForwardToConsole=yes
TTYPath=/dev/tty12
MaxLevelConsole=info
```

Restart journald with:

```
# systemctl restart systemd-journald
```

## 疑难解答

### Investigating systemd errors

As an example, we will investigate an error with `systemd-modules-load` service:

**1.** Lets find the *systemd* services which fail to start:

```
$ systemctl --state=failed
systemd-modules-load.service    loaded failed failed    Load Kernel Modules
```

**2.** Ok, we found a problem with `systemd-modules-load` service. We want to know more:

```
$ systemctl status systemd-modules-load
systemd-modules-load.service - Load Kernel Modules
  Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
  Active: failed (Result: exit-code) since So 2013-08-25 11:48:13 CEST; 32s ago
    Docs: man:systemd-modules-load.service(8).
          man:modules-load.d(5)
  Process: 15630 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited, status=1/FAILURE)
```

If the Process ID is not listed, just restart the failed service with `systemctl restart systemd-modules-load`

**3.** Now we have the process id (PID) to investigate this error in depth. Enter the following command with the current Process ID (here: 15630):

```
$ journalctl -b _PID=15630
```

```
-- Logs begin at Sa 2013-05-25 10:31:12 CEST, end at So 2013-08-25 11:51:17 CEST. --
```

```
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'blacklist usbhp'
```

```
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'install usbhp /bin/false'
```

**4.** We see that some of the kernel module configs have wrong settings. Therefore we have a look at these settings in `/etc/modules-load.d/`:

```
$ ls -Al /etc/modules-load.d/
```

```
...
```

```
-rw-r--r-- 1 root root 79 1. Dez 2012 blacklist.conf
```

```
-rw-r--r-- 1 root root 1 2. Mär 14:30 encrypt.conf
```

```
-rw-r--r-- 1 root root 3 5. Dez 2012 printing.conf
```

```
-rw-r--r-- 1 root root 6 14. Jul 11:01 realtek.conf
```

```
-rw-r--r-- 1 root root 65 2. Jun 23:01 virtualbox.conf
```

```
...
```

**5.** The Failed to find module 'blacklist usbhp' error message might be related to a wrong setting inside of `blacklist.conf`. Lets deactivate it with inserting a trailing `#` before each option we found via step 3:

```
/etc/modules-load.d/blacklist.conf
```

```
# blacklist usbhp
```

```
# install usbhp /bin/false
```

**6.** Now, try to start `systemd-modules-load`:

```
$ systemctl start systemd-modules-load
```

If it was successful, this should not prompt anything. If you see any error, go back to step 3 and use the new PID for solving the errors left.

If everything is ok, you can verify that the service was started successfully with:

```
$ systemctl status systemd-modules-load
```

```
systemd-modules-load.service - Load Kernel Modules
```

```
Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
```

```
Active: active (exited) since So 2013-08-25 12:22:31 CEST; 34s ago
```

```
Docs: man:systemd-modules-load.service(8)
```

```
man:modules-load.d(5)
```

```
Process: 19005 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited, status=0/SUCCESS)
```

```
Aug 25 12:22:31 mypc systemd[1]: Started Load Kernel Modules.
```

Often you can solve these kind of problems like shown above. For further investigation look at [#Diagnosing boot problems](#).

## 诊断启动问题

使用如下内核参数引导：`systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M`

更多有关调试的信息，参见该文 (<http://freedesktop.org/wiki/Software/systemd/Debugging>)。

## Diagnosing problems with a specific service

If some *systemd* service misbehaves and you want to get more information about what is going on, set the `SYSTEMD_LOG_LEVEL` environment variable to `debug`. For example, to run the *systemd-networkd* daemon in debug mode:

```
# systemctl stop systemd-networkd
# SYSTEMD_LOG_LEVEL=debug /lib/systemd/systemd-networkd
```

Or, equivalently, modify the service file temporarily for gathering enough output. For example:

```
/lib/systemd/system/systemd-networkd.service
[Service]
...
Environment=SYSTEMD_LOG_LEVEL=debug
....
```

If debug information is required long-term, add the variable the regular way.

## 关机/重启十分缓慢

如果关机特别慢（甚至跟死机了一样），很可能是某个拒不退出的服务在作怪。`systemd` 会等待一段时间，然后再尝试杀死它。请阅读这篇文章 ([http://freedesktop.org/wiki/Software/systemd/Debugging#Shutdown\\_Completes\\_Eventually](http://freedesktop.org/wiki/Software/systemd/Debugging#Shutdown_Completes_Eventually))，确认你是否是该问题受害者。

## 短时进程无日志记录

若 `journalctl -u foounit.service` 没有显示某个短时进程的任何输出，那么改用 `PID` 试试。例如，若 `systemd-modules-load.service` 执行失败，那么先用 `systemctl status systemd-modules-load` 查询其 `PID`（比如是123），然后检索该 `PID` 相关的

日志 `journalctl -b _PID=123`。运行时进程的日志元数据（诸如 `_SYSTEMD_UNIT` 和 `_COMM`）被乱序收集在 `/proc` 目录。要修复该问题，必须修改内核，使其通过套接字连接来提供上述数据，该过程类似于 `SCM_CREDENTIALS`。

## 禁止在程序崩溃时转储内存

要使用老的内核转储，创建下面文件：

```
-----  
/etc/sysctl.d/49-coredump.conf  
-----  
kernel.core_pattern = core  
kernel.core_uses_pid = 0  
-----
```

然后运行：

```
-----  
# /usr/lib/systemd/systemd-sysctl  
-----
```

同样可能需要执行“unlimit”设置文件大小：

```
-----  
$ ulimit -c unlimited  
-----
```

更多信息请参阅 `sysctl.d` (<http://www.freedesktop.org/software/systemd/man/sysctl.d.html>) 和 `/proc/sys/kernel` 文档 (<https://www.kernel.org/doc/Documentation/sysctl/kernel.txt>)。

## Error message on reboot or shutdown

**cgroup : option or name mismatch, new: 0x0 "", old: 0x4 "systemd"**

See this thread (<https://bbs.archlinux.org/viewtopic.php?pid=1372562#p1372562>) for an explanation.

**watchdog watchdog0: watchdog did not stop!**

See this thread (<https://bbs.archlinux.org/viewtopic.php?pid=1372562#p1372562>) for an explanation.

## Boot time increasing over time

After using `systemd-analyze` a number of users have noticed that their boot time has increased significantly in comparison with what it used to be. After using `systemd-analyze blame` NetworkManager is being reported as taking an unusually large amount of time to start.



The problem for some users has been due to `/var/log/journal` becoming too large. This may have other impacts on performance, such as for `systemctl status` or `journalctl`. As such the solution is to remove every file within the folder (ideally making a backup of it somewhere, at least temporarily) and then setting a journal file size limit as described in `#Journal size limit`.

## 相关资源

- 官方网站 (<http://www.freedesktop.org/wiki/Software/systemd>)
- 维基百科上的介绍
- man 手册页 (<http://0pointer.de/public/systemd-man/>)
- systemd 优化 (<http://freedesktop.org/wiki/Software/systemd/Optimizations>)
- 常见问题 FAQ (<http://www.freedesktop.org/wiki/Software/systemd/FrequentlyAskedQuestions>)
- 小技巧 (<http://www.freedesktop.org/wiki/Software/systemd/TipsAndTricks>)
- systemd for Administrators (PDF) (<http://0pointer.de/public/systemd-ebook-psankar.pdf>)
- Fedora 项目对 systemd 的介绍 (<http://fedoraproject.org/wiki/Systemd>)
- 如何调试 systemd 故障 ([http://fedoraproject.org/wiki/How\\_to\\_debug\\_Systemd\\_problems](http://fedoraproject.org/wiki/How_to_debug_Systemd_problems))
- *The H Open* 杂志上的两篇 (<http://www.h-online.com/open/features/Control-Centre-The-systemd-Linux-init-system-1565543.html>) 篇 (<http://www.h-online.com/open/features/Booting-up-Tools-and-tips-for-systemd-1570630.html>) 科普文章
- Lennart 的博文 (<http://0pointer.de/blog/projects/systemd.html>)
- 更新报告 (<http://0pointer.de/blog/projects/systemd-update.html>)
- 更新报告2 (<http://0pointer.de/blog/projects/systemd-update-2.html>)
- 更新报告3 (<http://0pointer.de/blog/projects/systemd-update-3.html>)
- 最新动态 (<http://0pointer.de/blog/projects/why.html>)
- Fedora Wiki 上的 SysVinit 和 systemd 命令对比表 ([https://fedoraproject.org/wiki/SysVinit\\_to\\_Systemd\\_Cheatsheet/zh](https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet/zh))

Retrieved from "[https://wiki.archlinux.org/index.php?title=Systemd\\_\(简体中文\)&oldid=401801](https://wiki.archlinux.org/index.php?title=Systemd_(简体中文)&oldid=401801)"

Categories: Daemons and system services (简体中文) | Boot process (简体中文)

- 
- This page was last modified on 26 September 2015, at 04:41.
  - Content is available under GNU Free Documentation License 1.3 or later unless otherwise noted.