SUSE

使用 systemd 计时器

解释

从定期运行备份脚本，到计算机引导后立即启动特定进程，Linux 系统上有大量的任务需要调度。systemd 计时器提供了一个灵活的机制用于调度及管理作业和服务。

原因

本文旨在提供 systemd 计时器的完整概述，涵盖创建、维护、测试、查错以及从 cron 迁移等方面的信息。

工作量

创建示例 systemd 计时器需要 10 分钟。最多花费 30 分钟即可完全了解 systemd 计时器的工作原理。

要求

- 基本了解 systemd。
- root 或 sudo 特权。要以普通用户的身份使用 systemd 计时器，请先参见第 7 节 “以普通用户身份使用计时器”。

出版日期：2025 年 12 月 11 日

目录

1 systemd 计时器概念

systemd 计时器单元提供一个用于在 Linux 上调度作业的机制。这些作业的执行时间可以基于时间和日期或基于事件。

systemd 计时器单元通过 .timer 文件扩展名标识。每个计时器文件都需要一个由它控制的相应服务文件。也就是说，计时器文件将激活并管理相应的服务文件。systemd 计时器支持以下功能：

- 使用计时器单元安排的作业可以依赖其他 systemd 服务。计时器单元被视为常规 systemd 服务，因此可以使用 **systemctl** 进行管理。
- 计时器可以是实时的（基于日历事件触发），也可以是单调性的（自特定起始时刻经过指定的时间后触发）。
- 时间单元将记录到系统日记中，这样用户便能更轻松地对时间单元进行监控和查错。
- 计时器使用集中式 systemd 管理服务。
- 如果系统在预期执行期间关闭，则一旦系统再次运行，就会执行计时器。

2 创建计时器

以下示例说明如何设置下面这样的计时器：在引导后触发 helloworld.sh 外壳脚本，并相对于其激活时间每隔 24 小时重复执行一次。此外，计时器会在星期一至星期五上午 10 点运行。

2.1 Hello World 示例

1. 创建包含以下内容的 /etc/systemd/system/helloworld.service 文件：

```
[Unit]
Description="Hello World script"
```

```
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

这是一个 `systemd` 服务文件，告知 `systemd` 要运行哪个应用程序。

2. 创建包含以下内容的 `/etc/systemd/system/helloworld.timer` 文件：

```
[Unit]
Description="Run helloworld.service 5min after boot and every 24 hours
 relative to activation time"

[Timer]
OnBootSec=5min
OnUnitActiveSec=24h
OnCalendar=Mon..Fri *-*-* 10:00:*
Unit=helloworld.service

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

这是计时器文件，用于控制相应服务文件的激活。

3. 校验您先前创建的文件是否不含错误：

```
> systemd-analyze verify /etc/systemd/system/helloworld.*
```

如果命令未返回任何输出，则表示文件成功通过了校验。

4. 启动计时器：

```
> sudo systemctl start helloworld.timer
```

仅针对当前会话激活计时器。

5. 启用计时器以确保在引导时将其激活：

```
> sudo systemctl enable helloworld.timer
```

2.2 示例解释

例 1： **服务文件**

```
[Unit]
Description="Hello World script" ❶

[Service]
ExecStart=/usr/local/bin/helloworld.sh ❷
```

❶ 解释服务文件用途的简短说明。

❷ 要执行的应用程序。

要使服务文件能够正常运行，至少需要指定 `[Unit]` 和 `[Service]` 部分。`systemd` 服务文件通常包含 `[Install]` 部分，该部分用于确定要加载的一个或多个服务目标。计时器的服务文件中不需要此部分，因为此信息已在计时器文件中提供。有关高级配置，请参见Managing `systemd` targets with systemctl (https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html)。

例 2： **计时器文件**

```
[Unit]
Description="Run helloworld.service 5min after boot and every 24 hours relative
 to activation time" ❶

[Timer]
OnBootSec=5min ❷
OnUnitActiveSec=24h ❸
OnCalendar=Mon..Fri *-*-* 10:00:* ❹
Unit=helloworld.service ❺

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

❶ 解释计时器文件用途的简短说明。

❷ 指定一个用于在系统引导 5 分钟后触发服务的计时器。有关详细信息，请参见单调计时器。

❸ 指定一个用于在激活服务 24 小时后触发服务（即每天触发一次）的计时器。有关详细信息，请参见实时计时器。

❹ 指定一个在固定时间点（在本示例中为星期一至星期五上午 10 点）触发服务的计时器。有关详细信息，请参见实时计时器。

❺ 要执行的服务文件。

❻ 要在其中激活计时器的 `systemd` 目标。有关 `systemd` 目标的详细信息，请参见 Managing `systemd` targets with systemctl (https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html)。

3 管理计时器

可以使用 **`systemctl`** 命令管理计时器。

启动和停止计时器

```
> sudo systemctl start TIMER.timer
> sudo systemctl restart TIMER.timer
> sudo systemctl stop TIMER.timer
```

启用和禁用计时器

```
> sudo systemctl enable TIMER.timer
> sudo systemctl disable TIMER.timer
```

显示计时器文件内容

```
> sudo systemctl cat TIMER.timer
```

检查特定的计时器

```
> sudo systemctl status TIMER.timer
```

例 3： **计时器状态**

```
> sudo systemctl status helloworld.timer
● helloworld.timer - "Run helloworld.service 5min after boot and every 24
 hours
```

```
relative to activation time"❶
Loaded: loaded (/etc/systemd/system/helloworld.timer; disabled; vendor
 preset: disabled)❷
Active: active (waiting) since Tue 2022-10-26 18:35:41 CEST; 6s ago❸
Trigger: Wed 2022-10-27 18:35:41 CEST; 23h left❹
Triggers: ● helloworld.service❺
❻
Oct 26 18:35:41 neo systemd[1]: Started "Run helloworld.service 5min after
 boot and every 24 hours relative to activation time".❼
```

❶ 计时器的文件名和说明。

❷ 列出计时器是否已成功分析并保留在内存中（已加载），显示计时器文件的完整路径，以及计时器在系统引导时会启动（已启用）还是不会启动（已禁用）。第一个值显示当前系统配置，第二个值显示供应商预设的值。

❸ 指示计时器是处于活动状态（正在等待触发事件）还是非活动状态。如果处于活动状态，它还会显示自上次激活以来经过的时间（在本示例中为 6 秒）。

❹ 下次触发计时器的日期和时间。

❺ 计时器触发的服务文件的名称。

❻ 指向文档（例如手册页）的可选行。如果不可用，则显示空行（如本示例中所示）。

❼ 计时器创建的最新日记项。

要列出系统上所有可用的计时器，请使用 **`systemctl list-timers`**。下列选项可用：

列出所有活动计时器：

```
> sudo systemctl list-timers
```

列出所有计时器，包括非活动的计时器：

```
> sudo systemctl list-timers --all
```

列出与模式匹配的所有计时器：

```
> sudo systemctl list-timers PATTERN
> sudo systemctl list-timers --allPATTERN
```

`PATTERN` 必须是一个名称或外壳通配表达式。可以使用运算符 `*`、`?` 和 `[]`。有关通配模式的详细信息，请参见 **`man 7 glob`**。

列出与特定状态匹配的计时器：

```
> sudo systemctl list-timers --state=STATE
```

STATE 接受以下值：active、failed、load、sub。有关详细信息，请参见**man systemctl**。

例 4： **列出计时器**

运行任何 **systemctl list-timers** 都会生成如下所示的表。此示例列出了与模式 snapper* 匹配的所有活动计时器：

```
> sudo systemctl list-timers snapper*
NEXT ❶                         LEFT ❷       LAST ❸
 PASSED ❹    UNIT ❺                   ACTIVATES ❻

-------------------------------------------------------------------------------
Tue 2022-10-26 19:00:00 CEST 39min left Tue 2022-10-26 18:00:29 CEST 19min
 ago snapper-timeline.timer snapper-timeline.service
Wed 2022-10-27 08:33:04 CEST 14h   left Tue 2022-10-26 08:33:04 CEST 9h ago
    snapper-cleanup.timer  snapper-cleanup.service
```

❶ 下次运行计时器的时间点。

❷ 距离下次运行计时器的剩余时间。

❸ 上次运行计时器的时间点。

❹ 自上次运行计时器以来经过的时间。

❺ 计时器单元的名称。

❻ 计时器激活的服务的名称。

4 计时器类型

systemd 支持两种类型的计时器：实时（基于日历）和单调（基于事件）。尽管计时器通常是永久性的，但 systemd 还允许设置仅对当前会话有效的瞬态计时器。

实时计时器

实时计时器由日历事件触发。它们是使用选项 OnCalendar 定义的。

您可以基于日期和时间指定何时触发事件。使用以下模板：

```
OnCalendar=DayOfWeek❶ Year-Month-Day❷ Hour:Minute:Second❸
```

❶ 星期日期。可能的值为 `Sun`、`Mon`、`Tue`、`Wed`、`Thu`、`Fri`、`Sat`。留空会忽略星期日期。

❷ 日期。用两位数指定月和日，用四位数指定年。可以用通配符 `*` 替换每个值，以匹配每个出现的值。

❸ 时间。用两位数指定每个值。可以用通配符 `*` 替换每个值，以匹配每个出现的值。

适用于所有值：用两个点定义连续范围 (`Mon..Fri`)。用逗号分隔各个不同值的列表 (`Mon,Wed,Fri`)。

例 5： 实时计时器示例

- 每个星期五的下午 6 点：

```
OnCalendar=Fri *-*-* 18:00:00
```

- 每天上午 5 点：

```
OnCalendar=Mon..Sun *-*-* 5:00:00
```

- 星期日和星期二凌晨 1 点和 3 点：

```
OnCalendar=Tue,Sun *-*-* 01,03:00:00
```

- 单个日期：

```
OnCalendar=Mo..Sun 2023-09-23 00:00:01
```

- 要指定不同时间的触发器，可以在单个计时器文件中创建多个 OnCalendar 项：

```
OnCalendar=Mon..Fri *-*-* 10:00
OnCalendar=Sat,Sun *-*-* 22:00
```

有关可用功能和选项的完整列表，请参见 **`man 7 systemd.time`**，其中提供了有关以下主题的其他信息：

- 缩短语法并使用缩写
- 指定重复次数
- 查找特定的月份日期（月份的最后一日、最后一个星期日等）
- 应用时区

单调计时器

单调计时器在发生特定事件（例如系统引导或系统单元激活事件）后经过指定的时间时触发。值按时间单位（分钟、小时、日、月、年等）定义。支持以下单位：`usec`、`msec`、`seconds`、`minutes`、`hours`、`days`、`weeks`、`months`、`years`。可以使用多个选项来定义单调计时器：

- `OnActiveSec`：激活单元后经过的时间

```
OnActiveSec=50minutes
```

- `OnBootSec`：系统引导后经过的时间

```
OnBootSec=10hours
```

- `OnStartupSec`：启动服务管理器后经过的时间。对于系统服务，此选项大致相当于 `OnActiveSec`。请将此选项用于相应服务管理器会在用户登录时启动的用户服务。

```
OnStartupSec=5minutes 20seconds
```

- `OnUnitActiveSec`：上次激活相应服务后经过的时间

```
OnUnitActiveSec=10seconds
```

- `OnUnitInactiveSec`：上次停用相应服务后经过的时间

```
OnUnitInactiveSec=2hours 15minutes 18 seconds
```

瞬态计时器

瞬态计时器是仅对当前会话有效的临时计时器。借助这些计时器，可以使用现有的服务文件或直接启动程序。可以运行 **`systemd-run`** 来调用瞬态计时器。

以下示例每隔两小时运行一次 `helloworld.service` 单元：

```
> sudo systemd-run --on-active="2hours" --unit="helloworld.service"
```

要直接运行某个命令，请使用以下语法。此示例直接调用脚本 `/usr/local/bin/helloworld.sh`：

```
> sudo systemd-run --on-active="2hours" /usr/local/bin/helloworld.sh
```

如果命令接受参数，请用空格分隔添加的参数：

```
> sudo systemd-run --on-active="2hours" /usr/local/bin/helloworld.sh --language=pt_BR
```

瞬态计时器可以是单调的，也可以是实时的。支持以下开关，其工作方式如单调计时器中所述：

- `--on-active`
- `--on-startup`
- `--on-unit-active`
- `--on-unit-inactive`
- `--on-calendar`

有关详细信息，请参见 **`man 1 systemd-run`**。

5 测试日历项

`systemd` 提供了用于为实时计时器测试和创建日历计时器项的工具，即 **`systemd-analyze calendar`**。此工具接受的参数与用于设置实时计时器的 `OnCalendar` 项相同。

您可以串联多个参数（用空格分隔）。如果要测试的字词正确，则输出会显示下次触发计时器的时间（以本地时间和 UTC 表示）。其中还会显示`Normalized form`的字符串，建议在计时器文件中使用该字符串。考虑下列示例：

```
> systemd-analyze calendar "Tue,Sun *-*-* 01,03:00:00"
```

```
Normalized form: Tue,Sun *-*-* 01,03:00:00
Next elapse: Sun 2021-10-31 01:00:00 CEST
(in UTC): Sat 2021-10-30 23:00:00 UTC
From now: 3 days left

> systemd-analyze calendar "Mon..Fri *-*-* 10:00" "Sat,Sun *-*-* 22:00"
Original form: Mon..Fri *-*-* 10:00
Normalized form: Mon..Fri *-*-* 10:00:00
Next elapse: Thu 2021-10-28 10:00:00 CEST
(in UTC): Thu 2021-10-28 08:00:00 UTC
From now: 19h left

Original form: Sat,Sun *-*-* 22:00
Normalized form: Sat,Sun *-*-* 22:00:00
Next elapse: Sat 2021-10-30 22:00:00 CEST
(in UTC): Sat 2021-10-30 20:00:00 UTC
From now: 3 days left
```

对于重复性计时器，请使用 `—iterations N` 开关列出触发时间，然后测试计时器是否按预期工作。参数 `N` 指定您要测试的迭代次数。以下示例字符串在星期日每隔 8 小时（从 00:00:00 开始）触发一次计时器：

```
> systemd-analyze calendar --iterations 5 "Sun *-*-* 0/08:00:00"
Original form: Sun *-*-* 0/08:00:00
Normalized form: Sun *-*-* 00/8:00:00
Next elapse: Sun 2021-10-31 00:00:00 CEST
(in UTC): Sat 2021-10-30 22:00:00 UTC
From now: 3 days left
Iter. #2: Sun 2021-10-31 08:00:00 CET
(in UTC): Sun 2021-10-31 07:00:00 UTC
From now: 3 days left
Iter. #3: Sun 2021-10-31 16:00:00 CET
(in UTC): Sun 2021-10-31 15:00:00 UTC
From now: 4 days left
Iter. #4: Sun 2021-11-07 00:00:00 CET
(in UTC): Sat 2021-11-06 23:00:00 UTC
From now: 1 week 3 days left
Iter. #5: Sun 2021-11-07 08:00:00 CET
(in UTC): Sun 2021-11-07 07:00:00 UTC
```

```
From now: 1 week 3 days left
```

6 当计时器失败时接收电子邮件通知

systemd 未提供与 cron 的 MAILTO 类似的功能。以下过程介绍了当计时器失败时启用电子邮件通知的解决方法。

该过程包括以下步骤：

1. 创建一个用于发送电子邮件的脚本。
2. 创建运行电子邮件脚本的 systemd 服务文件。
3. 测试电子邮件服务文件。
4. 在计时器控制的服务中，通过 OnFailure 调用创建的电子邮件服务文件。

> 以下示例将使用软件包 mailx 中的 **mailx** 命令。这需要安装并正确配置 Postfix 电子邮件服务器。
>
> 1. 创建脚本 /usr/local/bin/send_systemd_email。
>
>   a. 该脚本需要两个参数：$1（电子邮件地址）和 $2（收到的失败通知所对应服务文件的名称）。这两个参数均由运行邮件脚本的单元文件提供。
>
>    ```
>    #!/bin/sh
>    systemctl status --full "$2" | mailx -S sendwait\
>     -s "Service failure for $2" -r root@$HOSTNAME $1
>    ```
>
>   b. 确保该脚本可执行：
>
>    ```
>    > sudo chmod 755 /usr/local/bin/send_systemd_email
>    ```
>
> 2. 创建文件 /etc/systemd/system/send_email_to_USER@.service。
>
>    ```
>    [Unit]
>    Description=Send systemd status information by email for %i to USER
>    ```

```
[Service]
Type=oneshot
ExecStart=/usr/local/bin/send_systemd_email EMAIL_ADDRESS %i
User=root
Group=systemd-journal
```

将文件中的 `USER` 和 `EMAIL_ADDRESS` 分别替换为要接收电子邮件的用户的登录名和电子邮件地址。`%i` 是失败的服务的名称（它将通过 `%n` 参数传递给电子邮件服务）。

3. 校验服务文件并修复报告的问题：

```
> systemd-analyze verify /etc/systemd/system/send_email_to_USER@.service
```

如果命令未返回任何输出，则表示文件成功通过了校验。

4. 要校验整个过程，请使用 `dbus` 实例启动服务进行测试。（可以使用当前正在运行的任何其他服务。本示例之所以使用 dbus，是因为该服务肯定可以在任何安装的系统上运行。）

```
> sudo systemctl start send_email_to_USER@dbus.service
```

如果成功，`EMAIL_ADDRESS` 会收到一封电子邮件，其主题为 `Service failure for dbus`，正文包含 dbus 状态消息。（这只是一项测试，dbus 服务本身并未出现问题。您可以放心删除该电子邮件，无需执行任何操作）。
如果测试电子邮件已成功发送，请将其集成到您的服务文件中。

5. 要向服务添加电子邮件通知，请将 `OnFailure` 选项添加到在发生失败时用于接收通知的服务文件的 `Unit` 部分：

```
[Unit]
Description="Hello World script"
OnFailure❶=send_email_to_USER❷@%n❸.service

[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

❶ `OnFailure` 选项接受将某个服务作为参数。

❷ 将此服务单元文件名部分替换为登录名。

❸ 指定服务的名称（在本示例中为 `helloworld`）。此名称在电子邮件服务文件中以 %i 形式提供。

您已成功为 `systemd` 服务设置失败通知。

提示：向多个用户发送电子邮件通知

电子邮件服务文件已将收件人的电子邮件地址硬编码。要向其他用户发送通知电子邮件，请复制电子邮件服务文件，并替换文件名中的用户登录名，以及抄送行中的电子邮件地址。

要同时向多个收件人发送失败通知，请将相应的服务文件添加到该服务文件（使用空格作为分隔符）：

```
OnFailure=send_email_to_tux@%n.service send_email_to_wilber@%n.service
```

7 以普通用户身份使用计时器

普通用户也可以使用 `systemd` 计时器。这些计时器可帮助您自动完成重复性任务，例如备份、处理图像或将数据移到云中。

适用于系统范围计时器的过程和任务同样适用于 systemd 计时器。但是，两者存在以下差异：

- 计时器和服务文件必须放在 `~/.config/systemd/user/` 中。
- 必须结合 `--user` 开关运行所有 **`systemctl`** 和 **`journalctl`** 命令。**`systemd-analyze`** **不**需要此选项。
  作为普通用户，您必须提供单元文件的路径，如以下示例中所示。否则，如果存在同名的系统范围计时器，则会执行或列出该系统范围计时器。

```
> systemctl --user start ~/.config/systemd/user/helloworld.timer
> systemctl --user enable ~/.config/systemd/user/helloworld.timer
> systemctl --user list-timers
> journalctl --user -u helloworld.*
> systemd-analyze verify ~/.config/systemd/user/helloworld.timer
```

重要：用户计时器仅在活动会话期间运行

与以普通用户身份启动的其他 systemd 服务一样，用户计时器仅在用户登录后才运行。要在引导时启动用户计时器并在注销后使其保持运行，请为每个受影响的用户启用**存留**：

```
sudo loginctl enable-linger USER
```

有关详细信息，请参见 **man 1 loginctl**。

重要：不继承环境变量

systemd 用户实例不会继承 ~/.profile 或 ~/.bashrc 等脚本设置的环境变量。要检查 systemd 环境，请运行 **systemctl --user show-environment**。

要导入 systemd 环境中缺少的任何变量，请在 ~/.bashrc 末尾指定以下命令：

```
systemctl --user import-environment VARIABLE1 VARIABLE2
```

8 从 cron 迁移到 systemd 计时器

所有 cron 作业都可以迁移到 systemd 计时器。下面提供了说明和示例。

1. 创建执行脚本的服务文件。有关详细信息，请参见例 1 “服务文件”。
2. 创建执行服务文件的计时器文件。有关一般说明，请参见例 2 “计时器文件”。
   a. 转换日历项。cron 和 systemd 中指定时间的方式不相同。使用以下模式作为转换模板：

   ```
   Cron:                Minute Hour Day Month DayOfWeek
   systemd: OnCalendar=DayOfWeek Year-Month-Day Hour:Minute:Second
   ```

   要测试转换后的日历项，请按照第 5 节 “测试日历项”中的说明操作。

   b. 转换 cron 别名 (@NICK)：

```
Cron     : systemd timer
-------- : ---------------------------
@reboot  : OnBootSec=1s
@yearly  : OnCalendar=*-01-01 00:00:00
@annually: OnCalendar=*-01-01 00:00:00
@monthly : OnCalendar=*-*-01 00:00:00
@weekly  : OnCalendar=Sun *-*-* 00:00:00
@daily   : OnCalendar=*-*-* 00:00:00
@hourly  : OnCalendar=*-*-* *:00:00
```

c. 转换变量赋值。`systemd` 变量赋值必须在 `[Service]` 部分中定义。不能以这种方式转换 `MAILTO` - 具体请参见下一步。

```
cron: VARIABLE=VALUE
systemd: Environment="VARIABLE=VALUE"
```

d. 按照第 6 节 “当计时器失败时接收电子邮件通知”中的说明设置电子邮件通知，以替换 cron 的 MAILTO 功能。

例 6： 从 CRON 迁移到 `systemd` 计时器

下面是在引导后经过 5 分钟并在每个星期一至星期五 10 点调用脚本 `helloworld.sh` 的 crontab 项：

```
@reboot sleep 300 && /usr/local/bin/helloworld.sh
0 10 * * * 1-5 /usr/local/bin/helloworld.sh
```

调用脚本的 `systemd` 服务文件 (`helloworld.service`) 如下所示：

```
[Unit]
Description="Hello World script"
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

计时器文件 (`helloworld.timer`) 如下所示：

```
[Unit]
Description="Run helloworld.service 5min after boot and at 10am every Mon-Fri"
[Timer]
```

```
OnBootSec=5min
OnCalendar=Mon..Fri *-*-* 10:00:*
Unit=helloworld.service
[Install]
WantedBy=multi-user.target
```

9 查错和常见问题

了解如何对失败的 systemd 计时器进行调试和查错。查找有关 systemd 计时器的常见问题解答。

9.1 避免错误

为了避免 systemd 计时器出现错误，请确保遵循以下最佳实践：

- 校验您在服务中使用 ExecStart 指定的可执行文件是否正确运行。
- 运行 **systemd-analyze verify** FILE 来检查服务和计时器文件的语法。
- 运行 **systemd-analyze calendar** CALENDER_ENTRY 来检查日历项的执行时间。

9.2 事件未触发

当您激活包含非严重错误的计时器时，systemd 将静默忽略这些错误。例如：

例 7： 包含非致命错误的 systemd 计时器文件中断

```
[Timer]
OnBootSec=5min
OnClendar=Mon..Fri 10:00
Unit=helloworld.service
```

第 3 行包含语法错误（应该是 OnClendar，而不是 OnCalendar）。由于 [Timer] 部分包含另一个计时器项 (OnBoot)，因此该错误并不严重，将被静默忽略。因此，星期一至星期五的触发器不会执行。检测错误的唯一方法是使用命令 **systemd-analyze verify**：

```
#  systemd-analyze verify /etc/systemd/system/helloworld.timer
/etc/systemd/system/helloworld.timer:7: Unknown key name 'OnClendar' in
 section 'Timer', ignoring.
```

9.3 在系统日记中检查错误

与每项 systemd 服务一样，计时器触发的事件和操作会记录到系统日记中。如果触发器未按预期运行，请使用 **journalctl** 检查日志消息。要过滤日记以显示相关信息，请使用 -u 开关指定 systemd 计时器和服务文件。使用此选项可显示计时器**和**相应服务文件的日志项：

```
sudo journalctl -u  helloworld.timer -u helloworld.service
```

或使用更短的选项格式（如果适用）：

```
sudo journalctl -u  helloworld.*
```

journalctl 是支持许多选项和过滤器的工具。请参见 **man 1 journalctl** 获取深入信息。以下选项对于计时器查错非常有用：

- -b：仅显示当前引导对应的项。
- -S today：仅显示当日的项。
- -x：显示帮助文本以及日志项。
- -f：从最近的项开始，随着新项的不断添加持续列显日志。非常适合用于检查以较短间隔执行的触发器。按 Ctrl – C 退出。

9.4 systemd 计时器：弥补错过的轮次

如果 systemd 计时器处于非活动状态或系统在预期执行期间关闭，可以选择性地在计时器再次激活后立即触发错过的事件。要启用此功能，请将配置选项 Persistent=true 添加到 [Timer] 部分：

```
[Timer]
OnCalendar=Mon..Fri 10:00
Persistent=true
```

```
Unit=helloworld.service
```

9.5 如何从 cron 迁移到 systemd 计时器?

所有 cron 作业都可以迁移到 systemd 计时器。下面是有关迁移 cron 作业的一般说明：

1. 创建执行脚本的服务文件。有关详细信息，请参见例 1 “服务文件”。
2. 创建执行服务文件的计时器文件。有关一般说明，请参见例 2 “计时器文件”。
    a. 转换日历项。cron 和 systemd 中指定时间的方式不相同。使用以下模式作为转换模板：

    ```
    Cron:                Minute Hour Day Month DayOfWeek
    systemd: OnCalendar=DayOfWeek Year-Month-Day Hour:Minute:Second
    ```

    要测试转换后的日历项，请按照第 5 节 “测试日历项”中的说明操作。

    b. 转换 cron 别名 (@NICK)：

    ```
    Cron     : systemd timer
    -------- : ---------------------------
    @reboot  : OnBootSec=1s
    @yearly  : OnCalendar=*-01-01 00:00:00
    @annually: OnCalendar=*-01-01 00:00:00
    @monthly : OnCalendar=*-*-01 00:00:00
    @weekly  : OnCalendar=Sun *-*-* 00:00:00
    @daily   : OnCalendar=*-*-* 00:00:00
    @hourly  : OnCalendar=*-*-* *:00:00
    ```

    c. 转换变量赋值。systemd 变量赋值必须在 [Service] 部分中定义。不能以这种方式转换 MAILTO - 具体请参见下一步。

    ```
    cron: VARIABLE=VALUE
    systemd: Environment="VARIABLE=VALUE"
    ```

    d. 按照第 6 节 “当计时器失败时接收电子邮件通知”中的说明设置电子邮件通知，以替换 cron 的 MAILTO 功能。

例 8： **从 CRON 迁移到 systemd 计时器**

下面是在引导后经过 5 分钟并在每个星期一至星期五 10 点调用脚本 `helloworld.sh` 的 crontab 项：

```
@reboot sleep 300 && /usr/local/bin/helloworld.sh
0 10 * * * 1-5 /usr/local/bin/helloworld.sh
```

调用脚本的 `systemd` 服务文件 (`helloworld.service`) 如下所示：

```
[Unit]
Description="Hello World script"
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

计时器文件 (`helloworld.timer`) 如下所示：

```
[Unit]
Description="Run helloworld.service 5min after boot and at 10am every Mon-
Fri"
[Timer]
OnBootSec=5min
OnCalendar=Mon..Fri *-*-* 10:00:*
Unit=helloworld.service
[Install]
WantedBy=multi-user.target
```

10 更多信息

- 有关 `systemd` 计时器的完整参考，包括高级配置选项（例如延迟或者处理时钟或时区更改），请参见 **`man 5 systemd.timer`**。
- Basic `systemd` concepts (https://documentation.suse.com/smart/systems-management/html/concept-systemd/concept-systemd.html)
- Starting and stopping `systemd` services (https://documentation.suse.com/smart/systems-management/html/reference-systemctl-start-stop-services/reference-systemctl-start-stop-services.html)

- Enabling and disabling `systemd` services (https://documentation.suse.com/smart/systems-management/html/reference-systemctl-enable-disable-services/reference-systemctl-enable-disable-services.html)
- Debugging failed `systemd` services (https://documentation.suse.com/smart/systems-management/html/task-debug-failed-systemd-services/index.html)
- Sending termination signals to `systemd` services (https://documentation.suse.com/smart/systems-management/html/task-send-termination-signals-systemd/task-send-termination-signals-systemd.html)

11 法律声明

A GNU 自由文档许可证

0. 导言

此许可证的目的是赋予手册、教科书或其他功能性的和有用的文档以“自由”：即保证每个人都有复制和再分发此类文档的有效自由，无论是否进行修改，也无论将其用于商业或非商业用途。其次，此许可证为作者和出版者保留了因工作获得声誉但不视为对他人所做修改负责的方式。

本许可证是一种“非盈利版权”，这意味着从该文档衍生的作品也必须是以同一方式自由的。它补充了 GNU 通用公共许可证（为自由软件设计的非盈利版权许可证）。

我们设计此许可证旨在将其用于免费软件的手册，因为免费软件需要自由文档：免费程序所附手册应具有与软件本身同样的自由。但是此许可证不限于软件手册；它可以用于任何文本作品，无论主题如何或它是否作为印刷书籍出版。建议本许可证主要用于目的是指导或参考的作品。

1. 适用性和定义

此许可证适用的对象：由版权所有者在其中明确声明可按照此许可证条款以任何媒体分发的任何手册和其他作品。此类声明授予在此处所述的条款和条件下使用该作品的全球无限期无版权许可证。下述“文档”指任何此类手册或作品。任何公众成员都是一个被许可人，以下称为“您”。如果您以需要版权法许可的任何方式复制、修改或分发该作品，则表示您接受该许可证。

该文档的“修改版本”表示包含该文档或其一部分（或者逐字复制或者有修改和/或翻译为另一语言）的任何作品。

“次要章节”是该文档的命名附录或扉页章节，专门讲述该文档的出版者或作者与该文档整个主题（或相关问题）的关系，不包含与整个主题相关的内容。（因此，如果该文档是数学课本的一部分，则辅助部分可能不说明任何数学问题。）这种关系可以是与主题或相关问题的历史联系，或与它们相关的法律、商业、哲学、伦理或政治地位。

在该文档基于此许可证项发布的声明中，“固定章节”是将其标题指定为固定章节标题的一些辅助章节。如果一个章节不适用上述辅助章节的定义，则不允许将其指定为固定章节。该文档可能不包含固定章节。如果该文档不标识任何固定章节，则表示没有固定章节。

在该文档基于此许可证项发布的声明中，“封页文本”是作为封面文本或封底文本列出的简短文本段落。封面文本最多 5 个单词，封底文本最多 25 个单词。

文档的“透明”副本是一个机器可读的副本，使用公众可以得到其规范的格式表达，这样的副本适合于使用通用文本编辑器、（对于像素构成的图像）通用绘图程序、（对于绘制的图形）广泛使用的绘画编辑器直接修改文档，也适用于输入到文本格式处理程序或自动翻译成各种适于适用于输入到文本格式处理程序的格式。一个用其他透明文件格式表示的副本，如果该格式的标记（或缺少标记）已经构成了对读者的后续修改的障碍，那么就是不透明的。表示实质性数量的文本的图像格式都是不透明的。不“透明”的副本称为“不透明”。

适于作为透明副本的格式的示例有：没有标记的纯 ASCII 文本、Texinfo 输入格式、LaTeX 输入格式、使用公共可用 DTD 的 SGML 或 XML，符合标准的简单 HTML、可以人为修改的 PostScript 或 PDF。透明图像格式的示例有 PNG、XCF 和 JPG。不透明的格式包括：仅可以被私有版权的字处理软件使用的私有版权格式、所用的 DTD 和/或处理工具不是广泛可用的的 SGML 或 XML、机器生成的 HTML、一些字处理器生成的只用于输出目的的 PostScript 或 PDF。

对于印刷书籍，“扉页”就是扉页本身以及随后的一些用于补充的页，显然本许可资料需要出现在扉页上。对于那些没有扉页的作品形式，“扉页”代表接近作品最突出标题的、在文本正文之前的文本。

“命名为 XYZ”的章节表示文档的一个特定的子单元，其标题就是 XYZ 或在括号中包含 XYZ 且后跟 XYZ 的其他语言翻译文本。（这里 XYZ 代表下面提及的特定章节名称，比如“致谢”、“题献”、“签名”或“历史”。）要在修改文档时对这类章节“保留标题”就是依据此定义保持这样一个“命名为 XYZ”的章节。

文档可能在文档遵照此许可证的声明后面包含免责声明。这些免责声明应作为参考信息包含在此许可证中，但是只能将其视作免责声明：这些免责声明暗指的任何其他含义均无效，且对此许可证的含义不产生任何影响。

2. 逐字复制

您可以用任何媒体复制并分发文档，无论是出于商业还是非商业目的，只要保证此许可证、版权声明和声称此许可证应用于文档的声明都完整地、无任何附加条件地存在于所有副本中。不能使用任何技术手段阻碍或控制您制作或发布的副本的阅读或再次复制。不过您可以在副本交易中得到报酬。如果发布足够多的副本，则您必须遵循下面第三节中的条件。

您也可以在如上的条件下出租副本和向公众放映副本。

3. 大量复制

如果您出版的文档印刷版副本（或是有印制封页的其他媒体副本）多于 100 份，而文档的许可证声明中要求有封页文本，则您必须将它清晰地置于封页之上，封面文本在封面上，封底文本在封底上。封面和封底上还必须标明您是这些副本的出版者。封面必须同等显著地完整展现标题的所有文字。您可以在封页上加入其他资料。改动仅限于封页的复制，只要保持文档的标题不变并满足这些条件，可以在其他方面被视为逐字复制。

如果需要加上的文本对于封面或封底过多，无法明显地表示，您应该在封页上列出前面的（在合理的前提下尽量多），把其他的放在邻近的页面上。

如果您出版或分发了超过 100 份文档的不透明副本，则必须在每个不透明副本中包含一份计算机可读的透明副本，或是在每个不透明副本中给出一个计算机网络地址，通过这个地址，网络公共用户可以使用标准网络协议下载文档的无任何附加资料的完整透明副本。如果您选择后者，则必须在开始大量分发非透明副本的时候采用相当谨慎的步骤，保证透明副本在其所给出的位置在（直接或通过代理和零售商）分发最后一次该版本的非透明副本的时间之后一年之内始终是有效的。

在重新大量发布副本之前，请您（但不是必须）与文档的作者联系，以便他们可以有机会向您提供文档的更新版本。

4. 修改

在上述第 2、3 节的条件下，您可以复制和分发文档的修改版本，前提是严格按照此许可证发布修改后的文档，将修改版本用作文档，从而允许任何拥有此修改版副本的人执行分发或修改。另外，在修改版中，您需要做到如下几点：

A. 用于与文档以及以前各个版本（如果有，应该列在文档的“历史”章节中）显著不同的扉页（和封页，如果有）。如果那个版本的原始发行者允许的话，您可以使用和以前版本相同的标题。

B. 与作者一样，在扉页上列出承担修改版本中的修改的作者责任的一个或多个人或实体和至少五个文档的原作者（如果原作者不足五个就全部列出），除非他们免除了您的这个责任。

C. 与原来的发行者一样，在扉页上列出修改版的发行者的姓名。

D. 保持该文档的全部版权声明不变。

E. 在与其他版权声明邻近的位置加入恰当的针对您的修改的版权声明。

F. 在紧接着版权声明的位置加入许可声明，按照下面附录中给出的形式，以本许可证给公众授于是用修订版本的权利。

G. 保持原文档的许可声明中的全部不可变章节、封面文字和封底文字的声明不变。

H. 包含一份未作任何修改的本协议的副本。

I. 保持命名为“历史”的章节不变，保持它的标题不变，并在其中加入一项，至少声明扉页上的已修改版本的标题、年份、新作者和出版者。如果文档中没有命名为“历史”章节，则请新建它，并加入一项以声明原文档扉页上所列的标题、年份、作者与出版者，再在其后加入如上所说的描述修改版本的项。

J. 如果文档中有用于公共用户访问的文档透明副本的网址，则保持网址不变，并同样提供它所基于的以前文档版本的网址。这些网址可以放在“历史”章节。您可以不给出那些在原文档发行之前已经发行至少四年的版本给出的网址，或者该版本的发行者授权不列出网址。

K. 对于任何命名为“致谢”或“题献”的章节，保持其标题不变，并保持其全部内容以及对每位贡献者致谢和/或题献的语气不变。

L. 保持文档的所有固定章节不变，不改变它们的标题和内容。章节的编号或相当的内容不被认为是章节标题的一部分。

M. 删除命名为“签名”的章节。这样的章节不可以被包含在修改后的版本中。

N. 不要把任何现有章节重命名为“签名”或与任何不可变章节相冲突的标题。

O. 保持任何免责声明不变。

如果修改版本加入了新的符合次要章节定义的引言或附录章节，并且不含有从原文档中复制的内容，您可以选择将其标记为固定。如果需要这样做，则将它们的标题加入修改版本许可声明的不可变章节列表之中。这些标题必须和其他章节的标题相区分。

您可以加入一个命名为“签名”的章节，只要它只包含对您的修改版本由不同的各方给出的签名，例如书评或是声明文本已经被一个组织认定为一个标准的权威定义。

您可以加入一个最多 5 个字的段落作为封面文本和一个最多 25 个字的段落作为封底文本，将它们加入修改版本的封页文本列表末端。一个实体只可以添加（或编排）一段封面和一段封底文本。如果原文档已经为该封页（封面或封底）包含了封页文本，由您或您所代表的实体先前加入或排列的文本，不能再新加入一个，但您可以在原来的发行者的明确许可下替换掉原来的那个。

作者和发行者不能通过本许可证授权公众使用他们的名字推荐或暗示认可任何一个修改版本。

5. 组合文档

遵照第 4 节所说的修改版本的规定，您以将文档和其他文档合并并以本许可证发布，只要您在合并结果中包含原文档的所有不可变章节，对它们不加以任何改动，并在合并结果的许可声明中将它们全部列为不可变章节，而且维持原作者的免责声明不变。

合并作品仅需要包含一份此许可证，多个相同的固定章节可以由一个副本取代。如果有多个名称相同但内容不同的固定章节，通过在章节名称后面的括号中加上原作者或出版者的姓名（如果已知）来加以区别，或者使用唯一编号加以区别。并对合并作品许可声明中的固定章节列表中的章节标题做相同的调整。

在合并过程中，必须合并不同原始文档中任何命名为“历史”的章节，从而形成新的命名为“历史”的章节；类似地，还要合并命名为“致谢”和“题献”的章节。必须删除所有命名为“签名”的章节。

6. 文档的合集

您可以制作一个文档和其他文档的合集，在本许可证下发布，并在合集中将不同文档中的多个本许可证的副本以一个单独的副本来代替，只要您在文档的其他方面遵循本许可证的逐字复制的条款即可。

您可以从一个这样的合集中提取一个单独的文档，并将它在本许可证下单独发布，只要您想这个提取出的文档中加入一份本许可证的副本，并在文档的其他方面遵循本许可证的逐字复制的原则。

7. 独立作品的聚合体

将文档或其派生品以及其他独立和无关文档或作品编撰在一个储存卷中或分发媒体上，这称为文档的“聚合体”，前提是编撰成品的著作权对其使用者的法律权限的限制未超出各个独立作品的许可范围。当基于此许可证发布的文档包含在一个聚合体中时，此许可证不适用于聚合体中的本非该文档派生作品的其他作品。

如果第 3 节中的封页文本要求适用于这些文档的副本，则若文档在聚合体中所占的比重小于全作品的一半，文档的封页文本可以放置在聚合体内包含文档部分的封页上，或是电子文档中的等效部分。否则，它必须位于整个聚合体的印刷的封页上。

8. 翻译

翻译被视为一种修改，因此您可以根据第 4 节的条款分发文档的翻译。将固定章节替换为翻译内容需要经得其版权所有者的特别许可，但除了这些固定章节的原始版本之外，您还可以包含一部分或所有固定章节的翻译。您可以包含一个此许可证以及所有许可证声明和免责声明的翻译版本，前提是同时包含它们的原始英文版本。当翻译版本和英文版发生冲突的时候，原始版本有效。

如果在文档中有命名为“致谢”、“题献”或“历史”的章节，保持标题（第 1 节）的要求（第 4 节）恰恰需要更换实际的标题。

9. 终止

除非此许可证中有明确规定，否则您不能对该文档进行复制、修改、分授许可或分发。在此许可证规定外对该文档所进行的任何复制、修改、分授许可或分发都是无效的，并且将自动终止您在此许可证下所拥有的权利。但是，对于在此许可证的规定下从您这里获得副本或权利的各方，只要其完全遵守此许可证的规定，其许可证将不会被终止。

10. 本许可的未来修订版本

自由软件基金会有时会发布 GNU 自由文档许可证的新的修订版版本。这些新版本的主旨和精神与当前版本是一致的，但在解决新问题的具体细节方面可能有所不同。请参见 https://www.gnu.org/copyleft/。

许可证的每个版本都有一个不同的版本号。如果文档指定了适用于它的此许可证“或任何后续版本”的特定带编号版本，则您可以选择遵从指定版本或自由软件基金会发布的任何随后版本（非草稿）的条款和条件。如果文档没有指定此许可证的版本号，您可以选择自由软件基金会发布的任何许可证版本（非草稿）。

附录：如何针对您的文档使用此许可证

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

如果您有固定章节、封面文本和封底文本，请将“with...Texts”部分替换为：

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

如果有不可变章节而没有封页文本，或这三种内容（不可变章节、封面文本、封底文本）的任何其他组合，请合并这两个备选项以适应您的情况。

如果您的文档包含不一般的程序代码示例，建议同时选择自由软件许可证（如 GNU 通用公共许可证）发布这些示例，以允许它们可以用于自由软件。