SUSE

systemd 基础知识简介

解释

systemd 用于管理系统设置和服务。systemd 将任务组织成称为**单元**的组件，并将单元组组织成**目标**。

原因

了解 systemd 的基础知识，其中包括服务管理、依赖关系跟踪、日志记录、资源管理、套接字激活和系统控制等重要功能。

工作量

读完本文需要 20 分钟。

要求

- 基本了解 Linux 命令
- 基本了解 Linux 进程、守护程序和控制组

出版日期：2025 年 12 月 11 日

目录

1 什么是 systemd?

systemd 是 Linux 操作系统的系统和服务管理器。它是主要 Linux 发行套件的默认初始化系统。systemd 不是由用户直接启动，而是通过 /sbin/init 安装并在早期引导期间启动的。systemd 充当 init 系统，在引导期间作为第一个进程 (PID 1) 运行时，它会启动并维护用户空间服务。PID 1 称为 **init**，是创建的第一个 Linux 用户模式进程。它会一直运行到系统关闭为止。

systemd 拥有 PID 1，并且由内核直接启动。所有其他进程由 systemd 直接启动，或由它的一个子进程启动。systemd 会挂载主机的文件系统并管理临时文件。它与 SysV init 脚本向后兼容。SysV 是比 systemd 更早问世的初始化系统。

在 systemd 中，单元是系统知道如何操作和管理的资源。这是 systemd 工具使用的主要对象。这些资源是通过配置文件（称作单元文件）定义的。

systemctl 是用于控制 init 系统的中心管理工具。它用于检查及控制 systemd 系统和服务管理器的状态。

systemd 中的目标是在系统引导期间充当同步点的相关单元组。目标单元文件的扩展名为 .target。目标单元通过依赖关系链将各种 systemd 单元分组到一起。

要进行查错，您可以使用 **journalctl**，此工具用于查询和显示 systemd 日记中的日志消息。

有关 systemd 的详细信息，请参见 https://systemd.io 和 **man 1 systemd**。

2 关于 systemd 引导过程

引导过程的第一步是加载 Linux 内核，它是 Linux 操作系统的主要组件。加载后，内核会初始化硬件并启动 systemd 进程，这是在系统上运行的第一个进程。

2.1 Linux 引导过程

Linux 引导过程是操作系统启动的初始阶段。在这个过程中，操作系统会加载内存、初始化组件并准备执行用户应用程序。

Linux 引导过程分为四个主要阶段：

第 1 阶段：BIOS

当您打开计算机电源时，您的计算机会启动 BIOS（基本输入/输出系统）并执行 POST（开机自检）。这是一项完整性检查，用于探测硬盘、SSD、键盘、RAM、USB 端口等组件以及任何其他硬件的硬件功能。如果硬件按预期工作，引导过程将进入下一阶段。

第 2 阶段：引导加载程序

开机自检完成后，BIOS 将搜索并加载存储在 MBR（主引导记录）中的引导加载程序。MBR 是 512 个字节组成的代码，通常位于 `/dev/sda` 或 `/dev/hda` 中，具体取决于您的硬盘体系结构。MBR 也可以位于 Linux 的实时 USB 或 DVD 安装上。BIOS 会加载并执行此 MBR 代码。

Linux 中有三个主要的引导加载程序：LILO、GRUB 和 GRUB2。GRUB2（全称 Grand Unified Bootloader）引导加载程序是现代 Linux 发行套件中最新的主要引导加载程序。GRUB2 配置文件位于 `/boot/grub2/grub2.cfg`。BIOS 找到 GRUB2 引导加载程序后，就会执行它并将其加载到主内存 (RAM) 中。

第 3 阶段：Linux 内核初始化

Linux 内核是操作系统的核心。在 Linux 系统中，内核会与硬件交互、控制内存管理并管理进程。引导加载程序会加载选定的 Linux 内核。内核会从压缩版本自行解压缩并挂载根文件系统，然后会运行 `/sbin/init` 程序。

第 4 阶段：`systemd`

内核会加载 `systemd`，后者是 Linux 操作系统的系统和服务管理器。然后，`systemd` 会运行所有其他初始化进程。

2.2 由 systemd 控制的引导过程

一旦内核加载 `systemd`，`systemd` 即会接管并启动确保系统正常运行所需的其他系统服务，其中包括网络服务、登录管理器等服务。

引导过程按执行特定目标单元的顺序并行进行。`systemd` 使用 `/etc/systemd/system/default.target` 文件来确定 Linux 系统应引导到的目标。此文件是将会引导图形登录管理器的 `graphical.target` 的链接。`systemd` 会激活作为 `default.target` 的依赖项的所有目标单元，并以递归方式激活这些依赖项的所有依赖项。所有服务都启动后，您的系统就可以使用了，并会显示登录管理器。您现在可以登录并开始使用系统。

2.3 使用 systemd-analyze 命令分析系统引导过程性能

使用 **systemd-analyze** 命令可分析系统引导过程的性能。该命令还可用于从系统和服务管理器中检索其他状态和跟踪信息。它用于检查单元文件是否正确，并用于访问对高级系统管理器调试有用的特殊功能。

示例如下：

查看系统引导所花的时间

```
> systemd-analyze time
Startup finished in 3.404s (kernel) + 2.415s (initrd) + 13.125s (userspace)
 = 18.945s
graphical.target reached after 13.117s in userspace
```

粗略了解引导过程，包括已启动的服务以及每项服务启动所花的时间

```
> systemd-analyze critical-chain
  The time when unit became active or started is printed after the "@"
 character.
  The time the unit took to start is printed after the "+" character.

  graphical.target @13.117s
  └─multi-user.target @13.117s
    └─getty.target @13.117s
      └─getty@tty1.service @13.116s
        └─plymouth-quit-wait.service @10.775s +2.338s
          └─systemd-user-sessions.service @10.769s +3ms
            └─remote-fs.target @10.764s
              └─iscsi.service @10.747s +16ms
                └─network-online.target @10.744s
                  └─NetworkManager-wait-online.service @1.547s +9.197s
                    └─NetworkManager.service @1.507s +37ms
                      └─network-pre.target @1.504s
                        └─wpa_supplicant.service @2.341s +5ms
                          └─dbus.service @1.042s
                            └─basic.target @1.036s
                              └─sockets.target @1.036s
                                └─snapd.socket @1.035s +590us
                                  └─sysinit.target @1.030s
```

```
                                    └─systemd-update-utmp.service @1.025s
+5ms
                                      └─auditd.service @976ms +47ms
                                        └─systemd-tmpfiles-setup.service
@964ms +9ms
                                          └─local-fs.target @962ms
                                            └─snapd.mounts.target @961ms
                                              └─snap-core18-2796.mount
@417ms +543ms
                                                └─dev-loop9.device @961ms
+628us
```

此命令会针对每个指定单元或默认目标列显对时间要求严格的单元树。服务的初始化可能取决于套接字激活和单元的并行执行情况。与 **blame** 命令类似，它会显示激活某个单元所需的时间；对于直接转变为已启用状态的单元（如设备单元），不会定义该时间。

查看在引导过程中启动并按各服务所花时间显示的服务列表

```
> systemd-analyze blame
  9.197s NetworkManager-wait-online.service
  4.002s fwupd.service
  2.338s plymouth-quit-wait.service
  1.282s dracut-pre-udev.service
  1.062s sys-devices-platform-serial8250-tty-ttyS0.device
  1.062s dev-ttyS0.device
  1.061s dev-ttyS1.device
  1.061s sys-devices-platform-serial8250-tty-ttyS1.device
  1.060s dev-ttyS11.device
  1.060s sys-devices-platform-serial8250-tty-ttyS11.device
  1.059s sys-devices-platform-serial8250-tty-ttyS13.device
  1.059s dev-ttyS13.device
  1.059s sys-devices-platform-serial8250-tty-ttyS10.device
  1.059s dev-ttyS10.device
  1.058s sys-devices-platform-serial8250-tty-ttyS14.device
  1.058s dev-ttyS14.device
  1.058s dev-ttyS12.device
  1.058s sys-devices-platform-serial8250-tty-ttyS12.device
  1.056s sys-devices-platform-serial8250-tty-ttyS17.device
```

一项服务的初始化速度可能很慢，因为它正在等待另一项服务的初始化完成。它会显示激活某个单元所需的时间；对于直接转变为已启用状态的单元（如设备单元），不会定义该时间。此命令不会显示 `Type=simple` 的服务的结果，因为 `systemd` 将这些服务视为会立即启动，因而无法分析初始化延迟。

生成一个矢量图形文件以显示引导过程中发生的事件

```
> systemd-analyze plot > /temp/sample.svg
```

此命令会在 `temp` 目录中创建一个 SVG 文件。SVG 文件属于文本文件，它定义了一组图形向量，可供 LibreOffice Draw 等应用程序用来生成图表。

3 单元文件的结构

在 `systemd` 中，单元是指系统知道如何操作和管理的任何资源。这是 `systemd` 工具使用的主要对象。这些资源是使用配置文件（称作单元文件）定义的。如果您在使用 `systemd` 时了解单元文件，则管理工作就会变得更容易。单元文件使用简单的声明性语法，使您可以在激活单元后轻松知道单元的用途和影响。单元文件包含带有指令的部分，例如：

```
[Section]
        Directive1=value
        Directive2=value
        . . .
```

单元文件类型包括以下部分：

`[Unit]`

大多数单元文件中的第一个部分是 `[Unit]` 部分。此部分用于定义单元文件的元数据，以及配置该单元文件与其他单元文件的关系。此部分通常位于顶部，因为它提供单元文件的概览。

`[Automount] / [Mount] / [Path] / [Service] / [Slice] / [Socket] / [Swap] / [Timer]`

包含特定于相应类型的指令的部分。有关可用类型的列表，请参见第 4 节 “单元文件类型”。请注意，类型 `device`、`target`、`snapshot` 和 `scope` 不包含特定于类型的部分。

`[Install]`

此部分通常是单元文件中的最后一个部分，并且是可选的。此部分用于定义单元文件在启用或禁用后的行为。启用某个单元文件后，它会在引导时自动启动。根据特定的单元，可能需要依赖其他相关单元才能正常工作。例如，`chrony` 需要指令 `After`、`Wants` 和 `Before`，这些指令都是 `chrony` 使用的依赖项。

例 1： `systemd` 服务文件

```
[Unit]
Description=usbguard ❶

[Service]
ExecStart=/usr/sbin/usb-daemon ❷

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

❶ 解释服务文件用途的简短且有意义的说明。

❷ 指定当服务启动时要执行的程序。

❸ 启动具有网络但没有图形环境的多用户系统。此指令允许您指定依赖关系。

4 单元文件类型

您可以根据文件扩展名确定单元的类型。`systemd` 根据单元描述的资源类型对单元进行分类。

可用于 `systemd` 的单元文件类型：

`.service`

描述如何管理服务或应用程序。这包括如何启动或停止服务、重新加载服务配置文件（如果适用）、服务在哪种条件下自动启动，以及相关单元文件的依赖关系或层次结构信息。

`.scope`

此单元文件由 `systemd` 根据从 D-Bus 接口接收的信息自动创建，用于管理外部创建的系统进程集。

`.path`

定义基于路径的激活的路径。默认情况下，将激活具有相同基本名称的 `.service` 单元文件。`inotify` 是一个内核 API，由想要接收有关文件更改的通知的程序使用。

`.snapshot`

`systemctl snapshot` 命令自动创建 `.snapshot` 单元文件。此命令创建系统当前状态的临时快照。进行更改后，您可以修改系统的当前状态。快照用于回滚临时状态。

`.timer`

定义由 `systemd` 管理的计时器。这类似于针对延迟激活或已安排激活的 cron 作业。达到计时器时间时，将启动文件名相同、但扩展名为 `.service` 的单元文件。

`.slice`

关联 Linux 控制组节点，以便可以将资源指派或限制给与切片关联的任何进程。名称指示控制组树中的层次结构。默认情况下，单元根据其类型放置在切片中。

`.target`

在引导或状态更改期间为其他单元提供同步，或使系统进入新状态。其他单元指定它们与目标的关系，以便与目标的操作同步。

`.socket`

描述由 `systemd` 用于基于套接字的激活的网络、IPC 套接字或 FIFO 缓冲区。如果此单元定义的套接字上出现活动，则会启动一个关联的 `.service` 文件。

`.device`

定义一个设备，该设备已由 `udev` 或 `sysfs` 文件系统指定用于 `systemd` 管理。并非所有设备都有 `.device` 文件。排列、挂载或访问设备时需要此单元文件。

`.swap`

定义系统上的交换空间。单元文件的名称必须反映空间的设备或文件路径。

`.mount`

定义系统上由 `systemd` 管理的挂载点。此文件根据挂载路径命名（路径中的斜杠已更改为短划线）。`/etc/fstab` 中的项可以自动创建单元。

`.automount`

定义自动挂载的挂载点。根据文件引用的挂载点为文件命名。需要使用一个匹配的 `.mount` 单元文件来定义挂载细节。

5 单元依赖关系和顺序

`systemd` 有两种类型的依赖关系：要求依赖关系和顺序依赖关系。要求依赖关系指定在激活某个单元时必须启动或停止其他哪些单元。顺序依赖关系指定必须遵循的单元启动顺序。

单元依赖关系

单元文件具有依赖关系特性。某个单元可能需要在一个或多个其他单元运行之后才能运行。这些依赖关系是在单元文件中使用指令 `Wants` 和 `Requires` 设置的。

`Wants`

例如，如果单元 A 设置了 `Wants=unit B`，则当单元 A 运行时，单元 B 也会运行。但单元 B 能否成功启动不会影响单元 A 的成功运行。

`Requires`

如果单元 A 设置了 `Requires=unit B`，则这两个单元都会运行；但如果单元 B 未成功运行，则会停用单元 A。单元 A 的进程是否成功运行并不重要。

单元顺序

如果没有正确的指令，`systemd` 可以同时运行一组单元。以正确的顺序启动服务对于 Linux 系统的正常运行非常重要。可以使用单元文件指令 `Before` 和 `After` 来排列顺序。

`Before`

例如，如果单元 A 设置了 `Before=unit B`，当这两个单元同时运行时，会先完全执行单元 A，然后再执行单元 B。

`After`

如果单元 A 设置了 `After=unit B`，当这两个单元同时运行时，会先完全执行单元 B，然后再执行单元 A。

6 日志记录

日志文件和日记对于系统管理非常重要。它们提供有关系统的深入信息，对于查错和审计非常重要。日志文件包含内核、应用程序和登录到系统的用户所生成的事件和消息。您可以使用 **`journalctl`** 命令来查询日记。此命令可用于查看 `systemd` 收集的日志。`systemd-journald` 服务处理 `systemd` 的日志收集。`systemd-journald` 以二进制格式保存事件和消息。

7 systemd 目标

`systemd` 使用单元和目标。`systemd` 单元定义系统上的服务或操作，由名称、类型和配置文件组成。`systemd` 目标由多个单元组合而成，定义必须启动哪些服务才能达到目标。例如，在服务器上，这是网络正在运行并且多个用户可以登录的状态。这些文件由后缀 `.target` 标识。

与单元文件类似，可以通过依赖关系嵌套不同的目标。例如，`multi-user.target` 需要通过目标（及其他单元）来设置登录和用户会话服务。

常见 `systemd` 目标：

`default.target`

默认会引导。`default.target` 文件是指向实际目标文件（例如桌面工作站的 `graphical.target`）的符号链接。对于服务器，它通常是 `graphical.target`。

`poweroff.target`

关闭系统并关闭电源。

`rescue.target`

拉取基础系统并启动救援外壳会话的目标单元。

`multi-user.target`

设置非图形（控制台）多用户系统。

`graphical.target`

使用包含网络服务的图形多用户系统。

`reboot.target`

关闭再重引导系统。

有关 `systemd` 目标的详细信息，请参见 **man 5 systemd.target** 和 **man 7 systemd.special**。

8 以普通用户的身份使用 systemd

如果要实现更高的安全性或者您没有 `root` 用户特权，可以用普通用户的身份使用 `systemd`。可以通过创建 `user` 服务来运行非特权服务。

创建和使用用户服务时，请考虑以下事项：

- 当用户的会话结束时，用户服务会话也会终止。可以使用 **`loginctl enable-linger`** `USERNAME` 命令覆盖此设置。
- 用户服务文件位于 `/etc/systemd/user` 或 `$HOME/.config/systemd/user/` 中。
- 可以使用 **`systemctl --user`** 命令控制用户服务。

9 `systemctl` 命令

`systemctl` 命令用于检查及控制 `systemd` 和服务管理器的状态。

您可以使用以下常见 **systemctl** 命令并参见 **man systemctl** 页面。

9.1 查看 systemd 信息

要查看有关 systemd 组件的信息，可以使用以下命令：

systemctl list-units

列出 systemd 单元。可以使用可选参数：**--state=running** 用于显示已启用的单元，**--type=service** 用于显示已退出和已启用的单元。

systemctl list-unit-files

列出 systemd 单元和状态，例如 static、generated、disabled、alias、masked 和 enabled。

systemctl list-dependencies

列出依赖关系树。

systemctl list-dependencies UNIT_FILE

列出单元文件的依赖关系。

9.2 管理 systemd 服务

systemctl 命令可用于对服务执行以下任务。

systemctl status SERVICE

检查特定服务的状态。

systemctl show SERVICE

显示服务信息。

systemctl start SERVICE

不是手动启动服务，而是使用 **start** 命令。对配置文件进行更改后，必须再次启动相关服务。

systemctl stop SERVICE

停止正在运行的特定服务。

systemctl restart SERVICE

不是手动重启动服务，而是使用 **restart** 命令。对配置文件进行更改后，必须再次重启动相关服务。

systemctl enable SERVICE

在引导时启用服务。

systemctl disable SERVICE

在引导时禁用服务。

systemctl reload-or-restart SERVICE

如果服务支持重新加载，则重新加载服务，否则重启动服务。如果服务未运行，则将其重启动。

systemctl mask SERVICE

当服务被屏蔽时，这意味着单元文件与 `/dev/null` 建立了符号链接。将为屏蔽的服务创建从 `/etc/systemd/system` 指向 `/dev/null` 的符号链接。这样，即使另一个已启用的服务需要该服务，也无法加载该服务。必须手动停止该服务，否则它会继续在后台运行。可以使用 **--runtime** 选项来暂时屏蔽服务，直到系统下次重引导。

```
Created symlink /etc/systemd/system/FOSSLinux.service → /dev/null.
```

systemctl unmask SERVICE

取消屏蔽服务。手动启动或重启动系统时，此设置将会生效。

9.3 管理系统状态

systemctl 命令可用于在系统上执行重启动、关机等电源管理任务，如下文所述。

systemctl reboot

重引导系统 `reboot.target`。

systemctl poweroff

关闭系统 `poweroff.target` 的电源。

systemctl emergency

进入紧急模式 `emergency.target`。

systemctl default

返回默认目标 `multi-user.target`。

10 systemd 查错

可以使用以下查错提示来识别和解决 `systemd` 服务的问题，并确保系统平稳运行。

使用 `systemd-analyze verify SERVICE` 检查 `systemd` 单元文件的语法

在启动或启用 `systemd` 服务之前，请检查单元文件的语法，确保没有任何错误。例如：

```
> sudo systemd-analyze verify /etc/systemd/system/my-custom-service.service
```

该命令会分析单元文件，并报告任何语法错误、缺失的文件或其他问题。在启用和启动服务之前，您必须解决报告的所有问题。

使用 `journalctl -u SERVICE` 命令检查服务的日志

如果您在使用某个 `systemd` 服务时遇到任何问题，请检查该服务的日志。例如：

```
> sudo  journalctl -u my-custom-service.service
```

该命令显示指定服务的日志，包括任何错误消息、警告或其他相关信息。您可以使用这些日志来识别和修复服务问题。

使用 `systemd-analyze plot` 命令来可视化引导过程

如果某项服务导致引导过程出现问题，您可以使用 **`systemd-analyze plot command`** 来可视化引导过程并识别问题。例如：

```
> sudo systemd-analyze plot > boot-plot.svg
```

该命令会创建一个名为 `boot-plot.svg` 的 SVG 文件，其中包含引导过程和潜在问题的图形表示。这包括每项服务的启动和停止时间。您可以在 SVG 兼容的图像查看器或 Web 浏览器中打开此文件，以分析在引导过程中导致问题的服务。

对失败的服务进行查错

要确定哪些服务失败并检查日志输出，请运行以下命令：

```
> sudo systemctl --state=failed
```

检查服务的运行时状态

要确定服务的当前运行时状态，请运行以下命令：

```
> sudo systemctl status SERVICE
```

关机或重引导时间过长

如果关机或重引导时间过长，原因可能是某个服务未退出。systemd 会先等待一段时间让每个服务退出，然后再尝试终止它。一个常见问题是服务挂起或关机过程停滞不前。要确定原因，请使用以下命令：

```
> sudo systemctl poweroff
  Failed to power off system via logind: There's already a shutdown or
 sleep operation in progress
```

```
> sudo systemctl list-jobs
```

可以取消运行中和等待中的作业，然后再次关机或重引导：

```
> sudo systemctl cancel
```

```
> sudo systemctl stop systemd-suspend.service
```

11 systemd 最佳实践

您可以遵循一些最佳实践，来确保能够处理不同状况的 systemd 服务高效运行。

检查服务的运行时状态

要确定服务的当前运行时状态，请运行以下命令：

```
> sudo systemctl status SERVICE
```

在 systemd 单元文件中使用绝对路径

在 systemd 单元文件中使用可执行文件和所需文件（例如配置文件或脚本）的绝对路径。systemd 不依赖使用用户的环境变量（例如 $PATH）来查找文件。

使用 ExecReload 指令

如果您想定义一个在使用 **`systemctl reload`** 命令重新加载服务时要执行的特定命令，请在 `[SERVICE]` 部分使用 **ExecReload** 指令。此指令对于无需重启动即可动态重新加载其配置的服务非常有用。

```
[Service]
ExecStart=PATH_TO_EXECUTABLE
ExecReload=PATH_TO_RELOAD_SCRIPT
```

使用 RestartSec 指令

如果您想定义一段延迟时间（以秒为单位），发生故障后，需要经过这段时间才重启动服务，那么，请在 `[SERVICE]` 部分使用 **RestartSec** 指令。此指令对于需要在经过指定时间后才释放资源，或防止出现快速重启循环（这可能导致系统负载过高）的服务非常有用。

```
[Service]
ExecStart=PATH_TO_EXECUTABLE
Restart=on-failure
RestartSec=5
```

在远程计算机上禁用紧急模式

您可以在远程计算机（例如 Google Cloud 托管的虚拟机）上禁用紧急模式。如果启用此模式，将会阻止计算机连接到网络。例如：

```
> sudo systemctl mask emergency.service
```

```
> sudo systemctl mask emergency.target
```

12 法律声明

有关 SUSE 商标，请参见 https://www.suse.com/company/legal/。所有其他第三方商标分别为相应所有者的财产。商标符号（®、™ 等）代表 SUSE 及其关联公司的商标。星号 (*) 代表第三方商标。

本指南力求涵盖所有细节，但这不能确保本指南准确无误。SUSE LLC 及其关联公司、作者和译者对于可能出现的错误或由此造成的后果皆不承担责任。

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。