第 1 章 正确的起点

目录

本文档致力于为普通 Debian 用户，和希望对 Debian 软件包有所了解的开发人员讲述如何制作 Debian 软件包。在这里，我们尽可能使用通俗的语言，并辅以实例来直观地展示每一个细节。有一句古罗马谚语说得好：*一例胜千言！*

本教程文档已被重写为另外的 Debian 维护者指导 文档，其中包含了更新的内容与更多实际例子。请使用新的教程作为主要的教程文档。

该文档在 Debian **Buster** 发布时仍然可用，因为它已提供了许多语言的翻译版本。该文档会在之后的 Debian 版本中被移除，因为其内容正在逐渐过时。[1]

Debian 的软件包系统是使它跻身顶级发行版行列的重要原因之一。尽管已经有相当数量的软件被打包成 Debian 的格式，但有时还是需要安装一些并非该格式的软件。可能你正为如何制作自己的软件包而感到迷惑，也可能认为这么做很难。如果你是一个刚刚接触 Debian 的初学者，那么是的，它的确很难；不过假如你真的只是一个初入此门的新手，现在大概也不会来读这篇文档了。:-) 你的确需要对 Unix 编程有所了解，但显然没必要是这方面的天才。[2]

对于 Debian 软件包维护人员来说，有一件事是非常明确的：创建并维护一个 Debian 软件包需要花费很多精力，所需的时间很可能远不只是几个小时。维护人员需要有良好的技术基础，同时也需要十分勤奋，这样才能保证我们的系统正常运行而不出现问题。

如果你在软件包制作方面需要他人帮助，请阅读 第 1.4 节 “到何处求助”。

本文的最新版随时都可以在 http://www.debian.org/doc/maint-guide/ 上和 **maint-guide** 软件包里找到。文档的简体中文翻译可以在 **maint-guide-zh-cn** 软件包里找到。还有一点需要注意的是，这篇文档的内容相对于当前的开发情况可能会有略微的延迟。

由于这是一份手把手的教程，所以在一些重要的话题上会对每个步骤都做详细的解释。虽然你可能觉得它们之中有一些与你的想法毫不相干，但也请准备好足够的耐心来学习。另外，我也有意地省略了某些不必要的细节，以使这篇文档尽可能保持简洁。

1.1. Debian 的社会驱动力

以下是一些有关 Debian 的社会动力学报告，希望它们能够帮助你做好准备，以与 Debian 进行交互：

- 大家都是志愿者。
  - 任何人都不能把事情强加给他人。
  - 你应该主动地做自己想做的事情。
- 友好的合作是我们前行的动力。
  - 你的贡献不应致使他人增加负担。
  - 只有当别人欣赏和感激你的贡献时，它才有真正的价值。
- Debian 并不是一所学校，在这里没有所谓的老师会自动地注意到你。
  - 你需要有自学大量知识和技能的能力。
  - 其他志愿者的关注是非常稀缺的资源。
- Debian 一直在不断进步。
  - Debian 期望你制作出高质量的软件包。
  - 你应该随时调整自己来适应世界的变化。

在 Debian 社区中有这几类常见的角色：

- **Upstream author** (上游作者)：程序的原始作者。
- **Upstream maintainer** (上游维护者)：目前在上游维护程序代码的人。
- **Maintainer** (软件包维护者)：制作并维护该程序 Debian 软件包的人。
- **Sponsor** (赞助者)：帮助维护者上传软件包到 Debian 官方仓库的人（在通过内容检查之后）。
- **Mentor** (导师)：帮助新手维护者熟悉和深入打包的人。
- **Debian Developer** (DD，Debian开发者)：Debian 社区的官方成员。DD 拥有向 Debian 官方仓库上传的全部权限。
- **Debian Maintainer** (DM，Debian维护者)：拥有对 Debian 官方仓库部分上传权限的人。

注意，你不可能在一夜之间成为 **Debian Developer**，因为成为 DD 所需要的远不只是技术技巧。不过别因此而气馁，如果你的软件包对其他人有用，你可以当这个软件包的 **Maintainer**，然后通过一位 **Sponsor** 来上传这份软件，或者你可以申请成为 **Debian Maintainer**。

还有，要成为 Debian Developer 不一定要创建新软件包。对已有软件做出贡献也是成为 Debian Developer 的理想途径。眼下正有很多软件包等着好的维护者来接手(参看 第 2.2 节 “挑一个你喜欢的程序”)。

在这篇文档里，我们的重点在于制作软件包的技术细节。有关 Debian 是如何运转的，以及如何才能参与到其中之类的话题，请参考下边的文档：

- Debian: 17 years of Free Software, "do-ocracy", and democracy (幻灯片介绍)
- How can you help Debian? (官方文档)
- The Debian GNU/Linux FAQ, Chapter 13 - "Contributing to the Debian Project" (半官方文档)
- Debian Wiki, HelpDebian (补充材料)
- Debian New Member site (官方站点)
- Debian Mentors FAQ (补充文档)

1.2. 需要的开发工具

在开始之前，请确认你是否已经正确安装了开发所需要的工具集。注意这些软件包中没有任何一个会被标记为 **essential** 或 **required** —— 这里假设你已经安装了它们。

以下这些软件包已经随标准的 Debian 安装过程进入了系统，所以你可能不需要再动手安装它们（以及任何附加的依赖软件包）。然而，你还是应该用 **aptitude show *package*** 或者 **dpkg -s *package*** 来检查一下。（译注：apt show PACKAGE 亦可）

在你的开发环境中，最重要的软件包是 **build-essential**。一旦你尝试安装该包，它将 *拉取* 其他基本构建环境所需的工具链。

对于某些类型的软件，以上的就是所需要的全部。然而还有一组工具虽不是对于所有软件包都必须，却可能对你有用，或者你的软件包制作过程中会需要它们：

- **autoconf**、**automake** 和 **autotools-dev** - 很多新程序使用 configure 脚本和 **Makefile** 文件来帮助预处理程序。(参看 **info autoconf**、**info automake**)。**autotools-dev** 则用于保持指定的自动配置文件为最新，并带有关于使用那些文件的最佳方法的文档。
- **debhelper** 和 **dh-make** - **dh-make** 是用于创建我们示例软件包骨架所必须的，它会使用 **debhelper** 中的一些工具来创建软件包。他们不是创建软件包所必须的，但对新维护人员而言，我们 *强烈推荐*。它可以使整个过程极为简化，并易于在将来维护。(参看 dh_make(8)、debhelper(1)、**/usr/share/doc/debhelper/README**) [3]

  新的 **debmake** 可以作为标准 **dh-make** 的代替品。debmake 能做的事情更多，并且拥有包含非常多打包实例的 HTML 文档。文档可以通过 **debmake-doc** 软件包获取。
- **devscripts** - 此软件包提供了一些非常好且有用的脚本来帮助维护者，不过这写脚本并非制作软件包所必须。此软件包所推荐或建议的软件包都值得一看。(参看 **/usr/share/doc/devscripts/README.gz**)
- **fakeroot** - 这个工具使你可以在编译过程中必要的时候以普通用户来模拟 root 用户环境。(参看 fakeroot(1))
- **file** - 这个小程序可以检测文件的类型。(参看 file(1))
- **gfortran** - GNU Fortran 95 编译器，如果你的程序是用 Fortran 编写的则必须用此工具完成编译。(参看 gfortran(1))
- **git** - 此软件包提供了用于快捷处理大型项目的著名版本控制系统 - git。它被广泛用于各种开源项目，其中最著名的是 Linux 内核项目。(参见 git(1), git Manual (**/usr/share/doc/git-doc/index.html**).)
- **gnupg** - 让你可以使用 *数字签名* 签署你的软件包。当你想把它分发给其他人时这一点特别重要。如果你要把你的成果加入到 Debian 发行版中，那这是必须的步骤。(参看 gpg(1).)
- **gpc** - GNU Pascal 编译器。如果你的程序是用 Pascal 写的则需要此工具。值得一提的是 **fp-compiler**，Free Pascal 编译器(FPC)，也能够很好地胜任编译任务。(参见 gpc(1), ppc386(1).)
- **lintian** - Debian 软件包检查工具，使你可以在编译软件包后知道它是否犯了常见的错误，并对其找到的错误进行解释。(参见 lintian(1), Lintian User's Manual.)

- **patch** - 这是一个非常有用的工具，它可以把 **diff** 程序生成的差异清单文件应用到原先的文件上，从而生成一个打了补丁的版本。(参看 patch(1))
- **patchutils** - 此软件包提供了一些帮助处理补丁的工具，如 **lsdiff**、**interdiff** 和 **filterdiff** 命令。
- **pbuilder** - 此软件包提供了创建和维护 **chroot** 环境的工具。在它的 **chroot** 环境中编译 Debian 软件包可以检查编译依赖是否合适，并避免 FTBFS (Fails To Build From Source，源代码编译失败)的 Bug。(参看 pbuilder(8) 和 pdebuild(1))
- **perl** - Perl 是现今类Unix系统中使用最普遍的解释型脚本语言，它常被称作Unix的瑞士军刀。(参看 perl(1))
- **python** - Python 是 Debian 系统中另一个最常用的解释型脚本语言，它拥有着可圈可点的强大功能和十分清晰的语法。(参看 python(1))
- **quilt** - 此软件包帮助你管理一系列的补丁。它们被以逻辑栈的方式组织在一起。你可以 apply (=push)、un-apply (=pop) 或简单地刷新它们然后再放入栈内。(参看 quilt(1), and **/usr/share/doc/quilt/quilt.pdf.gz**.)
- **xutils-dev** - 一些通常用于 X11 的程序，使用其宏功能可以生成 **Makefile** 文件。(参看 imake(1)、xmkmf(1))

以上给出的简短描述仅仅是为了使你对这些工具有一个基本的印象。在继续前请详细阅读每个程序(包括通过依赖关系安装的程序，比如**make**)的文档，至少了解其一般的用途和用法。现在看来这是一项耗时巨大的任务，但在接下来的工作中你将为你阅读了它们而感觉到 *非常* 愉快。如果一会你遇到一些特定的问题，我会建议你重新阅读上面提到的文档。

1.3. 需要的开发文档

以下是 *非常重要* 的文档，你应该在读本文档时同时参看它们：

- **debian-policy** - the Debian Policy Manual 包含了对 Debian 软件仓库、操作系统设计问题、文件系统层级标准(FHS, Filesystem Hierarchy Standard，讲述每个文件和目录应该放在哪里)等的描述。对于你而言，最重要的是它描述了软件包进入官方仓库前必须满足的条件。(请参见 **/usr/share/doc/debian-policy/policy.pdf.gz** 和 **/usr/share/doc/debian-policy/fhs/fhs-3.0.pdf.gz** 的本地副本)
- **developers-reference** - Debian 开发者参考 描述了打包所需的包含技术细节在内的全部详细信息，如仓库结构、如何重命名/丢弃/接手软件包、如何进行 NMU(非维护者上传)、如何管理 Bug 以及打包最佳实践、何时向何处上传等。(参见 **/usr/share/doc/developers-reference/developers-reference.pdf** 的本地副本)

以下是 *重要* 的文档，你应该在读本文档时同时参看它们：

- Autotools Tutorial 为 the GNU Build System known as the GNU Autotools 中最重要的工具 —— Autoconf、Automake、Libtool 和 gettext 提供了很好的文档。
- **gnu-standards** - 此软件包包含了 GNU 项目中的两篇文档：GNU Coding Standards 和 Information for Maintainers of GNU Software。尽管 Debian 不要求遵守这些规范，但它们作为纲领和共识仍然很有帮助。(参见 **/usr/share/doc/gnu-standards/standards.pdf.gz** 和 **/usr/share/doc/gnu-standards/maintain.pdf.gz** 的本地副本)

若本文档所叙述的内容与 Debian Policy Manual 或 Debian Developer's Reference 有不符，则按照后两者的要求为准，并向 **maint-guide** 软件包提交 Bug 报告。

以下是替代性的教程文档，你可以在读本文档时同时参看它们：

- Debian Packaging Tutorial

1.4. 到何处求助

在你决定到公共场合提问之前，请先阅读这些(个)不错的文档：

- 所有相关软件包的 **/usr/share/doc/*package*** 目录之中的文件
- 所有相关命令的 **man *command*** 手册页
- 所有相关命令的 **info *command*** info 页
- 邮件列表档案 debian-mentors@lists.debian.org
- 邮件列表档案 debian-devel@lists.debian.org

在搜索时你可以在搜索引擎中使用类似这样的字符串 **:site:lists.debian.org**，以此限制域名从而提高效率。

制作小的测试软件包是学习打包细节的好方法，仔细查看维护较好的软件包则是了解他人如何制作软件包的最佳办法。

如果你仍然对打包存有疑问，并且在文档和WEB资源中都不能找到答案，你可以交互式地向他们提问：

- http://lists.debian.org/debian-mentors/ 邮件 debian-mentors@lists.debian.org。(该邮件列表是新手专区。)
- debian-devel@lists.debian.org。(该邮件列表汇集各路神仙。)
- IRC 比如 **#debian-mentors**。
- 专注于某个软件包集合的团队。(完整列表参见 https://wiki.debian.org/Teams)
- 特定语言的邮件列表，比如 debian-devel-{french,italian,portuguese,spanish}@lists.debian.org 或 debian-devel@debian.or.jp。(完整列表参见 https://lists.debian.org/devel.html 和 https://lists.debian.org/users.html)

如果你付出了一定的努力并且提问得当，那么有经验的 Debian 开发者会很乐意帮助你的。

当你收到一个 Bug 报告后 (没错，真正的 Bug 报告！)，你需要研究 Debian Bug Tracking System (Debian Bug 追踪系统，BTS)并阅读相关的文档以便高效处理这些报告。我在此强烈推荐阅读 Developer's Reference, 5.8. "Handling bugs"。

即使上边的那些问题都解决了，也不能高兴得太早。为什么？因为几个小时或几天内就会有人开始使用你的软件包，如果你犯了某些严重的错误，将会被无数生气的 Debian 用户进行邮件轰炸..... 哦，当然这只是开个玩笑。:-)

放松一点并准备好处理 Bug 报告，在你的软件包完全符合 Debian 的各项规范前你还需要付出很多努力，处理 Bug 对你也是很好的锻炼 (再一次提醒，阅读那些 *必须的文档* 来了解详情)。祝你好运！

---

[1] 在写这份文档时，我们默认你使用 **jessie** 或者更新的操作系统。如果你需要在 **更老版本的** 系统上(包括老版本的 Ubuntu 等)使用本文所记述的方法，则至少必须安装 backports 仓库中的 **dpkg** 和 **debhelper** 软件包。

[2] 在 Debian Reference 中，你可以了解到使用 Debian 系统的一些基本方法和关于 Unix 编程的一些指引。

[3] 还有几个类似但更针对某一类软件的工具，如 **dh-make-perl**、**dh-make-php** 等。