第 5 章 debian 目录下的其他文件

目录

要控制 **debhelper** 在构建软件包过程中的大部分行为，可以在 **debian** 目录中放置可选的配置文件。本章将会对这些文件和它们的格式进行概述。请阅读 Debian Policy Manual 和 Debian Developer's Reference 来了解更多关于打包方针的内容。

dh_make 命令会在 **debian** 目录中创建一些配置文件模板，它们的文件名多带有 **.ex** 后缀。其中的一些可能以二进制包名作为前缀，如 ***package***。现在我们来对它们进行一个大致的了解。[54]

还有一些 **debhelper** 的配置文件模板可能未被 **dh_make** 命令创建。在这种情况下如果你需要它们，则要使用文本编辑器手工创建。

如果你希望或需要激活它们中的任意一个或多个，请按照下面的方法做：

- 如果模板文件带有 **.ex** 或 **.EX** 后缀，则重命名它去掉后缀；
- 把配置文件的名称改为实际的二进制软件包名，例如 ***package***；
- 修改模板文件来满足你的需要；
- 删除不需要的模板文件；
- 如果需要，修改 **control** 文件(参看 第 4.1 节 “**control**”)。
- 如果需要，修改 **rules** 文件(参看 第 4.4 节 “**rules**”)。

任何不带有 ***package*** 前缀的**debhelper** 配置文件，比如应用到第一个二进制包的 **install**。当此处有多个二进制包时，可以通过在文件名中前置 它们各自的名称来指定它们各自的配置文件，比如***package-1*.install**，***package-2*.install**，之类。

5.1. README.Debian

所有关于你的 Debian 版本与原始版本间的额外信息或差别都应在这里记录。

dh_make 创建了一个默认的文件，以下是它的样子：

```
gentoo for Debian
-----------------
<possible notes regarding this package - if none, delete this file>
 -- Josip Rodin <joy-mg@debian.org>, Wed, 11 Nov 1998 21:02:14 +0100
```

如果你没有需要写在这里的东西，则删除这个文件。参看 dh_installdocs(1)

5.2. compat

compat 文件定义了 **debhelper** 的兼容级别。目前你应当使用如下方法将其设置为 **debhelper** V10：

```
$ echo 10 > debian/compat
```

在特定场景下，你可以在需要兼容旧版本系统时使用兼容等级9。然而，我们不建议你使用任何低于 9 的兼容等级，在新建软件包时也应避免使用这些低的等级。

5.3. conffiles

关于软件有件很恼人的事，就是当你付出了很多时间和精力来自定义一个程序，但是升级后所有的修改都被覆盖掉了。Debian 通过将配置文件单独标记来解决这个问题，[55] 当软件包升级的时候，你将会被询问是否要保留你的旧配置文件。

dh_installdeb(1) *automatically* 会把 **/etc** 目录下的任何文件都标记为 conffiles，所以如果你的程序在那只有 conffiles 的话就不需要再在这个文件中指定它们。对于大多数软件包类型，唯一合理的 conffiles 文件存放位置自始至终应当在 **/etc** 目录下，正因如此，该文件也没有存在的必要。

如果你的程序使用配置文件，但程序会自动对配置文件进行改写，则最好别将其标记为 conffiles，因为 **dpkg** 总是会要求用户校验变更。

如果你正在打包的程序需要所有用户都为自己修改 **/etc** 目录中的配置文件，那么有两种常见的方法让 **dpkg** 不将其记录为 conffiles，以使其沉默。

- 在 **/etc** 目录中创建指向 **/var** 中维护者脚本(maintainer scripts)生成的文件的符号链接。
- 用维护者脚本(maintainer scripts)在 **/etc** 目录中生成一个文件。

更多关于维护者脚本(maintainer scripts)的信息，参看 第 5.18 节 “**{pre,post}{inst,rm}**”

5.4. *package*.cron.*

如果你的软件包需要有计划运行的操作以保证正常工作，可以使用这个文件达成。你既可以设置常规的任务使其在每小时、每天、每星期、每月或其他情况或你希望的时间下运行。相应的文件名是：

- ***package*.cron.hourly** - 安装至 **/etc/cron.hourly/*package***；每小时运行一次。
- ***package*.cron.daily** - 安装至 **/etc/cron.daily/*package***；每天运行一次。
- ***package*.cron.weekly** - 安装至 **/etc/cron.weekly/*package***；每周运行一次。
- ***package*.cron.monthly** - 安装至 **/etc/cron.monthly/*package***：每月运行一次。
- ***package*.cron.d** - 安装至 **/etc/cron.d/*package***：对于其他的任何时间。

这些文件均为 shell 脚本。唯一不同的是 ***package*.cron.d**，它的格式需要参照 crontab(5)

有必要显式地用 **cron.*** 文件来设置日志轮转；关于这个问题，请参见 dh_installlogrotate(1) 和 logrotate(8).

5.5. dirs

这个文件指定了我们需要，但在正常安装过程(**dh_auto_install** 触发的 **make install DESTDIR=...**)里没有被安装的目录。通常这是由于 **Makefile** 中存在问题。

install 文件中列出的文件不需要为其提前创建目录，参看 第 5.11 节 “**install**”。

最好是先尝试安装，如果遇到了问题再使用这个文件。在 **dirs**中列出的目录名中不应有前导的 / (斜杠)符号。

5.6. *package*.doc-base

如果你的软件包在 man 手册页和 info 信息文档外还有其他文档，你应该使用 **doc-base** 文件注册它们，这样用户可以使用例如 dhelp(1)、dwww(1) 或 doccentral(1) 的工具找到它们。

这通常包括 HTML、PS 和 PDF 文件，放置在 **/usr/share/doc/*packagename*/**。

以下是 **gentoo** 的 doc-base 文件 **gentoo.doc-base** 的样子：

```
Document: gentoo
Title: Gentoo Manual
Author: Emil Brink
Abstract: This manual describes what Gentoo is, and how it can be used.
Section: File Management
Format: HTML
Index: /usr/share/doc/gentoo/html/index.html
Files: /usr/share/doc/gentoo/html/*.html
```

关于文件格式的更多信息，参看 install-docs(8) 以及 Debian **doc-base** 手册页，在 **/usr/share/doc/doc-base/doc-base.html/index.html** 有一份本地拷贝，这是由 **doc-base** 软件包提供的。

关于安装附加文档的更多信息，查看 第 3.3 节 “文件安装”。

5.7. docs

这个文件指定了我们使用 dh_installdocs(1) 安装到临时目录的文档文件名。

默认情况下它会加入代码目录顶层的所有名为 **BUGS**、**README***、**TODO** 等的文件。

对于 **gentoo**，这里还加入了一些其他文件：

```
BUGS
CONFIG-CHANGES
CREDITS
NEWS
README
README.gtkrc
TODO
```

5.8. emacsen-*

如果你的软件包提供可以在安装时编译为字节码的 Emacs 文件，你可以使用这些文件设置。

它们会被 dh_installemacsen(1) 安装到临时目录。

如果你不需要这些，就删除它们。

5.9. *package*.examples

dh_installexamples(1) 会将列出的文件和目录作为示例文件安装。

5.10. *package*.init 和 *package*.default

如果你的软件包需要在系统启动时运行一个守护进程，那么你显然没有按照我最初的建议做事，不是吗？ :-)

package*.init** 文件会被安装为 **/etc/init.d/*package 脚本，该脚本可用于启动和停止守护进程。**dh_make** 创建的 **init.d.ex** 是一个很好的骨架模板。你可能要对其改名并做很多修改，同时还要提供 Linux Standard Base (LSB，Linux标准规范) 的兼容头文件。用 dh_installinit(1) 可以将它安装到临时目录中。

package*.default** 文件 会被安装至 **/etc/default/*package。这个文件设置被 init 脚本使用的默认设置。***package*.default** 文件最经常用于设置 一些默认标记或者超时。如果你的 init 脚本中有某种 可配置的特性，你可以在 ***package*.default** 文件中设置它们，而不是 init 脚本本身。

如果你的上游程序中包含了给 init 用的脚本文件，那用不用它都可以。如果不使用，则创建相应的 ***package*.init** 文件；然而如果上游的 init 脚本很好且被安装到正确的位置，你仍然需要设置 **rc*** 符号链接。你需要按照以下的方法在 **rules** 文件中凌驾 **dh_installinit**：

```
override_dh_installinit:
        dh_installinit --onlyscripts
```

如果你不需要这些，就删除它们。

5.11. install

如果你的软件包需要那些标准的 **make install** 没有安装的文件，你可以把文件名和目标路径写入 **install** 文件，它们将被 dh_install(1) 安装。[56] 你需要首先检查要安装的文件是否有更有针对性的特定工具会对其进行安装。例如，文档应写在 **docs** 文件中安装，而不是这一个。

这个 **install** 文件每行安装一份文件，格式上先是相对于编译目录的文件路径，然后是一个空格，接下来是相对于安装目录的目标目录。例如，假设某个二进制文件 **src/*bar*** 没有被默认安装，则应让 **install** 呈现成这样：

```
src/bar usr/bin
```

这意味着安装这个软件包时，将有一个二进制文件 **/usr/bin/*bar***。

当然，在相对路径保持不变的情况下 **install** 文件也可以只包含相对的源路径而不带目标位置。这样的格式通常用于使用 ***package-1*.install**、***package-2*.install** 等将大软件包分割为多个二进制包的情况。

如果 **dh_install** 命令没有在当前目录(或者你可能使用 **--sourcedir** 参数指定的位置)找到文件，它会回滚至使用 **debian/tmp** 目录。

5.12. *package*.info

如果你的软件包有 info 信息页，应该使用 dh_installinfo(1) 安装，要安装的文件列于 ***package*.info** 文件中。

5.13. *package*.links

如果你作为包维护者需要在包中创建附加的符号链接，你应该使用 dh_link(1) 来安装，要安装的文件完整路径列于 ***package*.info** 文件中。

5.14. {*package*.,source/}lintian-overrides

如果 **lintian** 根据 Debian Policy 的某些规则允许例外从而报告了错误诊断，你可以使用 ***package*.lintian-overrides** 或 **source/lintian-overrides** 使其不再警告。请认真阅读 Lintian 用户手册(**https://lintian.debian.org/manual/index.html**) 并避免滥用。

package*.lintian-overrides** 是对于名为 ***package 的二进制包的有效配置，会由 **dh_lintian** 命令安装到**usr/share/lintian/overrides/*package***。

source/lintian-overrides 是针对源代码包的，不会安装。

5.15. manpage.*

你的程序应该有 man 手册页，如果它们没有自带则需要由你来创建。**dh_make** 命令创建了几个 man 手册页的模板。为每一个缺手册的命令拷贝一份模板，并妥善地编写，并且要删除未使用的模板。

5.15.1. manpage.1.ex

man 手册页通常是使用 nroff(1) 的格式编写的。**manpage.1.ex** 模板也是使用 **nroff** 格式的。参看 man(7) 手册页来简要了解如何编辑这个文件。

最终的 man 手册页文件的名称，应当为其对应的程序名称。所以我们将 **manpage** 重命名为 **gentoo**。同时，它的文件名当然要以 **.1** 作为第一个后缀，这表示本手册页是为一个用户命令而撰写的。再者，请校验本 man 手册处在正确的节(section)。这个简短的列表列举了 man 手册页的章节：

Section(部分)	Description(描述)	Notes(提示)
1	User commands(用户命令)	Executable commands or scripts(可执行命令或脚本)
2	System calls(系统调用)	Functions provided by the kernel(由内核提供的功能)
3	Library calls(库调用)	Functions within system libraries(系统库中的功能)
4	Special files(特殊文件)	经常在 **/dev** 目录中出没
5	File formats(文件格式)	例如，**/etc/passwd** 的格式
6	Games(游戏)	Games or other frivolous programs(游戏或无聊的程序)
7	Macro packages(宏包)	比如 **man** 宏
8	System administrarion(系统管理)	Programs typically only run by root(典型的root专用程序)
9	Kernel routines(内核惯例)	Non-standard calls and internals(非标准调用及内部构建)

所以 **gentoo** 的 man 手册页应叫 **gentoo.1**。如果原始代码中没有 **gentoo.1** man 手册页，你需要重命名 **manpage.1.ex** 模板为 **gentoo.1** 并按照示例和上游文档编辑它。

你也可以使用 **help2man** 命令来借助每个程序的 **--help** 和 **--version** 参数的输出来生成 man 手册页。[57]

5.15.2. manpage.sgml.ex

如果你希望使用 SGML 而非 **nroff** 格式编写 man 手册页，可以使用 **manpage.sgml.ex** 模板。如果你要这样，需要进行以下步骤：

- 重命名文件为类似 **gentoo.sgml** 的名称。

- 安装 **docbook-to-man** 软件包。
- 在 **control** 文件中将 **docbook-to-man** 添加到 **Build-Depends** 行。
- 在 **rules** 文件里添加 **override_dh_auto_build** target：

```
override_dh_auto_build:
        docbook-to-man debian/gentoo.sgml > debian/gentoo.1
        dh_auto_build
```

5.15.3. manpage.xml.ex

如果你希望使用XML 而非 SGML，可以使用 **manpage.xml.ex** 模板。如果你要这样，需要进行以下步骤：

- 重命名源文件为类似 **gentoo.1.xml** 的名称。
- 安装 **docbook-xsl** 软件包和一个 XSLT 处理器，例如 **xsltproc** (推荐)。
- 在 **control** 文件中将 **docbook-xsl**，**docbook-xml**，以及 **xsltproc** 软件包添加到 **Build-Depends** 行。
- 在 **rules** 文件里添加 **override_dh_auto_build** target：

```
override_dh_auto_build:
        xsltproc --nonet \
         --param make.year.ranges 1 \
         --param make.single.year.ranges 1 \
         --param man.charmap.use.subset 0 \
         -o debian/ \
http://docbook.sourceforge.net/release/xsl/current/manpages/docbook.xsl\
        debian/gentoo.1.xml
        dh_auto_build
```

5.16. *package*.manpages

如果你的软件包有 man 手册页，你应该将它们列在 ***package*.manpages** 文件中以便 dh_installman(1) 进行安装。

要将 **doc/gentoo.1** 安装为 **gentoo** 的 man 手册页，创建一个 **gentoo.manpages**，内容如下：

```
docs/gentoo.1
```

5.17. NEWS

dh_installchangelogs(1) 命令会安装这个文件。

5.18. {pre,post}{inst,rm}

postinst、**preinst**、**postrm** 和 **prerm** 文件 [58] 被称为 *maintainer scripts*。它们是放置于软件包控制区域，并由 **dpkg** 在软件包安装、升级或卸载时执行的脚本。

作为一个新维护人员，你应当避免手工编辑 maintainer scripts，因为它们常存在各种问题。更多信息请阅读 Debian Policy Manual, 6 "Package maintainer scripts and installation procedure" 并查看 **dh_make** 给出的示例文件。

如果你不听我的劝告，自己为一个软件包创建并定制了 maintainer scripts，你必须保证不仅测试 **install** 和 **upgrade**，还应测试 **remove** 和 **purge**。

升级到新版本应当是静默且非侵入式的(已有用户应当只在发现旧的 Bug 被修复或有新特性时注意到升级的变化)。

当更新必须以非静默模式进行时(例如分散在多个主目录中的配置文件都要改为完全不同的结构时)，作为最后的手段，你应该考虑将软件包设置到安全的回退状态(例如禁用服务)并按照 Debian Policy 提供相应的文档(**README.Debian** 和 **NEWS.Debian**)。不要在升级时使用 maintainer scripts 触发 **debconf** 来打扰用户。

ucf 软件包提供了 *类似 conffile* 对可能不标记为 *conffiles* 的文件的保留机制，比如由 maintainer scripts 来管理的配置文件。这可以把最大程度减少相关的问题。

这些 maintainer scripts 是 Debian 的增强特性，它们解释了 **人们为什么选择 Debian**。你必须非常小心，保证人们不因此产生烦恼。

5.19. *package*.examples

对于新维护者而言，打包一个库非常不易，因此不建议尝试。这样说吧，如果你的软件包有库，那你应该处理好 **debian/*package*.symbols** 文件。参见 第 A.2 节 “管理 **debian/*package*.symbols**”.

5.20. TODO

dh_installdocs(1) 命令会安装这个文件。

5.21. watch

watch 文件的格式被详述于 uscan(1) man 手册页中。**watch** 文件配置了 **uscan** 程序(它在 **devscripts** 中)以便监视你获得原始源代码的网站。它还被用于 Debian 软件包跟踪系统(PTS) 服务。

它的内容如下：

```
# watch control file for uscan
version=3
http://sf.net/gentoo/gentoo-(.+)\.tar\.gz debian uupdate
```

通常，在按照这个 **watch** 文件执行时，URL **http://sf.net/gentoo** 会被下载，程序会在其中搜索 **<a href=...>**。最后一个斜杠 **/** 后的基本名称会按照 Perl 正则表达式匹配(参看 perlre(1)) **gentoo-(.+)\.tar\.gz**。在所有匹配的文件里，将会下载带有最大版本号的，此后由 **uupdate** 程序创建更新的源代码树。

尽管上述内容对于所有站点都适用，但 SourceForge 下载服务(http://sf.net)仍是一个例外。当 **watch** 中包含匹配 Perl 正则表达式 **^http://sf\.net/** 的 URL 时，**uscan** 程序会将其替换为 **http://qa.debian.org/watch/sf.php/** 然后应用此规则。http://qa.debian.org/ 的 URL 重定向服务被设计用于提供一个稳定的重定向服务以满足 **watch** 文件中的 **http://sf.net/*project*/*tar-name*-(.+)\.tar\.gz** 形式，这样解决了 SourceForge 经常性的 URL 变更导致的问题。

如果上游提供 tarball 的加密签章，那么推荐校验其真实性，可以用 **pgpsigurlmangle** 选项来实现，这一点在 uscan(1) 中有描述。

5.22. source/format

在 **debian/source/format** 中只包含一行，写明了此源代码包的格式(查看 dpkg-source(1) 获得完整列表)。在 **squeeze** 后，它应该是以下二者之一：

- **3.0 (native)** - Debian 本土软件或者
- **3.0 (quilt)** - 其他所有软件

全新的 **3.0 (quilt)** 源代码格式将所有修改使用 **quilt** 补丁系列记录到 **debian/patches**。这些修改会在解压源代码包时自动应用。[59] Debian 修改保存于 **debian.tar.gz** 归档文件，其中包含了整个 **debian** 目录。这个新格式支持直接添加例如 PNG 图标等的二进制文件。[60]

dpkg-source 解压 **3.0 (quilt)** 格式的源码包时会自动应用所有列于 **debian/patches/series** 的补丁。你可以使用 **--skip-patches** 选项避免在解压后自动应用补丁。

5.23. source/local-options

如果你希望使用版本控制系统(VCS)管理 Debian 打包活动，你可以创建一个分支(例如叫做 **upstream**) 来跟踪上游代码，和另一个分支(对于 Git 而言典型的是 **master** 分支)来跟踪你的 Debian 软件包。对于后者，通常会将未应用补丁的上游代码和你的 **debian/*** 文件放在一起以便容易合并上游的新代码。

在编译软件包之后，源代码通常会被保持在应用补丁后的状态。你需要手工执行 **quilt pop -a** 来解除这些补丁，然后再提交到 **master** 分支。你可以向 **debian/source/local-options** 文件里添加一行 **unapply-patches** 来自动实现此目的。这个文件不会被加入到生成的源代码包，它只影响本地的编译构建行为。这个文件里还可以包含 **abort-on-upstream-changes** (参看 dpkg-source(1))。

```
unapply-patches
abort-on-upstream-changes
```

5.24. source/options

在源码树下自动生成的文件有时会超级讨厌，因为它们会生成毫无意义巨大无比的补丁文件。为了减轻这种问题，可以用一些定制模块比如 **dh_autoreconf**，参见 第 4.4.3 节 “定制 **rules** 文件”。

你可以给 **--extend-diff-ignore** 选项提供一个 Perl 正则表达式作为 dpkg-source(1) 的参数，以此忽略 在创建源码包时自动生成的文件所发生的变更。

作为这个自动生成文件的问题的通用解决办法 你可以存放像 **dpkg-source** 这样的选项参数到源码包的 **source/options** 文件中。如下将会跳过给 **config.sub**，**config.guess**，以及 **Makefile** 创建补丁文件。

```
extend-diff-ignore = "(^|/)(config\.sub|config\.guess|Makefile)$"
```

5.25. patches/*

旧的 `1.0` 源代码包格式使用单一的大 `diff.gz` 文件为源码保存 `debian` 中的维护文件和补丁。这样的软件包比较难于在事后检查和分析。这不是很好。

新的 `3.0 (quilt)` 源码格式将补丁存储在 `debian/patches/*` 中，用 **quilt** 命令。这些补丁和其他 `debian` 目录下的打包数据都会被打包成 `debian.tar.gz` 文件。由于 **dpkg-source** 命令可以处理 **quilt** 格式的补丁数据 (在格式为 `3.0 (quilt)` 的源码中)，而不需要 `quilt` 软件包；不需要在 `Build-Depends` 中添加 `quilt`。[61]

quilt 命令在 quilt(1) 中有详细描述。它将对源代码的修改维护于 `debian/patches` 中一系列 `-p1` 级别的补丁文件中，`debian` 目录外的文件没有任何修改。这些补丁的顺序记录于 `debian/patches/series` 文件中。你可以轻松地 apply (=push)、un-apply (=pop) 和 refresh 补丁。[62]

在 第 3 章 *修改源代码* 中，我们在 `debian/patches` 创建了三个补丁。

因为 Debian 补丁位于 `debian/patches`，请确定按照 第 3.1 节 “设置 **quilt**” 中的方法正确配置 **dquilt**。

如果在之后有人(包括你自己)提供了一个补丁 *`foo`*`.patch`，对于 `3.0 (quilt)` 源代码包格式可以很容易修改：

```
$ dpkg-source -x gentoo_0.9.12.dsc
$ cd gentoo-0.9.12
$ dquilt import ../foo.patch
$ dquilt push
$ dquilt refresh
$ dquilt header -e
... describe patch
```

用新的 `3.0 (quilt)` 源代码包格式存储的补丁*必须*有 *清晰的边界*。你应该通过 `dquilt pop -a; while quilt push; do quilt refresh; done` 来验证这点。

---

[54] 在本章节中，只要不产生歧义，所有提及的 `debian` 目录下的文件均会省去 `debian/` 前缀以求简洁方便。

[55] 参见 dpkg(1) 以及 Debian Policy Manual, “D.2.5 Conffiles”.

[56] 这取代了已经废弃的 dh_movefiles(1) 命令，它本是用 `files` 文件所配置的。

[57] 注意，**help2man** 的占位符性质 man 手册页会声称 在 info 系统中有着更为详尽的细节。如果命令 缺少 **info** 页，那你应该手工编辑由 **help2man** 命令创建的 man 手册页。

[58] 尽管这里使用 **bash** 表达式速记法 `{pre,post}{inst,rm}` 来指示这些文件名，但你应该使用 纯 POSIX 语法来编写 maintainer scripts，这是为了兼容 **dash** 作为系统 shell 的情况。

[59] 参看 DebSrc3.0 以了解转换到新的 `3.0 (quilt)` 和 `3.0 (native)` 源代码格式时的总结。

[60] 实际上新格式还支持多个上游 tarball 和更多的压缩方法，但这已超出本文档讲述的范围。

[61] 这里已经提出了若干种补丁集维护方法，并且正为 Debian 软件包所采用。**quilt** 系统是最推荐的维护系统。而其他的有包括 **dpatch**, **dbs**, 以及 **cdbs**。其中有许多将 补丁保存为 `debian/patches/*` 文件。

[62] 如果你需要一个 sponsor 上传你的软件包，这种情况下，清晰的软件包分离和记录更改的文档对于软件包审查过程的顺利程度非常重要。