第 8 章 更新软件包

目录

一旦你发布了一个软件包，在之后的某个时间里就需要对它进行更新。

8.1. 新的 Debian 版本

假设你收到一个针对你的软件包报告的 Bug，其编号为 **#654321**，它描述了一个你可以解决的问题。要创建软件包的一个新 Debian 修订版本，你需要：

- 如果要将它记录于新的补丁中，这样做：
  - **dquilt new *bugname.patch*** 设置补丁名称；
  - **dquilt add *buggy-file*** 声明文件将被更改；
  - 修正软件包代码中的上游 Bug；
  - **dquilt refresh** 将修改记录到 ***bugname.patch***；
  - **dquilt header -e** 添加对它的描述；
- 如果是更新一个已存在的补丁，这样做：
  - **dquilt pop *foo.patch*** 重现已存在的 ***foo.patch***；
  - 修正旧的 ***foo.patch*** 中的问题；
  - **dquilt refresh** 更新 ***foo.patch***；
  - **dquilt header -e** 更新对它的描述；
  - **while dquilt push; do dquilt refresh; done** 应用所有补丁以确保它们 *边界清晰*；
- 在 Debian **changelog** 文件的顶部添加一个条目。例如可以使用 **dch -i** 或用 **dch -v *version-revision*** 来指定版本，然后用你喜欢的编辑器插入信息。[77]
- 在 changlog 条目中简要描述 Bug 和相应的解决办法，并在后面添加 **Closes: #654321**。这样 Bug 报告会在你的软件包被 Debian 仓库接受的同时被仓库管理软件 *自动关闭*。
- 重复上述操作来修复更多的 Bug，并在需要的时候使用 **dch** 更新 Debian **changelog** 文件。
- 重复你在 第 6.1 节 “完整的(重)构建” 和 第 7 章 *检查软件包中的错误* 中所做的事情。
- 一旦你满意了，那就将 **changelog** 中的发行版值由 **UNRELEASED** 修改成 目标发行版值 **unstable** (或者是 **experimental**)。[78]
- 按照 第 9 章 *上传软件包* 来上传软件包。惟一的区别是这次不需要再包含原始代码文件，因为它们没有变化且已经存在于 Debian 仓库中。

有一种棘手的情况，当你在上传正常版本到官方仓库中之前，你制作了一个本地包以进行打包实验，例如，***1.0.1-1***。为了平滑升级，创建一个 **changelog** 条目，其中包含类似 ***1.0.1-1~rc1*** 这样的版本字符串不失为一剂良方。你可以通过合并这样的本地修改条目到官方包的单个条目中来整理 **changelog**。参见 第 2.6 节 “软件包名称和版本” 来了解版本字符串的排序。

8.2. 检查新上游版本

在为 Debian 仓库准备新上游版本的软件包前，你必须首先对新的上游发布版本进行检查。

检查工作应从阅读上游 **changelog**、**NEWS** 以及所有随新版本一同发布的文档。

然后应按照以下步骤检查新旧版本之间源码的差别，小心任何可疑的内容：

```
$ diff -urN foo-oldversion foo-newversion
```

对于 Autotools 自动生成的文件发生的改动，例如 **missing**、**aclocal.m4**、**config.guess**、**config.h.in**、**config.sub**、**configure**、**depcomp**、**install-sh**、**ltmain.sh** 和 **Makefile.in** 是可以忽略的。你可以在运行 **diff** 进行代码检查前删除它们。

8.3. 新上游版本

如果软件包 ***foo*** 是使用新的 **3.0 (native)** 或 **3.0 (quilt)** 格式打包的，制作新的上游版本时需要先把旧的 **debian** 目录移至新的源代码内。这可以通过在新解压的源代码目录里运行 **tar xvzf */path/to/foo_oldversion*.debian.tar.gz** 完成。[79]当然，你需要做几个很显然的杂事：

- 创建一份上游源代码的副本，命名为 ***foo_newversion*.orig.tar.gz**。
- 使用 **dch -v *newversion-1*** 更新 Debian **changelog** 文件。
  - 添加一个条目，内容为 **New upstream release**。
  - 简明地介绍 *在新上游版本中* 上游修复和关闭的 Bug (添加 **Closes: #*bug_number***) 。
  - 简明地介绍维护者对 *本个新上游版本* 做出的修改，修复和关闭的 Bug (添加 **Closes: #*bug_number***)。
- 运行 **while dquilt push; do dquilt refresh; done** 以应用全部补丁并使它们 *边界清晰*。

如果补丁没有干净地被应用，检查原因(线索在 **.rej** 文件里)。

- 如果你的补丁已经被上游接受，
  - 使用 **dquilt delete** 删除它。
- 如果你的补丁与上游代码中的变更有冲突：
  - 使用 **dquilt push -f** 应用旧补丁，未应用的部分会被保存为 ***baz*.rej**。
  - 手工编辑 ***baz*** 文件来在新的代码中实现 ***baz*.rej** 中应有的效果。
  - 使用 **dquilt refresh** 更新补丁。
- 正常继续，执行 **while dquilt push; do dquilt refresh; done**。

这个过程可以通过使用 uupdate(1) 来更自动化地完成：

```
$ apt-get source foo
...
dpkg-source: info: extracting foo in foo-oldversion
dpkg-source: info: unpacking foo_oldversion.orig.tar.gz
dpkg-source: info: applying foo_oldversion-1.debian.tar.gz
$ ls -F
foo-oldversion/
foo_oldversion-1.debian.tar.gz
foo_oldversion-1.dsc
foo_oldversion.orig.tar.gz
$ wget http://example.org/foo/foo-newversion.tar.gz
$ cd foo-oldversion
$ uupdate -v newversion ../foo-newversion.tar.gz
$ cd ../foo-newversion
$ while dquilt push; do dquilt refresh; done
$ dch
... document changes made
```

如果你按照 第 5.21 节 “**watch**” 的叙述设置了 **debian/watch** 文件，你可以跳过这个 **wget** 命令，转而在 ***foo-oldversion*** 目录中运行 uscan(1)，且无需再执行 **uupdate** 命令。它会 *自动* 查找新的源代码、下载并运行 **uupdate** 命令。[80]

重复 第 6.1 节 “完整的(重)构建”、第 7 章 *检查软件包中的错误* 和 第 9 章 *上传软件包* 中的操作，即可发布此更新的软件包。

8.4. 更新打包风格

更新打包风格不是更新软件包的必须步骤，但是这样可以使你的软件包得到对现代的 **debhelper** 系统和 **3.0** 源代码包格式完整的兼容性。[81]

- 如果你需要重新添加已删除的模板文件，可以在同一个 debian 软件包源代码树中运行 **dh_make**，并添加 **--addmissing** 选项。然后对模板进行相应的编辑。
- 如果软件包的 **debian/rules** 文件没有更新为使用 **debhelper** v7+ 的 **dh** 语法，则更新它使用 **dh**。在需要的时候更新 **debian/control** 文件。
- 如果你希望将使用 **cdbs** 的 **Makefile** 包含机制创建的 **rules** 文件更新为 **dh** 语法，参看下文并理解各 **DEB_*** 配置变量。
  - **/usr/share/doc/cdbs/cdbs-doc.pdf.gz** 的本地副本
  - The Common Debian Build System (CDBS), FOSDEM 2009
- 如果你有一个不带有 ***foo*.diff.gz** 文件的 **1.0** 格式的源代码包，你可以通过创建 **debian/source/format** 文件并在其中添加 **3.0 (native)** 来将其更新为新的 **3.0 (native)** 源代码包格式。**debian** 目录中的其他文件可以直接复制过来。
- 如果你有一个带有 ***foo*.diff.gz** 文件的 **1.0** 格式的源代码包，你可以通过创建 **debian/source/format** 文件并在其中添加 **3.0 (quilt)** 来将其更新为新的 **3.0 (quilt)** 源代码包格式。**debian** 目录中的其他文件可以直接复制过来。如果需要，把 **filterdiff -z -x '*/debian/*' *foo*.diff.gz > big.diff** 生成的 **big.diff** 文件导入到 **quilt** 系统。[82]
- 如果它使用了其他的补丁系统，例如 **dpatch**、**dbs** 或 **cdbs**，使用 **-p0**、**-p1** 或 **-p2** 级别，使用 http://bugs.debian.org/581186 的 **deb3** 命令将其转换到 **quilt** 系统。
- 如果它使用 **dh** 命令的 **--with quilt** 选项，或 **dh_quilt_patch** 和 **dh_quilt_unpatch** 命令，删除它们并使其使用新的 **3.0 (native)** 源代码包格式。

你应当查看 DEP - Debian Enhancement Proposals 并采纳 ACCEPTED 建议。

当然你还需要按照 第 8.3 节 “新上游版本” 完成其他的步骤。

8.5. UTF-8 转换

如果上游文档采用了老式编码，那么将其转换为 UTF-8 不失为一良方。

- 用 iconv(1) 来转换普通文本文件的编码。

```
iconv -f latin1 -t utf8 foo_in.txt > foo_out.txt
```

- 使用 w3m(1) 来把 HTML 文件转换为 UTF-8 普通文本文件。当你这样做的时候，请确认在 UTF-8 locale 下执行。

```
LC_ALL=en_US.UTF-8 w3m -o display_charset=UTF-8 \
        -cols 70 -dump -no-graph -T text/html \
        < foo_in.html > foo_out.txt
```

8.6. 对更新软件包的几点提示

以下是对更新软件包的几点提示：

- 保留旧的 **changelog** 条目(看似显然，但是总有可能把 **dch -i** 输入为 **dch**)。
- 已存在的 debian 修改需要被重新校验，去除上游已经合并的东西 (一种或多种形式)，除非有必要的原因，还要保留尚未被上游接受的部分。
- 如果对编译系统作出了修改(希望你已经在检查上游变更时了解了这些)，那么要在必要时更新 **debian/rules** 和 **debian/control** 编译依赖关系。
- 检查 Debian Bug Tracking System (BTS) 是否有人为某些仍然未修复的 bug 提供了补丁。
- 检查 **.changes** 文件以确保你正要上传到正确的发行版、正确的列出BUG关闭 **Closes** 字段、**Maintainer** 和 **Changed-By** 字段相匹配，以及文件是否已经使用 GPG 签署等。

---

[77] 要获得需要的日期格式，使用 **LANG=C date -R**。

[78] 如果你用 **dch -r** 命令来使它成为最后一笔更改，请确保用编辑器显式地保存 **changelog** 文件。

[79] 如果软件包 ***foo*** 是使用旧的 **1.0** 格式的，可以在新解压的源代码目录里运行 **zcat */path/to/foo_oldversion*.diff.gz|patch -p1** 来完成。

[80] 如果 **uscan** 命令下载并更新了源代码，但没有运行 **uupdate** 命令，你应该修正 **debian/watch** 文件，使 URL 末尾后带有 **debian uupdate**。

[81] 如果你的 sponsor 或其他维护者一定反对更新已有的打包风格，则不值得去为此烦恼或争论，总是有更重要的事要做。

[82] 你可能使用 **splitdiff** 命令将 **big.diff** 分割为多个增量补丁。