第 7 章 检查软件包中的错误

目录

7.1. 诡异可疑的改动
7.2. 校验软件包安装过程
7.3. 检验软件包的 maintainer scripts
7.4. 使用 lintian
7.5. debc 命令
7.6. debdiff 命令
7.7. interdiff 命令
7.8. mc 命令

在上传软件包到公共仓库前，你还需要知道一些检查软件包错误的技巧。

不仅在自己的机器上测试总是一个好主意。你必须谨慎地对待以下叙述的测试中显示的任何一个警告或错误信息。

7.1. 诡异可疑的改动

如果你在构建以 `3.0 (quilt)` 格式的非本土 Debian 软件包后，发现一个新的自动生成的补丁，比如 `debian-changes-*` 在 `debian/patches` 目录中有可能是你不小心更改了一些文件，或者构建脚本修改了上游源代码。如果这是你犯下的小错误，那就修复它。如果这是构建脚本干的好事，那就用 **dh-autoreconf** 来解决其根本问题，可参照 第 4.4.3 节 “定制 `rules` 文件” 或者可以变通一下，处理 `source/options` 文件，参照 第 5.24 节 “`source/options`”.

7.2. 校验软件包安装过程

你必须测试你的软件包看是否存在安装问题。debi(1) 命令可以帮助你测试所有生成的二进制软件包。

```
$ sudo debi gentoo_0.9.12-1_i386.changes
```

你必须使用从 Debian 仓库下载的 `Contents-`*`i386`* 文件校验是否在与不同包存在文件冲突，以阻止在不同的系统上发生安装故障。**apt-file** 命令正适合完成这个任务。如果存在冲突，请通过重命名、将公共文件分离到另一个受其他包依赖的包中、与受影响的软件包的维护者合作使用 alternatives 机制来避免实际问题(参看 update-alternatives(1))或在 `debian/control` 文件中设置 `Conflicts` 条目以声明冲突关系等方式避免问题的发生。

7.3. 检验软件包的 maintainer scripts

所有 maintainer scripts，包括 `preinst`、`prerm`、`postinst` 和 `postrm` 文件，都非常难以编写，除非是由 `debhelper` 程序自动生成的。如果你是新维护人员则不要使用它们(参看 第 5.18 节 “`{pre,post}{inst,rm}`”)。

如果软件包使用了这些需要严格测试的 maintainer scripts，请确保不仅测试 install，还要测试 remove、purge 和 upgrade。很多 maintainer scripts 的 Bug 都显现于卸载或彻底删除软件包时。使用 **dpkg** 命令按以下方法来测试它们：

```
$ sudo dpkg -r gentoo
$ sudo dpkg -P gentoo
$ sudo dpkg -i gentoo_version-revision_i386.deb
```

整个测试过程应按照以下操作序列完成：

- 如果可能，安装前一个版本的软件包；
- 从前一个版本升级软件包；
- 降级软件包到前一个版本(可选)；
- 彻底删除该软件包；
- 全新安装该软件包；
- 卸载该软件包；
- 再次安装该软件包。
- 彻底删除该软件包；

如果这是你的第一个软件包，你应该使用其他版本号创建一个测试用的软件包来完成升级测试，这样可以避免将来的问题。

请牢记如果你的软件包已经在以往版本的 Debian 中发布，人们通常会从最近发布的 Debian 发布里的版本升级，所以也要测试从那个版本升级到当前的版本。

尽管降级没有被正式支持，支持它也总是友好的。

7.4. 使用 `lintian`

使用 lintian(1) 检查你的 `.changes` 文件。**lintian** 命令会运行很多测试脚本来检查常见的打包错误。[76]

```
$ lintian -i -I --show-overrides gentoo_0.9.12-1_i386.changes
```

当然，要替换你自己软件包所生成的 `.changes` 文件的文件名。**lintian** 命令的输出常带有以下几种标记：

- `E:` 代表错误：确定违反了 Debian Policy 或是一个肯定的打包错误。
- `W:` 代表警告：可能违反了 Debian Policy 或是一个可能的打包错误。
- `I:` 代表信息：对于特定打包类别的信息。
- `N:` 代表注释：帮助你调试的详细信息。
- `O:` 代表已覆盖：一个被 `lintian-overrides` 文件覆盖的信息，但由于使用 `--show-overrides` 选项而显示。

对于警告，你应该改进软件包或者检查警告是否的确无意义。如果确定没有意义，则按照 第 5.14 节 “`{`*`package`*`,source/}lintian-overrides`” 中的叙述使用 `lintian-overrides` 文件将其覆盖。

注意，你可以用 **dpkg-buildpackage** 来构建软件包，并执行 **lintian**，只要你使用了 debuild(1) 或 pdebuild(1).

7.5. debc 命令

你可以使用 debc(1) 命令列出一个二进制 Debian 软件包中的文件。

```
$ debc package.changes
```

7.6. debdiff 命令

你可以使用 debdiff(1) 命令比较两个 Debian 源代码包的内容。

```
$ debdiff old-package.dsc new-package.dsc
```

你还可以使用 debdiff(1) 命令比较两个 Debian 二进制包的文件列表。

```
$ debdiff old-package.changes new-package.changes
```

这个命令对于检查源代码包中哪些文件被修改了非常有用，还可以发现二进制包中是否有文件在更新过程中发生的变动，比如被意外替换或删除。

7.7. interdiff 命令

你可以使用 interdiff(1) 命令比较两个 `diff.gz` 文件。这对于更新使用旧的 `1.0` 源代码格式的软件包时，检查是否有意外的变更非常有用。

```
$ interdiff -z old-package.diff.gz new-package.diff.gz
```

新的 `3.0` 源码格式会将更改保存在多个补丁文件中，如 第 5.25 节 “`patches/*`” 所述。你也可以使用 **interdiff** 跟踪 每一个 `debian/patches/*` 文件中的改动。

7.8. mc 命令

很多文件检查操作可以通过使用类似 mc(1) 的文件管理器来完成，它可以帮助你直接查看 `*.deb` 文件的内容，除此之外还可以用于 `*.udeb`、`*.debian.tar.gz`、`*.diff.gz` 和 `*.orig.tar.gz` 文件。

还要检查在二进制包和源代码包中是否有不需要的文件或者空文件。这些文件经常没有被正确清理，如果存在这种情况，要调整 `rules` 文件进行处理。

---

[76] 如果你按照 第 6.3 节 “debuild 命令” 中的叙述定义了 `/etc/devscripts.conf` 或 `~/.devscripts` 文件，就不需要再添加 **lintian** 选项 `-i -I --`

show-overrides。