第 3 章 修改源代码

目录

请注意这里没有足够篇幅来描述有关修改上游源码的 *全部* 细节，这里只介绍基本步骤和常见问题。

3.1. 设置 quilt

quilt 程序是 Debian 打包过程中采用的补丁管理工具。我们只需要在默认配置的基础上，加以少许修改即可。首先我们来创建一个别名 **dquilt**，以方便打包之需：添加以下几行内容到 **~/.bashrc** 文件中。其中第二行可以给 **dquilt** 命令提供与 **quilt** 命令相同的 shell 补全：

```
alias dquilt="quilt --quiltrc=${HOME}/.quiltrc-dpkg"
complete -F _quilt_completion -o filenames dquilt
```

现在我们来创建 **~/.quiltrc-dpkg** 文件：

```
d=. ; while [ ! -d $d/debian -a $(readlink -e $d) != / ]; do d=$d/..; done
if [ -d $d/debian ] && [ -z $QUILT_PATCHES ]; then
    # if in Debian packaging tree with unset $QUILT_PATCHES
    QUILT_PATCHES="debian/patches"
    QUILT_PATCH_OPTS="--reject-format=unified"
    QUILT_DIFF_ARGS="-p ab --no-timestamps --no-index --color=auto"
    QUILT_REFRESH_ARGS="-p ab --no-timestamps --no-index"
    QUILT_COLORS="diff_hdr=1;32:diff_add=1;34:diff_rem=1;31:diff_hunk=1;33:diff_ctx=35:diff_cctx=33"
    if ! [ -d $d/debian/patches ]; then mkdir $d/debian/patches; fi
fi
```

参见 quilt(1) 以及 **/usr/share/doc/quilt/quilt.pdf.gz** 来获取有关 **quilt** 命令用法的信息。

3.2. 修复上游 Bug

假设你在上游的 **Makefile** 文件中找到了一个错误，其中的 **install: gentoo** 应该修正为 **install: gentoo-target** 。

```
install: gentoo
        install ./gentoo $(BIN)
        install icons/* $(ICONS)
        install gentoorc-example $(HOME)/.gentoorc
```

使用 **dquilt** 修复这个问题，并把补丁命名为 **fix-gentoo-target.patch**。[22]

```
$ mkdir debian/patches
$ dquilt new fix-gentoo-target.patch
$ dquilt add Makefile
```

现在将 **Makefile** 修改为如下的样子：

```
install: gentoo-target
        install ./gentoo $(BIN)
        install icons/* $(ICONS)
        install gentoorc-example $(HOME)/.gentoorc
```

使用 **dquilt** 将补丁创建到 **debian/patches/fix-gentoo-targe.patch** 并根据 DEP-3: Patch Tagging Guidelines 添加描述：

```
$ dquilt refresh
$ dquilt header -e
... 描述补丁
```

3.3. 文件安装

大多数第三方程序会默认安装在 **/usr/local** 目录下。在 Debian 中，这是保留给系统管理员的私有位置，因此 Debian 软件包不可以使用比如 **/usr/local/bin** 这样的目录，而应当使用比如 **/usr/bin** 这样的系统目录，以遵循文件系统层级结构标准：Filesystem Hierarchy Standard (FHS)。

通常情况下，make(1) 被用来自动编译程序，接着执行 **make install** 就可以把程序按照 **Makefile** 文件中的 **install** target 安装到指定的位置。为使 Debian 能够提供编译好的二进制软件包，打包脚本将自动修正编译系统，使其将文件安装到一个在临时目录中创建的文件系统子树中，而非直接安装到实际的目标位置。

普通程序安装过程和 Debian 打包安装过程二者的区别可以由 **debhelper** 软件包中的 **dh_auto_configure** 和 **dh_auto_install** 透明地处理。但必须满足以下条件：

- **Makefile** 文件应当遵循 GNU 的规定支持 **$(DESTDIR)** 变量[23]
- 源代码必须遵循文件系统层级标准(FHS)。

使用 GNU **autoconf** 的程序默认遵守 GNU 的规定，这有利于打包过程的自动化。通过这项特点和其他启发式处理，估计 **debhelper** 软件可以直接制作约 90% 的软件包而不需维护者对编译系统做出大的改变。所以打包也不是看起来那样复杂。

如果你需要修改 **Makefile** 文件，就要确保其支持 **$(DESTDIR)** 变量，虽然默认情况下 **$(DESTDIR)** 变量没有设置并且在程序安装时会前置到每个文件的路径中。打包脚本会将 **$(DESTDIR)** 设置到临时目录上。

对于从源码生成单个二进制包的情况，**dh_auto_install** 将临时目录设置到 **debian/*package***。[24] 临时目录中的全部文件都将成为软件包内容，并在安装该包时被安装到用户系统。这里唯一的区别是 **dpkg** 会把文件安装到真实的根目录树中，而不是你的工作目录。

请记住，即使你的程序正确安装到了 **debian/*package***，仍然要考虑将 **.deb** 软件包文件安装到根目录下的情形。所以绝对不允许构建系统将诸如 **/home/me/deb/*package*-*version*/usr/share/*package*** 这种诡异的内容硬编码到软件包文件中。

以下是 **gentoo** 软件包的 **Makefile** 文件中的相关部分[25]：

```
# Where to put executable commands on 'make install'?
BIN     = /usr/local/bin
# Where to put icons on 'make install'?
ICONS   = /usr/local/share/gentoo
```

可以看到文件被放到了 **/usr/local** 下。按照上边的解释，该目录被 Debian 保留作本地用途，所以请把它们按如下方式修改：

```
# Where to put executable commands on 'make install'?
BIN     = $(DESTDIR)/usr/bin
# Where to put icons on 'make install'?
ICONS   = $(DESTDIR)/usr/share/gentoo
```

二进制文件、图标和文档等的更确切位置均已在文件层级标准(FHS)中作出了详尽描述。本教程建议你快速浏览相关章节以获取你打包需要用到的内容。

因此，我们应当把可执行二进制文件安装到 **/usr/bin** 而非 **/usr/local/bin**，而 man 手册页则应放在 **/usr/share/man/man1** 而非 **/usr/local/man/man1**，依此类推。注意，**gentoo** 的 **Makefile** 里没有提及手册页，而按照 Debian Policy 的要求，每个程序都应当有一个手册页，我们将在稍后制作一个并安装到 **/usr/share/man/man1**。

有些程序不使用 **Makefile** 变量定义路径，这意味着你可能需要去编辑 C 程序源代码来使他们使用正确的路径。但是到哪里去搜索，哪些才是呢？你可以通过以下的方法找到它们：

```
$ grep -nr --include='*.[ch]' -e 'usr/local/lib' .
```

grep 会递归搜索整个源代码树并告诉你所有匹配项的文件名和行号。

编辑那些文件，在那些行中用 **usr/lib** 替换 **usr/local/lib**。这个过程可以用如下方法自动化完成：

```
$ sed -i -e 's#usr/local/lib#usr/lib#g' \
        $(find . -type f -name '*.[ch]')
```

如果你想要确认每一个替换操作，那么下边的方法可以让你交互式地达成：

```
$ vim '+argdo %s#usr/local/lib#usr/lib#gce|update' +q \
        $(find . -type f -name '*.[ch]')
```

紧接着你应该找到 **install** target (通常搜索以 **install:** 开头的行即可)，并把除 **Makefile** 顶部定义变量之外的目录引用妥当修改。

原始的 **gentoo** 的 **install** target 是这样：

```
install: gentoo-target
        install ./gentoo $(BIN)
        install icons/* $(ICONS)
        install gentoorc-example $(HOME)/.gentoorc
```

让我们来修复这个上游BUG，并把修改使用 **dquilt** 命令记录到**debian/patches/install.patch**。

```
$ dquilt new install.patch
$ dquilt add Makefile
```

使用你喜欢的编辑器按照以下内容为 Debian 软件包作修改：

```
install: gentoo-target
        install -d $(BIN) $(ICONS) $(DESTDIR)/etc
        install ./gentoo $(BIN)
        install -m644 icons/* $(ICONS)
        install -m644 gentoorc-example $(DESTDIR)/etc/gentoorc
```

你一定会注意到规则里在其他命令前有了一个 **install -d** 命令。原始的 **Makefile** 文件中没有它，因为通常情况下当你执行 **make install** 命令时，**/usr/local/bin** 和用到的其他目录早已存在于系统中。然而当我们要向新建的私有目录树中安装时，我们必须创建其中的每一个目录。

我们还可以在末尾添加上其他的内容，比如上游作者有时会省略的附加文档：

```
        install -d $(DESTDIR)/usr/share/doc/gentoo/html
        cp -a docs/* $(DESTDIR)/usr/share/doc/gentoo/html
```

仔细检查后如果没有问题，使用 **dquilt** 创建 **debian/patches/install.patch** 补丁文件并添加对它的描述：

```
$ dquilt refresh
$ dquilt header -e
... 描述补丁
```

现在你有了一系列的补丁。

1. 修复上游 Bug:**debian/patches/fix-gentoo-target.patch**
2. Debian 特有的打包时修改:**debian/patches/install.patch**

当进行任何非 Debian 特有的修改时，比如 **debian/patches/fix-gentoo-target.patch**，一定要向上游作者进行反馈，以便上游作者方便在下一版本中以使更多人受益。同时请记住不要做出特别针对 Debian 或 Linux (甚至是 Unix！)的修改，要使其可以移植，这会使你的修改更容易被接受。

注意你不一定要把 **debian/*** 都提交到上游。

3.4. 不同的库名称

还有另外一个常见的问题：不同平台之间的库名称常常因平台而异。例如一个 **Makefile** 中可能引了用一个 Debian 系统中不存在的库。这种情况下我们需要将其修改为 Debian 中存在的、提供完全相同功能的库。

如果你手中程序的 **Makefile**(或 **Makefile.in**)中有如下的行。

```
LIBS = -lfoo -lbar
```

如果你的程序由于 **foo** 库不存在，而 Debian 系统中的 **foo2** 库提供了其等效，那么你可以修复这个构建问题并将修改记录到 **debian/patches/foo2.patch** 中，只需要将 **foo** 切换到 **foo2**:[26]

```
$ dquilt new foo2.patch
$ dquilt add Makefile
$ sed -i -e 's/-lfoo/-lfoo2/g' Makefile
$ quilt refresh
$ quilt header -e
... 描述补丁
```

---

[22] **debian/patches** 目录应当是运行 **dh_make** 时生成的。因为假设的是在更新一个已存在的软件包，所以在这个例子中我们新建它。

[23] 参见 GNU Coding Standards: 7.2.4 DESTDIR: Support for Staged Installs。

[24] 对于单份源码生成多个二进制包的情况，**dh_auto_install** 将临时目录设置为**debian/tmp**，而 **dh_install** 命令则将文件按照 **debian/*package-1*.install** 和 **debian/*package-2*.install** 等文件的描述将 **debian/tmp** 中的文件分别装入 **debian/*package-1*** 和 **debian/*package-2*** 等临时目录，用以创建 ***package-1*_*.deb** 和 ***package-2*_*.deb** 等二进制软件包。

[25] 这只是一个演示 **Makefile** 正常形态的例子。如果 **Makefile** 是通过 **./configure** 命令生成的，那么修复该类 **Makefile** 的方法是通过 **dh_auto_configure** 来执行 **./configure**，并带上包括 **--prefix=/usr** 的默认选项。

[26] 如果从 **foo** 库切换到 **foo2** 库时要更改应用程序接口(API)，这就要求我们修改源代码来符合新的 API。