



银河麒麟桌面操作系统 V10-DEB 包 开发者指南

麒麟软件有限公司

生态发展中心

2025 年 12 月 08 日

版本说明

版本号	版本说明	作者	日期	变更内容
V1.0	首次发布	于文洁	2021-07-26	首次创建
V1.1	内容变更	都熔佳	2021-09-13	增加 3.4.2 章对 control 文件字段说明
V1.2	模板变更	刘佳鑫	2022-11-30	模板变更
V1.3	内容变更	冯培培	2025-10-23	变更联系方式
V1.4	模板变更	冯培培	2025-12-08	模板变更

目 录

1 开发环境	3
1.1 需要的开发工具	3
1.2 设置电子邮件地址	3
1.3 可参考的开发文档	4
2 源码包打包成 deb 包举例	4
2.1 准备源码文件	4
2.1.1 新建打包目录	4
2.2 编写源代码	4
2.3 编写 Makefile	5
2.4 编译	6
2.5 验证程序	6
2.6 打包成二进制压缩包	6
3 二进制软件包打包成 deb 包举例	7
3.1 准备打包目录	7
3.1.1 创建构建 deb 包的目录	7
3.1.2 构建规范的软件目录	7
3.1.3 解压放置软件包	8
3.1.4 放置图标	8
3.2 创建.desktop 文件	8
3.2.1 新建.desktop 文件	8
3.2.2 desktop 语法解释	9
3.3 创建 DEBIAN 目录	10
3.4 修改 debian 目录下文件	11
3.4.1 修改 control 文件	11
3.4.2 control 文件几个字段说明	11
3.4.3 创建 install 文件	12
3.4.4 修改 rules 文件	12
3.5 构建软件包	13
3.6 验证 deb 包	14
4 联系我们	16
附录一 《麒麟桌面 V10-DEB 软件包打包规范》	16

本文主要介绍将源码或二进制软件包打成符合麒麟桌面 V10 软件商店上架规范的 deb 包的方法；如果您打算采用源码的方式进行打包，请参考第 2 和第 3 章节；如果您已经将源码包打包成二进制软件包，只是想将二进制软件包打成符合麒麟桌面 V10 软件商店上架规范的 deb 包，请参考第 3 章节。

本文中只是介绍了众多打包方式中比较常见的一种，仅供参考使用；示例中的很多打包规则都可以使用其他方法替代，比如本文中 debian 目录是通过 dh_make 生成，也可以直接编写或使用 debmake 生成。

1 开发环境

1.1 需要的开发工具

软件打包前需要安装：

```
$ sudo apt update
$ sudo apt-get install dh-make build-essential devscripts debhelper
```

1.2 设置电子邮件地址

许多 Debian 维护工具识别并使用 shell 环境变量 \$DEBEMAIL 和 \$DEBFULLNAME 作为 debian/control 文件中的电子邮箱和名称。

我们可以通过设置 ~/.bashrc 的方式对这些软件进行 shell 环境变量设置。

```
$ cat >> ~/.bashrc <<EOF
DEBEMAIL=" your.email.address@example.org"
DEBFULLNAME=" Firstname Lastname"
export DEBEMAIL DEBFULLNAME
EOF
$ . ~/.bashrc
```

其中，DEBEMAIL = "your.email.address@ example.org" 中 your .email. address @example.org 更换为你自己的邮件地址，DEBFULLNAME = "Firstname Lastname" 中，Firstname Lastname 则输入维护者的名字。

1.3 可参考的开发文档

这里提供一些网址可能会有所帮助。

Debian 新维护者手册，里面有更详细的 debian 软件打包过程。

<https://www.debian.org/doc/manuals/maint-guide/>

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。

<https://www.debian.org/doc/devel-manuals#policy>

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。

<https://www.debian.org/doc/devel-manuals#devref>

2 源码包打包成 deb 包举例

这里给出了从简单的 C 语言源代码创建简单的 deb 软件包的例子，并假设使用 Makefile 作为构建系统的工具。

2.1 准备源码文件

2.1.1 新建打包目录

新建一个用于打包的目录并进入，这里需要注意的是文件夹构成：包名-版

本号

```
$ mkdir -p hello-1.0.0/src  
$ cd hello-1.0.0/src
```

2.2 编写源代码

这里准备了一个简单的 hello world 示例程序，源代码就是所有学 C 的朋

友都写过的 hello world。

```
~/hello-1.0.0/src$ ls
~/hello-1.0.0/src$ cat hello_world.c
#include <stdio.h>
void main(void)
{
    printf("hello world!\r\n");
}
```

2.3 编写 Makefile

源码编译需要一个 Makefile 来指导 make 的编译，下面是一个简单的

Makefile 示例：

```
~/hello-1.0.0/src$ cd ..
~/hello-1.0.0$ cat Makefile
PREFIX=/opt/apps/hello
all: prepare build-bin

prepare:
    mkdir -p bin

build-bin:
    $(CC) src/hello_world.c -o bin/hello_world

install:
    mkdir -p ${DESTDIR}${PREFIX}/bin
    install -v bin/hello_world ${DESTDIR}${PREFIX}/bin

clean:
    rm -rf bin
    rm -rf ${DESTDIR}${PREFIX}

.PHONY: all clean
```

在 Makefile 文件中，指定编译目标，install 命令、clean 命令，此示例中编译过程中只会用到 build-bin, 其实就是将上面编写的源代码编译生成一个二进制文件；install 和 clean 命令只有在使用源码包安装和卸载此程序时才会用到，此示例中不涉及。

2.4 编译

执行 `make` 进行编译，得到二进制可执行程序 `hello_world`

```
~/hello-1.0.0$ make
mkdir -p bin
cc src/hello_world.c -o bin/hello_world
~/hello-1.0.0$ tree
.
├── bin
│   └── hello_world
├── Makefile
└── src
    └── hello_world.c

2 directories, 3 files
```

2.5 验证程序

编译完成后需要对二进制可执行程序进行验证，保证程序能够正常运行

```
~/hello-1.0.0$ ./bin/hello_world
hello world!
```

此时可以看到，程序能够正常运行，可以进行打包的后续操作，如果程序运行失败需要排查问题后再进行后续打包操作。

2.6 打包成二进制压缩包

自动构建 `deb` 包指令需要用户将源码包打包成压缩包，以 `$PKGNAME-$VERSION.tar.gz` 的形式对包进行命名，这个命名形式是严格要求的，这里我们使用 `tar` 指令进行打包：

```
~/hello-1.0.0$ cd ..
$ tar -czvf hello-1.0.0.tar.gz hello-1.0.0/
hello-1.0.0/
hello-1.0.0/Makefile
hello-1.0.0/src/
hello-1.0.0/src/hello_world.c
hello-1.0.0/bin/
hello-1.0.0/bin/hello_world
```

此时我们就会得到二进制软件压缩包 `hello-1.0.0.tar.gz`，所以后续的打包过程可以看成是二进制软件压缩包打成 `deb` 包的过程。

3 二进制软件包打包成 deb 包举例

本文的实例包为 jetbrains 公司的 IDE—Pycharm, [Pycharm 下载地址](#)

下载完后,我们可以得到一个名为 pycharm-professional-2020.3.tar.gz 的二进制压缩包,接下来我们来将他构建成一个 deb 包

下面以该包为例,详细介绍如何将二进制软件包打包成符合麒麟软件打包规范的 deb 包,其中 deb 包的具体命名和内容规范可参考附录一《麒麟桌面 V10-DEB 软件包打包规范》

3.1 准备打包目录

3.1.1 创建构建 deb 包的目录

创建一个构建 deb 包的目录,目录以 包名-版本号 命名,如

```
~$ mkdir pycharm-professional-2020.3
```

3.1.2 构建规范的软件目录

构建一个规范的软件目录,用来按规则放置软件的各类文件,完整的目录结构为

```
~/pycharm-professional-2020.3$ tree
.
├── opt
│   ├── apps
│   └── pycharm #此目录以包名命名,放置软件自身的文件
└── usr
    ├── share
    │   ├── applications #desktop 文件存放目录
    │   ├── icons
    │   │   ├── hicolor
    │   │   │   ├── 128x128
    │   │   │   │   ├── apps #png 格式 128x128 尺寸图标放置位置
    │   │   │   │   ├── 16x16
    │   │   │   │   ├── apps #png 格式 16x16 尺寸图标放置位置
    │   │   │   │   ├── 24x24
    │   │   │   │   ├── apps #png 格式 24x24 尺寸图标放置位置
    │   │   │   │   └── 256x256
```

```
|   └── apps #png 格式 256x256 尺寸图标放置位置
|   └── 32x32
|   └── apps #png 格式 32x32 尺寸图标放置位置
|   └── 48x48
|   └── apps #png 格式 48x48 尺寸图标放置位置
|   └── 512x512
|   └── apps #png 格式 512x512 尺寸图标放置位置
|   └── 64x64
|   └── apps #png 格式 64x64 尺寸图标放置位置
|   └── 96x96
|   └── apps #png 格式 96x96 尺寸图标放置位置
|   └── scalable
|       └── apps #svg 格式图标放置位置

28 directories, 0 files
```

3.1.3 解压放置软件包

将二进制包 `pycharm-professional-2020.3.tar.gz` 解压，并将包里面的所有的文件放入 `~/pycharm-professional-2020.3/opt/apps/pycharm` 目录

```
$ cd ~/soft
~/soft$ tar -zxvf pycharm-professional-2020.3.tar.gz
~/soft$ cd pycharm-2020.3/
~/soft/pycharm-2020.3$ mv *
~/pycharm-professional-2020.3/opt/apps/pycharm/
```

3.1.4 放置图标

将图标文件放入相应的位置(由于包中已经提供 `svg` 格式的图标，可以不拷贝 `png` 格式的图标)

```
$ cd ~/pycharm-professional-2020.3/opt/apps/pycharm/bin
~/pycharm-professional-2020.3/opt/apps/pycharm/bin$ cp pycharm.svg
~/pycharm-professional-2020.3/usr/share/icons/hicolor/scalable/apps/
~/pycharm-professional-2020.3/opt/apps/pycharm/bin$ cp pycharm.png
~/pycharm-professional-2020.3/usr/share/icons/hicolor/128x128/apps/
```

3.2 创建.desktop 文件

3.2.1 新建.desktop 文件

.desktop 文件以 软件名.desktop 命名，例如 `pycharm.desktop`

```
$ cd ~/pycharm-professional-2020.3/usr/share/applications/
~/pycharm-professional-2020.3/opt/apps/pycharm/bin$ vim pycharm.desktop
```

```
[Desktop Entry]
Version=2020.3
Name=PyCharm Professional Edition
Comment=Python IDE for Professional Developers
Icon=pycharm
Exec="/opt/apps/pycharm/bin/pycharm.sh" %f
Categories=Development
Terminal=false
Type=Application
StartupWMClass=jetbrains-pycharm
StartupNotify=true
```

3.2.2 desktop 语法解释

[Desktop Entry]	#文件头
Name	#英文名称
Name[zh_CN]	#中文名称
Comment	#软件英文注解
Comment[zh_CN]	#软件中文注解
Exec	#软件运行文件绝对路径
Icon	#图标名
Terminal	#是否使用终端
Type	#启动器类型
Categories	#应用类型

#Categories 分类要求:

安卓	Android
网络	Network
社交	Messaging
影音	Audio、Video

开发	Development
图像	Graphics
游戏	Game
办公	Office 、 Calculator 、 Spreadsheet 、 Presentation 、 WordProcessor、 TextEditor
教育	Education
系统	System、 Settings、 Security

3.3 创建 DEBIAN 目录

进入 pycharm-professional-2020.3 目录,使用 dh_make --createorig

-s 命令创建 debian 目录,并在上层目录初始化出来源码包

```
$ cd ~/pycharm-professional-2020.3/
~/pycharm-professional-2020.3$ dh_make --createorig -s
Maintainer Name      : wuzhaohui
Email-Address        : wuzhaohui@kylinos.cn
Date                 : Fri, 16 Jul 2021 16:38:21 +0800
Package Name         : pycharm-professional
Version              : 2020.3
License              : blank
Package Type         : single
Are the details correct? [Y/n/q]
```

确认信息无误输入 y 即可, 当前目录下会自动创建 debian 目录,目录下有很多打包使用的模板文件,以 .ex/.EX 结尾,但对于大部分软件包的打包是不需要这些模板文件的,所以全部删掉(如果有需要请根据实际情况留存)

```
~/pycharm-professional-2020.3$ ls debian/
changelog  copyright      manpage.sgml.ex  postinst.ex  preinst.ex
pycharm-professional.cron.d.ex          pycharm-professional-docs.docs
README.source  salsa-ci.yml.ex  watch.ex
control      manpage.1.ex  manpage.xml.ex  postrm.ex    prerm.ex
pycharm-professional.doc-base.EX  README.Debian                                     rules
```

```
source
~/pycharm-professional-2020.3$ rm -rf debian/*.ex debian/*.EX
~/pycharm-professional-2020.3$ ls debian
changelog      control        copyright      debhelper-build-stamp  files
pycharm-professional
pycharm-professional.substvars  README.Debian  README.source  rules
source
```

3.4 修改 debian 目录下文件

3.4.1 修改 control 文件

control 是 deb 打包必须具备的描述性文件，此处主要修改了 Section 字段、Homepage 字段、Architecture 字段和 Description 字段，修改后的 control 文件内容如下：

```
~/pycharm-professional-2020.3$ vim debian/control
1 Source: pycharm-professional
2 Section: devel
3 Priority: optional
4 Maintainer: wuzhaohui <wuzhaohui@kylinos.cn>
5 Build-Depends: debhelper
6 Standards-Version: 4.4.1
7 Homepage: http://jetbrains.com
8 #Vcs-Browser: https://salsa.debian.org/debian/pycharm-professional
9 #Vcs-Git: https://salsa.debian.org/debian/pycharm-professional.git
10
11 Package: pycharm-professional
12 Architecture: all
13 Depends: ${shlibs:Depends}, ${misc:Depends}
14 Description: PyCharm 是一种 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。
15 <insert long description, indented with spaces>
```

3.4.2 control 文件几个字段说明

sections	为软件分类,字段值参考 sections 字段值参考
homepage	软件的主页网址
Architecture	支持的架构,因为该软件不区分架构,所以写为 all

descriptions	软件的描述信息
--------------	---------

control 文件必须要有 Package、Version、Architecture、Maintainer 和 Description 字段且内容不能为空，否则在打包或安装软件过程中会出现报错或警告信息；建议有 Depends、Section、Priority 字段。参考：

<https://www.debian.org/doc/debian-policy/ch-controlfields.html>

3.4.3 创建 install 文件

因为是从二进制包构建的，所以我们创建的目录都没有通过标准的 make install 进行安装，所以需要在 debian 目录创建 install 文件已被 dh_install 安装,install 文件可以指定各个文件的安装路径,可以参考 官方文档

本包的 install 文件参考：

```
~/pycharm-professional-2020.3$ cat debian/install
opt/apps/pycharm/ /opt/apps
usr/share /usr/
```

3.4.4 修改 rules 文件

另外，由于是此包从二进制包构建的,不用编译和生成共享库等行为,所以我们可以 在 rules 文件里覆盖掉某些指令,debian 官方文档

本包的 rules 文件参考如下：（除了标注出来的行，其他为默认生成）

```
~/pycharm-professional-2020.3$ cat debian/rules
#!/usr/bin/make -f
# See debhelper(7) (uncomment to enable)
# output every command that modifies files on the build system.
export DH_VERBOSE = 1

# see FEATURE AREAS in dpkg-buildflags(1)
#export DEB_BUILD_MAINT_OPTIONS = hardening=+all

# see ENVIRONMENT in dpkg-buildflags(1)
# package maintainers to append CFLAGS
#export DEB_CFLAGS_MAINT_APPEND = -Wall -pedantic
# package maintainers to append LDFLAGS
```

```
#export DEB_LDFLAGS_MAINT_APPEND = -Wl,--as-needed

%:
    dh $@

override_dh_auto_build:    #新增

override_dh_shlibdeps:     #新增

override_dh_strip:        #新增

# dh_make generated override targets
# This is example for Cmake (See https://bugs.debian.org/641051 )
#override_dh_auto_configure:
# dh_auto_configure                --                                #
    -DCMAKE_LIBRARY_PATH=$(DEB_HOST_MULTIARCH)
```

3.5 构建软件包

使用 debuild 构建软件包

```
~/pycharm-professional-2020.3$ debuild -i -us -uc -b
dpkg-buildpackage -us -uc -ui
dpkg-buildpackage: info: 源码包 pycharm-professional
dpkg-buildpackage: info: 源码版本 2020.3-1
dpkg-buildpackage: info: source distribution unstable
dpkg-buildpackage: info: 源码修改者 wuzhaohui <wuzhaohui@kylinos.cn>
dpkg-source --before-build .
dpkg-buildpackage: info: 主机架构 amd64
fakeroot debian/rules clean
dh clean
    dh_clean
.....
pkg-deb: 正在 './pycharm-professional_2020.3-1_all.deb' 中构建软件包
'pycharm-professional'。
dpkg-genbuildinfo
dpkg-genchanges >../pycharm-professional_2020.3-1_amd64.changes
dpkg-genchanges: info: 上传数据中包含完整的原始代码
dpkg-source --after-build .
dpkg-buildpackage: info: 完整上载 (包含原始的代码)
Now signing changes and any dsc files...
    signfile                dsc                pycharm-professional_2020.3-1.dsc
F4E75851A674BCE35C16F4D9ECA09489AE177E19

fixup_buildinfo                pycharm-professional_2020.3-1.dsc
```

```
pycharm-professional_2020.3-1_amd64.buildinfo
  signfile      buildinfo      pycharm-professional_2020.3-1_amd64.buildinfo
F4E75851A674BCE35C16F4D9ECA09489AE177E19

  fixup_changes      dsc      pycharm-professional_2020.3-1.dsc
pycharm-professional_2020.3-1_amd64.changes
  fixup_changes  buildinfo  pycharm-professional_2020.3-1_amd64.buildinfo
pycharm-professional_2020.3-1_amd64.changes
  signfile      changes      pycharm-professional_2020.3-1_amd64.changes
F4E75851A674BCE35C16F4D9ECA09489AE177E19

Successfully signed dsc, buildinfo, changes files
```

查看构建的软件包

```
~/pycharm-professional-2020.3$ cd ..
~$ ls
pycharm-professional_2020.3-1.dsc
pycharm-professional_2020.3-1_amd64.build
pycharm-professional_2020.3.orig.tar.xz
pycharm-professional_2020.3-1_amd64.buildinfo
pycharm-professional-2020.3
pycharm-professional_2020.3-1_amd64.changes
pycharm-professional_2020.3-1_all
pycharm-professional_2020.3-1_all.deb
pycharm-professional_2020.3-1.debian.tar.xz
```

3.6 验证 deb 包

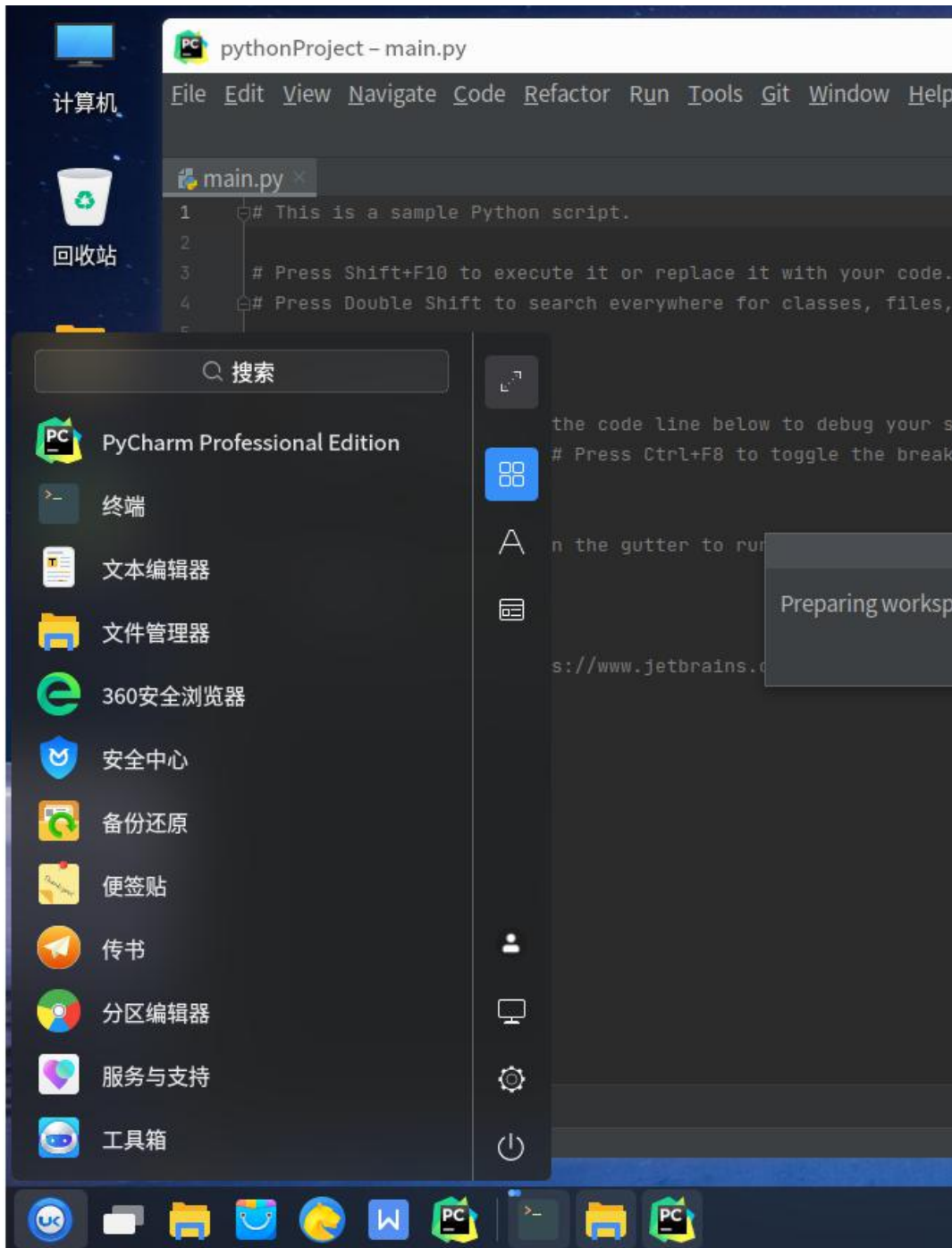
通过 `dpkg -i` 测试安装，看软件包是否可以在启动器显示图标，正常运行即可

```
~$ sudo dpkg -i pycharm-professional_2020.3-1_amd64.deb
QStandardPaths: wrong ownership on runtime directory /run/user/1000, 1000
instead of 0
PlatformTheme Create "ukui"
ProxyStyle create "kysec_auth" "ukui"
Qt5UKUIStyle create "kysec_auth" "ukui-default"
正在选中未选择的软件包 pycharm-professional。
(正在读取数据库 ... 系统当前共安装有 205061 个文件和目录。)
准备解压 pycharm-professional_2020.3-1_amd64.deb ...
正在解压 pycharm-professional (2020.3-1) ...
正在设置 pycharm-professional (2020.3-1) ...
正在处理用于 desktop-file-utils (0.24-1kylin2) 的触发器 ...
正在处理用于 bamfdaemon (0.5.3+18.04.20180207.2-0kylin2) 的触发器 ...
```

```
Rebuilding /usr/share/applications/bamf-2.index...  
正在处理用于 mime-support (3.64kylin1) 的触发器 ...  
正在处理用于 hicolor-icon-theme (0.17-2) 的触发器 ...
```

验证 deb 包是否正常

```
~$ dpkg -l pycharm-professional  
期望状态=未知(u)/安装(i)/删除(r)/清除(p)/保持(h)  
| 状态=未安装(n)/已安装(i)/仅存配置(c)/仅解压缩(U)/配置失败(F)/不完全安装(H)/触发器等待(W)/触发器未决(T)  
|/ 错误?=(无)/须重装(R) (状态, 错误: 大写=故障)  
||/ 名称                版本                体系结构          描述  
+++-----  
==-----  
ii  pycharm-professional 2020.3-1          amd64              <insert up to 60 chars  
description>
```



4 联系我们

如有需要帮助，可联系我们：

咨询热线

400-089-1870

附录一 《银河麒麟桌面操作系统 V10-DEB 包打包规范》

链接：麒麟生态网站—文档帮助—《银河麒麟桌面操作系统 V10-DEB 包打包规范》