KYLINSOFT
麒麟软件

银河麒麟高级服务器操作系统 RPM 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	康佳楠	2022-03-04	首次创建
V1.1	内容变更	冯培培	2022-09-15	变更联系人
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	2025-10-27	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 目的

为使软件开发者能够快速掌握打包方法，可以独立构建 RPM 包，特制定此开发者指南，使其了解打包过程，提高工作效率。

2 范围

本指南适用于软件开发者在银河麒麟高级服务器操作系统下的打包工作。

3 相关文件

《银河麒麟高级服务器操作系统 V10 开发指南-V0.1》

4 RPM 软件包介绍

RPM Package Manager (RPM) 是一个强大的命令行驱动的软件包管理工具，用来安装、卸载、校验、查询和更新 Linux 系统上的软件包，而银河麒麟高级服务器操作系统可以完全使用该格式的软件包。

RPM 软件包分为两类：二进制 RPM 包 xxx.rpm 和源码 RPM 包 xxx.src.rpm。

1) 二进制 RPM 包 xxx.rpm：包含从源码和补丁构建的二进制文件。

2) 源码 RPM 包 xxx.src.rpm：包含源代码、SPEC 文件及其它必需文件，该文件描述了如何将源代码构建成 xxx.rpm 格式。另外，还会包括源代码 patch。

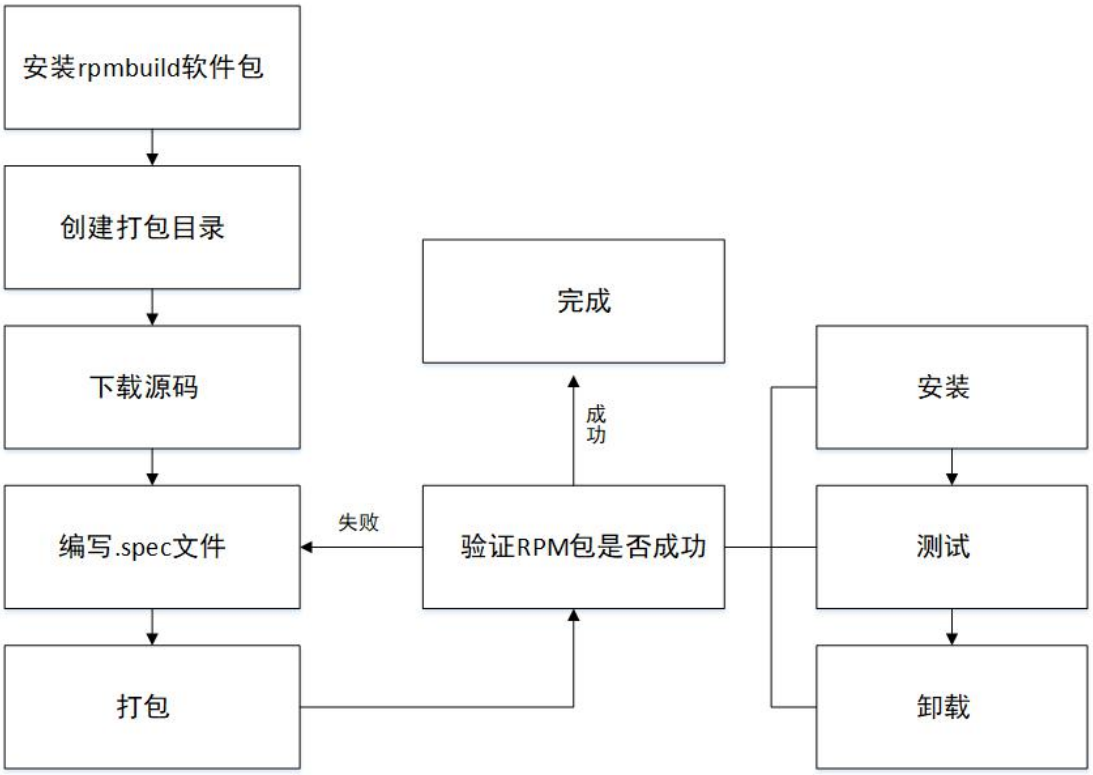
RPM 是为解决源码包不易安装和软件包相互之间依赖的问题，即源码包需要进行编译，而 RPM 包管理器也可以在一定程度解决依赖问题。它通过在探测源码包在 build 和 install 阶段的动作获得最终生成的需要安装的系统里的文

件，并记录下一些必要的操作，比如安装完成后执行某项操作，然后将其组成为一个整体，当在用户安装此包时把前面获得的所有问题和记录的所有操作原原本本的作用在实际系统上。

所以，源码需要打包安装，下面介绍的是 RPM 打包的过程。

5 打包流程

银河麒麟高级服务器操作系统 RPM 包打包流程如下图所示：



详细步骤如下：

1) 安装 rpmbuild 工具

2) 创建打包目录

创建 rpmbuild 目录及 BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS 六个子目录。

3) 下载源码

要把获取到的源代码放到 SOURCES 文件夹中。

4) 编写.spec 文件

将 Name、Summary、Version、Release、License、Sources 等相关内容写入.spec 文件。

5) 打包

执行 rpmbuild 命令打包。

6) 验证 RPM 包是否成功

通过安装、测试以及卸载打好的 RPM 包，验证 RPM 包是否成功。

6 RPM 打包

6.1 环境准备

首先，RPM 在打包之前需要安装 rpmbuild 工具，因为在打包过程中要用到 rpmbuild 命令，安装命令如下：

```
# yum install rpm-build
```

6.2 设置 RPM 打包工作目录

创建 rpmbuild 目录及 BUILD，BUILDROOT，RPMS，SOURCES，SPECS，SRPMS 六个子目录。其中，rpmbuild 目录必须放在~目录下。创建打包目录命令如下：

```
# mkdir -p ~/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
# tree ~/rpmbuild
  rpmbuild
      |-----BUILD
      |-----BUILDROOT
      |-----RPMS
      |-----SOURCES
      |-----SPECS
      |-----SRPMS
```

以下为各个目录的具体介绍：

默认位置	宏代码	名称	说明
~/rpmbuild/BUILD	%_builddir	构建目录	源码包被解压至此，并在该目录完成编译
~/rpmbuild/BUILDROOT	%buildrootdir	临时安装目录	编译后生成的软件临时安装目录，即保存 %install 阶段安装的文件
~/rpmbuild/RPMS	%_rpmdir	标准 RPM 包目录	保存打包生成的二进制 RPM 包（xxx.rpm）
~/rpmbuild/SOURCES	%_sourcedir	源代码目录	保存源码包（如 .tar 包、所有 patch 补丁以及所有用到的文件
~/rpmbuild/SPECS	%_specdir	.spec 文件目录	保存 RPM 包配置（.spec）文件
~/rpmbuild/SRPMS	%_srcrpmdir	源代码 RPM 包目录	保存打包生成的源码 RPM 包（xxx.src.rpm）

6.3 编写.spec 文件

构建一个标准的 RPM 包，需要编写.spec 文件，而.spec 文件也是打 RPM 包的核心。它定义了该如何构建、安装该软件包中的软件。也说明了该软件包的原始地址、版本号、代码存放目录、以及存在的补丁和依赖，还说明了整个执行流程，比如下载了软件包后，该如何处理源代码，如何编译构建软件，在构建完成后如何安装等。

操作如下：

```
# cd ~/rpmbuild/SPECS
# vim xxx.spec
```

xxx.spec 文件的基本格式如下：

```
Name：                  #软件包名称
Version:                #软件包版本号
Release:                #发布序列号
Summary:                #软件包的简介
Summary(zh_CN)：        #软件包的中文简介
License:                #软件许可证
URL:                    #项目网址
Sources0:               #源码包
%description            #描述
%prep                   #预处理
%build                  #编译
%install                #安装
%clean                  #清理临时文件
%check                  #检查软件
%pre                    #安装前执行的脚本
%post                   #安装后执行的脚本
%preun                  #卸载前执行的脚本
%postun                 #卸载后执行的脚本
%files                  #安装的文件列表
%changelog              #更改日志
```

xxx.spec 文件的具体内容说明：

SPEC 指令	定义
Name	软件包的名称，应该和.spec 文件的文件名一致。（.spec 文件中必须包括）
Version	软件包的版本号。若是开源软件，则版本号需要与上游代码的版本号一致。（.spec 文件中必须包括）
Release	发布序列号，标明第几次打包，根据自己发布该包的版本来写，每次进行新发布要增加 1。（.spec 文件中必须包括）
Summary	软件包的简介，要用简洁的语言对软件包进行描述，结尾不要有标点。如果是来自 github 的项目，Summary 可以直接用 github 项目中的 description。（.spec 文件中必须包括）
Summary(zh_CN)	软件包的中文简介，与 Summary 意义一样。（.spec 文件中可以不包括）
License	软件许可证。（.spec 文件中必需包括）
URL	项目网址。为了保持一致性,请使用%{name}的 RPM 宏变量，并使用 https://example.com/%{name}。(spec 文件中必须包括)
Source0	源码包的路径，最好填写可以获取到包中源码的 url。同样,发行版本也可以更改。为了简化这些潜在的将来更改,请使用%{name}和%{version}宏。通过使用这些,只需要更新 .spec 文件中的一个字段。如果有多个，可以添加更多 SourceX 指令，每次增加数量，例如：Source1、Source2、Source3 等。（.spec 文件中必须包括）
Patch	补丁。一个补丁只做一种修改，做多种修改需要拆分补丁，比如 Patch1、Patch2,...等。（.spec 文件中不必须包括，需要视具体情况而定）
BuildRequires	编译时需要的依赖。(.spec 文件中不必须包括，需要视具体情况而定）
Requires	安装时需要的依赖。(.spec 文件中不必须包括，需要视具体情况而定）

%description	软件包的详细说明。（.spec 文件中必须包括）
%prep	预处理脚本，可用于准备构建和编译的环境。一般来说这里进行的操作,都是将%_sourcedir 目录下的源代码解压到%_builddir 目录下、给解压好的代码打补丁、为接下来的操作做准备等。（.spec 文件中必须包括）
%build	源码编译阶段，将通过直接调用源码目录中自动构建工具在%_builddir 目录下完成源码编译操作， 比如编译 C 工程的 configure，make 等操作。（.spec 文件中必须包括）
%install	预安装阶段，将需要打包到 rpm 软件包里的文件从%_builddir 目录下拷贝到%buildroot 目录下。当用户用 rpm -ivh name-version.rpm 安装软件包时，这些文件会安装到用户系统中相应的目录里。（.spec 文件中必须包括）
%clean	清理临时文件，编译后的清理工作，这里可以执行 make clean 以及清空%_buildroot 目录等。（.spec 文件中必须包括）
%check	检查软件是否正常运行，通过执行 make test 的命令实现。（.spec 文件中可以不包括）
%files	列出需要打包的文件和目录。即本 RPM 包中的文件最终会被装到用户电脑上的文件的列表，以及这些文件的完整路径。（.spec 文件中必须包括）
%changelog	标签应包含每个 Release 所做的更改日志,如对软件打了新的补丁、构建过程变动等，尤其应包含上游的安全/漏洞补丁的说明。每个版本变动可以包含多个变动条目，每个条目以一个“-”开头。（.spec 文件中必须包括）
%pre	RPM 安装前执行的脚本。（.spec 文件中可以不包括）
%post	RPM 安装后执行的脚本。（.spec 文件中可以不包括）
%preun	RPM 卸载前执行的脚本。（.spec 文件中可以不包括）
%postun	RPM 卸载后执行的脚本。（.spec 文件中可以不包括）

其中，Name、Version 和 Release 指令组成 RPM 软件包的文件名。

具体命名为：softname-version-release.platform.rpm

命名规则可参考附录一《银河麒麟高级服务器操作系统 rpm 打包规范》。

示例如下：

```
hello-2.10-1.ky10.x86_64.rpm
```

6.4 构建 RPM 包

编写好.spec 文件之后，如果有依赖的话，可以自动安装打包过程中需要的依赖，然后再构建 RPM 包，具体命令如下：

```
# yum-builddep  ~/rpmbuild/SPECS/xxx.spec
```

构建 RPM 包要用到 rpmbuild 命令,可用以下方法来构建二进制 RPM 包（xxx.rpm）和源码 RPM 包（xxx.src.rpm）：

从 .spec 文件构建二进制 RPM 包（xxx.rpm）和源码 RPM 包（xxx.src.rpm），需要用到-ba 指令，具体命令如下：

```
# rpmbuild –ba ~/rpmbuild/SPECS/xxx.spec
```

至此，RPM 包构建完成，然后可进入相应的文件目录下查看生成的 RPM 包，具体命令如下：

```
# cd ~/rpmbuild/RPMSXXX/
```

其中，XXX 为架构名，xxx.rpm 包在 XXX 的目录下。

```
# cd ~/rpmbuild/SRPMS
```

如果只想构建二进制 RPM 包（xxx.rpm），不构建源码 RPM 包（xxx.src.rpm）需要用到-bb 指令，具体命令如下：

```
# rpmbuild –bb ~/rpmbuild/SPECS/xxx.spec
```

至此，xxx.rpm 包构建完成，然后可进入~/rpmbuild/RPMS 目录下查看生成的 xxx.rpm 包，具体命令如下：

```
# cd ~/rpmbuild/RPMSXXX/
```

其中，XXX 为架构名，xxx.rpm 包在 XXX 的目录下。

如果只想构建源码 RPM 包（xxx.src.rpm），不构建二进制 RPM 包

（xxx.rpm），需要用到-bs 指令，具体命令如下：

```
# rpmbuild –bs ~/rpmbuild/SPECS/xxx.spec
```

至此，xxx.src.rpm 包构建完成，然后可进入~/rpmbuild/SRPMS 目录下查看生成的 xxx.src.rpm 包，具体命令如下：

```
# cd ~/rpmbuild/SRPMS
```

7 验证 RPM 包

7.1 安装 RPM

构建完成 RPM 包之后，安装制作好的 xxx.rpm 包，具体命令如下：

```
# cd ~/rpmbuild/RPMS/XXX/
# rpm –ivh xxx.rpm
```

其中，XXX 为架构名，xxx.rpm 包在 XXX 的目录下。

如果有依赖存在，可以通过 yum 安装实现自动从源中解决依赖的问题，具体命令如下：

```
# yum install ~/rpmbuild/RPMS/XXX/xxx.rpm
```

然后，再检查 xxx.rpm 包是否安装成功，具体命令如下：

```
# rpm -qa 包名
```

7.2 卸载 RPM

具体卸载命令如下：

```
# cd ~/rpmbuild/RPMS/XXX/
# rpm –evh 包名
```

其中，XXX 为架构名，xxx.rpm 包在 XXX 的目录下。

检查卸载是否成功，具体命令如下：

```
# rpm -qa 包名
```

8 RPM 打包示例

8.1 具体打包示例

此示例以 hello world 为例构建 RPM 包。

1） 安装 rpmbuild 工具

```
# yum install rpm-build
```

2） 创建 rpmbuild 工具

```
# mkdir -p
~/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
```

```
[root@kylin ~]# tree rpmbuild
rpmbuild
├── BUILD
├── BUILDROOT
├── RPMS
├── SOURCES
├── SPECS
└── SRPMS

6 directories, 0 files
```

3） 准备源码：将源码下载并上传至 SOURCES 目录下

源码地址：http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz

```
[root@localhost ~]# tree ~/rpmbuild/SOURCES/
/root/rpmbuild/SOURCES/
└── hello-2.10.tar.gz

0 directories, 1 file
```

4） 编写.spec 文件

在 SPECS 目录下编写.spec 文件

```
# vim hello.spec
```

```
[root@localhost ~]# tree ~/rpmbuild/SPECS/
/root/rpmbuild/SPECS/
└── hello.spec

0 directories, 1 file
```

hello.spec 文件的具体内容如下：

```
Name:               hello
Version:            2.10
Release:            1
Summary:            The "Hello World" program from GNU
Summary(zh_CN):     GNU "Hello World"程序

License:            GPL
URL:                http://ftp.gnu.org/gnu/hello
Source0:            hello-2.10.tar.gz

%description
The "Hello World" program, done with all bells and whistles of a proper FOSS
project, including configuration, build, internationalization, help files, etc.


%prep
%setup -q

%build
%configure
%make_build


%install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=%{buildroot}
%define _unpackaged_files_terminate_build 0

%clean
rm -rf $RPM_BUILD_ROOT

%files
%{_mandir}/man1/hello.1.*
%{_infodir}/hello.info.*
%{_bindir}/hello
```

各个阶段说明：

%prep 阶段——预处理，主要对源代码包进行解压和打补丁

一般使用%setup -q 或者%setup -c 或%autosetup 命令来解压源码包，直接会将文件解压到%{builddir}。

```
[root@localhost ~]# cd ~/rpmbuild/BUILD
[root@localhost BUILD]# ls
hello-2.10
```

%build 阶段——对源代码包进行编译

编译阶段。即对解压到%_builddir 下的源码进行编译的阶段，整个过程在该目录下完成。

%install 阶段——源码预安装阶段

此阶段包含安装阶段需要执行的命令，首先在安装前清理一下%{_buildrootdir}目录中的文件，然后执行 make install 命令临时安装在 buildroot 目录下。

%clean——完成后的一些清理工作

主要是清理%{_buildrootdir}目录中产生的中间文件。

%file——列出需要打包的文件和目录

需要安装在系统中的文件在%files 中声明。注意不要使用形如/usr/bin 的绝对路径，应使用类似%{_bindir}/hello 这样的宏来替代。手册页应单独在%doc 中声明：%doc%{_mandir}/man1/hello.1.*。

5) 开始构建 RPM 包

```
# rpmbuild -ba ~/rpmbuild/SPECS/xxx.spec
```

打包成功后便可在 RPMS 和 SRPMS 目录下分别生成 xxx.rpm 包和 xxx.src.rpm 包。

```
[root@localhost ~]# tree ~/rpmbuild/RPMS/x86_64/
/root/rpmbuild/RPMS/x86_64/
├── hello-2.10-1.ky10.x86_64.rpm
├── hello-debuginfo-2.10-1.ky10.x86_64.rpm
└── hello-debugsource-2.10-1.ky10.x86_64.rpm

0 directories, 3 files

[root@localhost ~]# tree ~/rpmbuild/SRPMS/
/root/rpmbuild/SRPMS/
└── hello-2.10-1.ky10.src.rpm

0 directories, 1 file
```

6) 安装制作好的 xxx.rpm 包

```
# rpm -ivh ~/rpmbuild/RPMS/x86_64/hello-2.10.ky10.x86_64.rpm
```

```
[root@localhost ~]# rpm -ivh ~/rpmbuild/RPMS/x86_64/hello-2.10-1.ky10.x86_64.rpm

Verifying...                          ################################# [100%]
准备中...                             ################################# [100%]
正在升级/安装...
   1:hello-2.10-1.ky10                ################################# [100%]
```

7) 测试 xxx.rpm 包是否安装成功

```
# rpm -qa hello
```

```
[root@localhost ~]# rpm -qa hello
hello-2.10-1.ky10.x86 64
```

8) 卸载已安装的 xxx.rpm 包

```
# rpm -evh hello
```

```
[root@localhost ~]# rpm -evh hello
准备中...                          ################################# [100%]
正在清理/删除...
   1:hello-2.10-1.ky10             ################################# [100%]
```

9) 测试 xxx.rpm 安装包是否卸载完成

```
# rpm -qa hello
```

```
[root@localhost ~]# rpm -qa hello
[root@localhost ~]#
```

9 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 《银河麒麟高级服务器操作系统 RPM 包打包规范》

麒麟生态网站—技术文档—《银河麒麟高级服务器操作系统 RPM 包打包规范》