# RPM软件包

## RPM软件包简介

软件包引入即引入RPM软件包，RPM软件包是Linux下的一种软件的可执行程序，是类似于Windows上的exe文件。特点如下：

1. 安装的软件需要先编译成功，之后将软件打包成为按RPM机制生成的软件包。
2. RPM软件包中会记录并检查软件安装所需要的依赖包，若系统中已安装所需依赖，则成功安装，否则会安装失败。（若需要提前安装所需依赖，可通过rpm –ivh或者dnf install等方式将所依赖的RPM包安装至系统中。两种命令的使用方法在后续有展开说明。）
3. 安装RPM软件包时会就将该软件的信息全部写入 RPM 所属的数据库中，以便后续的查询、升级和卸载。

这种软件安装包通常是一个后缀为“.rpm”的文件，通过它能够更加轻松容易地实现Linux环境上软件的安装。

RPM软件包主要用的有二进制包（binary）和源代码包（source），二进制包可以直接安装在计算机中，而源代码包（SRPM）中包含未编译的源码以及SPEC文件、Patch补丁文件、配置文件等，可在具体的安装环境下编译成RPM包再进行安装。源代码包以“src.rpm”作为后缀名。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件格式 | 文件名 | 安装后能否直接使用 | 内含程序类型 | 能否修改内置参数并编译 |
| RPM | xxx.rpm | 是 | 已编译的二进制安装包 | 否 |
| SRPM | xxx.src.rpm | 否 | 未编译的源码以及spec文件、patch、配置文件等 | 是 |

### RPM的命名规则

RPM包名结构一般为name-version-release.architecture.rpm，依次为软件名、版本号、编译次数、硬件平台以及扩展名。例如：

pkgship-2.1.0-8.oe1.noarch.rpm 为二进制包，pkgship-2.1.0-8.oe1.src.rpm为源码包。

### 软件包的依赖

RPM软件包的依赖主要分为编译依赖和安装依赖，编译依赖是指源码RPM包编译生成二进制RPM包的时候所需要依赖的其他RPM二进制包；而安装依赖则是指在安装二进制RPM包的过程中，需要依赖到的其他的RPM二进制包。编译、安装依赖的指定，主要是在SPEC文件中呈现，SPEC文件的介绍和编写规范，会在后续的文档中呈现。

## RPM包的使用

### RPM软件管理器的使用

RPM软件包管理器主要是通过使用rpm命令来实现，该工具需要root权限：

1. 安装软件

执行rpm -ivh rpm包名

rpm -ivh apache-1.3.6.i386.rpm

-i: install

-v: 显示指令的执行过程

-h: 显示安装进度

1. 升级软件

执行rpm –Uvh rpm包名

rpm -Uvh apache-1.3.6.i386.rpm

-Uvh:无论软件有没有安装过旧版本，都会安装到更新版本。

-Fvh:若已经安装过旧版软件才会更新到新版本，否则不安装新版本软件

1. 卸载软件

执行rpm -e rpm包名

rpm –e apache-1.3.6

包名可以包含版本号等信息，但是不可以有后缀.rpm

1. 回退软件

执行rpm -Uvh xxx.rpm –oldpackage

rpm –Uvh apache-1.3.6 –oldpackage

指定旧版本的RPM包，作版本回退

1. 查询软件包的信息

查询本机上的软件包相关信息，不必要加上完整的版本名称，只需要加软件名即可。

命令格式：[rpm -qa 软件名]

查询某个具体的RPM文件，则需要加完整的文件名称。

命令格式：[rpm -qf 文件名]

查询已安装软件的信息：

-q ：仅查询，后面接的软件名称是否有安装。

-qa ：列出所有的，已经安装在本机 Linux 系统上面的所有软件名，用grep +软件名筛选所需软件是否安装。

-qi ：列出该软件的详绅信息 (information)，包括开发商、版本和说明。

-ql ：列出该软件所有的文件和目录所在完整文件名 (list)。

-qc ：列出该软件的所有配置文件 (找出在 /etc/ 底下的文件名而已)。

-qd ：列出该软件的所有说明文件 (找出不 man 有关的文件而已)。

-qR ：列出与该软件有关的相依软件所含的文件 (Required 的意思）。

-qf ：由后面接的文件名，找出该文件属于哪一个已安装的软件。

查询某个 RPM 文件内含有的信息：

-qp[icdlR]：注意 -qp 后面接的所有参数以上面的说明一致。但用途仅在于找出某个 RPM 档案内的信息，而非已安装的软件信息。

### DNF软件管理器的使用

DNF工具同样也是运行在Linux系统上的RPM包管理工具，优于RPM软件包管理器的是，yum基于 RPM包管理上，能够从指定的服务器自动下载 RPM包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载、安装。DNF通过软件仓库（repository）进行软件的下载、安装等，软件仓库可以是一个 HTTP 或 FTP 站点，也可以是一个本软件池，资源仓库也可以是多个，在 /etc/yum.conf 文件中进行相关配置即可。在DNF的资源库中，会包括 RPM的头信息（header），头信息中包括了软件的功能描述、依赖关系等。通过分析这些信息，DNF计算出依赖关系并进行相关的升级、安装、删除等操作。

openEuler提供了多种repo源供用户在线使用，使用root权限添加openEuler repo源，用户可直接基于这些repo源直接使用dnf下载所需要的RPM软件包。

1. 安装RPM包

dnf install [-y] [rpm包名]

1. 卸载同时把依赖包卸载

dnf remove [-y][rpm包名]

1. 更新

dnf update [-y][包名]

不加包名整个系统都会更新

1. 查询

dnf provides "/\*/vim"

使用这种通配的格式查找到底vim是使用哪个包

如果需要额外挂载repo源，可以按照如下方法添加repo源：

1. 打开repo源文件。

vim /etc/yum.repos.d/openEuler.repo

1. 添加所需要的repo镜像源。

在文件中添加以下内容：

[repository]

name=repository\_name

baseurl=repository\_url

|  |  |
| --- | --- |
| 参数 | 说明 |
| name=repository\_name | 软件仓库（repository ）描述的字符串。 |
| baseurl=repository\_url | 软件仓库（repository ）的地址。   * 使用http协议的网络位置：例如 <http://path/to/repo> * 使用ftp协议的网络位置：例如 <ftp://path/to/repo> * 本地位置：例如 file:///path/to/local/repo |

DNF详细使用方法请参考[openEuler文档](https://openeuler.org/zh/docs/20.03_LTS/docs/Administration/%E4%BD%BF%E7%94%A8DNF%E7%AE%A1%E7%90%86%E8%BD%AF%E4%BB%B6%E5%8C%85.html)。

## rpmbuild的使用

当获取到上游社区的源码tar包和编写好的SPEC文件后，需要使用rpmbuild命令打包RPM软件。

1. 安装该软件包。

dnf install rpm-build

1. 建立工作目录。

rpmdev-setuptree

执行之后，会在用户家目录下生成一下几个目录。

/root/rpmbuild/目录结构如下：

BUILD #源码包解压至此并完成编译

RPMS #生成/保存二进制 RPM 包

SOURCES #保存源码包(如 .tar 包)和所有 patch 补丁

SPECS #保存rpm包配置文件(.spec)

SRPMS #生成/保存源码 RPM 包(SRPM)

BUILDROOT #保存 %install 阶段安装的文件

将源码包（和补丁文件，如果有的话）放在SOURCES目录下，spec文件放在SPECS目录下后即可开始尝试rpm打包。

1. 执行编译生成RPM包。

rpmbuild -ba \*\*.spec

如果编译时报错说缺少依赖，需要提前安装编译依赖

dnf builddep \*\*.spec

#若dnf安装显示没有此包，说明该包暂未引入进openEuler发行版本中，请到<https://gitee.com/organizations/src-openeuler/projects> 中确认是否有引入该包，若没有，则需要先引入该包。

1. 验证RPM包。

编译完成后到RPMS目录下去拿到编译生成的RPM包进行验证。若是编译报错，可以根据报错日志或者分步编译去进行调试。在调试过程中可以不用执行全部的编译步骤，可以针对日志中报错的部分（比如%prep或者%check），使用rpmbuild命令只去执行到报错阶段，各阶段意义可以在spec文件介绍.doc中查看。rpmbuild -ba 可以分别拆解成如下几个阶段去分步编译：

bp %prep 阶段

bc %build 阶段

bi %install阶段

bb %file阶段

一般先rpmbuild -bp ,再-bc 再-bi 如果没问题，rpmbuild -bb制作rpm包。