第12 讲软件包安装与管理

1 软件包分类

Linux下的软件包众多，且几乎都是经 GPL 授权、免费开源（无偿公开源代码）的。这意味着如果你具备修改软件源代码的能力，只要你愿意，可以随意修改。

GPL，全称 General Public License，中文名称“通用性公开许可证”，简单理解 GPL 就是一个保护软件自由的一个协议，经 GPL 协议授权的软件必须开源，请猛击《[开源协议](http://c.biancheng.net/view/2947.html" \t "http://c.biancheng.net/view/_blank)》了解更多信息。

Linux下的软件包可细分为两种，分别是源码包和二进制包。

1 Linux源码包

实际上，源码包就是一大堆源代码程序，是由程序员按照特定的格式和语法编写出来的。

我们都知道，计算机只能识别机器语言，也就是二进制语言，所以源码包的安装需要一名“翻译官”将“abcd”翻译成二进制语言，这名“翻译官”通常被称为编译器。

“编译”指的是从源代码到直接被计算机（或虚拟机）执行的目标代码的翻译过程，编译器的功能就是把源代码翻译为二进制代码，让计算机识别并运行，读者可以猛击《[C语言编译器](http://c.biancheng.net/compiler/" \t "http://c.biancheng.net/view/_blank)》了解更多信息。

虽然源码包免费开源，但用户不会编程怎么办？一大堆源代码程序不会使用怎么办？源码包容易安装吗？等等这些都是使用源码包安装方式无法解答的问题。

另外，由于源码包的安装需要把源代码编译为二进制代码，因此安装时间较长。比如，大家应该都在 Windows下安装过 QQ，QQ 功能较多，程序相对较大（有 70 MB左右），但由于其并非是以源码包的形式发布，而是编译后才发布的，因此只需几分钟（经过简单的配置）即可安装成功。但如果我们以源码包安装的方式在 Linux 中安装一个 [MySQL](http://c.biancheng.net/mysql/" \t "http://c.biancheng.net/view/_blank) 数据库，即便此软件的压缩包仅有 23 MB左右，也需要 30 分钟左右的时间（根据硬件配置不同，略有差异）。

通过对比你会发现，源码包的编译是很费时间的，况且绝多大数用户并不熟悉程序语言，在安装过程中我们只能**祈祷程序**不要报错，否则初学者很难解决。

为了解决使用源码包安装方式的这些问题，Linux 软件包的安装出现了使用二进制包的安装方式。

2 Linux二进制包

二进制包，也就是源码包经过成功编译之后产生的包。由于二进制包在发布之前就已经完成了编译的工作，因此用户安装软件的速度较快（同 Windows下安装软件速度相当），且安装过程报错几率大大减小。  
  
二进制包是 Linux 下默认的软件安装包，因此二进制包又被称为默认安装软件包。目前主要有以下 2 大主流的二进制包管理系统：

* RPM 包管理系统：功能强大，安装、升级、査询和卸载非常简单方便，因此很多 Linux 发行版都默认使用此机制作为软件安装的管理方式，例如 Fedora、CentOS、SuSE 等。
* DPKG 包管理系统：由 Debian Linux 所开发的包管理机制，通过 DPKG 包，Debian Linux 就可以进行软件包管理，主要应用在 Debian 和 Ubuntu 中。

RPM 包管理系统和 DPKG 管理系统的原理和形式大同小异，可以触类旁通。由于本教程使用的是 CentOS 6.x 版本，因此本节主要讲解 RPM 二进制包。

3 源码包 VS RPM二进制包

源码包一般包含多个文件，为了方便发布，通常会将源码包做打包压缩处理，Linux 中最常用的打包压缩格式为“tar.gz”，因此源码包又被称为 Tarball。

Tarball 是 Linux 系统的一款打包工具，可以对源码包进行打包压缩处理，人们习惯上将最终得到的打包压缩文件称为 Tarball 文件。

源码包需要我们自己去软件官方网站进行下载，包中通常包含以下内容：

* 源代码文件。
* 配置和检测程序（如 configure 或 config 等）。
* 软件安装说明和软件说明（如 INSTALL 或 README）。

总的来说，使用源码包安装软件具有以下几点好处：

* 开源。如果你有足够的能力，则可以修改源代码。
* 可以自由选择所需的功能。
* 因为软件是编译安装的，所以更加适合自己的系统，更加稳定，效率也更高。
* 卸载方便。

但同时，使用源码包安装软件也有几点不足：

* 安装过程步骤较多，尤其是在安装较大的软件集合时（如 LAMP 环境搭建），容易出现拼写错误。
* 编译时间较长，所以安装时间比二进制安装要长。
* 因为软件是编译安装的，所以在安装过程中一旦报错，新手很难解决。

相比源码包，二进制包是在软件发布时已经进行过编译的软件包，所以安装速度比源码包快得多（和 Windows 下软件安装速度相当）。也正是因为已经进行通译，大家无法看到软件的源代码。  
  
使用 RMP 包安装软件具有以下 2 点好处：

1. 包管理系统简单，只通过几个命令就可以实现包的安装、升级、査询和卸载。
2. 安装速度比源码包安装快得多。

与此同时，使用 RMP 包安装软件有如下不足：

* 经过编译，不能在看到源代码。
* 功能选择不如源码包灵活。
* 依赖性。有时我们会发现，在安装软件包 a 时需要先安装 b 和 c，而在安装 b 时需要先安装 d 和 e。这就需要先安装 d 和 e，再安装 b 和 c，最后才能安装 a。比如，我买了一个漂亮的灯具，打算安装在客厅里，可是在安装灯具之前，客厅需要有顶棚，并且顶棚需要刷好油漆。安装软件和装修及其类似，需要有一定的顺序，但是有时依赖性会非常强。

4 如何选择

通过源码包和 RMP 二进制包的对比，在 Linux 进行软件安装时，我们应该使用哪种软件包呢？

为了更好的区别两种软件包，这里举个例子。假设我们想做一套家具，源码包就像所有的家具完全由自己动手手工打造（手工编译），想要什么样的板材、油漆、颜色和样式都由自己决定（功能自定义，甚至可以修改源代码）。想想就觉得爽，完全不用被黑心的厂商所左右，而且不用担心质量问题（软件更适合自己的系统，效率更高，更加稳定）。但是，所花费的时间大大超过了买一套家具的时间（编译浪费时间），而且自己真的有做木工这个能力吗（需要对源代码非常了解）？就算请别人定制好的家具，再由自己组装，万一哪个部件不匹配（报错很难解决），怎么办？

那么二进制包呢？也是我们需要一套家具，去商场买了一套（安装简单），家具都是现成的，不会有哪个部件不匹配，除非因为自身问题没有量好尺寸而导致放不下（报错很少）。但是我们完全不知道这套家具用的是什么材料、油漆是否合格，而且家具的样式不能随意选择（软件基本不能自定义功能）。

通过以上对源码包和二进制包的讲解，相信大家对两者的区别和特点有了更清楚的认识。

## 2 rpm软件管理

1 RPM包统一命名规则

RPM 二进制包的**命名需遵守统一的命名规则**，用户通过名称就可以直接获取这类包的版本、适用平台等信息。  
  
**RPM 二进制包命名的一般格式如下**：

**包名-版本号-发布次数-发行商-Linux平台-适合的硬件平台-包扩展名**

例如，RPM 包的名称是

httpd-2.2.15-15.el6.centos.1.i686.rpm

Httpd-devel-2.5

其中：

httpd：**软件包名**

这里需要注意，httpd 是包名，而 httpd-2.2.15-15.el6.centos.1.i686.rpm 通常称为包全名，包名和包全名是不同的，在某些 Linux 命令中，有些命令（如包的安装和升级）使用的是包全名，而有些命令（包的查询和卸载）使用的是包名，一不小心就会弄错。

2.2.15：包的版本号，版本号的格式通常为主版本号.次版本号.修正号。

15：二进制包发布的次数，表示此 RPM 包是第几次编程生成的。

el\*：软件发行商，el6 表示此包是由 Red Hat 公司发布，适合在 RHEL 6.x (Red Hat Enterprise Unux) 和 CentOS 6.x 上使用。

centos：表示此包适用于 CentOS 系统。

i686：表示此包使用的硬件平台，目前的 RPM 包支持的平台如表 1 所示：

|  |  |
| --- | --- |
| 表 1 RPM 包适用的硬件平台 | |
| 平台名称 | 适用平台信息 |
| i386 | 386 以上的计算机都可以安装 |
| i586 | 686 以上的计算机都可以安装 |
| i686 | 奔腾 II 以上的计算机都可以安装，目前所有的 CPU 是奔腾 II 以上的，所以这个软件版本居多 |
| x86\_64 | 64 位 CPU 可以安装 |
| noarch | 没有硬件限制 |

rpm：RPM 包的扩展名，表明这是编译好的二进制包，可以使用 rpm 命令直接安装。此外，还有以 src.rpm 作为扩展名的 RPM 包，这表明是源代码包，需要安装生成源码，然后对其编译并生成 rpm 格式的包，最后才能使用 rpm 命令进行安装。

有人可能会问，Linux 系统不靠扩展名分区文件类型，那为什么包全名中要包含 .rpm 扩展名呢？其实，这里的扩展名是为系统管理员准备的，如果我们不对 RPM 包标注扩展名，管理员很难知道这是一个 RPM 包，当然也就无法正确使用。

2 RPM包安装、卸载和升级（rpm命令）详解

RPM：红帽子包管理器（RedHat Package Manager)

下面讲解如何使用 rpm 命令对 RPM 二进制包进行安装、卸载和升级操作。  
我们以安装 apache 程序为例。因为后续章节还会介绍使用源码包的方式安装 apache 程序，读者可以直观地感受到源码包和 RPM 包的区别。

3 RPM包默认安装路径

通常情况下，RPM 包采用系统默认的安装路径，所有安装文件会按照类别分散安装到表 1 所示的目录中。

|  |  |
| --- | --- |
| 表 1 RPM 包默认安装路径 | |
| 安装路径 | 含 义 |
| /etc/ | 配置文件安装目录 |
| /usr/bin/ | 可执行的命令安装目录 |
| /usr/lib/ | 程序所使用的函数库保存位置 |
| /usr/share/doc/ | 基本的软件使用手册保存位置 |
| /usr/share/man/ | 帮助文件保存位置 |

RPM 包的默认安装路径是可以通过命令查询的。  
  
除此之外，RPM 包也支持手动指定安装路径，但此方式并不推荐。因为一旦手动指定安装路径，所有的安装文件会集中安装到指定位置，且系统中用来查询安装路径的命令也无法使用（需要进行手工配置才能被系统识别），得不偿失。  
  
与 RPM 包不同，源码包的安装通常采用手动指定安装路径（习惯安装到 /usr/local/ 中）的方式。既然安装路径不同，同一 apache 程序的源码包和 RPM 包就可以安装到一台 Linux 服务器上（但同一时间只能开启一个，因为它们需要占用同一个 80 端口）。  
  
实际情况中，一台服务器几乎不会同时包含两个 apache 程序，管理员不好管理，还会占用过多的服务器磁盘空间。

4 RPM 包的安装

安装 RPM 的命令格式为：

[root@localhost ~]# rpm -ivh /path/包全名

注意一定是包全名。涉及到包全名的命令，一定要注意路径，可能软件包在光盘中，因此需提前做好设备的挂载工作。  
  
此命令中各选项参数的含义为：

-i：安装（install）;

-v：显示更详细的信息（verbose）;

-h：打印 #，显示安装进度（hash）;

例如，使用此命令安装 apache 软件包，如下所示：

[root@localhost ~]# rpm -ivh /mnt/cdrom/Packages/httpd-2.2.15-15.el6.centos.1.i686.rpm  
Preparing...  
####################  
[100%]  
1:httpd  
####################  
[100%]

注意，直到出现两个 100% 才是真正的安装成功，第一个 100% 仅表示完成了安装准备工作。  
  
此命令还可以一次性安装多个软件包，仅需将包全名用空格分开即可，如下所示：

[root@localhost ~]# rpm -ivh a.rpm b.rpm c.rpm

如果还有其他安装要求（比如强制安装某软件而不管它是否有依赖性），可以通过以下选项进行调整：

--nodeps：不检测依赖性安装。软件安装时会检测依赖性，确定所需的底层软件是否安装，如果没有安装则会报错。如果不管依赖性，想强制安装，则可以使用这个选项。注意，这样不检测依赖性安装的软件基本上是不能使用的，所以不建议这样做。

--replacefiles：替换文件安装。如果要安装软件包，但是包中的部分文件已经存在，那么在正常安装时会报"某个文件已经存在"的错误，从而导致软件无法安装。使用这个选项可以忽略这个报错而覆盖安装。

--replacepkgs：替换软件包安装。如果软件包已经安装，那么此选项可以把软件包重复安装一遍。

--force：强制安装。不管是否已经安装，都重新安装。也就是 -replacefiles 和 -replacepkgs 的综合。

--test：测试安装。不会实际安装，只是检测一下依赖性。

--prefix：指定安装路径。为安装软件指定安装路径，而不使用默认安装路径。

apache 服务安装完成后，可以尝试启动：

[root@localhost ~]# service 服务名 start|stop|restart|status

各参数含义：

start：启动服务；

stop：停止服务；

restart：重启服务；

status: 查看服务状态；

例如：

[root@localhost ~]# service httpd start #启动apache服务

服务启动后，可以查看端口号 80 是否出现。命令如下：

[root@localhost ~]# netstat -tlun | grep 80  
tcp 0 0 :::80:::\* LISTEN

也可以在浏览器中输入 Linux 服务器的 IP 地址，访问这个 apache 服务器。目前在 apache 中没有建立任何网页，所以看到的只是测试页，如图 2 所示。

  
图 2 apache测试页

5 RPM包的升级

使用如下命令即可实现 RPM 包的升级：

[root@localhost ~]# rpm -Uvh 包全名

-U（大写）选项的含义是：如果该软件没安装过则直接安装；若没安装则升级至最新版本。

[root@localhost ~]# rpm -Fvh 包全名

-F（大写）选项的含义是：如果该软件没有安装，则不会安装，必须安装有较低版本才能升级。

6 RPM包的卸载

RPM 软件包的卸载要考虑包之间的依赖性。例如，我们先安装的 httpd 软件包，后安装 httpd 的功能模块 mod\_ssl 包，那么在卸载时，就必须先卸载 mod\_ssl，然后卸载 httpd，否则会报错。

软件包卸载和拆除大楼是一样的，本来先盖的 2 楼，后盖的 3 楼，那么拆楼时一定要先拆除 3 楼。

如果卸载 RPM 软件不考虑依赖性，执行卸载命令会包依赖性错误，例如：

[root@localhost ~]# rpm -e httpd  
error: Failed dependencies:  
httpd-mmn = 20051115 is needed by (installed) mod\_wsgi-3.2-1.el6.i686  
httpd-mmn = 20051115 is needed by (installed) php-5.3.3-3.el6\_2.8.i686  
httpd-mmn = 20051115 is needed by (installed) mod\_ssl-1:2.2.15-15.el6.  
centos.1.i686  
httpd-mmn = 20051115 is needed by (installed) mod\_perl-2.0.4-10.el6.i686  
httpd = 2.2.15-15.el6.centos.1 is needed by (installed) httpd-manual-2.2.  
15-15.el6.centos.1 .noarch  
httpd is needed by (installed) webalizer-2.21\_02-3.3.el6.i686  
httpd is needed by (installed) mod\_ssl-1:2.2.15-15.el6.centos.1.i686  
httpd=0:2.2.15-15.el6.centos.1 is needed by(installed)mod\_ssl-1:2.2.15-15.el6.centos.1.i686

RPM 软件包的卸载很简单，使用如下命令即可：

[root@localhost ~]# rpm -e 包名

-e 选项表示卸载，也就是 erase 的首字母。  
  
RPM 软件包的卸载命令支持使用“-nocteps”选项，即可以不检测依赖性直接卸载，但此方式不推荐大家使用，因为此操作很可能导致其他软件也无法征程使用。

3 rpm命令查询软件包（-q、-qa、-i、-p、-l、-f、-R）

rpm 命令还可用来对 RPM 软件包做查询操作，具体包括：

查询软件包是否已安装；

查询系统中所有已安装的软件包；

查看软件包的详细信息；

查询软件包的文件列表；

查询某系统文件具体属于哪个 RPM 包。

使用 rpm 做查询命令的格式如下：

[root@localhost ~]# rpm 选项 查询对象

下面教大家使用 rpm 命令一一实现以上查询操作。

1 rpm -q：查询软件包是否安装

用 rpm 查询软件包是否安装的命令格式为：

[root@localhost ~]# rpm -q 包名

-q 表示查询，是 query 的首字母。  
  
例如，查看 Linux 系统中是否安装 apache，rpm 查询命令应写成：

[root@localhost ~]# rpm -q httpd  
httpd-2.2.15-15.el6.centos.1.i686

注意这里使用的是包名，而不是包全名。因为已安装的软件包只需给出包名，系统就可以成功识别（使用包全名反而无法识别）。

2 rpm -qa：查询系统中所有安装的软件包

使用 rpm 查询 Linux 系统中所有已安装软件包的命令为：

[root@localhost ~]# rpm -qa  
libsamplerate-0.1.7-2.1.el6.i686  
startup-notification-0.10-2.1.el6.i686  
gnome-themes-2.28.1-6.el6.noarch  
fontpackages-filesystem-1.41-1.1.el6.noarch  
gdm-libs-2.30.4-33.el6\_2.i686  
gstreamer-0.10.29-1.el6.i686  
redhat-lsb-graphics-4.0-3.el6.centos.i686  
…省略部分输出…

此外，这里还可以使用管道符查找出需要的内容，比如：

[root@localhost ~]# rpm -qa | grep httpd  
httpd-devel-2.2.15-15.el6.centos.1.i686  
httpd-tools-2.2.15-15.el6.centos.1.i686  
httpd-manual-2.2.15-15.el6.centos.1.noarch  
httpd-2.2.15-15.el6.centos.1.i686

相比rpm -q 包名命令，采用这种方式可以找到含有包名的所有软件包。

3 rpm -qi：查询软件包的详细信息

通过 rpm 命令可以查询软件包的详细信息，命令格式如下：

[root@localhost ~]# rpm -qi 包名

-i 选项表示查询软件信息，是 information 的首字母。  
  
例如，想查看 apache 包的详细信息，可以使用如下命令：

[root@localhost ~]# rpm -qi httpd  
Name : httpd Relocations:(not relocatable)  
#包名  
Version : 2.2.15 Vendor:CentOS  
#版本和厂商  
Release : 15.el6.centos.1 Build Date: 2012年02月14日星期二 06时27分1秒  
#发行版本和建立时间  
Install Date: 2013年01月07日星期一19时22分43秒  
Build Host:  
c6b18n2.bsys.dev.centos.org  
#安装时间  
Group : System Environment/Daemons Source RPM:  
httpd-2.2.15-15.el6.centos.1.src.rpm  
#组和源RPM包文件名  
Size : 2896132 License: ASL 2.0  
#软件包大小和许可协议  
Signature :RSA/SHA1,2012年02月14日星期二 19时11分00秒，Key ID  
0946fca2c105b9de  
#数字签名  
Packager：CentOS BuildSystem <http://bugs.centos.org>  
URL : http://httpd.apache.org/  
#厂商网址  
Summary : Apache HTTP Server  
#软件包说明  
Description:  
The Apache HTTP Server is a powerful, efficient, and extensible web server.  
#描述

除此之外，还可以查询未安装软件包的详细信息，命令格式为：

[root@localhost ~]# rpm -qip 包全名

-p 选项表示查询未安装的软件包，是 package 的首字母。  
  
注意，这里用的是包全名，且未安装的软件包需使用“绝对路径+包全名”的方式才能确定包。

4 rpm -ql：命令查询软件包的文件列表

通过前面的学习我们知道，rpm 软件包通常采用默认路径安装，各安装文件会分门别类安放在适当的目录文件下。使用 rpm 命令可以查询到已安装软件包中包含的所有文件及各自安装路径，命令格式为：

[root@localhost ~]# rpm -ql 包名

-l 选项表示列出软件包所有文件的安装目录。  
  
例如，查看 apache 软件包中所有文件以及各自的安装位置，可使用如下命令：

[root@localhost ~]# rpm -ql httpd  
/etc/httpd  
/etc/httpd/conf  
/etc/httpd/conf.d  
/etc/httpd/conf.d/README  
/etc/httpd/conf.d/welcome.conf  
/etc/httpd/conf/httpd.conf  
/etc/httpd/conf/magic  
…省略部分输出…

同时，rpm 命令还可以查询未安装软件包中包含的所有文件以及打算安装的路径，命令格式如下：

[root@localhost ~]# rpm -qlp 包全名

-p 选项表示查询未安装的软件包信息，是 package 的首字母。

注意，由于软件包还未安装，因此需要使用“绝对路径+包全名”的方式才能确定包。

比如，我们想查看 bing 软件包（未安装，绝对路径为：/mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm）中的所有文件及各自打算安装的位置，可以执行如下命令：

[root@localhost ~]# rpm -qlp /mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm  
/etc/NetworkManager/dispatcher.d/13-named  
/etc/logrotate.d/named  
/etc/named  
/etc/named.conf  
/etc/named.iscdlv.key  
/etc/named.rfc1912.zones  
…省略部分输出…

5 rpm -qf：命令查询系统文件属于哪个RPM包

rpm -ql 命令是通过软件包查询所含文件的安装路径，rpm 还支持反向查询，即查询某系统文件所属哪个 RPM 软件包。其命令格式如下：

[root@localhost ~]# rpm -q  **-f 系统文件名**

-f 选项的含义是查询系统文件所属哪个软件包，是 file 的首字母。  
  
注意，只有使用 RPM 包安装的文件才能使用该命令，手动方式建立的文件无法使用此命令。  
  
例如，查询 ls 命令所属的软件包，可以执行如下命令：

[root@localhost ~]# rpm -qf /bin/ls  
coreutils-8.4-19.el6.i686

6 rpm -qR：查询软件包的依赖关系

使用 rpm 命令安装 RPM 包，需考虑与其他 RPM 包的依赖关系。rpm -qR 命令就用来查询某已安装软件包依赖的其他包，该命令的格式为：

[root@localhost ~]# rpm -qR 包名

-R（大写）选项的含义是查询软件包的依赖性，是 requires 的首字母。  
  
例如，查询 apache 软件包的依赖性，可执行以下命令：

[root@localhost ~]# rpm -qR httpd  
/bin/bash  
/bin/sh  
/etc/mime.types  
/usr/sbin/useradd  
apr-util-ldap  
chkconfig  
config(httpd) = 2.2.15-15.el6.centos.1  
httpd-tods = 2.2.15-15.el6.centos.1  
initscripts >= 8.36  
…省略部分输出…

同样，在此命令的基础上增加 -p 选项，即可实现查找未安装软件包的依赖性。  
  
例如，bind 软件包尚未安装（绝对路径为： /mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm），查看此软件包的依赖性可执行如下命令：

[root@localhost ~]# rpm -qRp /mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm  
/bin/bash  
/bin/sh  
bind-libs = 32:9.8.2-0.10.rc1.el6  
chkconfig  
chkconfig  
config(bind) = 32:9.8.2-0.10.rc1.el6  
grep  
libbind9.so.80  
libc.so.6  
libc.so.6(GLIBC\_2.0)  
libc.so.6(GLIBC\_2.1)  
…省略部分输出…

注意，这里使用的也是“绝对路径+包全名”的方式。

7 RPM包验证和数字证书（数字签名）

执行 rpm -qa 命令可以看到，Linux 系统中装有大量的 RPM 包，且每个包都含有大量的安装文件。因此，为了能够及时发现文件误删、误修改文件数据、恶意篡改文件内容等问题，Linux 提供了以下两种监控（检测）方式：

* RPM 包校验：其实就是将已安装文件和 /var/lib/rpm/ 目录下的数据库内容进行比较，确定文件内容是否被修改。
* RPM 包数字证书校验：用来校验 RPM 包本身是否被修改。

#### 1 Linux RPM 包校验

RPM 包校验可用来判断已安装的软件包（或文件）是否被修改，此方式可使用的命令格式分为以下 3 种。

[root@localhost ~]# rpm -Va

-Va 选项表示校验系统中已安装的所有软件包。

[root@localhost ~]# rpm -V 已安装的包名

-V 选项表示校验指定 RPM 包中的文件，是 verity 的首字母。

[root@localhost ~]# rpm -Vf 系统文件名

-Vf 选项表示校验某个系统文件是否被修改。  
  
例如我们校验 apache 软件包中所有的安装文件是否被修改，可执行如下命令：

[root@localhost -]# rpm -V httpd

可以看到，执行后无任何提示信息，表明所有用 apache 软件包安装的文件均未改动过，还和从原软件包安装的文件一样。  
  
接下来尝试对 apache 的配置文件 /etc/httpd/conf/httpd.conf 做适当修改，修改格式如下：

[root@localhost ~]#vim /etc/httpd/conf/httpd.conf  
...省略部分内容...  
Directorylndex index.html index.html.var index.php  
#这句话是定义apache可以识别的默认网页文件名。在后面加入了index.php  
#这句话大概有400行左右  
…省略部分内容...

由于我们还未学习如何配置 apache，为防止其崩溃，这里仅尝试修改 apache 的默认网页文件。按照以上格式对文件进行修改后保存退出，再次使用 rpm -V 命令对 apache 软件包进行验证：

[root@localhost ~]# rpm -V httpd  
S.5....T. c /etc/httpd/conf/httpd.conf

可以看到，结果显示了文件被修改的信息。该信息可分为以下 3 部分：

1. 最前面的 8 个字符（S.5....T）都属于验证信息，各字符的具体含义如下：

S：文件大小是否改变。

M：文件的类型或文件的权限（rwx）是否改变。

5：文件MD5校验和是否改变（可以看成文件内容是否改变）。

D：设备的主从代码是否改变。

L：文件路径是否改变。

U：文件的属主（所有者）是否改变。

G：文件的属组是否改变。

T：文件的修改时间是否改变。

.：若相关项没发生改变，用 . 表示。

1. 被修改文件类型，大致可分为以下几类：

c：配置文件（configuration file）。

d：普通文档（documentation）。

g："鬼"文件（ghost file），很少见，就是该文件不应该被这个 RPM 包包含。

l：授权文件（license file）。

r：描述文件（read me）。

1. 被修改文件所在绝对路径（包含文件名）。

由此，S.5....T. c S.5....T. c /etc/httpd/conf/httpd.conf 表达的完整含义是：配置文件 httpd.conf 的大小、内容、修改时间被人为修改过。

注意，并非所有对文件做修改的行为都是恶意的。通常情况下，对配置文件做修改是正常的，比如说配置 apache 就要修改其配置文件，而如果验证信息提示对二进制文件做了修改，这就需要小心，除非是自己故意修改的。

#### 2 Linux RPM数字证书验证

RPM 包校验方法只能用来校验已安装的 RPM 包及其安装文件，如果 RPM 包本身就被动过手脚，此方法将无法解决问题，需要使用 RPM 数字证书验证方法。

简单的理解，RPM 包校验其实就是将现有安装文件与最初使用 RPM 包安装时的初始文件进行对比，如果有改动则提示给用户，因此这种方式无法验证 RPM 包本身被修改的情况。

数字证书，又称数字签名，由软件开发商直接发布。Linux 系统安装数字证书后，若 RPM 包做了修改，此包携带的数字证书也会改变，将无法与系统成功匹配，软件无法安装。  
  
可以将数字证书想象成自己的签名，是不能被模仿的（厂商的数字证书是唯一的），只有我认可的文件才会签名（只要是厂商发布的软件，都符合数字证书验证）；如果我的文件被人修改了，那么我的签名就会变得不同（如果软件改变，数字证书就会改变，从而通不过验证。当然，现实中人的手工签名不会直接改变，所以数字证书比手工签名还要可靠）。  
  
使用数字证书验证 RPM 包的方法具有如下 2 个特点：

1. 必须找到原厂的公钥文件，然后才能进行安装。
2. 安装 RPM 包会提取 RPM 包中的证书信息，然后和本机安装的原厂证书进行验证。如果验证通过，则允许安装；如果验证不通过，则不允许安装并发出警告。

数字证书默认会放到系统中/etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6位置处，通过以下命令也可验证：

#系统中的数字证书位置  
[root@localhost ~]# ll /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6  
-rw-r--r--.1 root root 1706 6 月 26 17:29 /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

安装数字证书的命令如下：

[root@localhost ~]# rpm --import /efc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6  
--import表示导入数字证书

数字证书安装完成后，可使用如下命令进行验证：

[root@localhost ~]# rpm -qa|grep gpg-pubkey  
gpg-pubkey-c105b9de-4e0fd3a3

可以看到，数字证书已成功安装。在装有数字证书的系统上安装 RPM 包时，系统会自动验证包的数字证书，验证通过则可以安装，反之将无法安装（系统会报错）。  
  
数字证书本身也是一个 RPM 包，因此可以用 rpm 命令查询数字证书的详细信息，也可以将其卸载。查询数字证书详细信息的命令如下：

[root@localhost ~]# rpm -qi gpg-pubkey-c105b9de-4e0fd3a3  
#查询数字证书包的详细信息  
Name : gpg-pubkey  
Relocations: (not relocatable)  
Version : c105b9de Vendor: (none)  
Release : 4e0fd3a3 Build Date: 2012年11月12日 星期一 23时05分20秒  
Install Date: 2012年11月12日星期一23时05分20秒 Build Host: local host  
Group : Public Keys  
Source RPM: (none)  
Size : 0  
License: pubkey  
…省略部分输出…  
-----END PGP PUBLIC KEY BLOCK----

卸载数字证书可以使用 -e 选项，命令如下：

[root@localhost ~]# rpm -e gpg-pubkey-c105b9de-4ead3a3

虽然数字证书可以手动卸载，但不推荐大家将其卸载。

3 Yum安装

**墙裂推荐使用互联网源**，不要使用本地包源

rpm包安装方式需要手动解决包之间具有**依赖性**的问题，尤其是库文件依赖，需要自行去 [http://www.rpmfind.net](http://www.rpmfind.net/" \t "http://c.biancheng.net/view/_blank) 网站上查找相关的 RPM 包。本节介绍一种可自动安装软件包（**自动解决包之间依赖关系**）的安装方式。

yum，全称“Yellow dog Updater, Modified”，是一个专门为了解决包的依赖关系而存在的软件包管理器。就好像 Windows 系统上可以通过 360 软件管家实现软件的一键安装、升级和卸载，Linux 系统也提供有这样的工具，就是 yum。可以这么说，yum 是改进型的 RPM 软件管理器，它很好的解决了 RPM 所面临的软件包依赖问题。**yum 在服务器端存有所有的 RPM 包，并将各个包之间的依赖关系记录在文件中**，**当管理员使用 yum 安装 RPM 包时**，**yum 会先从服务器端下载包的依赖性文件，通过分析此文件从服务器端一次性下载所有相关的 RPM 包并进行安装**。yum 软件可以用 rpm 命令安装，安装之前可以通过如下命令查看 yum 是否已安装：

[root@localhost ~]# rpm -qa | grep yum  
yum-metadata-parser-1.1.2-16.el6.i686  
yum-3.2.29-30.el6.centos.noarch  
yum-utils-1.1.30-14.el6.noarch  
yum-plugin-fastestmirror-1.1.30-14.el6.noarch  
yum-plugin-security-1.1.30-14.el6.noarch

可以看到，系统上已经安装了 yum。

使用 rpm 命令安装 yum 的具体方式可查看《[Linux怎么安装yum](https://jingyan.baidu.com/article/e3c78d6483a02a3c4d85f578.html" \t "http://c.biancheng.net/view/_blank)》一节。

使用 yum 安装软件包之前，需指定好 yum 下载 RPM 包的位置，此位置称为 yum 源。换句话说，yum 源指的就是软件安装包的来源。  
  
使用 yum 安装软件时至少需要一个 yum 源。yum 源既可以使用网络 yum 源，也可以将本地光盘作为 yum 源。接下来就给大家介绍这两种 yum 源的搭建方式。

1 网络 yum 源搭建

**墙烈建议优先使用网络源**

这个是yum的主要来源

操作系统版本的维护周期（10年）

一般情况下，只要你的主机网络正常，可以直接使用网络 yum 源，不需要对配置文件做任何修改，这里对 yum 源配置文件做一下简单介绍。  
  
网络 yum 源配置文件位于 /etc/yum.repos.d/ 目录下，文件扩展名为"\*.repo"（只要扩展名为 "\*.repo" 的文件都是 yum 源的配置文件）。

[root@localhost ~]# ls /etc/yum.repos.d/  
CentOS-Base.repo  
CentOS-Media.repo  
CentOS-Debuginfo.repo.bak  
CentOS-Vault.repo

可以看到，该目录下有 4 个 yum 配置文件，通常情况下 CentOS-Base.repo 文件生效。我们可以尝试打开此文件，命令如下：

[root@localhost yum.repos.d]# vim /etc/yum.repos.d/ CentOS-Base.repo  
[base]  
name=CentOS-$releasever - Base  
mirrorlist=http://mirrorlist.centos.org/? release= $releasever&arch=$basearch&repo=os  
baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/  
gpgcheck=1  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6  
…省略部分输出…

此文件中含有 5 个 yum 源容器，这里只列出了 base 容器，其他容器和 base 容器类似。base 容器中各参数的含义分别为：

* [base]：容器名称，一定要放在[]中。
* name：容器说明，可以自己随便写。
* mirrorlist：镜像站点，这个可以注释掉。
* **baseurl：我们的 yum 源服务器的地址。默认是 CentOS 官方的 yum 源服务器，是可以使用的。如果你觉得慢，则可以改成你喜欢的 yum 源地址。**
* **enabled：此容器是否生效，如果不写或写成 enabled 则表示此容器生效，写成 enable=0 则表示此容器不生效。**
* gpgcheck：如果为 1 则表示 RPM 的数字证书生效；如果为 0 则表示 RPM 的数字证书不生效。
* gpgkey：数字证书的公钥文件保存位置。不用修改。

2 本地 yum 源

# **[createrepo创建本地yum源](https://www.cnblogs.com/dsqx/articles/17439715.html)**

createrepo命令需要安装createrepo\_c软件包

createrepo命令来自英文词组“create repository”的缩写，其功能是用于创建软件仓库及生成元数据。使用createrepo命令可以让用户很方便地创建自己的YUM/dnf软件仓库资源，即为存放于本地的众多RPM软件包提供索引，自动化分析各软件包之间所需依赖关系，并生成元数据。 语法格式：createrepo [参数] [对象] 常用参数： -c 指定用于存放软件包和校验信息的目录 -d 使用SQLite来存储生成的元数据 -g 设置本地软件仓库的组划分 -i 指定用于初始元数据参数的文件 -o 设置元数据的输出位置 -p 使用整洁格式输出xml文件 -q 静默执行模式 -u 设置Base URL的地址 -v 显示执行过程详细信息 -x 设置需要排除的软件包 --update 更新软件仓库信息 参考示例 将指定的目录（需内含所需RPM软件包）制作成软件仓库： [root@linuxcool ~]# createrepo /Dir 将当前目录（需内含所需RPM软件包）制作成软件仓库： [root@linuxcool ~]# createrepo . 当前目录中RPM软件包发生变化后，重做软件仓库及元数据信息： [root@linuxcool ~]# createrepo --update .  
  
在 /etc/yum.repos.d/ 目录下有一个 CentOS-Media.repo 文件，此文件就是以本地光盘作为 yum 源的模板文件，只需进行简单的修改即可，步骤如下：

放入 **CentOS7** 安装光盘，并挂载光盘到指定位置。命令如下：

[root@localhost ~]# mkdir /mnt/cdrom  
#创建cdrom目录，作为光盘的挂载点  
[root@localhost ~]# mount /dev/cdrom /mnt/cdrom/  
mount: block device/dev/srO is write-protected, mounting read-only  
#挂载光盘到/mnt/cdrom目录下

修改其他几个 yum 源配置文件的扩展名，让它们失效，因为只有扩展名是"\*.repo"的文件才能作为 yum 源配置文件。当也可以删除其他几个 yum 源配置文件，但是如果删除了，当又想用网络作为 yum 源时，就没有了参考文件，所以最好还是修改扩展名。

命令如下：

[root@localhost ~]# cd /etc/yum.repos.d/  
[root@localhost yum.repos.d]#mv CentOS-Base, repo CentOS-Base.repo.bak  
[root@localhost yum.repos.d]#mv CentOS-Debuginfo.repo CentOS-Debuginfo.repo.bak  
[root@localhost yum.repos.d]# mv CentOS-Vault.repo CentOS-Vault.repo.bak

修改光盘 yum 源配置文件 CentOS-Media.repo，参照以下方修改：

[root@localhost yum.repos.d]# vim CentOS-Media.repo  
[c7-media]  
name=CentOS-$releasever - Media  
**baseurl=file:///mnt/cdrom**  
#地址为你自己的光盘挂载地址  
#file:///media/cdrom/  
#file:///media/cdrecorder/  
#注释这两个的不存在地址  
gpgcheck=1  
enabled=1  
#把enabled=0改为enabled=1, 让这个yum源配置文件生效

如此，本地 yum 源就配置完成了。

yum命令详解（查询、安装、升级和卸载软件包）

3 yum查询命令

使用 yum 对软件包执行查询操作，常用命令可分为以下几种：

yum list：查询所有已安装和可安装的软件包。例如：

[root@localhost yum.repos.d]# yum list  
#查询所有可用软件包列表  
Installed Packages  
#已经安装的软件包  
ConsdeKit.i686 0.4.1-3.el6  
@anaconda-CentOS-201207051201 J386/6.3  
ConsdeKit-libs.i686 0.4.1-3.el6 @anaconda-CentOS-201207051201 J386/6.3  
…省略部分输出…  
Available Packages  
#还可以安装的软件包  
389-ds-base.i686 1.2.10.2-15.el6 c6-media  
389-ds-base-devel.i686 1.2.10.2-15.el6 c6-media  
#软件名 版本 所在位置（光盘）  
…省略部分输出…

yum list 包名：查询执行软件包的安装情况。例如：

[root@localhost yum.repos.d]# yum list samba  
Available Packages samba.i686 3.5.10-125.el6 c6-media  
#查询 samba 软件包的安装情况

yum search 关键字：从 yum 源服务器上查找与关键字相关的所有软件包。例如：

[root@localhost yum.repos.d]# yum search samba  
#搜索服务器上所有和samba相关的软件包  
========================N/S Matched:  
samba =============================  
samba-client.i686：Samba client programs  
samba-common.i686：Files used by both Samba servers and clients  
samba-doc.i686: Documentation for the Samba suite  
…省略部分输出…  
Name and summary matches only, use"search all" for everything.

yum info 包名：查询执行软件包的详细信息。例如：

[root@localhost yum.repos.d]# yum info samba  
#查询samba软件包的信息  
Available Packages <-没有安装  
Name : samba <-包名  
Arch : i686 <-适合的硬件平台  
Version : 3.5.10 <―版本  
Release : 125.el6 <—发布版本  
Size : 4.9M <—大小  
Repo : c6-media <-在光盘上  
…省略部分输出…

4 yum安装命令

yum 安装软件包的命令基本格式为：

[root@localhost yum.repos.d]# yum -y install 包名

其中：

命令：install：表示安装软件包。

**选项 -y：**

**自动回答 yes。如果不加 -y，那么每个安装的软件都需要手工回答 yes；**

**静默安装方式**

例如使用此 yum 命令安装 gcc：

[root@localhost yum jepos.d]#yum -y install gcc  
#使用yum自动安装gcc

gcc 是 C 语言的编译器，鉴于该软件包涉及到的依赖包较多，建议使用 yum 命令安装。

5 yum 升级命令

使用 yum 升级软件包，需确保 yum 源服务器中软件包的版本比本机安装的软件包版本高。  
  
yum 升级软件包常用命令如下：

yum -y update：升级所有软件包。不过考虑到服务器强调稳定性，因此该命令并不常用。

yum -y update 包名：升级特定的软件包。

6 yum 卸载命令

**使用 yum 卸载软件包时，会同时卸载所有与该包有依赖关系的其他软件包，即便有依赖包属于系统运行必备文件，也会被 yum 无情卸载，带来的直接后果就是使系统崩溃。**

**除非你能确定卸载此包以及它的所有依赖包不会对系统产生影响，否则不要使用 yum 卸载软件包。**

yum 卸载命令的基本格式如下：

[root@localhost yum.repos.d]# yum remove 包名  
#卸载指定的软件包

例如，使用 yum 卸载 samba 软件包的命令如下：

[root@localhost yum.repos.d]# yum remove samba  
#卸载samba软件包

查看软件包

yum list all ##列出yum源仓库里面的所有可用的安装包

yum list installed ##列出所有已经安装的安装包

yum list available ##列出没有安装的安装包

#安装软件

yum install softwarename ##安装指定的软件

yum reinstall softarename ##重新安装指定的软件

yum localinstall 第三方software ##安装第三方文件并且会解决软件的依赖关系

yum remove softwarename ##卸装指定的软件

#查找软件的信息

yum info software ##查看软的信息

yum search keywords ##根据关键字查找到相关安装包软件的信息

yum whatprovides filename ##查找包含指定文件的相关安装包

#对于软件组

yum groups list ##列出软件组

yum groups install ##安装一个软件组

yum group remove ##卸载一个软件组

yum groups info ##查看一个软件组的信息

查看安装好的仓库

yum repolist

查看rpm安装包的安装

rpm -qa | grep

安装X Window System

yum groupinstall "X Window System"

安装图形界面软件GNOME(GNOME Desktop)

yum groupinstall "GNOME Desktop" "Graphical Administration Tools"

常见的Yum命令

|  |  |
| --- | --- |
| 命令 | 作用 |
| yum repolist all | 列出所有仓库 |
| yum list all | 列出仓库中所有软件包 |
| yum info 软件包名称 | 查看软件包信息 |
| yum install 软件包名称 | 安装软件包 |
| yum reinstall 软件包名称 | 重新安装软件包 |
| yum update 软件包名称 | 升级软件包 |
| yum remove 软件包名称 | 移除软件包 |
| yum clean all | 清除所有仓库缓存 |
| yum check-update | 检查可更新的软件包 |
| yum grouplist | 查看系统中已经安装的软件包组 |
| yum groupinstall 软件包组 | 安装指定的软件包组 |
| yum groupremove 软件包组 | 移除指定的软件包组 |
| yum groupinfo 软件包组 | 查询指定的软件包组信息 |

**原本以为故事到此就要结束了，可是人们发现Yum虽然解决了软件的依赖关系问题，但仍然还是存在分析不准确、内存占用量大、不能多人同时安装软件等硬伤。终于，在2015年随着Fedora 22系统的发布，红帽又给了我们一个新的选择—DNF。DNF实际上就是解决了上述问题的Yum软件仓库的提升版，行业内称之为Yum v4版本。**

**作为Yum软件仓库v3版本的接替者，DNF特别友好地继承了原有的命令格式，且使用习惯上也保持了一致。大家不用担心不会操作，我们来看一个例子。以前，安装软件用的命令是“yum install软件包名称”，那么现在则是“dnf install软件包名称”（也就是说，将yum替换成dnf即可）。**

**当然RHEL 8系统也照顾到了老用户的习惯问题，同时兼容并保留了yum和dnf两个命令，大家在实际操作中随意选择就好。甚至这两个命令的提示信息都基本一样，感知不到什么区别。**

4 实验

1.想要知道我的系统当中，目前为止已经安装了多个软件包？

#rpm -qa | wc -l

2.查询httpd软件包是否已经安装

#rpm -q httpd

3.利用qi ql qc选项查询httpd

#rpm -qi httpd

#rpm -ql httpd

#rpm -qc httpd

4.利用qf 查询/bin/ls文件是由哪个软件包安装的

#rpm -q -f /bin/ls

卸载tree

rpm -e tree

5 yum在线安装

yum repolist

yum info httpd

yum -y remove httpd

yum -y install tree

yum -y install httpd

5.挂载系统光盘

创建本地源操作

使用本地源

rk8.8的光盘

1 挂载光盘到系统里面

先把光盘加载到虚拟机

挂载

创建一个挂载目录

# mkdir /mnt/cdrom

挂载光盘到上面目录

# mount 设备文件名 挂载目录

光盘设备文件名叫/dev/cdrom /dev/sr0 (这个可以使用lsblk查看)

file://本地的文件所在的绝对路径

/mnt/cdrom

file:///mnt/cdrom/BaseOS

file:///mnt/cdrom/BaseOS

file:///mnt/cdrom/AppStream

选把/etc/yum.reso.d下面所有的repo文件移动到/tmp目录去

mv /etc/yum.repos.d/\* /tmp

创建repo文件，BaseOSrepo.repo

[localBaseOSrepo]

name=Rocky Linux local dvd

baseurl=file:///mnt/cdrom/BaseOS

创建好后，执行如下操作

清空 yum clean all

创建缓存 yum makecache

再测试

yum -y remove httpd

yum -y remove mysql

yum -y install httpd

yum -y install mysql

创建nginx安装repo，然后测试安装nginx

To set up the yum repository, create the file named /etc/yum.repos.d/nginx.repo with the following contents:

[nginx-stable]

name=nginx stable repo

baseurl=http://nginx.org/packages/centos/$releasever/$basearch/

gpgcheck=1

enabled=1

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

[nginx-mainline]

name=nginx mainline repo

baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/

gpgcheck=1

enabled=0

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

下面的选做，自己去创建repodata

以及利用rpm文件夹创建repodata

准备rpm

createrepo [options] <directory>

mount -t 文件系统类型 设备文件名称 挂载目录

Linux程序包管理--马哥

概述

API：Application Program Interface

ABI：Application Binary Interface

Unix-like,

ELF

Windows

exe, msi

库级别的虚拟化：

Linux: WinE

Windows: Cywin

系统级开发：

C/C++：httpd, vsftpd, nginx

go

应用级开发：

java/Python/perl/ruby/php：

java: hadoop, hbase, (jvm)

Python：openstack, (pvm)

perl: (perl)

ruby: (ruby)

php: (php)

C/C++程序格式：

源代码：文本格式的程序代码；

编译开发环境：编译器、头文件、开发库

二进制格式：文本格式的程序代码 --> 编译器 --> 二进制格式（二进制程序、库文件、配置文件、帮助文件）

java/python程序格式：

源代码：编译成能够在其虚拟机(jvm/pvm)运行的格式；

开发环境：编译器、开发库

二进制

项目构建工具：

c/c++: make

java: maven

程序包管理器：

源代码 --> 目标二进制格式（二进制程序、库文件、配置文件、帮助文件） --> 组织成为一个或有限几个“包”文件；

安装、升级、卸载、查询、校验

程序包管理器：

debian：dpt, dpkg, ".deb"

redhat：redhat package manager, rpm, ".rpm"； rpm is package manager；

S.u.S.E：rpm, ".rpm",

Gentoo：ports

ArchLinux：

源代码：name-VERSION.tar.gz

VERSION：major.minor.release

rpm包命名格式：

name-VERSION-release.arch.rpm

VERSION：major.minor.release

release.arch：rpm包的发行号

release.os: 2.el7.i386.rpm

archetecture：i386, x64(amd64), ppc, noarch

redis-3.0.2.targz --> redis-3.0.2-1.centos7.x64.rpm

拆包：主包和支包

主包：name-VERSION-release.arch.rpm

支包：name-function-VERSION-release.arch.rpm

function：devel, utils, libs, ...

依赖关系：

X, Y, Z

X --> Y,Z

Y --> A, B, C

C --> Y

前端工具：自动解决依赖关系；

yum：rhel系列系统上rpm包管理器的前端工具；

apt-get (apt-cache)：deb包管理器的前端工具；

zypper：suse的rpm管理器前端工具；

dnf：Fedora 22+系统上rpm包管理器的前端工具；

程序包管理器：

功能：将编译好的应用程序的各组成文件打包成一个或几个程序包文件，从而更方便地实现程序包的安装、升级、卸载和查询等管理操作；

1、程序包的组成清单（每个程序包都单独实现）；

文件清单

安装或卸载时运行的脚本

2、数据库（公共）

程序包的名称和版本；

依赖关系；

功能说明；

安装生成的各文件的文件路径及校验码信息；

等等等

/var/lib/rpm/

获取程序包的途径：

(1) 系统发行版的光盘或官方的文件服务器（或镜像站点）：

http://mirrors.aliyun.com,

http://mirrors.sohu.com,

http://mirrors.163.com

(2) 项目的官方站点

(3) 第三方组织：

(a) EPEL

(b) 搜索引擎

http://pkgs.org

http://rpmfind.net

http://rpm.pbone.net

(4) 自动动手，丰衣足食

建议：检查其合法性

来源合法性；

程序包的完整性；

CentOS系统上rpm命令管理程序包：

安装、升级、卸载、查询和校验、数据库维护

rpm命令：rpm {functions} [OPTIONS] [PACKAGE\_FILE]

安装：-i, --install

升级：-U, --update, -F, --freshen

卸载：-e, --erase

查询：-q, --query

校验：-V, --verify

数据库维护：--builddb, --initdb

安装：

rpm {-i|--install} [install-options] PACKAGE\_FILE ...

rpm -ivh PACKAGE\_FILE ...

GENERAL OPTIONS：

-v：verbose，详细信息

-vv：更详细的输出

[install-options]：

-h：hash marks输出进度条；每个#表示2%的进度；

--test：测试安装，检查并报告依赖关系及冲突消息等；

--nodeps：忽略依赖关系；不建议；

--replacepkgs：重新安装

注意：rpm可以自带脚本；

四类：--noscripts

preinstall：安装过程开始之前运行的脚本，%pre ， --nopre

postinstall：安装过程完成之后运行的脚本，%post , --nopost

preuninstall：卸载过程真正开始执行之前运行的脚本，%preun, --nopreun

postuninstall：卸载过程完成之后运行的脚本，%postun , --nopostun

--nosignature：不检查包签名信息，不检查来源合法性；

--nodigest：不检查包完整性信息；

升级：

rpm {-U|--upgrade} [install-options] PACKAGE\_FILE ...

rpm {-F|--freshen} [install-options] PACKAGE\_FILE ...

-U：升级或安装；

-F：升级

rpm -Uvh PACKAGE\_FILE ...

rpm -Fvh PACKAGE\_FILE ...

--oldpackage：降级；

--force：强制升级；

注意：(1) 不要对内核做升级操作；Linux支持多内核版本并存，因此，直接安装新版本内核；

(2) 如果某原程序包的配置文件安装后曾被修改过，升级时，新版本的程序提供的同一个配置文件不会覆盖原有版本的配置文件，而是把新版本的配置文件重命名(FILENAME.rpmnew)后提供；

卸载：

rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--test] PACKAGE\_NAME ...

--allmatches：卸载所有匹配指定名称的程序包的各版本；

--nodeps：忽略依赖关系

--test：测试卸载，dry run模式

查询：

rpm {-q|--query} [select-options] [query-options]

[select-options]

PACKAGE\_NAME：查询指定的程序包是否已经安装，及其版本；

-a, --all：查询所有已经安装过的包；

-f FILE：查询指定的文件由哪个程序包安装生成；

-p, --package PACKAGE\_FILE：用于实现对未安装的程序包执行查询操作；

--whatprovides CAPABILITY：查询指定的CAPABILITY由哪个程序包提供；

--whatrequires CAPABILITY：查询指定的CAPABILITY被哪个包所依赖；

[query-options]

--changelog：查询rpm包的changlog；

-l, --list：程序安装生成的所有文件列表；

-i, --info：程序包相关的信息，版本号、大小、所属的包组，等；

-c, --configfiles：查询指定的程序包提供的配置文件；

-d, --docfiles：查询指定的程序包提供的文档；

--provides：列出指定的程序包提供的所有的CAPABILITY；

-R, --requires：查询指定的程序包的依赖关系；

--scripts：查看程序包自带的脚本片断；

用法：

-qi PACKAGE, -qf FILE, -qc PACKAGE, -ql PACKAGE, -qd PACKAGE

-qpi PACKAGE\_FILE, -qpl PACKAGE\_FILE, -qpc PACKAGE\_FILE, ...

校验：

rpm {-V|--verify} [select-options] [verify-options]

S file Size differs

M Mode differs (includes permissions and file type)

5 digest (formerly MD5 sum) differs

D Device major/minor number mismatch

L readLink(2) path mismatch

U User ownership differs

G Group ownership differs

T mTime differs

P caPabilities differ

包来源合法性验正和完整性验正：

来源合法性验正：

完整性验正：

获取并导入信任的包制作者的密钥：

对于CentOS发行版来说：rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

验正：

(1) 安装此组织签名的程序时，会自动执行验正；

(2) 手动验正：rpm -K PACKAGE\_FILE

数据库重建：

rpm管理器数据库路径：/var/lib/rpm/

查询操作：通过此处的数据库进行；

获取帮助：

CentOS 6：man rpm

CentOS 7：man rpmdb

rpm {--initdb|--rebuilddb} [--dbpath DIRECTORY] [--root DIRECTORY]

--initdb：初始化数据库，当前无任何数据库可实始化创建一个新的；当前有时不执行任何操作；

--rebuilddb：重新构建，通过读取当前系统上所有已经安装过的程序包进行重新创建；

博客作业：rpm包管理功能全解；

回顾：Linux程序包管理的实现、rpm包管理器

rpm命令实现程序管理：

安装：-ivh, --nodeps, --replacepkgs

卸载：-e, --nodeps

升级：-Uvh, -Fvh, --nodeps, --oldpackage

查询：-q, -qa, -qf, -qi, -qd, -qc, -q --scripts, -q --changlog, -q --provides, -q --requires

校验：-V

导入GPG密钥：--import, -K, --nodigest, --nosignature

数据库重建：--initdb, --rebuilddb

Linux程序包管理(2)

CentOS: yum, dnf

URL: ftp://172.16.0.1/pub/

YUM: yellow dog, Yellowdog Update Modifier

yum repository: yum repo

存储了众多rpm包，以及包的相关的元数据文件（放置于特定目录下：repodata）；

文件服务器：

ftp://

http://

nfs://

file:///

yum客户端：

配置文件：

/etc/yum.conf：为所有仓库提供公共配置

/etc/yum.repos.d/\*.repo：为仓库的指向提供配置

仓库指向的定义：

[repositoryID]

name=Some name for this repository

baseurl=url://path/to/repository/

enabled={1|0}

gpgcheck={1|0}

gpgkey=URL

enablegroups={1|0}

failovermethod={roundrobin|priority}

默认为：roundrobin，意为随机挑选；

cost=

默认为1000

教室里的yum源：http://172.16.0.1/cobbler/ks\_mirror/CentOS-6.6-x86\_64/

CentOS 6.6 X84\_64 epel: http://172.16.0.1/fedora-epel/6/x86\_64/

yum命令的用法：

yum [options] [command] [package ...]

command is one of:

\* install package1 [package2] [...]

\* update [package1] [package2] [...]

\* update-to [package1] [package2] [...]

\* check-update

\* upgrade [package1] [package2] [...]

\* upgrade-to [package1] [package2] [...]

\* distribution-synchronization [package1] [package2] [...]

\* remove | erase package1 [package2] [...]

\* list [...]

\* info [...]

\* provides | whatprovides feature1 [feature2] [...]

\* clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]

\* makecache

\* groupinstall group1 [group2] [...]

\* groupupdate group1 [group2] [...]

\* grouplist [hidden] [groupwildcard] [...]

\* groupremove group1 [group2] [...]

\* groupinfo group1 [...]

\* search string1 [string2] [...]

\* shell [filename]

\* resolvedep dep1 [dep2] [...]

\* localinstall rpmfile1 [rpmfile2] [...]

(maintained for legacy reasons only - use install)

\* localupdate rpmfile1 [rpmfile2] [...]

(maintained for legacy reasons only - use update)

\* reinstall package1 [package2] [...]

\* downgrade package1 [package2] [...]

\* deplist package1 [package2] [...]

\* repolist [all|enabled|disabled]

\* version [ all | installed | available | group-\* | nogroups\* | grouplist | groupinfo ]

\* history [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|new|sync|stats]

\* check

\* help [command]

显示仓库列表：

repolist [all|enabled|disabled]

显示程序包：

list

# yum list [all | glob\_exp1] [glob\_exp2] [...]

# yum list {available|installed|updates} [glob\_exp1] [...]

安装程序包：

install package1 [package2] [...]

reinstall package1 [package2] [...] (重新安装)

升级程序包：

update [package1] [package2] [...]

downgrade package1 [package2] [...] (降级)

检查可用升级：

check-update

卸载程序包：

remove | erase package1 [package2] [...]

查看程序包information：

info [...]

查看指定的特性(可以是某文件)是由哪个程序包所提供：

provides | whatprovides feature1 [feature2] [...]

清理本地缓存：

clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]

构建缓存：

makecache

搜索：

search string1 [string2] [...]

以指定的关键字搜索程序包名及summary信息；

查看指定包所依赖的capabilities：

deplist package1 [package2] [...]

查看yum事务历史：

history [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|new|sync|stats]

安装及升级本地程序包：

\* localinstall rpmfile1 [rpmfile2] [...]

(maintained for legacy reasons only - use install)

\* localupdate rpmfile1 [rpmfile2] [...]

(maintained for legacy reasons only - use update)

包组管理的相关命令：

\* groupinstall group1 [group2] [...]

\* groupupdate group1 [group2] [...]

\* grouplist [hidden] [groupwildcard] [...]

\* groupremove group1 [group2] [...]

\* groupinfo group1 [...]

如何使用光盘当作本地yum仓库：

(1) 挂载光盘至某目录，例如/media/cdrom

# mount -r -t iso9660 /dev/cdrom /media/cdrom

(2) 创建配置文件

[CentOS7]

name=

baseurl=

gpgcheck=

enabled=

yum的命令行选项：

--nogpgcheck：禁止进行gpg check；

-y: 自动回答为“yes”；

-q：静默模式；

--disablerepo=repoidglob：临时禁用此处指定的repo； nginx 1.2 1.14

--enablerepo=repoidglob：临时启用此处指定的repo；

--noplugins：禁用所有插件；

yum的repo配置文件中可用的变量：

$releasever: 当前OS的发行版的主版本号；

$arch: 平台；

$basearch：基础平台；

$YUM0-$YUM9

http://mirrors.magedu.com/centos/$releasever/$basearch/os

创建yum仓库：

createrepo [options] <directory>

程序包编译安装：

testapp-VERSION-release.src.rpm --> 安装后，使用rpmbuild命令制作成二进制格式的rpm包，而后再安装；

源代码 --> 预处理 --> 编译(gcc) --> 汇编 --> 链接 --> 执行

源代码组织格式：

多文件：文件中的代码之间，很可能存在跨文件依赖关系；

C、C++： make (configure --> Makefile.in --> makefile)

java: maven

C代码编译安装三步骤：

./configure：

(1) 通过选项传递参数，指定启用特性、安装路径等；执行时会参考用户的指定以及Makefile.in文件生成makefile；

(2) 检查依赖到的外部环境；

make：

根据makefile文件，构建应用程序；

make install

开发工具：

autoconf: 生成configure脚本

automake：生成Makefile.in

建议：安装前查看INSTALL，README

开源程序源代码的获取：

官方自建站点：

apache.org (ASF)

mariadb.org

...

代码托管：

SourceForge

Github.com

code.google.com

c/c++: gcc (GNU C Complier)

编译C源代码：

前提：提供开发工具及开发环境

开发工具：make, gcc等

开发环境：开发库，头文件

glibc：标准库

通过“包组”提供开发组件

CentOS 6: "Development Tools", "Server Platform Development",

第一步：configure脚本

选项：指定安装位置、指定启用的特性

--help: 获取其支持使用的选项

选项分类：

安装路径设定：

--prefix=/PATH/TO/SOMEWHERE: 指定默认安装位置；默认为/usr/local/

--sysconfdir=/PATH/TO/SOMEWHERE：配置文件安装位置；

System types:

Optional Features: 可选特性

--disable-FEATURE

--enable-FEATURE[=ARG]

Optional Packages: 可选包

--with-PACKAGE[=ARG]

--without-PACKAGE

第二步：make

第三步：make install

安装后的配置：

(1) 导出二进制程序目录至PATH环境变量中；

编辑文件/etc/profile.d/NAME.sh

export PATH=/PATH/TO/BIN:$PATH

(2) 导出库文件路径

编辑/etc/ld.so.conf.d/NAME.conf

添加新的库文件所在目录至此文件中；

让系统重新生成缓存：

ldconfig [-v]

(3) 导出头文件

基于链接的方式实现：

ln -sv

(4) 导出帮助手册

编辑/etc/man.config文件

添加一个MANPATH

练习：

1、yum的配置和使用；包括yum repository的创建；

2、编译安装apache 2.2; 启动此服务；

# 设置yum安装包缓存位置

场景适用

在公司内部网络, 离线(无法访问外网的情况下), 我们可以先将所需要的rpm包下载好, 传进服务器进行离线安装

yum 可以配置将安装的软件包(rpm包)进行缓存

配置

更改 yum.conf

vim /etc/yum.conf

# cachedir 为安装包缓存的路径

# keepcache 表示是否进行缓存: 1 是, 0 否

[main]

cachedir=/var/cache/yum/$basearch/$releasever # 可自行修改

keepcache=1

缓存的rpm包一般在: update/packages 或者 base/packages 下, 当然其他的也有可能

一键下载 rpm 所有的依赖

yumdownloader httpd --resolve --destdir=./

# httpd 是你想要下载的包名

# --resolve 下载其 rpm 包及其依赖包

# destdir 指定文件夹，可以不指定(当前目录下)

说明：yumdownloader需要安装yum-utils

一键安装所有rpm包

rpm -ivh ./\*.rpm --nodeps --force

# --force 强制安装

# --nodeps 忽略依赖安装

createrepo命令需要安装createrepo\_c