软件包安装管理

# 源码包管理

首先我们先来看一下源代码的方式。我们知道，在开源的环境下，大多数的开源软件都是以源代码的形式来发布，通常将源代码打包成tar.gz的归档压缩文件发布到网上供我们下载使用。但是我们下载下来的源代码方式我们还不能够直接使用，不像在windows系统上直接下载下来可执行的二进制文件，我们需要将下载好的源码编译成可执行的二进制文件才能运行使用。

## 程序编译安装

编写源代码->编译源码生成可执行文件->复制该文件到PATH包含的目录

## 源代码包安装

源代码形式安装流程一般如下：

1. ./configure:检查编译环境、相关库文件以及配置参数并生成makefile
2. make：将源代码编译成可执行的二进制文件
3. make install：安装编译好的可执行文件

总结一下：

* 源代码形式的缺点：操作复杂、编译时间长、极易出现错误
* 源代码形式的优点：适用于所有的操作系统、可定制

应用举例（proftpd）：

|  |
| --- |
| tar -xzvf proftpd-1.3.3d.tar.gz  cd proftpd-1.3.3d  #生成Makefile里面包含定义好的功能选项和检测系统环境的信息  ./configure --prefix=/usr/local/proftpd  make （编译）  make install （安装）  注意：源代码包卸载：把安装文件删除就好了。 |

# RPM软件包管理

因为源代码方式安装软件终究还是比较麻烦，所以说现在出现了许多代替源码方式安装的软件管理机制。RPM就是其中一个，RPM（redhat Package Manager）是为了方便软件管理使用所开发的一套开源软件格式，它的设计目标有以下几个：

* 使用简单通过 rpm 命令就可以完成rpm软件的安装
* 以单一软件包格式发布（.rpm文件）　　我们可以看到rpm软件包都只有一个以.rpm的文件(rpm软件包管理机制要求软件包必须以.rpm结尾)
* 可升级：rpm软件包是可以升级的，比如说我下载好了一个新版本的rpm软件，我们只需要通过 rpm -u 命令即可升级
* 追踪依赖关系　rpm软件包管理机制可以追踪软件之间的依赖关系，这样在安装时就会非常的方便
* 保存软件基本信息
* 软件验证功能
* 支持多平台：支持多平台并不是说一个rpm软件可以在多个操作系统平台上运行，而是说该软件会针对多个平台发布不同的rpm软件包

rpm 搜索路径/var/lib/rpm数据库, 安装更新使用全包名，卸载查询校验使用包名，因为包名已经存在。

## RPM软件包常用的命名规范

sudo-1.7.2p1-5.el5.i386.rpm

|  |
| --- |
| sudo：软件名，  1.7.2p1：软件的版本号，  5.el5：发型号，  i386：硬件平台。 |

## RPM包的使用和管理

### Rpm包安装

**rpm  –i[v|h]  [--excludedocs |prefix |test |replace[pkgs |files] |nodeps]  sudo**

[解析]  i为安装，v为显示详细信息，h为hash是进度信息

--excludedocs 不安装软件包的文档文件

--prefix PATH 将软件包安装到指定目录

--test 只对安装进行测试，并不实际安装（通常用来看下需要的依赖项）

--replace[pkgs|files] 当出现冲突时或想强制覆盖时可以使用

--nodeps 不考虑依赖关系，强制安装。（这样比较麻烦，而且通常无法使用。这也是使用RPM的一些不方便的地方，而YUM则解决了这个问题，在安装的时候会自动将所需要的都一起安装了）

#强行忽略依赖安装

rpm –ivh Package\_Name-Version.rpm --force --nodeps

### Rpm包查询

rpm **–**qa：列出所有已安装的rpm软件

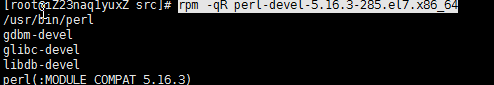
rpm -qf filename：查询指定文件属于哪个rpm包

rpm -qi packagename(全包名或者软件名)：查询软件包的详细信息

rpm -ql packagename(全包名）**：**查询指定软件包包含的文件及安装位置

rpm -qR packagename：  
常见使用：

|  |
| --- |
| #系统是否安装某个rpm包  rpm –pa | grep “要查找的rmp包”  #找到对应的安装位置  rpm –ql rpm对应的软件名 |



### Rpm包验证

rpm -V softname(包名)

* S 为文件大小
* 5 为文件的md5校验值
* L 链接文件
* T 文件创建时间
* D 设备文件
* U 文件的用户
* G 文件的用户组
* M 文件的权限

### Rpm包升级

rpm -Uvh 软件名

### Rpm包卸载

rpm –e [--test] [--nodeps] 软件包名(全名)

选项：

--test：只执行删除测试

--nodeps：不检查软件的依赖关系，强制删除

# YUM软件包管理(二进制)

首先必须先明确一点，yum并不是一种新的软件包管理形式，我们的rpm则是一种新的软件包管理形式，yum只是rpm的一个前端程序，yum最主要的功能就是帮助我们解决软件包的依赖性问题！

YUM（全称为 Yellow dog Updater, Modified）是一个在Fedora和RedHat以及SUSE、CentOS中的Shell前端软件包管理器。基于RPM管理，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。

## yum的特点

①**自动解决依赖性问题：**这个是yum最主要的功能，我们可以通过yum来自动的解决软件包直接的依赖性问题

②可以对RPM进行分组，并基于组来进行安装操作：有时候我们可能需要对软件进行分组，比如基于开发的软件放在一组，桌面软件又放在一组，在传统的RPM包管理机制里，我们需要自己手动的去将每一个组里面的软件都安装了才行，而yum就可以方便的对RPM进行分组，我只需要安装这个组，那么这个组里面的所有软件都会自动给我们安装好。

③引入了仓库的概念，支持多个仓库：通过yum来进行软件的安装其实就是基于仓库这个概念的，我将所有的rpm软件包都事先放在一个仓库里面，然后yum就可以在这个仓库里面寻找我们需要安装的软件，找到了就可以自动安装，没找到则就安装不了

④配置简单：在现在的RHEL、CentOS里面基本上都使用yum的方式来进行软件的安装，其主要原因也是基于其仓库的配置非常的简单。

## yum仓库

yum里引入了仓库的概念，仓库里其实就是用来存放我们所有现有的rpm软件包，因为yum就是基于rpm软件包管理的一个前端程序，所以yum也是通过rpm来进行软件的安装的。当使用yum进行软件安装时，如果存在依赖关系，那么yun也会自动的去仓库里面去寻找所依赖的rpm软件包，并自动安装，解决软件包的依赖问题。这个仓库既可以是本地的，也可以是互联网上的，可以通过http、ftp或者nfs等形式使用集中的、统一的网络仓库。

yum使用的是仓库来保持管理我们的rpm软件包，仓库的配置文件是存放在 /etc/yum.repos.d/ 这个文件目录下。格式如下：

|  |
| --- |
| #通常是一些对该仓库进行描述的语句  name=This is rpm soft repo  #这个是仓库的地址，可以是http、ftp、nfs等网络仓库的地址，  #也可以是我们本地仓库的地址  baseurl=http://www.xiaoluo.com/yum/soft/CentOS/x86\_64/rpms  # enabled=1表示我们使用这个仓库，默认就是enabled=1  enabled=1  #是否启用软件校验，基于安全考虑，主流的Linux发行版本都会对一些常用的  #rpm进行签名操作  gpgcheck=1 |

使用yum仓库需要注意一下几点：

|  |
| --- |
| 1. 仓库可以使用http、ftp、file、nfs等方式 2. yum配置文件必须以.repo结尾 3. 一个yum配置文件里面可以存放多个仓库的配置信息 4. /etc/yum.repos.d/ 目录下可以存放多个yum配置文件 |

## yum基本命令

### 安装操作

**yum install software-name**

**yum groupinstall group-name**

### 升级操作

yum check-update

**yum update**

**yum update software-name**

**yum groupupdate group-name**

### 查找操作

#列出全部的、安装的、最近的、更新的软件

yum list (all | installed | recent | updates)　

### 删除操作

yum remove **software-name**

yum groupremove **group-name**

### 清除操作

#清除暂缓的rmp包

yum clean all

注意：yum remove/update(不安全); 如果yum找不到包，可以添加取消注释baseurl=<http://mirrors.sohu.com/centos/6/os/x86_64/>

## 创建网络yum源

运维大规模的服务器，使用本地源不现实。创建步骤：

安装Apache服务->将安装介质共享出来)->在客户机上配置repo文件

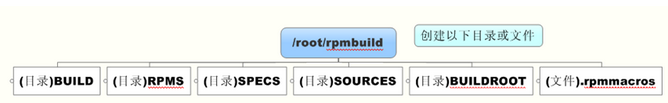
|  |
| --- |
| name=my repo  #设置apache存放资源地址  baseurl=http://ip/…  enabled=1  gpgcheck=1 |

# 重建RPM包

从大规模运维的角度来说，安全性，高效，易管理是排在第一位的，所以必须采用更方便包管理方式。如果想要同时享有编译软件和包管理的优点（高效运行和集中管理），那么我们就必须自己编译rpm包，同时使用包管理工具将这些包安装在同平台的服务器中，这就是重建RPM包。

## 创建重建环境

|  |
| --- |
| #安装完成后，会在当前用户主目录中创建几个目录  yum -y install rpm-build  rpmbuild --version  yum -y install gcc make |



* --BUILD #编译之前，如解压包后存放的路径
* --BUILDROOT #编译后存放的路径
* --RPMS #打包完成后rpm包存放的路径
* --SOURCES #源包所放置的路径
* --SPECS #spec文档放置的路径
* --SPRMS #源码rpm包放置的路径

## rpmbuild相关命令

基本格式：rpmbuild [options] [spec文档|tarball包|源码包]

以下依次列出三种不同文件：

1. 从spec文档建立有以下选项：

* -bp #只执行spec的%pre 段(解开源码包并打补丁，即只做准备)
* -bc #执行spec的%pre和%build 段(准备并编译)
* -bi #执行spec中%pre，%build与%install(准备，编译并安装)
* -bl #检查spec中的%file段(查看文件是否齐全)
* -ba #建立源码与二进制包(常用)
* -bb #只建立二进制包(常用)
* -bs #只建立源码包

## 从tarball包建立，与spec类似

* -tp #对应-bp
* -tc #对应-bc
* -ti #对应-bi
* -ta #对应-ba
* -tb #对应-bb
* -ts #对应-bs

## 从源码包建立

* --rebuild #建立二进制包，通-bb
* --recompile #同-bi

## 其他的一些选项

* --buildroot=DIRECTORY #确定以root目录建立包
* --clean #完成打包后清除BUILD下的文件目录
* --nobuild #不进行%build的阶段
* --nodeps #不检查建立包时的关联文件
* --nodirtokens
* --rmsource #完成打包后清除SOURCES
* --rmspec #完成打包后清除SPEC
* --short-cricuit
* --target=CPU-VENDOR-OS #确定包的最终使用平台

## 以spec文件重建RPM包

--下载源码到SOURCES目录

--SPECS目录中创建webbench.spec文件

|  |
| --- |
| Name: webbench#软件包名称  version: 1.5 #软件包版本  Release: 1%{?dist} #rpm发行版本  Summary: WEB test tools #软件包描述  Group: system administration tools #rpm包属组  License: GPL #授权信息  URL: http://home.tiscali.cz #软件主页  Source0: http://home.tiscali.cz/~cz210552/distfiles/webbench-1.5.tar.gz #源代码包名称，取URL最后/的名称  BuildRoot: %(mktemp -ud %{\_tmppath}/%{name}-%{version}-%{release}-XXXXXX)# make install安装路径  BuildRequires: gcc,make #打包的时候需要的软件包  %description #rpm包描述信息test webbench  %prep #预处理脚本  %setup -q #解压源码包  %build #开始构建  make %{?\_smp\_mflags} #执行make%install #make install  rm -rf %{buildroot} #先删除旧的目录  mkdir %{buildroot}/usr/local/ -p #创建新的目录mkdir %{buildroot}/usr/local/man/man1 -p  make install DESTDIR=%{buildroot} #执行make install安装在虚拟目录下  mkdir %{buildroot}/usr/bin -pmv %{buildroot}/usr/local/bin %{buildroot}/usr/bin/webbench |

--打包命令执行

rpmbuild -ba webbench.spec

## Spec文件简介

Spec文件有一定的模板，一般来说分为preamle(序言),prep(前期准备),build(编译),install(安装),clean(清理),files(文件列表),changelog(修改日志)这几个部分。

### preamble(序言)

Name:软件包的名称，后面可使用%{name}的方式引用，具体命令需跟源包一致  
Summary: 软件包的内容概要  
Version: 软件的实际版本号，具体命令需跟源包一致  
Release: 发布序列号，具体命令需跟源包一致

License: 软件授权方式，通常就是GPL  
Group: 软件分组，建议使用标准分组

-----------------------------------------  
软件包所属类别，具体类别有：  
Amusements/Games （娱乐/游戏）  
Amusements/Graphics（娱乐/图形）  
Applications/Archiving （应用/文档）  
Applications/Communications（应用/通讯）  
Applications/Databases （应用/数据库）  
Applications/Editors （应用/编辑器）  
Applications/Emulators （应用/仿真器）  
Applications/Engineering （应用/工程）  
Applications/File （应用/文件）  
Applications/Internet （应用/因特网）  
Applications/Multimedia（应用/多媒体）  
Applications/Productivity （应用/产品）  
Applications/Publishing（应用/印刷）  
Applications/System（应用/系统）  
Applications/Text （应用/文本）  
Development/Debuggers （开发/调试器）  
Development/Languages （开发/语言）  
Development/Libraries （开发/函数库）  
Development/System （开发/系统）  
Development/Tools （开发/工具）  
Documentation （文档）  
System Environment/Base（系统环境/基础）  
System Environment/Daemons （系统环境/守护）  
System Environment/Kernel （系统环境/内核）  
System Environment/Libraries （系统环境/函数库）  
System Environment/Shells （系统环境/接口）  
User Interface/Desktops（用户界面/桌面）  
User Interface/X （用户界面/X窗口）  
User Interface/X Hardware Support （用户界面/X硬件支持）

-----------------------------------------

BuildRoot: 编译使用的目录。

BuildPrereq：编译前需要满足的包

BuildRequires：编译时需要安装的包。

Source：源码包

Patch：补丁文件

Description：更详细的描述

Requires：安装该包时的依赖。

### prep(前期准备)

prep是预处理部分，以%prep开头，用于正式编译前的准备工作，包括删除老的源代码，解压源代码(%setup)，对源代码应用补丁(%patch)

### build(编译)

build是正式开始编译的部分，以%build开头，相当于./configure,make的工作。

### install(安装)

install用于完成实际的安装过程，以%install开头，相当于make install。

### clean(清理)

clean主要适用于安装完成后的清理工作，以%clean开头。

### files(文件列表)

file部分用于指定实际安装文件的放置目录和相关权限。以%files开头。这里所指定的所有文件都会被打包到最后生成的rpm包中，这些指定的文件分为三类，分别是说明文件，配置文件，可执行文件。

### changelog(修改日志)

changelog主要是注明该软件的开发记录，使用%changelog开头，主要的作用是让开发人员了解该软件以及经历的功能补全和bug修复。