**使用rpm进行软件管理**

**简介**

    Rpm原来是RedHat Linux下使用的软件包管理工具。本文介绍使用rpm相关工具制做和管理软件包。制作软件包就是根据源代码或者编译好的二进制文件生成rpm软件包。管理软件包就是使用rpm对软件包进行管理（安装，卸载，查询软件信息等）。重点讲解rpm软件包的制作。

**前言**

    什么是软件管理？这里提到的软件管理，范围主要限于软件的安装，卸载，查询。

    为便于理解，先给出一个虽然不太准确，但是比较直观的、手动管理软件的过程描述：a)当我们获取到软件源代码之后，将软件源代码进行编译，生成最终的可执行程序以及其依赖的库。b)将生成的库和可执行文件拷贝到机器中的系统目录中以便可以在一个统一的地方进行运行，拷贝完之后设置好一些环境变量（例如告诉系统可执行文件在哪里，甚至是添加一些菜单等），这样就可以运行软件了。c)当我们不需要软件的时候，将之前拷贝的文件从系统目录中删除，并把为了配置软件而设置的一些环境变量以及菜单恢复成修改之前的状态。

    对于以上的过程，其实b)就是对软件的安装过程。c)就是对软件的卸载。而a)过程是根据软件源代码编译生成可用的软件，一般对于最终用户来说，可能会没有这个步骤。从这个过程中我们可以看出，手动管理软件，优势是可以清楚地知道安装程序时在什么地方安装了哪些文件，卸载的时候也非常彻底，不会给系统留下垃圾文件；但是，却有很多缺点：首先，对用户要求较高。用户不仅会使用软件，而且要知道软件是如何来的，以及如何编译，整套软件具体包含哪些文件等等；其次，管理过程麻烦。一切拷贝都手动进行很容易出错；再次，维护不便。因为并没有什么机制来将自己曾经手动安装的过程一一记录下来，所以如果在机器上用这种方式安装许多软件，那么很难全部记住以前究竟安装了哪些软件，安装的软件是什么版本等，。

    针对以上缺点，rpm软件管理工具实现了软件的编译，安装，卸载自动处理，最初只需要得到软件的rpm包，然后双击或者通过一条特殊命令就可以实现软件的安装，省去了编译和手动拷贝的麻烦，而且rpm本身还提供了数据库，记录曾经安装过什么软件，以及软件的信息等，为软件维护也提供了方便。

    以下两点，可以帮助我们更容易了解所述内容：

    1)使用rpm软件管理与不使用rpm软件管理的情况

    并不是所有的软件都提供.rpm方式的安装。有些开发者会提供rpm方式的二进制软件包，以便用户选择使用。但只发布软件的源码，最后一部分的编译和安装需要用户自行解决也是很常见的。

    2)使用rpm管理软件的优势

    使用tar包或cpio等也可以打包文件，甚至比制作rpm要简单很多。但如果想在安装的过程中就把一套额外的软件部署完毕，那使用rpm方式通常是是更好的方法，而且rpm还有数据库协助软件升级、文件校验等，结合yum升级方式，特别是对于最终用户，提供rpm包几乎是最佳的方法。

    通过本文，我们将知道，在只有软件源代码的情况下，如何使用rpm工具制做相应的自动化软件安装包，以及一些基本的使用rpm管理软件的方法。以下将是本文主要内容，如果有问题或者建议可以联系我。

**主要内容**

主要内容包括

一、软件包制作

1，描述文件

2，制做rpm包的具体方法

二、一些软件包管理的命令

三、补充

**一、软件包制作**

**==========================**

**1，描述文件**

**--------------**

    制作rpm软件包的关键，就是编写用于生成软件包的spec文件（描述文件）。该文件描述了制作软件包时候所需的信息。下面介绍制作rpm包所需的spec文件的相关内容。

    首先，要在制做rpm包之前，建立好一些必要的目录结构，例如BUILD,SOURCES,等，然后rpm的相关工具会将这些目录的父目录以%{\_topdir}表示，这个%{\_topdir}默认值为系统中的某个目录，例如"/usr/local/redhat/"等，如果使用自己的目录，需要自行修改这个变量，后面的例子中会有所说明。

    spec脚本的主体中也包括了很多关键字，描述内容主要包含文件头以及各个段，主要的段包括：%prep段，%build段 ，%install段，%files段 ，%changelog段等，每个段大致代表一个处理阶段，下面依次介绍。更具体的信息，可以参见/usr/share/doc/rpm-<version>中的文档以及后面的参考资料。

**1)文件头**

    文件头内容主要如下：

Name:

    软件包的名称，后面可使用%{name}的方式引用

Summary:

    软件包的内容概要

Version:

    软件的实际版本号，例如：1.0.1等，后面可使用%{version}引用

Release:

    发布序列号，例如：1linuxing等，标明第几次打包，后面可使用%{release}引用

Group:

    软件分组，建议使用标准分组

License:

    软件授权方式，通常就是GPL

Source:

    源代码包，可以带多个用Source1、Source2等源，后面也可以用%{source1}、%{source2}引用

BuildRoot:

    这个是安装或编译时使用的“虚拟目录”，考虑到多用户的环境，一般定义为：

    %{\_tmppath}/%{name}-%{version}-%{release}-root或%{\_tmppath}/%{name}-%{version}-%{release}-buildroot-%(%{\_\_id\_u} -n}

    该参数非常重要，因为在生成rpm的过程中，执行make install时就会把软件安装到上述的路径中，在打包的时候，同样依赖“虚拟目录”为“根目录”进行操作。后面可使用$RPM\_BUILD\_ROOT 方式引用。

URL:

    软件的主页

Vendor:

    发行商或打包组织的信息，例如RedFlag Co,Ltd

Disstribution:

    发行版标识

Patch:

    补丁源码，可使用Patch1、Patch2等标识多个补丁，使用%patch0或%{patch0}引用

Prefix:

    %{\_prefix} 这个主要是为了解决今后安装rpm包时，并不一定把软件安装到rpm中打包的目录的情况。这样，必须在这里定义该标识，并在编写%install脚本的时候引用，才能实现rpm安装时重新指定位置的功能

Prefix:

    %{\_sysconfdir} 这个原因和上面的一样，但由于%{\_prefix}指/usr，而对于其他的文件，例如/etc下的配置文件，则需要用%{\_sysconfdir}标识

Build Arch:

    指编译的目标处理器架构，noarch标识不指定，但通常都是以/usr/lib/rpm/marcros中的内容为默认值

Requires:

    该rpm包所依赖的软件包名称，可以用>=或<=表示大于或小于某一特定版本，例如：

    libpng-devel >= 1.0.20 zlib

    这里，“>=”号两边需用空格隔开，而不同软件名称也用空格分开

    还有例如PreReq、Requires(pre)、Requires(post)、Requires(preun)、Requires(postun)、BuildRequires等都是针对不同阶段的依赖指定

Provides:

    指明本软件一些特定的功能，以便其他rpm识别

Packager:

    打包者的信息

%description

    软件的详细说明

**2)文件段**

**2.1)%prep段**

    包含预处理脚本，通常用来执行一些解开源程序包的命令，为下一步的编译安装作准备(一般来说，都是从SOURCES目录中将源代码解压到BUILD中)。脚本运行的当前目录在BUILD中。

    %prep和下面%build，%install段一样，除了可以执行RPM所定义的宏命令（以%开头）以外，还可以执行SHELL命令，命令可以有很多行。

    %setup -n %{name}-%{version} 把源码包解压并放好

    通常从%{\_topdir}/SOURCES里的包解压到%{\_topdir}/BUILD/%{name}-%{version}中。

    一般用%setup -c就可以了，但有两种情况：一就是同时编译多个源码包，二就是源码的tar包的名称与解压出来的目录不一致，此时，就需要使用-n参数指定一下了。

    对于%setup，具体还包含如下选项：

    %setup 不加任何选项，仅将软件包打开。

    %setup -n newdir 将软件包解压在newdir目录。

    %setup -c 解压缩之前先产生目录。

    %setup -b num 将第num个source文件解压缩。

    %setup -T 不使用default的解压缩操作。

    %setup -T -b 0 将第0个源代码文件解压缩。

    %setup -c -n newdir 指定目录名称newdir，并在此目录产生rpm套件。

    %patch 最简单的补丁方式，自动指定patch level。

    %patch 0 使用第0个补丁文件，相当于%patch ?p 0。

    %patch -s 不显示打补丁时的信息。

    %patch -T 将所有打补丁时产生的输出文件删除。

    %patch 打补丁

    通常补丁都会一起在源码tar.gz包中，或放到SOURCES目录下。一般参数为：

    %patch -p1 使用前面定义的Patch补丁进行，-p1是忽略patch的第一层目录

    %Patch2 -p1 -b xxx.patch 打上指定的补丁，-b是指生成备份文件

    %configure 这个不是关键字，而是rpm定义的标准宏命令。意思是执行源代码的configure配置

    在%{\_topdir}/BUILD/%{name}-%{version}目录中进行 ，使用标准写法，会引用/usr/lib/rpm/marcros中定义的参数。

    另一种不标准的写法是，可参考源码中的参数自定义，例如：

    CFLAGS="$RPM\_OPT\_FLAGS" CXXFLAGS="$RPM\_OPT\_FLAGS" ./configure --prefix=%{\_prefix}

**2.2)%build段**

    包含为生成软件包服务所执行的命令，通常是编译生成可执行文件。生成的可执行文件在install中会用到。脚本运行的当前目录在BUILD中。

    相当于在%{\_topdir}/BUILD/%{name}-%{version}目录中进行make的工作 ，常见写法如：make %{?\_smp\_mflags} OPTIMIZE="%{optflags}"都是一些优化参数，定义在/usr/lib/rpm/marcros。

**2.3)%install段**

    包含把软件安装到虚拟的根目录中的命令。这里的虚拟根目录就是一个子目录，其目录结构和根目录一样，%install会把最终安装好的软件的目录结构反应到这个虚拟根目录中。然后根据这个虚拟根目录和后面的%files段内容，将软件制作成rpm包。其当前工作目录也是BUILD。它一般会在%{\_topdir}/BUILD/%{name}-%{version}目录中进行make install的操作。这个很重要，因为如果这里的路径不对的话，则下面%file中寻找文件的时候就会失败。 常见内容有：

    %makeinstall

    这不是关键字，而是rpm定义的标准宏命令。也可以使用非标准写法：

    make DESTDIR=$RPM\_BUILD\_ROOT install

    make prefix=$RPM\_BUILD\_ROOT install

    需要说明的是，这里的%install主要就是为了后面的%file服务的。所以，还可以使用常规的系统命令：

    install -d $RPM\_BUILD\_ROOT/

    cp -a \* $RPM\_BUILD\_ROOT/

    总之，这里是制作rpm包，并不是安装软件，所以只是将文件安装到虚拟根目录中，千万要指定好了虚拟根目录，否则就真的安装到真实的系统中了，向虚拟根目录中安装好了之后，再利用虚拟根目录制作rpm包，而安装rpm包的时候，才真正将软件安装到你的系统上。

**2.4)%clean段**

    这个段用来清理临时文件，在制作好了rpm包之后，执行这个段中的内容，清除制作过程中产生的中间文件。例如BUILD中的，以及虚拟根目录中的内容。

    通常内容为：

    [ "$RPM\_BUILD\_ROOT" != "/" ] && rm -rf "$RPM\_BUILD\_ROOT"

    rm -rf $RPM\_BUILD\_DIR/%{name}-%{version}

    这里注意：对于$RPM\_BUILD\_ROOT和$RPM\_BUILD\_DIR：

    $RPM\_BUILD\_ROOT是指开头定义的BuildRoot，而$RPM\_BUILD\_DIR通常就是指%{\_topdir}/BUILD，其中，前面的才是%file需要的。

    %pre rpm安装前执行的脚本

    %post rpm安装后执行的脚本

    %preun rpm卸载前执行的脚本

    %postun rpm卸载后执行的脚本

    %preun %postun 的区别是什么呢？

    前者在升级的时候会执行，后者在升级rpm包的时候不会执行。

**2.5)%files段**

    定义哪些文件或目录会放入rpm中，rpm安装会安装哪些文件。

    用于定义软件包所包含的文件，分为三类--说明文档（doc），配置文件（config）及执行程序，还可定义文件存取权限，拥有者及组别。这里所指定的文件路径，是相对于虚拟根目录下的文件路径，千万不要写绝对路径，而应用宏或变量表示相对路径。 如果描述为目录，表示目录中除%exclude外的所有文件。

    %defattr (-,root,root) 指定包装文件的属性，分别是(mode,owner,group)，-表示默认值，对文本文件是0644，可执行文件是0755。

    %exclude 列出不想打包到rpm中的文件。这里注意，如果%exclude指定的文件不存在，也会出错的。

**2.6)%changelog段**

    本段是修改日志段。你可以将软件的每次修改记录到这里，保存到发布的软件包中，以便查询之用。

    每一个修改日志都有这样一种格式：

    第一行是：\* 星期 月 日 年 修改人 电子信箱。其中：星期、月份均用英文形式，用中文会出错。

    接下来的行写的应是修改了什么地方，可写多行。一般以减号起始，便于查阅。

**2.7)一些rpm相关杂乱信息**

\*以上各个段中，比较容易混淆的就是目录，我们需要记住两点：

    a）%build和%install中，都必须把编译和安装的文件定义到“虚拟根目录” 中。

    例如：

    %build

    make -f admin/Makefile.common cvs

    ./configure --prefix=%{\_prefix} --enable-final --disable-debug \

    --with-extra-includes=%{\_includedir}/freetype2 --includedir=%{\_includedir}

    make

    %install

    rm -fr $RPM\_BUILD\_ROOT

    make DESTDIR=$RPM\_BUILD\_ROOT install

    cp -r $RPM\_BUILD\_ROOT%{\_datadir}/apps/kolourpaint/icons/hicolor/\* $RPM\_BUILD\_ROOT%{\_datadir}/icons/crystalsvg/

    b）%file中用的是相对目录

    例如：

    %files

    %defattr(-,root,root)

    %{\_bindir}

    如果写成：

    %files

    %defattr(-,root,root)

    /usr/bin

    则打包的会是根目录下的/usr/bin中所有的文件，而不是本软件中的文件了。

\*

    rpm软件包系统的标准分组：/usr/share/doc/rpm-4.3.3/GROUPS

    各种宏定义： /usr/lib/rpm/macros

    已经安装的rpm包数据库： /var/lib/rpm

    如果要避免生成debuginfo包：这个是默认会生成的rpm包。则可以使用下面的命令：

    echo '%debug\_package %{nil}' >> ~/.rpmmacros

\*

    由于必须在%file中包括所有套件中的文件，所以，我们需要清楚编译完的套件到底包括那些文件？

    常见的做法是，人工模拟一次编译的过程：

    ./configrue --prefix=/usr/local/xxx

    make

    make DESTDIR=/usr/local/xxx install

    或

    make prefix=/usr/local/xxx install

    这样，整个套件的内容就会被放到/usr/local/xxx中，可根据情况编写%file和%exclude段(当然，这个只能对源码按GNU方式编写，并使用GNU autotool创建的包有效，若自定义Makefile则不能一概而论)。

\*关于rpm中的执行脚本

    如果正在制作的rpm包是准备作为放到系统安装光盘中的话，则需要考虑rpm中定义的脚本是否有问题。由于系统在安装的时候只是依赖于一个小环境进行，而该环境与实际安装完的环境有很大的区别，所以，大部分的脚本在该安装环境中都是无法生效，甚至会带来麻烦的。

    所以，对于这样的，需要放到安装光盘中的套件，不加入执行脚本是较佳的方法。

\*信号机制

    为提供操作中可参考的信息，rpm还提供了一种信号机制：不同的操作会返回不同的信息，并放到默认变量$1中。对于信号值，0代表卸载、1代表安装、2代表升级。

    例如可这样使用：

%postun

if [ "$1" = "0" ]; then

/sbin/ldconfig

fi

\*下面是宏对应路径一览表：

一些模仿autoconf的变量

%{\_sysconfdir} /etc

%{\_prefix} /usr

%{\_exec\_prefix}  %{\_prefix}

%{\_bindir}  %{\_exec\_prefix}/bin

%{\_lib} lib (lib64 on 64bit systems)

%{\_libdir}  %{\_exec\_prefix}/%{\_lib}

%{\_libexecdir}  %{\_exec\_prefix}/libexec

%{\_sbindir}  %{\_exec\_prefix}/sbin

%{\_sharedstatedir} /var/lib

%{\_datadir}  %{\_prefix}/share

%{\_includedir}  %{\_prefix}/include

%{\_oldincludedir} /usr/include

%{\_infodir} /usr/share/info

%{\_mandir} /usr/share/man

%{\_localstatedir} /var

%{\_initddir}  %{\_sysconfdir}/rc.d/init.d

注意：比Fedora10早的版本（或EPEL）中，%{\_initddir}没有。这时候可以使用旧的%{\_initrddir}宏。

RPM目录的宏

%{\_topdir}  %{getenv:HOME}/rpmbuild

%{\_builddir}  %{\_topdir}/BUILD

%{\_rpmdir}  %{\_topdir}/RPMS

%{\_sourcedir}  %{\_topdir}/SOURCES

%{\_specdir}  %{\_topdir}/SPECS

%{\_srcrpmdir}  %{\_topdir}/SRPMS

%{\_buildrootdir}  %{\_topdir}/BUILDROOT

注意：比Fedora10早的版本（或EPEL）中， %{\_buildrootdir} 没有。

编译编辑宏

%{\_global\_cflags} -O2 -g -pipe

%{\_optflags}  %{\_\_global\_cflags} -m32 -march=i386 -mtune=pentium4 # if redhat-rpm-config is installed

其它宏

%{\_var} /var

%{\_tmppath}  %{\_var}/tmp

%{\_usr} /usr

%{\_usrsrc}  %{\_usr}/src

%{\_docdir}  %{\_datadir}/doc

\*一个带有注释的较为完整的spec文件的例子

#+++++++++以下是文件内容+++++++++++#

*######头部######*

*Summary:一个简单的打印hello的程序*

*Name:myapplication*

*Version:1.0*

*Release:1*

*Vendor:QuietHeart (quiet\_heart000@126.com)*

*License:Share*

*SOURCE:%{name}-%{version}.tgz*

*BuildRoot:%{\_tmppath}/%{name}-%{version}-%{release}-root*

*Group:Applications/Editors*

*URL:quietheart.sf.net*

*Prefix:%{\_prefix}*

*Prefix:%{\_sysconfdir}*

*Prefix:%{\_datadir}*

*Packager:QuietHeart*

*%define myconfig myconfig.conf*

*%define mylib libmy.so*

*%description*

*描述省略。*

*######prep,build,install三个段都可以执行shell命令######*

*# 预处理段，为下一步的编译安装作准备。例如将源代码拷贝到BUILDS目录。*

*%prep*

*#   prep   start.*

*echo "======>In :prep section $(pwd)"*

*echo "======>In :prep section " %{\_topdir}*

*echo "======>In :prep section $RPM\_BUILD\_ROOT"*

*echo "======>In :prep section " %{\_sourcedir}*

*#echo "======>In :prep section $RPM\_SOURCE\_DIR"*

*#将源代码包解压并放好，也可以使用tar命令，解压好的目录:%{name}-%{version}，存在BUILD中。*

*%setup -n %{name}-%{version}*

*#tar -xzvf %{\_sourcedir}/%{name}-%{version}.tgz -C %{\_builddir}*

*# 建立段，执行一些编译命令以便生成软件包。工作于BUILDS。*

*%build*

*#   build   start.*

*echo "======>In :build section $(pwd)"*

*echo "======>In :build section " %{\_topdir}*

*echo "======>In :build section $RPM\_BUILD\_ROOT"*

*echo "======>In :build section " %{\_builddir}*

*#echo "======>In :build section $RPM\_BUILD\_DIR"*

*#自动进入myapplication-1.0中了。*

*make*

*# 安装段，在安装软件包阶段执行的命令。*

*# 安装的内容安装到“虚拟根目录中”install是为了后面的file段服务的。*

*%install*

*#   install   start.*

*echo "======>In :install section $(pwd)"*

*echo "======>In :install section " %{\_topdir}*

*echo "======>In :install section $RPM\_BUILD\_ROOT"*

*mkdir -p $RPM\_BUILD\_ROOT/usr/bin*

*mkdir -p $RPM\_BUILD\_ROOT/usr/lib*

*mkdir -p $RPM\_BUILD\_ROOT/etc*

*#这里使用%{name}表示开始定义的Name，即myapplication。*

*cp %{name} $RPM\_BUILD\_ROOT/usr/bin*

*cp lib/%{mylib} $RPM\_BUILD\_ROOT/usr/lib*

*cp %{myconfig} $RPM\_BUILD\_ROOT/etc*

*######其它的一些段######*

*%clean*

*#   clean   start.*

*echo "======>In :clean section $(pwd)"*

*#文件段，定义软件包所包含的文件。具体可由后面"%defattr","%doc","%config"宏定义。*

*#主要包含：config(配置文件),doc(文档文件),bin(二进制文件）*

*%files*

*#定义文件属性：%defattr(权限，属主，属组)。其中权限'-'表默认，可用'0777'形式。*

*%defattr (-,root,root)*

*#定义制作软件包包含的文档文件，每个宏定义一个，可指定绝对路径。*

*#一般文档文件会被安装到"/usr/doc/Name-Version-Release"目录下。*

*%doc   README*

*%doc   COPYRIGHT*

*#定义软件包所需要的配置文件(还是安装的配置文件位置??)*

*#当卸载软件的时候会将没有修改过的卸掉，修改过的备份为<name>.rpmsave。*

*%config   %{\_sysconfdir}/%{myconfig}*

*#软件包所包含的二进制文件，可以为目录*

*%{\_bindir}/%{name}*

*%{\_libdir}/%{mylib}*

*#修改日志段，可以记录每次的修改日志。*

*#第一行是：\*   星期   月   日   年   修改人   电子信箱。*

*#其中：星期、月份均用英文形式，用中文会出错。*

*#接下来的行写的应是修改了什么地方，可写多行。一般以减号起始，便于查阅。*

*#修改的时候，新的必须在上面。*

*%changelog*

*\*   Fri   Sep   30   2011   QuietHeart*

*-   增加XXX内容2*

*\*   Fri   Sep   29   2011   QuietHeart*

*-   增加XXX内容1*

#+++++++++以下是文件内容+++++++++++#

**2，制做rpm包的具体方法**

**--------------**

**2.1，工具**

    通过rpmbuild可以生成相应的软件包，下面介绍这个工具的使用，以及几个完整的制作rpm软件包的例子。更多使用这个工具的信息，参见"man rpmbuild"。

rpmbuild {-ba|-bb|-bp|-bc|-bi|-bl|-bs} [rpmbuild-options] SPECFILE ...

rpmbuild {-ta|-tb|-tp|-tc|-ti|-tl|-ts} [rpmbuild-options] TARBALL ...

rpmbuild {--rebuild|--recompile} SOURCEPKG ...

**[功能]**

    创建rpm软件包。

**[描述]**

    rpmbuild可以用来制作二进制或者源码软件包。软件包包含文件以及一些meta-data，可以用来安装或者卸载软件之用。mata-data包含帮助脚本，文件属性以及软件包的其他描述信息。软件包有两种形式：二进制软件包，用来安装软件时候使用（一般就是将已经编译好可以直接运行的程序打包而成）；源码软件包，包含源代码等可以构建二进制软件包的文件。

这个工具有三种必选的基本模式：Build Package, Build Package from Tarball, Recompile Package, Show Configuration.这里重点介绍第一种，也就是构建软件包（rpm包）的模式。

**\*通用选项：**

    以下是各种模式下都可以使用的通用选项：

    -?, --help

    打印帮助信息。

    --version

    打印单行的，被使用的rpm的版本信息。

    --quiet

    打印尽可能少的信息，通常只包含错误信息。

    -v

    打印详细信息，一般都包含处理过程。

    -vv

    打印尽可能多的信息。

    --rcfile FILELIST

    依次读取FILELIST中被":"分割的文件用于配置信息。第一个文件必须存在，默认FILELIST是:"/usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:/etc/rpmrc:~/.rpmrc".

    --pipe CMD

    将输出通过管道输送给CMD命令。

    --dbpath DIRECTORY

    使用DIRECTORY中的数据库，而不是默认的"var/lib/rpm"中的数据库。

    --root DIRECTORY

    使用DIRECTORY做为所有操作的相对的文件系统根目录。注意这也意味着chroot到DIRECTORY之后，DIRECTORY中的数据库被用来做依赖性的检查以及运行一些脚本片段(例如：安装时候的%post，或者编译时候的%prep)。

**\*BUILD模式的选项：**

    rpm的build命令一般形式为：

    rpmbuild -bSTAGE|-tSTAGE [      rpmbuild-options ] FILE ...

    如果使用spec文件进行构建，那么使用的为-b参数，如果使用tar文件中的spec文件进行构建，那么使用-t的参数。-b或者-t参数之后的STAGE参数指定了执行构建软件包中的哪个阶段(对应于spec文件中的预处理，构建，安装等各个阶段)。

    -ba

    构建二进制和源代码包（执行完spec文件中的%prep,%build,和%install阶段之后做）。

    -bb

    构建二进制包（执行完spec文件中的%prep,%build,和%install阶段之后做）。

    -bp

    执行spec文件中的"%prep"段中的内容。这个段一般都是用来解压准备生成将要编译的源代码以及应用补丁等时候用的。

    -bc

    执行spec文件中的"%build"段中的内容(先运行%prep段之后运行)。这个段一般就是相当于进入源代码中，执行"make"，生成可以安装甚至执行的二进制文件。

    -bi

    执行spec文件中的"%install"段中的内容(先运行%prep,%build段之后运行)。这个段一般就是相当于进入源代码中，执行"make install"，将生成的二进制等文件安装到一个虚拟的根目录中（就是将那个目录当做根目录来看待，那个目录中有usr,etc等根目录中相同的结构），以便后面打包时候使用，以后安装的时候，一般就按照虚拟根目录中的结构将软件安装。

    -bl

    进行文件列表检查，spec文件的"%files"段是一个宏扩展，这个检查用来确保每个文件是存在的。

    -bs    Build just the source package.

    只构建生成源码软件包。

    下面是一些build模式中的额外选项：

    --buildroot DIRECTORY

    构建软件包的时候，将spec文件中的BuildRoot路径用DIRECTORY来替代（就是指定DIRECTORY为虚拟根目录）。

    --clean

    制作好软件包之后，清除编译目录。

    如果不指定，那么构建好软件包之后，BUILD下还有解压的软件源代码以及编译中间文件。

    --nobuild

    不执行任何编译阶段的内容，用于测试spec文件。

    --rmsource

    构建之后，删除源码(可以单独使用，例如 "rpmbuild --rmsource foo.spec")。

    --rmspec

    构建之后，删除spec文件(可以单独使用，例如 "rpmbuild --rmspec foo.spec")。

    另外，命令: "rpmbuild --showrc" 可以显示rpmbuild将要使用的当前rpmrc中设置的所有选项和宏值。

**2.2，举例**

\*\*一个简单的测试

这里，对于rpmbuild命令，假设建立一个简单的spec文件myapplication.rpm.spec，然后对它的各种命令选项进行测试。

1)进行之前，查看当前路径以及文件如下：

# pwd

/root/test/rpm\_test/BUILD

# ls

Copyright  libmy.so  myapplication  myapplication.rpm.spec  myconfig1.conf  myconfig2.conf  README

2)查看myapplication.rpm.spec文件如下：

#cat myapplication.rpm.spec

#+++++++++以下是文件内容+++++++++++#

*######头部######*

*Summary:一个简单的打印hello的程序*

*Name:myapplication*

*Version:1.0*

*Release:1*

*Vendor:QuietHeart (quiet\_heart000@126.com)*

*License:Share*

*BuildRoot:%{\_tmppath}/%{name}-%{version}-%{release}-root*

*Group:Applications/Editors*

*URL:quietheart.sf.net*

*Prefix:%{\_prefix}*

*Prefix:%{\_sysconfdir}*

*Prefix:%{\_datadir}*

*Packager:QuietHeart*

*%description*

*描述省略。*

*######prep,build,install三个段都可以执行shell命令######*

*# 预处理段，为下一步的编译安装作准备。例如将源代码拷贝到BUILDS目录。*

*%prep*

*#   prep   start.*

*echo "======>In :prep section $(pwd)"*

*#echo "======>In :prep section " %{\_topdir}*

*#echo "======>In :prep section $RPM\_BUILD\_ROOT"*

*#echo "======>In :prep section " %{\_sourcedir}*

*#echo "======>In :prep section $RPM\_SOURCE\_DIR"*

*# 建立段，执行一些编译命令以便生成软件包。工作于BUILDS。*

*%build*

*#   build   start.*

*echo "======>In :build section $(pwd)"*

*#echo "======>In :build section " %{\_topdir}*

*#echo "======>In :build section $RPM\_BUILD\_ROOT"*

*#echo "======>In :build section " %{\_builddir}*

*#echo "======>In :build section $RPM\_BUILD\_DIR"*

*# 安装段，在安装软件包阶段执行的命令。*

*# 安装的内容安装到“虚拟根目录中”install是为了后面的file段服务的。*

*%install*

*#   install   start.*

*echo "======>In :install section $(pwd)"*

*#echo "======>In :install section " %{\_topdir}*

*#echo "======>In :install section $RPM\_BUILD\_ROOT"*

*######其它的一些段######*

*#文件段，定义软件包所包含的文件。具体可由后面"%defattr","%doc","%config"宏定义。*

*#主要包含：config(配置文件),doc(文档文件),bin(二进制文件）*

*%files*

*#定义文件属性：%defattr(权限，属主，属组)。其中权限'-'表默认，可用'0777'形式。*

*%defattr (-,root,root)*

*#定义制作软件包包含的文档文件，每个宏定义一个，可指定绝对路径。*

*#一般文档文件会被安装到"/usr/doc/Name-Version-Release"目录下。*

*%doc   README*

*%doc   Copyright*

*#定义软件包所需要的配置文件(还是安装的配置文件位置??)*

*#当卸载软件的时候会将没有修改过的卸掉，修改过的备份为<name>.rpmsave。*

*%config   %{\_sysconfdir}/myconfig1.conf*

*%config   %{\_sysconfdir}/myconfig2.conf*

*#软件包所包含的二进制文件，可以为目录*

*#%{\_bindir}/myapplication*

*#%{\_libdir}/libmy.so*

*#修改日志段，可以记录每次的修改日志。*

*#第一行是：\*   星期   月   日   年   修改人   电子信箱。*

*#其中：星期、月份均用英文形式，用中文会出错。*

*#接下来的行写的应是修改了什么地方，可写多行。一般以减号起始，便于查阅。*

*#修改的时候，新的必须在上面。*

*%changelog*

*\*   Fri   Sep   30   2011   QuietHeart*

*-   增加XXX内容2*

*\*   Fri   Sep   29   2011   QuietHeart*

*-   增加XXX内容1*

#+++++++++以上是文件内容+++++++++++#

3)依次测试常用的rpmbuild命令如下：

\*执行myapplication.rpm.spec文件中的prep段：

# rpmbuild -bp myapplication.rpm.spec

输入之后，输出如下：

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.23258

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /usr/src/redhat/BUILD'

======>In :prep section /usr/src/redhat/BUILD

+ exit 0

这里，prep段是编译之前的预处理段，一般都用做将一些类似源代码文件等拷贝到合适的位置，有待编译。由此可知，处理prep的时候，自动将当前路径切换到了"/usr/src/redhat/BUILD"中。

\*执行myapplication.rpm.spec文件中的build段：

#rpmbuild -bc myapplication.rpm.spec

输入之后，输出如下：

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.36144

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /usr/src/redhat/BUILD'

======>In :prep section /usr/src/redhat/BUILD

+ exit 0

Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.36144

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :build section /usr/src/redhat/BUILD'

======>In :build section /usr/src/redhat/BUILD

+ exit 0

这里，build段一般都是执行源代码编译的动作。由这里可见，执行build段之前会先执行prep段。处理build的时候，自动将当前路径切换到了"/usr/src/redhat/BUILD"中。

\*执行myapplication.rpm.spec文件中的install段：

# rpmbuild -bi myapplication.rpm.spec

输入之后，输出如下：

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.27835

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /usr/src/redhat/BUILD'

======>In :prep section /usr/src/redhat/BUILD

+ exit 0

Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.27835

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :build section /usr/src/redhat/BUILD'

======>In :build section /usr/src/redhat/BUILD

+ exit 0

Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.27835

+ umask 022

+ cd /usr/src/redhat/BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :install section /usr/src/redhat/BUILD'

======>In :install section /usr/src/redhat/BUILD

+ /usr/lib/rpm/redhat/brp-compress

+ /usr/lib/rpm/redhat/brp-strip /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump

+ /usr/lib/rpm/brp-python-bytecompile

+ /usr/lib/rpm/redhat/brp-java-repack-jars

Processing files: myapplication-1.0-1

error: File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig1.conf

error: File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig2.conf

Executing(%doc): /bin/sh -e /var/tmp/rpm-tmp.88394

+ umask 022

+ cd /usr/src/redhat/BUILD

+ DOCDIR=/var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ export DOCDIR

+ rm -rf /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ /bin/mkdir -p /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr README /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

cp: 无法 stat “README”: 没有那个文件或目录

error: Bad exit status from /var/tmp/rpm-tmp.88394 (%doc)

Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/myapplication-1.0-1-root

RPM build errors:

    File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig1.conf

    File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig2.conf

    Bad exit status from /var/tmp/rpm-tmp.88394 (%doc)

这里，由于install段没有建立相应的文件，出现了错误，但是根据错误的提示，我们还是可以知道究竟rpmbuild做了什么的。

install段主要就是为了后面指定的file段服务的，也就是说install段将需要安装的文件安装到虚拟根目录中，然后file段指定被安装到虚拟根目录中的文件。从这里看出，install没有做任何的动作，并且执行install段的时候已经执行了prep段和build段，同时当前路径为'/usr/src/redhat/BUILD'，然后就将install段执行完了。接着，将自动执行file段，根据file段指示的文件以及之前在install段拷贝好的一些文件，以及虚拟根目录myapplication.rpm.spec中的"BuildRoot:%{\_tmppath}/%{name}-%{version}-%{release}-root"指定的路径，执行一些动作。另外实践发现，这里使用"rpmbuild -ba myapplication.rpm.spec"以及"rpmbuild -bb myapplication.rpm.spec"所显示的输出一样。如果设置好了，没有出错的话，那么此处就会在虚拟根目录中安装上相应的文件（也就是files指定的文件）。如果使用的是"rpmbuild -bb myapplication.rpm.spec"那么就会继续根据虚拟根目录，制作相应的rpm包，存放在与BUILD目录同级的RPMS下。

\*自定义"%{\_topdir}"：

[root@lv-k BUILD]# pwd

/root/test/rpm\_test/BUILD

[root@lv-k BUILD]# echo %\_topdir $(pwd)/../ >~/.rpmmacros

[root@lv-k BUILD]# rpmbuild -bi myapplication.rpm.spec

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.18065

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /root/test/rpm\_test/BUILD'

======>In :prep section /root/test/rpm\_test/BUILD

+ exit 0

Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.18065

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :build section /root/test/rpm\_test/BUILD'

======>In :build section /root/test/rpm\_test/BUILD

+ exit 0

Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.18065

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :install section /root/test/rpm\_test/BUILD'

======>In :install section /root/test/rpm\_test/BUILD

+ /usr/lib/rpm/redhat/brp-compress

+ /usr/lib/rpm/redhat/brp-strip /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump

+ /usr/lib/rpm/brp-python-bytecompile

+ /usr/lib/rpm/redhat/brp-java-repack-jars

Processing files: myapplication-1.0-1

error: File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig1.conf

error: File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig2.conf

Executing(%doc): /bin/sh -e /var/tmp/rpm-tmp.18065

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ DOCDIR=/var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ export DOCDIR

+ rm -rf /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ /bin/mkdir -p /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr README /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr Copyright /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ exit 0

Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/myapplication-1.0-1-root

RPM build errors:

    File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig1.conf

    File not found: /var/tmp/myapplication-1.0-1-root/etc/myconfig2.conf

这里，其实前面例子的"/usr/src/redhat"是"%{\_topdir}"，类似的变量有很多，不列举了，具体可以查看"/usr/lib/rpm/macros"中的定义。如果修改它，就可以定义rpm默认的build路径，默认为/usr/src/redhat，所以我们这样修改：

"echo %\_topdir $(pwd)/../ >~/.rpmmacros"

当然，这假设当前路径为:"/root/test/rpm\_test/BUILD"，这样执行prep，build，install等就会在当前路径下进行而不是切换到默认的"/usr/src/redhat/BUILD"了。

\*\*

\*\*一个好用的例子

这里例子将展示把一个编译好的可执行程序打包。

1)查看制作之前的目录结构：

[root@lv-k rpm\_test]# pwd

/root/test/rpm\_test

[root@lv-k rpm\_test]# ls

BUILD  RPMS

[root@lv-k rpm\_test]# tree BUILD/ RPMS

BUILD/

|-- COPYRIGHT

|-- Copyright

|-- README

|-- libmy.so

|-- myapplication

|-- myapplication.rpm.spec

`-- myconfig.conf

RPMS

`-- i386

    `-- myapplication-1.0-1.i386.rpm

1 directory, 8 files

2)建立的rpm的spec文件如下：

[root@lv-k rpm\_test]# cd BUILD/

[root@lv-k rpm\_test]# cat myapplication.rpm.spec

#+++++++++以下是文件内容+++++++++++#

*######头部######*

*Summary:一个简单的打印hello的程序*

*Name:myapplication*

*Version:1.0*

*Release:1*

*Vendor:QuietHeart (quiet\_heart000@126.com)*

*License:Share*

*BuildRoot:%{\_tmppath}/%{name}-%{version}-%{release}-root*

*Group:Applications/Editors*

*URL:quietheart.sf.net*

*Prefix:%{\_prefix}*

*Prefix:%{\_sysconfdir}*

*Prefix:%{\_datadir}*

*Packager:QuietHeart*

*%define myconfig myconfig.conf*

*%define mylib libmy.so*

*%description*

*描述省略。*

*######prep,build,install三个段都可以执行shell命令######*

*# 预处理段，为下一步的编译安装作准备。例如将源代码拷贝到BUILDS目录。*

*%prep*

*#   prep   start.*

*echo "======>In :prep section $(pwd)"*

*echo "======>In :prep section " %{\_topdir}*

*echo "======>In :prep section $RPM\_BUILD\_ROOT"*

*echo "======>In :prep section " %{\_sourcedir}*

*#echo "======>In :prep section $RPM\_SOURCE\_DIR"*

*# 建立段，执行一些编译命令以便生成软件包。工作于BUILDS。*

*%build*

*#   build   start.*

*echo "======>In :build section $(pwd)"*

*echo "======>In :build section " %{\_topdir}*

*echo "======>In :build section $RPM\_BUILD\_ROOT"*

*echo "======>In :build section " %{\_builddir}*

*#echo "======>In :build section $RPM\_BUILD\_DIR"*

*echo "nothing" >>README*

*echo "By QuietHeart" >>COPYRIGHT*

*# 安装段，在安装软件包阶段执行的命令。*

*# 安装的内容安装到“虚拟根目录中”install是为了后面的file段服务的。*

*%install*

*#   install   start.*

*echo "======>In :install section $(pwd)"*

*echo "======>In :install section " %{\_topdir}*

*echo "======>In :install section $RPM\_BUILD\_ROOT"*

*mkdir -p $RPM\_BUILD\_ROOT/usr/bin*

*mkdir -p $RPM\_BUILD\_ROOT/usr/lib*

*mkdir -p $RPM\_BUILD\_ROOT/etc*

*#这里使用%{name}表示开始定义的Name，即myapplication。*

*cp %{name} $RPM\_BUILD\_ROOT/usr/bin*

*cp %{mylib} $RPM\_BUILD\_ROOT/usr/lib*

*cp %{myconfig} $RPM\_BUILD\_ROOT/etc*

*######其它的一些段######*

*#制作完之后，会执行这里，以便删除例如$RPM\_BUILD\_ROOT之类的临时文件。*

*%clean*

*#   clean   start.*

*echo "======>In :clean section $(pwd)"*

*echo "======>In :clean section $RPM\_BUILD\_ROOT"*

*#[ -d $RPM\_BUILD\_ROOT] && rm -rf $RPM\_BUILD\_ROOT*

*#文件段，定义软件包所包含的文件。具体可由后面"%defattr","%doc","%config"宏定义。*

*#主要包含：config(配置文件),doc(文档文件),bin(二进制文件）*

*%files*

*#定义文件属性：%defattr(权限，属主，属组)。其中权限'-'表默认，可用'0777'形式。*

*%defattr (-,root,root)*

*#定义制作软件包包含的文档文件，每个宏定义一个，可指定绝对路径。*

*#一般文档文件会被安装到"/usr/doc/Name-Version-Release"目录下。*

*%doc   README*

*%doc   Copyright*

*#定义软件包所需要的配置文件(还是安装的配置文件位置??)*

*#当卸载软件的时候会将没有修改过的卸掉，修改过的备份为<name>.rpmsave。*

*%config   %{\_sysconfdir}/%{myconfig}*

*#软件包所包含的二进制文件，可以为目录*

*%{\_bindir}/%{name}*

*%{\_libdir}/%{mylib}*

*#修改日志段，可以记录每次的修改日志。*

*#第一行是：\*   星期   月   日   年   修改人   电子信箱。*

*#其中：星期、月份均用英文形式，用中文会出错。*

*#接下来的行写的应是修改了什么地方，可写多行。一般以减号起始，便于查阅。*

*#修改的时候，新的必须在上面。*

*%changelog*

*\*   Fri   Sep   30   2011   QuietHeart*

*-   增加XXX内容2*

*\*   Fri   Sep   29   2011   QuietHeart*

*-   增加XXX内容1*

#+++++++++以上是文件内容+++++++++++#

3)制作rpm包：

[root@lv-k BUILD]# echo %\_topdir $(pwd)/../ >~/.rpmmacros

[root@lv-k BUILD]# rpmbuild -bb myapplication.rpm.spec

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.35758

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /root/test/rpm\_test/BUILD'

======>In :prep section /root/test/rpm\_test/BUILD

+ echo '======>In :prep section ' /root/test/rpm\_test/BUILD/../

======>In :prep section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :prep section /var/tmp/myapplication-1.0-1-root'

======>In :prep section /var/tmp/myapplication-1.0-1-root

+ echo '======>In :prep section ' /root/test/rpm\_test/BUILD/..//SOURCES

======>In :prep section  /root/test/rpm\_test/BUILD/..//SOURCES

+ exit 0

Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.35758

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :build section /root/test/rpm\_test/BUILD'

======>In :build section /root/test/rpm\_test/BUILD

+ echo '======>In :build section ' /root/test/rpm\_test/BUILD/../

======>In :build section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :build section /var/tmp/myapplication-1.0-1-root'

======>In :build section /var/tmp/myapplication-1.0-1-root

+ echo '======>In :build section ' /root/test/rpm\_test/BUILD/..//BUILD

======>In :build section  /root/test/rpm\_test/BUILD/..//BUILD

+ echo nothing

+ echo 'By QuietHeart'

+ exit 0

Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.35758

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :install section /root/test/rpm\_test/BUILD'

======>In :install section /root/test/rpm\_test/BUILD

+ echo '======>In :install section ' /root/test/rpm\_test/BUILD/../

======>In :install section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :install section /var/tmp/myapplication-1.0-1-root'

======>In :install section /var/tmp/myapplication-1.0-1-root

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/bin

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/lib

+ mkdir -p /var/tmp/myapplication-1.0-1-root/etc

+ cp myapplication /var/tmp/myapplication-1.0-1-root/usr/bin

+ cp libmy.so /var/tmp/myapplication-1.0-1-root/usr/lib

+ cp myconfig.conf /var/tmp/myapplication-1.0-1-root/etc

+ /usr/lib/rpm/redhat/brp-compress

+ /usr/lib/rpm/redhat/brp-strip /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump

+ /usr/lib/rpm/brp-python-bytecompile

+ /usr/lib/rpm/redhat/brp-java-repack-jars

Processing files: myapplication-1.0-1

Executing(%doc): /bin/sh -e /var/tmp/rpm-tmp.35758

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ DOCDIR=/var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ export DOCDIR

+ rm -rf /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ /bin/mkdir -p /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr README /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr Copyright /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ exit 0

Provides: config(myapplication) = 1.0-1 libmy.so

Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1

Requires: config(myapplication) = 1.0-1 libc.so.6 libc.so.6(GLIBC\_2.0) libc.so.6(GLIBC\_2.1.3) libgcc\_s.so.1 libm.so.6 libmy.so libstdc++.so.6 libstdc++.so.6(CXXABI\_1.3) libstdc++.so.6(GLIBCXX\_3.4) rtld(GNU\_HASH)

Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/myapplication-1.0-1-root

warning: Could not canonicalize hostname: lv-k.domain.com

Wrote: /root/test/rpm\_test/RPMS/i386/myapplication-1.0-1.i386.rpm

Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.13498

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

++ pwd

+ echo '======>In :clean section /root/test/rpm\_test/BUILD'

======>In :clean section /root/test/rpm\_test/BUILD

+ exit 0

4)查看制作结果：

[root@lv-k BUILD]# tree ../BUILD/ ../RPMS

../BUILD/

|-- COPYRIGHT

|-- Copyright

|-- README

|-- libmy.so

|-- myapplication

|-- myapplication.rpm.spec

`-- myconfig.conf

../RPMS

`-- i386

    `-- myapplication-1.0-1.i386.rpm

1 directory, 8 files

可见，生成的结果在../RPMS/i386之中。

另外，根据spec文件，可知道，在"/var/tmp/myapplication-1.0-1-root/"下还生成一些制作rpm包时候的临时文件，如下：

[root@lv-k BUILD]# tree /var/tmp/myapplication-1.0-1-root/

/var/tmp/myapplication-1.0-1-root/

|-- etc

|   `-- myconfig.conf

`-- usr

    |-- bin

    |   `-- myapplication

    |-- lib

    |   `-- libmy.so

    `-- share

        `-- doc

            `-- myapplication-1.0

                |-- Copyright

                `-- README

7 directories, 5 files

这些文件，可以在spec文件中的clean段里面指定删除。另外还需要注意的是，这里使用rpmbuild的-bb选项，如果使用-ba的话也会报错，如下：

[root@lv-k BUILD]# rpmbuild -ba myapplication.rpm.spec

...前面省略...

======>In :install section /var/tmp/myapplication-1.0-1-root

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/bin

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/lib

+ mkdir -p /var/tmp/myapplication-1.0-1-root/etc

+ cp myapplication /var/tmp/myapplication-1.0-1-root/usr/bin

+ cp libmy.so /var/tmp/myapplication-1.0-1-root/usr/lib

+ cp myconfig.conf /var/tmp/myapplication-1.0-1-root/etc

+ /usr/lib/rpm/redhat/brp-compress

+ /usr/lib/rpm/redhat/brp-strip /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump

+ /usr/lib/rpm/brp-python-bytecompile

+ /usr/lib/rpm/redhat/brp-java-repack-jars

Processing files: myapplication-1.0-1

Executing(%doc): /bin/sh -e /var/tmp/rpm-tmp.6317

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ DOCDIR=/var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ export DOCDIR

+ rm -rf /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ /bin/mkdir -p /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr README /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr Copyright /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ exit 0

Provides: config(myapplication) = 1.0-1 libmy.so

Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1

Requires: config(myapplication) = 1.0-1 libc.so.6 libc.so.6(GLIBC\_2.0) libc.so.6(GLIBC\_2.1.3) libgcc\_s.so.1 libm.so.6 libmy.so libstdc++.so.6 libstdc++.so.6(CXXABI\_1.3) libstdc++.so.6(GLIBCXX\_3.4) rtld(GNU\_HASH)

Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/myapplication-1.0-1-root

warning: Could not canonicalize hostname: lv-k.domain.com

error: Could not open /root/test/rpm\_test/SRPMS/myapplication-1.0-1.src.rpm: 没有那个文件或目录

5)使用制作好的rpm包

\*查看软件包头：

[root@lv-k rpm\_test]# rpm -qpi ../RPMS/i386/myapplication-1.0-1.i386.rpm

Name        : myapplication                Relocations: /usr /etc /usr/share

Version     : 1.0                               Vendor: QuietHeart (quiet\_heart000@126.com)

Release     : 1                             Build Date: 2011年10月07日 星期五 14时40分35秒

Install Date: (not installed)               Build Host: lv-k.domain.com

Group       : Applications/Editors          Source RPM: myapplication-1.0-1.src.rpm

Size        : 12504                            License: Share

Signature   : (none)

Packager    : QuietHeart

URL         : quietheart.sf.net

Summary     : 一个简单的打印hello的程序

Description :

描述省略。

\*查看软件包的全部文件列表：

[root@lv-k rpm\_test]# rpm -qpl ../RPMS/i386/myapplication-1.0-1.i386.rpm

/etc/myconfig.conf

/usr/bin/myapplication

/usr/lib/libmy.so

/usr/share/doc/myapplication-1.0

/usr/share/doc/myapplication-1.0/Copyright

/usr/share/doc/myapplication-1.0/README

这里，显示的就是安装软件将要向哪些目录安装什么文件。

\*安装软件：

[root@lv-k rpm\_test]# rpm -ivh ../RPMS/i386/myapplication-1.0-1.i386.rpm

Preparing...                ########################################### [100%]

   1:myapplication          ########################################### [100%]

\*查询被安装的软件：

[root@lv-k rpm\_test]# rpm -qa |grep myapplication

myapplication-1.0-1

\*运行软件：

[root@lv-k rpm\_test]# myapplication

begin test

hello

这里，软件源代码省略，以后会有更详细的讲解。

\*卸载软件：

[root@lv-k rpm\_test]# rpm -e myapplication-1.0-1

\*\*

\*\*一个较完整的例子

这里是一个较完整的制作rpm软件包的例子，先只给出简单过程，然后详细介绍。

**a，最简过程：**

[root@lv-k rpm\_test]# pwd

/root/test/rpm\_test

[root@lv-k rpm\_test]# tree

.

|-- BUILD

|-- RPMS

|   `-- i386

|-- SOURCES

|   `-- myapplication-1.0.tgz

|-- SPECS

|   `-- myapplication.rpm.spec

`-- SRPMS

6 directories, 2 files

[root@lv-k rpm\_test]# echo %\_topdir $(pwd) >~/.rpmmacrosmacros

[root@lv-k rpm\_test]# rpmbuild -ba SPECS/myapplication.rpm.spec

[root@lv-k rpm\_test]# tree

.

|-- BUILD

|   `-- myapplication-1.0

|       |-- COPYRIGHT

|       |-- Makefile

|       |-- README

|       |-- debugfiles.list

|       |-- debuglinks.list

|       |-- debugsources.list

|       |-- lib

|       |   |-- Makefile

|       |   |-- libmy.so

|       |   |-- myfile.cpp

|       |   |-- myfile.h

|       |   `-- myfile.o

|       |-- main.cpp

|       |-- main.o

|       |-- myapplication

|       |-- myconfig.conf

|       `-- myfile.h

|-- RPMS

|   `-- i386

|       |-- myapplication-1.0-1.i386.rpm

|       `-- myapplication-debuginfo-1.0-1.i386.rpm

|-- SOURCES

|   `-- myapplication-1.0.tgz

|-- SPECS

|   `-- myapplication.rpm.spec

`-- SRPMS

    `-- myapplication-1.0-1.src.rpm

8 directories, 21 files

**b，详细过程：**

1)查看文件结构：

[root@lv-k rpm\_test]# pwd

/root/test/rpm\_test

[root@lv-k rpm\_test]# tree

.

|-- BUILD

|-- RPMS

|   `-- i386

|-- SOURCES

|   `-- myapplication-1.0.tgz

|-- SPECS

|   `-- myapplication.rpm.spec

`-- SRPMS

6 directories, 2 files

2)源代码和rpm的spec文件内容：

2.1)源代码文件相关：

假设将SOURCES/myapplication-1.0.tgz解压(实际运行"rpm -bp"的时候就是这样做的，并且解到BUILD供后面的步骤使用)，

#tree myapplication

myapplication-1.0

|-- COPYRIGHT

|-- Makefile

|-- README

|-- lib

|   |-- Makefile

|   |-- myfile.cpp

|   `-- myfile.h

|-- main.cpp

|-- myconfig.conf

`-- myfile.h

1 directory, 9 files

每个文件内容依次如下：

[root@lv-k BUILD]# cat myapplication-1.0/COPYRIGHT

输出如下：

By QuietHeart

[root@lv-k BUILD]# cat myapplication-1.0/Makefile

输出如下：

myapplication:main.o libmy.so

        g++ main.o -L./lib/ -lmy -o myapplication

main.o:main.cpp

libmy.so:

        make -C ./lib

.PHONY:clean

clean:

        make -C ./lib clean

        rm \*.o myapplication

[root@lv-k BUILD]# cat myapplication-1.0/README

输出如下：

nothing

[root@lv-k BUILD]# cat myapplication-1.0/main.cpp

输出如下：

#include<iostream>

#include "myfile.h"

using std::cout;

using std::endl;

int main(int argc, char \*argv[])

{

        cout<<"begin test"<<endl;

        printInfo();

        return 0;

}

[root@lv-k BUILD]# cat myapplication-1.0/myconfig.conf

输出如下：

configure content

[root@lv-k BUILD]# cat myapplication-1.0/myfile.h

输出如下：

#ifndef \_\_MYFILE\_H

#define \_\_MYFILE\_H

void printInfo();

#endif

[root@lv-k BUILD]# cat myapplication-1.0/lib/Makefile

输出如下：

libmy.so:myfile.o

        g++ -shared -o libmy.so myfile.o

myfile.o:myfile.cpp

.PHONY:clean

clean:

        rm \*.so \*.o

[root@lv-k BUILD]# cat myapplication-1.0/lib/myfile.cpp

输出如下：

#include "myfile.h"

#include <iostream>

using std::cout;

using std::endl;

void printInfo()

{

        cout<<"hello"<<endl;

}

[root@lv-k BUILD]# cat myapplication-1.0/lib/myfile.h

输出如下：

#ifndef \_\_MYFILE\_H

#define \_\_MYFILE\_H

void printInfo();

#endif

2.2)spec文件相关：

[root@lv-k rpm\_test]# tree SPECS/

输出如下：

SPECS/

`-- myapplication.rpm.spec

0 directories, 1 file

查看其文件内容如下：

[root@lv-k rpm\_test]# cat SPECS/myapplication.rpm.spec

输出如下：

*######头部######*

*Summary:一个简单的打印hello的程序*

*Name:myapplication*

*Version:1.0*

*Release:1*

*Vendor:QuietHeart (quiet\_heart000@126.com)*

*License:Share*

*SOURCE:%{name}-%{version}.tgz*

*BuildRoot:%{\_tmppath}/%{name}-%{version}-%{release}-root*

*Group:Applications/Editors*

*URL:quietheart.sf.net*

*Prefix:%{\_prefix}*

*Prefix:%{\_sysconfdir}*

*Prefix:%{\_datadir}*

*Packager:QuietHeart*

*%define myconfig myconfig.conf*

*%define mylib libmy.so*

*%description*

*描述省略。*

*######prep,build,install三个段都可以执行shell命令######*

*# 预处理段，为下一步的编译安装作准备。例如将源代码拷贝到BUILDS目录。*

*%prep*

*#   prep   start.*

*echo "======>In :prep section $(pwd)"*

*echo "======>In :prep section " %{\_topdir}*

*echo "======>In :prep section $RPM\_BUILD\_ROOT"*

*echo "======>In :prep section " %{\_sourcedir}*

*#echo "======>In :prep section $RPM\_SOURCE\_DIR"*

*#将源代码包解压并放好，也可以使用tar命令，解压好的目录:%{name}-%{version}，存在BUILD中。*

*%setup -n %{name}-%{version}*

*#tar -xzvf %{\_sourcedir}/%{name}-%{version}.tgz -C %{\_builddir}*

*# 建立段，执行一些编译命令以便生成软件包。工作于BUILDS。*

*%build*

*#   build   start.*

*echo "======>In :build section $(pwd)"*

*echo "======>In :build section " %{\_topdir}*

*echo "======>In :build section $RPM\_BUILD\_ROOT"*

*echo "======>In :build section " %{\_builddir}*

*#echo "======>In :build section $RPM\_BUILD\_DIR"*

*#自动进入myapplication-1.0中了。*

*make*

*# 安装段，在安装软件包阶段执行的命令。*

*# 安装的内容安装到“虚拟根目录中”install是为了后面的file段服务的。*

*%install*

*#   install   start.*

*echo "======>In :install section $(pwd)"*

*echo "======>In :install section " %{\_topdir}*

*echo "======>In :install section $RPM\_BUILD\_ROOT"*

*mkdir -p $RPM\_BUILD\_ROOT/usr/bin*

*mkdir -p $RPM\_BUILD\_ROOT/usr/lib*

*mkdir -p $RPM\_BUILD\_ROOT/etc*

*#这里使用%{name}表示开始定义的Name，即myapplication。*

*cp %{name} $RPM\_BUILD\_ROOT/usr/bin*

*cp lib/%{mylib} $RPM\_BUILD\_ROOT/usr/lib*

*cp %{myconfig} $RPM\_BUILD\_ROOT/etc*

*######其它的一些段######*

*%clean*

*#   clean   start.*

*echo "======>In :clean section $(pwd)"*

*#文件段，定义软件包所包含的文件。具体可由后面"%defattr","%doc","%config"宏定义。*

*#主要包含：config(配置文件),doc(文档文件),bin(二进制文件）*

*%files*

*#定义文件属性：%defattr(权限，属主，属组)。其中权限'-'表默认，可用'0777'形式。*

*%defattr (-,root,root)*

*#定义制作软件包包含的文档文件，每个宏定义一个，可指定绝对路径。*

*#一般文档文件会被安装到"/usr/doc/Name-Version-Release"目录下。*

*%doc   README*

*%doc   COPYRIGHT*

*#定义软件包所需要的配置文件(还是安装的配置文件位置??)*

*#当卸载软件的时候会将没有修改过的卸掉，修改过的备份为<name>.rpmsave。*

*%config   %{\_sysconfdir}/%{myconfig}*

*#软件包所包含的二进制文件，可以为目录*

*%{\_bindir}/%{name}*

*%{\_libdir}/%{mylib}*

*#修改日志段，可以记录每次的修改日志。*

*#第一行是：\*   星期   月   日   年   修改人   电子信箱。*

*#其中：星期、月份均用英文形式，用中文会出错。*

*#接下来的行写的应是修改了什么地方，可写多行。一般以减号起始，便于查阅。*

*#修改的时候，新的必须在上面。*

*%changelog*

*\*   Fri   Sep   30   2011   QuietHeart*

*-   增加XXX内容2*

*\*   Fri   Sep   29   2011   QuietHeart*

*-   增加XXX内容1*

3)制作rpm包：

[root@lv-k rpm\_test]# echo %\_topdir $(pwd) >~/.rpmmacrosmacros

[root@lv-k rpm\_test]# rpmbuild -ba SPECS/myapplication.rpm.spec

输入之后，输出如下具体制做过程信息：

Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.87076

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :prep section /root/test/rpm\_test/BUILD'

======>In :prep section /root/test/rpm\_test/BUILD

+ echo '======>In :prep section ' /root/test/rpm\_test/BUILD/../

======>In :prep section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :prep section /var/tmp/myapplication-1.0-1-root'

======>In :prep section /var/tmp/myapplication-1.0-1-root

+ echo '======>In :prep section ' /root/test/rpm\_test/BUILD/..//SOURCES

======>In :prep section  /root/test/rpm\_test/BUILD/..//SOURCES

+ cd /root/test/rpm\_test/BUILD

+ rm -rf myapplication-1.0

+ /bin/gzip -dc /root/test/rpm\_test/SOURCES/myapplication-1.0.tgz

+ tar -xvvf -

drwxrwxrwx root/root         0 2011-10-07 15:29:19 myapplication-1.0/

-rw-r--r-- root/root        14 2011-10-07 15:28:56 myapplication-1.0/COPYRIGHT

drwxrwxrwx root/root         0 2011-10-07 15:16:34 myapplication-1.0/lib/

-rwxrwxrwx root/root       117 2009-11-21 17:12:18 myapplication-1.0/lib/myfile.cpp

-rwxr-xr-x root/root        63 2011-09-29 18:17:29 myapplication-1.0/lib/myfile.h

-rw-r--r-- root/root       106 2011-10-07 15:16:29 myapplication-1.0/lib/Makefile

-rwxrwxrwx root/root        63 2009-11-21 17:12:18 myapplication-1.0/myfile.h

-rw-r--r-- root/root         8 2011-10-07 15:28:51 myapplication-1.0/README

-rw-r--r-- root/root        18 2011-10-07 15:29:19 myapplication-1.0/myconfig.conf

-rwxrwxrwx root/root      1734 2009-12-10 09:09:06 myapplication-1.0/main.cpp

-rw-r--r-- root/root       178 2011-10-07 15:16:07 myapplication-1.0/Makefile

+ STATUS=0

+ '[' 0 -ne 0 ']'

+ cd myapplication-1.0

++ /usr/bin/id -u

+ '[' 0 = 0 ']'

+ /bin/chown -Rhf root .

++ /usr/bin/id -u

+ '[' 0 = 0 ']'

+ /bin/chgrp -Rhf root .

+ /bin/chmod -Rf a+rX,u+w,g-w,o-w .

+ exit 0

Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.87076

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ cd myapplication-1.0

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :build section /root/test/rpm\_test/BUILD/myapplication-1.0'

======>In :build section /root/test/rpm\_test/BUILD/myapplication-1.0

+ echo '======>In :build section ' /root/test/rpm\_test/BUILD/../

======>In :build section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :build section /var/tmp/myapplication-1.0-1-root'

======>In :build section /var/tmp/myapplication-1.0-1-root

+ echo '======>In :build section ' /root/test/rpm\_test/BUILD/..//BUILD

======>In :build section  /root/test/rpm\_test/BUILD/..//BUILD

+ make

g++    -c -o main.o main.cpp

make -C ./lib

make[1]: Entering directory `/root/test/rpm\_test/BUILD/myapplication-1.0/lib'

g++    -c -o myfile.o myfile.cpp

g++ -shared -o libmy.so myfile.o

make[1]: Leaving directory `/root/test/rpm\_test/BUILD/myapplication-1.0/lib'

g++ main.o -L./lib/ -lmy -o myapplication

+ exit 0

Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.83304

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ cd myapplication-1.0

+ LANG=C

+ export LANG

+ unset DISPLAY

++ pwd

+ echo '======>In :install section /root/test/rpm\_test/BUILD/myapplication-1.0'

======>In :install section /root/test/rpm\_test/BUILD/myapplication-1.0

+ echo '======>In :install section ' /root/test/rpm\_test/BUILD/../

======>In :install section  /root/test/rpm\_test/BUILD/../

+ echo '======>In :install section /var/tmp/myapplication-1.0-1-root'

======>In :install section /var/tmp/myapplication-1.0-1-root

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/bin

+ mkdir -p /var/tmp/myapplication-1.0-1-root/usr/lib

+ mkdir -p /var/tmp/myapplication-1.0-1-root/etc

+ cp myapplication /var/tmp/myapplication-1.0-1-root/usr/bin

+ cp lib/libmy.so /var/tmp/myapplication-1.0-1-root/usr/lib

+ cp myconfig.conf /var/tmp/myapplication-1.0-1-root/etc

+ /usr/lib/rpm/find-debuginfo.sh /root/test/rpm\_test/BUILD/..//BUILD/myapplication-1.0

+ /usr/lib/rpm/redhat/brp-compress

+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip

+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump

+ /usr/lib/rpm/brp-python-bytecompile

+ /usr/lib/rpm/redhat/brp-java-repack-jars

Processing files: myapplication-1.0-1

Executing(%doc): /bin/sh -e /var/tmp/rpm-tmp.83304

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ cd myapplication-1.0

+ DOCDIR=/var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ export DOCDIR

+ rm -rf /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ /bin/mkdir -p /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr README /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ cp -pr COPYRIGHT /var/tmp/myapplication-1.0-1-root/usr/share/doc/myapplication-1.0

+ exit 0

Provides: config(myapplication) = 1.0-1 libmy.so

Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1

Requires: config(myapplication) = 1.0-1 libc.so.6 libc.so.6(GLIBC\_2.0) libc.so.6(GLIBC\_2.1.3) libgcc\_s.so.1 libm.so.6 libmy.so libstdc++.so.6 libstdc++.so.6(CXXABI\_1.3) libstdc++.so.6(GLIBCXX\_3.4) rtld(GNU\_HASH)

Processing files: myapplication-debuginfo-1.0-1

Provides: libmy.so.debug

Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1

Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/myapplication-1.0-1-root

warning: Could not canonicalize hostname: lv-k.domain.com

Wrote: /root/test/rpm\_test/SRPMS/myapplication-1.0-1.src.rpm

Wrote: /root/test/rpm\_test/RPMS/i386/myapplication-1.0-1.i386.rpm

Wrote: /root/test/rpm\_test/RPMS/i386/myapplication-debuginfo-1.0-1.i386.rpm

Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.76622

+ umask 022

+ cd /root/test/rpm\_test/BUILD/..//BUILD

+ cd myapplication-1.0

++ pwd

+ echo '======>In :clean section /root/test/rpm\_test/BUILD/myapplication-1.0'

======>In :clean section /root/test/rpm\_test/BUILD/myapplication-1.0

+ exit 0

4)查看生成的文件：

[root@lv-k rpm\_test]# tree

.

|-- BUILD

|   `-- myapplication-1.0

|       |-- COPYRIGHT

|       |-- Makefile

|       |-- README

|       |-- debugfiles.list

|       |-- debuglinks.list

|       |-- debugsources.list

|       |-- lib

|       |   |-- Makefile

|       |   |-- libmy.so

|       |   |-- myfile.cpp

|       |   |-- myfile.h

|       |   `-- myfile.o

|       |-- main.cpp

|       |-- main.o

|       |-- myapplication

|       |-- myconfig.conf

|       `-- myfile.h

|-- RPMS

|   `-- i386

|       |-- myapplication-1.0-1.i386.rpm

|       `-- myapplication-debuginfo-1.0-1.i386.rpm

|-- SOURCES

|   `-- myapplication-1.0.tgz

|-- SPECS

|   `-- myapplication.rpm.spec

`-- SRPMS

    `-- myapplication-1.0-1.src.rpm

8 directories, 21 files

由上面可见，生成的文件分别是：

BUILD/myapplication-1.0目录，在prep段指定的命令中解压SOURCES目录中的内容生成。

RPMS/i386/\*.rpm，制作后期二进制rpm包生成。（可以用"rpmbuild -bb"单独生成这个）。

SRPMS/\*.rpm，制作后期源码rpm包生成。（可以用"rpmbuild -bs"单独生成这个）。

5)使用制作的rpm包：

\*查看软件包头：

[root@lv-k rpm\_test]# rpm -qpi RPMS/i386/myapplication-1.0-1.i386.rpm

Name        : myapplication                Relocations: /usr /etc /usr/share

Version     : 1.0                               Vendor: QuietHeart (quiet\_heart000@126.com)

Release     : 1                             Build Date: 2011年10月07日 星期五 16时13分09秒

Install Date: (not installed)               Build Host: lv-k.domain.com

Group       : Applications/Editors          Source RPM: myapplication-1.0-1.src.rpm

Size        : 12580                            License: Share

Signature   : (none)

Packager    : QuietHeart

URL         : quietheart.sf.net

Summary     : 一个简单的打印hello的程序

Description :

描述省略。

\*查看软件包的全部文件列表：

[root@lv-k rpm\_test]# rpm -qpl RPMS/i386/myapplication-1.0-1.i386.rpm

/etc/myconfig.conf

/usr/bin/myapplication

/usr/lib/libmy.so

/usr/share/doc/myapplication-1.0

/usr/share/doc/myapplication-1.0/COPYRIGHT

/usr/share/doc/myapplication-1.0/README

这里，显示的就是安装软件将要向哪些目录安装什么文件。

\*安装软件：

[root@lv-k rpm\_test]# rpm -ivh RPMS/i386/myapplication-1.0-1.i386.rpm

Preparing...                ########################################### [100%]

   1:myapplication          ########################################### [100%]

\*指定前缀的安装软件：

[root@lv-k i386]# rpm -ivh --relocate=/usr=/root/myprefix/usr --relocate=/etc=/root/myprefix/etc myapplication-1.0-1.i386.rpm

Preparing...                ########################################### [100%]

   1:myapplication          ########################################### [100%]

这里，因为spec中指定了多个Prefix，所以使用这个方式进行指定前缀，如果使用"--prefix"，那么会如下：

[root@lv-k i386]# rpm -ivh --prefix=/root/myprefix myapplication-1.0-1.i386.rpm

error: package myapplication is not relocatable

使用--relocate指定之后，查看安装好的路径，如下：

[root@lv-k i386]# tree /root/myprefix/

/root/myprefix/

|-- etc

|   `-- myconfig.conf

`-- usr

    |-- bin

    |   `-- myapplication

    |-- lib

    |   `-- libmy.so

    `-- share

        `-- doc

            `-- myapplication-1.0

                |-- COPYRIGHT

                `-- README

7 directories, 5 files

实践发现，如果不存在/root/myprefix会自动创建，如果使用"rpm -e"卸载这样安装的软件，那么只是删除了/root/myprefix，但是目录却保留，如下：

[root@lv-k i386]# tree /root/myprefix/

/root/myprefix/

|-- etc

`-- usr

    |-- bin

    |-- lib

    `-- share

        `-- doc

6 directories, 0 files

另外，如果卸之前修改了/etc文件，那么会提示，如下：

[root@lv-k i386]# rpm -e myapplication-1.0-1

warning: /root/myprefix/etc/myconfig.conf saved as /root/myprefix/etc/myconfig.conf.rpmsave

[root@lv-k i386]# tree /root/myprefix/

/root/myprefix/

|-- etc

|   `-- myconfig.conf.rpmsave

`-- usr

    |-- bin

    |-- lib

    `-- share

        `-- doc

6 directories, 1 file

\*查询被安装的软件：

[root@lv-k rpm\_test]# rpm -qa |grep myapplication

myapplication-1.0-1

\*运行软件：

[root@lv-k rpm\_test]# myapplication

begin test

hello

这里，软件源代码省略，以后会有更详细的讲解。

\*卸载软件：

[root@lv-k rpm\_test]# rpm -e myapplication-1.0-1

\*查看源码包头：

[root@lv-k rpm\_test]# rpm -qpi SRPMS/myapplication-1.0-1.src.rpm

Name        : myapplication                Relocations: (not relocatable)

Version     : 1.0                               Vendor: QuietHeart (quiet\_heart000@126.com)

Release     : 1                             Build Date: 2011年10月07日 星期五 16时13分09秒

Install Date: (not installed)               Build Host: lv-k.domain.com

Group       : Applications/Editors          Source RPM: (none)

Size        : 5084                             License: Share

Signature   : (none)

Packager    : QuietHeart

URL         : quietheart.sf.net

Summary     : 一个简单的打印hello的程序

Description :

描述省略。

\*查看源码包文件列表：

[root@lv-k rpm\_test]# rpm -qpi SRPMS/myapplication-1.0-1.src.rpm

myapplication-1.0.tgz

myapplication.rpm.spec

\*安装源码包:

[root@lv-k rpm\_test]# rpm -ivh SRPMS/myapplication-1.0-1.src.rpm

   1:myapplication          ########################################### [100%]

\*查看刚刚的安装：

[root@lv-k rpm\_test]# rpm -qa |grep myapplication

这里，看不出来，究竟安装到了哪里。但是实际上，是安装到了SOURCES中和SPECS中了，如果将SOURCES和SPECS中原来的删掉，再安装，就能够看出来了。

\*\*

**二、一些软件包管理的命令**

**==========================**

这里介绍管理软件包的rpm命令。介绍的仅为一些常用的功能，具体参见帮助手册"man rpm"。

**[举例]**

\*查询所有本机软件包安装信息：

#rpm -qa

这里，可知，如果查询已安装软件包的版本，命令类似"rpm -qa|grep uxengine"。

\*安装软件包，并显示进度和详细信息：

#rpm -ivh <package>.rpm

这里，-i表示安装，-h表示显示进度条，-v表示显示详细信息。这样会将package安装。

\*删除软件包：

#rpm -e <package>

这里，<package>是软件包名，与"rpm -qa"查询输出的相对应。

\*升级软件包：

#rpm -U <newpackage>

这样会将比已经安装的更新版本的newpackage安装。其他的版本会在安装之后被删除。

\*强制替换安装：

#rpm -i --force <package>.rpm

这个是从man手册上面查找到的。

\*查询软件包的头部相关信息：

# rpm -qpi fvwm-2.5.27-1.i386.rpm

输入之后，输出类似如下：

Name        : fvwm                         Relocations: (not relocatable)

Version     : 2.5.27                            Vendor: (none)

Release     : 1                             Build Date: 2009年03月11日 星期三 00时25分47秒

Install Date: (not installed)               Build Host: www.mevashlim.com

Group       : User Interface/Desktops       Source RPM: fvwm-2.5.27-1.src.rpm

Size        : 8237434                          License: GPL

Signature   : (none)

Packager    : Fvwm Workers <fvwm-workers@fvwm.org>

URL         : http://www.fvwm.org/

Summary     : F(?) Virtual Window Manager

Description :

Fvwm is a powerful ICCCM-compliant multiple virtual desktop window manager

for the X Window System.

This 2.5 version includes new features like full support of the EWMH

(Enhanced Window Manager Hints) specification, internationalization,

improved window decoration code (no flickering anymore), bi-directional

asian text support, FreeType font support (antialiasing), image rendering,

Perl based module library, support for PNG images, side titles and much more.

\*查询软件包的文件列表：

# rpm -qpl myapplication-1.0-1.i386.rpm

/etc/myconfig.conf

/usr/bin/myapplication

/usr/lib/libmy.so

/usr/share/doc/myapplication-1.0

/usr/share/doc/myapplication-1.0/Copyright

/usr/share/doc/myapplication-1.0/README

这里，显示的是安装软件将会在哪些目录安装什么文件。

**[其它]**

这里的内容，有待实践。

\*安装的时候想修改默认路径，则可以：

rpm -ivh --prefix=/opt/usr xxx.rpm

\*修改多个路径：

rpm xxx.rpm --relocate=/usr=/opt/usr --relocate=/etc=/usr/etc

可以通过命令rpm --showrc查看实现代码。另外直接通过 rpm --eval "%{macro}"来查看具体对应路径。

比如我们要查看%{\_bindir}的路径，就可以使用命令rpm --eval "%{ \_bindir}"来查看。

另外，所有的宏都可以在/usr/lib/rpm/macros里找到。

**三、补充**

**==========================**

**参考资料:**

http://bbs.chinaunix.net/thread-449396-1-1.html

http://fedoraproject.org/wiki/Packaging/RPMMacros#RPM\_directory\_macros

http://www-900.ibm.com/developerWorks/cn/linux/management/package/rpm/part1/index.shtml[/url]

http://www-900.ibm.com/developerWorks/cn/linux/management/package/rpm/part2/index.shtml

http://www-900.ibm.com/developerWorks/cn/linux/management/package/rpm/part3/index.shtml

http://www.rpm.org/RPM-HOWTO/build.html#SCRIPTS

http://www.linuxfans.org/nuke/modules.php?name=Forums&file=printview&t=86980&start=0

http://oa.jmu.edu.cn/netoa/libq/pubdisc.nsf/66175841be38919248256e35005f4497/ba6fbec275186d5d4825709100163428?OpenDocument

终于写完，好多!^\_^。

本想将其中涉及内容制做成附件，可是这样又无法在所有文档中下载，所以直接将涉及的文件内容都集成到文档中了，如果照着做的话，所需要的文件都有。另外，如果链接还好使的话，我在sourceforge上面上传了这个完整的例子，可以参考。网址为："https://sourceforge.net/projects/quietheart/files/temp/test/"其中的rpm\_test.tar.gz

以上内容，为参考相应参考资料，并且实践后的总结，有些地方可能会有不准处，如有更好建议可以通过如下方式联系我，谢谢。

**作者：QuietHeart**

**Email:quiet\_heart000@126.com**

**日期：2011年10月8日**