利用安装光盘创建本地 yum 源补装 RPM 软件包

戴月, 软件工程师

发布日期： 2010 年 3 月 26 日
级别： 初级
访问情况 1548 次浏览
建议：

简介： 为系统补装 RPM 软件包，对一部分 Linux 用户来讲，是一件繁琐的事。众所周知的 RPM 依赖问题困扰了众多用户，让他们在面临这个场合时懊悔当初没有选择完全系统安装。甚至盛传着“完全安装吧，不差这几 G 空间，省得麻烦”的流行说法。虽然 yum 已经很大程度上帮助我们解决了这个问题，但在某些特殊环境，如网络连接，带宽的限制，或者时间紧迫的场合下，我们希望有一种方法，像系统初装一样，可以简单的选择，并迅速补装上这些遗漏的软件包。本文以两个用户在 RHEL 5u4 和 Fedora12 上的真实操作开始，介绍了利用安装光盘创建本地 yum 源的方式以快速补装 RPM 软件包的过程。

引言

如果你是一个 Linux 新手，刚安装完一个崭新的 RHEL 5u4 系统，在默认配置下使用了几天，感觉不错。这时你想用 gcc 编一个程序，发现默认配置里并没有安装这个包。你找到安装光盘，小心翼翼地挂载上，并找到了 gcc-4.1.2-46.el5.i386.rpm 。尝试着用 rpm -ivh gcc-4.1.2-46.el5.i386.rpm 进行安装。系统告之缺少 glibc-devel 和 libgomp，当你试着安装 glibc-devel，系统又提示缺少 glibc-headers。继续下去，又发现缺少 kernel-headers。先前网上关于 RPM 依赖关系如何复杂的传言终于得到了印证。你很苦恼，心想如果当初选择了完全安装该有多好！同时也懊悔为了结省那不值钱的几 GB 空间而给自己今天带来了如此多的麻烦。最终，你按照依赖提示，递归似的完成了 gcc 的安装。你很庆幸的完成了这个任务。其实只是因为这个安装任务的依赖关系还不够复杂到把你吓退罢了。

如果你是一位忠实的 Fefora Linux 用户。有一天下午，你想为一台刚装完的 Fedora12 安装上 GNOME 桌面。依赖性并不是你所担心的，因为你知道 yum 可以解决这个问题。但是在键入 yum install 过后，发现自己并不知道要安装上 GNOME 桌面系统需要哪些软件包，甚至从来没有听说过有一个叫 GNOME 的包可以安装。你试着查询了一下仓库中包含 GNOME 字样的包，一共有 258 个。这个列表显然不是你想要的。在咨询过 google 过后，发现 yum 有一个 grouplist 选项，看起来有点像自己想要的。通过 yum grouplist 又找到了名叫 "X Window System" 和 "GNOME Desktop Environment" 的分组，这正是你想要的。但是在键入 yum groupinstall "X Window System" 过后，提示信息说 299 个包需要下载安装，一共 60M。看着下载进度条上的速度，只有 27KB/s，这样下完这 60M 需要将近一个小时。接下来还要安装一个不知道多大的 "GNOME Desktop Environment"。这意味着老板交待的事可能要明天才能开始，而且仅仅是开始，万一下载安装没有顺利完成 ...... 你一边抱怨着公司的网络为什么久久不扩容，一边也懊恼为什么当初没有进行完全系统安装。

但是，“完全安装”真的是最好的方法吗？面对这种已经没有“完全安装”的情况，我们有什么更好的办法吗？

回页首

关于 RPM 包依赖的思考

RPM 的依赖性是个很让人头疼的问题。每次系统安装完成过后，想再装一个软件包，敲下回车之前，都会心惊胆战，生怕跳出个依赖性错误。而在一个依赖性错误的背后，可能还牵连着更多的依赖性。这个问题让很多 Linux 用户望而生畏。

RPM 包的依赖关系说起来是简单的因为它所谓的依赖关系仅仅是一个个简单的如“要安装 A，需要提前准备好 a1, a2, a3”这样的条件。我们在安装前可以通过一个简单的 rpm 命令来查询一个包的依赖关系。

```
# rpm -q --requires -p wireshark-1.2.2-1.fc12.i686.rpm
/sbin/ldconfig
config(wireshark) = 1.2.2-1.fc12
libc.so.6
...
libcom_err.so.2
...
libpcap.so.1
...
libsmi.so.2
...
python(abi) = 2.6
...
```

RPM 的依赖性同时又是繁琐的，因为它的依赖关系里只包含了 A 需要 a1, a2 和 a3 的信息，并没有包含 a1, a2 和 a3 是由哪个包来提供。对于如 python 这样直接给出依赖包名，我们可以轻松地找到 python-xxx.rpm；对于如 libpcap.so.1 这样熟悉的库，我们知道它对应 libpcap-xxx.rpm；对于一部分不熟悉的，如 libsmi.so.2 这样的依赖库，我们也可以找到恰好与该库同名的 libsmi-xxx.rpm。但是对于 libglib-2.0.so.0 这样的库，你会很容易“猜”出它是来自于 glib2 软件包的吗？正是因为这样的喋喋不休，以及对应依赖包的含糊性，使得用户对 RPM 的依赖性产生了一定的畏惧心理。

那么怎么确定一个库由哪个包提供呢？对于已安装的包，可以用通过下面的方式来查询：

```
# rpm -q --whatprovides libpcap.so.1
libpcap-1.0.0-4.20090922gite154e2.fc12.i686
```

但是对于在安装过程中存在依赖的库，显示这种方法是不可行的。在一台独立的系统里，这个答案甚至是无解的。某个库包含在哪里，只能遍历安装包，在每个包查询“我提供了哪些东西”的才能找到。

下面是通过安装盘中找 libglib-2.0.so.0 由哪个包提供的例子，用时 21 秒，着实不短。如果用这种方法查找 libz.so.1，需要数分钟才能完成。

mount /dev/hdc /mnt/cdrom

```
mount: block device /dev/hdc is write-protected, mounting read-only
# cd /mnt/cdrom/Server
# time for i in `ls -1`;
do
 if rpm -q --provides -p $i | grep -q libglib-2.0.so.0
 then
   echo $i
   break
 fi
done
glib2-2.22.2-2.fc12.i686.rpm

real    0m21.804s
user    0m3.946s
sys     0m14.396s
```

yum 对 RPM 包的处理是一个典型的“空间换时间”过程。通过对源中所有 RPM 包的预处理，生成所有安装包关于包含文件，依赖，冲突等信息的索引，并且以 sqlite 格式存放在 /var/cache/yum 对应目录下面。

基于 yum 索引查找 libglib-2.0.so.0 由哪个包提供的例子如下：

```
# time echo "select packages.name from packages inner join provides
#on packages.pkgKey = provides.pkgKey where provides.name = 'libglib-2.0.so.0';" |
#sqlite3 /var/cache/yum/i386/12/dvd/primary.sqlite  glib2

real    0m0.008s
user    0m0.003s
sys     0m0.005s
```

同样的查询，用索引数据来处理，只需要约 1/100 秒。数十兆的空间，换来的是成百上千倍时间效率的提升。

多年来，RPM 关于依赖性的原理并没有大的变化，只是经由这么一包装，让大家觉得依赖性逐渐不再是一个大的问题了。

回页首

用安装 DVD 建立本地 yum 源

既然 yum 已经很好的解决了 RPM 包的依赖问题，本文要解决的就是速度问题。本地光驱的访问速度甚至快过 100M 局域网，而且在最近一些发行版安装盘中，已经预置好 repo 信息，并存放在名为 repodata 的目录中。可以说安装盘本身就是一个现成的 yum 源。在 RHEL5 中，预置了多个 repo，分别存放在 <CDROOT>/Server/repodata，<CDROOT>/Cluster/repodata 等目录中；在 Fedora12 中，存放在 <CDROOT>/repodata 中。在 RHEL4 等一些旧发行版中，没有预置 repo 信息。在这种情况下，需要手工建立 repo 信息供 yum 使用。关于手工建立 repo 的信息，请参考下一节“利用安装 CD 建立本地 yum 源中”关于 createrepo 部分的示例。

首先，把光盘放入光驱，挂载光盘。

```
# mount /dev/hdc /media/cdrom
mount: block device /dev/hdc is write-protected, mounting read-only
```

如果在本地有安装盘镜像，也通过 loop 方式挂载到相应目录。

```
# mount -o loop rhel-server-5.4-i386-dvd.iso /media/cdrom
```

然后，编辑 yum 源配置文件，添加一个本地 yum 源。这里是一张 RHEL5u4 的 DVD 安装盘，它的 repo 对应 <CDROOT>/Server。

```
# cat > /etc/yum.repos.d/dvd.repo <<END
[dvd]
name=install dvd
baseurl=file:///media/cdrom/Server
enabled=1
gpgcheck=0
END
```

对于 Fedora，系统安装后有默认远程 yum 源存在，建议在补装 RPM 包之前通过在 *.repo 文件中设置 "enabled=0" 临时禁用这些远程 yum 源。这样会迫使 yum 只使用本地光盘作为安装源，从来避免因外部下载带来时间的浪费。关于这部分的操作，请参考 doris_install_gnome_on_fc12.txt 中的相关部分。

回页首

用安装 CD 建立本地 yum 源

对于 RHEL4 或者其它相对旧的发行版，或者仅有安装 CD 的情况下，还可以通过类似的方式补装 RPM 包吗？答案是肯定的，不过需要一个多余的步骤，即手工创建 repo 数据。

下面的示例，通过把 RHEL4u6 的 5CD 中的 RPMS 混合到一个目录中，然后用 createrepo 命令建立 repo 数据。

首先混装 CD：

```
[root@localhost root]# mkdir -p /mnt/dvd
# insert CD1
[root@localhost root]# mount /dev/hdc /media/cdrom/
[root@localhost root]# cp -prf /media/cdrom/Server /mnt/dvd/RPMS
[root@localhost root]# umount /media/cdrom

# insert CD2
...
# insert CD3
...
# insert CD4
...

# insert CD5
[root@localhost root]# mount /dev/hdc /media/cdrom/
[root@localhost root]# cp -prf /media/cdrom/Server /mnt/dvd/RPMS
[root@localhost root]# umount /media/cdrom

# remove unused files
[root@localhost root]# find /mnt/dvd/RPMS -name TRANS.TBL -exec rm -f {} \;
```

然后通过 createrepo 命令创建 yum 源数据。这正是“空间换时间”的预处理过程，视 RPM 数据的多少，大约需要几分钟。"-d" 参数表示在生成 xml 索引的同时生成 sqlite 格式的索引。完成过后，这两种索引文件均以压缩格式存放于 repodata 目录中，供 yum 查询所用。

```
[root@localhost root]# cd /mnt/dvd

[root@localhost dvd]# createrepo -d .
  1/2399 - RPMS/plymouth-theme-charge-0.8.0-0.2009.29.09.18.fc12.i686.rpm
  2/2399 - RPMS/aspell-0.60.6-7.fc12.i686.rpm
  3/2399 - RPMS/xml-common-0.6.3-30.fc12.noarch.rpm

......

2397/2399 - RPMS/mythes-en-3.0-5.fc12.noarch.rpm
2398/2399 - RPMS/hunspell-mi-0.20080630-3.fc12.noarch.rpm
2399/2399 - RPMS/hicolor-icon-theme-0.11-1.fc12.noarch.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Generating sqlite DBs
Sqlite DBs complete

[root@localhost dvd]# ls -l repodata/
total 10004
-rw-r--r--. 1 root root 2974902 2009-12-06 09:48 filelists.sqlite.bz2
-rw-r--r--. 1 root root 2618445 2009-12-06 09:48 filelists.xml.gz
-rw-r--r--. 1 root root  912087 2009-12-06 09:48 other.sqlite.bz2
-rw-r--r--. 1 root root  951651 2009-12-06 09:48 other.xml.gz
-rw-r--r--. 1 root root 1809806 2009-12-06 09:48 primary.sqlite.bz2
-rw-r--r--. 1 root root  958302 2009-12-06 09:48 primary.xml.gz
-rw-r--r--. 1 root root    2726 2009-12-06 09:48 repomd.xml
```

最后，编辑 yum 源配置文件，添加一个本地 yum 源。注意对应的目录，这次是 /mnt/dvd，因为 repodata 是在此路径下创建的。

```
[root@localhost dvd]# cat > /etc/yum.repos.d/dvd.repo <<END
[dvd]
name=install dvd
baseurl=file:///mnt/dvd
enabled=1
gpgcheck=0
END
```

对于 RHEL4 一类不带 repodata 的安装 DVD，去掉混装 RPM 的一步，直接用 createrepo 命令建立源数据即可。

回页首

补装单个软件包

完成本地源配置过后，接下来就可以用 yum 进行 RPM 包的补装了。首先，查看刚刚配置好的 yum 源。

```
# yum list
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
dvd                                                     | 1.3 kB     00:00
dvd/primary                                             | 732 kB     00:00
dvd: [##############################################################] 2292/2292/

Installed Packages
Deployment_Guide-en-US.noarch              5.2-11                       installed
GConf2.i386                                2.14.0-9.el5                 installed
ImageMagick.i386                           6.2.8.0-4.el5_1.1            installed
......

Available Packages
Deployment_Guide-as-IN.noarch              5.2-11                       dvd
Deployment_Guide-bn-IN.noarch              5.2-11                       dvd
Deployment_Guide-de-DE.noarch              5.2-11                       dvd
......
zlib-devel.i386                            1.2.3-3                      dvd
zsh.i386                                   4.2.6-3.el5                  dvd
zsh-html.i386                              4.2.6-3.el5                  dvd
```

然后，通过熟知的 yum install 来补装软件包。从输出信息可以看到，yum 会帮我们处理好 RPM 包的依赖关系。同时由于包都在本地，下载的时间几乎可以忽略。很快 gcc 将安装到系统中。

```
# yum install gcc
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package gcc.i386 0:4.1.2-46.el5 set to be updated
--> Processing Dependency: libgomp >= 4.1.2-46.el5 for package: gcc
--> Processing Dependency: glibc-devel >= 2.2.90-12 for package: gcc
...
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package                Arch         Version                  Repository   Size
================================================================================
Installing:
gcc                    i386         4.1.2-46.el5             dvd         5.2 M
Installing for dependencies:
glibc-devel            i386         2.5-42                   dvd         2.0 M
...

Transaction Summary
================================================================================
Install      5 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 8.8 M
Is this ok [y/N]: y
Downloading Packages:
--------------------------------------------------------------------------------
Total                                            1.2 GB/s | 8.8 MB     00:00
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
 Installing     : kernel-headers                                          1/5
 ...
 Installing     : gcc                                                     5/5

Installed:
 gcc.i386 0:4.1.2-46.el5

Dependency Installed:
 glibc-devel.i386 0:2.5-42                   glibc-headers.i386 0:2.5-42
 kernel-headers.i386 0:2.6.18-164.el5        libgomp.i386 0:4.4.0-6.el5

Complete!
```

回页首

补装分组软件包

分组安装在需要补装如桌面系统，开发工具的时候，显得犹为方便快捷。

首先，查看 yum 源中的分组列表。

```
[root@localhost root]# yum grouplist
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process
dvd/group
   | 1.0 MB     00:00
Installed Groups:
  Administration Tools
  Editors
  GNOME Desktop Environment
  ...

Available Groups:
  Authoring and Publishing
  DNS Name Server
  Development Libraries
  ...

Done
```

然后，通过 yum groupinstall 来补装软件包。yum 会自动解析出分组所包含的 RPM 软件包，并处理好它们的依赖关系。同样，mysql-server 很快被安装到系统中。

```
[root@localhost root] # yum groupinstall "MySQL Database"
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process
Resolving Dependencies
...
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package                   Arch       Version                  Repository Size
================================================================================
Installing:
MySQL-python              i386       1.2.1-1                  dvd        82 k
libdbi-dbd-mysql          i386       0.8.1a-1.2.2             dvd        17 k
...
Installing for dependencies:
libdbi                    i386       0.8.1-2.1                dvd        35 k
libdbi-drivers            i386       0.8.1a-1.2.2             dvd        14 k
...

Transaction Summary
============================================
Install     12 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 17 M
Is this ok [y/N]: y
Downloading Packages:
----------------------------------------------------------------------------
Total                                            1.3 GB/s |  17 MB     00:00
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
 Installing     : perl-DBI                                                1/12
 Installing     : mysql                                                   2/12
 ...
 Installing     : libdbi-dbd-mysql                                       12/12

Installed:
 MySQL-python.i386 0:1.2.1-1
 libdbi-dbd-mysql.i386 0:0.8.1a-1.2.2
 ...

Dependency Installed:
 libdbi.i386 0:0.8.1-2.1              libdbi-drivers.i386 0:0.8.1a-1.2.2
 ...

Complete!
```

回页首

在图形界面下补装软件包

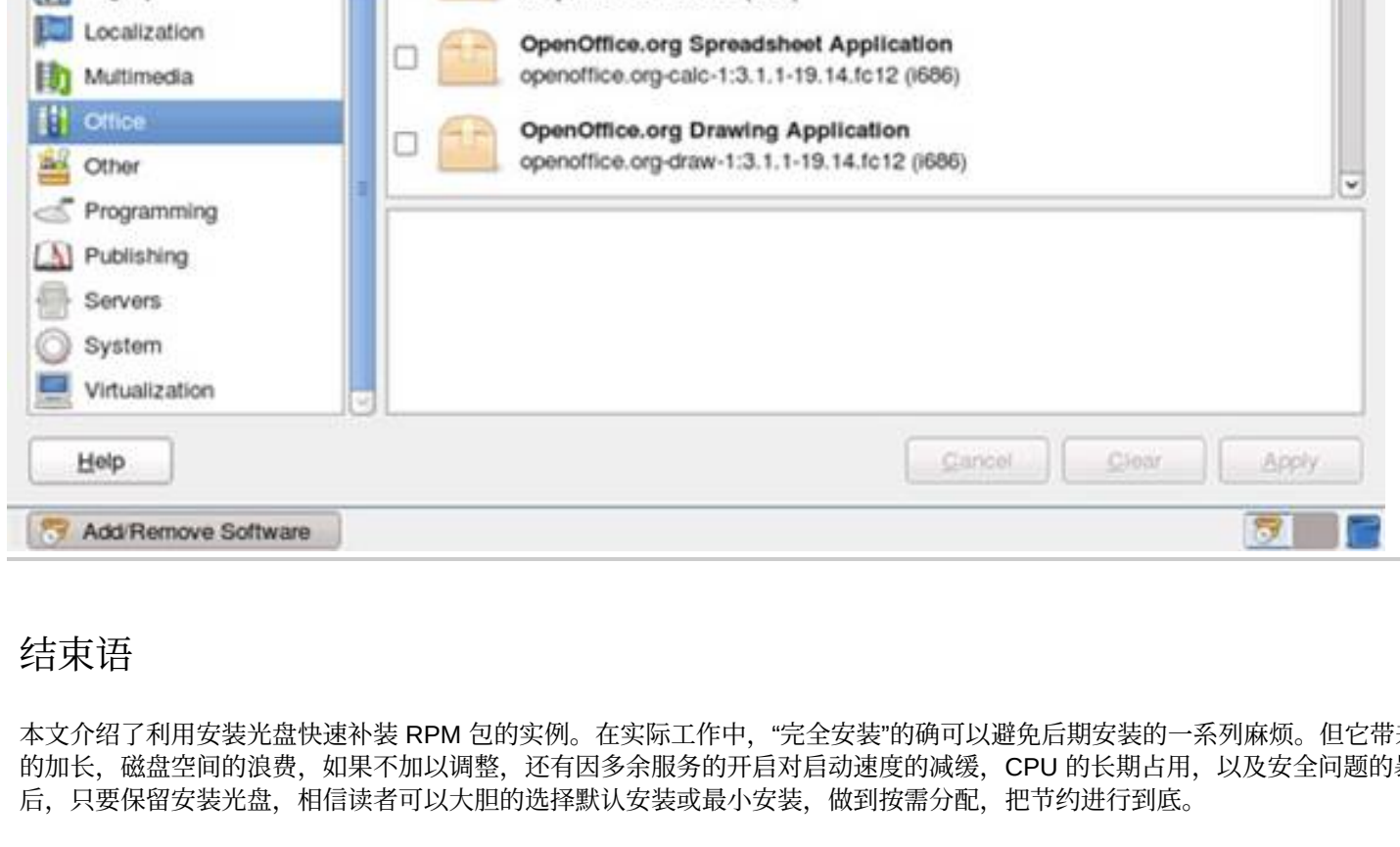
在桌面环境已经具备的情况下，RPM 软件包的补装将更加便捷。你甚至不需要记住任何一个 yum 命令的参数，而且操作界面与系统初装时几乎一致。相信这正是大部分用户在想要补装软件包时想要的。

在 RHEL 5 中，通过菜单里的 "Add/Remove Software" 启动包管理器。同时，也可能在 console 下通过 "system-config-packages" 或者 "pirut" 启动它。

图 1. 启动包管理器

在 Fedora 中，同样可以通过菜单里的 "Add/Remove Software" 启动 GNOME 包管理器。不同的是，该工具由 gnome-packagekit 提供，可以在 console 下通过 "gpk-application" 启动。

图 2. 启动 GNOME 包管理器



回页首

结束语

本文介绍了利用安装光盘快速补装 RPM 包的实例。在实际工作中，“完全安装”的确可以避免后期安装的一系列麻烦。但它带来的缺点，除了安装过程的加长，磁盘空间的浪费，如果不加以调整，还有因多余服务的开启对启动速度的减缓，CPU 的长期占用，以及安全问题的暴露，等等。在阅读本文后，只要保留安装光盘，相信读者可以大胆的选择默认安装或最小安装，做到按需分配，把节约进行到底。

参考资料

- 查看 DeveloperWorks 文章 Linux平台上的软件包管理，了解 RPM 命令常用参数的使用示例，以及实用的常用问题列表。
- 查看 YUM 在 wiki 上的描述。
- 查看 RPM 的命令手册，了解详细的命令参数及用法。
- 查看 RHEL5安装手册关于包选择的部分。
- 查看 YUM手册安装源向导，了解创建安装源的详细信息。
- 查看 YUM手册分组包向导，了解与包组相关的 yum 命令，以及如何自定义分组包。

关于作者

戴月，曾是 IBM 中国软件开发中心的一名软件工程师，从事 Lotus Notes 测试工作。联系方式：daiyue715@163.com。

建议

回页首