第六章：软件包安装

尚硅谷云计算 Linux 课程

版本：V1.0

讲师：沈超

一　软件包分类

1、软件包分类

- 源码包
- 二进制包

（脚本安装包）

2、源码包

2.1 源码包什么样

```
[root@localhost ~]# vim  hello.c
#include <stdio.h>
int main (void)
{
        printf ("hello world\n");
}
```

```
[root@localhost ~]# rpm -ivh /mnt/cdrom/Packages/gcc-4.4.6-4.el6.i686.rpm
```

```
[root@localhost ~]# gcc -c hello.c
#-c 生成“.o”头文件。这里会生成 hello.o 头文件，但是不会生成执行文件
[root@localhost ~]# gcc -o hello hello.o
#-o 生成执行文件，并制定执行文件名。这里生成的 hello 就是可执行文件
[root@localhost ~]# ./hello
hello world
#执行 hello 文件
```

2.2 源码包特点

源码包的优点是：

- ✧ 开源，如果有足够的能力，可以修改源代码
- ✧ 可以自由选择所需的功能
- ✧ 软件是编译安装，所以更加适合自己的系统，更加稳定也效率更高

- ✧ 卸载方便

源码包有缺点吗？

- ✧ 安装过程步骤较多，尤其安装较大的软件集合时（如 LAMP 环境搭建），容易出现拼写错误
- ✧ 编译过程时间较长，安装比二进制安装时间长
- ✧ 因为是编译安装，安装过程中一旦报错新手很难解决

3、二进制包

3.1 二进制包分类

- ✧ DPKG 包：是由 Debian Linux 所开发出来的包管理机制，通过 DPKG 包，Debian Linux 就可以进行软件包管理。主要应用在 Debian 和 unbuntu 中。
- ✧ RPM 包：是由 Red Hat 公司所开发的包管理系统。功能强大，安装、升级、查询和卸载都非常简单和方便。目前很多 Linux 都在使用这种包管理方式，包括 Fedora、CentOS、SuSE 等。我们学习的是 CentOS 6.3，所以我们将要学习 RPM 包管理系统

3.2 特点

RPM 包的优点：

- ✧ 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、查询和卸载
- ✧ 安装速度比源码包安装快的多

RPM 包的缺点：

- ✧ 经过编译，不再可以看到源代码
- ✧ 功能选择不如源码包灵活
- ✧ 依赖性。有时我们会发现需要安装软件包 a 时需要先安装 b 和 c，而安装 b 时需要安装 d 和 e。这是需要先安装 d 和 e，再安装 b 和 c，最后才能安装 a 包。比如说，我买了个漂亮的灯具，打算安装到我们家客厅，可是在安装灯具之前我们家客厅总要有顶棚吧，顶棚总要是做好了防水和刷好油漆了吧，这个装修和安装软件其实类似总要有一定的顺序的。可是有时依赖性会非常繁琐

3.3 RPM包依赖

1）树形依赖　　a----->b----->c

2）环形依赖　　a----->b----->c----->a

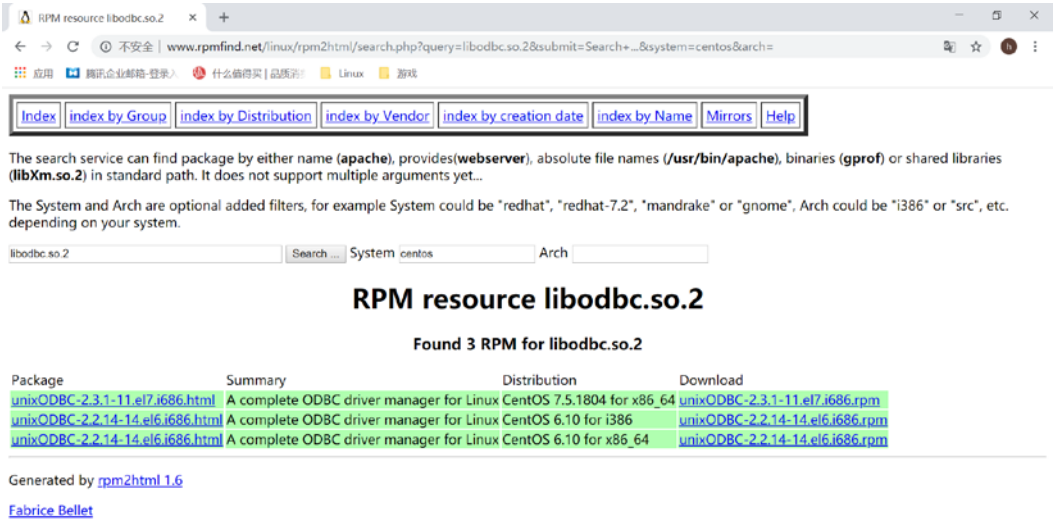
3）函数库依赖

什么是模块依赖？我们举一个例子，尝试安装以下文件：

```
[root@localhost Packages]# rpm  -ivh  mysql-connector-odbc-5.2.5-7.el7.x86_64.rpm
```

```
错误：依赖检测失败：
    libodbc.so.2()(64bit) 被 mysql-connector-odbc-5.2.5-7.el7.x86_64 需要
    libodbcinst.so.2()(64bit) 被 mysql-connector-odbc-5.2.5-7.el7.x86_64 需要
```

发现报错，需要安装“libodbc.so.2”函数库文件，这时会发现在光盘中根本找不到这个文件。那是因为函数库没有单独成包，是包含在某一个软件包中的。而如果要知道在哪个软件包中，需要查询网站 www.rpmfind.net，如图：



二　rpm安装

1　rpm包命名规则

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

httpd	软件包名
2.2.15	软件版本
15	软件发布的次数
el6	软件发行商。el6 是 RedHat 公司发布，适合 RHEL6.x(Red Hat Enterprise Linux)和 CentOS6.x 下使用
i686	适合的硬件平台。RPM 包可以在不同的硬件平台安装，选择适合不同 CPU 的软件版本，可以最大化的发挥 CPU 性能，所以出现了所谓的 i386（386 以上计算机都可以安装）、i586（586 以上的计算机都可以安装）、i686（奔腾 II 以上计算机都可以安装，目前所有的 CPU 都是奔腾 II 以上，所以这个软件版本居多）、x86_64（64 位 CPU 可以安装）和 noarch（没有硬件限制）等文件名了。
rpm	rpm 包的扩展名。我们说过 Linux 下文件不是靠扩展名区分文件类型，也就是 Linux 中扩展名没有任何含义。可是这里怎么又出现了扩展名呢？原因很简单，如果我不把 RPM 的扩展名叫做“.rpm”，管理员很难知道这是一个 RPM 包，当然也就无法正确安装了。也就是说如果 RPM 包不用“.rpm”作为扩展名，系统可以正确识别没有问题，可是管理员很难识别这是个什么样的软件。

包全名：如果操作的是未安装软件包，则使用包全名，而且需要注意绝对路径

包名：如果操作的是已经安装的软件包，则使用包名即可，系统会生产 RPM 包的数据库

（/var/lib/rpm/），而且可以在任意路径下操作

2 依赖性

已经讲过了，不再重复。

3 rpm包手工命令安装

3.1 默认安装位置

RPM 包默认安装路径	
/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

3.2 RPM 包安装

1)安装命令

```
rpm  -ivh 包全名
#注意一定是包全名。如果跟包全名的命令要注意路径，因为软件包在光盘当中
选项：
    -i      install 安装（install）
    -v      显示更详细的信息（verbose）
    -h      打印#显示安装进度（hash）
```

- --nodeps 不检测依赖性安装。软件时会检测依赖性，确定所需的底层软件是否安装。如果没有安装则会报错。如果我不管依赖性，想强行安装，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是不能使用的，所以不建议这样做
- --replacefiles 替换文件安装。如果安装软件包，可是包中部分文件已经存在，那么正常安装时候，会报错“某个文件已经存在”从而导致软件无法安装，使用这个选项可以忽视这个报错，而覆盖安装
- --replacepkgs 替换软件包安装。如果软件包已经安装，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。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。注意：如果指定了安装路径，软件没有安装到系统默认路径中的话，系统会找不到这些安装的软件，需要进行手工配置才能被系统识别。所以 rpm 包我们一般都采用默认路径安装。

2）服务启动

```
[root@localhost ~]# service 服务名 start|stop|restart|status
参数：
    start:      启动服务
    stop:       停止服务
```

```
    restart:    重启服务
    status:     查看服务状态
[root@localhost ~]# systemctl restart httpd
#这个命令也行
```

3.3 RPM 包升级

```
[root@localhost ~]#  rpm  -Uvh 包全名
选项:
    -U（大写）   升级安装，如果没有安装过，系统直接安装。如果安装过的版本较旧，则
                 升级到新版本（upgrade）
[root@localhost ~]#  rpm  -Fvh 包全名
选项:
-F（大写）       升级安装,如果没有安装过,则不会安装。必须安装有较旧版本,才能升级(freshen)
```

3.4 卸载

```
[root@localhost ~]# rpm  -e  包名
选项:
    --nodeps    不检查依赖性
    -e          卸载
```

3.5 查询

1）查询软件包是否安装

可以查询软件包是否安装，命令格式如下：

```
[root@localhost ~]#  rpm  -q 包名
选项:
    -q:  查询（query）
```

2）查询系统中的所有安装软件包

可以查询 Linux 系统中所有已经安装的软件包，命令格式如下：

```
[root@localhost ~]# rpm -qa
选项:
    -a:所有（all）
```

当然，可以用管道符来查看所需的内容，比如：

```
[root@localhost ~]# rpm -qa | grep httpd
```

你会发现，使用“rpm -q 包名”只能查看这个包是否安装，但是使用“rpm -qa | grep 包名”会把包含包名称的所有包都列出来。

3）查询软件包的详细信息

可以查询已经安装的某个软件包的详细信息，命令格式如下：

```
[root@localhost ~]#  rpm  -qi 包名
选项：
    -i：  查询软件信息（information）
```

也可以查询还没有安装的软件包的详细信息，命令格式如下：

```
[root@localhost ~]#  rpm  -qip 包全名
选项：
    -p：  查询没有安装的软件包（package）
```

4）查询软件包中的文件列表

可以查询已经安装的软件包中的文件列表和安装的完整目录，命令格式如下：

```
[root@localhost ~]#  rpm  -ql 包名
选项：
    -l：  列出软件包中所有的文件列表和软件所安装的目录（list）
```

那么，可以查询还没有安装的软件包中的文件列表和打算安装的位置吗？答案是可以，命令格式如下：

```
[root@localhost ~]#  rpm  -qlp 包全名
选项：
    -p：  查询没有安装的软件包信息（package）
```

5）查询系统文件属于哪个 RPM 包

既然可以知道每个 RPM 包中的文件的安装位置，那么可以查询系统文件属于哪个 RPM 包吗？当然可以，不过需要注意的是，手工建立的文件是不能查询的，因为这些文件不是通过 RPM 包安装的，当然不能反向查询它属于哪个 RPM 包。命令格式如下：

```
[root@localhost ~]#  rpm  -qf 系统文件名
选项：
    -f：  查询系统文件属于哪个软件包（file）
```

6）查询软件包所依赖的软件包

查询系统中和已经安装的软件包有依赖关系的软件包，命令格式如下：

```
[root@localhost ~]# rpm  -qR 包名
选项：
    -R：  查询软件包的依赖性（requires）
```

可以查询没有安装的软件包的依赖性吗？加“-p”选项即可。例如，查看一下还没有安装的 bind 软件包的依赖包，可以执行如下命令：

```
[root@localhost ~]# rpm -qRp /mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm
```

3.6 验证

1）基本命令

```
[root@localhost ~]# rpm  -Va
选项：
    -Va 校验本机已经安装的所有软件包
```

```
[root@localhost ~]# rpm -V 已安装的包名
选项：
    -V  校验指定 RPM 包中的文件（verify）

 [root@localhost ~]# rpm -Vf 系统文件名
选项：
    -Vf 校验某个系统文件是否被修改
```

2）验证举例

```
[root@localhost ~]# rpm -V httpd
S.5....T.            c       /etc/httpd/conf/httpd.conf
验证内容       文件类型       文件名
```

出现了提示信息，我们来解释下最前面共有 8 个信息内容，是表示验证内容的。文件名前面的 c 是表示这是个配置文件（configuration）。最后是文件名。那么验证内容中的 8 个信息的具体内容如下：

- S 文件大小是否改变
- M 文件的类型或文件的权限（rwx）是否被改变
- 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

apache 配置文件的文件类型是 c，那么还有哪些文件类型呢？

- c 配置文件（config 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）

3.7 数字证书

刚刚的校验方法只能对已经安装的 RPM 包中的文件进行校验，但是如果 RPM 包本身就被动过手脚，那么校验就不能解决问题了。我们就必须使用数字证书验证了。

数字证书有如下特点：

- 首先必须找到原厂的公钥文件，然后进行安装
- 再安装 RPM 包是，会去提取 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，则不允许安装并警告

1）数字证书位置

那么数字证书在哪里呢？其实在 CentOS 6.3 的第一张光盘中就有，当然它默认也会放在系统中。

```
[root@localhost ~]# ll /mnt/cdrom/RPM-GPG-KEY-CentOS-6
-r--r--r-- 2 root root 1706 7 月    2 04:21 /mnt/cdrom/RPM-GPG-KEY-CentOS-6
#光盘中的数字证书位置
```

```
[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
#系统中的数字证书位置
```

2）数字证书导入

```
[root@localhost ~]# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
选项：
    --import    导入数字证书
```

我们如何查询系统中安装好的数字证书呢？命令如下：

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

3.8 RPM包中文件的提取

1）cpio命令

cpio 命令主要有三种基本模式：“-o”模式指的是 copy-out 模式，就是把数据备份到文件库中；“-i”模式指的是 copy-in 模式，就是把数据从文件库中恢复；“-p”模式指的是复制模式，就是不把数据备份到 cpio 库中，而是直接复制为其他文件。命令如下：

```
[root@localhost ~]# cpio -o[vcB] > [文件|设备]
#备份
选项：
    -o：copy-out 模式，备份
    -v：显示备份过程
    -c：使用较新的 portable format 存储方式
    -B：设定输入输出块为 5120bytes，而不是模式的 512butes
[root@localhost ~]# cpio -i[vcdu] < [文件|设备]
#还原
选项：
    -i：copy-in 模式，还原
    -v：显示还原过程
    -c：使用较新的 portable format 存储方式
    -d：还原时自动新建目录
    -u：自动使用较新的文件覆盖较旧的文件
[root@localhost ~]# cpio -p 目标目录
```

举几个例子吧，先来看看使用 cpio 备份数据的方法，命令如下：

```
例子：利用 find 命令找到文件，备份
[root@localhost ~]# find /etc -print | cpio -ocvB > /root/etc.cpio
#利用 find 指定要备份/etc/目录，使用>导出到 etc.cpio 文件
[root@localhost ~]# ll -h etc.cpio
-rw-r--r--. 1 root root 21M 6 月   5 12:29 etc.cpio
#etc.cpio 文件生成
```

再来看看如何恢复 cpio 的备份数据，命令如下：

```
[root@localhost ~]# cpio -idvcu < /root/etc.cpio
#还原 etc 的备份
```

```
#但是如果大家查看下当前目录/root，会发现没有生成 etc 目录。这是因为备份是/etc 目录使用的是绝对路径，所以恢复的数据直接恢复到了/etc 系统目录中，而没有生成在/root/etc 中。
```

在 CentOS5.x 的版本中，是可以利用上面的命令备份与恢复指定的文件。但是到 CentOS6.x 当中，需要更加严谨。如果备份时使用绝对路径，则恢复的数据会直接到绝对路径指定的路径中，如果需要把数据恢复到当前目录中，则需要使用相对路径，例如：

```
备份：
[root@localhost ~]# cd /etc
#进入/etc 目录
[root@localhost ~]# find . -print | cpio -ocvB > /root/etc.cpio
#利用 find 指定要备份/etc/目录，使用>导出到 etc.cpio 文件

恢复：
[root@localhost ~]# cd /root
#回到/root 目录中
[root@localhost ~]# mkdir etc_test
#建立恢复测试目录
[root@localhost ~]# cd etc_test
#进入测试目录，数据恢复到此
[root@localhost etc_test]# cpio -idvcu < /root/etc.cpio
#还原/etc 目录的数据，因为备份时使用的是相对路径，则会还原到/root/etc_test/目录下
```

最后来演示一下 cpio 命令的“-p”复制模式，命令如下：

```
[root@localhost ~]# cd /tmp/
#进入/tmp 目录
[root@localhost tmp]# rm -rf *
#删除/tmp 目录中所有数据
[root@localhost tmp]# mkdir test
#建立备份目录
[root@localhost tmp]# find /boot/ -print | cpio -p /tmp/test
#备份/boot/目录到/tmp/test/目录中
[root@localhost tmp]# ls test/
boot
#在/tmp/test/目录中备份出了 boot 目录
```

2）提取 RPM 包中文件

```
[root@localhost ~]# rpm2cpio 包全名 | cpio -idv .文件绝对路径
    rpm2cpio          ←将 rpm 包转换为 cpio 格式的命令
    cpio              ←是一个标准工具，它用于创建软件档案文件和从档案文件中提取文件
```

举个例子，现在我假设把系统中的/bin/ls 命令不小心误删除了，那么我可以修复回来吗？这时有两种方法修复，要不就是使用—force 选项覆盖安装一遍 coreutils-8.4-19.el6.i686 包，要不就可以使用 cpio 命令提取出/bin/ls 命令文件，再把它拷贝到对应位置就可以了。不过我是怎么知道/bin/ls 命令是属于 coreutils-8.4-19.el6.i686 这个软件包的呢？还记得-qf 选项吗？命令如下：

```
[root@localhost ~]# rpm -qf /bin/ls
```

```
coreutils-8.4-19.el6.i686
#查看 ls 文件属于哪个软件包
```

那么我们在讲 RPM 包中文件提取，所以我们使用第二章方法，cpio 命令提取出 ls 命令文件，然后拷贝到对应位置，命令如下：

```
[root@localhost ~]# mv /bin/ls /root/
#把/bin/ls 命令移动到/root 目录下，造成误删除的假象

[root@localhost ~]# ls
-bash: ls: command not found
#这时执行 ls 命令，系统会报错“命令没有找到”

[root@localhost ~]# rpm2cpio /mnt/cdrom/Packages/coreutils-8.4-19.el6.i686.rpm | cpio -idv ./bin/ls
./bin/ls
24772 块
#提取 ls 命令文件到当前目录下

[root@localhost ~]# cp /root/bin/ls /bin/
#把提取出来的 ls 命令文件复制到/bin 目录下

[root@localhost ~]# ls
anaconda-ks.cfg  bin  inittab  install.log  install.log.syslog  ls
#恭喜你，ls 命令又可以正常使用了
```

4. rpm包在线安装（yum安装）

4.1 yum源文件解析

yum 源配置文件保存在/etc/yum.repos.d/目录中，文件的扩展名一定是“*.repo”。也就是说，yum 源配置文件只要扩展名是“*.repo”就会生效。

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

这个目录中有 5 个 yum 源配置文件，默认情况下 CentOS-Base.repo 文件生效。我们打开这个文件看看，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…
```

在 CentOS-Base.repo 文件中有 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。我们解释一下 base 这个容器。

- [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=1 则表示此容器生效，写成 enabled=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。

4.2 搭建本地光盘yum源

第一步：放入 CentOS 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/sr0 is write-protected, mounting read-only
#挂载光盘到/mnt/cdrom 目录下
```

第二步：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。当然也可以删除其他几个 yum 源配置文件，但是如果删除了，当你又想用网络作为 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
```

第三步：修改光盘 yum 源配置文件 CentOS-Media.repo，参照以下方法修改：

```
[root@localhost yum.repos.d]# vim CentOS-Media.repo
[c6-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/
#注释这两个不存在的地址
gpgcheck=1
enabled=1
#把 enabled=0 改为 enabled=1，让这个 yum 源配置文件生效
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

配置完成，现在可以感受一下yum的便捷了。

4.3 yum命令

1）查询

- 查询 yum 源服务器上所有可安装的软件包列表。

```
[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
ConsoleKit.i686 0.4.1-3.el6  @anaconda-CentOS-201207051201.i386/6.3
ConsoleKit-libs.i686 0.4.1-3.el6  @anaconda-CentOS-201207051201.i386/6.3
...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...
```

- 查询 yum 源服务器中是否包含某个软件包。

```
[root@localhost yum.repos.d]# yum list 包名
#查询单个软件包
例如：
[root@localhost yum.repos.d]# yum list samba
Available Packages
samba.i686                3.5.10-125.el6                    c6-media
```

- 搜索 yum 源服务器上所有和关键字相关的软件包。

```
[root@localhost yum.repos.d]# yum search 关键字
#搜索服务器上所有和关键字相关的软件包
例如：
[root@localhost ~]# yum search ifconfig
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
================================================================    匹    配    ：    ifconfig
================================================================
net-tools.x86_64 : Basic networking tools
```

yum search 搜索可以用于确定某个软件在哪个相关包当中。此例子可以确定“ifconfig”命令需要安装“net-tools”包。

- 查询指定软件包的信息。

```
[root@localhost yum.repos.d]# yum info samba
#查询 samba 软件包的信息
```

```
Available Packages          ←还没有安装
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        : 4.9 M         ←大小
Repo        : c6-media      ←在光盘上
…省略部分输出…
```

2）安装

```
[root@localhost yum.repos.d]# yum -y install 包名
选项：
    install     安装
    -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
例如：
[root@localhost yum.repos.d]# yum -y install gcc
#使用 yum 自动安装 gcc
```

3）升级

```
[root@localhost yum.repos.d]# yum -y update 包名
#升级指定的软件包
选项：
    update:         升级
    -y:             自动回答 yes
```

注意：在进行升级操作时，yum 源服务器中软件包的版本要比本机安装的软件包的版本高。

```
[root@localhost yum.repos.d]# yum -y update
#升级本机所有软件包
```

这条命令会升级系统中所有的软件包。不过我们的生产服务器是稳定优先的，所以这种全系统升级的情况并不多见。

4）卸载

再次强调一下，除非你确定卸载的软件的依赖包不会对系统产生影响，否则不要执行 yum 的卸载，因为很有可能在卸载软件包的同时卸载的依赖包也是重要的系统文件，这就有可能导致系统崩溃。卸载命令如下：

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

4.4 yum组管理命令

✧ 查询可以安装的软件组

```
[root@localhost ~]# yum grouplist
#列出所有可用的软件组列表
```

✧ 查询软件组内包含的软件

```
[root@localhost ~]# yum groupinfo 软件组名
#列出软件组中包含的软件
例如：
[root@localhost ~]# yum groupinfo "Web Server"
#查询软件组"Web Server"中包含的软件
```

✧ 安装软件组

```
[root@localhost ~]# yum groupinstall 软件组名
#安装指定软件组，组名可以由 grouplist 查询出来
例如：
[root@localhost ~]# yum groupinstall "Web Server"
#安装网页服务软件组
```

✧ 卸载软件组

```
[root@localhost ~]# yum groupremove 软件组名
#卸载指定软件组
```

三. 源码包安装

1. 注意事项

1.1 应该选择哪种软件包？

- 如果软件包是给大量客户提供访问，建议使用源码包安装，如LAMP环境搭建，因为源码包效率更高。
- 如果软件包是给Linux底层使用，或只给少量客户访问，建议使用rpm包安装，因为rpm包简单。

1.2 源码包是从哪里来的？

rpm包是光盘中直接包含的，所以不需要用户单独下载。而源码包是通过官方网站下载的，如果需要使用，是需要单独下载的。

1.3 是否可以在系统中即安装rpm包的Apache，又安装源码包的Apache？

答案是可以，因为两种安装方法安装的Apache，安装位置是不一样的，例如：

```
RPM 包：不建议指定安装位置的，建议安装在默认位置（RPM 包安装的服务有标准卸载命令，不怕文件到处安装）
    配置文件：      /etc/httpd/conf/httpd.conf
    网页位置：      /var/www/html/
    日志位置：      /var/log/httpd/
    启动方法：      1） service  httpd  restart
                    2） /etc/rc.d/init.d/httpd  restart
```

源码包：必须制定安装位置（源码包没有安装数据库，没有删除命令）
配置文件：　　/usr/local/apache2/conf/httpd.conf
网页文件：　　/usr/local/apache2/htdocs/
日志位置：　　/usr/local/apache2/logs/
启动方法：　　/usr/local/apache2/bin/apachectl　start

1.4 生产服务器上，是否会同时安装两种Apache？

当然不会啊，因为系统中只有一个80端口，所以你只能启动一个Apache，装多个只能浪费资源。我们建议安装源码包的Apache。

2. 安装过程

我们来解释一下源码包安装的具体步骤。

（1）下载软件包。

（2）解压缩。

（3）进入解压目录。

（4）./configure　　编译前准备

这一步主要有三个作用：

① 在安装之前需要检测系统环境是否符合安装要求。

② 定义需要的功能选项。“./configure”支持的功能选项较多，可以执行“./configure --help”命令查询其支持的功能。一般都会通过“./configure --prefix=安装路径”来指定安装路径。

③ 把系统环境的检测结果和定义好的功能选项写入 Makefile 文件，后续的编译和安装需要依赖这个文件的内容。

需要注意的是，configure 不是系统命令，而是源码包软件自带的一个脚本程序，所以必须采用“./configure”方式执行（“./”代表在当前目录下）。

（5）make　　编译

make 会调用 gcc 编译器，并读取 Makefile 文件中的信息进行系统软件编译。编译的目的就是把源码程序转变为能被 Linux 识别的可执行文件，这些可执行文件保存在当前目录下。编译过程较为耗时，需要有足够的耐心。

（6）make clean：清空编译内容（非必需步骤）。

如果在“./configure”或“make”编译中报错，那么我们在重新执行命令前一定要记得执行 make clean 命令，它会清空 Makefile 文件或编译产生的“.o”头文件。

（7）make install：　　编译安装

这才是真正的安装过程，一般会写清楚程序的安装位置。如果忘记指定安装目录，则可以把这个命令的执行过程保存下来，以备将来删除使用。

3. 删除

源码包没有删除命令，如果需要删除，直接删除安装目录即可。

4. 打入补丁

4.1 补丁的生成

```
[root@localhost ~]# diff 选项 old new
#比较 old 和 new 文件的不同
```

选项：

```
    -a      将任何文档当做文本文档处理
    -b      忽略空格造成的不同
    -B      忽略空白行造成的不同
    -I      忽略大小写造成的不同
    -N      当比较两个目录时，如果某个文件只在一个目录中，则在另一个目录中视作空
        文件
    -r      当比较目录时，递归比较子目录
    -u      使用同一的输出格式
```

举例

```
[root@localhost ~]# mkdir test
#建立测试目录
[root@localhost ~]# cd test
#进入测试目录
[root@localhost test]# vi old.txt
our
school
is
atguigu
#文件 old.txt，为了一会输出便于比较，每行分开
[root@localhost test]# vi new.txt
our
school
is
atguigu
in
Beijing
#文件 new.txt
```

比较下两个文件的不同，并生成补丁文件“txt.patch”，命令如下：

```
[root@localhost test]# diff -Naur /root/test/old.txt /root/test/new.txt > txt.patch
#比较两个文件的不同，同时生成 txt.patch 补丁文件

[root@localhost test]# vi txt.patch
#查看下这个文件
--- /root/test/old.txt  2012-11-23 05:51:14.347954373 +0800
#前一个文件
+++ /root/test/new.txt  2012-11-23 05:50:05.772988210 +0800
#后一个文件
```

```
@@ -2,3 +2,5 @@
 school
 is
 atguigu
+in
+beijing
```

#后一个文件比前一个文件多两行（+表示）

4.2 打入补丁

```
[root@localhost test]# patch  -pn < 补丁文件
#按照补丁文件进行更新
```

选项：

-pn　　n 为数字。代表按照补丁文件中的路径，指定更新文件的位置。

“-pn”不好理解，我们说明下。补丁文件是要打入旧文件的，但是你当前所在的目录和补丁文件中的记录的目录是不一定匹配的，所以就需要“-pn”来同步两个目录。

比如我当前是在“/root/test”目录中（我要打补丁的旧文件就在当前目录下），补丁文件中记录的文件目录为“/root/test/old.txt”，这时如果写入“-p1”（在补丁文件目录中取消一级目录）那么补丁文件就会打入“/root/test/root/test/old.txt”文件中，这显然是不对的。那如果写入的是“-p2”（在补丁文件目录中取消二级目录）那么补丁文件打入的就是“/root/test/test/old.txt”，这显然也不对。如果写入的是“-p3”（在补丁文件目录中取消三级目录）那么补丁文件就是打入的“/root/test/old.txt”，我们的 old.txt 文件就在这个目录下，所以就应该是“-p3”。

那么我们更新下“old.txt”文件，命令如下：

```
[root@localhost test]# patch -p3 < txt.patch
patching file old.txt
#给 old.txt 文件打补丁

[root@localhost test]# cat old.txt
#查看下 old.txt 的内容吧。
our
school
is
atguigu
in
Beijing
#多出来了 in Beijing 两行
```

四 脚本安装程序

1　脚本程序简介

脚本程序包并不多见，所以在软件包分类中并没有把它列为一类。它更加类似于 Windows 下的程

序安装，有一个可执行的安装程序，只要运行安装程序，然后进行简单的功能定制选择（比如指定安装目录等），就可以安装成功，只不过是在字符界面下完成的。

目前常见的脚本程序以各类硬件的驱动居多，我们需要学习一下这类软件的安装方式，以备将来不时之需。

2 Webmin 安装

2.1. 简介

我们来看看脚本程序如何安装和使用。安装一个叫作 Webmin 的工具软件，Webmin 是一个基于 Web 的系统管理界面。借助任何支持表格和表单的浏览器（和 File Manager 模块所需要的 Java），你就可以设置用户账号、apache、DNS、文件共享等。Webmin 包括一个简单的 Web 服务器和许多 CGI 程序，这些程序可以直接修改系统文件，比如/etc/inetd.conf 和 /etc/passwd。Web 服务器和所有的 CGI 程序都是用 Perl 5 编写的，没有使用任何非标准 Perl 模块。也就是说，Webmin 是一个用 Perl 语言写的、可以通过浏览器管理 Linux 的软件。

2．2. 安装步骤

首先下载 Webmin 软件，地址为 http://sourceforge.net/projects/webadmin/files/webmin/，这里下载的是 webmin-1.610.tar.gz。

接下来解压缩软件，命令如下：

```
[root@localhost ~]# tar -zxvf webmin-1.610.tar.gz
```

进入解压目录，命令如下：

```
[root@localhost ~]# cd webmin-1.610
```

执行安装程序 setup.sh，并指定功能选项，命令如下：

```
[root@localhost webmin-1.610]# ./setup.sh
**************************************************************************
*          Welcome to the Webmin setup script, version 1.610       *
**************************************************************************
Webmin is a web-based interface that allows Unix-like operating
systems and common Unix services to be easily administered.

Installing Webmin in /root/webmin-1.610 ...

**************************************************************************
Webmin uses separate directories for configuration files and log files.
Unless you want to run multiple versions of Webmin at the same time
you can just accept the defaults.

Config file directory [/etc/webmin]:
#选择安装位置，默认安装在/etc/webmin 目录下。如果安装到默认位置，则直接回车
Log file directory [/var/webmin]:
```

```
#日志文件保存位置，直接回车，选择默认位置
*********************************************************************
Webmin is written entirely in Perl. Please enter the full path to the
Perl 5 interpreter on your system.

Full path to perl (default /usr/bin/perl):
#指定 Perl 语言的安装位置，直接回车，选择默认位置，Perl 默认就安装在这里
Testing Perl ...
Perl seems to be installed ok

*********************************************************************
Operating system name:    CentOS Linux
Operating system version: 6.3

*********************************************************************
Webmin uses its own password protected web server to provide access
to the administration programs. The setup script needs to know :
 - What port to run the web server on. There must not be another
   web server already using this port.
 - The login name required to access the web server.
 - The password required to access the web server.
 - If the webserver should use SSL (if your system supports it).
 - Whether to start webmin at boot time.

Web server port (default 10000):
#指定 Webmin 监听的端口，直接回车，默认选定 10000
Login name (default admin):admin
#输入登录 Webmin 的用户名
Login password:
Password again:
#输入登录密码
The Perl SSLeay library is not installed. SSL not available.
#apache 默认没有启动 SSL 功能，所以 SSL 没有被激活
Start Webmin at boot time (y/n):y
#是否在开机的同时启动 Webmin
...安装过程省略...
Webmin has been installed and started successfully. Use your web
browser to go to

  http://localhost:10000/

and login with the name and password you entered previously.
#安装完成
```