软件包安装与管理

**软件包类型**

* 二进制包（二进制封装包）

rpm软件包 扩展名为“.rpm”

dpkg软件包 扩展名为“.deb”

* 源代码包（tar包）

一般为“.tar.gz”、“.tar.bz2”等格式的压缩包，包含程序的原始代码

* 附带安装程序的软件包

在压缩包内提供install.sh、setup等脚本安装程序

* 绿色免安装软件包

在压缩包内提供已编译好的执行程序，解压到指定位置即可直接使用

**二进制包的特点**

RPM

RPM是Red Hat Package Manager的缩写，Red Hat提供的一种软件包管理系统，用于安装、卸载、升级等操作，以\*.rpm作为软件包的后缀名。

DPGK

DPGK是Debian Packager的简写，Debian提供的一种软件包管理系统，用于安装、卸载、升级等操作，以\*.deb作为软件包的后缀名。

二进制软件包的特点

1. 专门的工具进行安装、卸载、升级和管理软件
2. 安装简洁、速度快
3. 经过封装，无法直接获取源代码
4. 功能选择性差，功能定制不灵活

Linux软件包的命名也有一定的规律，一般遵循：名称-版本-修正版-类型

tree-1.2.3-4.x86\_64.tar.gz

tree-1.2.3-4.el6.x86.rpm

软件名：tree

版本号：1.2.3

修订次数：4

适用系统：el5 el6 el7

应用平台：i386 i686 x86 x86\_64 amd64 noarch不区分平台

软件类型：.rpm .deb .tar.gz 等

二进制软件包安装位置

/etc/ 配置文件保存位置

/usr/bin/ 软件命令保存位置

/usr/lib/ 函数文件保存位置

/usr/share/doc/ 使用手册保存位置

/usr/share/man/ 帮助文档保存位置

二进制软件包的依赖关系

树形依赖

A

B

A

A

B

C

B

C

C

依赖方式：

软件A—软件B—软件C

软件A—软件B—函数C

**二进制软件包管理之--RPM**

1） 安装命令：

格式：rpm -ivh 软件包全名

-i 安装一个新的rpm软件包

-v 显示安装过程中的详细信息

-h 以“#”号显示安装的进度

--nodeps 不检测依赖安装

--force 强制安装，不管软件是否存在，都强制重新安装（修复软件）

--test 测试安装，不真正安装，检测软件依赖关系

--prefix=path 指定安装路径，不按照默认路径安装

2） 卸载命令:

格式：rpm –e 软件名

-e 卸载软件（注意卸载顺序）

--nodeps 不检测依赖卸载

3） 升级命令:

格式：rpm –Uvh 软件包全名

-U 升级安装，若没安装，则直接安装，若安装有旧版本，则升级安装

-F 升级安装，若没安装，则不安装，必须有旧版本，才进行升级安装

4） 查询已安装的软件信息：

格式：rpm -q[子选项] 软件名

-q 软件名 #查看指定软件是否安装

-qa #查看系统中所有已安装的软件

-qi 软件名 #查看指定软件的详细信息

-ql 软件名 #查看指定软件安装的目录和文件列表

-qR 软件名 #查看指定软件的依赖关系

5） 查询未安装的软件包信息：

格式：rpm -qp[子选项] 软件包全名

-qpi 包全名 #查看指定未安装软件的详细信息

-qpl 包全名 #查看指定未安装软件的默认安装位置（预览）

-qpR 包全名 #查看指定未安装软件的依赖关系

6） 查询文件或者目录是由哪个软件安装产生的：

格式：rpm –qf 文件或者目录

7） 校验已安装的软件包信息

格式：rpm -V 软件名

#验证指定软件安装后的完整性，及文件的元数据是否发生变化

rpm -Vf 文件名（一般指软件产生的）

#验证指定文件是否被修改过

校验例证：

[root@lchuan ~]rpm -V httpd

.......T. c /etc/httpd/conf/httpd.conf

打印到屏幕上的信息共分为3部分：

第一部分为验证结果；第二部分为文件类型；第三部分为文件位置。

验证结果具体的信息：

s 文件大小是否改变

M 权限是否被改变

5 文件的MD5校验和是否改变（即文件内容）

D 主从设备号是否改变（/dev/）

L 文件路径是否改变

U 文件的属主是否改变

G 文件的属组是否改变

T 文件的修改时间是否改变

文件类型具体分类

c 配置文件

d 普通文件

g “鬼”文件，很少见，该文件不应在RPM包中

l 授权文件

r 描述文件

8） 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，但如果RPM包本身就被动过手脚，那校验就不能解决问题了,我们就必须使用数字证书验证。

数字证书的使用：

* 首先必须找到原厂的公钥文件，然后进行安装
* 然后安装RPM包时，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

数字证书位置

* 1. 光盘中（随出厂镜像）

[root@lchuan ~]#ls -l /mnt/RPM-GPG-KEY-CentOS-6

* 1. 系统中（用镜像安装时安装到系统某一位置处）

[root@lchuan ~]#ls -l /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

数字证书导入

rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

--import 导入数字证书

数字证书查询

rpm –qa |grep gpg-pubkey

#查询系统已安装的数字证书

9） RPM包中提取文件

cpio命令主要是用来建立或者还原备份的工具，cpio命令可以复制文件到归档包中（压缩），或者从归档包中导出文件（解压缩）。

cpio的三种模式：

备份

格式：cpio –ovcB > [文件|设备]

-o：进入copy-out模式（压缩）

-v：显示过程

-c：使用portable format存储方式

-B：设定输入输出块为512bytes

例子：

cpio备份：

find /etc -print |　cpio –ocvB > /root/etc.cpio

#利用find 找出/etc 然后备份到/root下

还原：

格式：cpio –idvcu < [cpio文件]

-i：进入copy-in模式（解压）

-d：恢复到指定位置（视压缩方式恢复）

-v：显示过程

-c：使用portable format存储方式

-u：替换所有文件，无提示

例子：

cpio还原

cpio –idvcu < /root/etc.cpio

注：由于备份时备份的路径使用的是绝对路径，所有恢复时会按照绝对路径恢复回去，所以不会在/root下生成文件。

例子2：

备份：

cd /etc #切换到/etc下

find . –print | cpio -ocvB > /root/etc.cpio

#以相对路径方式进行备份

恢复：

cd /root #切换到/root

mkdir test #创建恢复目录

cd test #进入恢复目录

cpio -idvcu < /root/etc.cpio

#恢复以相对路径备份的etc.cpio文件，内容会被恢复到/root/test/目录下。

10）rpm2cpio 提取RPM软件包中的文件

格式：rpm2cpio 包全名 | cpio –idv ./命令绝对路径

# ./ 是当前目录 + 命令的绝对路径是which查询出的绝对路径（必须）

例子：

假设不小心把系统下的/bin/ls 命令删除了，使用时报错了，需要恢复。

方案：

A、重新安装ls命令，使用--force选项强制执行

B、从相对应的软件包内提取ls 命令

步骤：

* 先确定ls 命令属于哪一个软件包
* 先模拟出ls命令被删除的样子
* 提取RPM包中的ls命令，并恢复到/bin 目录下（站到/目录下进行恢复）

**二进制软件包管理之—yum**

需要解决的问题：

如何让管理命令自己解决查询软件包和依赖关系？

解决方案：

搭建yum仓库（yum源），让管理命令能自己找到软件包所在位置

yum搭建方式：

* + - * 本地光盘源
      * 局域网yum源
      * 互联网yum源

本地光盘源

1）修改位于/etc/yum.repos.d/目录下的配置文件

A．mv CentOS-Base.repo CentOS-Base.repo.bak #改名 B．mount /dev/sr0 /mnt #挂载光盘

C．vi /etc/yum.repos.d/CentOS-Media.repo #修改文件

baseurl=file:///mnt #指定yum源位置

enabled=1 #yum源文件是否生效

gpgcheck=0 #rpm验证是否生效

拓展：

* + - * [base]：容器名称，必须写在 [ ] 中
      * name：容器说明，随便写
      * mirrorlist：镜像站点，忽略或注释掉
      * baseurl：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，如果你觉得慢可以改成你喜欢的yum源地址
      * enabled：此容器是否生效，1 是生效；0 是不生效。
      * gpgchack：数字证书是否生效，1 是生效； 0 是不生效。
      * gpgkey：数字证书公钥存放位置，不用修改。

‘

2）挂载光盘文件

mount -t iso9660 /dev/sr0 /mnt

3）查看是否挂载成功：

df –h #查看Linux中文件系统的占用比例

yum 管理命令

1) yum安装：

yum install [软件名]

yum groupinstall [软件组名]

2） yum卸载：  
 yum remove [软件名]

yum groupremove [软件组名]

1. yum升级

yum update [软件名]

yum groupupdate [软件组名]

4） yum软件包查询

yum list #查询所有已安装和可安装的软件包

yum list 软件名 #查询指定软件是否已安装

yum info 软件名 #查询指定软件包的信息

yum grouplist 软件组名 #查询所有已安装和可安装的软件组

yum groupinfo 软件组名 #列出软件组中包含的软件

例：“yum groupinfo Web Server”

yum search <关键词> #查询中所有和关键词相关的包

5）yum其他命令

yum clean all #清除下载的临时软件包和缓存

yum makecache #建立软件包缓存，提高查询速度

yum update #更新所有需要更新的软件&内核

拓展：

实验：（用到两张光盘，学会挂载两张甚至更多的光盘）

通过yum为纯字符系统，安装图形化桌面

* yum -y install xorg\*
* yum -y groupinstall "X Window System"
* yum -y groupinstall Desktop
* yum -y groupinstall "Chinese Support"
* startx #启动图形界面

**源代码软件包**

特点：

最新的软件版本

根据用户需要，灵活定制软件功能

应用场合举例：

安装较新版本的应用程序时

当二进制软件无法满足需要时

编译安装可由用户自行修改、定制功能

需要为应用程序添加新的功能时

准备环境：

yum 仓库

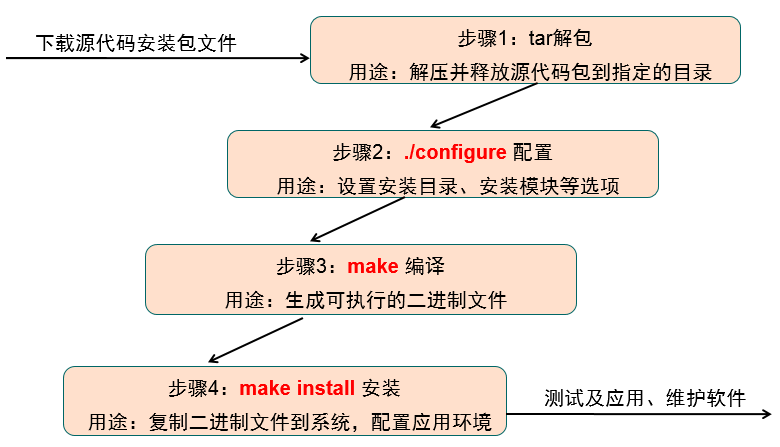
确认源代码编译环境

需安装支持 C/C++程序语言的 编译器

gcc-4.1.1-52.el5、gcc-c++-4.1.1-52.el5

make-3.81-1.1.i386 ……

安装步骤：



补丁文件的生成和使用

命令格式：diff 选项 old new

比较old和new文档的不同

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

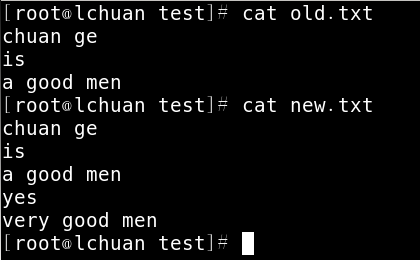
-N 当比较目录时，若某个文件只在一个目录中，则另一个目录中视作空文件

-r 当比较目录时，递归比较子目录

-u 使用同一的输出格式

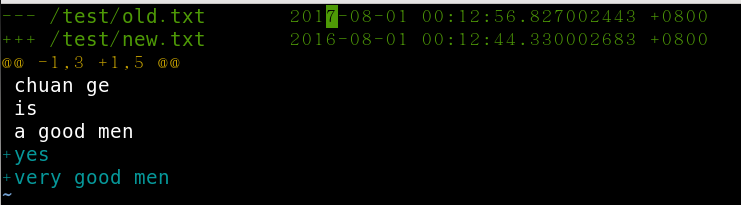
实验：

分别创建old 和 new 两个文本文件，有相同的内容，也有不同的内容，形成对比。、



#创建两个文件，内容要有差别

diff -Naur /test/old.txt /test/new.txt > ./txt.patch



打开生成的txt.patch，会发现new 比 old 多出来的两行被单独标记出来（用+号）

打入补丁

命令格式：patch -pn < 补丁文件

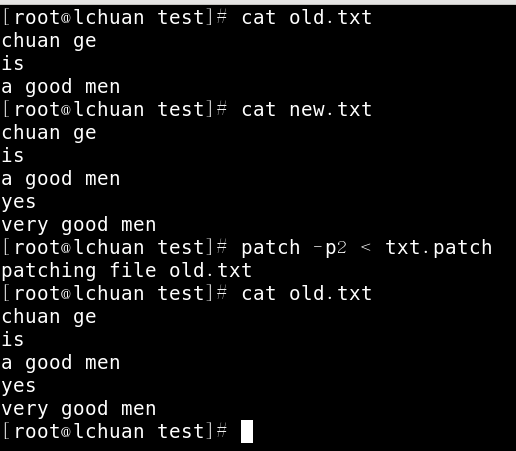
-p：设置要剥离的目录层数

n：代表补丁文件中记录的文件（old.txt）所在目录层数，用于更新old.txt文 件时匹配正确路径

实验：

patch -p2 < txt.patch

#按照补丁文件对old.txt 进行更新



**附带安装程序的源代码软件**

webmin安装

文件导入

解压，执行 setup.sh （按要求选择）

安装位置

日志位置

perl安装位置

端口

管理员名称

管理员密码

ssl： 不启用

开启启动

函数库分类：

静态函数库

函数库文件一般以“\*.a”扩展名结尾。这种函数库在被程序调用时会被直接整合 到程序当中

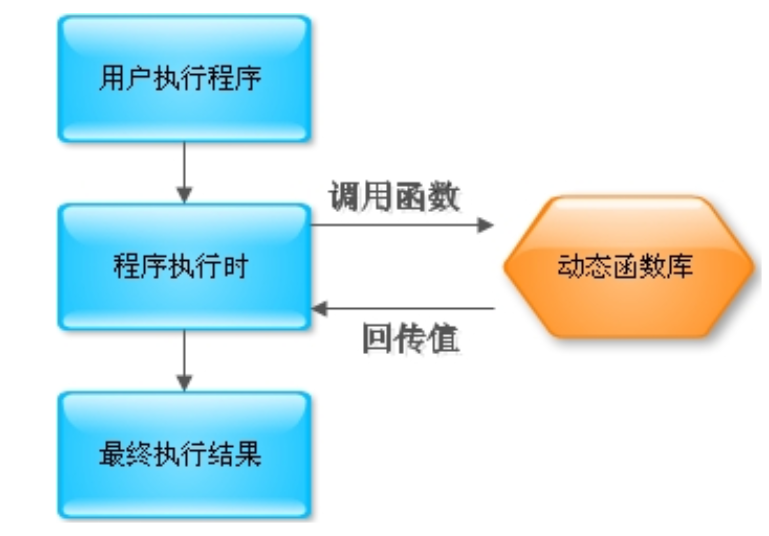
优点：程序执行时，不需要再调用外部数据，可以直接执行

缺点：因为把所有内容都整合到程序中，所以编译生成的文件会比较大升级比较困难，需要把整个程序都从新编译

动态函数库

函数库文件通常以“\*.so”扩展名结尾。这种函数库被程序调用时，并没有直接整合到程序当中，当程序需要用到函数库的功能时，再去读取函数库。在程序中只保存了函数库的指向。

如图所示：



函数库管理：

ldd 可执行文件名

-v 显示详细版本信息

ldd /bin/ls



函数库一般可以直接在网上下载到，那么应该如何安装呢？

一般情况下函数库文件的存放位置是在/usr/lib/ 或 /lib 下；然后会在/etc/ld.so.conf 配置文件中记录 这两个目录或者新增加的目录，这个配置文件中只记录函数库存放目录，不记录实际函数库名称。

1. cp \*.so /usr/lib/

2. 由于/usr/lib/已经被/etc/ld.so.conf配置文件记录所以不需要重写

3. 假有新的目录需要改写到/etc/ld.so.conf配置文件中，写完后需要刷新下缓存

ldconfig 刷新函数库配置文件缓存

-p 列出缓存中识别的所有的函数库

补充：

date #查看系统时间

date -s 20130220 #设定日期

date -s 09:30:00 #设定时间

du -sh 目录名 #统计目录大小

-s 和

-h 常见单位