<https://blog.csdn.net/astraylinux/article/details/7039744>

**1. dpkg描述**

     dpkg是一个安装、创建、移除和管理debian包的工具。更多使用界面友好的dpkg的前端"dselect",。dpkg完全受控于命令行参数，和零个或更多参数正确的组成一个操作方式。参数告诉dpkg要做什么和多种选项控制不同的行为。dpkg不仅提供了大量的参数选项, 同时也提供了许多子命令比如:  
dpkg-deb  
dpkg-divert  
dpkg-query  
dpkg-split  
dpkg-statoverride  
start-stop-daemon  
  
**2. deb软件包命名规则:**  
Debian软件包命名遵循下列约定: \_-\_.deb  
    \* 软件包名称(Package Name):  
    \* 版本(Version Number):  
    \* 修订号(Build Number):  
    \* 平台(Architecture):  
          o i386  
          o all: 平台无关. 即适用于所有平台.比如文本, 网页, 图片, 媒体, pdf 等.  
例如  
    \* nano\_1.3.10-2\_i386.deb  
          o 软件包名称: nano  
          o 版本: 1.3.10  
          o 修订号: 2  
          o 平台: i386  
  
**3. 软件包处理**  
    dpkg可用作dpkg-deb的前端，处理本地Debian软件包. 可以提取从.deb文件中提取软件包信息和数据. 详细信息,请参阅 dpkg-deb 的联机手册.以下是dpkg-deb操作方式：  
    -b, --build   //dpkg-deb -b nano nano\_1.3.10-2\_ill.deb  重新打包  
    -c, --contents //dpkg-deb -c nano\_1.3.10-2\_i386.deb  |more  列出软件包的内容  
    -I, --info  //dpkg-deb -I nano\_1.3.10-2\_i386.deb  查看软件包的信息  
    -f, --field  //重新构建软件包,比如nano(1.3.10)需要满足以下依赖关系:$ dpkg-deb -f nano\_1.3.10-2\_i386.deb depends     depends 为对应查询段)  
    -e, --control  //dpkg-deb -e ./nano\_1.3.10-2\_i386.deb ./nano/DEBIAN  解压控制文件  
    -x, --extract  //dpkg-deb -x ./nano\_1.3.10-2\_i386.deb ./nano   解压程序文件  
    -X, --vextract, and --fsystarfile.//dpkg-deb -X ./nano\_1.3.10-2\_i386.deb ./nano   解压程序文件并列出软件包内容。

**4. 数据库查询**

当前版本的 dpkg 使用文本文件作为数据库来维护系统中软件，包括文件清单, 依赖关系, 软件状态, 等等详细的内容,通常在 /var/lib/dpkg 目录下。 通常在 status 文件中存储软件状态和控制信息。 在 info/ 目录下备份控制文件， 并在其下的 .list 文件中记录安装文件清单， 其下的 .mdasums 保存文件的 MD5 编码。

体验使用数据库：

$ dpkg -l  
Desired=Unknown/Install/Remove/Purge/Hold  
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed  
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)  
||/ Name       Version           Description  
+++-===========-================-========================================  
ii  aalib1     1.4p5-28          ascii art library - transitional package  
ii  adduser    3.85              Add and remove users and groups  
ii  alien      .63               install non-native packages with dpkg  
... ...  
每条记录对应一个软件包, 注意每条记录的第一, 二, 三个字符. 这就是软件包的状态标识, 后边依此是软件包名称, 版本号, 和简单描述.  
    \* 第一字符为期望值,它包括:  
          o u 状态未知,这意味着软件包未安装,并且用户也未发出安装请求.  
          o i 用户请求安装软件包.  
          o r 用户请求卸载软件包.  
          o p 用户请求清除软件包.  
          o h 用户请求保持软件包版本锁定.  
    \* 第二列,是软件包的当前状态.此列包括软件包的六种状态.  
          o n 软件包未安装.  
          o i 软件包安装并完成配置.  
          o c 软件包以前安装过,现在删除了,但是它的配置文件还留在系统中.  
          o u 软件包被解包,但还未配置.  
          o f 试图配置软件包,但是失败了.  
          o h 软件包安装,但是但是没有成功.  
    \* 第三列标识错误状态,可以总结为四种状态. 第一种状态标识没有问题,为空. 其它三种符号则标识相应问题.  
          o h 软件包被强制保持,因为有其它软件包依赖需求,无法升级.  
          o r 软件包被破坏,可能需要重新安装才能正常使用(包括删除).  
          o x 软包件被破坏,并且被强制保持.

也可以以统配符模式进行模糊查询, 比如我要查找以nano字符开始的所有软件包:

$ dpkg -l nano\*  
Desired=Unknown/Install/Remove/Purge/Hold  
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed  
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)  
||/ Name           Version        Description  
+++-==============-==============-============================================  
ii  nano           1.3.10-2       free Pico clone with some new features  
pn  nano-tiny               (no description available)

un  nanoblogger             (no description available)

以上状态说明: 系统中安装了 nano 版本为 1.3.10-2;安装过 nano-tiny, 后来又清除了; 从未安装过nanoblogger.  
如果觉得 dpkg 的参数过多, 不利于记忆的话, 完全可以使用 dpkg-query 进行 dpkg 数据库查询.  
使用 dpkg-query 进行 dpkg 数据库查询.  
应用范例:  
查询系统中属于nano的文件:  
$ dpkg --listfiles nano  
or  $ dpkg-query -L nano  
查看软件nano的详细信息:  
$ dpkg -s nano  
or  $ dpkg-query -s nano  
查看系统中软件包状态, 支持模糊查询:  
$ dpkg -l  
or  $dpkg-query -l  
查看某个文件的归属包:  
$ dpkg-query -S nano

or  $ dpkg -S nano

**5. 安装软件包**  
dpkg -i | --install package\_file...  
例如：$dpkg -i ./nano\_1.3.10-2\_i386.deb  
如果指定--recursive 或 -R 选项，必须指定一个目录代替包文件。  
安装步骤如下：  
1). 导出新包的控制文件。                 
2). 如果相同包的另一个版本在新包安装前已经被安装，执行旧包的prerm脚本程序。  
3). 如果由包提供，运行postinst脚本。  
4). 解压新文件同时备份旧文件，如果遇到问题以便修复。  
5). 如果另一个版本的同一个包在新包安装前已经安装，执行旧包的postrm脚本。注意：执行postrm 脚本后执行新文件的preint脚本，因为写新文件的同时移除旧文件。  
6). 配置软件包。  
或者  
$dpkg-deb -e ./nano\_1.3.10-2\_i386.deb  
当使用 dpkg 安装软件包时, 主要分为两个阶段: 首先解包; 然后运行postinst控制脚本(如果有的话),这就是所谓的配置阶段. 当完成软件包当解包后, dpkg 的--install选项自动调用配置阶段. 注意, 操作时应当指定 .deb 文件的实际路径, 仅仅给出文件名无法找到对应文件.  
这样, 软件的安装就可被拆分为两个对立的过程:  
    \* dpkg --unpack    (解包)  
    \* dpkg --configure   (配置)  
解包  
运行  
#dpkg --unpack nano\_1.3.10-2\_i386.deb  
  
然后  
$ dpkg -l nano  
iU  nano           1.3.10-2       free Pico clone with some new features  
说明此时系统中安装了nano 包, 但是未经配置.  
解包可能包括包以下步骤:  
    \* 将控制文件解压到临时目录.  
    \* 运行 preinst(如果有的话). 通常为停止相关服务.  
    \* 解压配置文件到 /etc 目录, 并添加.dpkg-new 后缀, 以防止冲突.  
    \* 解压 data.tar.gz 到系统根目录,(--root=dir).  
    \* 将控制文件解压到 /var/lib/dpkg/info 目录并添加软件名前缀. 对数据库进行更新.  
    \* 数据库中将软件包的状态设置为"unpacked".  
仅仅解包, 并不能确保软件可以正常运行.  
配置  
#dpkg --configure nano  
  
$ dpkg -l nano  
ii  nano           1.3.10-2       free Pico clone with some new features  
如输出所示, nano 已经正常安装.  
软件解包后, 还需要对其进行调整, 比如根据实际情况对配置文件进行修改. 这一步通常是在安装过程中自动进行的, 当然可以手动调用, 这次用到的是软件名, 而不是deb文件名.  
通过命令“ dpkg -L nano "可以得以验证  
这一过程可能包括以下步骤:  
    \* 询问对配置文件的处理方法, 覆盖当前版本, 还是不做处理.(当软件升级时会碰到这种情况)  
    \* 运行 postinst 脚本  
    \* 将软件包标记为 "installed"

如果想再次修改配置选项, 可以运行:

    #dpkg-reconfigure nano

**6. 软件卸载**  
dpkg -r | --remove | -P | --purge package ... | -a | --pending  
在Debian中卸载和清除软件包是两个不同的概念. 不同之处在于软件包被删除(卸载)后,它的配置文件仍会留在系统中,只有清除时才会删除它们. 默认情况下, Debian 仅会做删除操作, 除非你明确指出, 才会将配置文件删除. 如果要清除软件包, 则在清除前将会隐含地执行删除操作.  
要删除一个软件包,dpkg需要使用--remove选项将软件包卸载.与安装不同,删除只需要软件包名,而不是实际的deb文件名.  
删除  
    #dpkg -r nano  
    \* 首先运行 prerm 脚本(如果有的话).  
    \* 然后, 保留配置文件, 其他文件全部删除.  
    \* 删除 /var/lib/dpkg/info 目录下软件包的所有配置文件, 保留postrm和list文件.  
    \* 最后, dpkg 在数据库中将软件状态修改为删除.  
运行  
$dpkg -l nano  
rc  nano    1.3.10-2     free Pico clone with some new features  
清除  
-P or --purge 删除所有的包括配置文件。 (主要删除的是/var/lib/dpkg/status中指定文件，)如果给出的是-a或者--pending而不是包的名字，那么所有的包都被打开（解压缩），但是在/var/bin/dpkg/status文件中标记被移除或清除的，将会分别被移除或清除（但是在 /var/bin/dpkg/status文件中标记“要”移除）  
    #dpkg -P nano  
    \* 删除所有的文件. 所有的文件都被解除了（从系统中删除了）.  
    \* 运行postrm脚本.如果还存在,运行postrm脚本.  
    \* 删除 postrm 和 list 文件.保留的两个文件,从系统中解除postfix.postrm和postfix.files.  
    \* 在dpkg的数据库中将软件包标记为未安装  
$dpkg -l nano  
pn  nano                                          (no description available)  
注意输出的状态: pn 这意味着软件包实际上是被清除了,但是在数据库中还有这个软件包的记录.因此,pn 明确指出以前安装过 nano, 在软件包数据库中出现过,并且现在已被完全清除.  
附加：  
dpkg-reconfigure  
重新配制一个已经安装的包裹,如果它使用的是 debconf (debconf 为包裹安装提供了一个统一的配制界面).你能够重新配制 debconf 它本身,如你想改变它的前端或提问的优先权.例如,重新配制 debconf ,使用一个 dialog 前端,简单运行:  
dpkg-reconfigure --frontend=dialog debconf (如果你安装时选错了,这里可以改回来哟  
echo " hold" | dpkg --set-selections  
设置  的状态为 hlod (命令行方式)  
dpkg --get-selections ""  
取的  的当前状态 (命令行方式)

支持通配符,如:

Debian:~# dpkg --get-selections \*wine\*  
libwine hold  
libwine-alsa hold  
libwine-arts hold  
libwine-dev hold  
libwine-nas hold  
libwine-print hold  
libwine-twain hold  
wine hold  
wine+ hold  
wine-doc hold  
wine-utils hold  
dpkg-source -x  
如果你手工下载了一个程序的源码包，其中包含了几个类似 .orig.tar.gz , .dsc ,  
以及 .diff.gz 之类的文件，  
那么你就可以对 .dsc 文件使用这个命令来 unpack 源码包。  
dpkg-buildpackage  
从 Debian 源码树建立一个deb包。你必须在source tree的主目录才能生效。例如：  
dpkg-buildpackage -rfakeroot -uc -b  
这里 '-rfakeroot' 指定命令使用 fakeroot 程序来模仿 root 权限 (来实现所有者(ownership)目的)，  
'-uc' 表示 "Don't cryptographically sign the changelog", '-b' 代表只建立二进制包.  
debuild  
一个快速打包脚本类似 dpkg-buildpackage ,能自动的识别是否使用 fakeroot,  
同时为你运行 lintian 和 gpg  
修正倚赖关系  
dpkg --configure --pending  
如果dpkg在apt-get install upgrade dist-uptradeing 的时候出错退出，  
尝试使用此命令来配置已经unpack的包。  
然后再用 apt-get install ，upgrade, or dist-upgrade -f ，  
然后再用 apt-get install, upgrade, or dist-upgrade.  
可能会重复多次，这样通常可以解决大多数的依赖性问题。  
(同时,如果提示由于某种原因需要某个特定的包裹,你可以常识安装或卸载这个包)  
apt-get install -f  
apt-get upgrade -f  
apt-get dist-upgrade -f  
尝试修正上述过程中出现依赖性关系  
注意 apt-get install -f 不需要  作为参数。