yum使用总结

yum [options] [command] [package ...]

[功能]

center os下面的软件包管理工具。

[描述]

yum是一个基于rpm的交互的软件包管理工具。

1，常用的command(子命令)简介

install package1 [package2] [...]

安装最新版本的软件包或者一组软件包，同时确保满足依赖关系。如果没有和指定名称匹配的软件包，那么会视为shell的一个部分并且将任何的匹配安装上去。如果名字以@开始，那么后面的名字就直接像是传递给了groupinstall。如果名字是一个文件，那么工作的方式就类似localinstall，注意可以指定统配符号以及文件列表。(具体参见man手册)

update [package1] [package2] [...]

如果运行的时候，没指定任何package，那么update子命令将会更新当前所有被安装的软件包，否则会更新列出的软件包。更新的时候，yum要保证所有的依赖关系被满足，如果指定的软件包当前没有被安装，那么update不会安装它们。

upgrade [package1] [package2] [...]

这个命令和使用了--obsoletes的update命令一样。

list [...]

列出可用软件包的各种信息。

remove | erase package1 [package2] [...]

用来从系统卸载软件包，并且根据被卸载的软件包，卸载其他相关的软件包。命令格式类似install。

groupinfo

用于给出一个组的描述以及软件包列表。如果加上-v选项，那么会输出已经安装和可用的类似list命令的软件包的名称。

groupupdate

当软件包X已经被安装的情况下，"yum update X"和"yum install X"所做的工作一样。

groupinstall

用来安装一个组中的指定类型的所有的独立的软件(这个工作的方式类似于你把这些软件包的名字放到"yum install"命令中进行运行).配置选项group\_package\_types指定了需要安装什么类型。

groupremove

用来将一个组中的所有软件包删除，和"groupinstall"不一样，这个将会忽略group\_package\_types，并且将所有的内容移走。如有必要，应该指出软件包可以在多个组中，这样跟在"groupremove Y"后面的"groupinstall X Y"不会给你一个和"groupinstall X"一样的执行结果。

clean [ packages | headers | metadata | dbcache | all ]

用来清除随着时间逐渐增多的yum目录缓存中的内容。这个命令还有更多的选项。

makecache

用于从当前激活的yum软件仓库中下载可用的元数据（元数据在安装软件的时候会被用来，应该包含了软件包的下载位置等）。

localinstall rpmfile1 [rpmfile2] [...]

用于安装一系列本地的rpm文件，如果需要，那么被激活的repositories将会用来解决一些依赖。注意给定名字之后，安装的命令会做一些本地的安装。注意有时我们安装的时候会提示签名，否则不让安装，那么我们可以指定"--nogpgcheck"选项，这样就不用签名了。

search string1 [string2] [...]

用来通过一个部分的名字寻找一个软件包，其中这个软件rpm的描述，概要或者软件包名称中包含传入的string字符串。

2，常用的command选项

下面大多数的命令项可以通过配置文件(例如yum.conf)进行设置，并且也给出了使用配置文件设置时候的配置选项。

-y 假设为yes.假设所有的问题的回答都是yes。配置的选项：assumeyes.

-c [config file] 指定配置文件的位置，可以是http,ftp的url，或者是本地的文件路径。

--installroot=root 指定一个可选的安装路径，所有的软件包都是基于这个路径被安装的。配置的选项为：installroot.

--enablerepo=repoidglob 激活特定的，通过id或者glob指定的，但是在配置文件中使用enable=0选项被禁止了的软件仓库。配置的选项为：enabled.

--disablerepo=repoidglob 禁止一个通过id或者glob指定的特定的软件仓库。配置的选项为：enabled.

-x, --exclude=package 通过指定名字，排除软件仓库中更新的软件。配置的选项为：exclude.

--nogpgcheck 运行的时候，不进行gpg签名检查。配置选项为：gpgcheck.

--noplugins 运行的时候不使用插件。配置的选项为：plugins.

3，一些子命令的常用选项

1)list命令的选项：

下面的项指定了install命令的运行方式。注意这里的list出来的信息包含了软件包的版本信息。

yum list [all | glob\_exp1] [glob\_exp2] [...]

列出所可用的和安装的软件包。

yum list available [glob\_exp1] [...]

列出所yum仓库中可以被安装的软件包。

yum list updates [glob\_exp1] [...]

列出所yum仓库中可以被更新的软件包。

yum list installed [glob\_exp1] [...]

List the packages specified by args.  If an argument does not match the name of an  avail-

able package, it is assumed to be a shell-style glob and any matches are printed.

列出args指定的软件包。如果参数没有匹配一个可用的软件包的名称，那么...

yum list extras [glob\_exp1] [...]

列出系统中被安装的，但是在yum配置文件仓库中没有列出的软件包。

yum list obsoletes [glob\_exp1] [...]

列出系统中已经安装的，在配置文件中的yum仓库列表被废弃的软件包。

yum list recent

列出最近添加到仓库中的软件包。

2)clean命令的选项：

下面指定你的clean命令的运行方式。这里命令中的“所有文件”实际就是当前被激活的软件仓库中的所有文件。如果你临时想要清除其他非激活的软件仓库中的内容，那么使用 --enablerepo='\*'选项。

yum clean expire-cache

当元数据和每个软件仓库中的镜像列表下载的时候，删除本地的数据。也就是说，yum将会为每一个软件仓库刷新缓存。下次会使用到。但是如果缓存的数据仍然有用，那么不会删除任何重要的数据。

yum clean packages

从系统中删除任何缓存的软件包。我们需要注意的是当我们下载软件包之后，软件包并不会被自动地删除。

yum clean headers

删除所的头文件，这些头文件用于yum解决软件的以来关系。

yum clean metadata

删除yum用于确定软件包可用的一些文件（元数据文件）。使用这个选项会强制让yum下次运行的时候下载所有的元数据。

yum clean dbcache

删除sqlite缓存，这个缓存用来以很快的速度访问原数据。使用这个选项将会使得yum在下次运行的时候重新创建缓存文件。

yum clean all

删除上面说过的所的内容。

4，关于插件

你可以通过插件来进行扩展。一个插件就是一个".py"的python脚本文件，这个文件会被安装到一个通过yum.conf的pluginpath选项指定的目录。如果想要一个插件工作，必须做如下的工作：

1)插件模块文件必须被安装在插件的目录（如前所述）。

2）/etc/yum/yum.conf中的全局插件选项必须被设置为1。

3)必须存在一个/etc/yum/pluginconf.d/<plugin\_name>.conf文件，并且这个文件中的enabled必须被设置。例如如下就是这样的配置文件的一个非常简单的例子：

[main]

enabled = 1

可以通过"man 5 yum.conf"查看关于插件配置选项更多的信息。

5，配置文件

yum使用/etc/yum/yum.conf配置文件。可以通过指定reposdir选项（默认值为/etc/yum/repos.d），读取其目录中的其他配置文件。具体参见man手册中的详细内容。

在yum配置文件中有两个类型的节：main和repository。Main定义的是全局的配置选项。一个时刻只能有一个main节。repository节定义了每个repository/server的配置选项，可能会有多个repository节。

1)main节：

具体关于main节所支持的选项参数可以参见"man yum.conf"中的手册说明。

下面是一个main的简单例子：

############文件来自/etc/yum.conf##############

[main]

#yum的缓存和数据库文件的存放路径，默认为"/var/cache/yum"

cachedir=/var/cache/yum

#是否在成功安装之后保持头和软件包的缓存。默认为1，表示保持。

keepcache=0

#yum的日志文件的完整路径

logfile=/var/log/yum.log

#对于软件包yum是否在命令行上面提示错误，例如已经安装一个安装的软件。默认为0。

tolerant=1

#表示是否对软件包进行签名验证，默认为0。

gpgcheck=1

#是否使用插件，默认为0即不使用。

plugins=1

#yum使用的代理服务器，可以为proxy。

Proxy=http://<user>:<passwd>@<poxyIP>:<port>

############文件来自/etc/yum.conf##############

2)repository节:

repository节的内容形式如下：

Example: [repositoryid]

name=Some name for this repository

baseurl=url://path/to/repository/

这里，

repositoryid是一个单词，唯一的名字，表示每个软件仓库。

name是一个描述软件仓库的字符串。

baseurl是一个目录的URL，这个目录包含相应的yum软件仓库中的数据。形式可以是"http://","ftp://",或者"file://"，你也可以在一个baseurl中指定多个URL。类似如下：

[repositoryid]

name=Some name for this repository

baseurl=url://server1/path/to/repository/

url://server2/path/to/repository/

url://server3/path/to/repository/

如果你在一个软件仓库上列举出来多余"baseurl="，那么你将会发现yum会忽略前面的，并且表现很奇怪。你可以在baseurl行给服务器追加"user:password@"前缀使用包含权限的http。例如:"baseurl=http://user:passwd@example.com/"

具体关于repository节所支持的选项参数可以参见"man yum.conf"中的手册说明。

下面是两个repository的简单例子：

############文件来自/etc/yum.repos.d/CentOS-Base.repo##############

#additional packages that may be useful

[extras]

name=CentOS-$releasever - Extras

mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=extras

baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5

#additional packages that extend functionality of existing packages

[centosplus]

name=CentOS-$releasever - Plus

mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=centosplus

#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$basearch/

gpgcheck=1

enabled=0

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5

############文件来自/etc/yum.repos.d/CentOS-Base.repo##############

另外，yum的配置文件还支持include和默认的一些环境变量，具体参见"man yum.conf"手册的输出。

[举例]

这里，给出常用的一些命令。

\*列出所有可用的和安装的软件包:

#yum list

\*列出所有安装的软件包：

#yum list installed

\*列出所有安装的软件包不使用插件：

#yum list installed --noplugins

这里，时使用插件会感觉很慢，这个时候可以使用--noplugins表示运行yum的时候不运行插件。

\*列出所有可以被更新的软件包：

#yum list updates

\*查找包含"meld"的软件软件包：

# yum search meld

输入之后，输出可能会包含如下信息：

...省略前面...

Trying other mirror.

http://mirror01.idc.hinet.net/CentOS/5.6/updates/i386/repodata/repomd.xml: [Errno 14] HTTP Error 407: authenticationrequired

Trying other mirror.

http://mirrors.ta139.com/centos/5.6/updates/i386/repodata/repomd.xml: [Errno 14] HTTP Error 407: authenticationrequired

Trying other mirror.

================================================== Matched: meld ===================================================

meld.noarch : Graphical visual diff and merge tool

这里，搜索完了之后，就可以使用"yum install meld.noarch"来进行安装了。

\*安装一个软件：

#yum install meld.noarch

这样，会安装软件仓库里面的meld.noarch软件。

\*使用yum安装本地的一个软件：

# yum localinstall mind-2.1-1.i386.rpm --noplugin --nogpgcheck

或# yum install mind-2.1-1.i386.rpm --nogpgcheck --noplugin

这里，注意使用--noplugin是为了运行yum的时候不加载欧额外插件，这样会快一些；另外，如果安装的时候提示没有gpg签名就无法安装这个rpm文件，这个时候使用--nogpgcheck选项就行了。安装之后，可以使用"yum list installed |grep mind"查询到刚刚安装的软件，输出类似如下:

mind.i386                                      2.1-1                   installed

使用"rpm -qa |grep mind"也能够查询到。另外，使用install的时候如果直接指定文件名称，也会安装，效果和localinstall类似。安装的时候会自动根据软件仓库解决依赖关系的。

\*升级一个软件：

#yum update mind-2.1-2.i386.rpm --noplugin --nogpgcheck

这里，使用文件名称也会升级本地的软件。本地个rpm文件包。升级实际就是先卸掉原来的软件再安装新的。

\*卸载一个软件：

#yum remove mind.i386

这样会卸载软件。

[其它]

yum与rpm的区别：

YUM是从软件仓库自动下载，自动安装，自动配置的。

RPM是要你自己下载软件包以及相关的包，之后再依次用rpm命令装完各个包。

参考资料：

"man yum"

"man 5 yum.conf"

以上是使用center os上面的yum软件包工具的一些总结，有问题和建议请联系。^\_^

作者：QuietHeart

Email:quiet\_heart000@126.com

日期：2011年7月27日