Linux基础

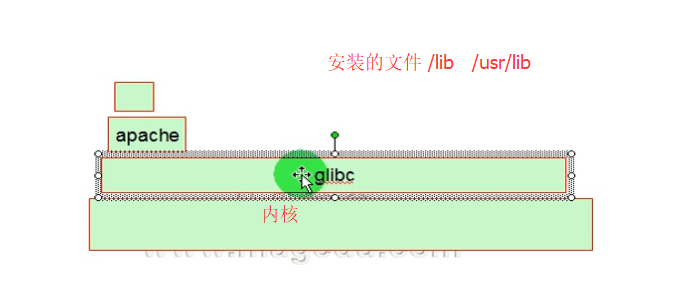
## 程序的组成部分

**二进制程序**

**库**

**配置文件**

**帮助文件**



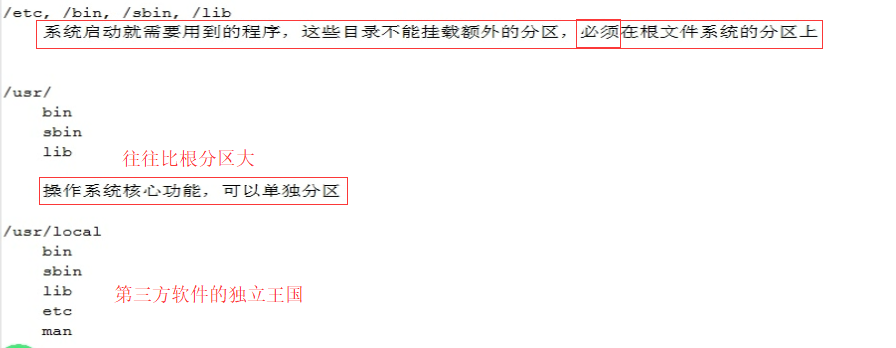
/etc 配置文件

/bin 二进制文件

/sbin 二进制文件

/lib 库

/usr/share/man 帮助文件



/etc , /bin ,/sbin , /lib 系统启动需要用到的程序

/usr/bin ,/usr/sbin ,/usr/lib 操作系统的核心功能,可以单独分区

他们保证了系统的正常启动和运行

/opt 早期安装第三方软件的地方,现在一般安装在/usr/local下

/proc 内核

/sys

伪文件系统

不能单独分区,默认为空,

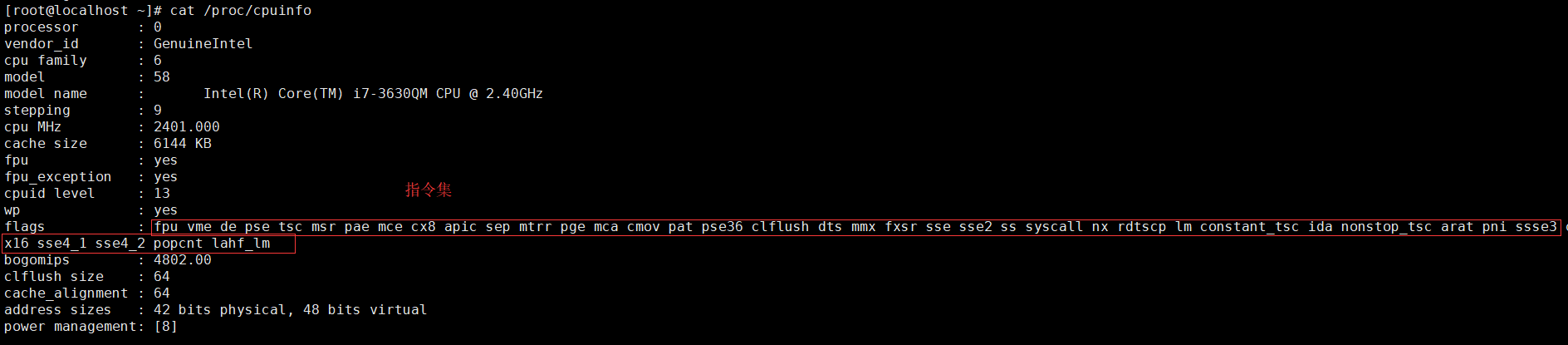
/dev 设备,不能单独分区

udev,内核2.6以后就可以自动设备创建文件,内核识别设备靠的是驱动程序

/root 不能单独分区

/var 日志文件,运行过程中的信息,会越来越大,可以单独分区

/boot 内核, initrd(initramfs),应该单独分区



根分区不能使用LVM文件系统,LVM要内核启动后,才能使用

## 软件包管理器

软件包管理器本身也是一个软件

目前主流的有3个发行商

**Redhat** RPM (Redhat Package Manager) RPM is Package Manager 模仿dpt开发

**suse** RPM 因为suse的软件组织位置管理不同,所以尽管都是RPM,但是红帽的包不能在suse中使用

**debian** dpt

RPM Linux界软件包管理的工业标准,被众多发行商所使用

功能

1. 制作软件包
2. 打包 **二进制文件** **库文件** **配置文件** **帮助文件** 成为一个文件
3. 生成数据块,追踪记录所安装的每一个文件的位置
4. 安装, 卸载, 升级, 查询, 校验

前端工具: yum(yellowdog红帽的) , apt-get

yum: yellowdog update modifier

后端工具: RPM , dpt

## rpm

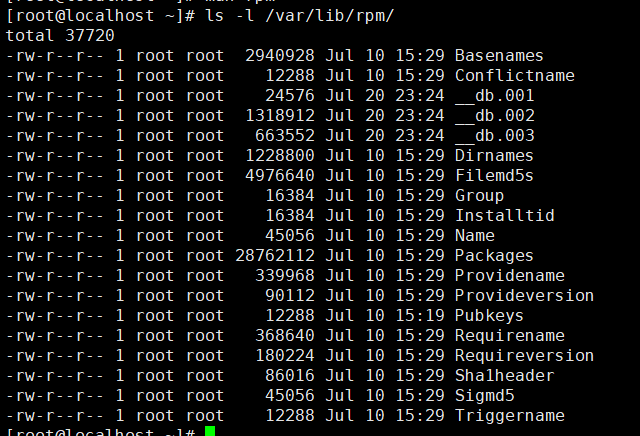
rpm :

数据库 /var/lib/rpm

rpmbuild

安装,查询,卸载,升级,校验,数据库的重建,验证软件数据包等工作

数据库都是以hash编码的格式,查找快



rpm命名

包:组成部分

主包:

bind-9.7.1-1.el5.i586.rpm

子包:

bind-libs-9.7.1-1.el5.i586.rpm

bind-utils-9.7.1-1.el5.i586.rpm

报名格式:

name-version-release.arch.rpm

rpm格式

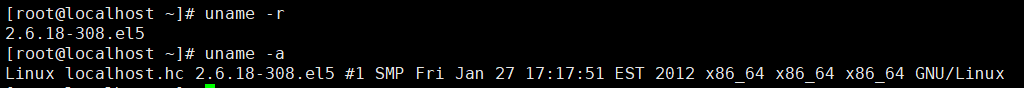
二进制格式 : 发行的时候就编译好了,不知道是根据哪个平台编译的,应该具有通用性,

源码格式 : 安装的时候先要针对平台对源码进行编译,编译成二进制格式,但是麻烦,能发挥平台的最佳性能

查看操作系统的版本

uname –a

uname -r



### 安装

rpm

-i /xxx/xxx.rpm

-h: 以#号显示进度,每个#表示2%

-v: 显示详细过程

-vv: 更详细的过程

rpm –ivh /xxx/xxx.rpm

--nodeps 忽略依赖关系 ,如果有依赖关系装上去不能正常使用

--replacepkgs 重新安装,替换原有的安装

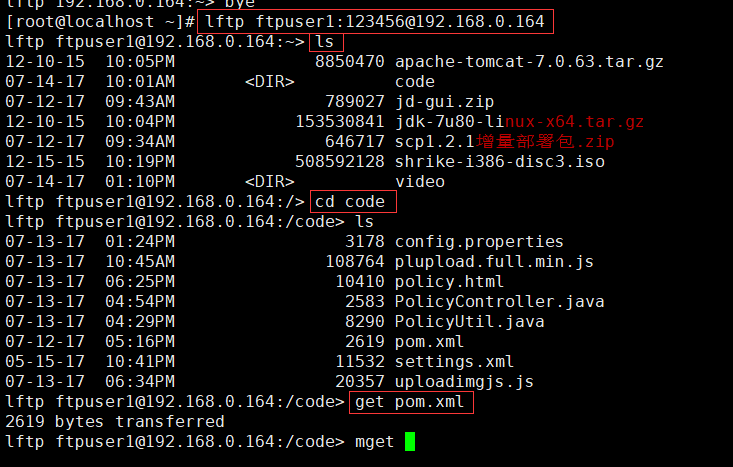
--replacefiles 替换文件

--oldpackage 降级

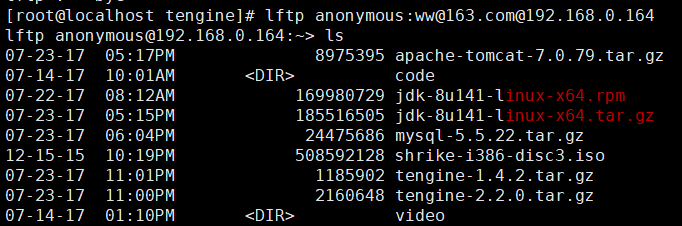
--force 强行安装或者降级

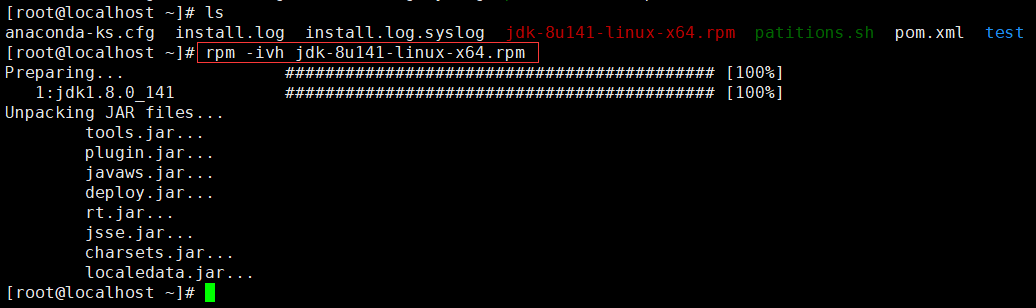
使用xshell 在 ftp服务器上获取文件

lftp ftpuser1:123456@192.168.0.164



匿名用户

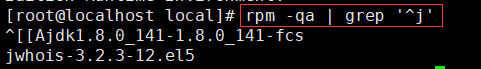




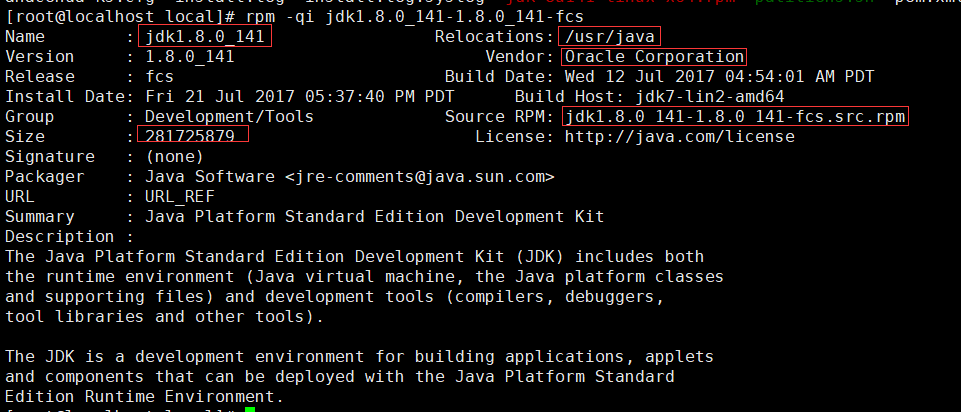
### 查询

rpm –qa :查看查询已经安装的所有包

rpm –q packgename 查询指定

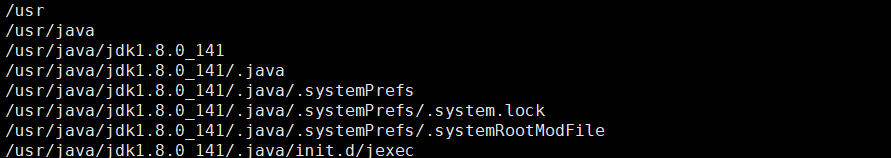


rpm –qi packagename 查询指定包的说明信息

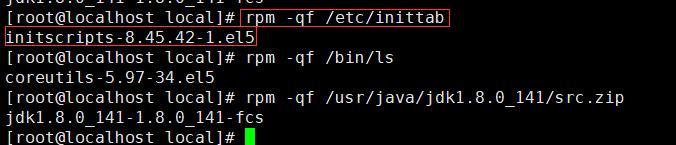


rpm –ql packagename 查询指定包安装后生成的文件列表

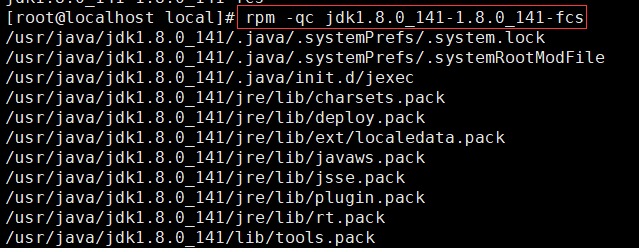




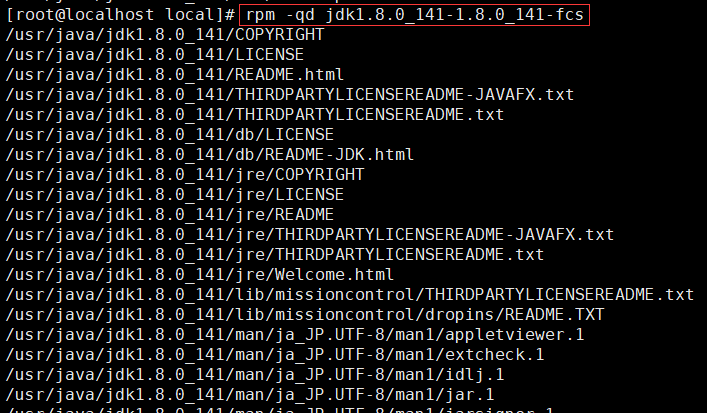
rpm –qf /xxx/xxx 查询指定文件是有哪个rpm包安装生成的



rpm –qc packagename 查询rpm包的安装后的配置文件



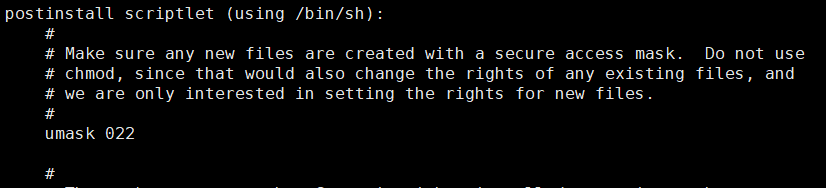
rpm –qd packagename 查询rpm包的安装的帮助文件



rpm –q --scripts packagename 查询指定rpm安装包的脚本

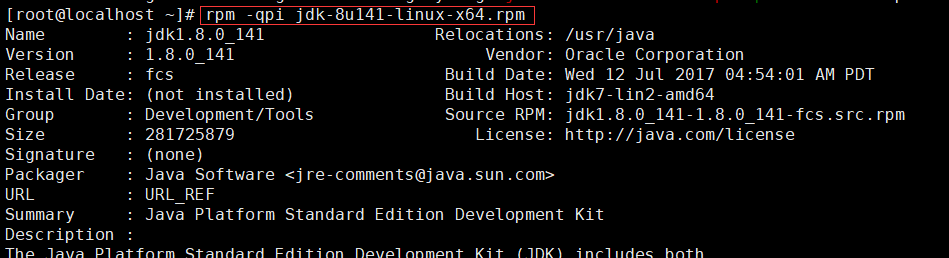


安装前,安装后,卸载前,卸载后…使用的脚本



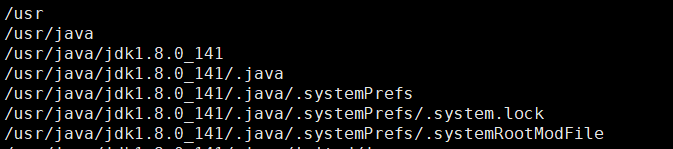
如果某rpm包尚未安装,我们需查询其说明信息,安装以后会生成的文件列表

rpm –qpi /xxx/xxx.rpm 说明信息



rpm –qpl /xxx/xxx.rpm 安装以后会生成的文件列表



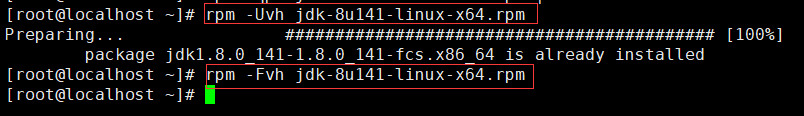


### 升级

rpm –Uvh /xxx/xxx.rpm 如果装有老版本,则升级,否则安装

rpm –Fvh /xxx/xxx.rpm 如果装有老版本,则升级,否则退出

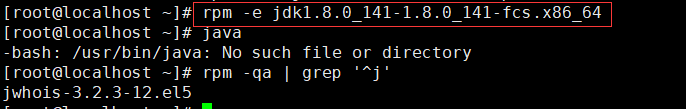
--oldpackage 降级



### 卸载

rpm –e packagename

--nodeps



### 校验

rpm –v /xxx/xxx 是否被改变,丢失等

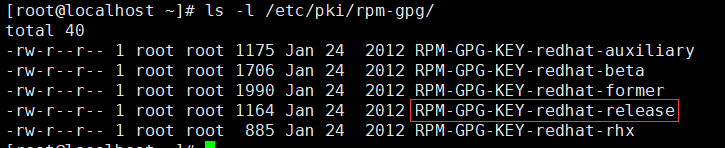


检验来源的合法性,及软件的合法性.



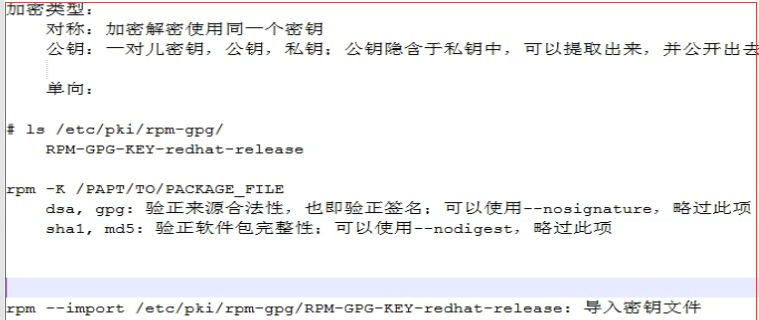
sha1 md5 验证完整性可以使用--nodigest忽略

dsa gpg 验证来源的合法性,及验证签名,可以使用--nosignature忽略



导入密钥文件,这只能验证红帽的rpm包



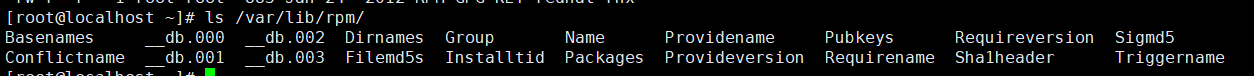


### 重建数据库

rpm

--rebuild 重建数据库,一定会重新建立

--initdb 初始化数据库,没有才建立



## yum

### yum软件及配置文件

yum类似于maven

rmp有个缺点,就是对库的依赖关系处理不是很好而yum 弥补了这个缺点

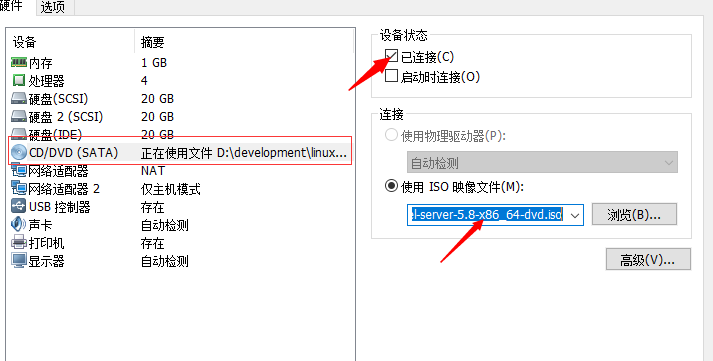
yum是一个C/S架构的软件

yum repository 仓库,所有rpm存储的地方,是一个文件服务器

1. ftp
2. web
3. file

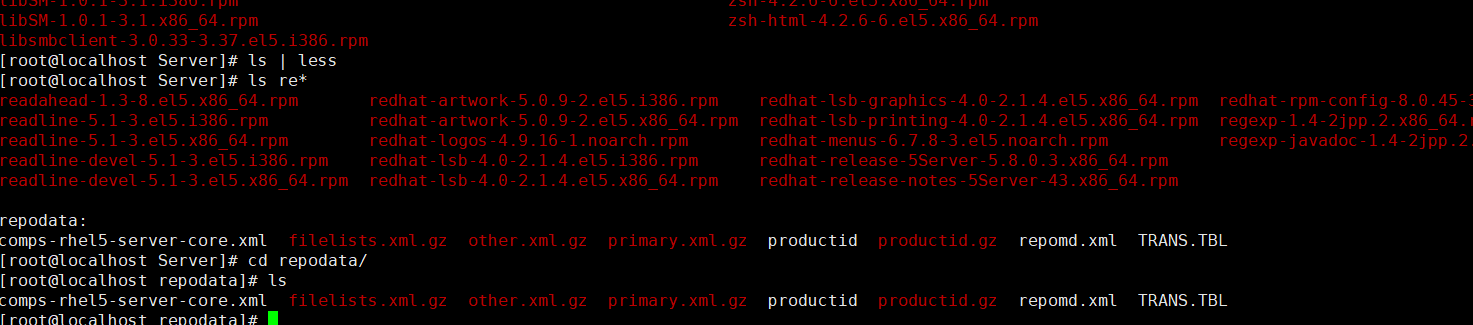
yum分为client server端

挂载rhel-server-5.8-x86\_64-dvd.iso



mount /dev/cdrom /mnt/cdrom

cd /mnt/cdrom/Server/



yum仓库中的元数据文件在repodata目录下

primary.xml.gz

所有rpm包的列表

依赖关系

每个rpm安装生成的文件列表

filelists.xml.gz

当前仓库中所有rpm包的所有文件列表

other.xml.gz

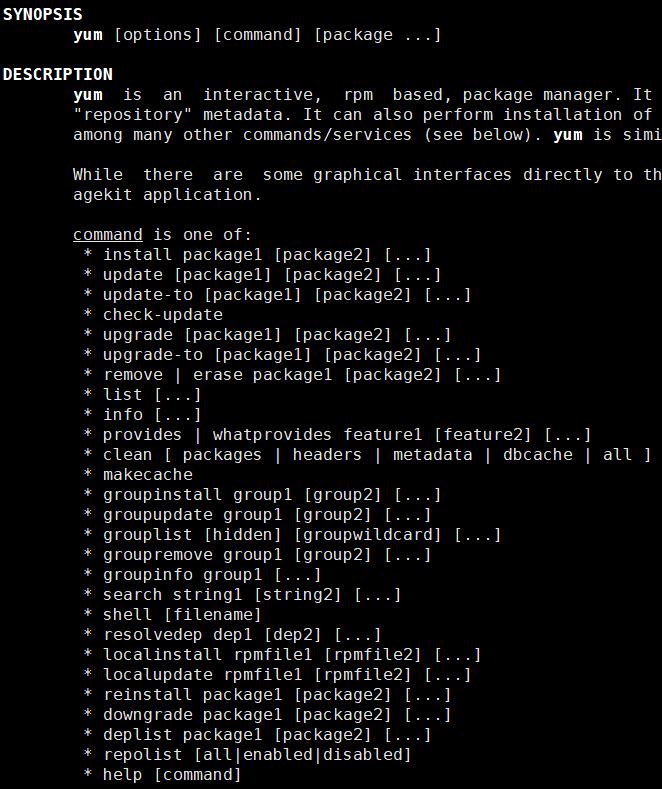
额外信息,rpm包修改日志

repomd.xml

记录的是上面3个文件的时间戳和校验和

comps\*.xml

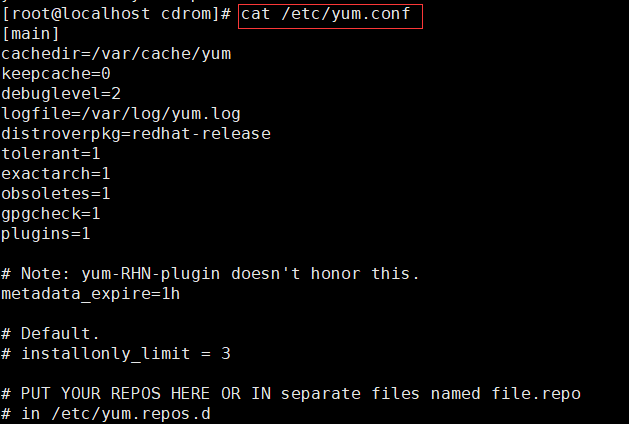
rpm分组信息

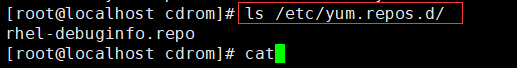


yum的配置文件

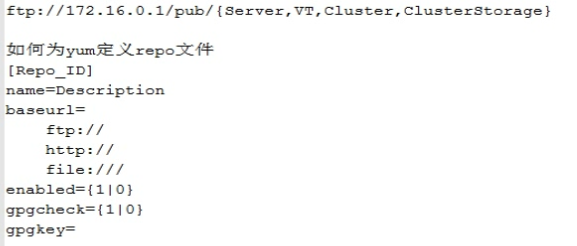
/etc/yum.conf

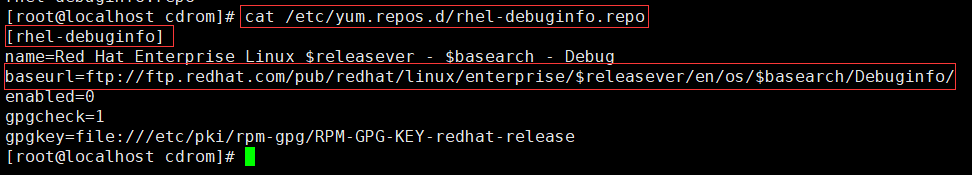
/etc/yum.repos.d/\*





如何定义yum中repo文件





### yum常用命令

**yum**

**list** 显示rpm包列表

**available** 可用的

**updates** 更新的

**installed** 安装的

**clear** 清理缓存

**packages**

**headers**

**metadata**

**dbcache**

**all**

**repolist** 显示yum仓库列表

**all** 全部

**enabled** 可用的

**disabled** 不可用的

**install** 安装

**-y** 自动回到yes

--nogpgcheck 不用检查

**update** 升级

**update-to** 升级到指定版本

**remove | erase** 卸载

**info** 查看软件包信息

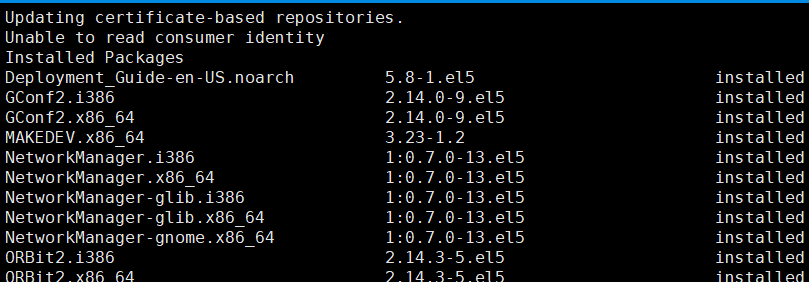
**provides | whatprovides** 查看指定的文件由哪个包安装生成

**grouplist**

**groupinstall**

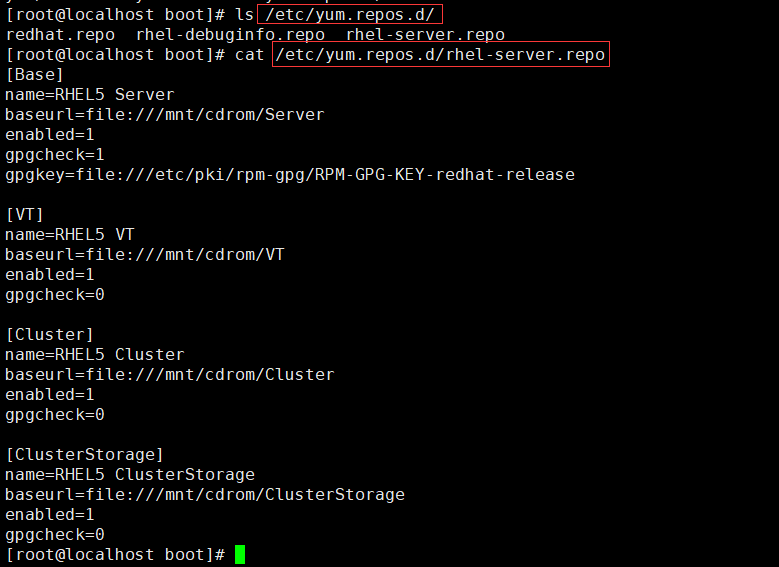
**yum localinstall** /xxx/xxx.rpm 自动处理依赖,如果包来自互联网可以—nogpgcheck,如果yum仓库没有相关依赖是不行的

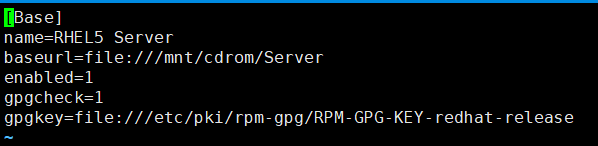
yum list | less



### 仓库配置

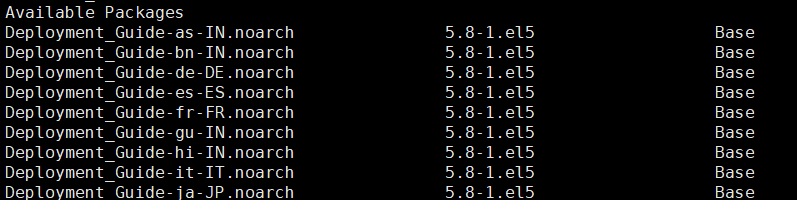
/etc/yum.repos.d/目录下





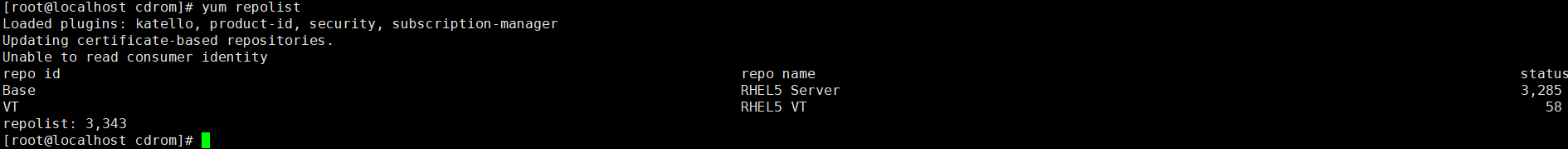
也可以配置多个

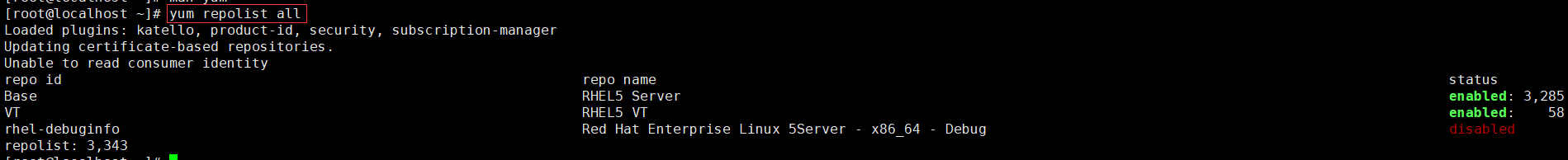




### 仓库列表

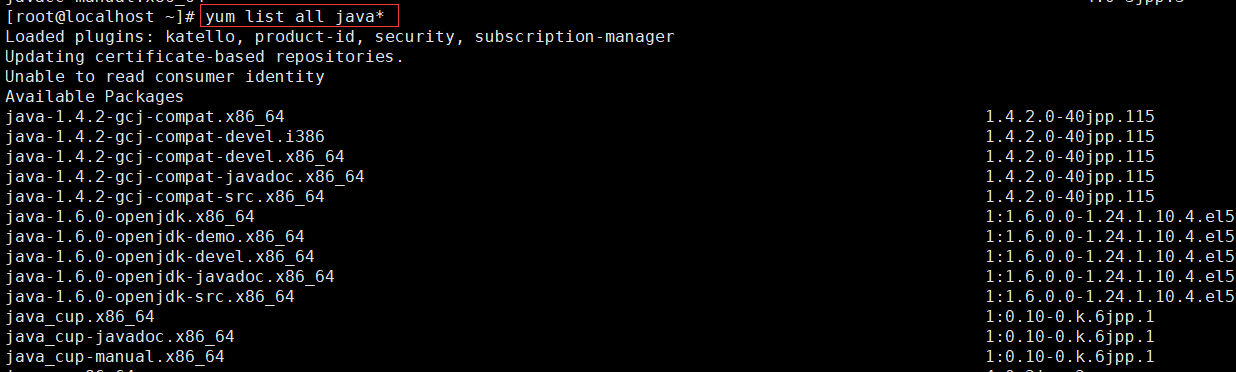
yum repolist

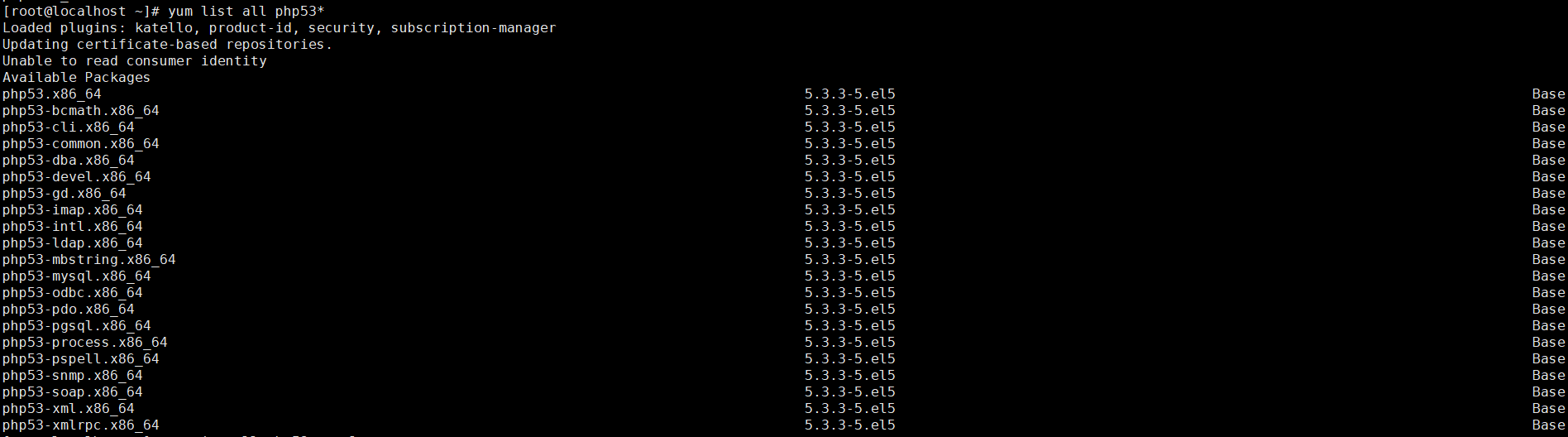




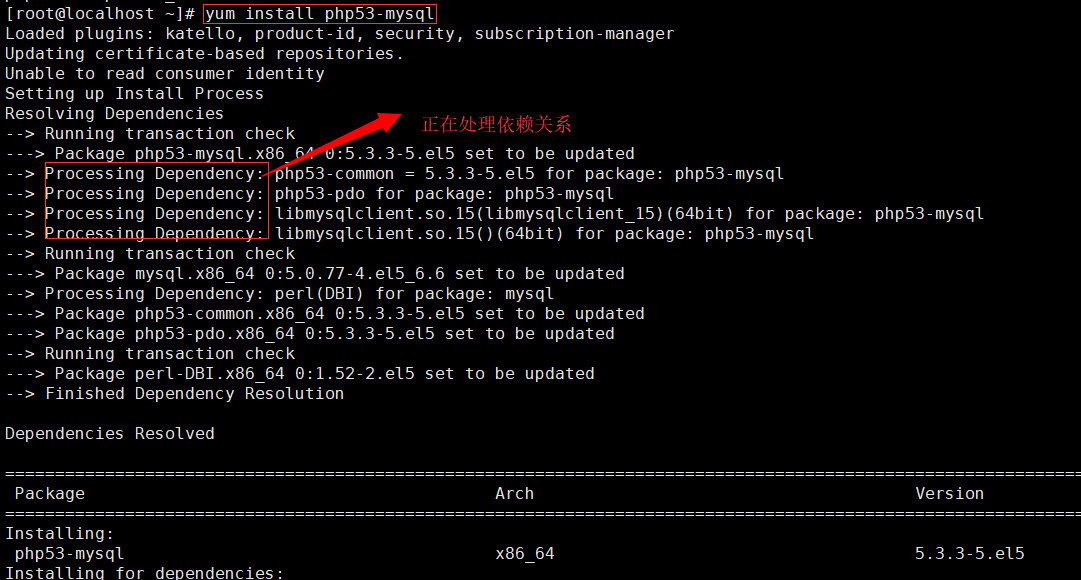
### 仓库中的安装包

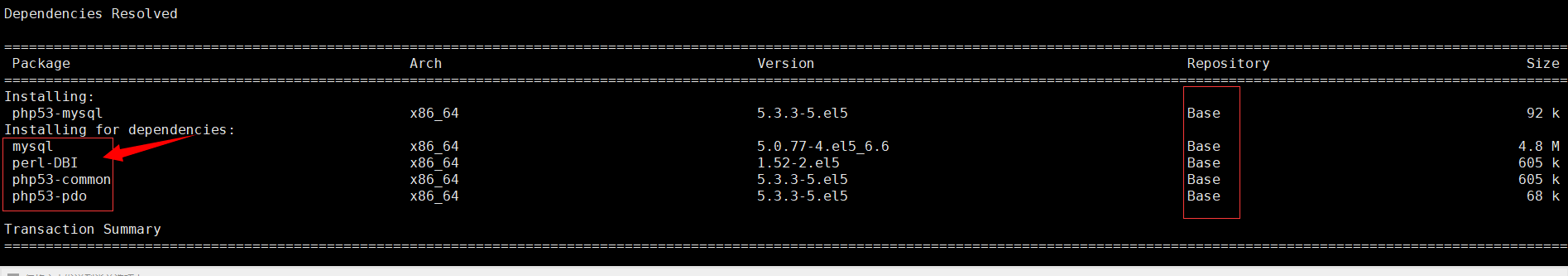
yum list all java\*

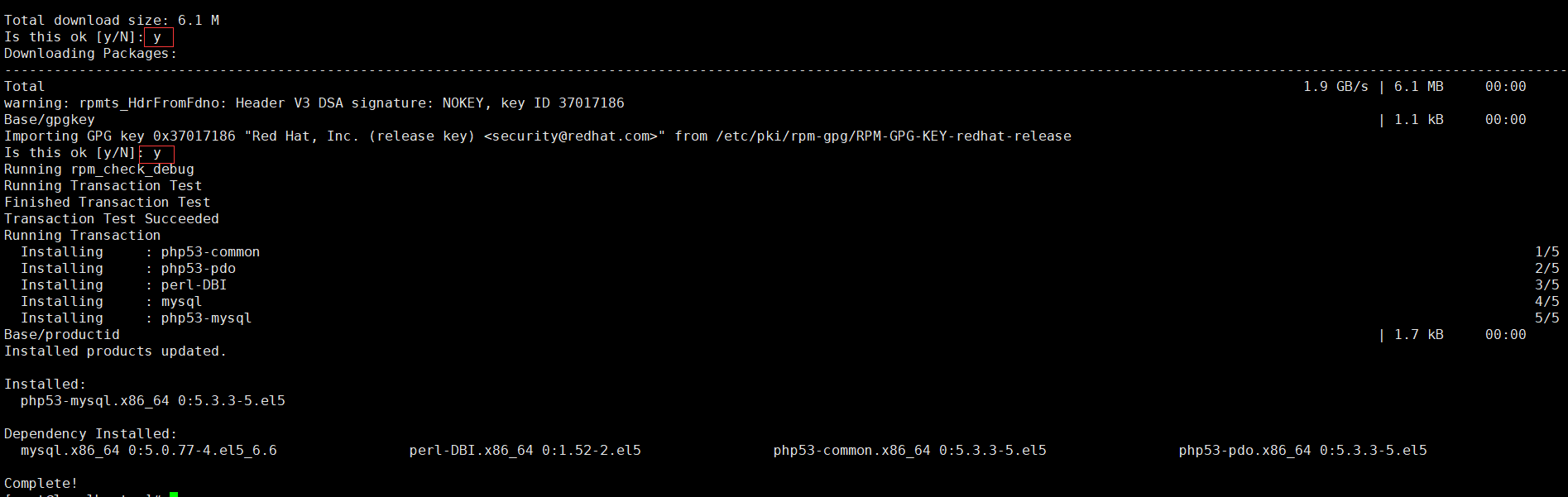




### 安装

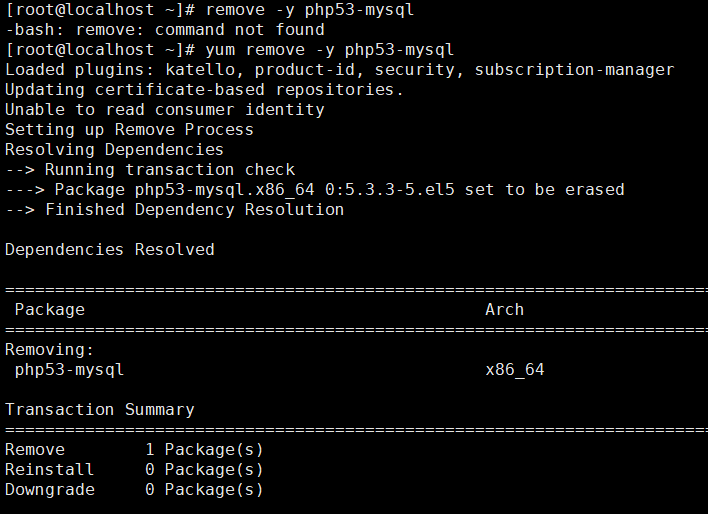






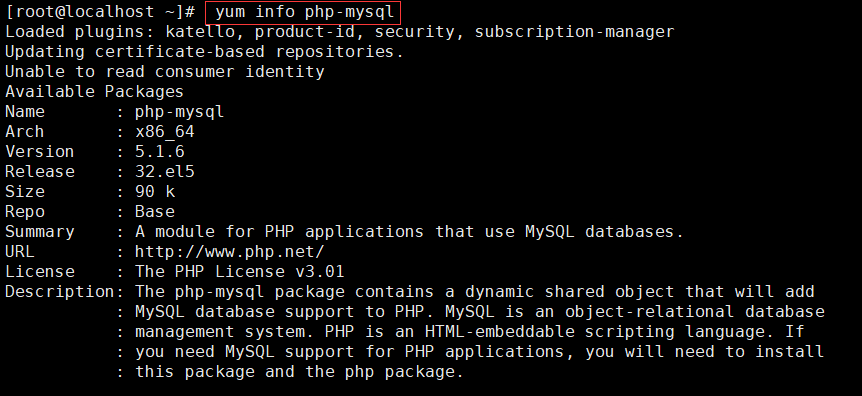
### 卸载

也会卸载依赖的包

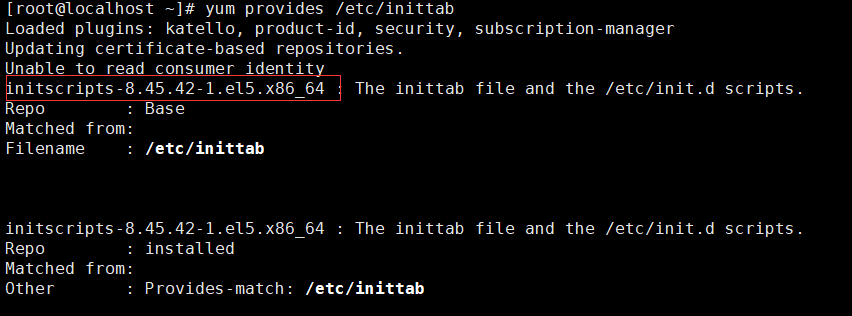


查看信息

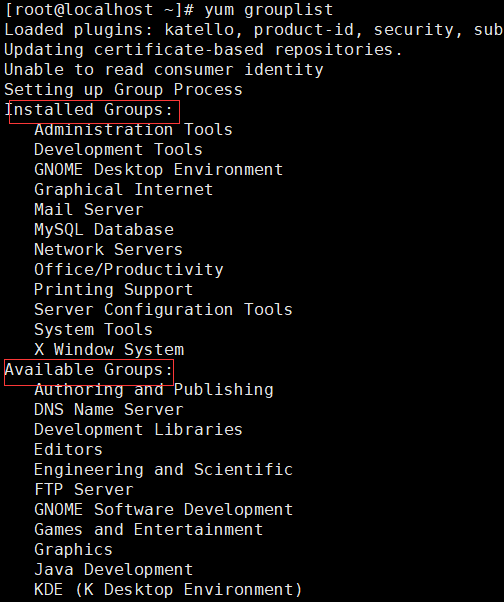
yum info xxx = rpm –qi

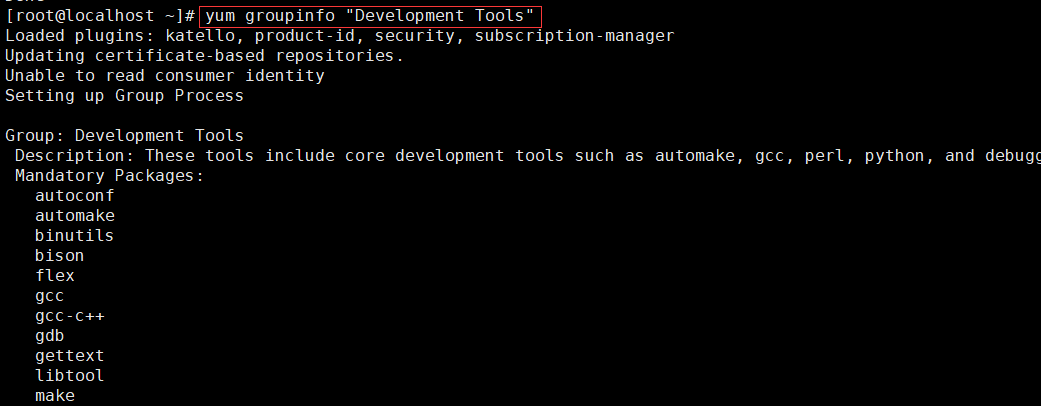


yum provides /etc/inittab



yum grouplist





## 编译源程序

rpm包,是二进制格式的

源程序🡪编译🡪二进制格式

有些特性是编译时选定的,如果编译未选定此特性,将无法使用.

rpm包的版本会落后于源码包的,甚至是落后很多

定制: 手动编译安装

所以要有: 编译环境,开发环境,开发库,开发工具

linux 是由C开发的和汇编语言所完成

C : gcc

C++ : g++

java : jdk

make由 gcc提供的项目管理工具,

makefile 配置文件gcc按何种次序去编译源程序文件中的源程序

make(gcc,g++)

automake由开发者定义,由auotmake生成makefile.in(半成品)

makefile.in由autoconf生成一个软件脚本configure(当前这个软件源程序如何编译的)

configure脚本提供了一个接口,让用户可以选择哪些特性,安装路径等,然后结合makefile.in生成makefile, 就可使用make makefile 来对源程序进行编译

**编译安装步骤**

前提: 准备编译环境

yum grouplist

redhat5上安装”Devlopment Tools”,”Devlopment Libraries”

redhat6 上安装”Devlopment Tools”,”Compatibility Libraries”

yum groupinstall ”Devlopment Tools”

yum groupinstall ” Devlopment Libraries”

1. tar 解压缩

2 ./configure

--help

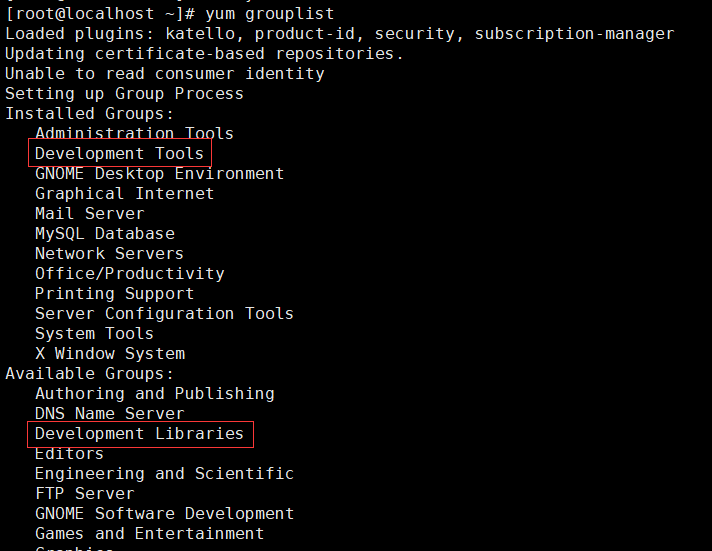
--prefix=/xxx/xxx/

--sysconfdir=/xxx/xxxx/

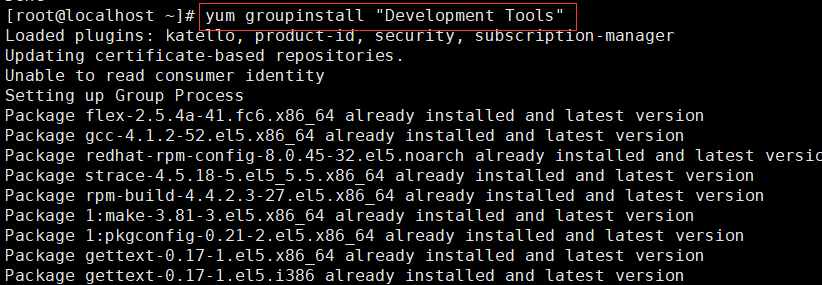
configure功能: 1.让用户选定编译特性 2.检查编译环境

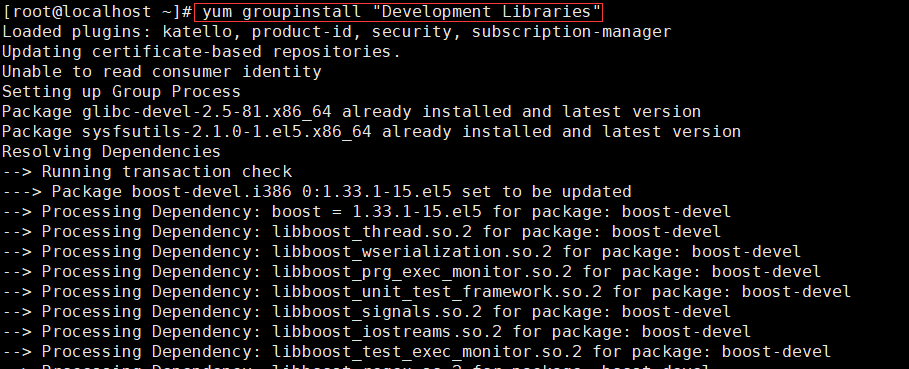
3 make

4 make install



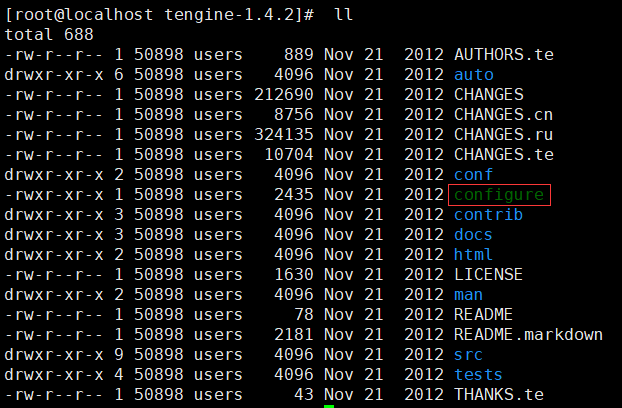
安装Development Tools 和 Development Libaraies





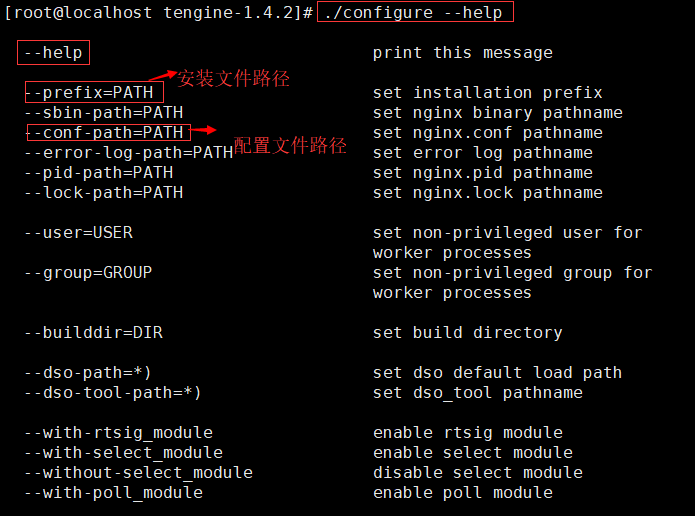
### 以安装tengine为例

#### 1 tar –xf tengine-1.4.2.tar.gz



#### 2 ./configure --help

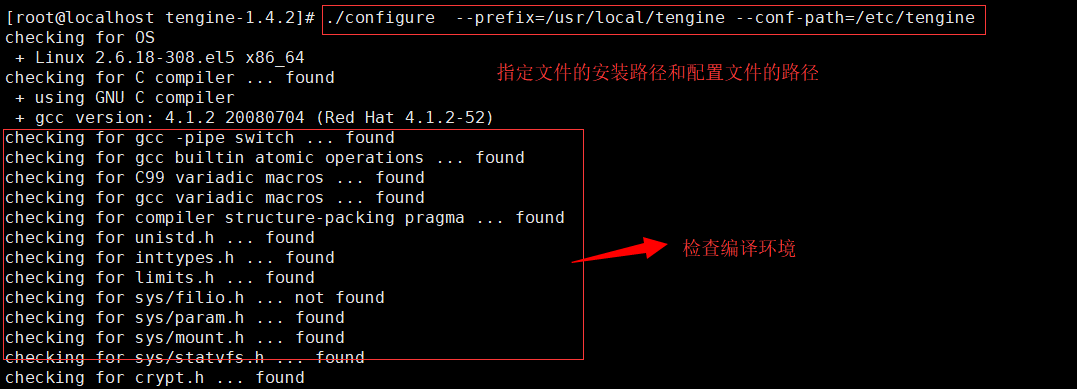
查看特性文件,不是所有文件都要指定

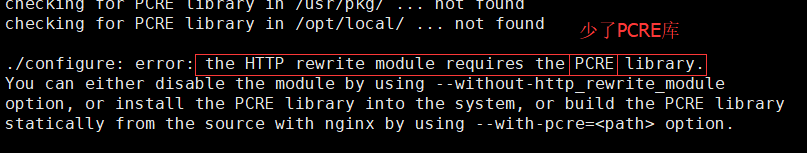


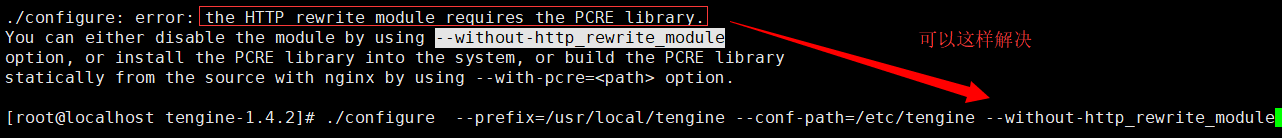
指定文件安装路径,和指定配置文件路径

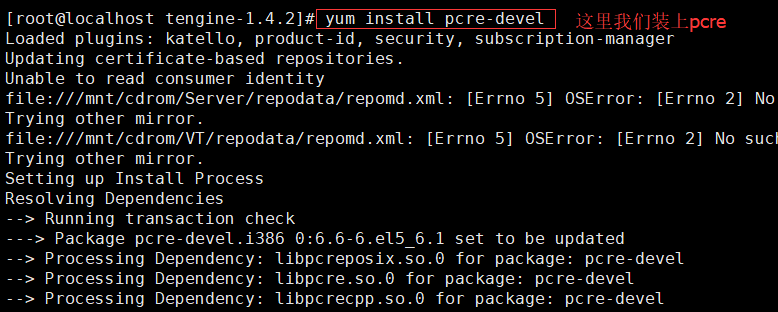
#### 3 ./configure –options

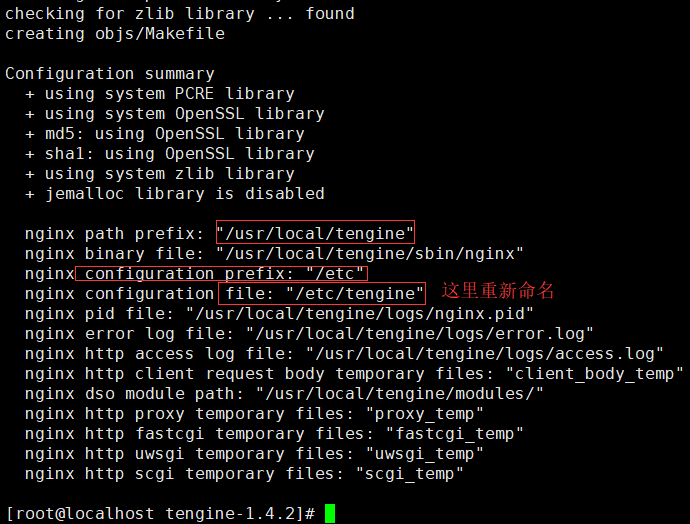
./configure --prefix=/user/local/tengine --conf-path=/etc/tengine/tengine.conf



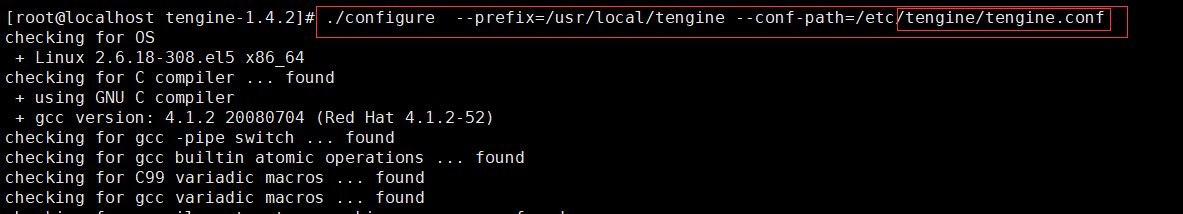


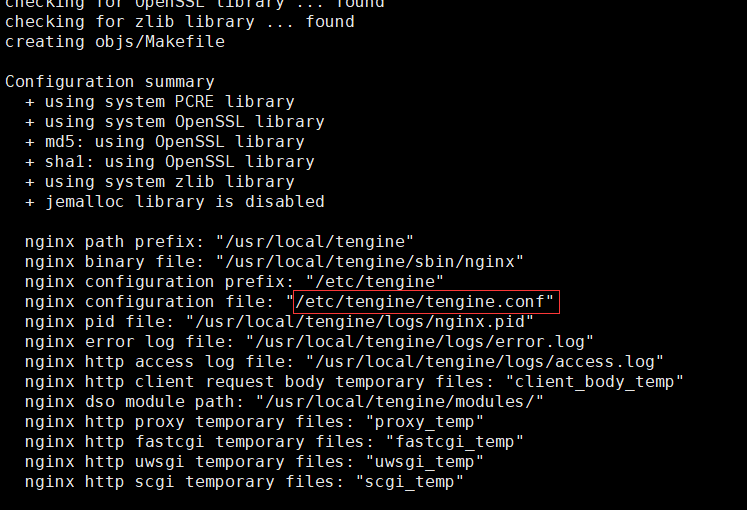




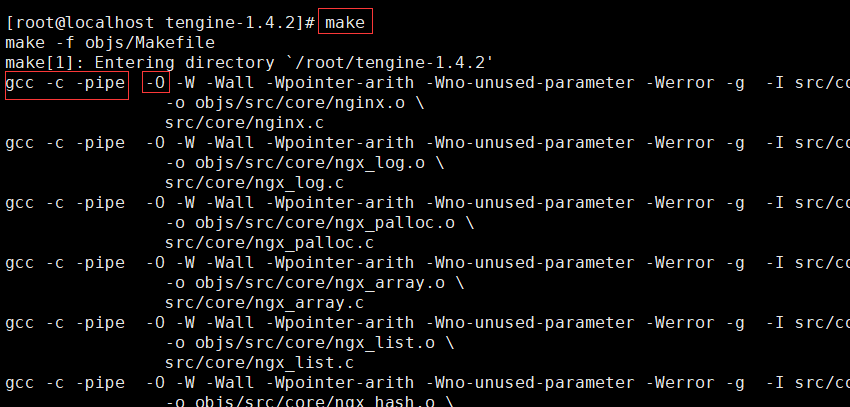


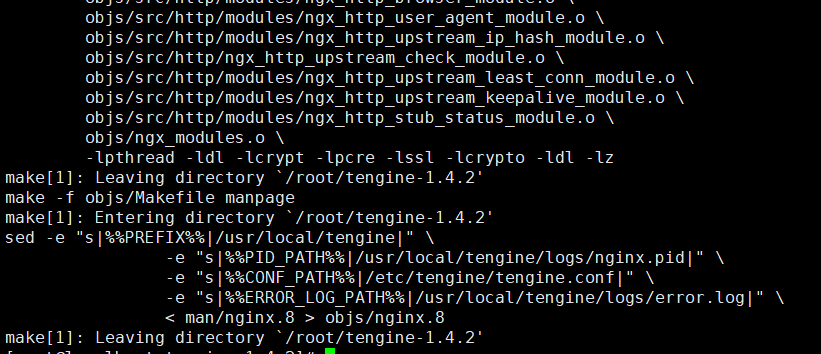
重新指定安装路径和配置文件路径



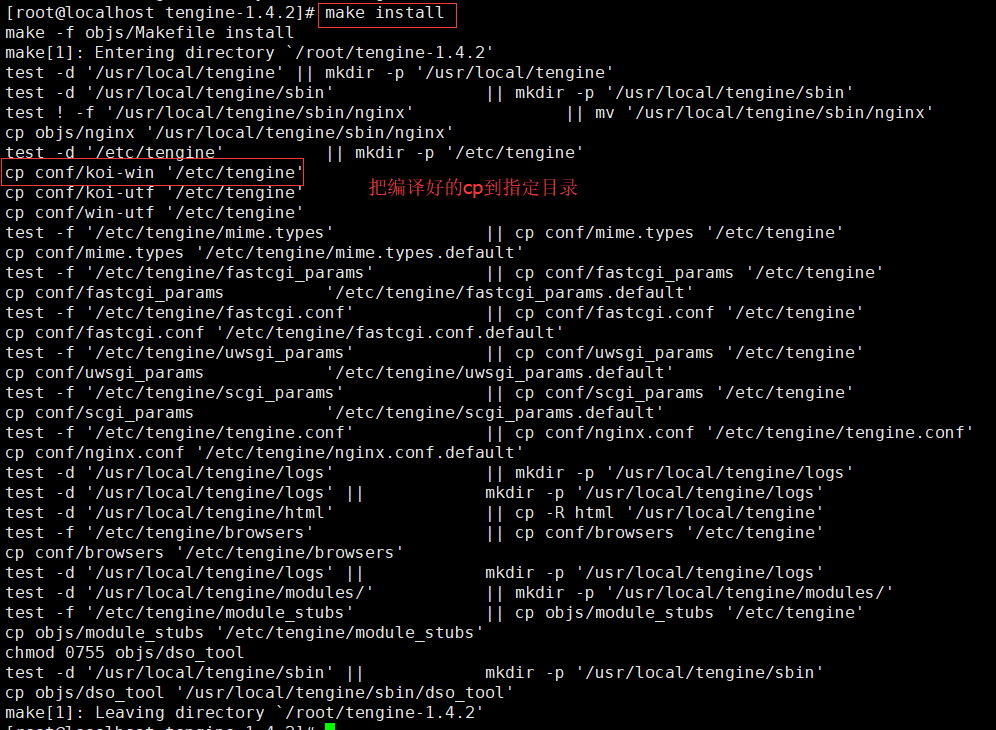


#### 4 make

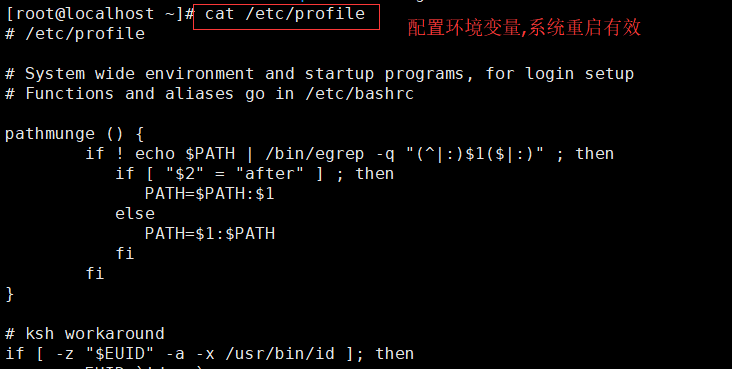




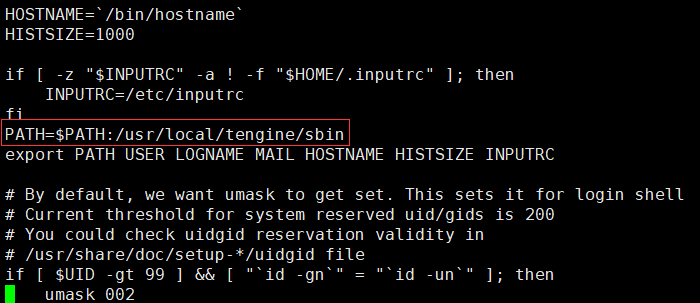
#### 5 make install



#### 6 配置环境变量



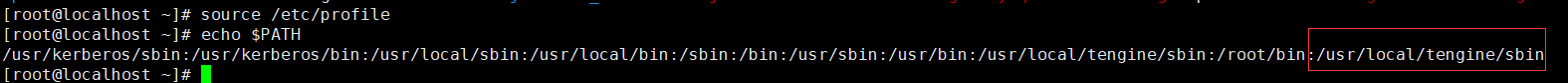
PATH=$PATH:/usr/local/tengine /sbin

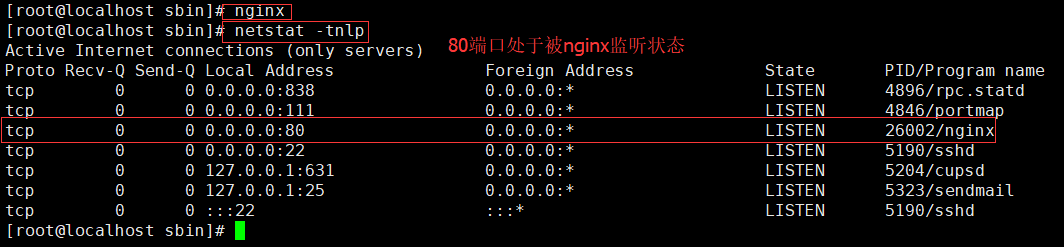


source /etc/profile 让系统重读配置文件

重新登录也可以







### 输出能力

注意,如果安装在了默认的路径下

1. 修改PATH环境变量,也能够识别此程序的二进制文件路径

1.1**可以修改/etc/profile,添加PATH=$PATH:/usr/local/xxxx/sbin**

1.2 **也可以在/etc/profile.d/目录下建立一个以.sh为后缀的文件名称,在里面定义**

**export PATH=$PATH:/usr/local/xxx/sbin**

1. **有很多软件都有输出库文件的能力,叫做lib目录,这个目录不能被其他程序所使用**.

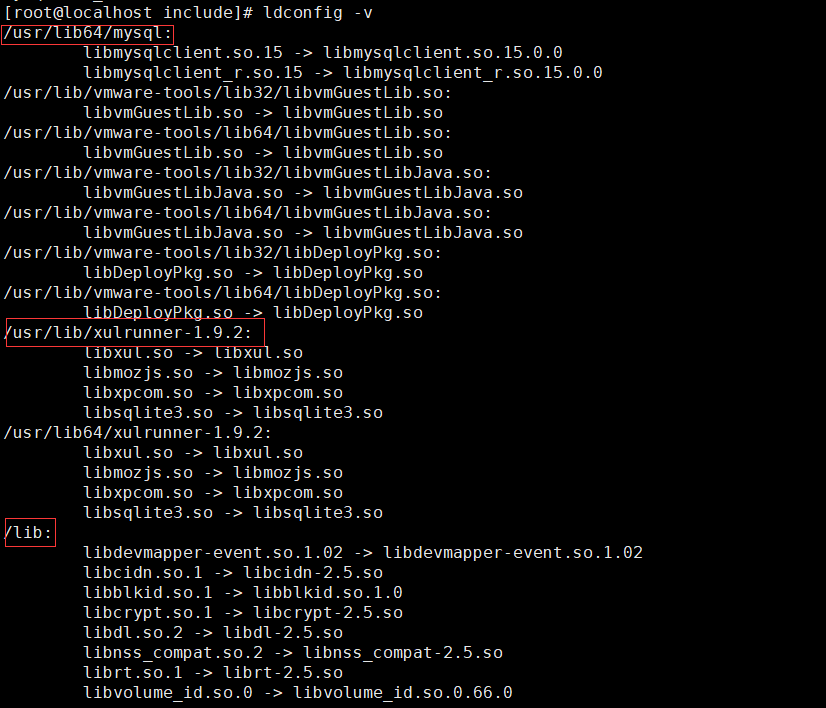
默认情况下,系统搜索库文件的路径有/lib, /usr/lib;需要额外增添搜寻路径,

**需要在/etc/ld.so.conf.d/目录中创建以.conf为后缀名的文件,而后把要增添的路径直接写到此文件中**,使用命令 ldconfig

**ldconfig** 通知系统重新搜寻库文件

-v : 显示重新搜寻库的过程

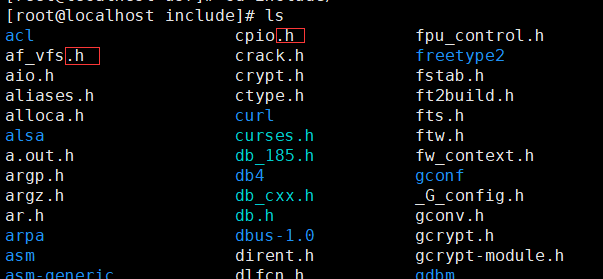




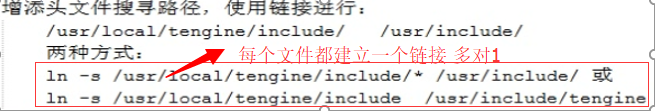
1. 头文件(c ,c++): 输出给系统,使用者才能调用软件中的库函数

默认:/usr/include



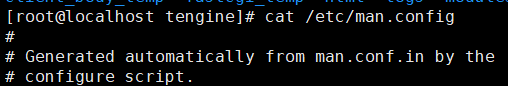


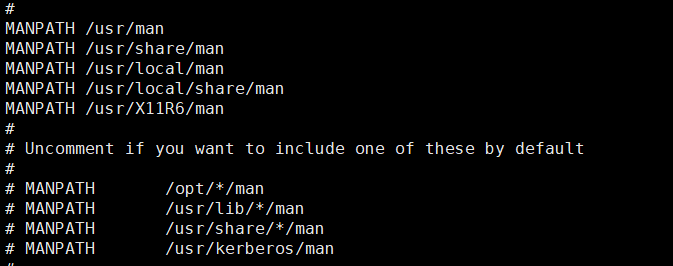
自定义增加头文件的搜寻路径,使用链接才行



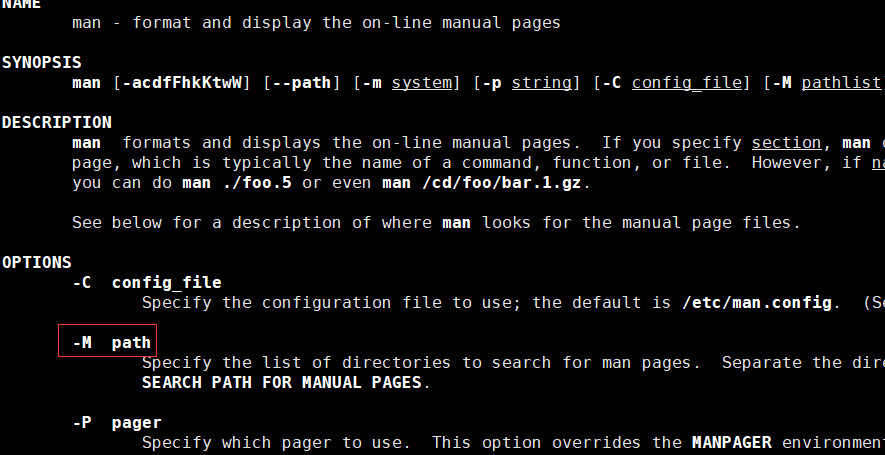
建议使用最后一条好维护

1. man文件路径 安装在--prefix指定目录下的man目录:/usr/share/man



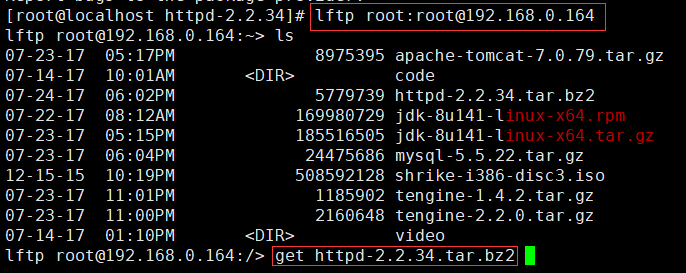




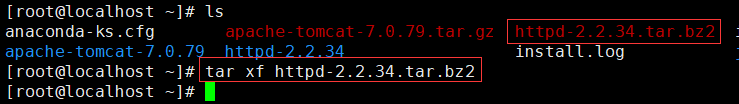


### 以安装httpd为例

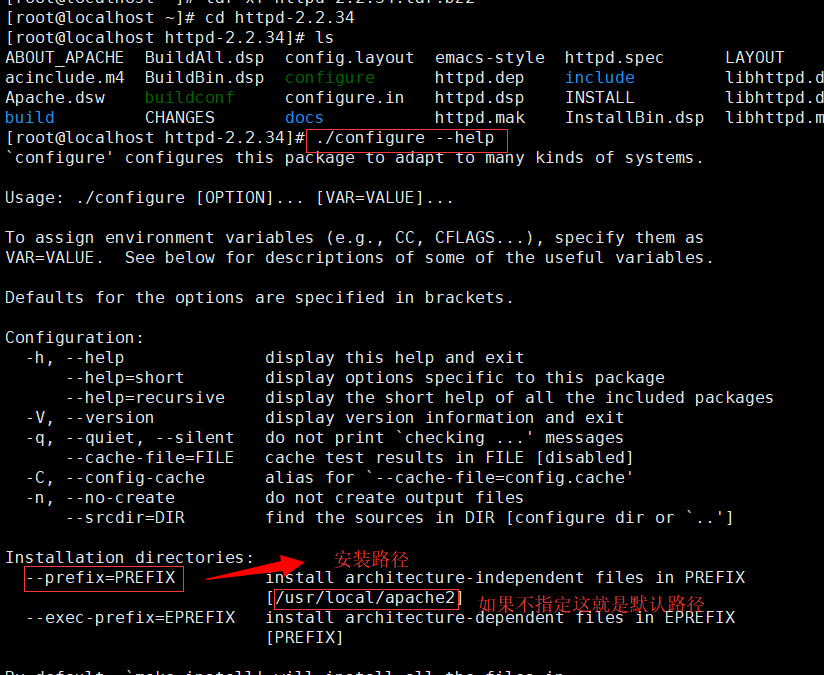
#### 1 lftp root:root@192.168.0.164

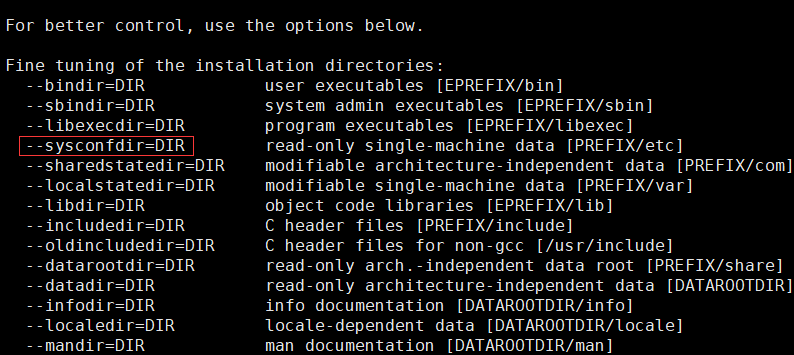


#### 2 tar xf httpd.2.2.34.tr.bz2



#### 3 ./configure --help

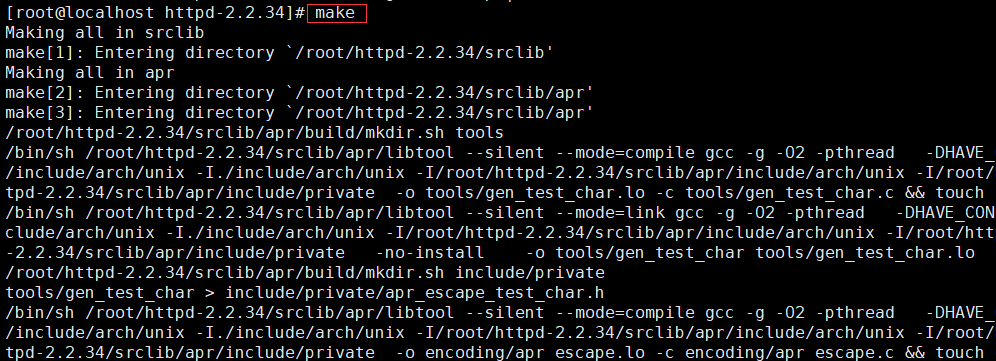




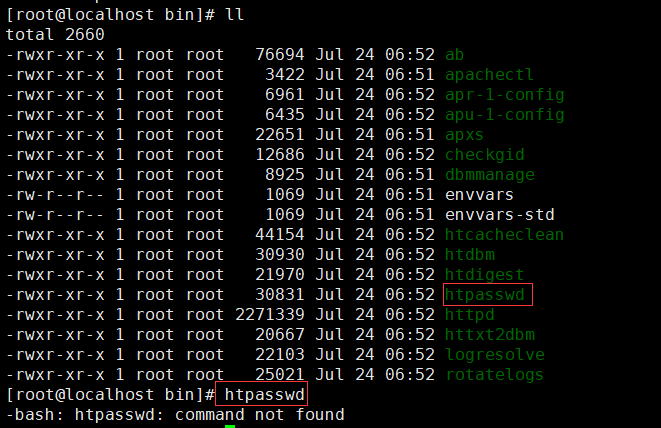
#### 4 指定安装路径,配置文件路径



#### 5 make

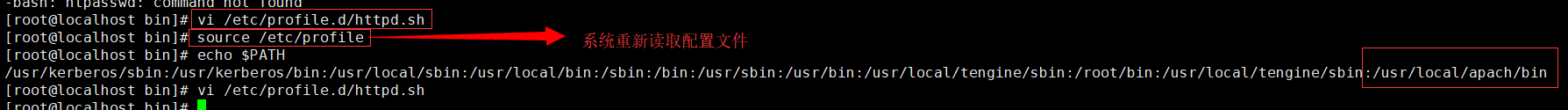


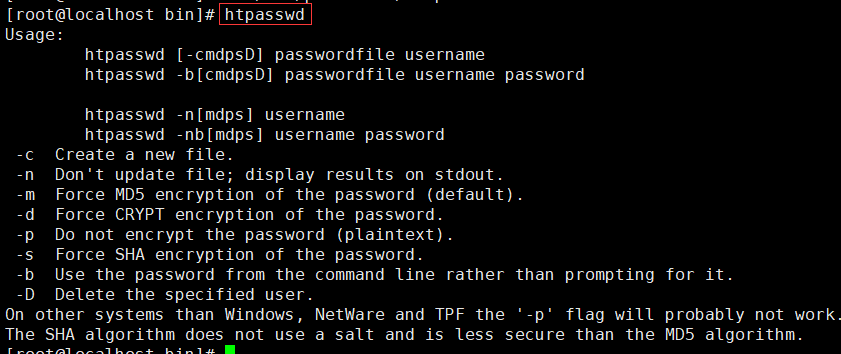
配置环境变量



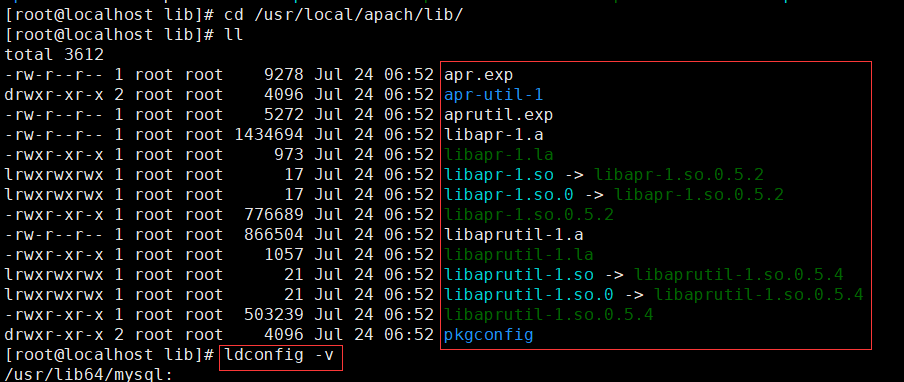
vi /etc/profile.d/httpd.sh







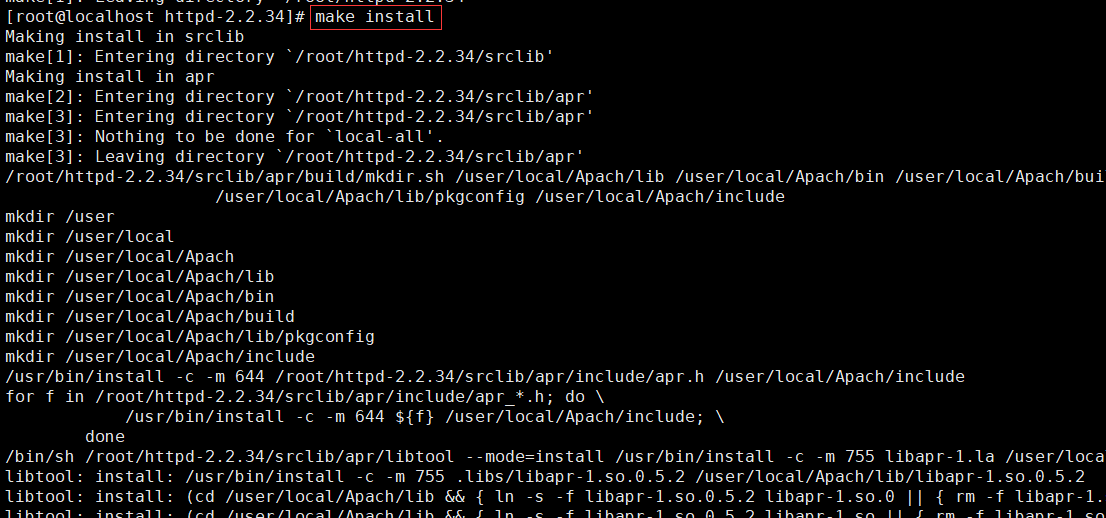
/usr/local/apach/lib/目录下的库文件



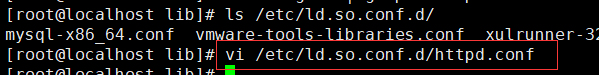
系统找不到/usr/local/apach/lib/目录下的库文件



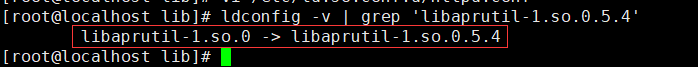
#### 6 make install



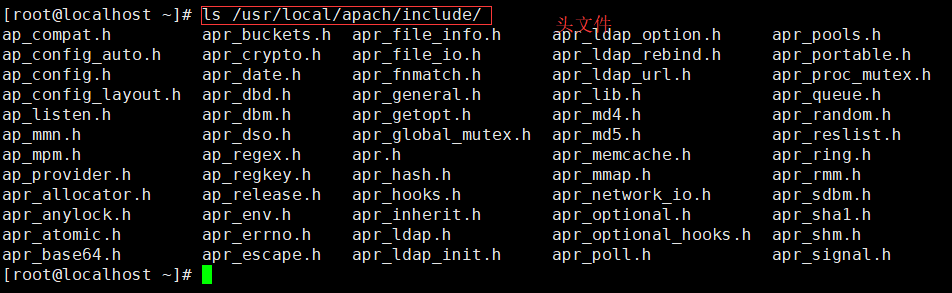
#### 7 向系统公开自己的库文件配置







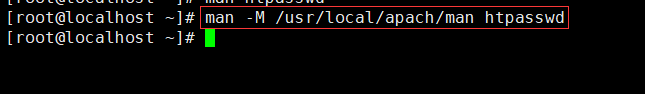
#### 8 输出头文件





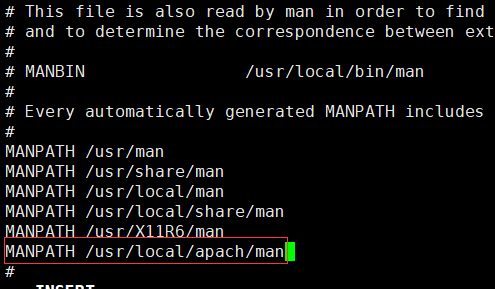
#### 9 man 文件路径

可以指定文件路径查看man 文档



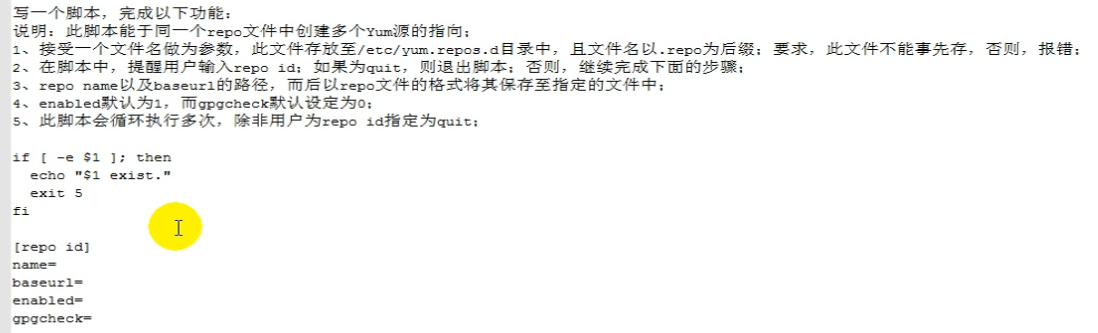
也可以在/etc/man.config配置文件中配置

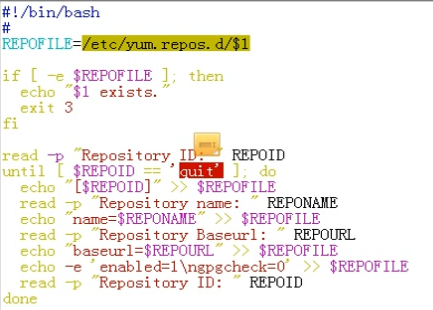








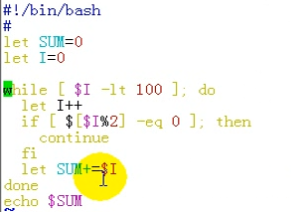


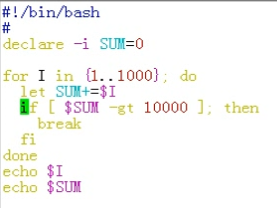


## 循环控制

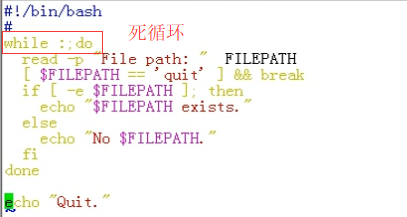
break : 提前退出循环

continue: 提前结束本轮循环,进入下一轮循环





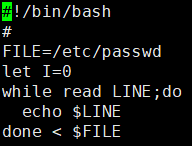
while特殊用法1 死循环

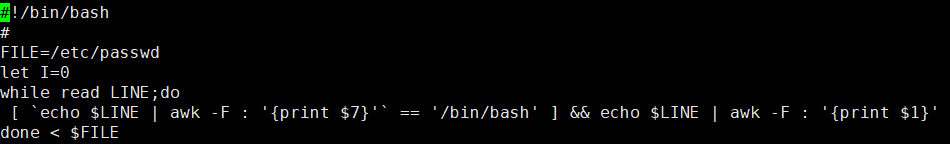


while 特殊用法2 读文件

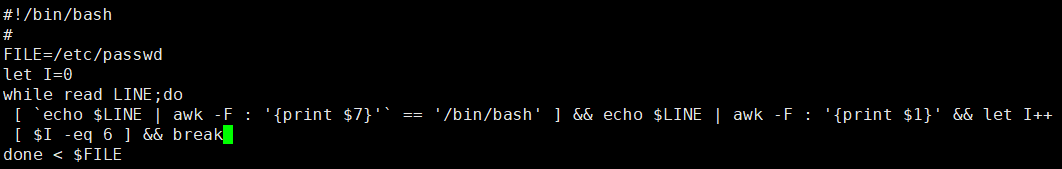
while read LINE;do

done < /xxx/xxx



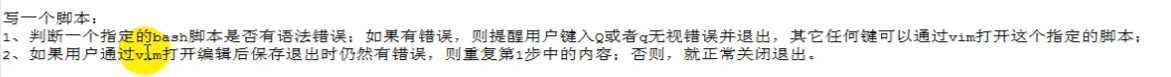


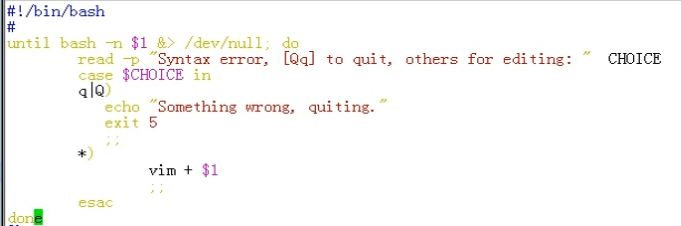
前6行



id号大于505的







## 函数

