[**Jdk1.8在CentOS7中的安装与配置**](http://www.cnblogs.com/cstzhou/p/4555423.html)

自从2014年3月19日甲骨文公司发布Java 8.0的正式版以来，面向对象的Java语言不仅朝着一个更好的方向发展，而且吸取了当前比较流行的函数式编程的特性——Java 8.0加入了函数式编程的特点。可以说是Java 8比Java 7有一个更好的性质。今天我将为大家演示一下Jdk 8在CentOS 7的安装与配置：

一、前期准备

环境：最小安装的CentOS 7（虚拟机安装的新版的centos系统，与6有一些配置文件上的差异），提前装好了lrssz工具（不会安装的，可以参看我的另一篇随笔——lrssz的安装）。

a)         首先从官网上下载[Jdk 8 for Linux x64](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)到window下。

b)         我这边用的最小安装，所以没有安装centos自带的openjdk，如果你安装时，不是最小安装的话，可能集成了系统的openjdk，所以我们先要删除自带的openjdk，具体步骤如下：

1)       在系统终端输入：rpm -qa | grep java，如果有openjdk的话，会出现类似于XXXX\_openjdk\_XXX的信息

2)       删除openjdk，在终端输入：rpm -e –-nodeps XXXX\_openjdk\_XXX 。即可删除自带的openjdk。

注意：上面这两步可以一次性完成，而且很简单哦！

rpm -e --nodeps `rpm -qa | grep java`

二、Jdk的安装

a)         我们登录超级用户，在超级用户目录root下建立一个目录app：

su root  #然后输入密码

mkdir app

b)         我们进入app，将下载好的jdk压缩包导入centos：

cd app

rz   #使用rz命令将jdk压缩包从window导入centos当下目录app中

c)         解压jdk压缩包

tar –xzvf jdk-8u45-linux-x64.gz

d)         解压后的jdk的文件夹名字为jdk1.8.0\_45,我们把它改为jdk1.8：

mv jdk1.8.0\_45/ jdk1.8/

e)         进入jdk文件夹，获取目录绝对路径（我的绝对路径是：/root/app/jdk1.8）：

cd jdk1.8

pwd

三、jdk的配置

a)         使用vim命令打开系统的环境变量配置文件：

vi /etc/profile

b)         在profile文件最后加入：

## JAVA

export JAVA\_HOME=jdk的绝对路径（我的是：/root/app/jdk1.8）

export PATH=$PATH:$JAVA\_HOME/bin

四、配置验证

a)         首先，我们输入java –version，会得到：

java version "1.8.0\_45"

Java(TM) SE Runtime Environment (build 1.8.0\_45-b14)

Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

b)         我们再输入javac，会得出：

Usage: javac <options> <source files>

where possible options include:

 -g                         Generate all debugging info

 -g:none                    Generate no debugging info

 -g:{lines,vars,source}     Generate only some debugging info

-nowarn                    Generate no warnings

  -verbose                   Output messages about what the compiler is doing

  -deprecation               Output source locations where deprecated APIs are used

  -classpath <path>          Specify where to find user class files and annotation processors

-cp <path>                 Specify where to find user class files and annotation processors

-sourcepath <path>         Specify where to find input source files

-bootclasspath <path>      Override location of bootstrap class files

-extdirs <dirs>            Override location of installed extensions

-endorseddirs <dirs>       Override location of endorsed standards path

-proc:{none,only}          Control whether annotation processing and/or compilation is done.

-processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process

-processorpath <path>      Specify where to find annotation processors

-parameters                Generate metadata for reflection on method parameters

-d <directory>             Specify where to place generated class files

-s <directory>             Specify where to place generated source files

-h <directory>             Specify where to place generated native header files

-implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files

-encoding <encoding>       Specify character encoding used by source files

-source <release>          Provide source compatibility with specified release

-target <release>          Generate class files for specific VM version

-profile <profile>         Check that API used is available in the specified profile

-version                   Version information

-help                      Print a synopsis of standard options

-Akey[=value]              Options to pass to annotation processors

-X                         Print a synopsis of nonstandard options

-J<flag>                   Pass <flag> directly to the runtime system

-Werror                    Terminate compilation if warnings occur

@<filename>                Read options and filenames from file

c)         如果得到以上两步正确结果的话，恭喜你，你的jdk8就已经配置好了。