**CentOS7.0安装JDK1.8.0\_31**

1.检查一下系统中的jdk版本

1. $>java -version

java version "1.7.0\_"  
OpenJDK Runtime Environment (IcedTea6 1.11.1) (rhel-1.45.1.11.1.el6-x86\_64)  
OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)

2.检测JDK安装包

[$>rpm -qa | grep java

avapackages-tools-3.4.1-6.el7\_0.noarch  
tzdata-java-2014i-1.el7.noarch  
java-1.7.0-openjdk-headless-1.7.0.71-2.5.3.1.el7\_0.x86\_64  
java-1.7.0-openjdk-1.7.0.71-2.5.3.1.el7\_0.x86\_64  
python-javapackages-3.4.1-6.el7\_0.noarch

3.卸载OpenJDK

1. $>rpm -e --nodeps tzdata-java-2014i-1.el7.noarch
2. $>rpm -e --nodeps java-1.7.0-openjdk-headless-1.7.0.71-2.5.3.1.el7\_0.x86\_64
3. $>rpm -e --nodeps java-1.7.0-openjdk-1.7.0.71-2.5.3.1.el7\_0.x86\_64

4.安装新的Jdk1.8.0\_31

 解压缩tar.gz

1. $>mkdir  -p /usr/lib/jvm
2. $>tar  -zxvf jdk1.8.0\_31.tar.gz -C  /usr/lib/jvm

5.设置环境变量

1. $> vi    /etc/profile

在最后添加：

1. export JAVA\_HOME=/usr/lib/jvm/jdk1.8.0\_31
2. export JRE\_HOME=${JAVA\_HOME}/jre
3. export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib
4. export  PATH=${JAVA\_HOME}/bin:$PATH

6.执行profile

[j$>source /etc/profile

7.检查新安装的jdk

1. $>java -version

## Centos7下配置DNS过程：

[root@centos7 git-1.8.3]# systemctl status NetworkManager.service

鈼[0m NetworkManager.service - Network Manager

Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2017-05-12 23:19:24 CST; 3 weeks 2 days ago

Docs: man:NetworkManager(8)

Main PID: 609 (NetworkManager)

CGroup: /system.slice/NetworkManager.service

鈹溾攢 609 /usr/sbin/NetworkManager --no-daemon

鈹溾攢28040 /sbin/dhclient -d -q -sf /usr/libexec/nm-dhcp-helper -pf /var/run/dhclient-enp0s9.pid -lf /var/lib/NetworkManager/dhclient-efa9fa33-6e8c-446b-a3b9-95108d5c8b13-enp0s9.lease -cf /var/lib/NetworkManager/dhclient-enp0...

鈹斺攢28046 /sbin/dhclient -d -q -sf /usr/libexec/nm-dhcp-helper -pf /var/run/dhclient-enp0s8.pid -lf /var/lib/NetworkManager/dhclient-defbd8a3-cc97-4426-9202-dc0916873b58-enp0s8.lease -cf /var/lib/NetworkManager/dhclient-enp0...

Jun 05 08:09:36 centos7 dhclient[28040]: bound to 192.168.56.102 -- renewal in 478 seconds.

Jun 05 08:10:47 centos7 dhclient[28046]: DHCPREQUEST on enp0s8 to 10.0.2.3 port 67 (xid=0x40868807)

Jun 05 08:10:47 centos7 dhclient[28046]: DHCPACK from 10.0.2.3 (xid=0x40868807)

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8918] dhcp4 (enp0s8): address 10.0.2.4

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8919] dhcp4 (enp0s8): plen 24 (255.255.255.0)

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8919] dhcp4 (enp0s8): gateway 10.0.2.1

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8919] dhcp4 (enp0s8): server identifier 10.0.2.3

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8919] dhcp4 (enp0s8): lease time 1200

Jun 05 08:10:47 centos7 NetworkManager[609]: <info> [1496621447.8920] dhcp4 (enp0s8): state changed bound -> bound

Jun 05 08:10:47 centos7 dhclient[28046]: bound to 10.0.2.4 -- renewal in 520 seconds.

[root@centos7 git-1.8.3]#

[root@centos7 git-1.8.3]# nmcli dev status

DEVICE TYPE STATE CONNECTION

enp0s8 ethernet connected enp0s8

enp0s9 ethernet connected enp0s9

enp0s10 ethernet disconnected --

enp0s3 ethernet disconnected --

lo loopback unmanaged --

[root@centos7 git-1.8.3]# nmcli con mod enp0s8 ipv4.dns "114.114.114.114 8.8.8.8"

[root@centos7 git-1.8.3]# nmcli con up enp0s8

Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/21)

[root@centos7 git-1.8.3]#

[root@centos7 git-1.8.3]#

[root@centos7 git-1.8.3]# ping baidu.com

PING baidu.com (111.13.101.208) 56(84) bytes of data.

64 bytes from 111.13.101.208 (111.13.101.208): icmp\_seq=1 ttl=49 time=67.9 ms

64 bytes from 111.13.101.208 (111.13.101.208): icmp\_seq=2 ttl=49 time=31.9 ms

^C

--- baidu.com ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1002ms

rtt min/avg/max/mdev = 31.967/49.963/67.959/17.996 ms

[root@centos7 git-1.8.3]#

## Centos6.7 配置公网DNS方法

修改网卡配置文件，红色字体

[root@Centos6 network-scripts]# vi ifcfg-eth1

DEVICE=eth1

HWADDR=08:00:27:00:12:E4

TYPE=Ethernet

UUID=07294dea-479a-4c96-9b29-24dde4b5aa11

ONBOOT=yes

NM\_CONTROLLED=yes

BOOTPROTO=none

IPADDR=10.0.2.5

NETMASK=255.255.255.0

GATEWAY=10.0.2.1

DNS1=8.8.8.8

DNS2=114.114.114.114

~

保存后，重启网络服务即可。

[root@Centos6 network-scripts]# service network restart

[root@Centos6 network-scripts]# ping www.baidu.com

PING www.a.shifen.com (183.232.231.172) 56(84) bytes of data.

64 bytes from 183.232.231.172: icmp\_seq=1 ttl=49 time=33.2 ms

64 bytes from 183.232.231.172: icmp\_seq=2 ttl=49 time=32.9 ms

## 4.Yum本地源配置

[root@Centos6 gcc\_rpm]# mkdir /mnt/cdrom

[root@Centos6 gcc\_rpm]# mount /dev/cdrom /mnt/cdrom

mount: block device /dev/sr0 is write-protected, mounting read-only

[root@Centos6 gcc\_rpm]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/vg\_centos6-lv\_root

50G 3.7G 43G 8% /

tmpfs 344M 76K 344M 1% /dev/shm

/dev/sda1 477M 37M 415M 9% /boot

/dev/mapper/vg\_centos6-lv\_home

28G 159M 27G 1% /home

/dev/sr0 3.7G 3.7G 0 100% /media/CentOS\_6.7\_Final

/dev/sr0 3.7G 3.7G 0 100% /mnt/cdrom

[root@Centos6 gcc\_rpm]#

[root@Centos6 gcc\_rpm]# cd /etc/yum.repos.d/

[root@Centos6 yum.repos.d]# ls

CentOS-Base.repo CentOS-Debuginfo.repo CentOS-fasttrack.repo CentOS-Media.repo CentOS-Vault.repo

[root@Centos6 yum.repos.d]# mkdir bak

[root@Centos6 yum.repos.d]# mv \*.repo bak/

[root@Centos6 yum.repos.d]# ls

bak

[root@Centos6 yum.repos.d]# cat /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-

RPM-GPG-KEY-CentOS-6 RPM-GPG-KEY-CentOS-Security-6

RPM-GPG-KEY-CentOS-Debug-6 RPM-GPG-KEY-CentOS-Testing-6

[root@Centos6 yum.repos.d]# sudo vim local.repo

[local\_server]

name=Thisis a local repo

baseurl=file:///mnt/cdrom

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

[root@Centos6 yum.repos.d]# yum clean all

[root@Centos6 yum.repos.d]# yum makecache

...

Metadata Cache Created

[root@Centos6 yum.repos.d]#

## 5.[CentOS 配置网络yum源](http://www.cnblogs.com/zoulongbin/p/5773330.html)

配置网络yum源(需要保证外网开通，使用网易163提供开源镜像站)

1、下载repo文件

方法一：下载到本地windows再上传到linux服务器上:

下载地址 ： <http://mirrors.163.com/.help/CentOS6-Base-163.repo>

方法二：Linux服务器直接下载：

 wget –P /home/backup/  <http://mirrors.163.com/.help/CentOS6-Base-163.repo>

2. 备份并替换系统的repo文件

[root@localhost ~]# cp /etc/yum.repos.d/CentOS-Base.repo  /home/backup/CentOS-Base.repo           #备份原来的.repo配置文

[root@localhost~]#mv/home/backup/CentOS6-Base-163.repo  /etc/yum.repos.d/CentOS-Base.repo     #复制网易下载的.repo并修改文件名称

3. 执行yum源更新

[root@localhost ~]# yum clean all       #清除以前使用yum的缓存

[root@localhost ~]# yum makecache    #建立一个缓存，以后方便在缓存中搜索

4、检查网络yum源

[root@localhost ~]#yum list | grep telnet\*      #查找网络yum源的telnet安装包

telnet.x86\_64                              1:0.17-59.el7               base

telnet-server.x86\_64                        1:0.17-59.el7               base

[root@localhost ~]# yum install telnet.x86\_64    #执行安装

标注：把网易163的/etc/yum.repos.d/ CentOS-Base.repo多余配置项删除掉文件修改成如下也能正常使用yum下载，把配置多余配置项删除掉

[base]

name=CentOS-$releasever - Base - 163.com

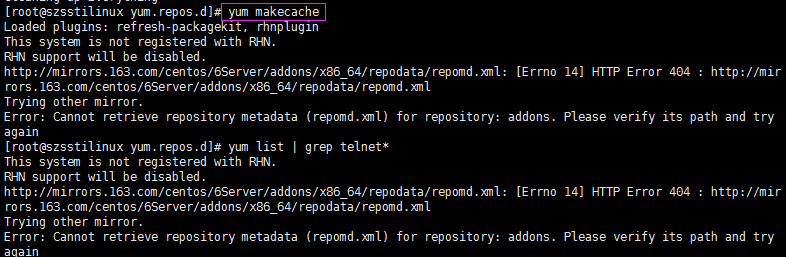
baseurl=http://mirrors.163.com/centos/$releasever/os/$basearch/

gpgcheck=1

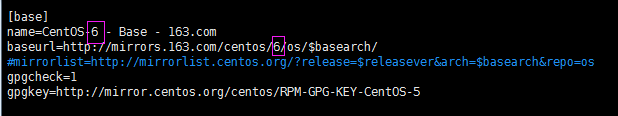
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6

**特殊情况：**

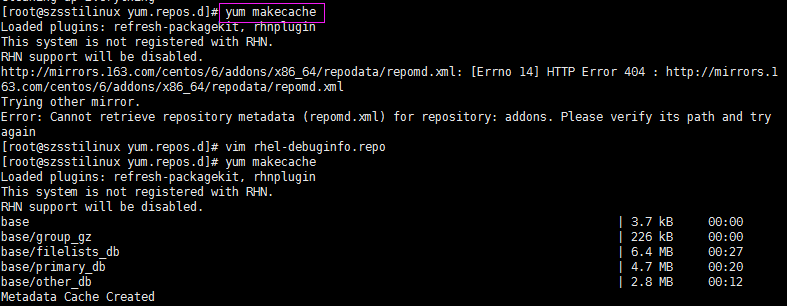
如果运行 yum  makecache  提示如下错误：



那就是你系统没有配置环境变量$releasever 和 $basearch ，需要把配置文件就把$releasever 都 改成6或者i386



运行 yum  makecache  正常情况应该是这样提示：



## [CentOS6.5 安装Python2.7后， yum出现“No module named yum”错误](http://blog.csdn.net/alsmile/article/details/20745265)

安装如下方法安装python2.7：

yum install –y python27 python27-devel [**Python**](http://lib.csdn.net/base/python)-docutils

cd /usr/bin/

rm -rf [**python**](http://lib.csdn.net/base/python)

cp python2.7 python

出现yum错误：No module named yum

解决方法，查看 /usr/bin下python有哪几个版本

ll /usr/bin

我这里是：2.6  和  2.7 （刚安装的）

由于yum命令不兼容python2.7，需修改/usr/bin/yum文件，将第一行由“#!/usr/bin/python”改为“#!/usr/bin/python2.6"

**Linux下环境变量设置的三种方法**：

如想将一个路径加入到$PATH中，可以像下面这样做：

**1、**控制台中设置，不赞成这种方式，因为他只对当前的shell 起作用，换一个shell设置就无效了：

$PATH="$PATH":/NEW\_PATH  (关闭shell Path会还原为原来的path)

**2、**修改 /etc/profile 文件，如果你的计算机仅仅作为开发使用时推存使用这种方法，因为所有用户的shell都有权使用这个环境变量，可能会给系统带来安全性问题。这里是针对所有的用户的，所有的shell

在/etc/profile的最下面添加：  export  PATH="$PATH:/NEW\_PATH"

**3、**修改bashrc文件，这种方法更为安全，它可以把使用这些环境变量的权限控制到用户级别，这里是针对某一特定的用户，如果你需要给某个用户权限使用这些环境变量，你只需要修改其个人用户主目录下的 .bashrc文件就可以了。

在下面添加：

Export  PATH="$PATH:/NEW\_PATH"

使生效：

source /etc/profile

## 使用命令：

如查找mysql\_config文件的位置，使用  
 [root@Centos6 MySQL-python-1.2.5]# find / -name mysql\_config

/opt/mysql/bin/mysql\_config

# 替换文件中的所有匹配项  
 sed -i 's/原字符串/home /g' filename

**目前可以直接使用 tar xvJf  \*\*\*.tar.xz来解压**

# 2、正则表达式 - 语法

正则表达式(regular expression)描述了一种字符串匹配的模式（pattern），可以用来检查一个串是否含有某种子串、将匹配的子串替换或者从某个串中取出符合某个条件的子串等。

例如：

* **runoo+b**，可以匹配 runoob、runooob、runoooooob 等，+ 号代表前面的字符必须至少出现一次（1次或多次）。
* **runoo\*b**，可以匹配 runob、runoob、runoooooob 等，\* 号代表字符可以不出现，也可以出现一次或者多次（0次、或1次、或多次）。
* **colou?r** 可以匹配 color 或者 colour，? 问号代表前面的字符最多只可以出现一次（0次、或1次）。

构造正则表达式的方法和创建数学表达式的方法一样。也就是用多种元字符与运算符可以将小的表达式结合在一起来创建更大的表达式。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者所有这些组件的任意组合。

正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为"元字符"）组成的文字模式。模式描述在搜索文本时要匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

## 普通字符

普通字符包括没有显式指定为元字符的所有可打印和不可打印字符。这包括所有大写和小写字母、所有数字、所有标点符号和一些其他符号。

## 非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

|  |  |
| --- | --- |
| **字符** | **描述** |
| \cx | 匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。 |
| \f | 匹配一个换页符。等价于 \x0c 和 \cL。 |
| \n | 匹配一个换行符。等价于 \x0a 和 \cJ。 |
| \r | 匹配一个回车符。等价于 \x0d 和 \cM。 |
| \s | 匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。 |
| \S | 匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。 |
| \t | 匹配一个制表符。等价于 \x09 和 \cI。 |
| \v | 匹配一个垂直制表符。等价于 \x0b 和 \cK。 |

## 特殊字符

所谓特殊字符，就是一些有特殊含义的字符，如上面说的 **runoo\*b** 中的 **\***，简单的说就是表示任何字符串的意思。如果要查找字符串中的 **\*** 符号，则需要对 **\*** 进行转义，即在其前加一个 **\**: **runo\\*ob** 匹配 runo\*ob。

许多元字符要求在试图匹配它们时特别对待。若要匹配这些特殊字符，必须首先使字符"转义"，即，将反斜杠字符**\** 放在它们前面。下表列出了正则表达式中的特殊字符：

|  |  |
| --- | --- |
| **特别字符** | **描述** |
| $ | 匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。 |
| ( ) | 标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \( 和 \)。 |
| \* | 匹配前面的子表达式零次或多次。要匹配 \* 字符，请使用 \\*。 |
| + | 匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。 |
| . | 匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用 \. 。 |
| [ | 标记一个中括号表达式的开始。要匹配 [，请使用 \[。 |
| ? | 匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。 |
| \ | 将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。 |
| ^ | 匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。 |
| { | 标记限定符表达式的开始。要匹配 {，请使用 \{。 |
| | | 指明两项之间的一个选择。要匹配 |，请使用 \|。 |

## 限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 **\*** 或 **+** 或 **?** 或 **{n}** 或 **{n,}** 或 **{n,m}** 共6种。

正则表达式的限定符有：

|  |  |
| --- | --- |
| **字符** | **描述** |
| \* | 匹配前面的子表达式零次或多次。例如，zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}。 |
| + | 匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。 |
| ? | 匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。 |
| {n} | n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。 |
| {n,} | n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o\*'。 |
| {n,m} | m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。 |

由于章节编号在大的输入文档中会很可能超过九，所以您需要一种方式来处理两位或三位章节编号。限定符给您这种能力。下面的正则表达式匹配编号为任何位数的章节标题：

/Chapter [1-9][0-9]\*/

请注意，限定符出现在范围表达式之后。因此，它应用于整个范围表达式，在本例中，只指定从 0 到 9 的数字（包括 0 和 9）。

这里不使用 + 限定符，因为在第二个位置或后面的位置不一定需要有一个数字。也不使用？字符，因为它将章节编号限制到只有两位数。您需要至少匹配 Chapter 和空格字符后面的一个数字。

如果您知道章节编号被限制为只有 99 章，可以使用下面的表达式来至少指定一位但至多两位数字。

/Chapter [0-9]{1,2}/

上面的表达式的缺点是，大于 99 的章节编号仍只匹配开头两位数字。另一个缺点是 Chapter 0 也将匹配。只匹配两位数字的更好的表达式如下：

/Chapter [1-9][0-9]?/

或

/Chapter [1-9][0-9]{0,1}/

**\*、+和?限定符都是贪婪的，因为它们会尽可能多的匹配文字，只有在它们的后面加上一个?就可以实现非贪婪或最小匹配。**

例如，您可能搜索 HTML 文档，以查找括在 H1 标记内的章节标题。该文本在您的文档中如下：

<H1>Chapter 1 - 介绍正则表达式</H1>

**贪婪：**下面的表达式匹配从开始小于符号 (<) 到关闭 H1 标记的大于符号 (>) 之间的所有内容。

/<.\*>/

**非贪婪：**如果您只需要匹配开始和介绍 H1 标记，下面的非贪婪表达式只匹配 <H1>。

/<.\*?>/

如果只想匹配开始的 H1 标签，表达式则是：

/<\w+?>/

通过在 **\***、**+** 或 **?** 限定符之后放置 **?**，该表达式从"贪心"表达式转换为"非贪心"表达式或者最小匹配。

## 定位符

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

定位符用来描述字符串或单词的边界，**^** 和 **$** 分别指字符串的开始与结束，**\b** 描述单词的前或后边界，**\B** 表示非单词边界。

正则表达式的定位符有：

|  |  |
| --- | --- |
| **字符** | **描述** |
| ^ | 匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配。 |
| $ | 匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与 \n 或 \r 之前的位置匹配。 |
| \b | 匹配一个字边界，即字与空格间的位置。 |
| \B | 非字边界匹配。 |

**注意**：不能将限定符与定位符一起使用。由于在紧靠换行或者字边界的前面或后面不能有一个以上位置，因此不允许诸如 **^\*** 之类的表达式。

若要匹配一行文本开始处的文本，请在正则表达式的开始使用 **^** 字符。不要将 **^** 的这种用法与中括号表达式内的用法混淆。

若要匹配一行文本的结束处的文本，请在正则表达式的结束处使用 **$** 字符。

若要在搜索章节标题时使用定位点，下面的正则表达式匹配一个章节标题，该标题只包含两个尾随数字，并且出现在行首：

/^Chapter [1-9][0-9]{0,1}/

真正的章节标题不仅出现行的开始处，而且它还是该行中仅有的文本。它即出现在行首又出现在同一行的结尾。下面的表达式能确保指定的匹配只匹配章节而不匹配交叉引用。通过创建只匹配一行文本的开始和结尾的正则表达式，就可做到这一点。

/^Chapter [1-9][0-9]{0,1}$/

匹配字边界稍有不同，但向正则表达式添加了很重要的能力。字边界是单词和空格之间的位置。非字边界是任何其他位置。下面的表达式匹配单词 Chapter 的开头三个字符，因为这三个字符出现字边界后面：

/\bCha/

**\b** 字符的位置是非常重要的。如果它位于要匹配的字符串的开始，它在单词的开始处查找匹配项。如果它位于字符串的结尾，它在单词的结尾处查找匹配项。例如，下面的表达式匹配单词 Chapter 中的字符串 ter，因为它出现在字边界的前面：

/ter\b/

下面的表达式匹配 Chapter 中的字符串 apt，但不匹配 aptitude 中的字符串 apt：

/\Bapt/

字符串 apt 出现在单词 Chapter 中的非字边界处，但出现在单词 aptitude 中的字边界处。对于 **\B** 非字边界运算符，位置并不重要，因为匹配不关心究竟是单词的开头还是结尾。

## 选择

用圆括号将所有选择项括起来，相邻的选择项之间用|分隔。但用圆括号会有一个副作用，是相关的匹配会被缓存，此时可用?:放在第一个选项前来消除这种副作用。

其中 **?:** 是非捕获元之一，还有两个非捕获元是 **?=** 和 **?!**，这两个还有更多的含义，前者为正向预查，在任何开始匹配圆括号内的正则表达式模式的位置来匹配搜索字符串，后者为负向预查，在任何开始不匹配该正则表达式模式的位置来匹配搜索字符串。

## 反向引用

对一个正则表达式模式或部分模式两边添加圆括号将导致相关匹配存储到一个临时缓冲区中，所捕获的每个子匹配都按照在正则表达式模式中从左到右出现的顺序存储。缓冲区编号从 1 开始，最多可存储 99 个捕获的子表达式。每个缓冲区都可以使用 **\n**访问，其中 n 为一个标识特定缓冲区的一位或两位十进制数。

可以使用非捕获元字符 **?:**、**?=** 或 **?!** 来重写捕获，忽略对相关匹配的保存。

反向引用的最简单的、最有用的应用之一，是提供查找文本中两个相同的相邻单词的匹配项的能力。以下面的句子为例：

Is is the cost of of gasoline going up up?

上面的句子很显然有多个重复的单词。如果能设计一种方法定位该句子，而不必查找每个单词的重复出现，那该有多好。下面的正则表达式使用单个子表达式来实现这一点：

## 实例

查找重复的单词：

var str = "Is is the cost of of gasoline going up up"; var patt1 = /\b([a-z]+) \1\b/; document.write(str.match(patt1));

[**尝试一下 »**](http://www.runoob.com/try/try.php?filename=tryjsref_regexp3)

捕获的表达式，正如 **[a-z]+** 指定的，包括一个或多个字母。正则表达式的第二部分是对以前捕获的子匹配项的引用，即，单词的第二个匹配项正好由括号表达式匹配。**\1** 指定第一个子匹配项。

字边界元字符确保只检测整个单词。否则，诸如 "is issued" 或 "this is" 之类的词组将不能正确地被此表达式识别。

正则表达式后面的全局标记 **g** 指定将该表达式应用到输入字符串中能够查找到的尽可能多的匹配。

表达式的结尾处的不区分大小写 **i** 标记指定不区分大小写。

多行标记指定换行符的两边可能出现潜在的匹配。

反向引用还可以将通用资源指示符 (URI) 分解为其组件。假定您想将下面的 URI 分解为协议（ftp、http 等等）、域地址和页/路径：

http://www.runoob.com:80/html/html-tutorial.html

下面的正则表达式提供该功能：

## 实例

输出所有匹配的数据：

var str = "https://www.runoob.com:80/html/html-tutorial.html"; var patt1 = /(\w+):\/\/([^/:]+)(:\d\*)?([^# ]\*)/; arr = str.match(patt1); for (var i = 0; i < arr.length ; i++) { document.write(arr[i]); document.write("<br>"); }

[**尝试一下 »**](http://www.runoob.com/try/try.php?filename=tryjsref_regexp4)

第一个括号子表达式捕获 Web 地址的协议部分。该子表达式匹配在冒号和两个正斜杠前面的任何单词。

第二个括号子表达式捕获地址的域地址部分。子表达式匹配 / 和 : 之外的一个或多个字符。

第三个括号子表达式捕获端口号（如果指定了的话）。该子表达式匹配冒号后面的零个或多个数字。只能重复一次该子表达式。

最后，第四个括号子表达式捕获 Web 地址指定的路径和 / 或页信息。该子表达式能匹配不包括 # 或空格字符的任何字符序列。

将正则表达式应用到上面的 URI，各子匹配项包含下面的内容：

* 第一个括号子表达式包含"http"
* 第二个括号子表达式包含"www.runoob.com"
* 第三个括号子表达式包含":80"
* 第四个括号子表达式包含"/html/html-tutorial.html"

**3、linux下安装或升级GCC4.8，以支持C++11标准**

转载 2016年02月25日 16:53:43

* 1866

++11标准在2011年8月份获得一致通过，这是自1998年后C++语言第一次大修订，对C++语言进行了改进和扩充。随后各编译器厂商都各自实现或部分实现了C++中的特性。

目标：升级GCC到4.8.2，以支持C++11。

1. 获取GCC 4.8.2包：wget [http://gcc.skazkaforyou.com/releases/gcc-4.8.2/gcc-4.8.2.tar.gz；](http://gcc.skazkaforyou.com/releases/gcc-4.8.2/gcc-4.8.2.tar.gz%EF%BC%9B)

(如果想升级到最高版本可以打开 [http://gcc.skazkaforyou.com/releases/](http://gcc.skazkaforyou.com/releases/gcc-4.8.2/gcc-4.8.2.tar.gz%EF%BC%9B) 查看，将链接修改成最高版本的就可以)

2. 解压缩：tar -xf gcc-4.8.2.tar.gz；

3. 进入到目录gcc-4.8.2，运行：./contrib/download\_prerequisites。这个神奇的脚本文件会帮我们下载、配置、安装依赖库，可以节约我们大量的时间和精力。

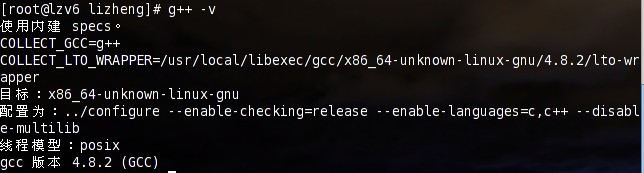
4. 编译安装 gcc

gcc 编译参数说明:  
--prefix                        //默认 gcc 的安装目录，CentOS-6.5 下面最好安装到 /usr 下，不然可能会有问题  
--enable-languages        //指定 gcc 能编译哪些语言的文件，每种语言用逗号分隔, 例如 c,c++,java  
--disable-multilib           //默认gcc 能在32位系统上将代码编译成64位程序，或者在64位系统上编译成32位程序，如果加上这个编译选项则表示关闭这个gcc的交叉编译功能。

# tar -zxvf gcc-4.8.2.tar.gz  
# cd gcc-4.8.2  
# mkdir build  
# cd build  
# ../configure --prefix=/usr  
# make && make install

注意: 当 gmp、mpfr、mpc 编译的时候是自定义指定的安装路径，在编译 gcc 时需要加上下面的选项：  
--with-mpc=MPC的安装目录  
--with-mpfr=MPFR的安装目录  
--with-gmp=GMP的安装目录

5. 验证：gcc -v；或者g++ -v，如果显示的gcc版本仍是以前的版本，就需要重启系统；或者可以查看gcc的安装位置：which gcc；然后在查看版本 /usr/local/bin/gcc -v，通常gcc都安装在该处位置，如果显示为；

[](http://static.oschina.net/uploads/img/201403/17174520_8GyN.jpg)

则表示已经升级成功。