10、搜索命令

whereis 命令

whereis 是搜索系统命令的命令（像绕口令一样），也就是说，whereis 命令不能搜索普通文件，而只能搜索系统命令。

作用：能够超除命令的二进制源文件的位置和帮助文档的位置

which 命令

which 也是搜索系统命令的命令。和 whereis 命令的区别在于他们的显示结果不同：

which 命令作用：二进制命令的位置，如果这个命令有别名，则还可以显示别名。

find命令

因为Linux下面一切皆文件，经常需要搜索某些文件来配置，所以对于linux来说find是一条很重要的命令。linux下面的find指令用于在目录结构中搜索文件，并执行指定的操作。它提供了相当多的查找条件，功能很强大。在不指定查找目录的情况下，find会在对整个系统进行遍历。

所以：find命令是一个非常耗时，耗资源的命令，一定记住：

1、不能在系统繁忙时段运行；

2、尽可能在最小的搜索范围的前提下使用

1、命令格式

```
find [查找目录] [查找规则] [查找完后的操作]
```

例子：

通过文件名字查找，如名字为test的文件或目录，这个是精准查找

```
find ./ -name test
```

加通配符，查找名字包含test的文件或目录，这个是模糊查找

```
find ./ -name *test*        //  * 表示任意字符， ？表示一个字符
```

不区分大小写的查找：find ./ -iname *test*

查询文件大小大于100M的文件

```
find ./ -size  +204800    //  注意这里的单位是数据块 ，  它和K的换算：  1数据块=215字节=0.5K    所以100M=102400K=20800，
                    // +表示大于，-表示小于，不写表示等于
```

查询所有者为xxx的所拥有文件

```
find /  -user  xxxx
```

查询用户组为xxx的所拥有文件

```
find /  -group xxxx
```

查询在etc目录下5分钟内被修改过文件属性的文件和目录

```
find /etc  -cmin  -5    // amin被访问， cmin属性被修改，mmin内容被修改
```

多条件查询，在/etc目录下查找文件大小大于80M,并且小于100M的文件

```
find /etc -size +163840  -a  -size -204800  //  -a:表示and，并且关系， 此外还有-o：表示or，或者关系
```

默认查找的内容是目录和文件，但是我们只想找到文件或者目录中的一个，如：查找/etc目录下的init开头的文件

```
fine /etc -name init* -a -type f    //这里f:表示文件，d:表示目录， l：表示软链接
```

查找文件后，在进一步执行某些命令，如：查找出文件后在显示详细属性信息

```
find /etc -name init*  -exec ls -l {}  \;
```

// {} \; 后面的这三个符号，你把它当作固定结构记住就行，其中“{}”就代表 find 命令的查找结果。注意-exec可以替换-ok，功能一样只是多一个提示

通过i节点查询文件,举个例子就好了：

```
find /etc -inum 12313 -exec rm {} \;
```

grep命令

grep是一种强大的文本搜索工具，它和find命令的主要区别就是，find是搜索文件/目录本身，也就是说我们在一个大目录里找子目录或在大目录里的文件，而grep命令呢？它不是找文件/目录本身，它是搜索在文件里边的内容！

grep的工作方式是这样的，它在一个或多个文件中搜索与字符串模板（正则表达式）匹配的内容。如果模板包括空格，则必须打引号，模板后的所有字符串被看作文件名。搜索的结果被送到标准输出，不影响原文件内容。

grep可用于shell脚本，因为grep通过返回一个状态值来说明搜索的状态，如果模板搜索成功，则返回0，如果搜索不成功，则返回1，如果搜索的文件不存在，则返回2。我们利用这些返回值就可进行一些自动化的文本处理工作。讲shell编程的部分，在细说！

1. 命令格式：

```
grep  [option]  pattern  file
```

2. 常用命令参数：

-E ：开启扩展（Extend）的正则表达式。grep -E ==== egrep

-i ：忽略大小写（ignore case）。

-v ：反过来（invert），只打印没有匹配的，而匹配的反而不打印。

-n ：显示行号

-w ：被匹配的文本只能是单词，而不能是单词中的某一部分，如文本中有liker，而我搜寻的只是like，就可以使用-w选项来避免匹配liker

-c ：显示总共有多少行被匹配到了，而不是显示被匹配到的内容，**注意如果同时使用-cv选项是显示有多少行没有被匹配到。**

-o ：只显示被模式匹配到的字符串。

--color :将匹配到的内容以颜色高亮显示。

-A n：显示匹配到的字符串所在的行及其后n行，after

-B n：显示匹配到的字符串所在的行及其前n行，before

-C n：显示匹配到的字符串所在的行及其前后各n行，context

案例1：基本使用方式

```
[root@CentOS7 ~]# grep "abc" ./grepdemo.txt    // 找文件中包含“abc”这三个字符的内容行
```

案例2：查询结果忽略大小写

```
[root@CentOS7 ~]# grep  -i "abc" ./grepdemo.txt
```

案例3：查询结果显示行号

```
[root@CentOS7 ~]# grep -n "abc" ./grepdemo.txt    //显示的行号，不是给结果编一个新的行号，是源文件中的行号
```

案例4：显示有几行没被匹配到

```
[root@CentOS7 ~]# grep -cv "abc" ./grepdemo.txt
```

案例5：只显示与搜索匹配的内容

```
[root@CentOS7 ~]# grep -o "abc" ./grepdemo.txt
```

案例6：显示匹配的内容以及它的前n行，匹配的内容以及它的后n行，匹配的内容以及它的前后n行

```
[root@CentOS7 ~]# grep -A 2 "abc" ./grepdemo.txt
[root@CentOS7 ~]# grep -B 2 "abc" ./grepdemo.txt
[root@CentOS7 ~]# grep -C 2 "abc" ./grepdemo.txt
```

3、模式部分：

1、直接输入要匹配的字符串，这个可以用fgrep（fast grep）代替来提高查找速度，比如[root@CentOS7 ~]# fgrep "abc" ./grepdemo.txt

2、使用**基本正则表达式**，下面谈关于基本正则表达式的使用：

匹配字符：

. ：任意一个字符。

例子：[root@CentOS7 ~]# grep "abc." ./grepdemo.txt

[abc] ：表示匹配一个字符，这个字符必须是abc中的一个。

例子：[root@CentOS7 ~]# grep "[abc]" ./grepdemo.txt

[a-zA-Z] ：表示匹配一个字符，这个字符必须是a-z或A-Z这52个字母中的一个。

例子：[root@CentOS7 ~]# grep "[A-Z]" ./grepdemo.txt

[^123] ：匹配一个字符，这个字符是除了1、2、3以外的所有字符。

例子：[root@CentOS7 ~]# grep "abc[^A-Za-z]" ./grepdemo.txt

对于一些常用的字符集，系统做了定义：

[A-Za-z] 等价于 [[:alpha:]]

[0-9] 等价于 [[:digit:]]

[A-Za-z0-9] 等价于 [[:alnum:]]

tab,space 等所有空白字符 [[:space:]]

[A-Z] 等价于 [[:upper:]]

[a-z] 等价于 [[:lower:]]

标点符号 [[:punct:]]

例子：[root@CentOS7 ~]# grep "[[:punct:]]" ./grepdemo.txt

例子：[root@CentOS7 ~]# grep "abc[^[:upper:]]" ./grepdemo.txt

匹配次数：

\{m,n\} ：匹配其前面出现的字符至少m次，至多n次。

\? ：匹配其前面出现的内容0次或1次，等价于\{0,1\}。

* ：匹配其前面出现的内容0次或多次，匹配其前面出现的内容任意次，等价于\{0,\}，所以 ".*" 表述任意字符任意次，即

无论什么内容全部匹配。

例子：搜索/etc/passwd文件中以/开始，以sh结尾的字符串

[root@CentOS7 ~]# grep "/.*sh" /etc/passwd

root:x:0:0:root:/root:/bin/bash

注意：默认情况下，正则表达式的匹配工作在贪婪模式下，也就是说它会尽可能长地去匹配，所有这里不会

匹配/bash，匹配/bin/bash ，而是会匹

配/root:/bin/bash

例子：搜索/etc/passwd文件中以/开始，以sh结尾的字符串，在/和sh中间可以有0-2个任意字符

```
[root@CentOS7 ~]# grep "/.\{0,2\}sh" /etc/passwd
```

位置锚定：

^ ：锚定行首，[^]:范围的取反，用在[]的外面表示：行首

例子：搜索以root开始的行

```
[root@CentOS7 ~]# grep "^root" /etc/passwd
```

$ ：锚定行尾。技巧："^$"用于匹配空白行。

例子：搜索以h结尾的行

```
[root@CentOS7 ~]# grep "h$" /etc/passwd
```

\b或\<：锚定单词的词首。如"\blike"不会匹配alike，但是会匹配liker

```
[root@CentOS7 ~]# grep "\bsh" /etc/passwd
```

\b或\>：锚定单词的词尾。如"\blike\b"不会匹配alike和liker，只会匹配like

```
[root@CentOS7 ~]# grep "sh\>" /etc/passwd
```

\B ：与\b作用相反。

```
[root@CentOS7 ~]# grep "\Bsh\B" /etc/passwd
```

找sh，但是不能在单词的开头也不能在结尾

分组及引用：（1+2）*3

\(string\) ：将string作为一个整体方便后面引用

\1 ：引用第1个左括号及其对应的右括号所匹配的内容。

\2 ：引用第2个左括号及其对应的右括号所匹配的内容。

\n ：引用第n个左括号及其对应的右括号所匹配的内容。

例子：在/etc/passwd中找到开始字母和结尾字母一样的行

```
[root@CentOS7 ~]# grep "^\([a-z]\).*\1$" /etc/passwd
```

3、扩展的（Extend）正则表达式（注意要使用扩展的正则表达式要加-E选项，或者直接使用egrep）我用正则更喜欢用这个一点：

匹配字符：这部分和基本正则表达式一样

匹配次数：

* ：和基本正则表达式一样

? ：基本正则表达式是\?，二这里没有\。
{m,n} ：相比基本正则表达式也是没有了\。
+ ：匹配其前面的字符至少一次，相当于{1,}。

位置锚定：和基本正则表达式一样。

分组及引用：
(string) ：相比基本正则表达式也是没有了\。
\1 ：引用部分和基本正则表达式一样。
\n ：引用部分和基本正则表达式一样。

或者：
[ab]==a|b ：匹配a或b，注意a是指 | 的左边的整体，b也同理。比如C|cat 表示的是 C或cat，而不是Cat或cat Cat|cat，如果要表示Cat或cat，则应该写为(C|c)at 。

记住(string)除了用于引用还用于分组。

注意：所有的正则字符，也就是正则里有特殊意义的符号字符，如 [、* 、(等，如要搜索 * 本身，而不是想把 * 解释为重复先前字符任意次，可以使用* 来转义。

最后一个例子：在网络配置文件 /etc/sysconfig/network-scripts/ifcfg-ens33 中检索出所有的 IP

首先分析ip的组成规则：192.168.1.56

```
egrep "(\b[1-9][0-9]{0,2}\.){3,3}\b[1-9][0-9]{0,2}" /etc/sysconfig/network-scripts/ifcfg-ens33
```

locate命令

1、命令简介

locate(locate) 命令用来查找文件，在这点上和find命令一样。 locate命令要比find -name快得多，原因在于它不搜索具体目录，而是搜索一个数据库/var/lib/mlocate/mlocate.db 。这个数据库中含有本地所有文件信息。Linux系统自动创建这个数据库，并且每天自动更新一次，因此，我们在用locate 查找文件时，有时会找到已经被删除的数据，或者刚刚建立文件，却无法查找到，原因就是因为数据库文件没有被更新。为了避免这种情况，可以在使用locate之前，先使用updatedb命令，手动更新数据库。

2、用法

locate [选项]... [参数]...

-i， 忽略大小写

3、示例

演示之前先要安装：CentOS7默认没有安装该命令

1、安装"locate"命令即可。运行“yum install mlocate”命令

2、mlocate已安装完成。接下来需更新后台数据库，输入命令：updatedb。

这里需要注意一点：如果没有updatedb更新后台数据库，直接输入查找命令：locate，还是未能找到命令。原因是Linux不是实时更新它的后台数据库，所以我们并不能马上执行。

3、上面2步后，locate命令就可以使用了！

示例1： 搜索etc目录下所有以my开头的文件

```
[root@cent6 lib]# locate /etc/my      #注意整个命令类似find /etc/  -name my*，但也有区别
/etc/my.cnf
```

示例2：新增的文件无法locate，使用updatedb

```
[root@cent6 ~]# touch new.txt
[root@cent6 ~]# locate new.txt
[root@cent6 ~]# updatedb
[root@cent6 ~]# locate new.txt
/root/new.txt
```

示例3：updatedb的配置文件/etc/updatedb.conf

```
[root@cent6 ~]# cat /etc/updatedb.conf
PRUNE_BIND_MOUNTS = "yes"
PRUNEFS = "9p afs anon_inodefs auto autofs bdev binfmt_misc cgroup cifs coda configfs cpuset debugfs devpts ecryptfs exofs fuse fusectl gfs gfs2 hugetlbfs inotifyfs iso9660 jffs2 lustre mqueue ncpfs nfs nfs4 nfsd pipefs proc ramfs rootfs rpc_pipefs securityfs selinuxfs sfs sockfs sysfs tmpfs ubifs udf usbfs"
PRUNENAMES = ".git .hg .svn"
PRUNEPATHS = "/afs /media /net /sfs /tmp /udev /var/cache/ccache /var/spool/cups /var/spool/squid /var/tmp"
```

第一行PRUNE_BIND_MOUNTS="yes"的意思是：是否进行限制搜索。

第二行是排除检索的文件系统类型，即列出的文件系统类型不进行检索。

第三行表示对哪些后缀的文件排除检索，也就是列在这里面的后缀的文件跳过不进行检索。不同后缀之间用空格隔开。

第四行是排除检索的路径，即列出的路径下的文件和子文件夹均跳过不进行检索。

updatedb之后使用locate仍然找不到想要文件

看这个文件主要目的：检查挂载的目录是否被忽略了，如果需要搜索挂载的目录，自己开启