第1章 通配符号

1.什么是通配符

```
1.通配符的作用是查找和匹配文件名称而不是文件里的内容!
2.通配符是shell的内置功能
3.Linux大部分命令都支持通配符
```

2.常用特殊说明

```
*           匹配任意(0个或多个)字符或字符串，包含空字符串
?           匹配任意1个字符，有且只有一个字符
[abcd]      匹配[abcd]中任何一个字符，abcd也可是其他任意不连续字符
[a-z]       匹配a-z之间的任意一个字符，字符前后要练习，也可以是连续的数字，即[1-9]
[!abcd]     不匹配括号里面的任意字符，也可以写作[!a-d]，这里的!可以用^替代，即[^abcd]
```

3.通配符练习

3.1 创建测试环境

```
mkdir test
cd test/
touch oldboy.txt oldgirl.txt test.txt oldzhang.sh
ls
```

3.2 * 匹配任意字符

```
#1.查看所有结尾为txt的文件
[root@centos7 test]# ls *.txt
oldboy.txt  oldgirl.txt  test.txt

#2.查看以sh结尾的文件
[root@centos7 test]# ls *.sh
oldzhang.sh

#3.查找以old开头，txt结尾的文件
[root@centos7 test]# ls old*.txt
oldboy.txt  oldgirl.txt
```

3.3 ? 匹配任意一个字符--用的不多，了解即可

```
#1.尝试匹配?.sh
[root@centos7 test]# ls ?.sh
ls: 无法访问?.sh: 没有那个文件或目录      #这里报错了，为什么呢，因为?只代表任意一个字符，因此没有符合条件的文件

#2.创建一个符合条件的文件再次查找
[root@centos7 test]# touch a.sh
[root@centos7 test]# ls ?.sh
```

```
a.sh

#3.那么如何查找特定字符长度的文件名呢？
[root@centos7 test]# ls *.sh
a.sh  oldzhang.sh
[root@centos7 test]# ls ????????.sh          #使用8个?就匹配了
oldzhang.sh
```

3.4 [] 匹配指定的内容

```
#1.查找匹配a-z任意一个字符的以.sh结尾的文件
[root@centos7 test]# ls [a-z].sh
a.sh

#2.匹配以old开头后 + xyz任意一个字符 + hang.sh结尾的文件
[root@centos7 test]# ls old[xyz]hang.sh
oldzhang.sh

#3.复杂一点的匹配，起始很好理解
[root@centos7 test]# ls old[a-z]???g.sh
oldzhang.sh
```

3.5 [!abcd] 取反匹配指定的内容

```
#1.排除所有以o开头的任意名称并且以.txt结尾的文件
[root@centos7 test]# ls [!o]*.txt
test.txt

#2.排除所有以o开头的任意名称并且以.sh结尾的文件
[root@centos7 test]# ls [!o]*.sh
a.sh

#3.排除所有以a开头的任意名称并且以.sh结尾的文件
[root@centos7 test]# ls [!a]*.sh
oldzhang.sh
```

3.6 通配符综合练习

```
#1.查找/etc目录下包含hosts字符串的所有文件
[root@centos7 ~]# find /etc -type f -name "*hosts*"
/etc/hosts
/etc/hosts.allow
/etc/hosts.deny

#2.查找/etc目录下所有网卡配置文件
[root@centos7 ~]# find /etc -type f -name "ifcfg-*"
/etc/sysconfig/network-scripts/ifcfg-lo
/etc/sysconfig/network-scripts/ifcfg-eth0

#3.只查找包含了数字的网卡配置
[root@centos7 ~]# find /etc -type f -name "ifcfg-*[0-9]"
/etc/sysconfig/network-scripts/ifcfg-eth0

#4.查找/dev目录下第1块到第4块磁盘文件
```

```
[root@centos7 ~]# find /dev/ -name "sd[a-d]"
/dev/sdd
/dev/sdc
/dev/sdb
/dev/sda

#5.查找/dev/sda一共有几个分区
[root@centos7 ~]# find /dev/ -name "sda?"
/dev/sda2
/dev/sda1
[root@centos7 ~]# find /dev/ -name "sda[0-9]"
/dev/sda2
/dev/sda1
[root@centos7 ~]# find /dev/ -name "sda*"
/dev/sda2
/dev/sda1
/dev/sda
```

第2章 特殊符号

1.什么是特殊符号

```
相比通配符来说，Linux特殊符号更加复杂，且杂乱无章，但是， 要想成为一个合格的Linux运维工程师，就必须要掌握这些特殊符号。
```

2.常用特殊符号说明

2.1 路径相关

```
~    用户家目录
-    上一次目录
.    当前目录
..   上一层目录
```

2.2 引号相关

```
''    单引号，所见即所得
""    双引号，可以解析变量和引用命令
``    反引号，可以解析命令
      无引号，类似于双引号，但是有歧义
```

2.3 重定向符号

```
<       标准输入
>       标准正确重定向
>>      标准正确追加重定向
2>      标准错误重定向
2>>     标准错误追加重定向
```

2.4 命令运行

```
&&      前面执行成功后面才执行
||      前面执行失败后面才执行
;       不管前面是否执行成功，都执行后面的命令
\       转义特殊字符，还原字符本来的含义
$()     优先执行小括号里的命令，并且结果可以被其他命令使用
``      同$()效果一样，但是容易和单双引号搞混
$?      上一条命令执行的结果
|       管道
{}      生成序列
```

3.引号实验

3.1 单引号

使用时间命令作为测试

```
[root@centos7 ~]# date
2021年 04月 12日 星期一 21:48:21 CST
[root@centos7 ~]# echo 'date'
date
[root@centos7 ~]# echo '$?'
$?
[root@centos7 ~]# echo '$PATH'
$PATH
```

结论：

```
单引号里的内容所见即所得，单引号引起来的内容是什么，输出就是什么，不会任何改变，特殊符号也会失去特殊含义。
```

3.2 反引号

```
[root@centos7 ~]# echo `date`
2021年 04月 12日 星期一 21:55:39 CST
[root@centos7 ~]# echo `pwd`
/root
[root@centos7 ~]#
```

结论：

```
简单来说,当将待处理的字符串用反引号引起来的时候，系统首先会将反引号里字符串当作命令进行解析，然后针对解析后的结果做进一步的处理。
```

3.3 双引号

```
[root@centos7 ~]# echo "date"
date
[root@centos7 ~]# echo "`date`"
2021年 04月 12日 星期一 21:55:11 CST

[root@centos7 ~]# echo "$PATH"
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

结论:

```
1.当输出双引号内的所有内容时，如果内容中有命令(要反引一 下)、变量、特殊转义符等，则会先将变量、命令、转义字符解析出来，然后输出最终的内容。
2.若在平时引用字符串时，不知道应该如何引用，则可以默认使用双引号
```

3.4 无引号

```
[root@centos7 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@centos7 ~]# echo $?
0
```

结论:

```
不使用引号的功能与双引号类似，但由于没有引号，很难确定字符串的边界，因此很容易出现各种未知的操作错误
```

4.特殊符号练习

因为路径和重定向在基础命令阶段已经讲过，所以这里不再练习这两块内容。

4.1 && 前面命令执行成功才会执行后面的命令

```
#1.创建目录执行成功，就进入到这个目录里
[root@centos7 ~]# mkdir /data && cd /data
[root@centos7 data]#

#2.如果前面的命令执行失败，不会执行后面的命令
[root@centos7 ~]# pwd
/root
[root@centos7 ~]# mkdir /data && cd /data
mkdir: 无法创建目录"/data": 文件已存在
[root@centos7 ~]# pwd
/root
```

4.2 || 前面命令执行不成功才会执行后面的命令

```
#1.如果目录已经存在则切换到/opt目录下
[root@centos7 ~]# mkdir /data || cd /opt
mkdir: 无法创建目录"/data": 文件已存在
[root@centos7 opt]# pwd
/opt

#2.如果创建目录成功，就不执行后面的命令
[root@centos7 ~]# mkdir /data3 || cd /opt
[root@centos7 ~]# pwd
/root
```

4.3 ; 分隔多个命令

```
[root@centos7 ~]# mkdir /data;echo 123;cd /opt/;pwd
mkdir: 无法创建目录"/data": 文件已存在
123
/opt
[root@centos7 opt]#
```

4.4 $() 引用命令

```
#1.引用时间命令创建文件，命名为时间.txt
[root@centos7 ~]# touch $(date +%F).txt
[root@centos7 ~]# ll
总用量 0
-rw-r--r-- 1 root root 0 4月  12 21:44 2021-04-12.txt

#2.rm结合find命令删除文件
[root@centos7 ~]# find . -type f -name "*.txt"
./2021-04-12.txt
[root@centos7 ~]# rm -rf $(find . -type f -name "*.txt")
[root@centos7 ~]# ll
总用量 0
[root@centos7 ~]#
```

4.5 {} 生成序列

```
#1.生成序列
[root@centos7 ~]# echo {1..10}
1 2 3 4 5 6 7 8 9 10
[root@centos7 ~]# echo {a..g}
a b c d e f g

#2.特殊应用
[root@centos7 ~]# cp /etc/profile{,.bak}
[root@centos7 ~]# ll /etc/profile*
-rw-r--r--. 1 root root 1835 12月 13 11:25 /etc/profile
-rw-r--r--  1 root root 1835 4月  13 08:11 /etc/profile.bak

#3.变量分隔符
[root@centos7 ~]# name=zhang
[root@centos7 ~]# echo $name
zhang
[root@centos7 ~]# echo $name_ya
```

```
18
19 [root@centos7 ~]# echo ${name}_ya
20 zhang_ya
```