

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

- | | | |
|---|---------|--|
| 1 | * | 匹配任意(0个或多个)字符或字符串，包含空字符串 |
| 2 | ? | 匹配任意1个字符，有且只有一个字符 |
| 3 | [abcd] | 匹配[abcd]中任何一个字符，abcd也可能是其他任意不连续字符 |
| 4 | [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] |
| 5 | [!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

```
1 mkdir test
2 cd test/
3 touch oldboy.txt oldgirl.txt test.txt oldzhang.sh
4 ls
```

3.2 * 匹配任意字符

```
1 #1.查看所有结尾为txt的文件
2 [root@centos7 test]# ls *.txt
3 oldboy.txt oldgirl.txt test.txt
4
5 #2.查看以sh结尾的文件
6 [root@centos7 test]# ls *.sh
7 oldzhang.sh
8
9 #3.查找以old开头，txt结尾的文件
10 [root@centos7 test]# ls old*.txt
11 oldboy.txt oldgirl.txt
```

3.3 ? 匹配任意一个字符--用的不多，了解即可

```
1 #1.尝试匹配?.sh
2 [root@centos7 test]# ls ?.sh
3 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
4
5 #2.创建一个符合条件的文件再次查找
6 [root@centos7 test]# touch a.sh
7 [root@centos7 test]# ls ?.sh
```

```
8 a.sh
9
10 #3.那么如何查找特定字符长度的文件名?
11 [root@centos7 test]# ls *.sh
12 a.sh oldzhang.sh
13 [root@centos7 test]# ls ?????????.sh #使用8个?就匹配了
14 oldzhang.sh
```

3.4 [] 匹配指定的内容

```
1 #1.查找匹配a-z任意一个字符的以.sh结尾的文件
2 [root@centos7 test]# ls [a-z].sh
3 a.sh
4
5 #2.匹配以old开头后 + xyz任意一个字符 + hang.sh结尾的文件
6 [root@centos7 test]# ls old[xyz]hang.sh
7 oldzhang.sh
8
9 #3.复杂一点的匹配,起始很好理解
10 [root@centos7 test]# ls old[a-z]???g.sh
11 oldzhang.sh
```

3.5 [!abcd] 取反匹配指定的内容

```
1 #1.排除所有以o开头的任意名称并且以.txt结尾的文件
2 [root@centos7 test]# ls [!o]*.txt
3 test.txt
4
5 #2.排除所有以o开头的任意名称并且以.sh结尾的文件
6 [root@centos7 test]# ls [!o]*.sh
7 a.sh
8
9 #3.排除所有以a开头的任意名称并且以.sh结尾的文件
10 [root@centos7 test]# ls [!a]*.sh
11 oldzhang.sh
```

3.6 通配符综合练习

```
1 #1.查找/etc目录下包含hosts字符串的所有文件
2 [root@centos7 ~]# find /etc -type f -name "*hosts*"
3 /etc/hosts
4 /etc/hosts.allow
5 /etc/hosts.deny
6
7 #2.查找/etc目录下所有网卡配置文件
8 [root@centos7 ~]# find /etc -type f -name "ifcfg-*"
9 /etc/sysconfig/network-scripts/ifcfg-lo
10 /etc/sysconfig/network-scripts/ifcfg-eth0
11
12 #3.只查找包含了数字的网卡配置
13 [root@centos7 ~]# find /etc -type f -name "ifcfg-*[0-9]"
14 /etc/sysconfig/network-scripts/ifcfg-eth0
15
16 #4.查找/dev目录下第1块到第4块磁盘文件
```

```
17 [root@centos7 ~]# find /dev/ -name "sd[a-d]"
18 /dev/sdd
19 /dev/sdc
20 /dev/sdb
21 /dev/sda
22
23 #5. 查找/dev/sda一共有几个分区
24 [root@centos7 ~]# find /dev/ -name "sda?"
25 /dev/sda2
26 /dev/sda1
27 [root@centos7 ~]# find /dev/ -name "sda[0-9]"
28 /dev/sda2
29 /dev/sda1
30 [root@centos7 ~]# find /dev/ -name "sda*"
31 /dev/sda2
32 /dev/sda1
33 /dev/sda
```

第2章 特殊符号

1. 什么是特殊符号

- 1 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，就必须掌握这些特殊符号。

2. 常用特殊符号说明

2.1 路径相关

- | | | |
|---|----|-------|
| 1 | ~ | 用户家目录 |
| 2 | - | 上一次目录 |
| 3 | . | 当前目录 |
| 4 | .. | 上一层目录 |

2.2 引号相关

- | | | |
|---|----|------------------|
| 1 | '' | 单引号，所见即所得 |
| 2 | "" | 双引号，可以解析变量和引用命令 |
| 3 | `` | 反引号，可以解析命令 |
| 4 | | 无引号，类似于双引号，但是有歧义 |

2.3 重定向符号

- | | | |
|---|-----|-----------|
| 1 | < | 标准输入 |
| 2 | > | 标准正确重定向 |
| 3 | >> | 标准正确追加重定向 |
| 4 | 2> | 标准错误重定向 |
| 5 | 2>> | 标准错误追加重定向 |

2.4 命令运行

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

3.引号实验

3.1 单引号

使用时间命令作为测试

```
1 [root@centos7 ~]# date
2 2021年 04月 12日 星期一 21:48:21 CST
3 [root@centos7 ~]# echo 'date'
4 date
5 [root@centos7 ~]# echo '$?'
6 $_
7 [root@centos7 ~]# echo '$PATH'
8 $PATH
```

结论:

- 1 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

```
1 [root@centos7 ~]# echo `date`
2 2021年 04月 12日 星期一 21:55:39 CST
3 [root@centos7 ~]# echo `pwd`
4 /root
5 [root@centos7 ~]#
```

结论:

- 1 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

```
1 [root@centos7 ~]# echo "date"
2 date
3 [root@centos7 ~]# echo "`date`"
4 2021年 04月 12日 星期一 21:55:11 CST
5
6 [root@centos7 ~]# echo "$PATH"
7 /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

```
1 [root@centos7 ~]# echo $PATH
2 /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
3 [root@centos7 ~]# echo $?
4 0
```

结论:

- 1 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 #1.创建目录执行成功, 就进入到这个目录里
2 [root@centos7 ~]# mkdir /data && cd /data
3 [root@centos7 data]#
4
5 #2.如果前面的命令执行失败, 不会执行后面的命令
6 [root@centos7 ~]# pwd
7 /root
8 [root@centos7 ~]# mkdir /data && cd /data
9 mkdir: 无法创建目录"/data": 文件已存在
10 [root@centos7 ~]# pwd
11 /root
```

4.2 || 前面命令执行不成功才会执行后面的命令

```
1 #1.如果目录已经存在则切换到/opt目录下
2 [root@centos7 ~]# mkdir /data || cd /opt
3 mkdir: 无法创建目录"/data": 文件已存在
4 [root@centos7 opt]# pwd
5 /opt
6
7 #2.如果创建目录成功,就不执行后面的命令
8 [root@centos7 ~]# mkdir /data3 || cd /opt
9 [root@centos7 ~]# pwd
10 /root
```

4.3 ; 分隔多个命令

```
1 [root@centos7 ~]# mkdir /data;echo 123;cd /opt;/pwd
2 mkdir: 无法创建目录"/data": 文件已存在
3 123
4 /opt
5 [root@centos7 opt]#
```

4.4 \$() 引用命令

```
1 #1.引用时间命令创建文件,命名为时间.txt
2 [root@centos7 ~]# touch $(date +%F).txt
3 [root@centos7 ~]# ll
4 总用量 0
5 -rw-r--r-- 1 root root 0 4月 12 21:44 2021-04-12.txt
6
7 #2.rm结合find命令删除文件
8 [root@centos7 ~]# find . -type f -name "*.txt"
9 ./2021-04-12.txt
10 [root@centos7 ~]# rm -rf $(find . -type f -name "*.txt")
11 [root@centos7 ~]# ll
12 总用量 0
13 [root@centos7 ~]#
```

4.5 {} 生成序列

```
1 #1.生成序列
2 [root@centos7 ~]# echo {1..10}
3 1 2 3 4 5 6 7 8 9 10
4 [root@centos7 ~]# echo {a..g}
5 a b c d e f g
6
7 #2.特殊应用
8 [root@centos7 ~]# cp /etc/profile{,.bak}
9 [root@centos7 ~]# ll /etc/profile*
10 -rw-r--r-- 1 root root 1835 12月 13 11:25 /etc/profile
11 -rw-r--r-- 1 root root 1835 4月 13 08:11 /etc/profile.bak
12
13 #3.变量分隔符
14 [root@centos7 ~]# name=zhang
15 [root@centos7 ~]# echo $name
16 zhang
17 [root@centos7 ~]# echo $name_ya
```

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