

SHELL DAY04



Shell脚本编程

NSD SHELL

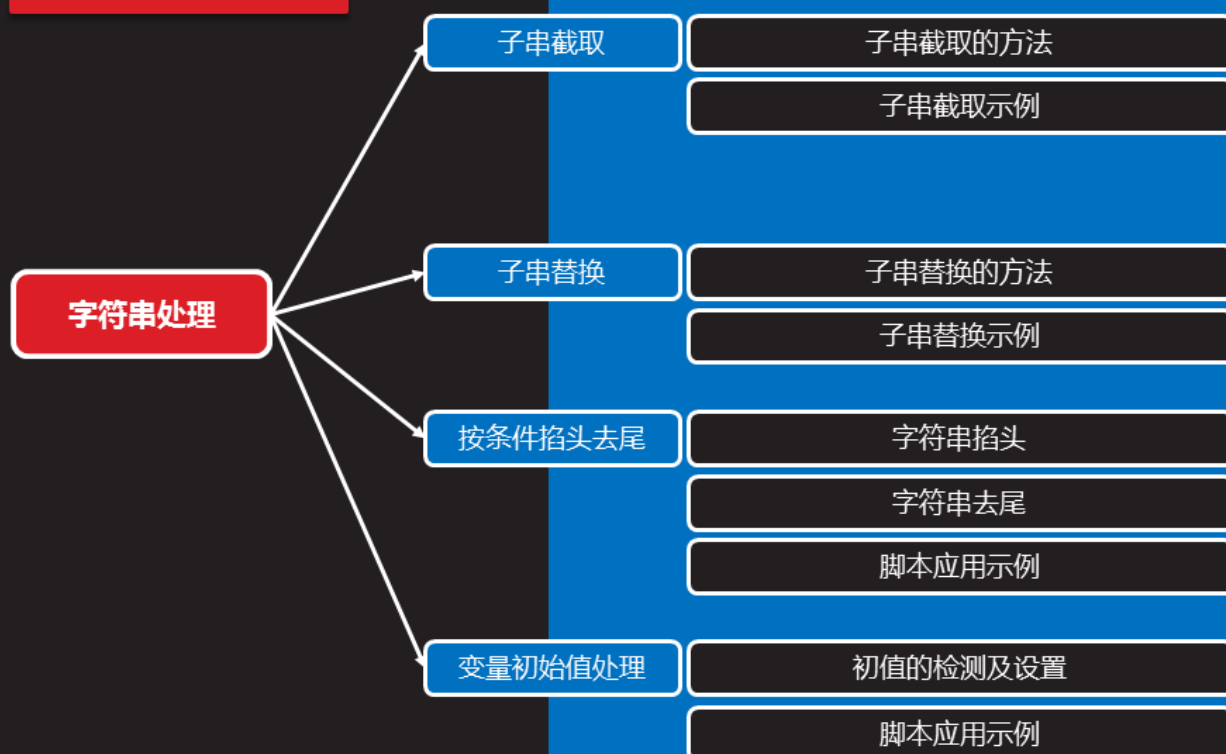
DAY04

内容

上午	09:00 ~ 09:30	作业讲解与回顾
	09:30 ~ 10:20	字符串处理
	10:30 ~ 11:20	
	11:30 ~ 12:00	扩展的脚本技巧
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	正则表达式
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



字符串处理



子串截取

子串截取的方法

知识讲解

- 方法一，使用 `${}` 表达式
 - 格式：`${var:起始位置:长度}`
 - 编号从0开始，可省略
- 方法二，使用 `expr substr`
 - 格式：`expr substr "$var" 起始位置 长度`
 - 编号均从1开始
- 方法三，使用 `cut` 工具
 - 格式：`echo $var | cut -b 起始位置-结束位置`



子串截取示例

知识讲解

- 任务目标

- 截取变量NM的前6个字符

```
[root@svr5 ~]# NM="Tarena IT Group."
```

```
[root@svr5 ~]# echo ${NM:0:6}
```

```
Tarena
```

//采用方法1，与 \${NM::6} 等效

```
[root@svr5 ~]# expr substr "$NM" 1 6
```

```
Tarena
```

//采用方法2

```
[root@svr5 ~]# echo $NM | cut -b 1-6
```

```
Tarena
```

//采用方法3，与 cut -b -6 等效



子串替换

子串替换的方法

知识讲解

- 只替换第1个匹配结果
 - 格式：`${var/old/new}`
- 替换全部匹配结果
 - 格式：`${var//old/new}`



按条件掐头去尾

字符串去尾

知识讲解

- 从右向左，最短匹配删除
 - 格式：`${变量名%关键词*}`
 - 从右向左，最长匹配删除
 - 格式：`${变量名%%关键词*}`
- % 用来删除头部，* 通配

```
[root@svr5 ~]# MDIR="/var/spool/mail/root"
[root@svr5 ~]# echo ${MDIR%o*}
/var/spool/mail/ro                //删除到最近匹配
[root@svr5 ~]# echo ${MDIR%%o*}
/var/sp                            //删除到最远匹配
```

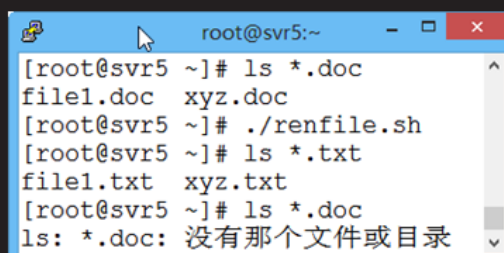


脚本应用示例

- 任务目标
 - 实现批量改名，将扩展名 .doc 改为 .txt

知识讲解

```
[root@svr5 ~]# cat renfile.sh
#!/bin/bash
for FILE in *.doc
do
    mv $FILE ${FILE%.doc}.txt
done
```



```
root@svr5:~
[root@svr5 ~]# ls *.doc
file1.doc  xyz.doc
[root@svr5 ~]# ./renfile.sh
[root@svr5 ~]# ls *.txt
file1.txt  xyz.txt
[root@svr5 ~]# ls *.doc
ls: *.doc: 没有那个文件或目录
```



案例1：字符串截取及切割

1. 参考PPT示范操作，完成子串截取、替换等操作
2. 根据课上的批量改名脚本，编写改进版renfilex.sh：
 - 1) 能够批量修改文件的扩展名
 - 2) 修改前/后的扩展名通过位置参数\$1、\$2提供

课堂练习



变量初始值处理

初值的检测及设置

- 取值，`${var:-word}`
 - 若变量`var`已存在且非Null，则返回 `$var` 的值
 - 否则返回字符串 “word”，变量`var`值不变

知识讲解

```
[root@svr5 ~]# NM="Tarena IT Group."
```

```
[root@svr5 ~]# echo ${NM:-Tarena}
```

```
Tarena IT Group.
```

//变量NM已设置

```
[root@svr5 ~]# unset NM
```

//清除NM变量

```
[root@svr5 ~]# echo ${NM:-Tarena}
```

```
Tarena
```

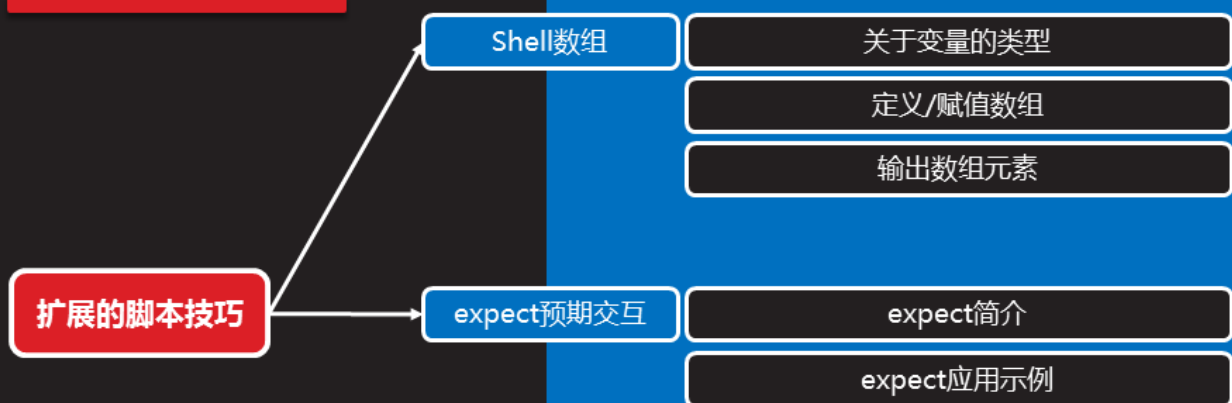
//输出提供的字符串

```
[root@svr5 ~]# echo $NM
```

//前面已清空，所以无结果



扩展的脚本技巧



Shell数组

关于变量的类型

知识讲解

- Shell对变量类型的管理比较松散
 - 变量的值默认均视为文本
 - 用在数学运算中时，自动将其转换为整数

```
[root@svr5 ~]# var1=123
```

```
[root@svr5 ~]# var2=$var1+20
```

```
[root@svr5 ~]# echo $var2
```

```
123+20
```

//123作为文本字符串

```
[root@svr5 ~]# expr $var1 + 20
```

```
143
```

//123作为整数值



定义/赋值数组

知识讲解

- 方法一，整体赋值：
 - 格式：**数组名=(值1 值2 ... 值n)**
 - 示例：SVRS=(www ftp mail club)
- 方法二，为单个元素赋值：
 - 格式：**数组名[下标]=值**
 - 示例：FQDNS[0]=www.tarena.com
FQDNS[1]=mail.tarena.com
FQDNS[2]=club.tarena.com

下标从 0 开始



expect预期交互

expect简介

- 基于TCL编写的自动交互式程序
 - 可以用在Shell脚本中，为交互式过程自动输送预先准备的文本或指令，而无需人工干预
 - 触发的依据是预期会出现的特征提示文本

知识讲解

```
[root@svr5 ~]# yum -y install expect
```

```
...
```

```
Installed:
```

```
expect.x86_64 0:5.44.1.15-5.el6_4
```

```
Dependency Installed:
```

```
tcl.x86_64 1:8.5.7-6.el6
```



expect应用示例

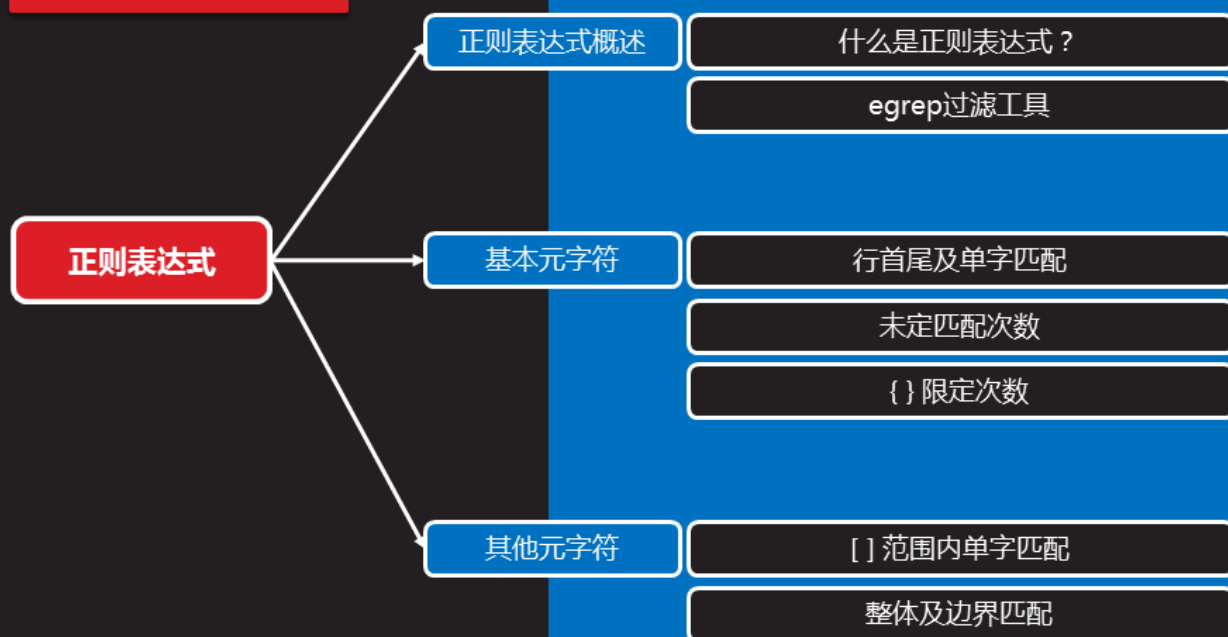
- 任务目标
 - 实现SSH自动登录，并远程执行指令

知识讲解

```
[root@svr5 ~]# vim ssh.sh
#!/bin/bash
host=192.168.4.5
expect << EOF
spawn ssh $host
expect "password"      {send "123456\r"}
expect "#"              {send "touch /a.txt\r"}
expect "#"              {send "exit\r"}
```



正则表达式

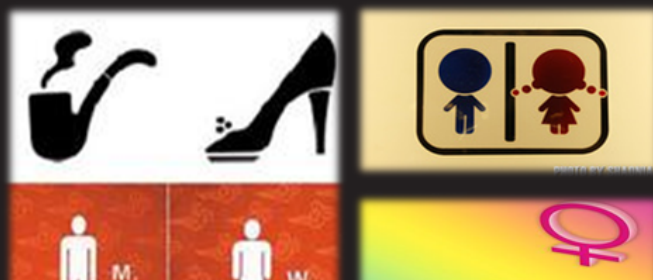


正则表达式概述

什么是正则表达式？

- Regular Express ?
 - 使用 “一串符号” 来描述有共同属性的数据

知识讲解



egrep过滤工具（续1）

知识讲解

- 常用命令选项
 - -i：忽略字母大小写
 - -v：条件取反
 - -c：统计匹配的行数
 - -q：静默、无任何输出，一般用于检测
 - -n：显示出匹配结果所在的行号
 - --color：标红显示匹配字符串

—— 看 \$? 返回值，
如果为0，说明有匹配，否则无匹配



未定匹配次数

知识讲解

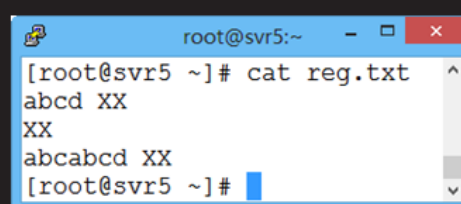
类型	含义	示例	说明
+	最少匹配一次	a+	一个或多个连续的 a
		(abc)+	一个或多个连续的 abc
?	最多匹配一次	a?	0个或1个 a
		(abc)?	0个或1个 abc
*	匹配任意次数	a*	0个或多个连续的 a
		(abc)*	0个或多个连续的 abc
		.*	任意长度的任意字符串



未定匹配次数（续1）

知识讲解

```
[root@svr5 ~]# egrep '(abc)+' reg.txt
abcd XX
abcabcd XX
[root@svr5 ~]# egrep '(abc)*' reg.txt
abcd XX
XX
abcabcd XX
```



```
root@svr5:~ - □ ×
[root@svr5 ~]# cat reg.txt
abcd XX
XX
abcabcd XX
[root@svr5 ~]#
```



其他元字符

[] 范围内单字匹配

- 匹配指定字符集合内的任何一个字符
 - []内加^可取反

知识讲解

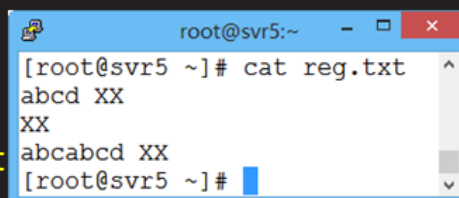
示 例	说 明
[alc45_?]	匹配 a、l、c、4、5、_、?
[a-z]	匹配任意小写字母
[A-Z]	匹配任意大写字母
[0-9]	匹配任意数字
[a-Z0-9]	匹配任意字母或数字
[^A-Z]	匹配包括非大写字母的行
^[^a-z]	匹配不以小写字母开头的行



[] 范围内单字匹配 (续1)

知识讲解

```
[root@svr5 ~]# egrep '^[A-Z]' reg.txt
abcd XX
abcabcd XX
[root@svr5 ~]# egrep '^[^a-z]' reg.txt
XX
[root@svr5 ~]# egrep 'bc[dfx]' reg.txt
abcd XX
abcabcd XX
```



```
root@svr5:~  
[root@svr5 ~]# cat reg.txt  
abcd XX  
XX  
abcabcd XX  
[root@svr5 ~]#
```



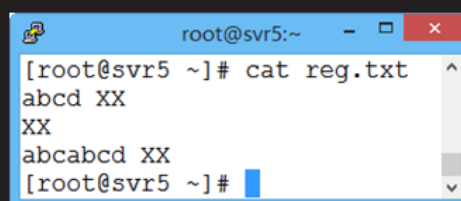
整体及边界匹配（续1）

知识讲解

```
[root@svr5 ~]# egrep '^root|^bin' /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin
```

```
[root@svr5 ~]# egrep '\<abcd\>' reg.txt  
abcd XX
```

```
[root@svr5 ~]# egrep 'abcd\>' reg.txt  
abcd XX  
abcabcd XX
```



```
root@svr5:~  
[root@svr5 ~]# cat reg.txt  
abcd XX  
XX  
abcabcd XX  
[root@svr5 ~]#
```

