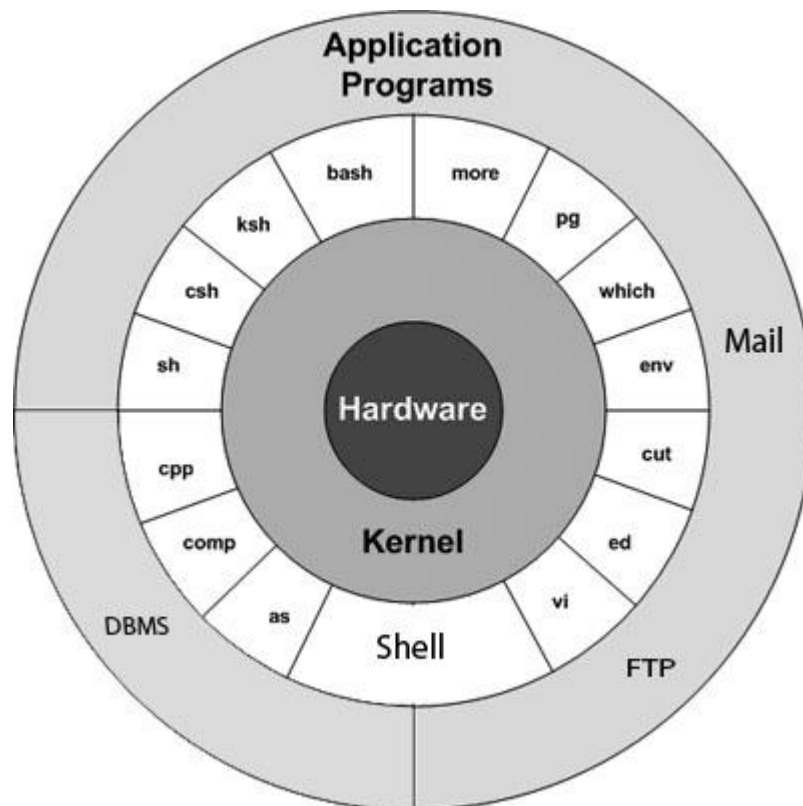


Shell脚本编程基础



内容

①

- Shell概述

- Shell的概念
- Shell的特点
- Shell版本
- Shell程序示例
- Shell程序执行方法

- Linux命令帮手

- 命令行编辑
- screen工具

②

- Shell特殊字符

- 通配符
- 正则表达式
- 引号
- 转义符与路径符
- 输入输出重定向
- 注释和后台命令
- 命令组合符
- 成组命令

③

- Shell变量

- 用户自定义变量
- 位置变量
- Shell预定义变量
- 环境变量

- 算术运算

- 控制结构

- 函数

④

- 作业控制

- Shell内置命令

- Shell脚本调试

- Shell脚本示例

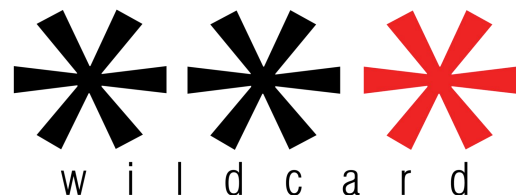
⑤

Shell特殊字符

- 通配符
- 正则表达式
- 引号
- 转义符与路径符
- 输入输出重定向
- 注释和后台命令
- 命令组合符
- 成组命令

通配符

（星号）	如file、*.fa	#任意个任意字符
?（问号）	如file?.txt	#1个任意字符
[字符集合]	如[13579] [agh]	#集合中的任意一个字符
[字符范围]	如[0-9] [a-z]	#范围中的任意一个字符
!或^	如file[!3-9]	#除了某范围或集合中的任意一个字符



通配符实例

\$ ls

ant antilope bat bear camel cat cow crab dog duck fish fox frog hog pig wolf

\$ ls *g

dog frog hog pig

ant antilope bat bear camel cat cow crab dog duck fish fox frog hog pig wolf

\$ ls ?og

dog hog

ant antilope bat bear camel cat cow crab dog duck fish fox frog hog pig wolf

\$ ls [a-c]*

ant antilope bat bear camel cat cow crab

ant antilope bat bear camel cat cow crab dog duck fish fox frog hog pig wolf

\$ ls [!a-c]*

dog duck fish fox frog hog pig wolf

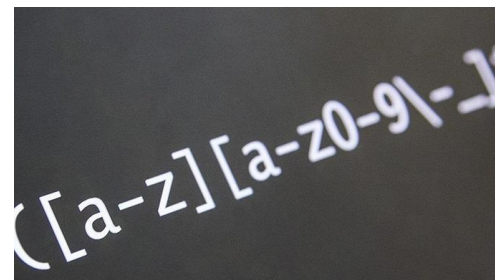
\$ ls [dfhp]*g

dog frog hog pig

ant antilope bat bear camel cat cow crab dog duck fish fox frog hog pig wolf

正则表达式

- **正则表达式**（Regular Expression）是一种文本模式，包括**普通字符**和特殊字符（称为“元字符”）。正则表达式使用单个字符串来描述、匹配一系列符合某个规则的字符串。
- grep、sed、vi、less等命令都支持正则表达式，另外Perl、Python、PHP、JavaScript、R、Ruby、AWK等脚本语言及Java、C#、Delphi等编译语言也都支持正则表达式。
- 正则表达式包括**基本正则表达式**和**扩展的正则表达式**。



基本正则表达式

元字符	含义
*	匹配之前的项0次或多次
.	匹配除了换行符外的任意一个字符
^	匹配行首，如^M 即 匹配以M开头的行
\$	匹配行尾，如G\$ 即 匹配以G结尾的行
[]	匹配中括号中指定的任意一个字符，如[aeiou]匹配任意一个元音字母，[a-z][0-9]匹配一个小写字母和一位数字构成的两个字符
[^]	匹配除中括号内的字符以外的任意一个字符
\	转义符，用于取消特殊符号的含义

基本正则表达式示例

```
$ cat nucleic_acid
```

DNA

RNA

mRNA

tRNA

rRNA

snRNA

snoRNA

miRNA

lncRNA

```
$ grep '[mt]RNA' nucleic_acid
```

mRNA

tRNA

```
$ grep '^mtr]RNA' nucleic_acid
```

snRNA

snoRNA

miRNA

lncRNA

扩展的正则表达式

元字符	含义
?	匹配之前的项0次或1次
+	匹配之前的项1次或多次
()	匹配表达式，创建一个用于匹配的字串
{n}	匹配其前面的字符恰好出现n次，如[0-9]\{2\} 匹2位数字
{n,}	匹配其前面的字符出现不小于n次
{n,m}	匹配其前面的字符至少出现n次，最多出现m次，如[a-z]\{6,8\} 匹配6到8个小写字母
	匹配 两边的任意一项

扩展的正则表达式示例

```
$ cat nucleic_acid
```

DNA

RNA

mRNA

tRNA

rRNA

snRNA

snoRNA

miRNA

lncRNA

```
$ grep -E '^.?RNA' nucleic_acid
```

RNA

mRNA

tRNA

rRNA

```
$ grep -E "[a-z]{3}RNA" nucleic_acid
```

snoRNA

lncRNA

通配符与正则表达式的区别

- **通配符**是用来**匹配文件名的**，是**完全匹配**；
- **正则表达式**是用来**匹配文件中的文本的**，是**包含匹配**，如**grep**查找一个正则表达式时，任意包含该表达式的行都会显示出来。

引号

- 双引号:

- 由双引号括起来的字符，除\$、`和\外都作为普通字符处理

- 单引号:

- 由单引号括起来的字符除\外都作为普通字符处理

- 反引号:

- 反引号括起来的字符shell作为命令处理

双引号

双引号中的内容：

- **\$**表示变量引用（关于Shell变量，我们在后面还要学习），就是用变量值替换\$及后面的变量名；
- **反引号（`）**表示命令替换，反引号及其中的内容用反引号中的命令的执行结果替换；
- **反斜杠（\）**表示转义，其后如果是\$、"、`、\时，它们会被当做普通字符对待，另外\t表示制表符，\n表示换行。

双引号示例

```
$ echo "Hello $LOGNAME"
```

Hello peter

```
$ echo "Hello \${LOGNAME}"
```

Hello \$LOGNAME

```
$ echo "Current time is `date`"
```

Current time is Thu Sep 3 21:10:25 CST 2020

```
$ echo "Ross is \"good at marriage\"."
```

Ross is "good at marriage".

```
$ echo "Current time is \`${date}`"
```

Current time is `date`

单引号

- 单引号的使用要简单一些，其中的字符除反斜杠（转义符，\）外都被Shell作为字符本身对待
- 与双引号相比，\$符号在单引号中将作为其自身输出而不是变量开始的标志；反引号（`）是其自身而不作为命令替换的标志；但反斜杠（\）与在双引号中相同，也解释为转义符

单引号示例

```
$ echo 'Hello $LOGNAME'
```

\$原样输出

```
Hello $LOGNAME
```

```
$ echo 'Hello `echo Peter`'
```

反引号原样输出

```
Hello `echo Peter`
```

```
$ echo -e 'Hello\tPeter'
```

转义符，\t输出制表符

```
Hello    Peter
```

```
$ echo -e 'Hello\nPeter'
```

转义符，\n输出换行符

```
Hello
```

```
Peter
```

```
$ echo -e 'Hello\\tPeter'
```

转义符，\\输出\

```
Hello\tPeter
```

单引号与双引号中的特殊字符比较

符号	双引号中	单引号中
\$	变量引用	\$本身
`	命令替换	`本身
\	转义	转义

反引号

- 反引号（```）表示命令替换，Shell在执行命令行之前，先用反引号中的命令的执行结果替换反引号及其中的内容。`$( )`与反引号的作用相同。
- 例如：

```
$ echo "The current time is `date`"
```

```
The current time is Thu Nov 28 15:42:47 CST 2024
```

```
$ echo "The current time is $(date)"
```

```
The current time is Thu Nov 28 15:42:47 CST 2024
```

转义符与路径符

•转义符\:

- 放在特殊符号之前，该特殊符号表示符号本身
- 表示换行、制表符等：\n，\t
- 用在命令行的末端，换行后继续输入（相当于转义了换行符）

•路径符/:

- 路径中跟在目录名的后面，在路径的开始则表示根目录
- 在算术运算中还表示除法运算

输入输出重定向

- 标准输出重定向 (**>**) : `ls >file`
- 标准输出添加重定向 (**>>**) : `ls >>file`
- 标准错误输出重定向 (**2>**) : `ls genome 2>file`
- 标准错误输出添加重定向 (**2>>**) : `ls genome 2>>file`
- 标准输入重定向 (**<**) : `tr b B <file`
- 即时输入重定向 (**<<**) : `cat >file <<eof`

注释和后台命令

- 注释（#）：

This is an annotation line

#!/bin/bash #特殊的注释行，脚本解释器路径

- 后台命令（&）：

sort file &

命令组合符

- 无条件命令组合符

- 管道:

cat fern | grep sinensis #命令1的输出作为命令2的输入

- 顺序执行:

cd plant; touch rose #依次执行命令1和命令2

- 条件命令组合符

- 逻辑与:

[-f cow] && rm -f cow #如果命令1执行成功则执行命令2

- 逻辑或:

test -d animal || mkdir animal #如果命令1执行不成功再执行命令2

成组命令

{}: 成组命令（不新建进程，在当前Shell中运行命令），如：

```
{ echo "My working path is:";pwd;}
```

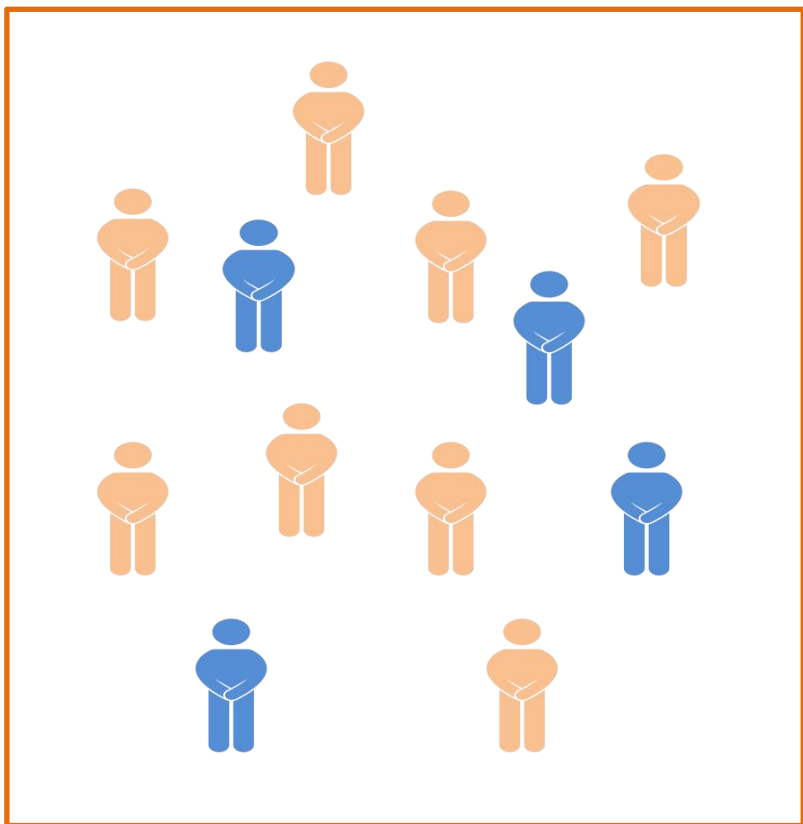
空格

分号

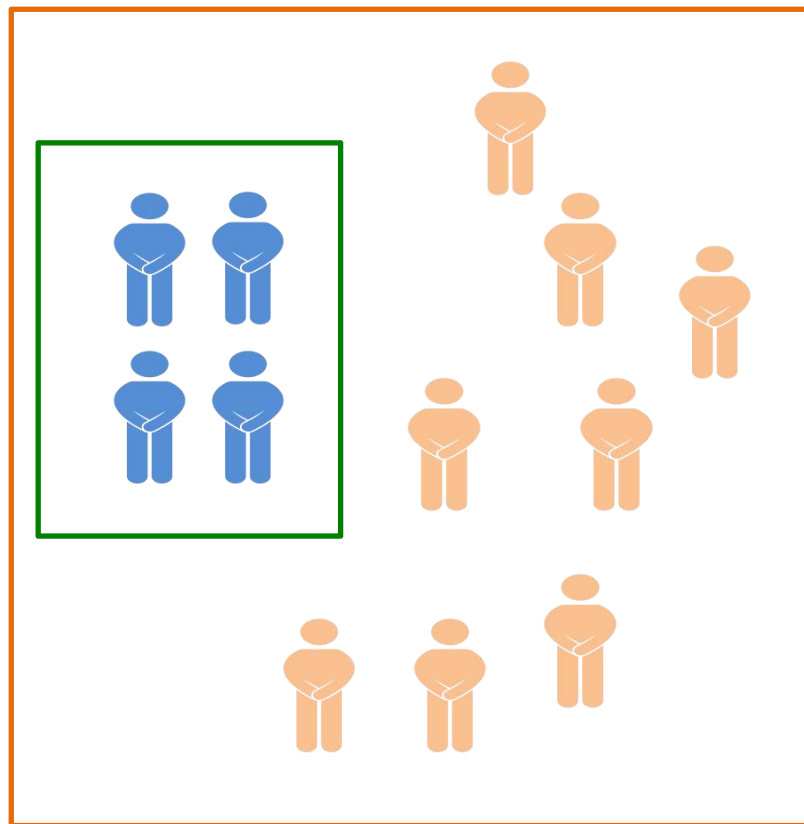
—另外，{}还可用来组合字符串，如：

```
touch {file1,2}_{1,2,3}
```

(): 成组命令（新建子进程，在子Shell中运行命令），如：
(echo "My current path is:";pwd)



{ }



()

{ }组合命令就像是在大房间里把几个人编成一组（左图），但他们只是逻辑上的一组，在物理空间上与其他人还是在一个房间中。

()组合命令就像在大房间中建一个小房间（右图），把几个人编成一组并放到这个小房间，他们做的事不会影响大房间中的其他人。

课堂作业8

课后作业7

在本目录下编辑shell脚本文件ex_07.sh并运行，运行该脚本能实现如下功能：

- （1）当前目录（运行ex_07.sh时的当前目录）下新建目录ex_dir；
- （2）进入目录ex_dir；
- （3）在目录ex_dir中新建文件ex_07.txt；
- （4）往文件ex_07.txt中写入内容"This is exercise 7"（不包括引号）；
- （5）复制文件/home/pub/seq/at_LEC1_CDS.fa到目录ex_dir中。

未完待续

LOADING