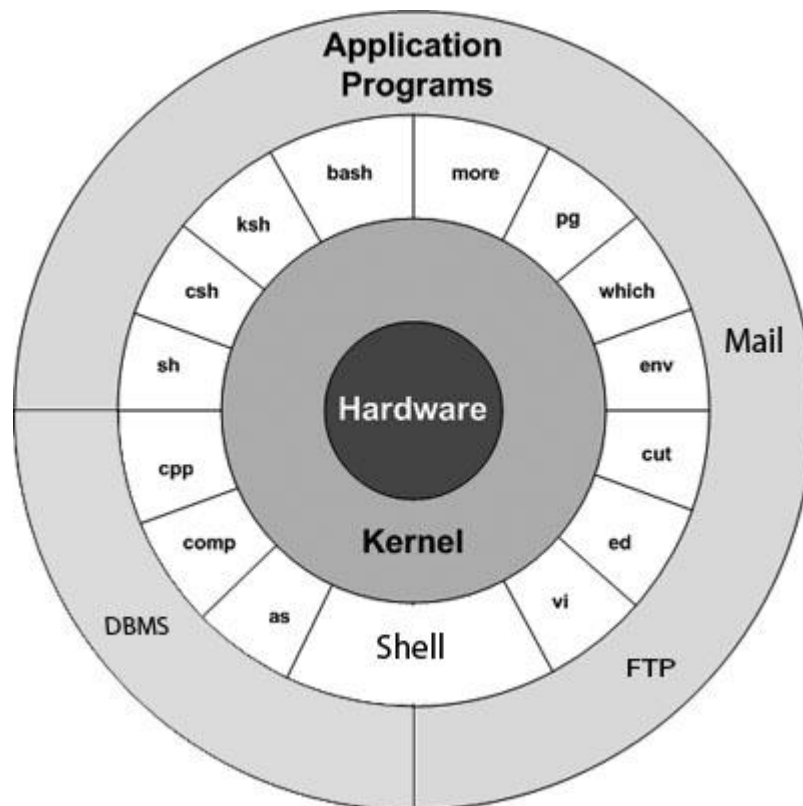


# Shell脚本编程基础



# 内容

## ①

- Linux命令帮手
  - 命令行编辑
  - screen工具
- Shell特殊字符（1）
  - 通配符
  - 正则表达式

## ②

- Shell特殊字符（2）
  - 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简介
  - Shell的概念
  - Shell的特点
  - Shell版本
  - Shell程序示例
  - Shell程序执行方法
- Shell变量
  - 用户自定义变量
  - 位置变量
  - Shell预定义变量
  - 环境变量

## ④

- 算术运算
- 控制结构
- 函数
- 作业控制
- Shell内置命令
- Shell脚本调试
- Shell脚本实例

# 内容

## ①

- Linux命令帮手
  - 命令行编辑
  - screen工具
- Shell特殊字符（1）
  - 通配符
  - 正则表达式

## ②

- Shell特殊字符（2）
  - 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简介
  - Shell的概念
  - Shell的特点
  - Shell版本
  - Shell程序示例
  - Shell程序执行方法
- Shell变量
  - 用户自定义变量
  - 位置变量
  - Shell预定义变量
  - 环境变量

## ④

- 算术运算
- 控制结构
- 函数
- 作业控制
- Shell内置命令
- Shell脚本调试
- Shell脚本实例

## 第7章 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

命令连接符

成组命令

# 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'
```

```
Hello $LOGNAME
```

# \$原样输出

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

```
Hello `echo Peter`
```

# 反引号原样输出

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

```
Hello    Peter
```

# 转义符, \t输出制表符

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

```
Hello
```

```
Peter
```

# 转义符, \n输出换行符

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

```
Hello\tPeter
```

# 转义符, \\输出\

# 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

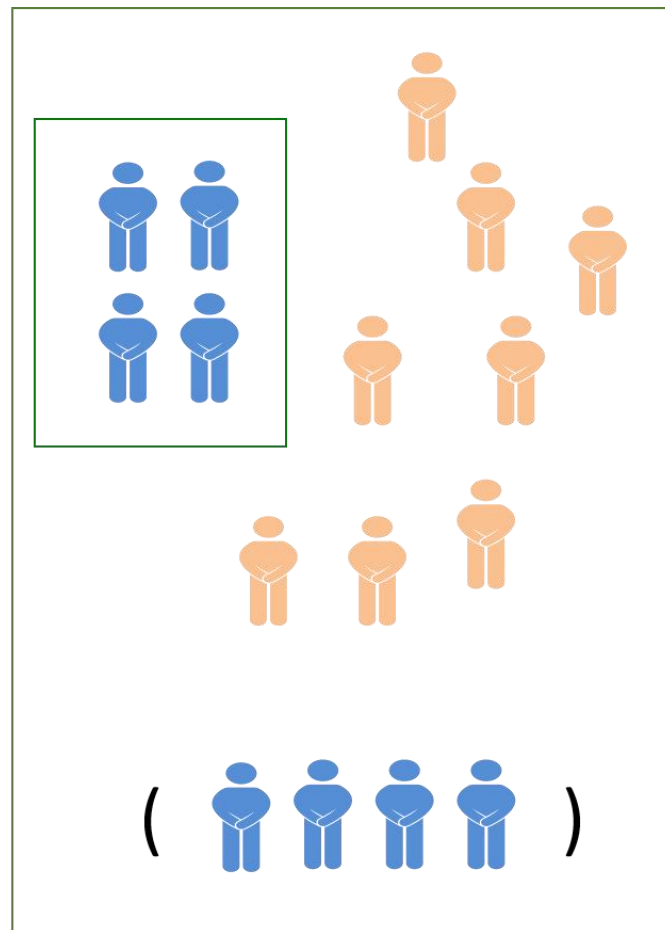
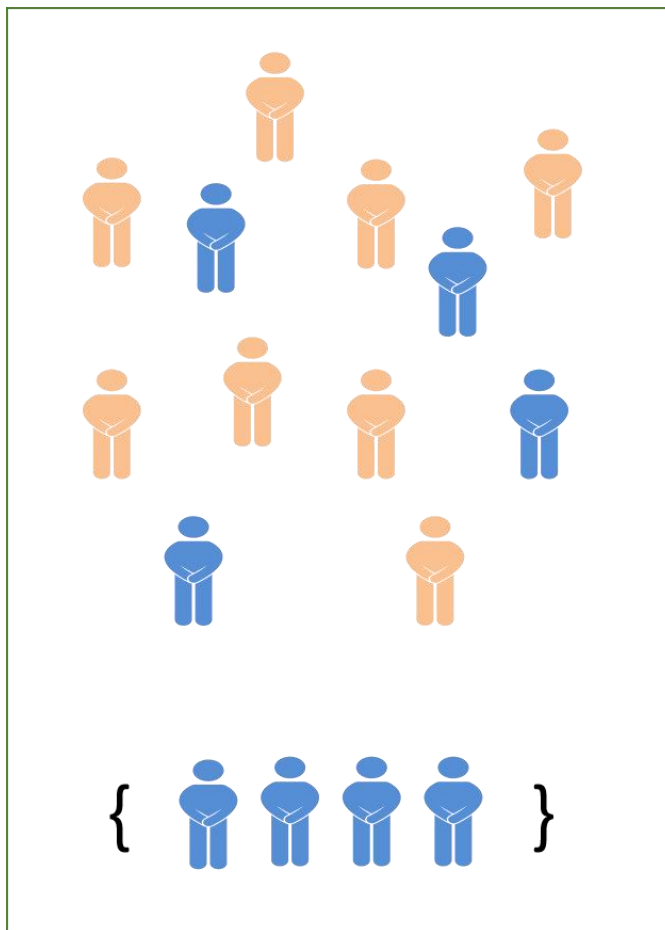
分号

—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。

