

SHELL DAY03



# Shell脚本编程

NSD SHELL

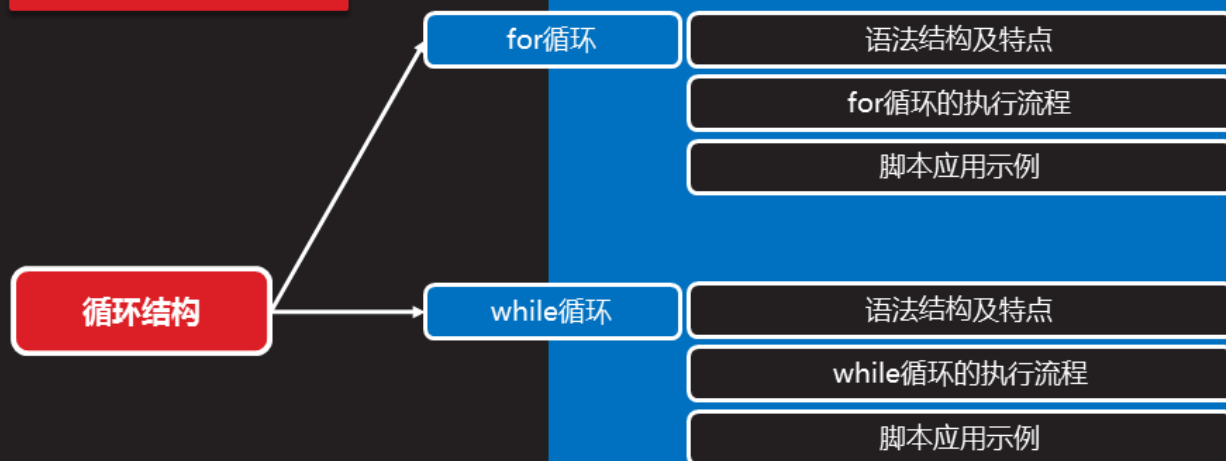
DAY03

# 内容

上午	09:00 ~ 09:30	作业讲解与回顾
	09:30 ~ 10:20	循环结构
	10:30 ~ 11:20	
	11:30 ~ 12:00	case语句
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	函数及中断控制
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



## 循环结构



# for循环

## 语法结构及特点

- 遍历/列表式循环
  - 根据变量的不同取值，重复执行命令序列

知识讲解

```
for 变量名 in 值列表
do
    命令序列
done
```

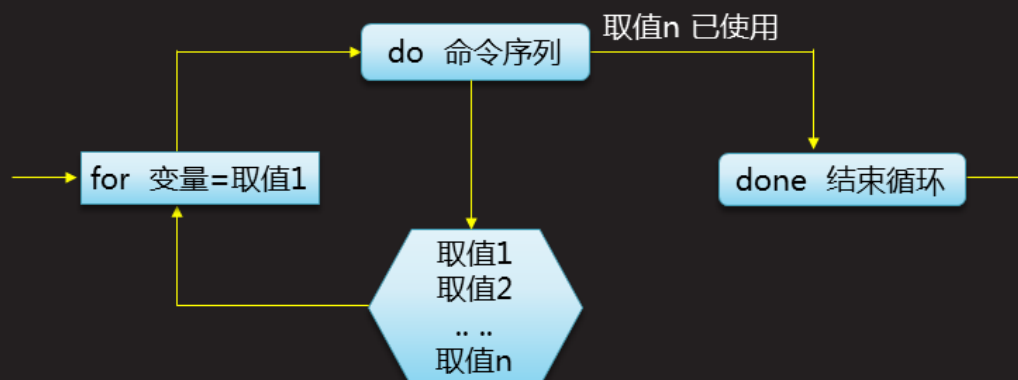


```
for 收件人 in 邮件列表
do
    发送邮件
done
```



# for循环的执行流程

- 流程示意图



## 脚本应用示例

- 任务目标
  - 批量添加用户账号（名称无规律）

知识讲解

```
[root@svr5 ~]# cat uaddfor.sh
#!/bin/bash
ULIST=$(cat /root/users.txt)
for UNAME in $ULIST
```

```
do
    useradd $UNAME
    echo "123456" | passwd --stdin $UNAME
done
```

```
root@svr5:~
[root@svr5 ~]# cat /root/users.txt
zhangsan
lisi
[root@svr5 ~]# ./uaddfor.sh
Changing password for user zhangsan.
passwd: all authentication tokens up'd
Changing password for user lisi.
passwd: all authentication tokens up'd
[root@svr5 ~]#
```



## 脚本应用示例（续1）

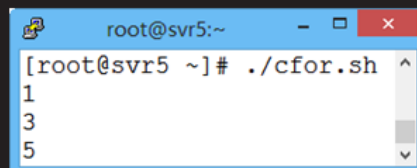
知识讲解

- C语言风格的for循环
  - 通过变量控制，条件成立时循环
  - 步长可控次数

```
for ((初值; 条件; 步长控制))  
do  
    命令序列  
done
```



```
[root@svr5 ~]# cat cfor.sh  
#!/bin/bash  
for ((i=1;i<=5;i+=2))  
do  
    echo $i  
done
```



```
root@svr5:~  
[root@svr5 ~]# ./cfor.sh  
1  
3  
5
```

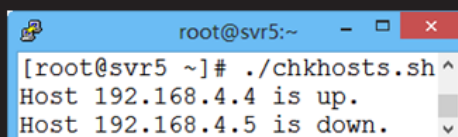


## 案例1：使用for循环结构

批量检测多个主机的存活状态：

- 1) 对192.168.4.0/24网段执行ping检测
- 2) 脚本能遍历ping各主机，并反馈存活状态

课堂练习



```
root@svr5:~  
[root@svr5 ~]# ./chkhosts.sh  
Host 192.168.4.4 is up.  
Host 192.168.4.5 is down.
```



# while循环

## 语法结构及特点

- 条件式循环
  - 反复测试条件，只要成立就执行命令序列

知识讲解

```
while 条件测试  
do  
    命令序列  
done
```



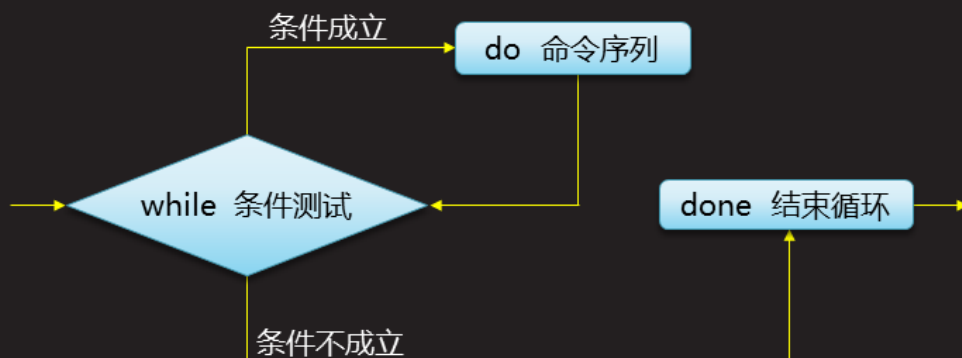
```
while 未猜中正确价格  
do  
    反复猜商品价格  
done
```



# while循环的执行流程

- 流程示意图

知识讲解



## 脚本应用示例

- 任务目标
  - 批量添加用户（名称有规律）

知识讲解

```
[root@svr5 ~]# cat uaddwhile.sh
```

```
#!/bin/bash
```

```
PREFIX="tuser" ; i=1
```

```
while [ $i -le 5 ]
```

```
do
```

```
    useradd ${PREFIX}$i
```

```
    echo "123456" | passwd --stdin ${PREFIX}$i &> /dev/null
```

```
    let i++
```

```
done
```

//递增控制，避免死循环

```
root@svr5:~  
[root@svr5 ~]# ./uaddwhile.sh  
[root@svr5 ~]# tail -2 /etc/passwd  
tuser4:x:521:521::/home/tuser4:/bin/bash  
tuser5:x:522:522::/home/tuser5:/bin/bash
```

The screenshot shows a terminal window titled "root@svr5:~". It displays the execution of the script ./uaddwhile.sh, followed by a tail command showing the last two lines of /etc/passwd, which now include the newly created users tuser4 and tuser5.



## 案例2：使用while循环结构

编写2个Shell脚本，分别实现以下目标：

- 1) 提示用户猜测一个随机数，直到才对为止
- 2) 检测192.168.4.0/24网段，列出不在线的主机地址

课堂练习



### case语句

case语句

case分支结构

语法结构及特点

case分支的执行流程

脚本应用示例



# case分支结构

## 语法结构及特点

- 检查变量的实际取值
  - 如果与预设的值相匹配，则执行对应的操作

知识讲解

```
case 变量值 in
  模式1)
    命令序列1 ;;
  模式2)
    命令序列2 ;;
  ...
  *)
    默认命令序列
esac
```

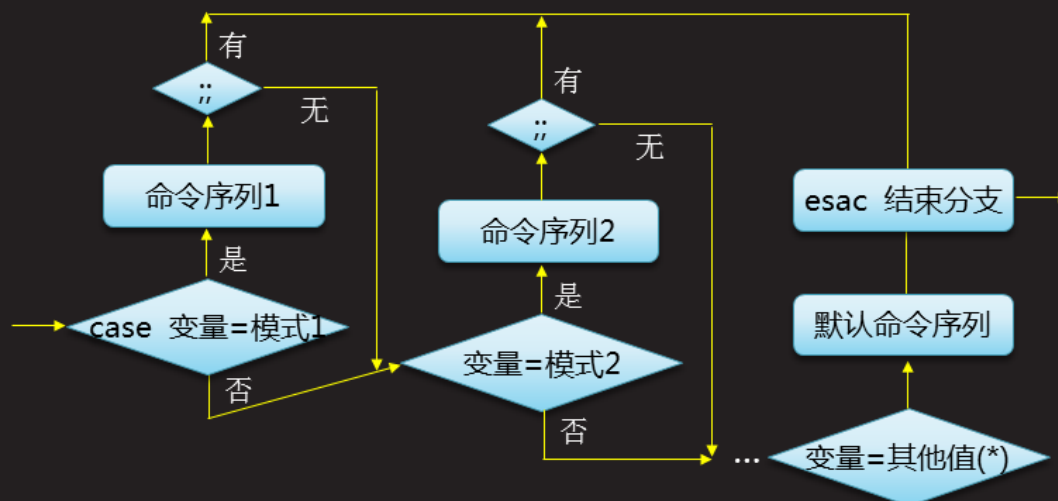


```
case 控制参数 in
  start)
    启动XX服务 ;;
  stop)
    停止XX服务 ;;
  ...
  *)
    显示服务脚本用法
esac
```



# case分支的执行流程

## • 流程示意图



# 脚本应用示例

## • 应用示例

### – 判断用户输入

```

[root@svr5 ~]# cat key.sh
#!/bin/bash
case $1 in
  redhat)
    echo "fedora";;
  fedora)
    echo "redhat";;
  *)
    echo "用法: $0 {redhat|fedora}"
esac
  
```

//默认输出脚本用法



## 案例3：基于case分支编写脚本

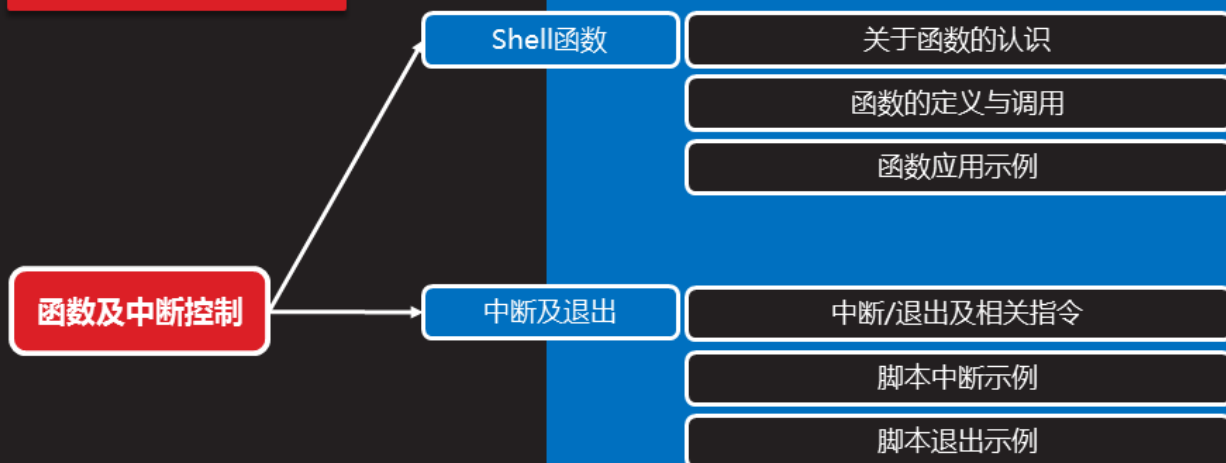
编写test.sh脚本，要求如下：

- 1) 能使用redhat、fedora控制参数
- 2) 控制参数通过位置变量\$1传入
- 4) 当用户输入redhat参数，脚本返回fedora
- 5) 当用户输入fedora参数，脚本返回redhat
- 6) 当用户输入其他参数，则提示错误信息

课堂练习



### 函数及中断控制



# Shell函数

## 关于函数的认识

知识讲解

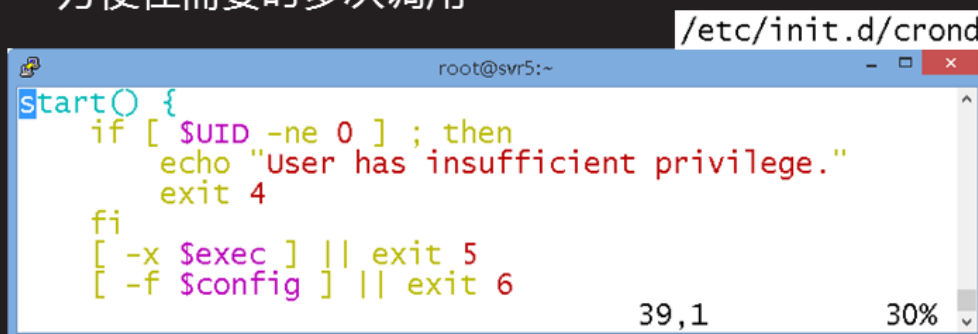
- 什么是函数？
  - 在Shell环境中，将一些需要重复使用的操作，定义为公共的语句块，即可称为函数
- 使用函数的好处？
  - 使脚本代码更简洁，增强易读性
  - 提高Shell脚本的执行效率



## 关于函数的认识（续1）

- 服务脚本中的函数应用
  - 适用于比较复杂的启动/终止控制操作
  - 方便在需要时多次调用

知识讲解



```
root@svr5:~  
start() {  
    if [ $UID -ne 0 ] ; then  
        echo "User has insufficient privilege."  
        exit 4  
    fi  
    [ -x $exec ] || exit 5  
    [ -f $config ] || exit 6  
}
```

39,1 30%



## 函数的定义与调用

- 如何定义一个函数

知识讲解

```
function 函数名 {  
    命令序列  
    ...  
}
```

或者

```
函数名() {  
    命令序列  
    ...  
}
```



## 函数的定义与调用（续1）

知识讲解

- 调用已定义的函数
  - 格式：**函数名**
  - 先定义了才能调用，就好比脚本的“内部命令”
- 函数传值
  - 格式：**函数名 值1 值2 ...**
  - 传递的值作为函数的“位置参数”



## 函数应用示例

- 任务目标
  - 创建一个对2个整数求和的加法器

知识讲解

```
[root@svr5 ~]# function adder {  
> echo ${$1+$2}  
> }
```

```
[root@svr5 ~]# type adder  
adder is a function  
...
```

A terminal window titled "root@svr5:~" showing the execution of the adder function. The first command is "adder 12 34" which outputs "46". The second command is "adder 123 456" which outputs "579". The prompt is now "[root@svr5 ~]#".

```
root@svr5:~  
[root@svr5 ~]# adder 12 34  
46  
[root@svr5 ~]# adder 123 456  
579  
[root@svr5 ~]#
```

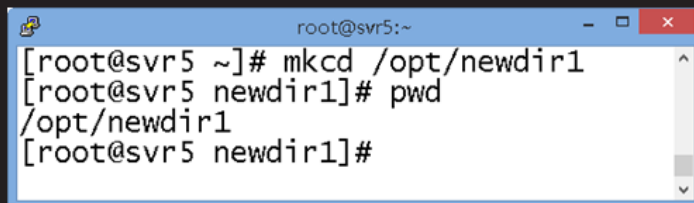


## 函数应用示例（续1）

- 任务目标
  - 新建函数mkcd，用来创建一个目录，并切换到此目录

知识讲解

```
[root@svr5 ~]# mkcd() {  
> mkdir $1  
> cd $1  
> }
```



```
root@svr5:~  
[root@svr5 ~]# mkcd /opt/newdir1  
[root@svr5 newdir1]# pwd  
/opt/newdir1  
[root@svr5 newdir1]#
```



## 函数应用示例（续2）

- Shell版fork炸弹
  - 仅13个字符：`.() { .& };.`
  - 递归死循环，可迅速耗尽系统资源

知识讲解

### 代码解析

<code>.()</code>	//定义一个名为.的函数
<code>{</code>	//函数块开始
<code>.&amp;</code>	//在后台递归调用函数.
<code>}</code>	//函数块结束
<code>;</code>	//与下一条执行语句的分隔
<code>.</code>	//再次调用函数



## 案例4：使用Shell函数

### 1. 编写一个计算器脚本mycolor.sh

- 1) 将颜色输出的功能定义为函数
- 2) 调用函数，可以自定义输出内容和颜色

课堂  
练习



## 中断及退出



# 中断/退出及相关指令

- 中断、继续、退出

知识讲解

类 型	含 义
break	跳出当前所在的循环体，执行循环体后的语句块
continue	跳过循环体内余下的语句，重新判断条件以决定是否需 要执行下一次循环
exit	退出脚本，默认的回值是 0



## 脚本中断示例

- 任务目标
  - 从键盘循环取整数（0结束）并求和，输出最终结果

知识讲解

```
[root@svr5 ~]# cat brkwhile.sh
#!/bin/bash
while read -p "请输入待累加的整数（0表示结束）：" x
do
    [ $x -eq 0 ] && break
    SUM=$((SUM+x))
done
echo "总和是：$SUM"
```

```
root@svr5:~
[root@svr5 ~]# ./brkwhile.sh
请输入待累加的整数（0表示结束）: 12
请输入待累加的整数（0表示结束）: 34
请输入待累加的整数（0表示结束）: 0
总和是: 46
[root@svr5 ~]#
```



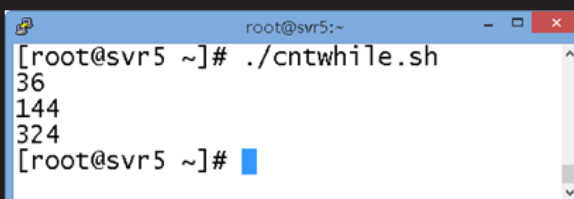
## 脚本中断示例（续1）

- 任务目标

- 跳过1~20以内非6的倍数，输出其他数的平方值

知识讲解

```
[root@svr5 ~]# cat cntwhile.sh
#!/bin/bash
i=0
while [ $i -le 20 ]
do
    let i++
    [ ${i%6} -ne 0 ] && continue
    echo ${i*i}
done
```



```
root@svr5:~
[root@svr5 ~]# ./cntwhile.sh
36
144
324
[root@svr5 ~]#
```



## 脚本退出示例

- 任务目标

- 利用位置参数获取2个整数，计算出这两个整数的和
- 如果参数不够2个，则提示正确用法并退出脚本

知识讲解

```
[root@svr5 ~]# cat exit.sh
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "用法:$0 num1 num2"
    exit 10
fi
expr $1 + $2
```

//退出脚本，返回值设为10



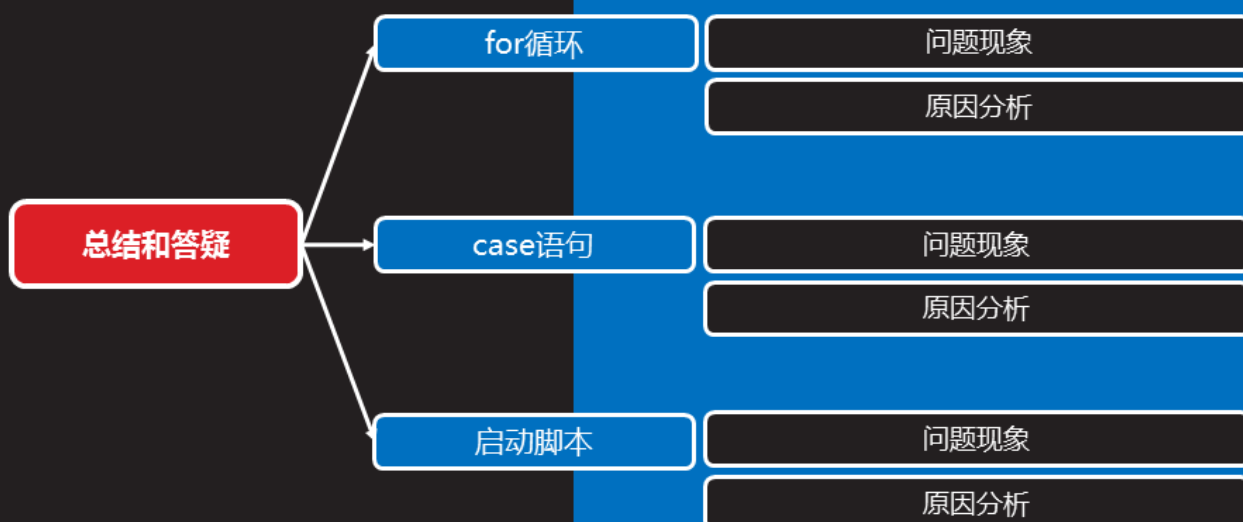
## 案例5：中断与退出

课堂练习

- 从键盘循环取整数（0结束）并求和，输出最终结果
- 找出1~20以内6的倍数，并输出她的平方值



### 总结和答疑



# for循环



## 问题现象

- 故障错误信息

```
[root@svr5 ~]# for(i=1;i<=5;i++)  
-bash: syntax error near unexpected token `('
```

```
[root@svr5 ~]# for i in 1 2  
> echo $i  
-bash: syntax error near unexpected token
```



# 原因分析

知识讲解

- 分析故障
  - 报错信息：-bash: syntax error near unexpected token `('
- 分析故障原因
  - 基本语法错误
  - for循环的执行体，需要嵌入到do和done中间



## case语句

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# cat test.sh
#!/bin/bash
case $1 in
start)
    echo start
stop)
    echo stop
esac
[root@svr5 ~]# bash test.sh
test.sh: line 5: syntax error near unexpected token `)'
test.sh: line 5: `stop)'
```

知识讲解



## 原因分析

- 分析故障
  - 报错信息：-bash: syntax error near unexpected token `)'
- 分析故障原因
  - 基本语法错误
  - 使用case语句时，命令序列需要使用;;作为结束符

知识讲解



# 启动脚本

## 问题现象

- 故障错误信息

```
[root@svr5 init.d]# cat test
#!/bin/bash
#chkconfig: - 120 110
case $1 in
start)
    echo start;;
stop)
    echo stop;;
esac
[root@svr5 init.d]# chkconfig --add test
service a does not support chkconfig
```



# 原因分析

知识讲解

- 分析故障
  - 报错信息：service a does not support chkconfig
- 分析故障原因
  - chkconfig启动和关闭的数字不能大于99

