

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	case语句
	10:30 ~ 11:20	
	11:30 ~ 12:00	函数及中断控制
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	字符串处理
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



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

编写部署软件脚本

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
```



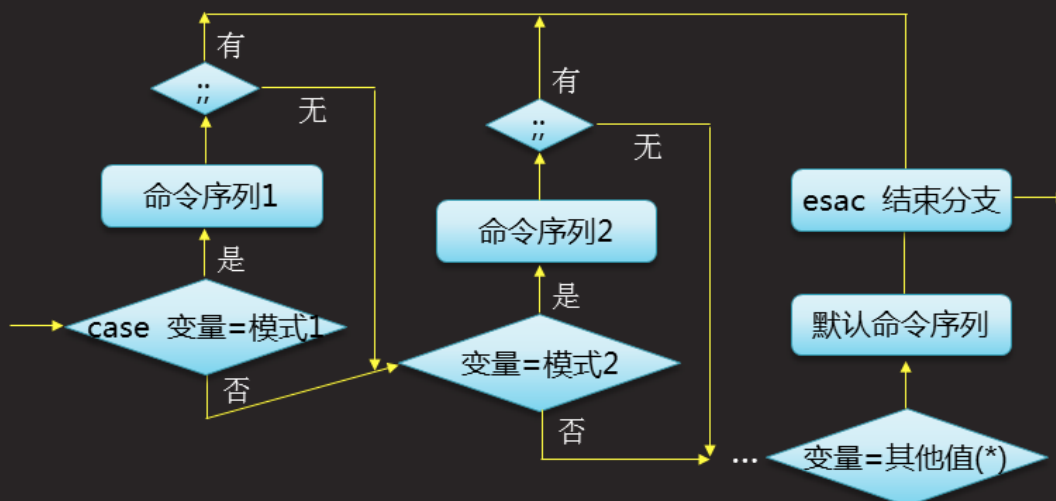
```
case 控制参数 in
start)
    启动XX服务 ;;
stop)
    停止XX服务 ;;
...
*)
    显示服务脚本用法
esac
```



case分支的执行流程

- 流程图示意图

知识讲解



脚本应用示例

- 应用示例
 - 判断用户输入

知识讲解

```
[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1：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



编写部署软件脚本

- 一键安装Nginx
 - 一键源码安装Nginx软件
 - 脚本自动安装相关软件的依赖包
 - 脚本自动判断yum是否可用

知识讲解



案例2：编写一键部署软件脚本

课堂练习

- 一键安装Nginx
 - 一键源码安装Nginx软件
 - 脚本自动安装相关软件的依赖包



编写部署软件脚本（续1）

知识讲解

- 编写Nginx启动脚本
 - 脚本支持start、stop、restart、status
 - 脚本支持报错提示
 - 脚本具有判断是否已经开启或关闭的功能



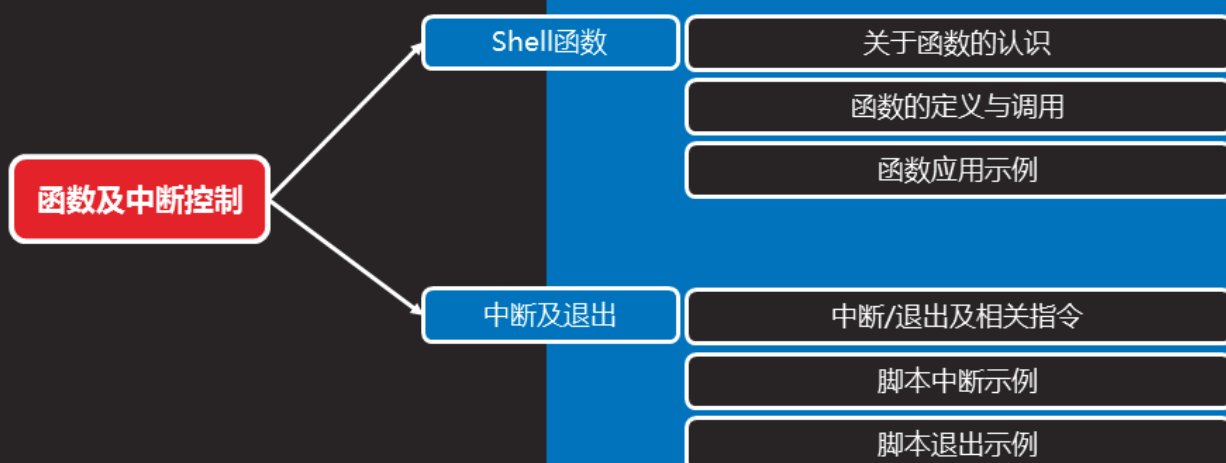
案例3：编写启动脚本

课堂练习

- 编写Nginx启动脚本
 - 脚本支持start、stop、reload、restart、status
 - 脚本支持报错提示
 - 脚本具有判断是否已经开启或关闭的功能



函数及中断控制



Shell函数

关于函数的认识

- 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  
}
```



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

知识讲解

- 任务目标
 - 创建一个对2个整数求和的加法器

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

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

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

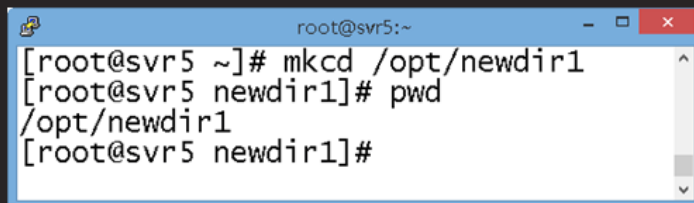


函数应用示例（续1）

- 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>.&</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

课堂
练习

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



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 ~]#
```

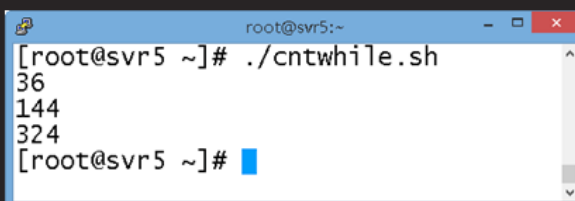


脚本中断示例（续1）

- 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



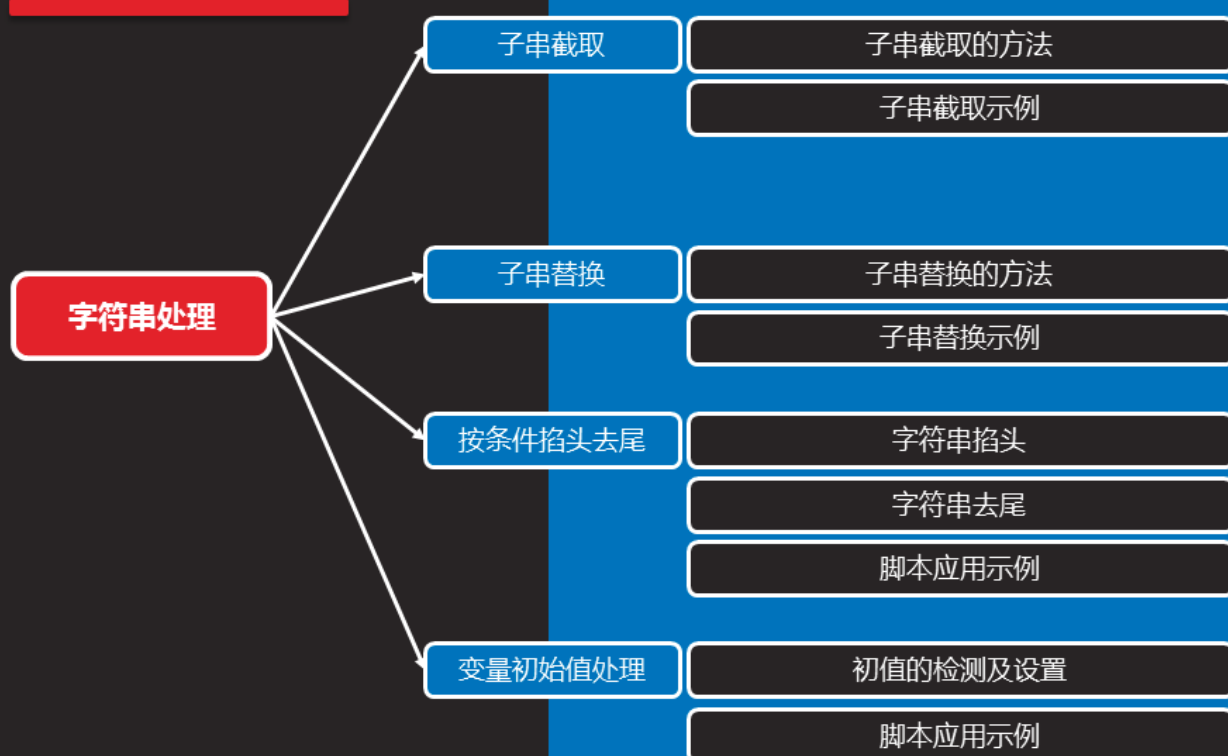
案例5：中断与退出

- 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

课堂练习



字符串处理



子串截取

子串截取的方法

- 使用 \${}表达式
 - 格式：\${var:起始位置:长度}
 - 编号从0开始，可省略

子串截取示例

- 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
```

知识讲解



## 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}`



## 子串替换示例

知识讲解

- 任务目标
  - 将变量NM中的a替换为##

```
[root@svr5 ~]# echo ${NM/a/##}
T##rena IT Group.
```

//只替换掉第一个 a

```
[root@svr5 ~]# echo ${NM//a/##}
T##ren## IT Group.
```

//替换掉所有的 a



# 按条件掐头去尾

## 字符串掐头

知识讲解

- 从左向右，最短匹配删除
    - 格式：`${变量名}*关键词}`
  - 从左向右，最长匹配删除
    - 格式：`${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#*/}
var/spool/mail/root //删除到最近匹配
[root@svr5 ~]# echo ${MDIR##*/}
root //删除到最远匹配
```



## 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

知识讲解

- 任务目标
  - 实现批量改名，将扩展名 .doc 改为 .txt

```
[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6：字符串截取及切割

课堂练习

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



## 变量初始值处理

---

## 初值的检测及设置

知识讲解

- 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

```
[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



## 脚本应用示例

知识讲解

- 任务目标
  - 提示输入一个正整数x, 求从1~x的和
  - 若用户未输入值, 则赋初值 `x=1`, 避免执行出错

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

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

```
read -p "请输入一个正整数: " x
```

```
x=${x:-1}; i=1; SUM=0
```

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

```
do
```

```
 let SUM+=i ; let i++
```

```
done
```

```
echo "从1到$x的总和是: $SUM"
```

```
root@svr5:~
[root@svr5 ~]# ./sumx.sh
请输入一个正整数:
从1到1的总和是: 1
[root@svr5 ~]# ./sumx.sh
请输入一个正整数: 50
从1到50的总和是: 1275
[root@svr5 ~]# ./sumx.sh
请输入一个正整数: 100
从1到100的总和是: 5050
[root@svr5 ~]#
```



## 案例7：字符串初值的处理

编写一个脚本sumx.sh，求从1-x的和：

- 1) 从键盘读入x值
- 2) 当用户未输入任何值时，默认按1计算

课堂练习



### 总结和答疑



# 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)'
```



# 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
  - 使用case语句时，命令序列需要使用;;作为结束符



# 启动脚本

---

## 问题现象

知识讲解

- 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



