

SHELL DAY02



# Shell脚本编程

NSD SHELL

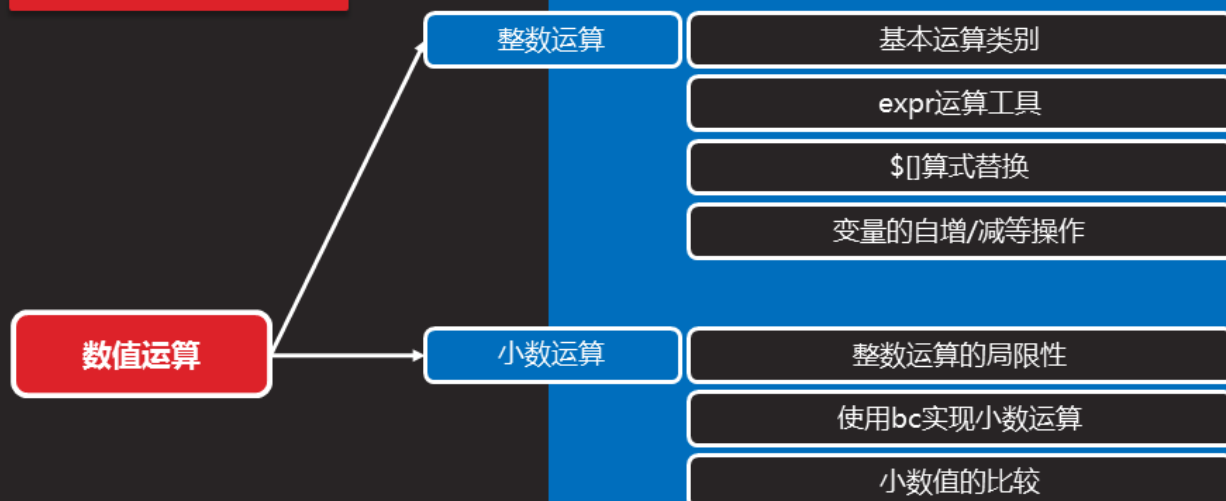
DAY02

# 内容

上午	09:00 ~ 09:30	作业讲解与回顾
	09:30 ~ 10:20	数值运算
	10:30 ~ 11:20	
	11:30 ~ 12:00	条件测试
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	if选择结构
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



## 数值运算



# 整数运算

## 基本运算类别

- 四则运算
  - 加法：  $\text{num1} + \text{num2}$
  - 减法：  $\text{num1} - \text{num2}$
  - 乘法：  $\text{num1} * \text{num2}$
  - 整除：  $\text{num1} / \text{num2}$
- 取余数运算
  - 求模：  $\text{num1} \% \text{num2}$



## expr运算工具

知识讲解

- 计算指定的表达式，并输出结果
  - 格式：expr 整数1 运算符 整数2 ...
  - 乘法操作应采用 \\* 转义，避免被作为Shell通配符

类 型	运算符	示例
加法	+	expr 43 + 21、expr \$X + \$Y
减法	-	expr 43 - 21、expr \$X - \$Y
乘法	\*	expr 43 \* 21、expr \$X \* \$Y
除法	/	expr 43 / 21、expr \$X / \$Y
取余数	%	expr 43 % 21、expr \$X % \$Y



## \$[]算式替换

知识讲解

- 使用 \$[ ] 或 \$(( )) 表达式
  - 格式：\$[整数1 运算符 整数2 ...]
  - 乘法操作\*无需转义，运算符两侧可以无空格
  - 引用变量可省略 \$ 符号
  - 计算结果替换表达式本身，可结合echo命令输出

```
[root@svr5 ~]# X=43
[root@svr5 ~]# echo $[X+21]
64
[root@svr5 ~]# echo $((X-21)), $((X*21))
22, 903
```



## 变量的自增/减等操作

- 使用 `$[ ]` 替换，或者 `let` 命令来完成
  - 结合 `echo` 命令查看结果

知识讲解

简写表达式	完整表达式
<code>i++</code>	<code>i=i+1</code>
<code>i--</code>	<code>i=i-1</code>
<code>i+=2</code>	<code>i=i+2</code>
<code>i-=2</code>	<code>i=i-2</code>
<code>i*=2</code>	<code>i=i*2</code>
<code>i/=2</code>	<code>i=i/2</code>
<code>i%=2</code>	<code>i=i%2</code>

```
[root@svr5 ~]# i=43
[root@svr5 ~]# echo ${i+=2}
45
[root@svr5 ~]# echo ${i-=8}
37
[root@svr5 ~]# let i++ ; echo $i
38
[root@svr5 ~]# let i-=7 ; echo $i
31
```



## 小数运算

## 整数运算的局限性

- Bash内建机制仅支持整数值运算
  - expr命令、\$[ ] 算式替换 不支持有小数的运算

知识讲解

```
[root@svr5 ~]# expr 123 + 45.678  
expr: 参数数目错误
```

```
[root@svr5 ~]# echo ${3.14*2}  
-bash: 3.14*2: syntax error: invalid arithmetic operator (error token is  
".14*2")
```



## 使用bc实现小数运算

- 多数Linux系统默认安装此工具
  - 支持高精度的数值运算
  - 直接运行bc可进入交互式运算界面，quit退出
  - 设置 scale=n 可约束小数位

知识讲解

```
[root@svr5 ~]# bc //打开bc计算器程序  
12.34*56.78 //提交表达式  
700.66  
scale=4 //将可用的小数位增加为4  
12.34*56.78 //重新计算表达式  
700.6652  
quit //退出计算器  
[root@svr5 ~]#
```



## 使用bc实现小数运算（续1）

知识讲解

- 结合管道向bc发送表达式
  - 多个表达式以分号分隔
  - 通过echo命令+管道传递要计算的表达式

```
[root@svr5 ~]# A=12.34
[root@svr5 ~]# echo "$A*56.789" | bc           //单表达式
700.776
[root@svr5 ~]# echo "scale=4; $A*56.789; 5/3" | bc //多表达式
700.7762
1.6666
```



## 小数值的比较

知识讲解

- 基本用法
  - echo "数值1 比较符 数值2" | bc
  - 如果表达式成立，则返回的计算结果为1，否则返回0
  - 常见比较操作：>、>=、<、<=、==、!=

```
[root@svr5 ~]# A=12.34 ; B=56.78
[root@svr5 ~]# echo "$A <= $B" | bc           //A是否小于或等于B
1
[root@svr5 ~]# echo "$A > $B" | bc            //A是否大于或等于B
0
```



# 案例1：Shell中的数值运算

课堂练习

## 1. 使用expr、\$[ ]、let等整数运算工具

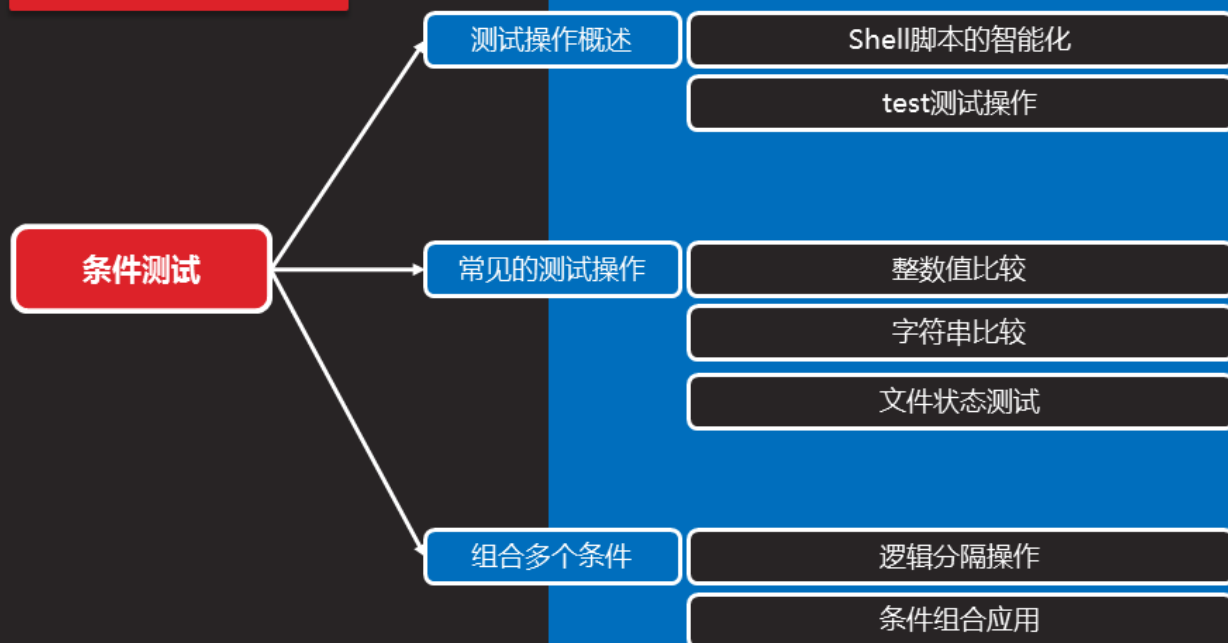
- 1) 定义变量X=1234
- 2) 计算X与78的四则运算及求模结果

## 2. 使用bc实现小数运算操作

- 1) 以交互方式计算12.34与5.678的四则运算结果
- 2) 非交互重复上述计算，最多4位小数



## 条件测试





# 测试操作概述

## Shell脚本的智能化

- 使Shell脚本获得识别能力？
- 为命令的执行提供最直接的识别依据
  - 文件或目录的读/写等状态
  - 数值的大小
  - 字符串是否匹配
  - 多条件组合



# test测试操作

- 语法格式
  - test 选项 参数
  - [ 选项 参数 ]

知识讲解

help test 查阅帮助

```
root@svr5:~# help test
test: test [expr]
Exits with a status of 0 (true) or 1 (false) depending on
the evaluation of EXPR.  Expressions may be unary or binary.  Unar
y
expressions are often used to examine the status of a file.  There
are string operators as well, and numeric comparison operators.

File operators:

-a FILE      True if file exists.
-b FILE      True if file is block special.
-c FILE      True if file is character special.
-d FILE      True if file is a directory.
-e FILE      True if file exists.
-f FILE      True if file exists and is a regular file.
-g FILE      True if file is set-group-id.
-h FILE      True if file is a symbolic link.
-L FILE      True if file is a symbolic link.
```



## 常见的测试操作

# 字符串比较

- [ 操作符 字符串 ]

操作符	含 义
-z	字符串的值为空
-n	字符串的值不为空 ( 相当于 ! -z )

```
[root@svr5 ~]# [ -z "" ] && echo YES || echo NO  
YES
```

```
[root@svr5 ~]# [ -z " " ] && echo YES || echo NO  
NO
```

知识讲解



## 字符串比较 ( 续1 )

- [ 字符串1 操作符 字符串2 ]

操作符	含 义
==	两个字符串相同
!=	两个字符串不相同

```
[root@svr5 ~]# [ $USER == "root" ] && echo "超级用户"  
超级用户
```

```
[root@svr5 ~]# [ $PWD != "/" ] && pwd  
/root
```

知识讲解



## 整数值比较

- [ 整数值1 操作符 整数值2 ]

操作符	含 义
-eq	等于 ( Equal )
-ne	不等于 ( Not Equal )
-ge	大于或等于 ( Greater or Equal )
-le	小于或等于 ( Lesser or Equal )
-gt	大于 ( Greater Than )
-lt	小于 ( Lesser Than )

知识讲解



## 整数值比较 ( 续1 )

- 检查已登录的用户数，是否不超过5个

```
[root@svr5 ~]# who | wc -l
```

```
2
```

```
[root@svr5 ~]# [ $(who | wc -l) -le 5 ] && echo "OK"
```

```
OK
```

知识讲解



# 文件状态测试

- [ 操作符 文件或目录 ]

知识讲解

操作符	含 义
-e	判断对象是否存在 ( Exist ) , 若存在则结果为真
-d	判断对象是否为目录 ( Directory ) , 是则为真
-f	判断对象是否为一般文件 ( File ) , 是则为真
-r	判断对象是否有可读 ( Read ) 权限, 是则为真
-w	判断对象是否有可写 ( Write ) 权限, 是则为真
-x	对象是否有可执行 ( eXcute ) 权限, 是则为真



## 文件状态测试 ( 续1 )

知识讲解

```
[root@localhost ~]# [ -d /etc/hosts ]  
[root@localhost ~]# echo $?  
1 //目录/etc/hosts不存在, 测试不成立
```

```
[root@localhost ~]# [ -d /etc/vsftpd ]  
[root@localhost ~]# echo $?  
0 //目录/etc/vsftpd存在, 测试成立
```

等效简化



```
[root@svr5 ~]# [ -d /etc/vsftpd ] && echo "YES"  
YES
```



# 组合多个条件

## 逻辑分隔操作

- 主要用法：
  - 命令1 操作符 命令2 ..
  - [ 条件1 ] 操作符 [ 条件2 ] ..

知识讲解

操作符	含 义
&&	给定条件必须都成立，整个测试结果才为真
	只要其中一个条件成立，则整个测试结果为真



## 条件组合应用

知识讲解

- 当前用户为root，且位于/root目录下

```
[root@svr5 ~]# [ $USER == "root" ] && [ $PWD == "/root" ]  
[root@svr5 ~]# [ $? -eq 0 ] && echo YES  
YES
```

- 当/opt/testdir目录不存在时，创建该目录

```
[root@svr5 ~]# [ -d "/opt/testdir" ] || mkdir -p /opt/testdir  
[root@svr5 ~]# ls -ld /opt/testdir/  
drwxr-xr-x 2 root root 4096 12-03 18:13 /opt/testdir/
```



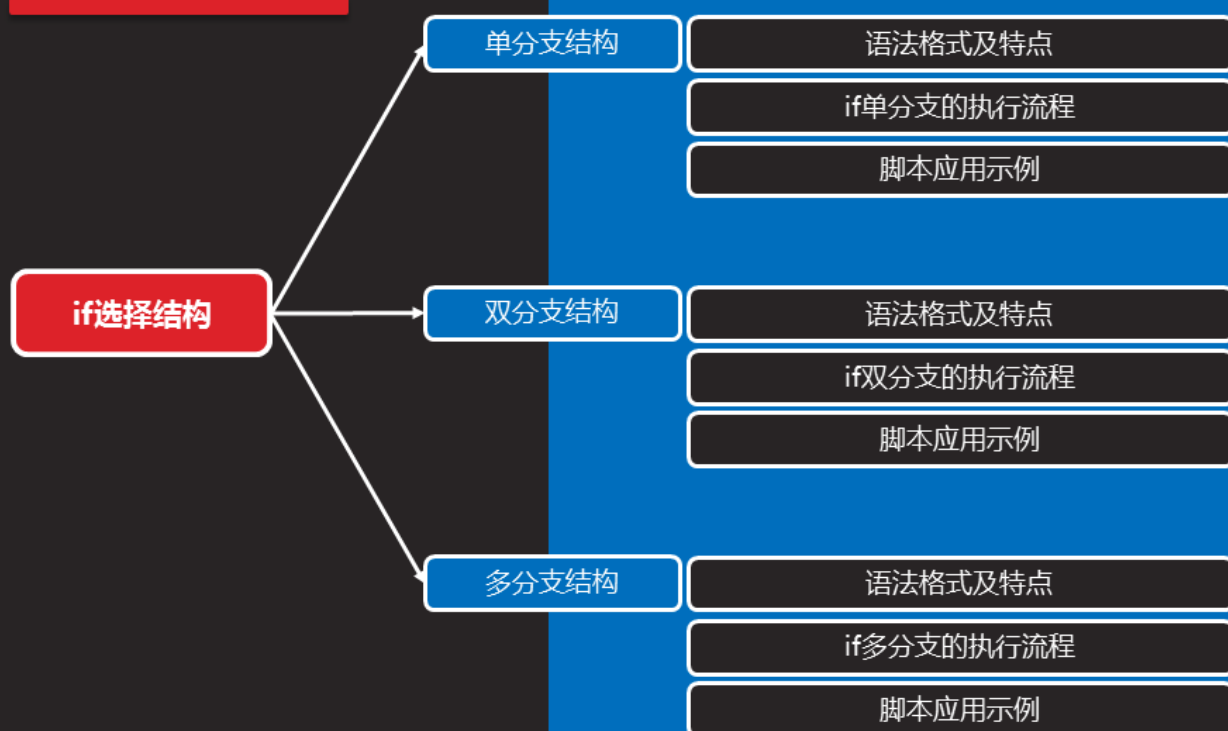
## 案例2：条件测试操作

课堂练习

1. 识别文件/目录的状态
2. 比较整数值的大小
3. 字符串匹配
4. 多个条件/操作的逻辑组合



## if选择结构



## 单分支结构



## 语法格式及特点

- 当“条件成立”时执行命令序列
- 否则，不执行任何操作

知识讲解

```
if 条件测试  
then 命令序列  
fi
```



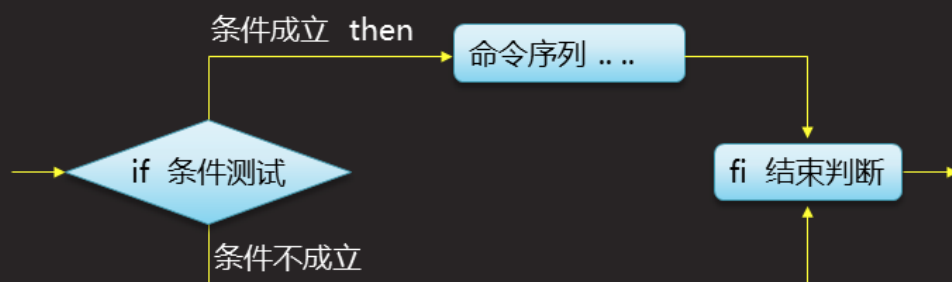
```
if 磁盘已用空间>80%  
then 报警  
fi
```



## if单分支的执行流程

- 流程示意图

知识讲解

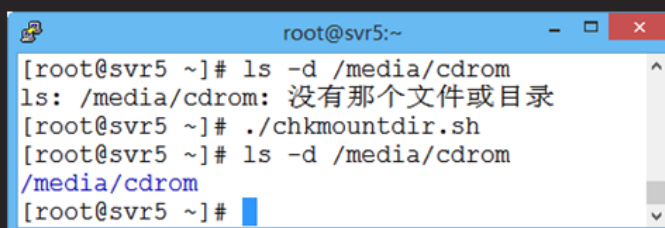


## 脚本应用示例

- 任务目标
  - 判断挂载点目录，若不存在则创建

知识讲解

```
[root@svr5 ~]# cat chkmountdir.sh
#!/bin/bash
MOUNT_DIR="/media/cdrom/"
if [ ! -d $MOUNT_DIR ]
then
    mkdir -p $MOUNT_DIR
fi
```



```
root@svr5:~
[root@svr5 ~]# ls -d /media/cdrom
ls: /media/cdrom: 没有那个文件或目录
[root@svr5 ~]# ./chkmountdir.sh
[root@svr5 ~]# ls -d /media/cdrom
/media/cdrom
[root@svr5 ~]#
```



## 双分支结构

## 语法格式及特点

- 当“条件成立”时执行命令序列1
- 否则，执行命令序列2

知识讲解

```
if 条件测试
then 命令序列1
else 命令序列2
fi
```



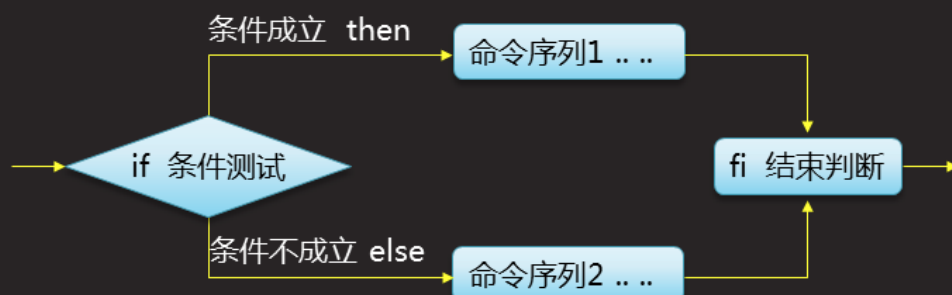
```
if 已监听TCP 80端口
then 提示“网站已运行”
else 启动httpd服务
fi
```



## if双分支的执行流程

- 流程示意图

知识讲解

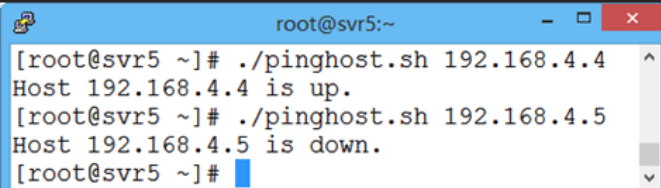


# 脚本应用示例

知识讲解

- 任务目标
  - 检测并判断指定的主机是否可ping通
  - 目标主机的地址以位置参数提供

```
[root@svr5 ~]# cat pinghost.sh
#!/bin/bash
ping -c 3 -i 0.2 -W 3 $1 &> /dev/null
if [ $? -eq 0 ]; then
    echo "Host $1 is up."
else
    echo "Host $1 is down."
fi
```



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



## 多分支结构

## 语法格式及特点

- 相当于if语句嵌套
- 针对多个条件分别执行不同的操作

知识讲解

```
if 条件测试1
then 命令序列1
elif 条件测试2
then 命令序列2
else 命令序列n
fi
```



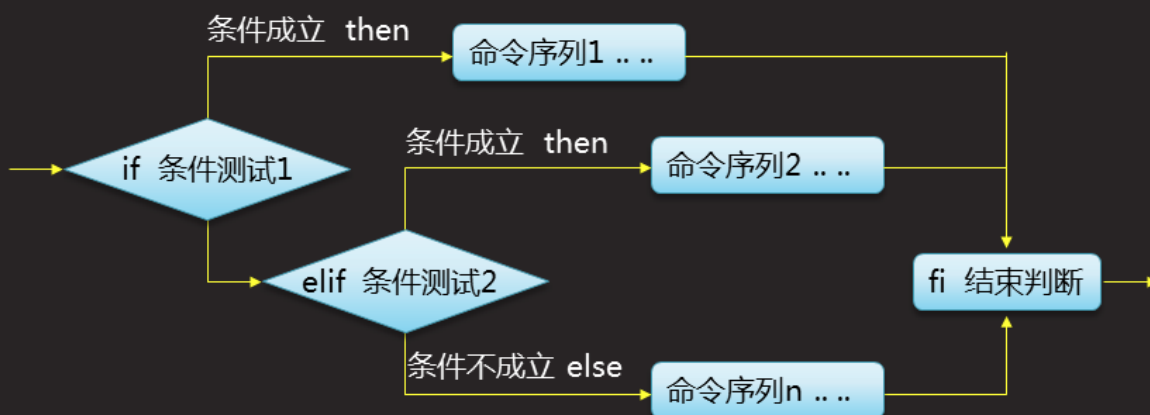
```
if 分数在85~100之间
then 判为“优秀”
elif 分数在70~84之间
then 判为“合格”
esle 判为“不合格”
fi
```



## if多分支的执行流程

- 流程示意图

知识讲解



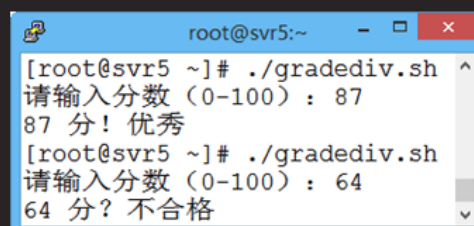
## 脚本应用示例

知识讲解

- 任务目标

- 输入一个分数，判断成绩分档
- 85~100 优秀、70~84 良好、低于70分 不及格

```
[root@svr5 ~]# cat gradediv.sh
#!/bin/bash
read -p "请输入分数（0-100）：" FS
if [ $FS -ge 85 ] && [ $FS -le 100 ]; then
    echo "$FS 分！优秀"
elif [ $FS -ge 70 ] && [ $FS -le 84 ]; then
    echo "$FS 分，合格"
else
    echo "$FS 分？不合格"
fi
```



```
root@svr5:~
[root@svr5 ~]# ./gradediv.sh
请输入分数（0-100）： 87
87 分！优秀
[root@svr5 ~]# ./gradediv.sh
请输入分数（0-100）： 64
64 分？不合格
```



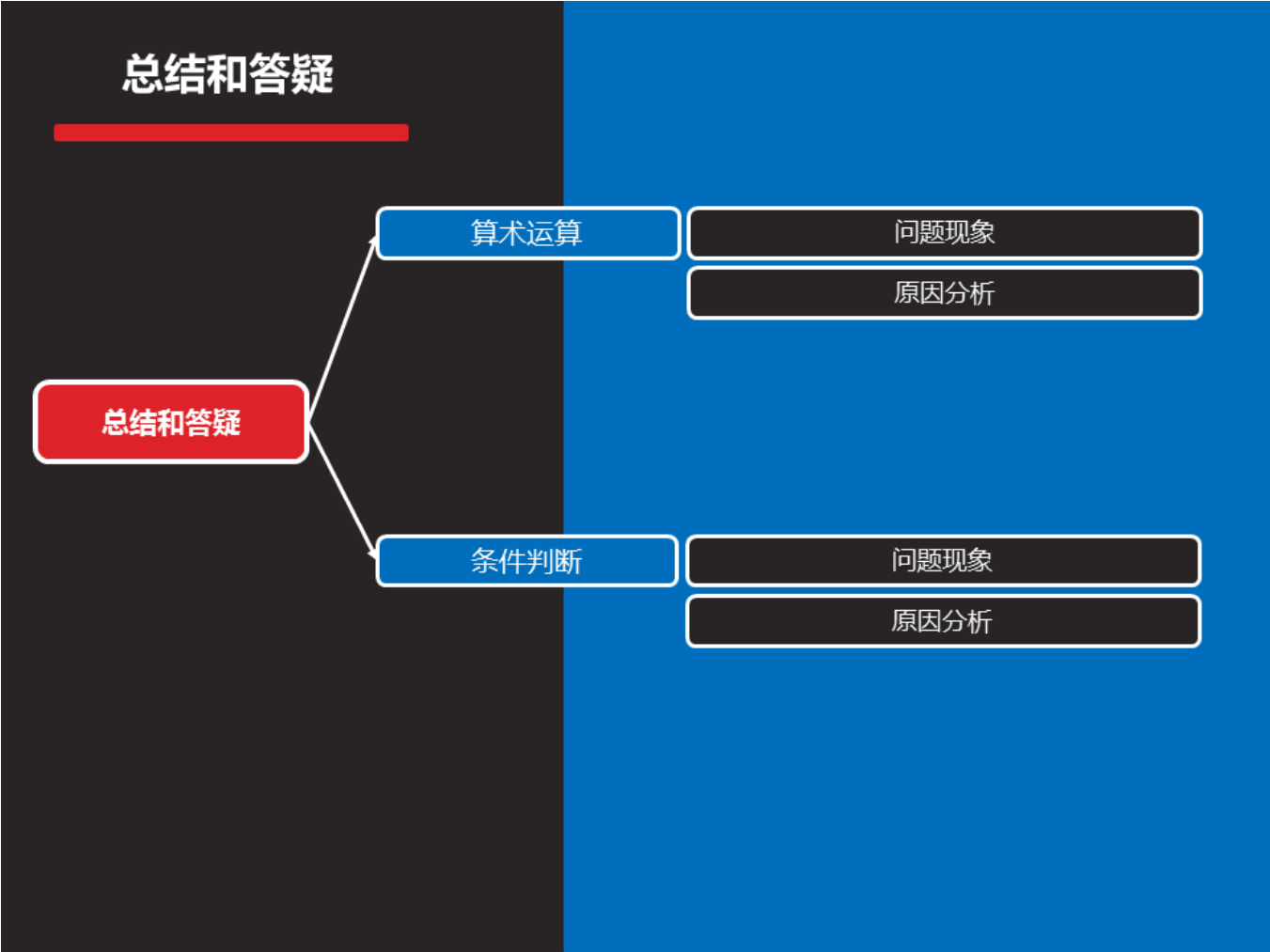
## 案例3：使用if选择结构


编写3个Shell脚本，分别实现以下目标：

- 1) 检测/media/cdrom目录，若不存在则创建
- 2) 检测并判断指定的主机是否可ping通
- 3) 从键盘读取一个论坛等级分，判断用户等级

课堂练习







达内教育

# 算术运算

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# # expr 1+2
1+2
[root@svr5 ~]# expr 10 * 5
expr: syntax error
[root@svr5 ~]# echo ${3.14*2}
3.14*2:syntax error: invalid arithmetic operator (error token is ".14*2")
```

知识讲解



## 原因分析

- 分析故障
  - 报错信息：无正确运算结果
  - 报错信息：syntax error
  - 报错信息：: invalid arithmetic operator
- 分析故障原因
  - 使用expr运算时，运算符两边需要有空格
  - expr进行乘法运算时，需要使用\屏蔽\*
  - expr无法进行小数运算

知识讲解





# 条件判断

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# [1 -gt 2]
-bash: [1: command not found
[root@svr5 ~]# [ 1-gt2 ]
[root@svr5 ~]# echo $?
0
[root@svr5 ~]# cat test.sh
#!/bin/bash
if [ a = a ];then
    echo a
[root@svr5 ~]# bash test.sh
syntax error: unexpected end of file
```



## 问题现象

- 故障错误信息

```
[root@svr5 ~]# cat test.sh
#!/bin/bash
if [ $a -eq 2 ];then
echo a
fi
[root@svr5 ~]# bash test.sh
a.sh: line 2: [: -eq: unary operator expected
```

知识讲解



## 原因分析

- 分析故障
  - 报错信息：**command not found**
  - 报错信息：判断结果有误
  - 报错信息：没有合理的结束符
  - 报错信息[: -eq: unary operator expected
- 分析故障原因
  - [的右边，]的左边，都需要空格
  - 判断符合两边需要有空格
  - if语句开始，需要配合使用fi作为判断结尾符号
  - 无法对空值进行判断

知识讲解



