Tedu.cn
达内教育

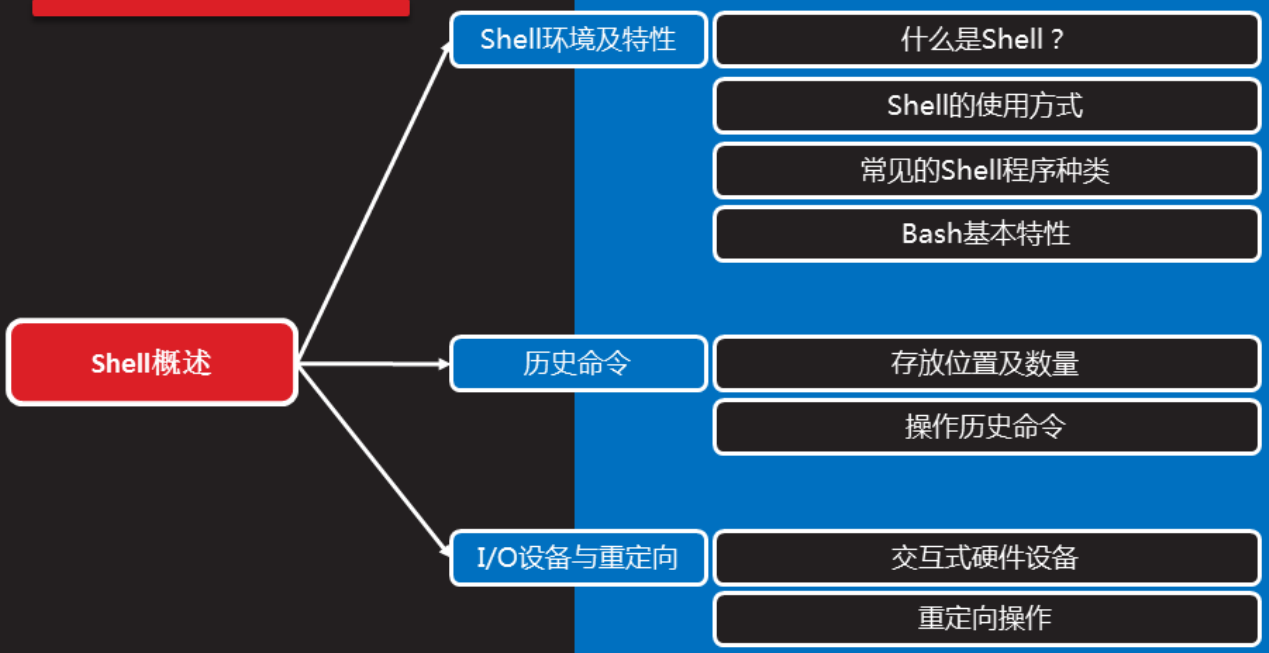
Shell脚本编程

NSD SHELL DAY01

内容

上午	09:00 ~ 09:30	Shell概述
	09:30 ~ 10:20	
	10:30 ~ 11:20	编写及执行脚本
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	Shell变量
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑

Shell概述

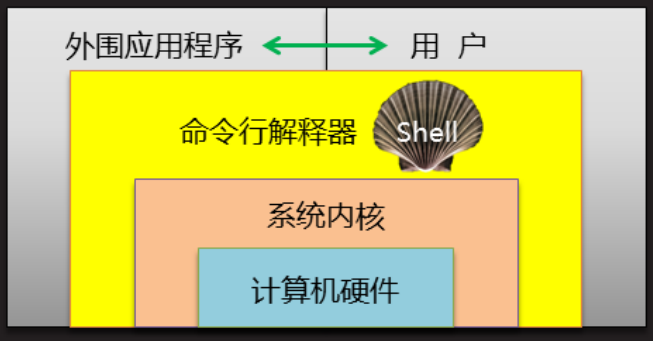


Shell环境及特性

什么是Shell？

知识讲解

- 在Linux内核与用户之间的解释器程序
  - 通常指 /bin/bash
  - 负责向内核翻译及传达用户/程序指令
  - 相当于操作系统的“外壳”



Shell的使用方式

知识讲解

- 交互式 —— 命令行
  - 人工干预、智能化程度高
  - 逐条解释执行、效率低
- 非交互式 —— 脚本
  - 需要提前设计、智能化难度大
  - 批量执行、效率高
  - 方便在后台静悄悄地运行

常见的Shell程序种类

知识讲解

- 如何切换Shell环境
  - 通过usermod、chsh更改登录Shell
  - 手动执行目标Shell程序

```
[root@svr5 ~]# cat  /etc/shells
/bin/sh                    //多数Unix默认的Shell
/bin/bash                  //多数Linux默认使用的Shell
/sbin/nologin              //非登录Shell
/bin/tcsh
/bin/csh
/bin/ksh
```

在RHEL系统中，实际上sh是bash的符号链接

Bash基本特性

知识讲解

- 命令行环境回顾
  - 快捷键、Tab键补齐
  - 命令历史
  - 命令别名
  - 标准输入输出
  - 重定向
  - 管道操作

历史命令

存放位置及数量

知识讲解

- 默认记录1000条
  - 保存位置：~/.bash_history
  - 控制历史命令的数量：/etc/profile

```
[root@svr5 ~]# grep  ^HISTSIZE  /etc/profile
HISTSIZE=1000                          //全局设置的记录个数
```

操作历史命令

知识讲解

- history工具
  - histroy：查看历史命令列表
  - history -c：清空历史命令
- 调用历史命令
  - !78：执行历史记录中的第78条命令
  - !str：执行最近一次以str开头的历史命令

Tedu.cn 达内教育

I/O设备与重定向

Tedu.cn 达内教育

交互式硬件设备

知识讲解

- 标准输入：从该设备接收用户输入的数据
- 标准输出：通过该设备向用户输出数据
- 标准错误：通过该设备报告执行中的出错信息

类 型	设备文件	文件描述号	默认设备
标准输入	/dev/stdin	0	键盘
标准输出	/dev/stdout	1	显示器
标准错误输出	/dev/stderr	2	显示器

重定向操作

Tedu.cn 达内教育

- 改变标准输入/输出/错误输出的方向

知识讲解

类 型	操作符	用 途
重定向输入	<	将文本输入来源由键盘改为指定的文件
重定向输出	>	将命令行的正常执行输出保存到文件，而不是直接显示在屏幕上
	>>	与>类似，但操作是追加而不是覆盖
重定向错误	2>	将命令行的执行出错信息保存到文件，而不是直接显示在屏幕上
	2>>	与2>类似，但操作是追加而不是覆盖
混合重定向	&>	相当于>和2>，覆盖到同一个文件

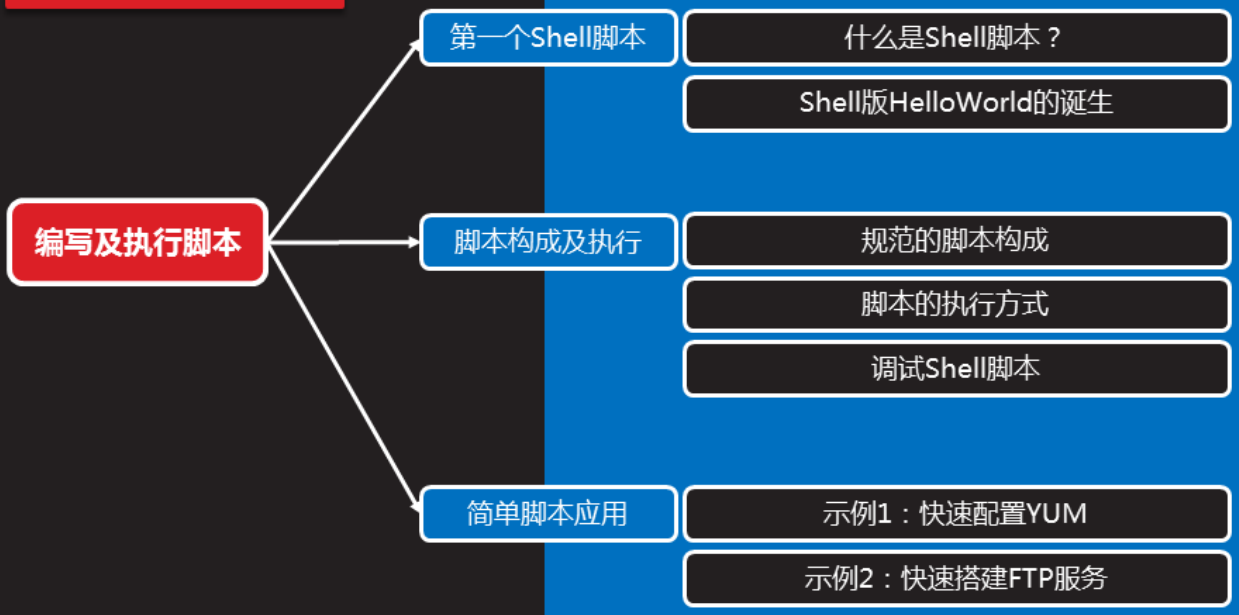
案例1：Shell基础应用

Tedu.cn 达内教育

课堂练习

1. 切换用户的Shell环境
2. 练习命令历史、命令别名
3. 重定向标准输入/输出/错误输出
4. 管道操作实践

编写及执行脚本



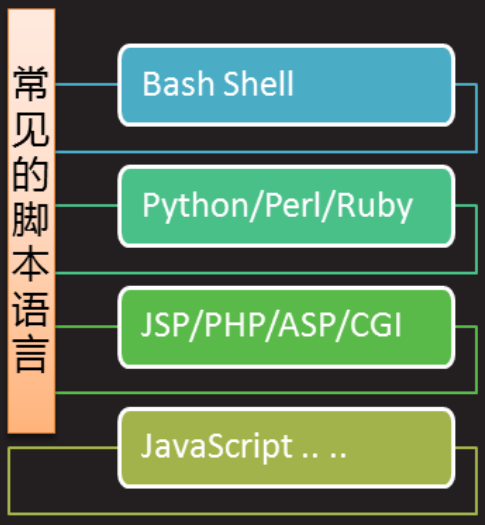
Tedu.cn 达内教育

第一个Shell脚本

什么是Shell脚本？

知识讲解

- 提前写好可执行语句，能够完成特定任务的文件
  - 顺序、批量化处理
  - 解释型程序



Shell版HelloWorld的诞生

知识讲解

- 脚本创建“三步走”
  - 1. 新建文本文件
  - 2. 添加可执行的脚本语句（命令行）
  - 3. 添加 x 执行权限

```
[root@svr5 ~]# vim  /root/first.sh                  //1. 建文件
echo  'Hello World'                                 //2. 写脚本语句
[root@svr5 ~]# chmod  +x  /root/first.sh            //3. 加执行权限

[root@svr5 ~]# /root/first.sh
Hello World
```

Tedu.cn 达内教育

脚本构成及执行

Tedu.cn 达内教育

规范的脚本构成

知识讲解

- #! 脚本声明（使用哪种解释器）
- # 注释信息（步骤、思路、用途、变量含义等）
- 可执行的语句

```
[root@svr5 ~]# vim  /root/first.sh
#!/bin/bash                          //Sha-Bang调用标记
# A  test  program  for  Shell-Script //注释信息
echo  'Hello World'                  //可执行的脚本语句或命令行
... ..
```

脚本的执行方式

知识讲解

- 方法一，作为“命令字”
  - 指定脚本文件的路径，前提是有 x 权限
- 方法二，作为“参数”
  - sh 脚本文件路径
  - source 脚本文件路径
  - . 脚本文件路径

不要求 x 权限

```
[root@svr5 ~]# sh  /root/first.sh
Hello World
[root@svr5 ~]# .  /root/first.sh
Hello World
```

调试Shell脚本

知识讲解

- 主要途径
  - 直接观察执行中的输出、报错信息
  - 通过 sh -x 开启调试模式
  - 在可能出错的地方设置echo

```
[root@svr5 ~]# sh  -x  /root/first.sh
+ echo 'Hello World'
Hello World
```

简单脚本应用

示例1：快速配置YUM

知识讲解

- 为新装的客户机配好Yum仓库
  - 软件源位于 file:///misc/cd
  - 通过脚本建立 /etc/yum.repos.d/rhel6.repo 文件

```
[root@svr5 ~]# cat  /root/el6repo.sh
#!/bin/bash
rm  -rf  /etc/yum.repos.d/*.repo                #//清理配置目录
echo  '[rhel-packages]
name=Red Hat Enterprise Linux 6
baseurl=file:///misc/cd
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release'  >
/etc/yum.repos.d/rhel7.repo                     #//新建仓库配置文件
```

示例2：快速搭建FTP服务

知识讲解

- 为新装的客户机搭好vsftpd服务
  - 装包、起服务、设开机自运行
  - 通过脚本实现上述操作

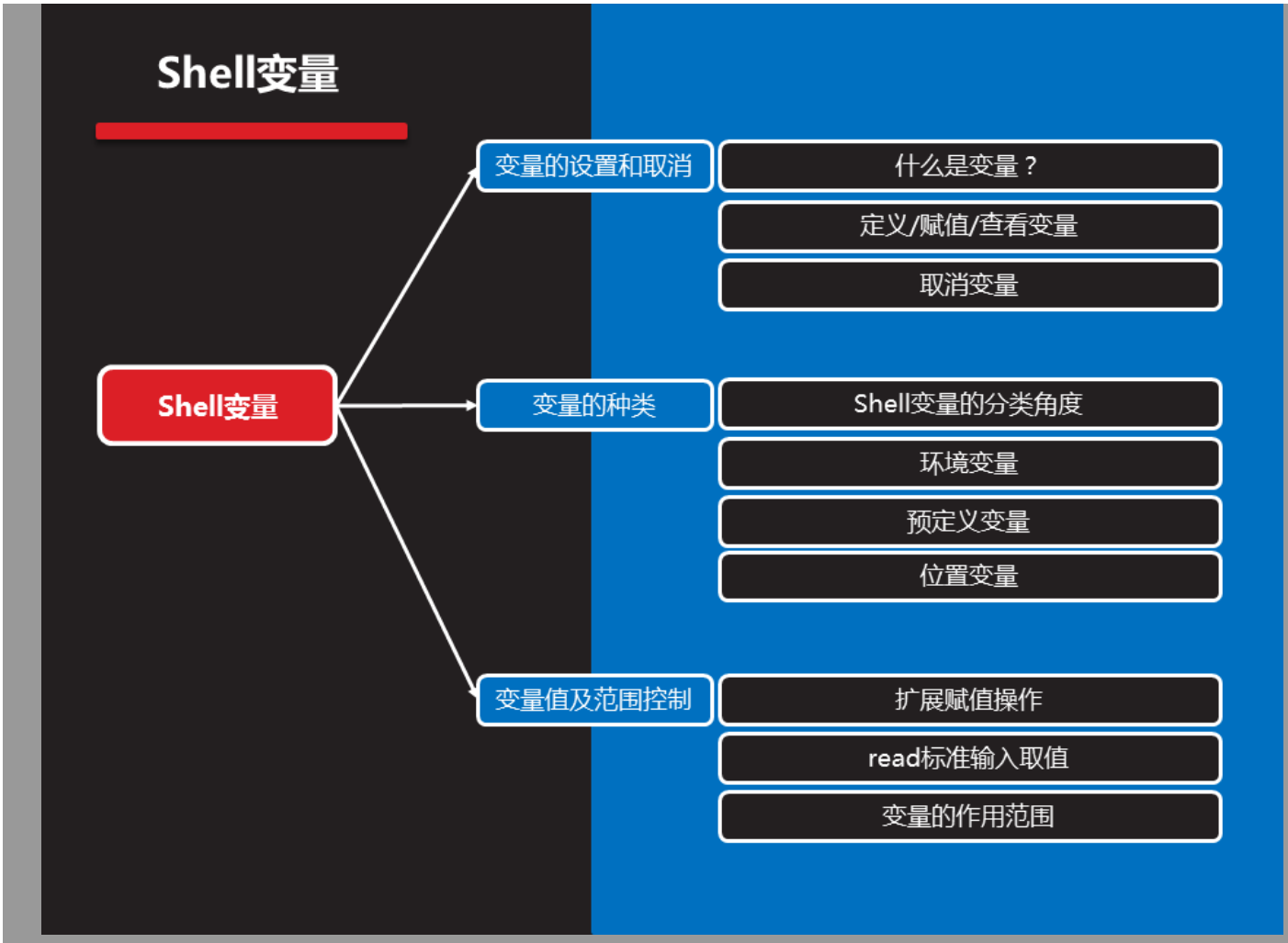
```
[root@svr5 ~]# cat  /root/ftpon.sh
#!/bin/bash
yum  -y  install  vsftpd  &> /dev/null          #//装包，忽略输出
systemctl start  vsftpd                        #//起服务
systemctl enable  vsftpd                       #//设为开机自运行
```

案例2：简单Shell脚本的设计

课堂练习

编写三个脚本程序，分别实现以下目标

- 1）在屏幕上输出一段文字“Hello World”
- 2）能够为本机快速配好Yum仓库
- 3）能够为本机快速装配好vsftpd服务



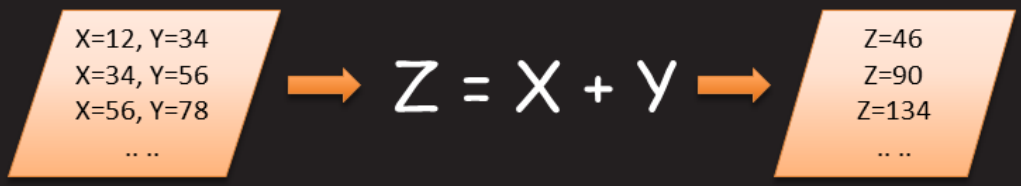
Tedu.cn
达内教育

变量的设置和取消

什么是变量？

- 以固定名称存放，可能会变化的值
  - 提高脚本对任务需求、运行环境变化的适应能力
  - 方便在脚本中重复使用

知识讲解



定义/赋值/查看变量

- 定义/赋值变量
  - 变量名=变量值

知识讲解

```
[root@svr5 ~]# X=12                  //定义变量 X，赋值 12
[root@svr5 ~]# var1=CentOS           //定义变量 var1，赋值 CentOS
```

相关注意事项：
1. 若指定的变量名已存在，相当于为此变量重新赋值
2. 等号两边不要有空格
3. 变量名由字母/数字/下划线组成，区分大小写
4. 变量名不能以数字开头，不要使用关键字和特殊字符

定义/赋值/查看变量（续1）

知识讲解

- 查看变量
  - 引用变量值：$变量名
  - 查看变量值：echo $变量名、echo ${变量名}

```
[root@svr5 ~]# echo $X, $var1
12, CentOS
[root@svr5 ~]# echo $var16.5      //未定义的变量无取值
.5
[root@svr5 ~]# echo ${var1}6.5    //变量名易混淆时，以{}界定
CentOS6.5
```

取消变量

知识讲解

- 变量的失效
  - 退出定义变量的Shell环境时，变量会自动失效
  - 也可手动取消：unset 变量名 .. ..

```
[root@svr5 ~]# unset X
[root@svr5 ~]# echo $X

[root@svr5 ~]#
```

Tedu.cn 达内教育

变量的种类

Tedu.cn 达内教育

Shell变量的分类角度

知识讲解

- 存储类型
  - 整数型、浮点型、双精度浮点型、字符型、……
  - Shell脚本语言对存储类型要求较松散
- 使用类型

类 型	说 明
环境变量	变量名通常都大写，由系统维护，用来设置工作环境，只有个别变量用户可以直接更改
位置变量	bash内置，存储执行脚本时提供的参数
预定义变量	bash内置，一类有特殊用途的变量，可直接调用，但不能直接赋值或修改
自定义变量	由用户自主设置、修改及使用

环境变量

知识讲解

- 配置文件
  - /etc/profile、~/.bash_profile
- 相关操作
  - env：列出所有的环境变量
  - set：列出所有变量
- 常见的环境变量
  - PWD、PATH、USER 、LOGNAME、UID
  - SHELL、HOME、PS1、PS2、……

预定义变量

知识讲解

- 用来保存脚本程序的执行信息
  - 直接使用这些变量
  - 不能直接为这些变量赋值

变量名	含 义
$0	当前所在的进程或脚本名
$$	当前运行进程的PID号
$?	命令执行后的返回状态，0表示正常，1或其他值表示异常
$#	已加载的位置变量的个数
$*	所有位置变量的值

位置变量

知识讲解

- 在执行脚本时提供的命令行参数
  - 表示为 $n，n为序号
  - $1、$2、.. .. ${10}、${11}、.. ..

```
[root@svr5 ~]# cat  /root/a.sh
#!/bin/bash
echo  $1  ${10}                    //查看第1、10个位置参数

[root@svr5 ~]# /root/a.sh 1 2 3 4 5 6 7 8 9 10 11
1 10
```

位置变量（续1）

知识讲解

- 应用示例：快速添加用户，并设好登录密码
  - 在执行脚本时，提供用户名作为参数
  - 将登录密码设为 1234567

```
[root@svr5 ~]# cat  /root/uad.sh
#!/bin/bash
useradd  $1  2>  /tmp/err.log
echo  1234567 | passwd  --stdin  $1  &> /dev/null
```

Tedu.cn 达内教育

案例3：使用Shell变量

1. 定义/赋值/查看变量
2. 环境/预定义/位置变量的应用
   - 环境变量PWD、USER、HOME、SHELL
   - 预定义变量$0、$$、$?、$#、$*
   - 位置变量$1、$2、$10、……

课堂练习

Tedu.cn 达内教育

变量值及范围控制

扩展赋值操作

知识讲解

- 区分三种定界符
  - 双引号 " "：允许扩展，以 $ 引用其他变量
  - 单引号 ' '：禁用扩展，即便 $ 也视为普通字符
  - 反撇号 ``：将命令的执行输出作为变量值

```
[root@svr5 ~]# echo "当前用户是：$USER"
当前用户是：root
[root@svr5 ~]# echo '当前用户是：$USER'
当前用户是：$USER
[root@svr5 ~]# echo 当前工作目录：$(pwd)
当前工作目录：/root
```

$( ) 与 `` 等效，但 $( ) 更方便嵌套使用

read标准输入取值

知识讲解

- read 从键盘读入变量值完成赋值
  - 格式： read [-p "提示信息"] 变量名
  - -p可选，-t 可指定超时秒数
- 终端显示控制
  - stty -echo：关闭终端输出（无显示）
  - stty echo：恢复终端输出（显示）

变量的作用范围

知识讲解

- 局部变量
  - 新定义的变量默认只在当前Shell环境中有效
  - 无法在子Shell环境中使用
- 全局变量
  - 全局变量在当前Shell及子Shell环境中均有效
  - 使用export可将局部变量声明为全局变量

export 局部变量名[=变量值].. .. ：为局部变量添加全局属性
export -n 全局变量名.. ..：取消指定变量的全局属性

变量的作用范围（续1）

知识讲解

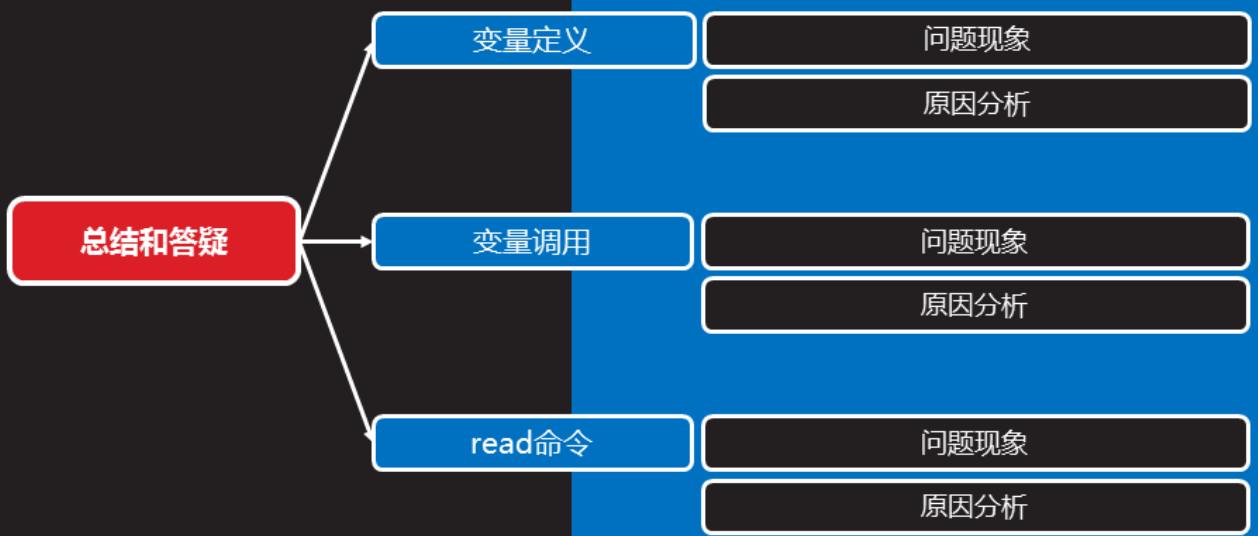
```
[root@svr5 ~]# SCHOOL="Tarena IT Group."
[root@svr5 ~]# sh                          //进入子Shell环境
sh-3.2# echo $SCHOOL
                                           //无此变量，输出空行
sh-3.2# exit                               //返回原Shell环境
exit
[root@svr5 ~]# export SCHOOL               //声明为全局变量
[root@svr5 ~]# sh                          //再次进入子Shell
sh-3.2# echo $SCHOOL
Tarena IT Group.                           //可成功使用该变量
```

案例4：变量的扩展应用

课堂练习

1. 三种引号对赋值的影响
2. 使用read命令从键盘读取变量值
3. 使用export发布全局变量

总结和答疑



Tedu.cn 达内教育

变量定义

Tedu.cn 达内教育

问题现象

- 故障错误信息

知识讲解

```
[root@svr5 ~]# 2test=123
-bash: 2test=123: command not found
[root@svr5 ~]# my#test=123
-bash: my#test=123: command not found
[root@svr5 ~]# my@13=33
-bash: my@13=33: command not found
```

原因分析

知识讲解

- 分析故障
  - 报错信息：my#test=123: command not found
- 分析故障原因
  - 变量名称不可以使用数字开头
  - 变量名不可以为特殊字符
  - 变量名可以为字母、数字、下划线

变量调用

问题现象

Tedu.cn 达内教育

知识讲解

- 故障错误信息

```
[root@svr5 ~]# test=12
[root@svr5 ~]# echo test
test
[root@svr5 ~]# echo $testyuan
```

原因分析

Tedu.cn 达内教育

知识讲解

- 分析故障
  - 调用变量未获得预期的值
- 分析故障原因
  - 调用变量时需要使用$加变量名
  - 调用变量时，为了防止歧义需要使用{}

Tedu.cn 达内教育

read命令

Tedu.cn 达内教育

问题现象

- 故障错误信息

知识讲解

```
[root@svr5 ~]# read -p "Please input pass:"pass
Please input pass:pass
[root@svr5 ~]# echo $pass

[root@svr5 ~]#
```

原因分析

知识讲解

- 分析故障
  - read提示符和变量之间缺少空格
  - 导致读取变量内容为空
- 分析故障原因
  - 因为没有空格，read将提示符和变量识别为一个整体