SHELL脚本编程基础

讲师：王晓春

本章内容

- 编程基础
- 脚本基本格式
- 变量
- 运算
- 条件测试
- 配置用户环境

编程基础

- 程序
  - 程序：算法+数据结构
  - 数据：是程序的核心
  - 数据结构：数据在计算机中的类型和组织方式
  - 算法：处理数据的方式
- 程序编程风格：
  - 过程式：以指令为中心，数据服务于指令
  - 对象式：以数据为中心，指令服务于数据
- shell程序：提供了编程能力，解释执行

面向对象

程序的执行方式

- 计算机：运行二进制指令
- 编程语言：人与计算机之间交互的语言
- 低级编程语言：

  机器：二进制的0和1的序列，称为机器指令。与自然语言差异太大，难懂、难写

  汇编：用一些助记符号替代机器指令，称为汇编语言

  如：ADD A,B 将寄存器A的数与寄存器B的数相加得到的数放到寄存器A中

  汇编语言写好的程序需要汇编程序转换成机器指令

  汇编语言稍微好理解，即机器指令对应的助记符，助记符更接近自然语言
- 高级编程语言：

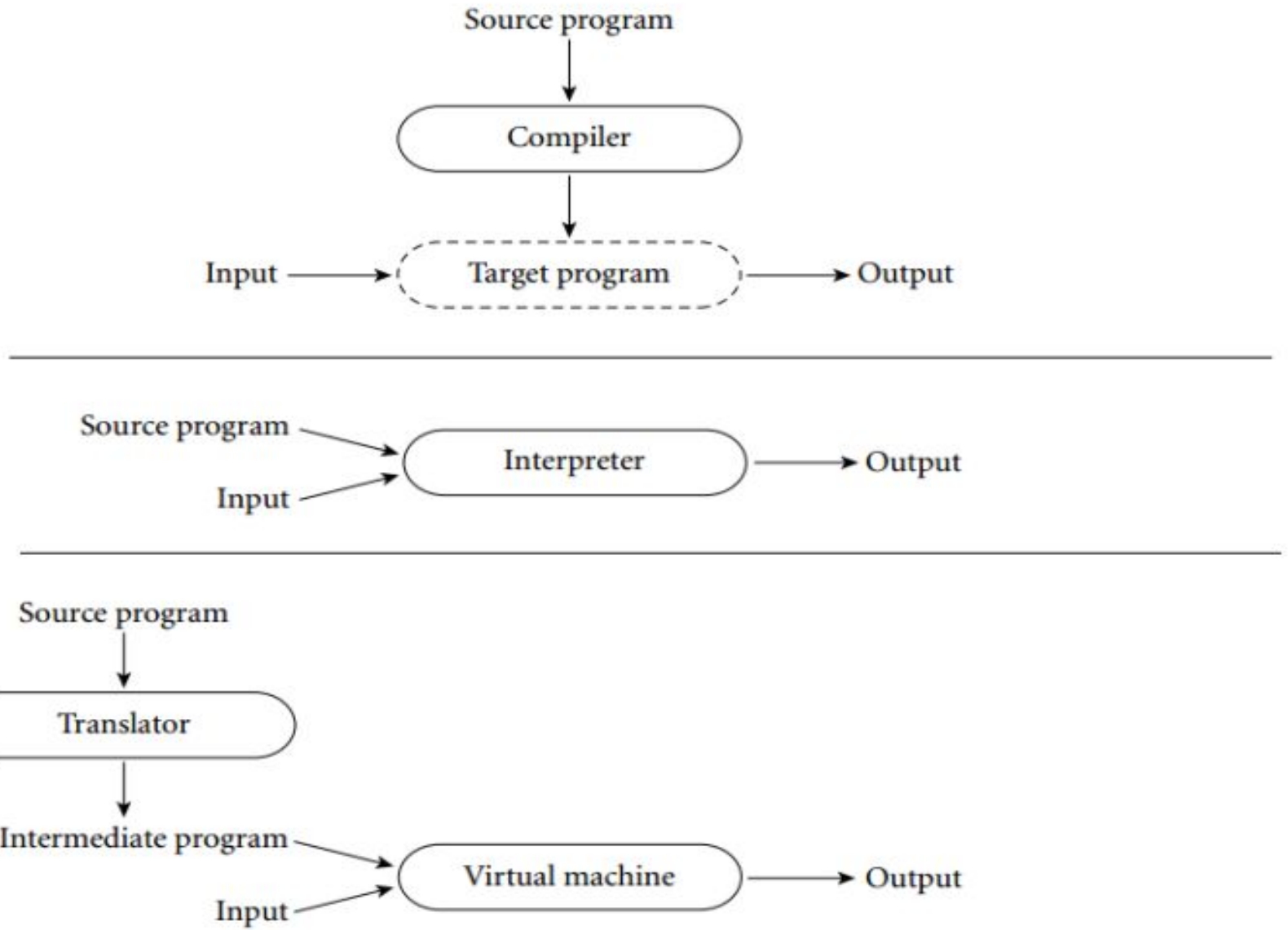
  编译：高级语言-->编译器-->机器代码-->执行

  C，C++

  解释：高级语言-->执行-->解释器-->机器代码

  shell，python，php，JavaScript，perl

编译和解释型语言



编程基本概念

- 编程逻辑处理方式：
  - 顺序执行
  - 循环执行
  - 选择执行
- shell编程：过程式、解释执行
  - 编程语言的基本结构：
    - 各种系统命令的组合
    - 数据存储：变量、数组
    - 表达式: a + b
    - 语句:if

shell脚本基础

- shell脚本:

  包含一些命令或声明，并符合一定格式的文本文件

- 格式要求：首行shebang机制

  #!/bin/bash

  #!/usr/bin/python

  #!/usr/bin/perl

- shell脚本的用途有：
  - 自动化常用命令
  - 执行系统管理和故障排除
  - 创建简单的应用程序
  - 处理文本或文件

创建shell脚本

- 第一步：使用文本编辑器来创建文本文件
  - 第一行必须包括shell声明序列：#!

    #!/bin/bash
  - 添加注释

    注释以#开头
- 第二步：运行脚本
  - 给予执行权限，在命令行上指定脚本的绝对或相对路径
  - 直接运行解释器，将脚本作为解释器程序的参数运行

脚本规范

- 脚本代码开头约定
  1、第一行一般为调用使用的语言
  2、程序名，避免更改文件名为无法找到正确的文件
  3、版本号
  4、更改后的时间
  5、作者相关信息
  6、该程序的作用，及注意事项
  7、最后是各版本的更新简要说明

脚本的基本结构

- 脚本的基本结构

  #!SHEBANG

  CONFIGURATION_VARIABLES

  FUNCTION_DEFINITIONS

  MAIN_CODE

shell脚本示例


```
#!/bin/bash
# ----------------------------------------
# Filename:    hello.sh
# Revision:    1.1
# Date:        2017/06/01
# Author:      wang
# Email:       wang@gmail.com
# Website:     www.magedu.com
# Description: This is the first script
# ----------------------------------------
# Copyright:   2017 wang
# License:     GPL
 echo   “hello world”
```

脚本调试

- 检测脚本中的语法错误

```
bash -n /path/to/some_script
```

- 调试执行

```
bash -x /path/to/some_script
```

变量

- 变量：命名的内存空间
  数据存储方式：
  字符：
  数值：整型，浮点型
- 变量：变量类型
  作用：
  1、数据存储格式
  2、参与的运算
  3、表示的数据范围
  类型：
  字符
  数值：整型、浮点型

变量

- 强类型：变量不经过强制转换，它永远是这个数据类型，不允许隐式的类型转换。一般定义变量时必须指定类型、参与运算必须符合类型要求；调用未声明变量会产生错误

  如：java ， c# ， python

  如：print（‘magedu’ + 10)　　提示出错，不会自动转换类型

  print（‘magedu’ +str(10))　　结果为magedu10，需要显示转换类型

- 弱类型：语言的运行时会隐式做数据类型转换。无须指定类型，默认均为字符型；参与运算会自动进行隐式类型转换；变量无须事先定义可直接调用

  如：bash 不支持浮点数，php，javascript

- 变量命名法则：

  1、不能使程序中的保留字：例如if, for

  2、只能使用数字、字母及下划线，且不能以数字开头

  3、见名知义

  4、统一命名规则：驼峰命名法

bash中变量的种类

- 根据变量的生效范围等标准划分下面变量类型：

局部变量：生效范围为当前shell进程；对当前shell之外的其它shell进程，包括当前shell的子shell进程均无效

环境（全局）变量：生效范围为当前shell进程及其子进程

本地变量：生效范围为当前shell进程中某代码片断，通常指函数

位置变量：$1, $2, ...来表示，用于让脚本在脚本代码中调用通过命令行传递给它的参数

特殊变量：$?, $0, $*, $@, $#,$$

局部变量

- 变量赋值：name= ‘value’
- 可以使用引用value:
  - (1) 可以是直接字串; name= “root"
  - (2) 变量引用：name="$USER"
  - (3) 命令引用：name=`COMMAND` name=$(COMMAND)
- 变量引用：${name} $name
  - "": 弱引用，其中的变量引用会被替换为变量值
  - '': 强引用，其中的变量引用不会被替换为变量值，而保持原字符串
- 显示已定义的所有变量：set
- 删除变量：unset name

练习

- 1、编写脚本/root/bin/systeminfo.sh,显示当前主机系统信息，包括主机名，IPv4地址，操作系统版本，内核版本，CPU型号，内存大小，硬盘大小
- 2、编写脚本/root/bin/backup.sh，可实现每日将/etc/目录备份到/root/etcYYYY-mm-dd中
- 3、编写脚本/root/bin/disk.sh,显示当前硬盘分区中空间利用率最大的值
- 4、编写脚本/root/bin/links.sh,显示正连接本主机的每个远程主机的IPv4地址和连接数，并按连接数从大到小排序

环境变量

- 变量声明、赋值：
  - export name=VALUE
  - declare -x name=VALUE
- 变量引用：$name, ${name}
- 显示所有环境变量：
  - env
  - printenv
  - export
  - declare -x
- 删除变量：
  - unset name

环境变量

- bash内建的环境变量：
  - PATH
  - SHELL
  - USER
  - UID
  - HOME
  - PWD
  - SHLVL
  - LANG
  - MAIL
  - HOSTNAME
  - HISTSIZE
  - —

只读和位置变量

- 只读变量：只能声明，但不能修改和删除
  - 声明只读变量：

    readonly name

    declare -r name
  - 查看只读变量：

    readonly –p
- 位置变量：在脚本代码中调用通过命令行传递给脚本的参数

  $1, $2, ...：对应第1、第2等参数，shift [n]换位置

  $0: 命令本身

  $*: 传递给脚本的所有参数，全部参数合为一个字符串

  $@: 传递给脚本的所有参数，每个参数为独立字符串

  $#: 传递给脚本的参数的个数

  $@ $* 只在被双引号包起来的时候才会有差异

  set -- 清空所有位置变量

退出状态

- 进程使用退出状态来报告成功或失败
  - 0 代表成功，1 - 255代表失败
  - $? 变量保存最近的命令退出状态
- 例如：

```
ping -c1 -W1 hostdown &> /dev/null
echo $?
```

退出状态码

- bash自定义退出状态码

  exit [n]：自定义退出状态码

  注意：脚本中一旦遇到exit命令，脚本会立即终止；终止退出状态取决于exit命令后面的数字

  注意：如果未给脚本指定退出状态码，整个脚本的退出状态码取决于脚本中执行的最后一条命令的状态码

算术运算

- bash中的算术运算:help let

  +, -, *, /, %取模（取余），**（乘方）

  实现算术运算：

  (1) let var=算术表达式

  (2) var=$[算术表达式]

  (3) var=$((算术表达式))

  (4) var=$(expr arg1 arg2 arg3 ...)

  (5) declare –i var = 数值

  (6) echo ‘算术表达式’ | bc

- 乘法符号有些场景中需要转义，如*
- bash有内建的随机数生成器：$RANDOM（0-32767）

  echo $[$RANDOM%50]：0-49之间随机数

赋值

- 增强型赋值：

  +=, -=, *=, /=, %=

- let varOPERvalue

  例如:let count+=3

  自加3后自赋值

- 自增，自减：

  let var+=1

  let var++

  let var-=1

  let var--

练习

- 1、编写脚本/root/bin/sumid.sh，计算/etc/passwd文件中的第10个用户和第20用户的ID之和
- 2、编写脚本/root/bin/sumspace.sh，传递两个文件路径作为参数给脚本，计算这两个文件中所有空白行之和
- 3、编写脚本/root/bin/sumfile.sh,统计/etc, /var, /usr目录中共有多少个一级子目录和文件

逻辑运算

- true, false
  1, 0
- 与:
  1 与 1 = 1
  1 与 0 = 0
  0 与 1 = 0
  0 与 0 = 0
- 或:
  1 或 1 = 1
  1 或 0 = 1
  0 或 1 = 1
  0 或 0 = 0

逻辑运算

- 非：！

  ！1 = 0　　！true

  ！0 = 1　　！false

- 短路运算

  短路与

  第一个为0，结果必定为0

  第一个为1，第二个必须要参与运算

  短路或

  第一个为1，结果必定为1

  第一个为0，第二个必须要参与运算

- 异或：^

  异或的两个值,相同为假，不同为真

条件测试

- 判断某需求是否满足，需要由测试机制来实现

  专用的测试表达式需要由测试命令辅助完成测试过程
- 评估布尔声明，以便用在条件性执行中
  - 若真，则返回0
  - 若假，则返回1
- 测试命令：
  - test EXPRESSION
  - [EXPRESSION]
  - [[EXPRESSION]]

  注意：EXPRESSION前后必须有空白字符

条件性的执行操作符

- 根据退出状态而定，命令可以有条件地运行
  - && 代表条件性的AND THEN
  - || 代表条件性的OR ELSE
- 例如：

```
grep -q no_such_user /etc/passwd \
|| echo 'No such user'
No such user
ping -c1 -W2 station1 &> /dev/null  \
>    && echo "station1 is up"          \
>    || (echo 'station1 is unreachable'; exit 1)
station1 is up
```

test命令

- 长格式的例子：

```
test  "$A" =  "$B" && echo "Strings are equal"
test  “$A”  -eq  “$B”  && echo "Integers are equal"
```

- 简写格式的例子：

```
[ "$A" =  "$B" ] && echo "Strings are equal"
[ "$A" -eq "$B" ] && echo "Integers are equal"
```

bash的数值测试

- -v VAR

  变量VAR是否设置

- 数值测试：

  -gt 是否大于

  -ge 是否大于等于

  -eq 是否等于

  -ne 是否不等于

  -lt 是否小于

  -le 是否小于等于

bash的字符串测试

- 字符串测试：

  = 是否等于

  > ascii码是否大于ascii码

  < 是否小于

  != 是否不等于

  =~ 左侧字符串是否能够被右侧的PATTERN所匹配

  注意: 此表达式一般用于[[ ]]中；扩展的正则表达式

  -z "STRING “ 字符串是否为空，空为真，不空为假

  -n "STRING “ 字符串是否不空，不空为真，空为假

- 注意：用于字符串比较时的用到的操作数都应该使用引号

练习

- 1、编写脚本/root/bin/argsnum.sh，接受一个文件路径作为参数；如果参数个数小于1，则提示用户“至少应该给一个参数”，并立即退出；如果参数个数不小于1，则显示第一个参数所指向的文件中的空白行数
- 2、编写脚本/root/bin/hostping.sh，接受一个主机的IPv4地址做为参数，测试是否可连通。如果能ping通，则提示用户“该IP地址可访问”；如果不可ping通，则提示用户“该IP地址不可访问”
- 3、编写脚本/root/bin/checkdisk.sh，检查磁盘分区空间和inode使用率，如果超过80%，就发广播警告空间将满

Bash的文件测试

- 存在性测试
  - -a FILE：同-e
  - -e FILE: 文件存在性测试，存在为真，否则为假
- 存在性及类别测试
  - -b FILE：是否存在且为块设备文件
  - -c FILE：是否存在且为字符设备文件
  - -d FILE：是否存在且为目录文件
  - -f FILE：是否存在且为普通文件
  - -h FILE 或 -L FILE：存在且为符号链接文件
  - -p FILE：是否存在且为命名管道文件
  - -S FILE：是否存在且为套接字文件

Bash的文件权限测试

- 文件权限测试：
  - -r FILE：是否存在且可读
  - -w FILE: 是否存在且可写
  - -x FILE: 是否存在且可执行
- 文件特殊权限测试：
  - -u FILE：是否存在且拥有suid权限
  - -g FILE：是否存在且拥有sgid权限
  - -k FILE：是否存在且拥有sticky权限

Bash的文件属性测试

- 文件大小测试：
  - -s FILE: 是否存在且非空
- 文件是否打开：
  - -t fd: fd 文件描述符是否在某终端已经打开
  - -N FILE：文件自从上一次被读取之后是否被修改过
  - -O FILE：当前有效用户是否为文件属主
  - -G FILE：当前有效用户是否为文件属组

Bash的文件属性测试

- 双目测试：

  FILE1 -ef FILE2: FILE1是否是FILE2的硬链接

  FILE1 -nt FILE2: FILE1是否新于FILE2（mtime）

  FILE1 -ot FILE2: FILE1是否旧于FILE2

Bash的组合测试条件

- 第一种方式：

  COMMAND1 && COMMAND2 并且

  COMMAND1 || COMMAND2 或者

  ! COMMAND 非

  如：[-f “$FILE”] && [[“$FILE” =~ .*\.sh$]]

- 第二种方式：

  EXPRESSION1 -a EXPRESSION2 并且

  EXPRESSION1 -o EXPRESSION2 或者

  ! EXPRESSION

  必须使用测试命令进行，[[ ]] 不支持

- 示例：

```
[ -z “$HOSTNAME” -o $HOSTNAME "=="localhost.localdomain" ] \
        && hostname www.magedu.com
[ -f /bin/cat -a -x /bin/cat ] && cat /etc/fstab
```

练习

- 1、编写脚本per.sh,判断当前用户对指定参数文件，是否不可读并且不可写
- 2、编写脚本excute.sh ，判断参数文件是否为sh后缀的普通文件，如果是，添加所有人可执行权限，否则提示用户非脚本文件
- 3、编写脚本nologin.sh和login.sh，实现禁止和允许普通用户登录系统

使用read命令来接受输入

- 使用read来把输入值分配给一个或多个shell变量

  -p　指定要显示的提示

  -s　静默输入，一般用于密码

  -n　N 指定输入的字符长度N

  -d　‘字符’　输入结束符

  -t　N　TIMEOUT为N秒

  read　从标准输入中读取值，给每个单词分配一个变量

  所有剩余单词都被分配给最后一个变量

  read -p “Enter a filename: “ FILE

bash如何展开命令行

- 把命令行分成单个命令词
- 展开别名
- 展开大括号的声明（{}）
- 展开波浪符声明（~）
- 命令替换$()　和　``）
- 再次把命令行分成命令词
- 展开文件通配（*、?、[abc]等等）
- 准备I/0重导向（<、>）
- 运行命令

防止扩展

- 反斜线（\）会使随后的字符按原意解释

```
$ echo Your cost: \$5.00
Your cost: $5.00
```

- 加引号来防止扩展
  - 单引号（‘ ）防止所有扩展
  - 双引号（” ）也防止所有扩展，但是以下情况例外：

$（美元符号）	-	变量扩展
`（反引号）	-	命令替换
\（反斜线）	-	禁止单个字符扩展
!（叹号）	-	历史命令替换

bash的配置文件

- 按生效范围划分，存在两类：
- 全局配置：
  - /etc/profile
  - /etc/profile.d/*.sh
  - /etc/bashrc
- 个人配置：
  - ~/.bash_profile
  - ~/.bashrc

shell登录两种方式

- 交互式登录：

  (1)直接通过终端输入账号密码登录

  (2)使用“su - UserName” 切换的用户

  执行顺序：/etc/profile --> /etc/profile.d/*.sh --> ~/.bash_profile --> ~/.bashrc --> /etc/bashrc

- 非交互式登录：

  (1)su UserName

  (2)图形界面下打开的终端

  (3)执行脚本

  (4)任何其它的bash实例

  执行顺序： /etc/profile.d/*.sh --> /etc/bashrc -->~/.bashrc

Profile类

- 按功能划分，存在两类：
  profile类和bashrc类
- profile类：为交互式登录的shell提供配置
  全局：/etc/profile, /etc/profile.d/*.sh
  个人：~/.bash_profile
  功用：
  (1) 用于定义环境变量
  (2) 运行命令或脚本

Bashrc类

- bashrc类：为非交互式和交互式登录的shell提供配置

  全局：/etc/bashrc

  个人：~/.bashrc

  功用：

  (1) 定义命令别名和函数

  (2) 定义本地变量

编辑配置文件生效

- 修改profile和bashrc文件后需生效

  两种方法:

  1重新启动shell进程

  2 . 或source

  例:

  . ~/.bashrc

Bash 退出任务

- 保存在~/.bash_logout文件中（用户）
- 在退出登录shell时运行
- 用于
  - 创建自动备份
  - 清除临时文件

$-变量

- h：hashall，打开这个选项后，Shell 会将命令所在的路径hash下来，避免每次都要查询。通过set +h将h选项关闭
- i：interactive-comments，包含这个选项说明当前的 shell 是一个交互式的 shell。所谓的交互式shell,在脚本中，i选项是关闭的。
- m：monitor，打开监控模式，就可以通过Job control来控制进程的停止、继续，后台或者前台执行等。
- B：braceexpand，大括号扩展
- H：history，H选项打开，可以展开历史列表中的命令，可以通过!感叹号来完成，例如“!!”返回上最近的一个历史命令，“!n”返回第 n 个历史命令

练习

- 1、让所有用户的PATH环境变量的值多出一个路径，例如：/usr/local/apache/bin
- 2、用户root登录时，将命令指示符变成红色，并自动启用如下别名：
  rm= ‘rm –i’
  cdnet= ‘cd /etc/sysconfig/network-scripts/’
  editnet= ‘vim /etc/sysconfig/network-scripts/ifcfg-eth0’
  editnet= ‘vim /etc/sysconfig/network-scripts/ifcfg-eno16777736 或 ifcfg-ens33 ’ (如果系统是CentOS7)
- 3、任意用户登录系统时，显示红色字体的警示提醒信息“Hi,dangerous！”
- 4、编写生成脚本基本格式的脚本，包括作者，联系方式，版本，时间，描述等
- 5、编写用户的环境初始化脚本reset.sh，包括别名，登录提示符，vim的设置，环境变量等

关于马哥教育

马哥教育
IT人的高薪职业学院

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：1661815153, 113228115
- QQ群：203585050, 279599283

祝大家学业有成

谢　谢