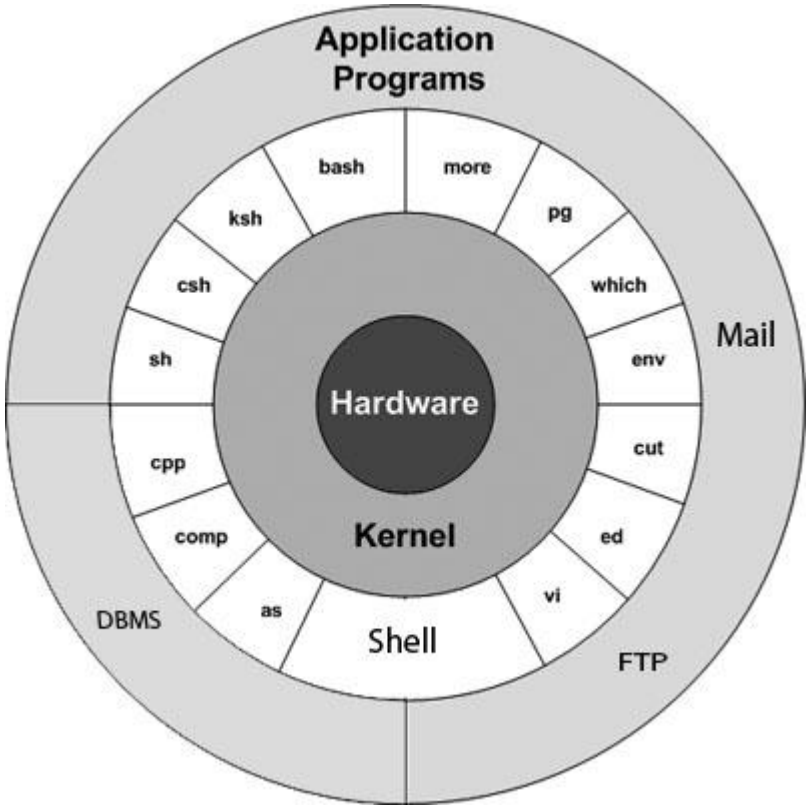
Shell脚本编程基础



生命健康信息学院 解增言

内容

①

- Linux命令帮手
  - 命令行编辑
  - screen工具
- Shell特殊字符（1）
  - 通配符
  - 正则表达式

②

- Shell特殊字符（2）
  - 引号
  - 转义符与路径符
  - 输入输出重定向
  - 注释和后台命令
  - 命令组合符
  - 成组命令

③

- Shell简介
  - Shell的概念
  - Shell的特点
  - Shell版本
  - Shell程序示例
  - Shell程序执行方法
- Shell变量
  - 用户自定义变量
  - 位置变量
  - Shell预定义变量
  - 环境变量

④

- 算术运算
- 控制结构
- 函数
- 作业控制
- Shell内置命令
- Shell脚本调试
- Shell脚本实例

内容

①

- Linux命令帮手
  - 命令行编辑
  - screen工具
- Shell特殊字符（1）
  - 通配符
  - 正则表达式

②

- Shell特殊字符（2）
  - 引号
  - 转义符与路径符
  - 输入输出重定向
  - 注释和后台命令
  - 命令组合符
  - 成组命令

③

- Shell简介
  - Shell的概念
  - Shell的特点
  - Shell版本
  - Shell程序示例
  - Shell程序执行方法
- Shell变量
  - 用户自定义变量
  - 位置变量
  - Shell预定义变量
  - 环境变量

④

- 算术运算
- 控制结构
- 函数
- 作业控制
- Shell内置命令
- Shell脚本调试
- Shell脚本实例

Shell简介

- Shell的概念
- Shell的特点
- Shell版本
- Shell程序示例
- Shell程序的执行

Shell概念

- Linux shell是连接Linux内核与用户之间的桥梁，是Linux系统的用户界面及Linux命令的解释程序。
- Shell也是一种高级程序设计语言，有变量、关键字、控制结构和函数等程序设计语言要素。

Shell的特点

- 利用控制结构（如判断、循环等）和管道将Linux命令组合起来
- 使用正则表达式
- 可以直接使用shell内置命令，而无须创建新的进程
- 灵活使用数据流，提供管道和输入/输出重定向机制，方便数据处理
- 提供后台执行命令方式
- 用户可以配置环境

Shell版本

- **Bourne shell (sh)**
  - Steven Bourne (AT&T Bell)
- **C shell (csh)**
  - Bill Joy (美国加州大学伯克利分校)
- **Korn shell (ksh)**
  - David Korn (AT&T Bell)
- **Bourne Again shell (bash)**
  - GNU
- **Debian Almquist Shell (dash)**
  - 简化版的bash，Ubuntu默认的shell

Shell程序示例

```
$ cat shell_example.sh        #显示脚本内容
mkdir animal
cd animal
mkdir bird
touch fox

$ sh shell_example.sh         #运行脚本
```

Shell程序的执行

4种方式（shell脚本文件名为example.sh）：

- sh <example.sh 或 bash <example.sh
- sh example.sh 或 bash example.sh
- path/example.sh （需有执行权限并在脚本文件第一行指定解释器的路径，如：#!/bin/bash）
- . example.sh 或 source example.sh

CentOS中，sh是bash的符号链接；
Ubuntu中，sh是dash的符号链接

Shell变量类型

- 用户自定义变量
- 位置变量
- Shell预先定义的特殊变量
- 环境变量

自定义变量：命名

- **变量命名规则**
  - 字母或下划线开头
  - 字母、数字或下划线组成
  - 区分大小写

ps：变量名应能体现其含义，实际编写Shell脚本时

不建议：tmp temp tmp1 tmp_1 var1 var2 …

建议：protein_human plant_cds flag …

自定义变量：赋值

- **变量赋值**
  - 变量名=字符串，如：

```
myfile=tmp/c.fa                    #等号两边不能有空格
mypath=/home/xiezy/$myfile         #字符串中可以包含变量
name="Zhang San"                   #字符串中有空格要用引号
```

- **命令替换（将命令执行结果赋值给变量）**
  - current_dir=`pwd`
  - current_dir=$(pwd)
- **输入（read）**
  - –read a

自定义变量：数组

- bash只提供一维数组
- 数组大小没有限定
- 数组的下标从0开始
- 赋值：

```
city[0]=Beijing
week=(Mon Tue Sun)
```

自定义变量：引用

- 变量直接引用
- 变量内容修改后引用
  - 变量内容删除
  - 变量内容部分替换
  - 变量内容整体替换
- 数组引用

变量直接引用

- **格式：$var 或 ${var}**

- **例1**

```
color=red                #给变量color赋值red
echo ${color}            #输出变量color的值red
echo $color              #输出red，此处变量名后无其他内容，省略了大括号
```

- **例2**

```
color=background
echo ${color}_color      #输出background_color，此处大括号不能省略
```

变量内容删除

- **变量内容头部删除：**

  ${var#pattern}：最短匹配

  ${var##pattern}：最长匹配

- **变量内容尾部删除：**

  ${var%pattern}：最短匹配

  ${var%%pattern}：最长匹配

path=/bin:/usr/bin:/usr/local/bin:/home/peter/bin

${path#*:}	=	~~/bin:~~/usr/bin:/usr/local/bin:/home/peter/bin	到第一个冒号
${path##*:}	=	~~/bin:/usr/bin:/usr/local/bin:~~/home/peter/bin	到最后一个冒号
${path%:*}	=	/bin:/usr/bin:/usr/local/bin~~:/home/peter/bin~~	到第一个冒号
${path%%:*}	=	/bin~~:/usr/bin:/usr/local/bin:/home/peter/bin~~	到最后一个冒号

变量内容部分替换

${var/old_pattern/new_pattern}：替换第一个模式

${var//old_pattern/new_pattern}：替换所有的模式

如果：path=/bin:/usr/bin:/usr/local/bin:/home/peter/bin

则：

${path/bin/BIN}　　= /BIN/:/usr/bin:/usr/local/bin:/home/peter/bin

${path//bin/BIN}　　= /BIN/:/usr/BIN/:/usr/local/BIN/:/home/peter/BIN/

变量内容整体替换

变量设置方式	a没有设置	a为空字符串	a为非空字符串
b=${a-str}	b=str	b=	b=$a
b=${a:-str}	b=str	b=str	b=$a
b=${a+str}	b=	b=str	b=str
b=${a:+str}	b=	b=	b=str
b=${a=str}	a=str b=str	a不变 b=	a不变 b=$a
b=${a:=str}	a=str b=str	a=str b=str	a不变 b=$a
b=${a?str}	str => stderr	b=	b=$a
b=${a:?str}	str => stderr	str => stderr	b=$a

根据变量a的内容，决定变量b等于str还是$a，以及变量a等于str还是不变

应用实例

- 测试变量username是否存在，如不存在则给其赋值root，如存在则使用原来的值。

  username=${username-root}

- 测试变量username是否存在或为空，如果不存在或为空值，则username和group都赋值peter，若变量username存在且不为空值则username的值不变，group赋值为username的值。

  group=${username:=peter}

数组引用

- 数组第i个元素
  - –${array[i]}
- 数组所有元素
  - –${array[*]}
  - –${array[@]}
- 数组第i个元素的长度
  - –${#array[i]}
- 数组元素的个数
  - –${#array[*]}
  - –${#array[@]}

自定义变量：变量输出

- **echo**

  $echo $a

- **printf**

  $pi=3.1415926

  $printf “%-6.4f\n” $pi　　#输出总宽度为6个字符，如果不够6个字符则右侧留空白，小数点后保留4位（四舍五入）

  $printf “%8.2f\n” $pi　　#输出总宽度为8个字符，如果不够8个字符则左侧留空白，小数点后保留2位（四舍五入）

自定义变量：删除

- **unset**

  unset var

注意：readonly定义的只读变量无法删除

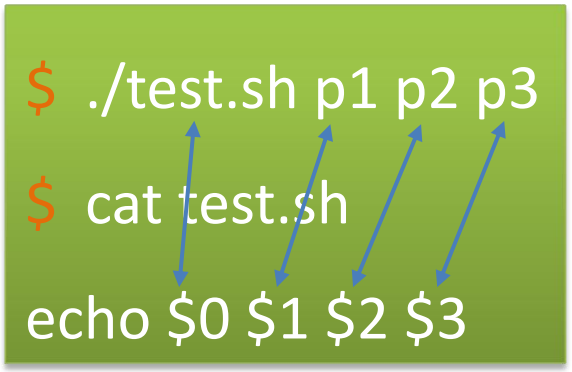
位置变量

•程序运行时提供的参数（即命令行参数或位置参数）在程序内有对应的位置变量，如：

./test.sh p1 p2 p3，在程序中对应的位置变量分别是：$0（脚本名字）、$1（p1）、$2（p2）、$3（p3）

•用set命令修改或设置位置变量：

```
#!/bin/bash
set Hello world
echo $1 $2
```



位置变量的值=对应的位置参数

移动位置参数

- 使用内置命令**shift**移动位置参数：

```
$cat positional _parameter.sh
#!/bin/bash
echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
shift
echo $1 $2 $3 $4 $5 $6 $7 $8 $9
shift 4
echo $1 $2 $3 $4 $5 $6 $7 $8 $9
```

```
$./positional_parameter.sh 1 2 3 4 5 6 7 8 9
./positional_parameter.sh 1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9
6 7 8 9
```

shell预定义的特殊变量

- **$#** 命令行参数的个数
- **$*** 所有命令行参数，间隔符号为IFS
- **$@** 所有命令行参数，无间隔符号
- **$?** 上一条命令执行的返回值
- **$$** 当前进程的进程号
- **$!** 上一个后台命令的进程号
- **$-** 由当前Shell设置的执行选项组成的字符串

环境变量

- **环境变量**指在操作系统中用来指定操作系统运行环境的一些参数。
- Linux的环境变量分为 **shell变量**和**用户变量**两种。shell变量包含用户变量。
- 可以用**unset**命令删除环境变量

显示环境变量

- **显示单个环境变量**
  - –echo $HOME
- **显示所有环境变量**
  - set/declare　　#显示当前 Shell 中定义的所有变量，包括用户的环境变量和自定义变量
  - env/export/declare -x　　#显示用户环境变量，但不会显示用户自定义变量

环境变量设置

- **export命令**

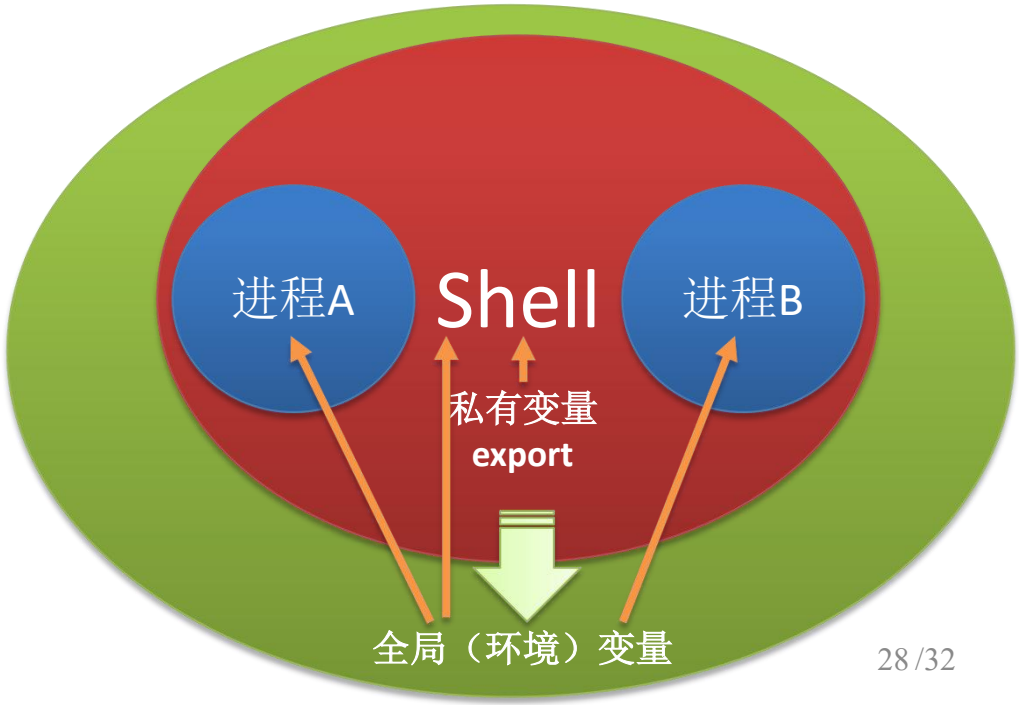
  $ export PATH=$PATH:/usr/bin

或：

  $ PATH=$PATH:/usr/bin

  $ export PATH

  *注意： export PATH的PATH前没有$



环境文件

bash的环境文件：

.bash_profile

.bashrc

.bash_logout

修改环境文件：

$ vi ~/.bash_profile

$ echo “http_proxy=somehost” >>~/.bash_profile

修改后立即生效：

$ source ~/.bash_profile

例：配置历史命令环境

- **显示配置：**
  - $ echo $HISTFILE
  - $ echo $HISTSIZE

- **修改配置：**
  - $ export HISTFILE=new_file_path
  - $ export HISTSIZE=n

  或者将上面两句写入主目录下的.bashrc文件

环境变量

Bash常用环境变量：

–HOME	当前用户主目录
–PATH	命令查找路径
–LOGNAME	当前用户的登录名
–UID	当前用户的ID
–SHELL	当前用户的Shell类型
–TERM	终端类型
–PWD	当前工作目录的绝对路径名
–MAIL	当前用户的邮件存放目录
–HISTSIZE	保存历史命令记录的条数
–HOSTNAME	主机名称
–PS1	主命令提示符
–PS2	从命令提示符
–IFS	输入域分隔符

课后作业8

（1）编辑shell脚本city.sh，利用数组存放5个城市的名字（"Bei Jing" "Shang Hai" "Tian Jin" "Chong Qing" "Guang Zhou"），并利用for循环打印出来，每行一个城市；

（2）编辑shell脚本param.sh，分两行输出命令行参数的个数（1行）及所有的命令行参数（1行）；

（3）编辑shell脚本var.sh，利用变量内容整体替换实现：如果第1个命令行参数为空，则变量param的值为"No parameter"（不包括引号），如果不为空，则param的值为第一个命令行参数，并用echo输出param的值；

（4）编辑shell脚本prevar.sh，分两行输出当前进程的进程号（1行）及其父进程（也就是运行脚本prevar.sh的shell）的进程号（1行）。


未完待续
LOADING