Shell编程笔记

1. 命令执行

$()和``都可以用来执行命令，例如

示例1：

|  |
| --- |
| cmd=$(date "+%Y-%m-%d")  echo $cmd # 一般情况等价于echo ${cmd}  输出： |

也可以使用echo打印执行命令的输出，echo `cmd`，但是之后调用$?获取的返回参数不再是cmd而是echo。

示例2：

|  |
| --- |
| cmd=`date "+%Y-%m-%d"`  echo $cmd  输出： |

但如果cmd只是一个带系统命令的字符串，直接echo则为字符串。

示例3：

|  |
| --- |
| cmd="date \"+%Y-%m-%d\""  echo $cmd  输出： |

这种情况下如果想执行命令，则需要使用eval（eval会扫描命令行并替换，再执行命令，注意eval的返回值和扫描命令的执行结果相同）

示例4：

|  |
| --- |
| cmd="date \"+%Y-%m-%d\""  eval $cmd  输出： |

1. 脚本包含

被包含的脚本会先执行，一行只能包含一个shell脚本

. parent.sh

source parent.sh # 推荐

1. 格式

第一行必须写#!/bin/bash

1. 字符串操作
2. sed：文本处理工具，再不修改文件的情况下对文本字符串修改并打印，一般和cat/grep之类配合管道命令使用

使用正则表达式替换字符串：sed 's/\t/ /g'

1. 变量
2. 作用域：默认全局，局部变量需要加local前缀
3. 赋值：‘=’两边不能有空格

|  |
| --- |
| A="123"  echo “A = ”$A  unset A # 取消变量  echo “A = ”$A  输出： |

1. 环境变量

|  |
| --- |
| export FILE\_NAME="test.sh"  echo $FILE\_NAME  输出： |

1. 预定义变量

|  |
| --- |
| echo "Current PID = "$$ # 当前进程ID  date  echo "Return state of last command = "$? # 上一条指令执行的返回状态  输出： |

1. 计算

|  |
| --- |
| echo $(((2+3)\*2)  sum=`expr 2 + 3`  echo `expr $sum \\* 2`  输出： |