1-2章笔记

一、变量替换总结

例子1:
variable_1=“I love you,Do you love me”

1、${变量#匹配规则}　　# 从头开始匹配，最短删除
2、${变量##匹配规则}　　# 从头开始匹配，最长删除
控制台输出：

```
sunwj@sunwjdeMacBook-Pro ~ % echo $variable_1
I love you,Do you love me
sunwj@sunwjdeMacBook-Pro ~ % var1=${variable_1#*ov}
sunwj@sunwjdeMacBook-Pro ~ % echo $var1
e you,Do you love me
sunwj@sunwjdeMacBook-Pro ~ % var2=${variable_1##*ov}
sunwj@sunwjdeMacBook-Pro ~ % echo $var2
e me
```

例子2:
3、${变量%匹配规则}　　# 从尾开始匹配，最短删除
4、${变量%%匹配规则}　　# 从尾开始匹配，最长删除

```
sunwj@sunwjdeMacBook-Pro ~ % var3=${variable_1%ov*}
sunwj@sunwjdeMacBook-Pro ~ % echo $var3
I love you,Do you l
sunwj@sunwjdeMacBook-Pro ~ % var4=${variable_1%%ov*}
sunwj@sunwjdeMacBook-Pro ~ % echo $var4
I l
```

例子3:
5、${变量/旧字符串/新字符串}　　# 替换变量内的旧字符串为新字符串，只替换第一个
6、${变量//旧字符串/新字符串}　　# 替换变量内的旧字符串为新字符串，全部替换

```
sunwj@sunwjdeMacBook-Pro ~ % echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
sunwj@sunwjdeMacBook-Pro ~ % var5=${PATH}
sunwj@sunwjdeMacBook-Pro ~ % echo var5
var5
sunwj@sunwjdeMacBook-Pro ~ % echo $var5
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
sunwj@sunwjdeMacBook-Pro ~ % var5=${PATH/bin/BIN}
sunwj@sunwjdeMacBook-Pro ~ % echo $var5
/usr/local/BIN:/usr/bin:/bin:/usr/sbin:/sbin
sunwj@sunwjdeMacBook-Pro ~ %
sunwj@sunwjdeMacBook-Pro ~ % var6=${PATH//bin/BIN}
sunwj@sunwjdeMacBook-Pro ~ % echo $var6
/usr/local/BIN:/usr/BIN:/BIN:/usr/sBIN:/sBIN
```

二、字符串处理

1、计算字符串长度

方法1：${#String}
方法2：expr length $String 注意在Mac系统没有expr这个命令

```
sunwj@sunwjdeMacBook-Pro ~ % var1="Hello World"
sunwj@sunwjdeMacBook-Pro ~ % len=${#var1}
sunwj@sunwjdeMacBook-Pro ~ % echo $len
11
```

注意:使用expr,索引计数是从1开始计算；使用${string:posotion}，索引计数是从0开始计数

2、获取字符索引位置
　　方法：expr index “$string” substr

例子：
　　var1=“quicstart is a app”
　　ind=`expr index “$var1” start`

3、获取子串长度
　　方法：expr match “$string” subset
例子：
　　var1=“quicstart is a app”
　　sub_len=`expr match “$var1” app`

4、抽取字符串中的子串

方法一：

1、${String:position}

2、${String:position:length}

3、${String:-position} 或者 ${String:(position)}

方法二：

expr substrate $string $position $length

例子：

var1="kafka hadoop yarn mapreduce"

1、${String:position}

```
sunwj@sunwjdeMacBook-Pro ~ % var1="kafka hadoop yarn mapreduce"
sunwj@sunwjdeMacBook-Pro ~ % substr_1=${var1:10}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_1
op yarn mapreduce
sunwj@sunwjdeMacBook-Pro ~ %
```

2、${String:position:length}

```
sunwj@sunwjdeMacBook-Pro ~ % var1="kafka hadoop yarn mapreduce"
sunwj@sunwjdeMacBook-Pro ~ % substr_1=${var1:10}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_1
op yarn mapreduce
sunwj@sunwjdeMacBook-Pro ~ % substr_2=${var1:5:10}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_2
 hadoop ya
```

3、${String: -position} 或者 ${String:(position)}

```
sunwj@sunwjdeMacBook-Pro ~ % substr_3=${var1: -5}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_3
educe
sunwj@sunwjdeMacBook-Pro ~ % substr_3=${var1:(-5)}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_3
educe
sunwj@sunwjdeMacBook-Pro ~ % substr_3=${var1: -5:2}
sunwj@sunwjdeMacBook-Pro ~ % echo $substr_3
ed
```

练习：

需求描述：

变量string=“Bigdata process framework is Hadoop,Hadoop is an open source project”

执行脚本之后，打印输出string字符串变量，并给出用户以下选项：

1、打印string长度
2、删除字符串中所有Hadoop
3、替换第一个Hadoop为Mapreduce
4、替换全部Hadoop为Mapreduce

用户输入数字1｜2｜3｜4，可以执行对应项的功能；输入q｜Q则退出交互模式

思路分析：

1、将不同的功能模块划分，并编写函数

function print_tips 打印提示信息函数
function len_of_string 打印string长度函数
function del_hadoop 删除字符串中所有Hadoop
function rep_hadoop_mapreduce_first 替换第一个Hadoop为Mapreduce
function rep_hadoop_mapreduce_all 替换全部Hadoop为Mapreduce

2、实现第一步定义的功能函数

print_tips
len_of_string
del_hadoop
rep_hadoop_mapreduce_first
rep_hadoop_mapreduce_all

3、程序主流程设计

程序脚本：

lx_string.sh
shell脚本
1KB

2-6 命令替换

有两种方法：

方法一：`command`
方法二：$(command)

例子 1:
获取系统得所有用户并输出

```
sunwj@sunwjdeMacBook-Pro udian_cloud % cat /etc/passwd | cut -d ":" -f
1
##
# User Database
#
# Note that this file is consulted directly only when the system is ru
nning
# in single-user mode.  At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
nobody
root
daemon
_uucp
_taskgated
```

脚本：

```
#!/bin/bash
#

index=1

for user in `cat /etc/passwd |cut -d ":" -f 1`
do
        echo "This is $index user: $user"
        index=$(($index + 1))
done
```

例子 2:
根据系统时间计算今年或明年

```
sunwj@sunwjdeMacBook-Pro shell_script % echo "This is $(date +%Y) year"

This is 2020 year
sunwj@sunwjdeMacBook-Pro shell_script % echo "This is $(($(date +%Y) +1)) y
ear"
This is 2021 year
```

例子 3:

根据系统时间获取今年还剩下多少星期，已经过了多少星期

```
vim
#!/bin/bash
#

echo "This year have passed $(($(date +%j)/7)) weeks"

echo "This is $((365 - $(date +%j))) days before new year"
echo "This is $(((365 - $(date +%j))/7)) weeks before new year"
~
```

例子 4:

判定 nginx 进程是否存在，若不存在则自动拉起该进程

```
#!/bin/bash
#

wechat_process_num=$(ps -ef | grep WeChat | grep -v grep | wc -l)

if [[ $wechat_process_num -eq 0 ]]; then
    echo "systemctl start WeChat"
fi
```

总结:

``和$() 两者是等价的，推荐初学者使用$(),易于掌握；缺点是极少数 UNIX 可能不支持，但都支持``

$(()) 主要用来进行整数运算，包括加减乘除，引用变量前面可以加$，也可以不加$

例子:

$(((100 + 30) / 13))

num1=20;num2=30
((num++))
((num—))

$(($num1+$num2*2))

2-8 有类型变量

declare和typeset命令

把某一个变量声明成特定的类型，需要用此两个命令声明

declare命令和typeset命令两者等价

declare、typeset命令都是用来定义变量类型的

declare命令参数解释

-r 将变量设为只读

```
sunwj@sunwjdeMacBook-Pro ~ % var2="hello world"
sunwj@sunwjdeMacBook-Pro ~ % var2="hello python"
sunwj@sunwjdeMacBook-Pro ~ % echo $var2
hello python
sunwj@sunwjdeMacBook-Pro ~ % declare var2="hello world"
sunwj@sunwjdeMacBook-Pro ~ % var2="1111"
sunwj@sunwjdeMacBook-Pro ~ % echo $var2
1111
sunwj@sunwjdeMacBook-Pro ~ % declare -r var2
sunwj@sunwjdeMacBook-Pro ~ % var2="hello world"
zsh: read-only variable: var2
```

-I 将变量设置为整数

```
sunwj@sunwjdeMacBook-Pro ~ % num1=10
sunwj@sunwjdeMacBook-Pro ~ % num2=$+20
sunwj@sunwjdeMacBook-Pro ~ % num2=$num1+20
sunwj@sunwjdeMacBook-Pro ~ % echo $num2
10+20
sunwj@sunwjdeMacBook-Pro ~ % declare -i num1
sunwj@sunwjdeMacBook-Pro ~ % declare -i num2
sunwj@sunwjdeMacBook-Pro ~ % num1=10
sunwj@sunwjdeMacBook-Pro ~ % num2=$num1+20
sunwj@sunwjdeMacBook-Pro ~ % echo $num2
30
```

Liunx 默认是将变量当字符串处理的

-a 将变量定义为数组

```
sunwj@sunwjdeMacBook-Pro ~ % array=("1" "2" "3" "4")
sunwj@sunwjdeMacBook-Pro ~ % echo ${array[0]}

sunwj@sunwjdeMacBook-Pro ~ % echo ${array[1]}
1
sunwj@sunwjdeMacBook-Pro ~ % echo ${array[2]}
2
sunwj@sunwjdeMacBook-Pro ~ % echo ${#array[@]}
4
sunwj@sunwjdeMacBook-Pro ~ % echo ${array[@]}
1 2 3 4
sunwj@sunwjdeMacBook-Pro ~ % echo ${#array[1]}
1
sunwj@sunwjdeMacBook-Pro ~ % array=("1" "2" "3" "44444")
sunwj@sunwjdeMacBook-Pro ~ % echo ${#array[4]}
5
```

-f 显示此脚本定义过的所有函数及内容
-F 仅显示此脚本前定义过的函数名
-x 将变量声明为环境变量

2-9 bash数学运算之expr(上)

1、语法格式

方法一、expr $num1 operator $num2

expr $num1 + $num2

方法二、$(($num1 operator $num2))

2、常用的操作符
部分操作符需要转译。如 | < > * 等

操作符	含义
num1 | num2	num1不为空且非0，返回num1；否则返回num2
num1 & num2	num1不为空且非0，返回num1；否则返回0
num1 < num2	num1小于num2，返回1；否则返回0
num1 <= num2	num1小于等于num2，返回1；否则返回0
num1 = num2	num1等于num2，返回1；否则返回0
num1 != num2	num1不等于num2，返回1；否则返回0
num1 > num2	num1大于num2，返回1；否则返回0
num1 >= num2	num1大于等于num2，返回1；否则返回0

操作符	含义
num1 + num2	求和
num1 - num2	求差
num1 * num2	求积
num1 / num2	求商
num1 % num2	求余

3、

2-10 bash数学运算之expr(下)

计算 1+2+3+.....+num的和

lx_sum.sh
shell脚本
453字节

2-11 bash数学运算之bc

1、bc是bash内建的运算器，支持浮点数运算

```
sunwj@sunwjdeMacBook-Pro ecop-recharge % bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.

23 + 5
28
23 - 5
18
23 * 5
115
23 / 5
4
23 % 5
3
```

```
scale=2

23 / 7
3.28

scale=6
23 / 7
3.285714
```

```
sunwj@sunwjdeMacBook-Pro ecop-recharge % echo "23+35" | bc
58
sunwj@sunwjdeMacBook-Pro ecop-recharge % echo "23.5+35" | bc
58.5
sunwj@sunwjdeMacBook-Pro ecop-recharge % echo "scale=4;23.5+35" | bc
58.5
```

2、内建变量scale可以设置，默认为0

3、操作符对照表

bc操作符对照表

操作符	含义
num1 + num2	求和
num1 - num2	求差
num1 * num2	求积
num1 / num2	求商

4、脚本

lx_bc.sh
shell脚本
128字节

```
sunwj@sunwjdeMacBook-Pro shell_script % ./bc.sh
num1: 6
num2: 2.1
6 / 2.1 = 2.8571
```