# shell基础知识

## 1.1echo 和printf

### 1.1.1echo

**1)echo取消自动换行**

默认echo xxx 是会自动换行的,通过参数 -n 能取消换行

echo -n

**2)echo转义字符**

echo默认不认识转义字符,需要加上-e参数

echo -e “\t hahah”

echo -e ‘\t hahah’

**3)echo单双引号区别**

a=hello

echo ‘$a haha’ #结果:$a haha

echo “$a haha”#结果:hello haha

**4)echo 感叹号的输出**

echo hello! ok

echo ‘hello!’ ok

echo “hello!” error

所以对于各种特殊符号的输出用单引号将符号单独包围住

echo “hello”’!’

### 1.1.2printf

**1)printf换行符**

printf默认不会加换行符,要自己添加\n

**2)printf单双引号的区别和echo一样**

a=hello

print ‘$a’ #输出:$a

printf “$a”输出:hello

print $a 输出:hello 和双引号有点像

**3)printf %和参数个数不一致**

printf “hello%s” 1 2 3

#结果:

hello1hello2hello3

意思就是相当于来了个for循环,并且把输出的结果连起来

**4)printf格式化**

printf “%5s” a #5个字符的宽度,不足的左边补上空格

结果: a

printf “%-5s” a #5个字符的宽度,不足5个字符右变补空格

结果:a

printf “%6.3f” 4.3 #结果:4.300

分析:宽度为6位其中”.”也包含在宽度6中,3表示保留小数点3位,不足3为添0



## 1.2命令执行顺序

**1)分号”;”**

command1;command2

命令之间没有逻辑关系,命令从前往后执行,前面的命令执行错了,也不影响后面命令的执行

**2)&&**

command1&&command2

逻辑与,命令从前往后执行,command1执行成功后,才执行command2命令,command1执行失败,command2不会被执行

判断是否执行成功主要看回传码 echo $?

**3)||**

command1 || command2

逻辑或,命令总前往后执行,当command1执行失败后才会执行command2,当command1执行成功后,不会执行command2

**4)&**

command1 & command2

这个和”;”一样,没有逻辑关系,只是先执行command1后放入后台,再执行command2

## 1.3变量

1.变量的赋值str=hello 注意等号左右没有空格。如果有空格就是进行比较运算符的比较运算了

2.变量的引用:$str或者${str}

3.删除变量:unset str

4.查看所有变量set或declare不加参数

5.定义只读变量readonly str=hello,注意str将不能被修改或者unset删除了,只能退出脚本,或者退出shell

6.环境变量:export str可以临时将普通变量变为环境变量,永久的环境变量要在/etc/profile里定义

7.获取变量的长度str=hello;echo ${#str}

8.变量内容的删除和替换:

删除和替换,原来的str不会变化

**从左向右匹配,**

str=hello,world,hello,linux,linux

删除最短的: echo ${str#\*hello,} 格式 ${变量名称#被删除的字符串}

结果:~~hello~~,world,hello,linux,linux

删除最长的echo ${str#\*hello,}

结果:~~hello,world,hello,~~linux,linux

**从右往左匹配(实际上只是从右边看起,还是从左往右):**

删除最短的: echo ${str%,linux\*}

hello,world,hello,linux~~,linux~~

删除最长的:echo ${str%%,linux\*}

hello,world,hello~~,linux,linux~~

替换一个字符串:echo ${str/hello/lw} 格式: ${变量/旧字符串/新字符串}

lw,world,hello,linux,linux

替换所有字符串:echo ${str//hello/lw} 格式: ${变量//旧字符串/新字符串}

lw,world,lw,linux,linux

|  |  |
| --- | --- |
| ${变量#关键词} ${变量##关键词} | 若变量内容从头开始的数据符合『关键词』，则将符合的最短数据删除 若变量内容从头开始的数据符合『关键词』，则将符合的最长数据删除 |
| ${变量%关键词} ${变量%%关键词} | 若变量内容从尾向前的数据符合『关键词』，则将符合的最短数据删除 若变量内容从尾向前的数据符合『关键词』，则将符合的最长数据删除 |
| ${变量/旧字符串/新字符串} ${变量//旧字符串/新字符串} | 若变量内容符合『旧字符串』则『第一个旧字符串会被新字符串取代』 若变量内容符合『旧字符串』则『全部的旧字符串会被新字符串取代』 |

9.变量的赋值

echo ${str-hello}:如果str没有被定义,输出hello,str不变

echo ${str=hello}:”=”和”-”很像,输出hello,str也变为hello了

echo ${str:-hello}:如果str没有被定义或者字符串为空,则输出hello,str不变

echo ${str:=hello}:”:=”和”:-”很像,输出hello,str也变为hello了

echo ${str+hello}:”+”和”-”相反,即如果字符串定义了,则输出hello,str不变

echo ${str:+hello}:”:+”和”:-”相反即如果字符串不为空,则输出hello,str不变

10.取子字符串

格式:${str:offset:length}

str=hello;

echo ${str:0} #hello

echo ${str:1} #ello

echo ${str:2:2} #ll

11.特殊变量

$$ 当前SHELL的PID

$# 参数的个数,如果是脚本传参,则不算脚本名称,如果是函数,则不算函数名称

$\* 参数的整体,你可以想象\*就代表所有,整体的意思比如你传入了 a b c d:在for循环时写

for i in “$\*”就是 for i in “a b c d”其实只循环了一次

$@ 所有单个参数,比如传入了a b c d四个参数,for i in “$@”会循环4次,其实就相当于

for i in “a” “b” “c” “d”

$0 脚本名,注意可能是函数名

$1..$n 参数位置

## 1.4算术运算

let i=i-1 其他:let i=$i-1, let i-=1,

i=$((i-1)) 其他:i=$(($i-1))

i=$[ i - 1 ] 其他:i=$[ $i - 1 ]

echo $((i=i-1)) 其他:echo $((i=$i-1))

bc命令:其实如果只是简单的加减乘除上面的方法够用了,如果要用到高级的运算,那只能用bc了,例如如何开平方等等

bc方法1:

echo "scale=3;r=3;5/r" | bc #注意scale只对除法、取余、乘幂有效,缺点,如果计算量大了,则写在一行太挤了

bc方法2:

[root@localhost ~]# bc<<EOF

> r=1.3

> scale=3

> r/2

> EOF

.650

[root@localhost ~]#

## 1.5重定向

**1).将stdout和stderr都重定向同一个文件中**

a. find / -name ls > a.txt 2>&1

b. find / -name ls &> a.txt(重点,因为好理解,也简单) 注意&>>和&>很像,&>>是追加

cat 01.txt 是将文件的内容输出到标准输出(默认)

如何输出标准输入到屏幕中呢？

cat << EOF

hello,worl

EOF

这样就可以了

如何输出标准输入到01.txt中呢？

cat << EOF > 01.txt

hello,world

EOF

或者 cat >01.txt << EOF

haha

EOF

**2)双重定向tee**

tee:是可以将文件首先重定向到文件中,再将内容输出到标准输出

-:表示标准输入

参数: -a 是追加到文件中

cat a.txt | tee b.log|cat –

cat << EOF | tee 01.txt

hello,world

EOF

hello,world

## 1.6数组

定义数组:

1. arr=(1 2 3) 注意:不能加上逗号,加上”()”表示数组

echo ${arr[0]} #1

echo ${arr[1]} #2

echo ${arr[2]} #3

错误写法: arr=(1,2,3);这样也是个数组只是只有一个元素 echo ${arr[0]} #1,2,3

1. arr[0]=1;arr[1]=1;arr[2]=2;

数组的长度:

echo ${#arr[\*]}或者 ${#arr[@]}

关联数组

declare -A arr #先声明一个关联数组

arr=([name]=lw [age]=14)

echo ${arr[name]} #lw

arr[name]=bob #修改

echo ${arr[\*]}或者${arr[@]} 查看所有

echo ${!arr[@]}或者${!arr[\*]} 查看索引列表结果为lw 14