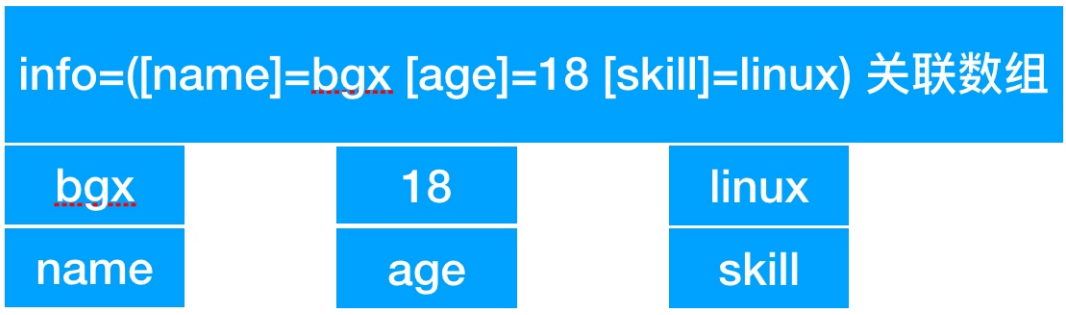
07.Shell数组应用

> 徐亮伟, 江湖人称标杆徐。多年互联网运维工作经验，曾负责过大规模集群架构自动化运维管理工作。擅长Web集群架构与自动化运维，曾负责国内某大型电商运维工作。
> 个人博客"徐亮伟架构师之路"累计受益数万人。
> 笔者Q:552408925、572891887
> 架构师群:471443208

1.数组分类

> 普通数组：只能使用整数 作为数组索引
> 关联数组：可以使用字符串 作为数组索引



2.普通数组

1.数组赋值方式

```
//针对每个索引进行赋值
[root@Shell ~]# array1[0]=pear
[root@Shell ~]# array1[1]=apple
[root@Shell ~]# array1[2]=orange
[root@Shell ~]# array1[3]=peach

//一次赋多个值,数组名=(多个变量值)
[root@Shell ~]# array2=(tom jack alice)
[root@Shell ~]# array3=(tom jack alice "bash shell")
[root@Shell ~]# array4=(1 2 3 "linux shell" [20]=puppet)

//将该文件中的每一个行作为一个元数赋 值给数组 array3
[root@Shell ~]# array5=(`cat /etc/passwd`)
```

2.查看数组赋值结果

```
[root@Shell ~]# declare -a
```

3.访问数组元数

```
//统计数组元数的个数
[root@Shell ~]# echo ${#array1[@]}
4

//访问数组中的第一个元素
[root@Shell ~]# echo ${array1[0]}
pear

//从数组索引1开始
[root@Shell ~]# echo ${array1[@]:1}
apple orange peach

//从数组索引1开始，访问两个元素
[root@Shell ~]# echo ${array1[@]:1:2}
apple orange

//访问数组中所有数据,相当于echo ${array1[*]}
[root@Shell ~]# echo ${array1[@]}
pear apple orange peach
```

4.获取数组索引

```
//获取数组元数的索引
[root@Shell ~]# echo ${!array1[@]}
0 1 2 3
```

3.关联数组

1.定义关联数组, 申明是关联数据

```
[root@Shell ~]# declare -A tt_array_1
[root@Shell ~]# declare -A tt_array_2
```

2.给关联数组进行赋值

```
数组名[索引]=变量值
[root@Shell ~]# tt_array1[index1]=pear
[root@Shell ~]# tt_array1[index2]=apple
[root@Shell ~]# tt_array1[index3]=orange
[root@Shell ~]# tt_array1[index4]=peach

//给关联数组一次赋多个值
[root@Shell ~]# tt_array2=([index1]=tom [index2]=jack [index3]=alice [index4]='bash
 shell')
```

3.查看关联数组

```
[root@Shell ~]# declare -A
```

4.访问数据元数

```
//访问数组中的第二个元数
[root@Shell ~]# echo ${tt_array2[index2]}
jack

//访问数组中所有元数 等同于 echo ${array1[*]}
[root@Shell ~]# echo ${tt_array2[@]}
bash shell tom jack alice

//访问数组中所有元数的索引
```

```
[root@Shell ~]# echo ${!tt_array2[@]}
index4 index1 index2 index3
```

4.遍历数组

> 1.通过数组元数的个数进行遍历(不推荐)
> 2.通过数组元数的索引进行遍历(推荐)
> 注意: 将统计的对象作为数组的索引, 仅针对关联数据

1.数据赋值与遍历

```
#!/usr/bin/bash
while read line
do
    hosts[++i]=$line
done </etc/hosts

echo "hosts first: ${hosts[1]}"
echo


//for循环遍历方式
for i in ${!hosts[@]}
do
    echo "$i: ${hosts[i]}"
done

//for
#!/usr/bin/bash
IFS=$'\n'

for line in `cat /etc/hosts`
do
    hosts[++j]=$line
done

for i in ${!hosts[@]}
do
    echo "$i: ${hosts[i]}"
done
```

2.统计`/etc/passwd`的`shell`数量

```
[root@Shell ~]# cat array_passwd_count.sh
#!/usr/bin/bash

declare -A array_passwd
#1.对数组进行赋值
while read line
do
    type=$(echo $line|awk -F ':' '{print $NF}')
    let array_passwd[$type]++
done </etc/passwd

#2.对数组进行遍历
for i in ${!array_passwd[@]}
do
    echo 索引是:$i,索引的值是: ${array_passwd[$i]}
done
```

3.统计`Nginx`日志`IP`访问次数

```
[root@Shell ~]# cat array_nginx_count.sh
#!/usr/bin/bash
# nginx log top 10 IP conut
declare -A array_nginx
#1.给关联数组的索引进行赋值
while read line
do
    type=$(echo $line|awk '{print $1}')
    let array_nginx[$type]++
done </var/log/nginx/access.log

for i in ${!array_nginx[@]}
do
    echo "IP是:$i 出现多少次${array_nginx[$i]}"
done
```

4.统计`tcp`的状态信息

```
[root@Shell ~]# cat array_ss_state.sh
#!/usr/bin/bash

declare -A array_state
type=$(ss -an |grep :80|awk '{print $2}')
#1.对数组进行的索引赋值
for i in $type
do
```

```
    let array_state[$i]++
done

#2.遍历数组
for j in ${!array_state[@]}
do
    echo "当前的状态是:$j,当前状态出现了多少次:${array_state[$j]}"
done
```