

if控制语句

case

for循环语句

while

util

break continue

if控制语句

```
1 格式一：
if [条件1]; then
    执行第一段程序
else
    执行第二段程序
fi
格式二：
if [条件1]; then
    执行第一段程序
elif [条件2]; then
    执行第二段程序
else
    执行第三段程序
fi
```

```
1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# vim if01.sh
2
3 #!/bin/sh
4 read -p "请输入y继续: " yn
5 if [ $yn = "y" ]; then
6     echo "继续执行"
7 else
8     echo "停止执行"
9 fi
```

结果：

```
1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./if01.sh
2 请输入y继续: y
```

```
3 继续执行
4 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./if01.sh
5 请输入y继续: n
6 停止执行
```

案例：判断当前路径下有没有文件夹 有就进入创建文件 没有 就创建文件夹 再进入创建文件

```
1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# vim if_dir.sh
2
3 #!/bin/sh
4 read -p "请输入文件夹的名字: " dir
5 #判断文件夹是否存在
6 if [ -e $dir ]; then
7     echo "$dir是存在的, 即将进入文件夹"
8     cd $dir
9     echo "即将创建文件名为new01.txt"
10    touch new01.txt
11 else
12     echo "该文件不存在, 即将创建该文件夹"
13     mkdir $dir
14     echo "即将进入该文件夹"
15     cd $dir
16     echo "即将创建文件名为new01.txt"
17     touch new01.txt
18 fi
19 tree
```


结果:

```
1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./if_dir.sh
2 请输入文件夹的名字: new_dir
3 该文件不存在, 即将创建该文件夹
4 即将进入该文件夹
5 即将创建文件名为new01.txt
6 .
7 └─ new01.txt
8
9 0 directories, 1 file
10 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./if_dir.sh
11 请输入文件夹的名字: new
```

```
12 该文件不存在，即将创建该文件夹
13 即将进入该文件夹
14 即将创建文件名为new01.txt
15 .
16 └─ new01.txt
17
18 0 directories, 1 file
```

case

```
case $变量名称 in
    “第一个变量内容” )
        程序段一
        ;;
    “第二个变量内容” )
        程序段二
        ;;
    *)
        其它程序段
        exit 1
esac
```



https://blog.csdn.net/weixin_43288201

```
1 [root@iZuf6gy6qwvkl6k8vp21vrZ script]# vim case01.sh
2
3 #!/bin/sh
4 read -p "请输入yes/no: " choice
5 case $choice in
6     yes | y* | Y*)
7         echo "输入了yes"
8         ;; # ;;相当于break
9     no | n* | N*)
```

```

10             echo "输入了no"
11             ;;
12         *) #相当于default
13             ;;
14     esac

```

结果:

```

1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./case01.sh
2 请输入yes/no: y
3 输入了yes
4 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./case01.sh
5 请输入yes/no: nssss
6 输入了no

```

for循环语句

形式一:

```

for (( 初始值; 限制值; 执行步阶 ))
do
    程序段
done

```

初始值: 变量在循环中的起始值

限制值: 当变量值在这个限制范围内时, 就继续进行循环

执行步阶: 每作一次循环时, 变量的变化量

declare 是 bash 的一个内建命令, 可以用来声明 shell 变量、设置变量的属性。declare 也可以写作 typeset。

declare -i s 代表强制把 s 变量当做 int 型参数运算。

https://blog.csdn.net/welxin_43268201

```

1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# vim for01.sh
2
3 #!/bin/sh
4 #显示使用declare执行为int类型
5 declare -i sum=0
6 declare -i i=0
7 for (( i=0; i<=100; i++ ))
8 do
9     sum=$sum+$i;
10 done
11 echo "sum=$sum"
12
13 for i in 1 2 3 4 5 6

```

```
14 do
15     sum=$sum+$i;
16 done
17 echo "sum2=$sum"
```

结果:

```
1 [root@iZuf6gy6qwvk16k8vp21vrZ script]# ./for01.sh
2 sum=5050
3 sum2=5071
```

while

```
while [ condition ]
do
    程序段
done
```

当 condition 成立的时候进入 while 循环，直到 condition 不成立时才退出循环。

https://blog.csdn.net/weixin_43288201

例：12_while.sh

```
#!/bin/bash
declare -i i
declare -i s
while [ "$i" != "101" ]
do
    s+=i;
    i=i+1;
done
echo "The count is $s"
```

https://blog.csdn.net/weixin_43288201

util

```
until [ condition ]  
do  
    程序段  
done
```

这种方式与 **while** 恰恰相反，当 **condition** 成立的时候退出循环，否则继续循环。
https://blog.csdn.net/weixin_43288201

例： 13_until.sh

```
#!/bin/bash  
declare -i i  
declare -i s  
until [ "$i" = "101" ]  
do  
    s+=i;  
    i=i+1;  
done  
echo "The count is $s"
```

https://blog.csdn.net/weixin_43288201

break continue

break

break 命令允许跳出循环。

break 通常在进行一些处理后退出循环或 case 语句

continue

continue 命令类似于 break 命令

只有一点重要差别，它不会跳出循环，只是跳过这个循环步

https://blog.csdn.net/weixin_43288201