

# Bash 脚本进阶，经典用法及其案例

点击关注 📌 民工哥技术之路 2 days ago

点击上方“民工哥技术之路”，选择“设为星标”

回复“1024”获取独家整理的学习资料！



## 前言

在linux中，Bash 脚本是很基础的知识，大家可能一听bash脚本感觉很高大上，像小编当初刚开始学一样，感觉会写脚本的都是大牛。虽然复杂的bash脚本是很烧脑，但是，当我们熟练的掌握了其中的用法与技巧，再多加练习，总有一天也会成为得心应手的bash脚本大牛。

脚本在生产中的作用，想必小编我不说，大家也都知道，脚本写的6，可以省下很多复杂的操作，减轻自己的工作压力。推荐大家先看看Shell 脚本编程入门最佳实践这篇文章。

## 一、条件选择、判断

### 1、条件选择if

#### (1) 用法格式

```
if 判断条件 1 ; then
    条件为真的分支代码
elif 判断条件 2 ; then
    条件为真的分支代码
elif 判断条件 3 ; then
    条件为真的分支代码
else
    以上条件都为假的分支代码
fi
```

逐条件进行判断，第一次遇为“真”条件时，执行其分支，而后结束整个if。

#### (2) 经典案例：

```
#判断年纪
#!/bin/bash
read -p "Please input your age: " age
if [[ $age =~ [^0-9] ]] ;then
    echo "please input a int"
    exit 10

elif [ $age -ge 150 ]:then
```

```
    elif [ $age -gt 150 ];then
        echo "your age is wrong"
        exit 20
    elif [ $age -gt 18 ];then
        echo "good good work,day day up"
    else
        echo "good good study,day day up"
    fi
```

**分析：**请输入年纪，先判断输入的是否含有除数字以外的字符，有，就报错；没有，继续判断是否小于150，是否大于18。

```
#判断分数
#!/bin/bash
read -p "Please input your score: " score
if [[ $score =~ [^0-9] ]] ;then
    echo "please input a int"
    exit 10
elif [ $score -gt 100 ];then
    echo "Your score is wrong"
    exit 20
elif [ $score -ge 85 ];then
    echo "Your score is very good"
elif [ $score -ge 60 ];then
    echo "Your score is soso"
else
    echo "You are loser"
fi
```

**分析：**请输入成绩，先判断输入的是否含有除数字以外的字符，有，就报错；没有，继续判断是否大于100，是否大于85，是否大于60。

## 2、条件判断 case

### (1) 用法格式

```
case $name in
PART1)
    cmd
    ;;
PART2)
    cmd
    ;;
*)
    cmd
    ;;
esac
```

注意：case 支持glob 风格的通配符：

- \*： 任意长度任意字符
- ?： 任意单个字符
- [] ： 指定范围内的任意单个字符
- a|b： a 或b

### (2) 案例：

```
#判断yes or no
```

```
#!/bin/bash
read -p "Please input yes or no: " anw
case $anw in
    [yY][eE][sS]|[yY])
        echo yes
        ;;
    [nN][oO]|[nN])
        echo no
        ;;
    *)
        echo false
        ;;
esac
```

**分析：**请输入yes or no，回答Y/y、yes各种大小写组合为yes；回答N/n、No各种大小写组合为no。

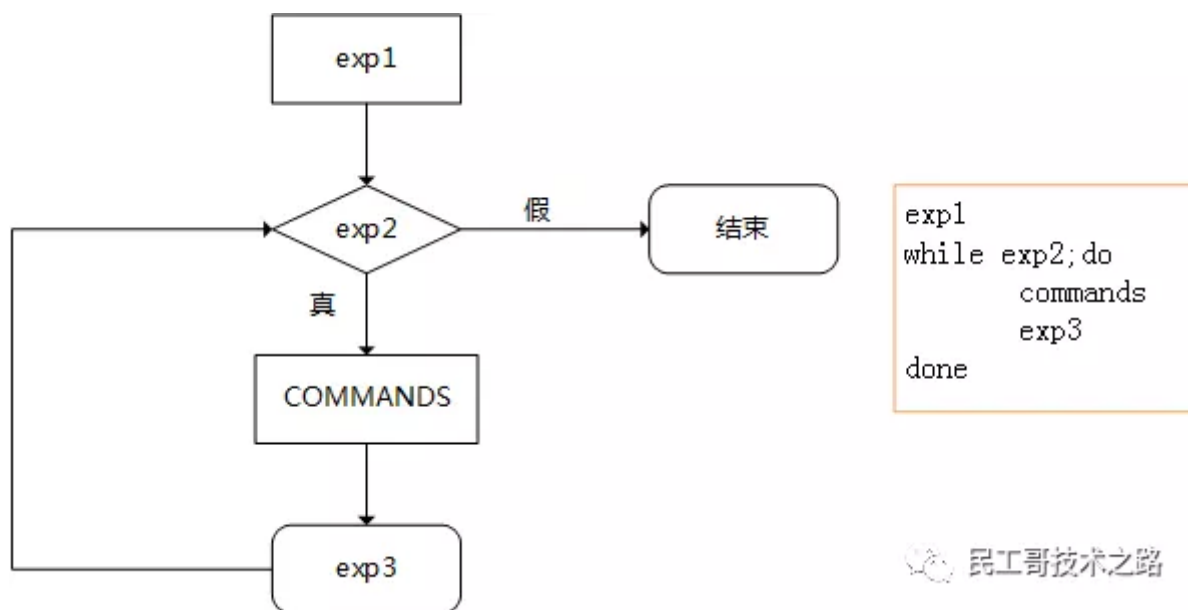
## 二、四个循环

### 1、for

#### (1) 用法格式

```
① for name in 列表 ;do
    循环体
done

② for (( exp1; exp2; exp3 )) ;do
    cmd
done
```



民工哥技术之路

exp1 只执行一次，相当于在for里嵌了while

### ③ 执行机制：

- 依次将列表中的元素赋值给“变量名”；每次赋值后即执行一次循环体；直到列表中的元素耗尽，循环结束
- 列表的表示方法，可以glob 通配符，如{1..10}、\*.sh；也可以变量引用，如：`seq 1 $name`

### (2) 案例

*#求出 (1+2+...+n) 的总和*

```
sum=0
```

```
read -p "Please input a positive integer: " num
```

```
if [[ $num =~ [^0-9] ]] ;then
```

```
    echo "input error"
```

```
elif [[ $num -eq 0 ]] ;then
```

```
    echo "input error"
```

```
fi
```

```
else
    for i in `seq 1 $num` ;do
        sum=$((sum+i))
    done
    echo $sum
fi
unset zhi
```

**分析：**sum初始值为0，请输入一个数，先判断输入的是否含有除数字以外的字符，有，就报错；没有判断是否为0，不为0进入for循环，i的范围为1~输入的数，每次的循环为sum=sum+i，循环结束，最后输出sum的值。

*#求出 (1+2+...+100) 的总和*

```
for (( i=1,num=0;i<=100;i++ ));do
    [ ${i%2} -eq 1 ] && let sum+=i
done
echo sum=$sum
```

分析：i=1,num=0；当i<=100，进入循环，若i÷2取余=1，则sum=sum+i，i=i+1。

## 2、while

### (1) 用法格式

```
while 循环控制条件 ;do
    循环
done
```

循环控制条件；进入循环之前，先做一次判断；每一次循环之后会再次做判断；条件为“true”，则执行一次循环；直到条件测试状态为“false”终止循环

(2) 特殊用法（遍历文件的每一行）：

```
while read line; do控制变量初始化
    循环体
done < /PATH/FROM/SOMEFILE
或cat /PATH/FROM/SOMEFILE | while read line; do
    循环体
done
```

依次读取/PATH/FROM/SOMEFILE文件中的每一行，且将行赋值给变量line

(3) 案例：

```
#100以内所有正奇数之和
sum=0
i=1
while [ $i -le 100 ] ;do
if [ ${i%2} -ne 0 ];then
    let sum+=i
    let i++
else
    let i++
fi
done
echo "sum is $sum"
```



```
echo sum ls sum
```

**分析：**sum初始值为0，i的初始值为1；请输入一个数，先判断输入的是否含有除数字以外的字符，有，就报错；没有当 $i < 100$ 时，进入循环，判断  $i \div 2$  取余 是否不为0，不为0时为奇数， $sum = sum + i$ ， $i + 1$ ，为0， $i + 1$ ；循环结束，最后输出sum的值。

### 3、until 循环

#### (1) 用法

```
until 循环条件 ;do  
    循环  
done
```

进入条件：循环条件为true；退出条件：循环条件为false；刚好和while相反，所以不常用，用while就行。

#### (2) 案例

```
#监控xiaoming用户，登录就杀死  
until pgrep -u xiaoming &> /dev/null ;do  
    sleep 0.5  
done  
pkill -9 -u xiaoming
```

**分析：**每隔0.5秒扫描，直到发现xiaoming用户登录，杀死这个进程，退出脚本，用于监控用户登录。

## 4、select 循环与菜单

### (1) 用法

```
select variable in list
do
    循环体命令
done
```

- ① select 循环主要用于创建菜单，按数字顺序排列的菜单项将显示在标准错误上，并显示PS3 提示符，等待用户输入
- ② 用户输入菜单列表中的某个数字，执行相应的命令
- ③ 用户输入被保存在内置变量 REPLY 中
- ④ select 是个无限循环，因此要记住用 break 命令退出循环，或用 exit 按 命令终止脚本。也可以按 ctrl+c退出循环
- ⑤ select 和 经常和 case 联合使用
- ⑥ 与for循环类似，可以省略 in list， 此时使用位置参量

### (2) 案例

```
#生成菜单，并显示选中的价钱
PS3="Please choose the menu: "
select menu in mifan huimian jiaozi babaozhou quit
do
    case $REPLY in
        1|4)
            echo "the price is 15"
            ;;
        2|3)
            echo "the price is 20"
```

```
;;  
5)  
    break  
    ;;  
*)  
    echo "no the option"  
esac  
done
```

```
[root@cetos7 bin]# ./menu.sh  
1) mifan  
2) huimian  
3) jiaozi  
4) babaozhou  
5) quit  
Please choose the menu: 3  
the price is 20  
Please choose the menu: 7  
no the option  
Please choose the menu: 5  
[root@cetos7 bin]#
```

**分析：**PS3是select的提示符，自动生成菜单，选择5break退出循环。

### 三、循环里的一些用法

#### 1、循环控制语句

##### (1) 语法

`continue [N]`：提前结束第N层的本轮循环，而直接进入下一轮判断；最内层为第1层 `break [N]`：提前结束第N层循环，最内侧为第1层

```
例：while CONDITON1; do
    CMD1
if CONDITION2; then
    continue / break
fi
    CMD2
done
```

(2) 案例：

*#①求 (1+3+...+49+53+...+100) 的和*

```
#!/bin/bash
sum=0
for i in {1..100} ;do
    [ $i -eq 51 ] && continue
    [ ${i%2} -eq 1 ] && { let sum+=i;let i++; }
done
echo sum=$sum
```

**分析：**做1+2+...+100的循环，当i=51时，跳过这次循环，但是继续整个循环，结果为：sum=2449

*#②求 (1+3+...+49) 的和*

```
#!/bin/bash
sum=0
for i in {1..100} ;do
    [ $i -eq 51 ] && break
    [ ${i%2} -eq 1 ] && { let sum+=i;let i++; }
done
echo sum=$sum
```

**分析：** 做1+2+...+100的循环，当i=51时，跳出整个循环，结果为：sum=625

## 2、循环控制shift命令

### (1) 作用

用于将参数列表list左移指定次数，最左端的那个参数就从列表中删除，其后边的参数继续进入循环

### (2) 案例：

*#①创建指定的多个用户*

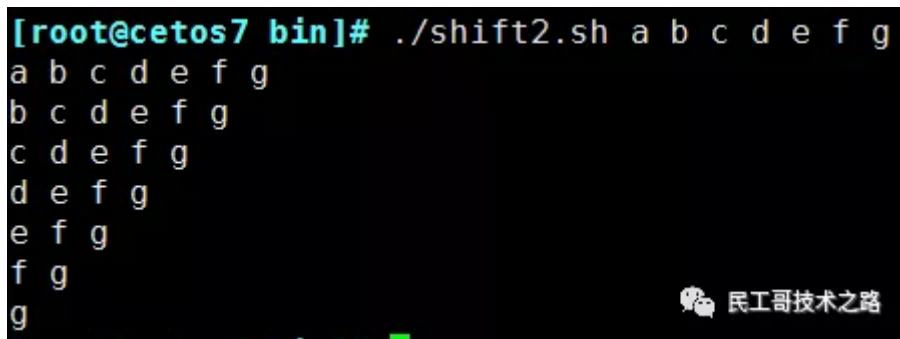
```
#!/binbash
if [ $# -eq 0 ] ;then
    echo "Please input a arg(eg:`basename $0` arg1)"
    exit 1
else
    while [ -n "$1" ];do
        useradd $1 && /dev/null
```

```
useradd $1 && /dev/null  
shift  
done  
fi
```

**分析：**如果没有输入参数（参数的总数为0），提示错误并退出；反之，进入循环；若第一个参数不为空字符，则创建以第一个参数为名的用户，并移除第一个参数，将紧跟的参数左移作为第一个参数，直到没有第一个参数，退出。

*#②打印直角三角形的字符*

```
#!/binbash  
while (( $# > 0 ))  
do  
    echo "$*"   
    shift  
done
```



```
[root@cetos7 bin]# ./shift2.sh a b c d e f g  
a b c d e f g  
b c d e f g  
c d e f g  
d e f g  
e f g  
f g  
g
```

The screenshot shows a terminal window with the command prompt [root@cetos7 bin]#. The user enters ./shift2.sh a b c d e f g. The script outputs a right-angled triangle of characters, with each line containing one fewer character than the previous line, starting from 'a b c d e f g' down to 'g'. A watermark '民工哥技术之路' is visible in the bottom right corner of the terminal output.

### 3、返回值结果

`true` 永远返回成功结果

`:` `null command` , 什么都不干, 返回成功结果  
`false` 永远返回错误结果

## 创建无限循环

```
while true ;do  
    循环体  
done
```

## 4、循环中可并行执行，使脚本运行更快

### (1) 用法

```
for name in 列表 ;do  
    {  
        循环体  
    }&  
done  
wait
```

### (2) 实例：

```
#搜寻自己指定ip（子网掩码为24的）的网段中，UP的ip地址  
read -p "Please input network (eg:192.168.0.0): " net  
echo $net | egrep -o "\<((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]))"
```

```
[ $? -eq 0 ] || ( echo "input error";exit 10 )
IP=`echo $net |egrep -o "^[0-9]{1,3}\.){3}"`
for i in {1..254};do
{
ping -c 1 -w 1 $IP$i &> /dev/null && \
echo "$IP$i is up"
}&

done
wait
```

**分析：** 请输入一个IP地址例192.168.37.234，如果格式不是0.0.0.0 则报错退出；正确则进入循环，IP变量的值为192.168.37.i的范围为1-254，并行ping 192.168.37.1-154，ping通就输出此IP为UP。直到循环结束。

## 四、信号捕获trap

### 1、用法格式

```
trap ' 触发指令' 信号，自定义进程收到系统发出的指定信号后，将执行触发指令，而不会执行原操作
trap '' 信号，忽略信号的操作
trap '-' 信号，恢复原信号的操作
trap -p，列出自定义信号操作
```

信号可以3种表达方法：信号的数字2、全名SIGINT、缩写INT

### 2、常用信号

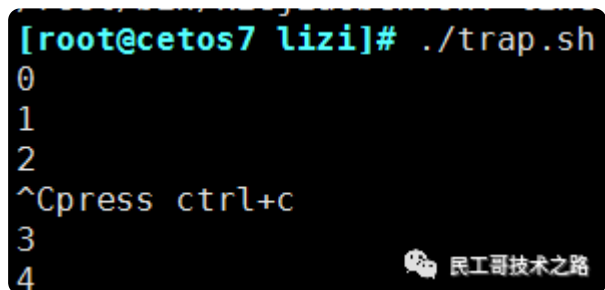


- 1) SIGHUP: 无须关闭进程而让其重读配置文件
- 2) SIGINT: 中止正在运行的进程; 相当于Ctrl+c
- 3) SIGQUIT: 相当于ctrl+\
- 9) SIGKILL: 强制杀死正在运行的进程
- 15) SIGTERM : 终止正在运行的进程 (默认为15)
- 18) SIGCONT : 继续运行
- 19) SIGSTOP : 后台休眠
- 9 信号, 强制杀死, 捕获不住

### 3、案例

```
#①打印0-9, ctrl+c不能终止
#!/bin/bash
trap 'echo press ctrl+c' 2
for ((i=0;i<10;i++));do
    sleep 1
    echo $i
done
```

**分析:** i=0, 当i<10, 每休眠1秒, i+1, 捕获2信号, 并执行echo press ctrl+c



```
[root@cetos7 lizi]# ./trap.sh
0
1
2
^Cpress ctrl+c
3
4
```

#②打印0-2, ctrl+c不能终止 2→F18有 能终止

*#C/C++编译，编译+运行能终止，运行后恢复，能终止*

```
#!/bin/bash
trap '' 2
trap -p
for ((i=0;i<3;i++));do
    sleep 1
    echo $i
done
trap '-' SIGINT
for ((i=3;i<10;i++));do
    sleep 1
    echo $i
done
```

**分析：** i=0，当i<3，每休眠1秒，i+1，捕获2信号；i>3时，解除捕获2信号。

```
[root@cetos7 lizi]# ./trap.sh
trap -- '' SIGINT
0
1
^C2
3
4
^C
[root@cetos7 lizi]#
```

民工哥技术之路

## 五、脚本小知识

### 1、生成随机字符 cat /dev/urandom

*#生成8个随机大小写字母或数字*

```
cat /dev/urandom |tr -dc [:alnum:] |head -c 8
```

## 2、生成随机数 echo \$RANDOM

确定范围 `echo ${RANDOM%7}` 随机7个数 (0-6)

`echo ${${RANDOM%7}+31}` 随机7个数 (31-37)

## 3、echo打印颜色字

`echo -e "\033[31malong\033[0m"` 显示红色along

`echo -e "\033[1;31malong\033[0m"` 高亮显示红色along

`echo -e "\033[41malong\033[0m"` 显示背景色为红色的along

`echo -e "\033[31;5malong\033[0m"` 显示闪烁的红色along

`color=${${RANDOM%7}+31}`

`echo -ne "\033[1;${color};5m*\033[0m"` 显示闪烁的随机色along

# 六、分享几个有意思的小脚本

## 1、9x9乘法表

```
#!/bin/bash
```

```
for a in {1..9};do
```

```
    for b in `seq 1 $a`;do
```

```
        let c=$a*$b ;echo -e "${a}x${b}=${c}\t\t"
```

done

echo

done

```
[root@cetos7 zuoye]# ./9x963.sh
1x1=1
2x1=2   2x2=4
3x1=3   3x2=6   3x3=9
4x1=4   4x2=8   4x3=12  4x4=16
5x1=5   5x2=10  5x3=15  5x4=20  5x5=25
6x1=6   6x2=12  6x3=18  6x4=24  6x5=30  6x6=36
7x1=7   7x2=14  7x3=21  7x4=28  7x5=35  7x6=42  7x7=49
8x1=8   8x2=16  8x3=24  8x4=32  8x5=40  8x6=48  8x7=56  8x8=64  8x9=72
9x1=9   9x2=18  9x3=27  9x4=36  9x5=45  9x6=54  9x7=63  9x8=72  9x9=81
```

## 2、彩色等腰三角形

```
#!/bin/bash
read -p "Please input a num: " num
if [[ $num =~ [^0-9] ]];then
    echo "input error"
else
    for i in `seq 1 $num`;do
        xing=${2*$i-1}
        for j in `seq 1 ${num-$i}`;do
            echo -ne " "
        done
        for k in `seq 1 $xing`;do
```

```

        color=${${RANDOM%7}+31}
        echo -ne "\033[1;${color};5m*\033[0m"

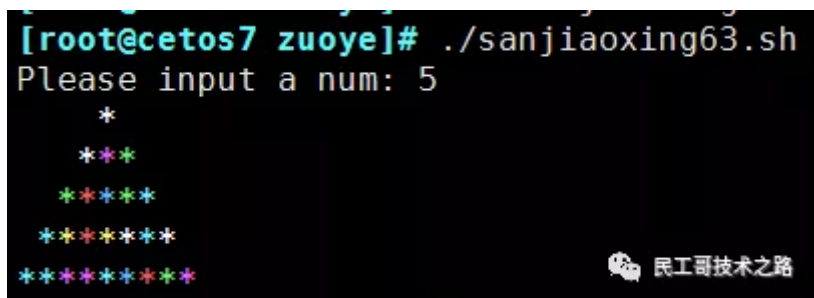
    done

    echo

done

fi

```



### 3、国际象棋棋盘

```

#!/bin/bash
red="\033[1;41m  \033[0m"
yellow="\033[1;43m  \033[0m"

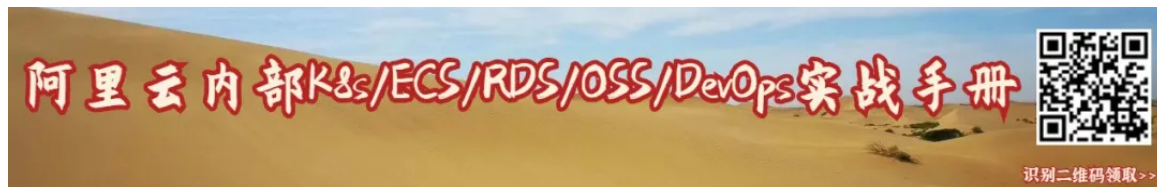
for i in {1..8};do
    if [ ${i%2} -eq 0 ];then
        for i in {1..4};do
            echo -e -n "$red$yellow";
        done
        echo
    else
        for i in {1..4};do

```

```
        echo -e -n "$yellow$red";  
    done  
    echo  
fi  
done
```



来源: <https://www.cnblogs.com/along21/p/7519710.html>





推荐阅读    点击标题可跳转

[全球当下最厉害的 14 位程序员，说没听过简直离谱~](#)

[别瞎学了，这几门语言要被淘汰了！](#)

[CentOS搭建VPN服务，一次性成功，收藏了](#)

[配置 Linux 的时钟同步](#)

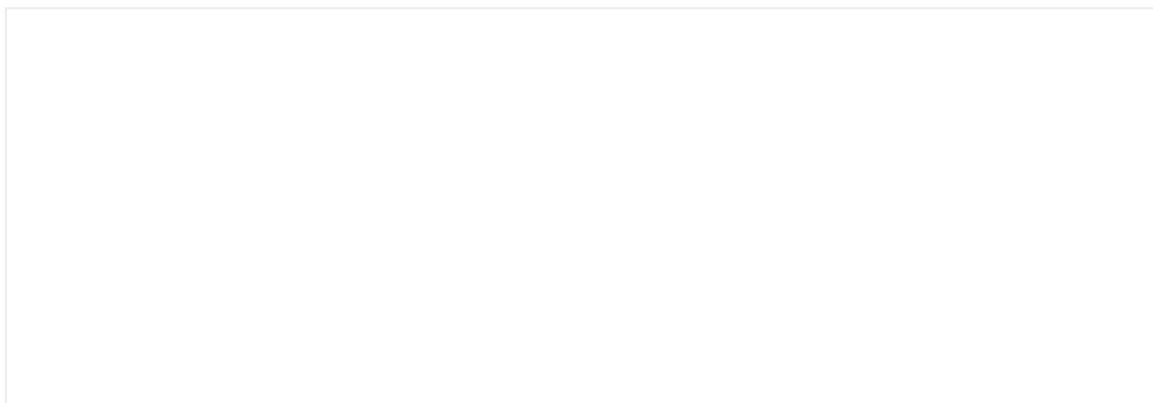
[每天学一个 Linux 命令（33）：uniq](#)

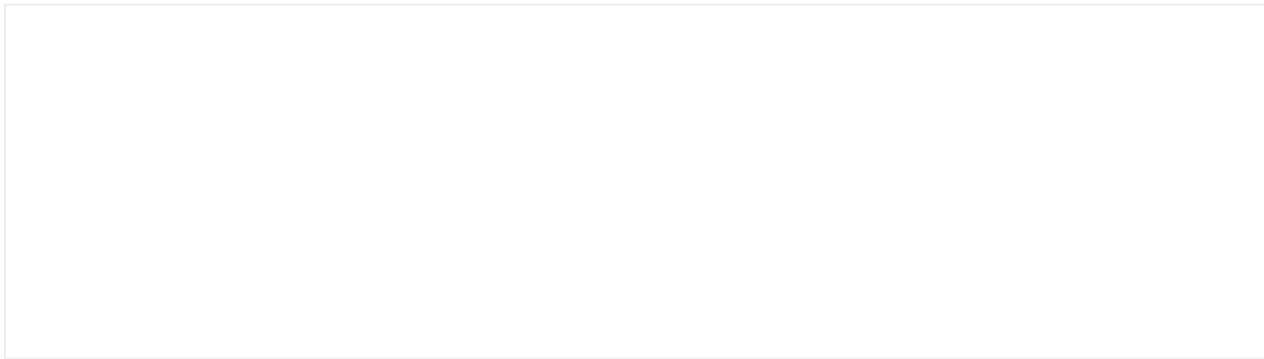
[收藏！17 张程序员专属壁纸（使用频率很高）](#)

[只知道HDFS和GFS？你其实并不懂分布式文件系统](#)

[Kubernetes生产环境最佳实践](#)

[6 个JVM性能监控、调优工具使用详解](#)





Read more

喜欢此内容的人还喜欢

再见 VBA！神器工具统一 Excel 和 Python

程序员的那些事

---

学习嵌入式Linux，做底层还是应用？底层要掌握哪些技能？

技术让梦想更伟大

---

图解NumPy，这是理解数组最形象的一份教程了

Python猫