

函数是什么



shell函数是什么?

先看linux的别名

```
1 [root@chaogelinux ~]# alias
2 alias cls='clear'
3 alias cp='cp -i'
4 alias egrep='egrep --color=auto'
5 alias fgrep='fgrep --color=auto'
6 alias grep='grep --color=auto'
7 alias l.='ls -d .* --color=auto'
8 alias ll='ls -l --color=auto'
9 alias ls='ls --color=auto'
10 alias mv='mv -i'
11 alias rm='rm -i'
12 alias which='alias | /usr/bin/which --tty-only --read-alias --
    show-dot --show-tilde'
13
```

别名的功能是简化命令操作，这个大家都知道，让命令更易读，易用。

函数，就是将shell脚本里需要多次被调用的相同代码，组合起来，称之为函数体。

并且我们要给这个函数体，起个别名，就叫做，函数名。

以后想要用这个函数体时，使用这个函数名，就可以了。

使用函数好处

- 相同的程序定义为函数，减少整个程序的代码数量，提高开发效率，让你少写点代码，有时间早点下班回家休息
- 增加程序可读性，易读性，容易管理

定义函数

通过function系统关键字定义函数

```
1 # 标准shell语法
2 function 函数名(){
3     代码。。。
4     return 返回值
5 }
6
7 # 简写，省去括号，不建议这么用
8
9 function 函数名{
10     代码。。。
11     return 返回值
12 }
13
14
15 # 更简化的写法，
16 函数名(){
```

```
17  代码。。。
18  return 返回值
19  }
20
21
```

执行函数

有关函数执行的基础概念

- 执行shell函数，直接写函数名字即可，无需添加其他内容
- 函数必须先定义，再执行，shell脚本自上而下加载
- 函数体内定义的变量，称之为局部变量
- 函数体内需要添加return语句，作用是退出函数，且赋予返回值给调用该函数的程序，也就是shell脚本
- return语句和exit不同
 - return是结束函数的执行，返回一个（退出值、返回值）
 - exit是结束shell环境，返回一个（退出值、返回值）给当前的shell
- 函数如果单独写入一个文件里，需要用source读取
- 函数内，使用local关键字，定义局部变量。

函数基础实践

先定义函数、再执行函数

场景1.

函数体，和调用函数，写在同一个脚本

```
1 [root@chaogelinux shell_program]# cat func1.sh
2 #!/bin/bash
3
4 pyyu(){
5     echo "超哥你好，你的linux讲的蛮有意思的"
6 }
7
8
9 function chaochao(){
10     echo "努力学好shell编程自动化，奥力给"
11 }
12
13 # 执行函数
14 pyyu
15 chaochao
```

执行

```
1 [root@chaogelinux shell_program]# bash func1.sh
2 超哥你好，你的linux讲的蛮有意思的
3 努力学好shell编程自动化，奥力给
```

场景2:

函数定义，执行不在一个脚本，更为专业的操作

从文件中读取函数，加载

```
1 1.自定义函数，函数体写入文件中
2 [root@chaogelinux shell_program]# cat my_func.sh
3 #!/bin/bash
4
5 pyyu(){
```

```
6     echo "Hello~pyyu"
7 }
8
9 2.开发一个脚本，调用该函数
10 [root@chaogelinux shell_program]# cat func2.sh
11 #!/bin/bash
12 # 判断该文件是否存在，在则加载，否则退出
13 [ -f /shell_program/my_func.sh ] && .
    /shell_program/my_func.sh || exit
14
15 # 读取该文件后，该文件中定义的函数，会被加载到当前shell环境
16 # 可以执行自定义的函数
17 pyyu
18
19 [root@chaogelinux shell_program]#
20 [root@chaogelinux shell_program]#
21 [root@chaogelinux shell_program]# bash func2.sh
22 Hello~pyyu
23
24
```

加载该函数到当前的shell环境

```
1 [root@chaogelinux shell_program]# pyyu
2 -bash: pyyu: 未找到命令
3 [root@chaogelinux shell_program]# source
  /shell_program/my_func.sh
4 [root@chaogelinux shell_program]# pyyu
5 Hello~pyyu
6 [root@chaogelinux shell_program]# set |grep pyyu
7 _=pyyu
8 pyyu ()
9     echo "Hello~pyyu"
```

给脚本传入参数

修改自定义函数的文件

```
1 [root@chaogelinux shell_program]# cat my_func.sh
2 #!/bin/bash
3
4 pyyu(){
5     echo "Hello~pyyu"
6 }
7
8
9 helloPyyu(){
10     echo "你给脚本传入的参数依次是：$1 、$2、$3、参数个数一共：$#"
11 }
```

开发执行函数的脚本func2.sh

```
1 [root@chaogelinux shell_program]# cat func2.sh
2 #!/bin/bash
3 # 判断该文件是否存在，在则加载，否则退出
4 [ -f /shell_program/my_func.sh ] && .
  /shell_program/my_func.sh || exit
5
6 # 读取该文件后，该文件中定义的函数，会被加载到当前shell环境
7 # 可以执行自定义的函数
8 pyyu
9
10 # 执行第二个函数，且给函数传入参数
11 helloPyyu $1 $2 $3
```

执行脚本

```
1 [root@chaogelinux shell_program]# bash func2.sh yu chao heihei
2 Hello~pyyu
3 你给脚本传入的参数依次是：yu 、chao、heihei、参数个数一共： 3
```

图解

修改自定义函数的文件

```
[root@chaogelinux shell_program]# cat my_func.sh
#!/bin/bash

pyyu(){
    echo "Hello~pyyu"
}

helloPyyu(){
    echo "你给脚本传入的参数依次是: $1 、$2、$3、参数个数一共: $#"
```

开发执行函数的脚本func2.sh

```
[root@chaogelinux shell_program]# cat func2.sh
#!/bin/bash
# 判断该文件是否存在，在则加载，否则退出
[ -f /shell_program/my_func.sh ] && . /shell_program/my_func.sh || exit

# 读取该文件后，该文件中定义的函数，会被加载到当前shell环境
# 可以执行自定义的函数
pyyu

# 执行第二个函数，且给函数传入参数
helloPyyu $1 $2 $3
```

执行脚本

```
[root@chaogelinux shell_program]# bash func2.sh yu chao heihei
Hello~pyyu
你给脚本传入的参数依次是: yu 、chao、heihei、参数个数一共: 3
```

开发URL检测脚本

功能：给脚本传入参数，检测url是否正常

```
1  #!/bin/bash
2  if [ "$#" -ne 1 ]
3      then
4          echo "Usage: $0 url"
5          exit 1
6  fi
7
8  # 利用wget命令测试访问
9  wget --spider -q -o /dev/null --tries=1 -T 5 $1
10
```



```
11 if [ "$?" -eq 0 ]
12     then
13         echo "$1 is yes."
14 else
15     echo "$1 is no."
16 fi
17
18
```

现在超哥希望大家能给该脚本，改造为函数执行

思考，函数是干什么的？封装功能的

函数版本shell脚本

check_url_func3.sh

```
1  #!/bin/bash
2
3  # 帮助函数
4  function usage(){
5      echo "Usage: $0 url"
6      exit 1
7  }
8
9  # 检测url的函数
10 check_url_func(){
11
12     # 利用wget命令测试访问
13     wget --spider -q -o /dev/null --tries=1 -T 5 $1
14
15     if [ "$?" -eq 0 ]
16     then
17         echo "$1 is yes."
```

```

18     else
19         echo "$1 is no."
20     fi
21 }
22
23 # 程序开发的习惯，设置一个入口函数，对需要执行的函数统一管理
24 # 我们可以在脚本中定义很多函数，到底执行哪一个，进行管理
25 main(){
26     # 如果用户传入的参数数量不等于1，表示用法有误
27     if [ $# -ne 1 ]
28     then
29         usage # 执行该帮助函数
30     fi
31     # 否则就执行正确的函数，以及传入的参数
32     check_url_func $1
33 }
34
35 # 这里的特殊变量$*，是将所有传入进来的参数，当作一个整体，是常见用法
36 main $*

```

执行脚本

```

1 # 正确执行
2 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn
3 www.pythonav.cn is yes.
4
5 # 参数传入个数不正确的情况
6 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn 123
7 Usage: check_url_func3.sh url
8 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn 123 qweqwe

```

```
9 Usage: check_url_func3.sh url
10
11 # 演示网站挂了
12 [root@chaogelinux shell_program]# nginx -s stop
13 [root@chaogelinux shell_program]#
14 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn
15 www.pythonav.cn is no.
16 [root@chaogelinux shell_program]# nginx
17 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn
18 www.pythonav.cn is yes.
```

添加功能，让结果显示的更专业

```
1 #!/bin/bash
2 # Use LSB init script functions for printing messages, if
  possible
3 #
4 lsb_functions="/lib/lsb/init-functions"
5 if test -f $lsb_functions ; then
6     . $lsb_functions
7 else
8     # Include non-LSB RedHat init functions to make systemctl
  redirect work
9     init_functions="/etc/init.d/functions"
10    if test -f $init_functions; then
11        . $init_functions
12    fi
13    log_success_msg()
14    {
15        echo " SUCCESS! $@"
16    }
```

```
17     log_failure_msg()
18     {
19         echo " ERROR! @$"
20     }
21 fi
22
23
24
25
26 # 帮助函数
27 function usage(){
28     echo "Usage: $0 url"
29     exit 1
30 }
31
32 # 检测url的函数
33 check_url_func(){
34
35     # 利用wget命令测试访问
36     wget --spider -q -o /dev/null --tries=1 -T 5 $1
37
38     if [ "$?" -eq 0 ]
39     then
40         log_success_msg "$1 is yes."
41     else
42         log_failure_msg "$1 is no."
43     fi
44 }
45
46 # 程序开发的习惯，设置一个入口函数，对需要执行的函数统一管理
47 # 我们可以在脚本中定义很多函数，到底执行哪一个，进行管理
48 main(){
49     # 如果用户传入的参数数量不等于1，表示用法有误
50     if [ $# -ne 1 ]
```

```

51     then
52         usage # 执行该帮助函数
53     fi
54     # 否则就执行正确的函数，以及传入的参数
55     check_url_func $1
56 }
57
58 # 这里的特殊变量$*, 是将所有传入进来的参数，当作一个整体，是常见用法
59 main $*

```

执行结果

```

1 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cn
2 www.pythonav.cn is yes. [
  确定 ]
3
4 [root@chaogelinux shell_program]# bash check_url_func3.sh
  www.pythonav.cnw
5 www.pythonav.cnw is no. [失
  败]

```

```

[root@chaogelinux shell_program]# bash check_url_func3.sh www.pythonav.cn
www.pythonav.cn is yes. [ 确定 ]
[root@chaogelinux shell_program]# bash check_url_func3.sh www.pythonav.cnw
www.pythonav.cnw is no. [失败]

```

开发rsync起停脚本

当然超哥这里讲的是函数版本

```

1 [root@chaogelinux shell_program]# touch /etc/init.d/cc_rsyncd

```

```
2
3 #!/bin/bash
4 # Use LSB init script functions for printing messages, if
  possible
5 #
6 lsb_functions="/lib/lsb/init-functions"
7 if test -f $lsb_functions ; then
8     . $lsb_functions
9 else
10     # Include non-LSB RedHat init functions to make systemctl
    redirect work
11     init_functions="/etc/init.d/functions"
12     if test -f $init_functions; then
13         . $init_functions
14     fi
15     log_success_msg()
16     {
17         echo " SUCCESS! $@"
18     }
19     log_failure_msg()
20     {
21         echo " ERROR! $@"
22     }
23 fi
24
25 function usage(){
26     echo "Usage: $0 {start|stop|restart}"
27     exit 1
28 }
29
30 function start(){
31     /usr/bin/rsync --daemon
32     sleep 1
33     if [ `netstat -tunlp|grep rsync|wc -l` -ge "1" ]
```

```
34     then
35         log_success_msg "rsyncd is started."
36     else
37         log_failure_msg "Rsyncd isn't started. "
38     fi
39 }
40
41 function stop(){
42     killall rsync &>/dev/null
43     sleep 2
44     if [ `netstat -tunlp|grep rsync|wc -l` -eq "0" ]
45     then
46         log_success_msg "Rsyncd is stopped"
47     else
48         log_failure_msg "Rsyncd isn't stopped. "
49     fi
50 }
51
52 function main(){
53     if [ "$#" -ne "1" ]
54     then
55         usage
56     fi
57
58     if [ "$1" = "start" ]
59     then
60         start
61     elif [ "$1" = "stop" ]
62     then
63         stop
64     elif [ "$1" = "restart" ]
65     then
66         stop
67         sleep 1
```

```
68     start
69 else
70     usage
71 fi
72 }
73
74 # 程序入口
75 main $*
76
```

执行脚本

```
1 [root@chaogelinux shell_program]# /etc/init.d/cc_rsyncd stop
2 Rsyncd is stopped [
 确定 ]
3 [root@chaogelinux shell_program]# /etc/init.d/cc_rsyncd start
4 rsyncd is started. [
 确定 ]
5 [root@chaogelinux shell_program]# /etc/init.d/cc_rsyncd restart
6 Rsyncd is stopped [
 确定 ]
7 rsyncd is started. [
 确定 ]
```

希望大家学习超哥开发脚本的习惯，实现高度的模块化，对功能封装，这样观看美观，且专业规范。

