## 介绍

在脚本开发中将常用信息规范化，统一使用标准模板，很大程度上降低重复code开发，提高开发效率，减少维护脚本成本.

## 基本表规范

1. 函数定义完成,统一放置脚本前段,避免定义函数和执行其他流程混杂在一起,显得臃肿.
2. 返回码信息要统一规范书写，同样的异常描述信息使用一致描述.
3. 脚本命名根据功能要求命名，eg：redis\_set\_monitor进行大value监控设置，可命名:set\_bigValue\_monitor,仅是参照具体命名视功能而定.
4. 附上代码注释,方便快速明白具体实现内容.
5. 检查服务情况,可统一使用标准检查项封装成的函数用于检查:服务进程,检查用户,检查端口号,检查进程所属用户组.
6. 细化执行失败场景,避免直接输出执行$? ≠0 信息,造成模糊界定,使异常返回码对应输出信息不精确.

## 一、附属信息

1. #!/bin/bash下空一行，如下

#!/bin/bash

#

1. 附上版本信息包括新增或优化迭代

eg：Version:1.0.0

小版本迭代请升到Version:1.0.1

大版本迭代请升到Version:1.1.0

版本优化项较多请升到Version:2.0.0

1. 附上脚本功能描述信息，可以中文或英文，语言描述需要言简意赅，避免繁杂无法突出重点.

信息包括：脚本作用，需要传入的参数，及注意事项或备注信息，其他辅助信息亦可。

1. 附上脚本执行方法

sh xxx.sh -h [参数1] -p [参数2] -x [参数。。。]

1. 附上脚本作者和创建时间,适用用初次创建的脚本

eg：Author:Zhoushaohua Version:1.0.0 Date:2018/5/10 15:22:24

1. 附上脚本优化信息包括如下:版本、时间、开发人员、优化内容等信息

eg：

# History:

# <version> <time> <author> <desc>

# 1.1.0 18/10/26 zhoush 启动前增加机器内存校验

# 1.1.1 18/12/25 zhoush 启动前修改client-output-buffer-limit

# 1.1.2 19/05/30 zhoush 增加/opt/redisinfo是否为空文件判断

# 1.1.3 19/06/21 zhoush /tmp/.redisinfo -> /opt/redisinfo

1. 附上 return code返回码信息，根据报错时返回码快速查找功能，定位报错异常的位置。
   1. 注意返回码依次从小到大顺序使用
   2. 返回码1不可使用，是由于acm系统的调用脚本时返回码占用，具体可以见acm使用说明

###################################################################

# return code

# 0 OK

# 2 redisuser doesn't exist

# 3 get redis's info is empty from /opt/redisinfo

# 4 redis file doesn't exist

# 5 starting Redis error

# 6 master-slave synchronization timeout

# 7 maxmemory modify failed

# 8 parameter error

# 9 target machine's memory abnormal

# 10 master\_link\_status is NULL

# 11 modify client-output-buffer-limit slave failed

# 12 /opt/redisinfo file not found or empty

# 13 kill -9 redis-server error

# 14 user of the redis-server is not redisuser

# 15 redis.conf not found or empty

# 16 /proc/${redis\_pid}/oom\_score\_adj set -1000 failed

###################################################################

附属信息案例

#!/bin/bash

#

# Version:1.1.7

# Function:start Redis server by /opt/redisinfo（redis 2.8.X） in memory modification

# Use:sh start\_redis.sh -m 8

# Author:Zhoushaohua Version:1.0.0 Date:2018/5/10 15:22:24

#

# History:

# <version> <time> <author> <desc>

# 1.1.0 18/10/26 zhoush 启动前增加机器内存校验

# 1.1.1 18/12/25 zhoush 启动前修改client-output-buffer-limit

# 1.1.2 19/05/30 zhoush 增加/opt/redisinfo是否为空文件判断

###################################################################

# return code

# 0 OK

# 2 redisuser doesn't exist

# 3 get redis's info is empty from /opt/redisinfo

# 4 redis file doesn't exist

# 5 starting Redis error

# 6 master-slave synchronization timeout

# 7 maxmemory modify failed

# 8 parameter error

# 9 target machine's memory abnormal

# 10 master\_link\_status is NULL

# 11 modify client-output-buffer-limit slave failed

# 12 /opt/redisinfo file not found or empty

# 13 kill -9 redis-server error

# 14 user of the redis-server is not redisuser

# 15 redis.conf not found or empty

# 17 /proc/${redis\_pid}/oom\_score\_adj set -1000 failed

###################################################################

## 脚本入参

针对脚本执行时需要的入参项，常用参数使用一致的入参项，避免同样参数不同脚本入参项不同，造成混乱。

* 1. Host 使用 -h host\_ip #IP地址
  2. Port 使用 -p host\_port #端口号 redis默认6379/sentinel默认26379
  3. Shard 使用 -n shard\_name #实例名 SNRS\_SIT\_1
  4. Version 使用 -v zedis\_version #版本 Zedis 1.1.4
  5. Switch 使用 -l zedis\_limit switch #限流开/关 [on|off]
  6. 等等

function my\_usage()

{

cat << EOF

Usage: $0 [OPTIONS]

Valid options are:

-h sentinel ip

-p sentinel port

EOF

}

while getopts "h:p:H" opt; do

case $opt in

h) host=$OPTARG;;

p) port=$OPTARG;;

H) my\_usage;exit 0;;

\?)my\_usage;exit 1;;

esac

done

if [ -z "$host" -o -z "$port" ] ; then

my\_usage;

exit 2

fi

## 日志函数

统一使用日志函数，根据脚本名规范定义输出日志名称

function log()

{

echo `date "+%Y.%m.%d %k:%M:%S"` $@ >> $logfile

}

logfile="/tmp/\_set\_client\_output\_buffer.log"

如上所示:

1. 目前输出脚本的日志统一放在tmp目录下
2. 输出日志文件名是在脚本明前加上下划线\_统一按照此方法命名

\*注：后期日志存放路径如有更改，可视情况优化输出日志文件的路径

## 返回码

统一使用执行退出函数，显示每个返回码相应的错误信息

\*注：

1. 请不要使用exit 1的返回码，其他返回码使用依次从小到大按照顺序进行采用
2. 不要跳出顺序码，使用127,126等特殊码
3. 如有特殊需要强调的返回码按照特殊情况采用需要的返回码
4. 返回码的信息需要描述准确，简短，采用英文请使用翻译工具。

function quit()

{

case $1 in

0)

echo "OK" >&1

exit 0

;;

2)

echo "redis\_cli ping failed! $2" >&2

exit 2

;;

3)

echo "modify client-output-buffer-limit failed! $2" >&2

exit 3

;;

4)

echo "rewrite client-output-buffer-limit failed! $2" >&2

exit 4

;;

esac

}

## 服务检查

* 1. 检查redis-server进程数
  2. 检查redisuser用户
  3. 检查redisuser进程所属用户
  4. 检查端口，默认是6379

以上是redis-server服务的标准检查，sentinel-server可以参照上面安排统一使用函数，进行标准检查.

eg：

function standard\_check()

{

#检查用户redisuser

id $zedis\_user >> $logfile 2>&1

if [ $? -ne 0 ];then

log "---E--- redisuser doesn't exist ! the script stopped !"

quit 3

fi

#检查进程启动数目

redis\_num=$(ps -ef | egrep -w "[r]edis-server" | awk '{print $2}' | wc -l)

if [ $redis\_num -ge 2 ];then

log "---E--- there are $redis\_num redis server alive! the script stopped , please check!"

quit 3

elif [ $redis\_num -eq 0 ];then

log "---E--- redis server doesn't exist ! the script stopped !"

quit 3

fi

#检查进程所属用户

redis\_pid=$(ps -u $zedis\_user | egrep -w "[r]edis-server" | awk '{print $1}')

if [ -z "$redis\_pid" ];then

log "---E--- The user of the Redis server is not $zedis\_user ! the script stopped !"

quit 3

fi

#检查端口（6379）

server\_port=$(ps aux | egrep -w "[r]edis-server" | awk -F '\*:' '{print $NF}' | awk -F ' ' '{print$1}' |sed 's/ //g')

if [ $server\_port != "6379" ];then

log "---E--- The redis-server port is not standard! the script stopped !"

quit 3

fi

}

## [ ] 与[[ ]]

1. 使用-n 或者 -z 方式判断变量是否为空时[] 与 [[]]是有区别的,使用[]时需要变量加双引号””,使用[[ ]]则不需要.

[root@snrsdevapp13 ~]# echo $c

[root@snrsdevapp13 ~]# [ -n $c ]

[root@snrsdevapp13 ~]# echo $?

0

[root@snrsdevapp13 ~]# [ -n "$c" ]

[root@snrsdevapp13 ~]# echo $?

1

1. 使用[[]]时,不能使用 -a 或者 -o 对多个条件进行连接

[root@snrsdevapp13 ~]# [[ 5 -gt 1 ]] -o [[ 8 -gt 7 ]]

-bash: syntax error near unexpected token `-o'

[root@snrsdevapp13 ~]# [[ 5 -gt 1 -a -1 -lt 0 ]]

-bash: syntax error in conditional expression

-bash: syntax error near `-a'

1. 使用[]时，如果使用 -a 或者 -o 对多个条件进行连接，-a 或者 -o 必须包含在[ ]内

[root@snrsdevapp13 ~]# [ 5 -gt 1 ] -o [ 8 -gt 7 ]

-bash: [: too many arguments

[root@snrsdevapp13 ~]# [ 5 -gt 1 -o 8 -gt 7 ]

[root@snrsdevapp13 ~]# echo $?

0

1. 使用[]时，如果使用 && 或者 || 对多个条件进行连接，&& 或者 || 必须在[ ]之外

[root@snrsdevapp13 ~]# [ 5 -gt 1 && 8 -gt 7 ]

-bash: [: missing `]'

[root@snrsdevapp13 ~]# [ 5 -gt 1 ] && [ 8 -gt 7 ]

[root@snrsdevapp13 ~]# echo $?

0

1. 使用[[]]时，如果使用 && 或者 || 对多个条件进行连接，&& 或者 || 可放置在在[[ ] ]内侧和外侧

[root@snrsdevapp13 ~]# [[ 5 -gt 1 && -1 -lt 0 ]]

[root@snrsdevapp13 ~]# echo $?

0

[root@snrsdevapp13 ~]# [[ 5 -gt 1 && -1 -lt 0 ]] && [[ 8 -gt 7 ]]

[root@snrsdevapp13 ~]# echo $?

0