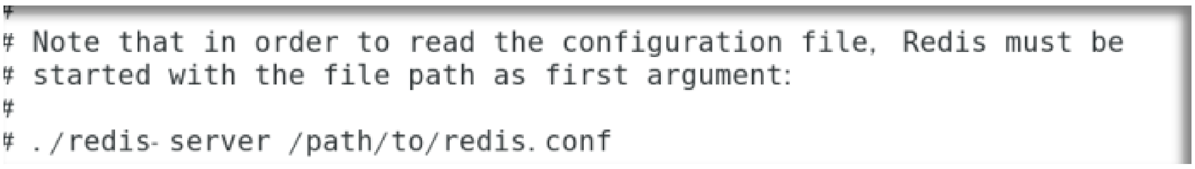
Redis配置文件详解

配置文件位置

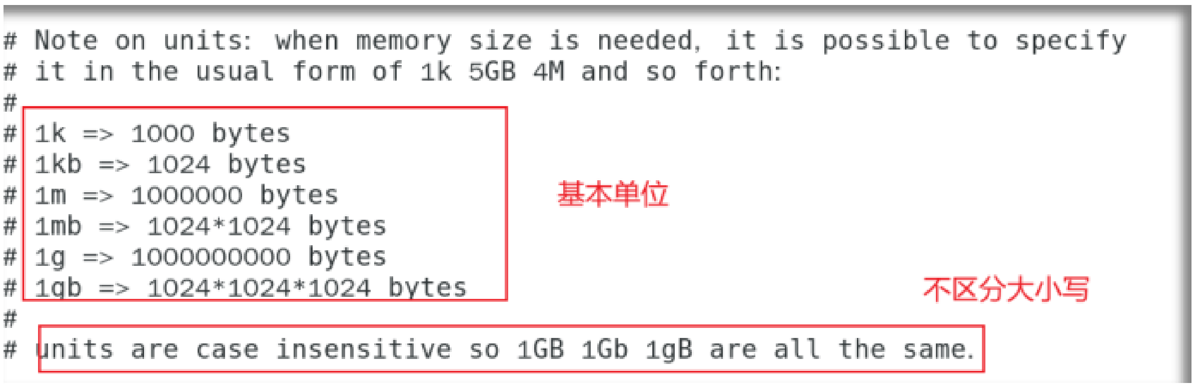
以配置文件启动

```
    Redis 的配置文件位于 Redis 安装目录下，文件名为 redis.conf(Windows 名为
redis.windows.conf)
```



```
#
# Note that in order to read the configuration file, Redis must be
# started with the file path as first argument:
#
# ./redis-server /path/to/redis.conf
```

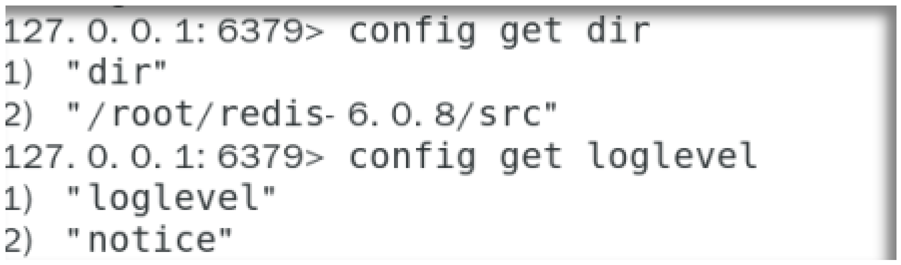
```
    Redis对配置文件对大小写不敏感
```



```
# Note on units: when memory size is needed, it is possible to specify
# it in the usual form of 1k 5GB 4M and so forth:
#
# 1k => 1000 bytes
# 1kb => 1024 bytes
# 1m => 1000000 bytes
# 1mb => 1024*1024 bytes
# 1g => 1000000000 bytes
# 1gb => 1024*1024*1024 bytes
#
# units are case insensitive so 1GB 1Gb 1gB are all the same.
```

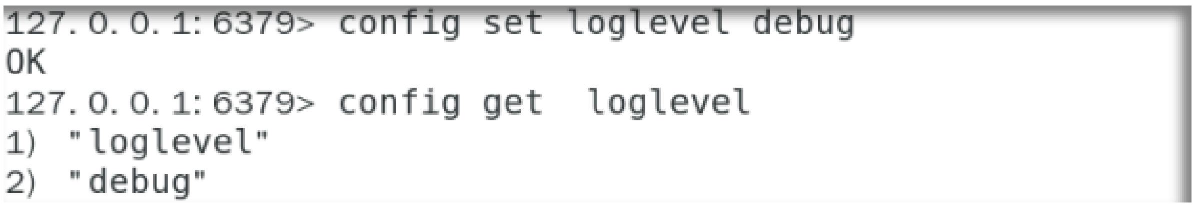
配置命令

```
    当前服务中修改配置
    获得配置内容命令  config get   config get *
```



```
127.0.0.1:6379> config get dir
1) "dir"
2) "/root/redis-6.0.8/src"
127.0.0.1:6379> config get loglevel
1) "loglevel"
2) "notice"
```

```
    设置配置命令：你可以通过修改 redis.conf 文件或使用 CONFIG set 命令来修改配置。
    Config set loglevel debug
```

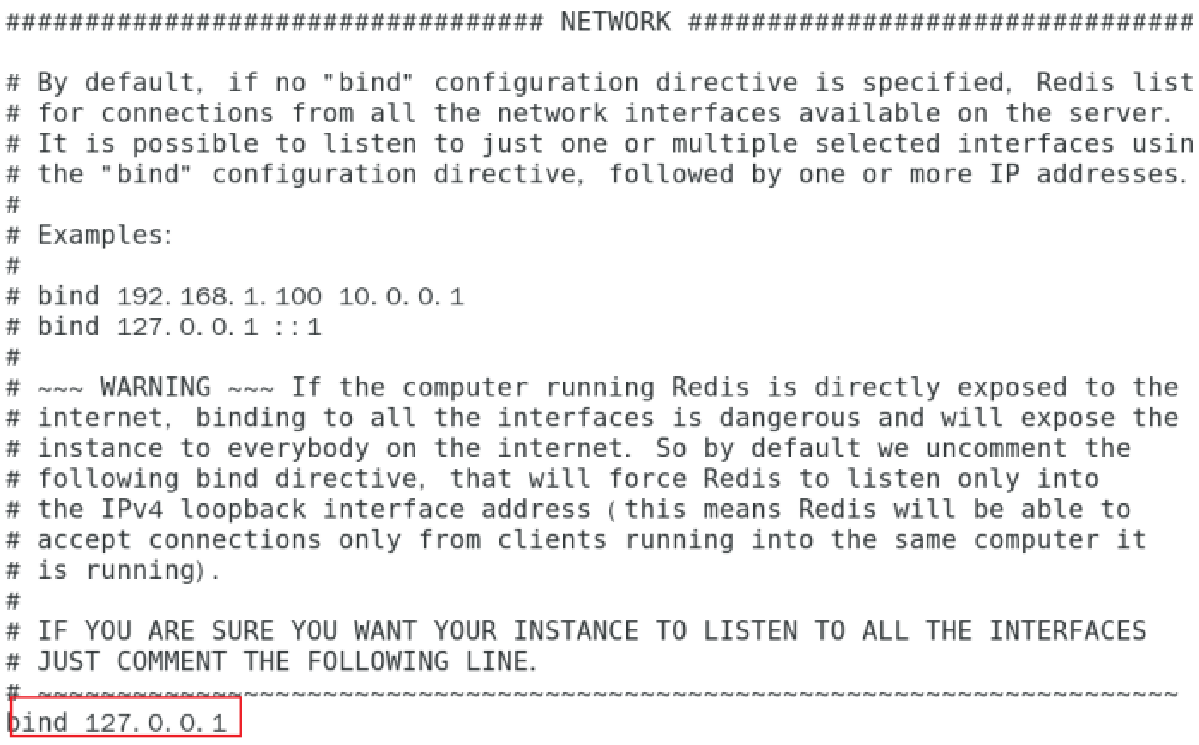


```
127.0.0.1:6379> config set loglevel debug
OK
127.0.0.1:6379> config get  loglevel
1) "loglevel"
2) "debug"
```

配置文件中和安全相关的配置

网络配置

绑定IP

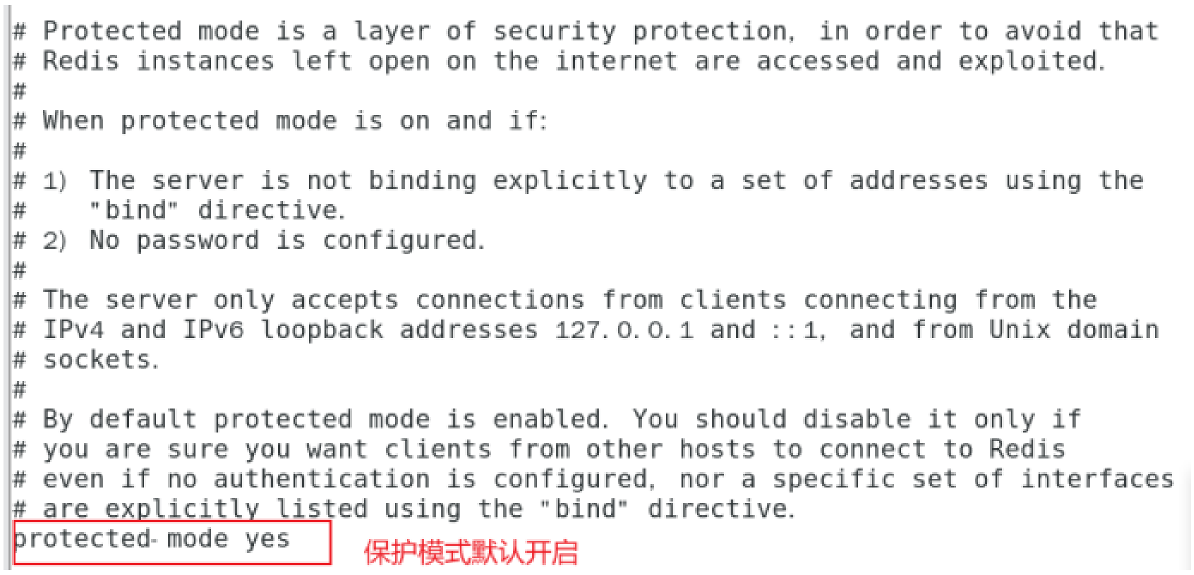


```
################################### NETWORK ##################################

# By default, if no "bind" configuration directive is specified, Redis list
# for connections from all the network interfaces available on the server.
# It is possible to listen to just one or multiple selected interfaces usin
# the "bind" configuration directive, followed by one or more IP addresses.
#
# Examples:
#
# bind 192.168.1.100 10.0.0.1
# bind 127.0.0.1 ::1
#
# ~~~ WARNING ~~~ If the computer running Redis is directly exposed to the
# internet, binding to all the interfaces is dangerous and will expose the
# instance to everybody on the internet. So by default we uncomment the
# following bind directive, that will force Redis to listen only into
# the IPv4 loopback interface address (this means Redis will be able to
# accept connections only from clients running into the same computer it
# is running).
#
# IF YOU ARE SURE YOU WANT YOUR INSTANCE TO LISTEN TO ALL THE INTERFACES
# JUST COMMENT THE FOLLOWING LINE.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
bind 127.0.0.1
```

保护模式

> 保护模式默认开启
> 保护模式为开启 如果是远程链接不能进行 CRUD 等操作，如果进行该操作报错如下

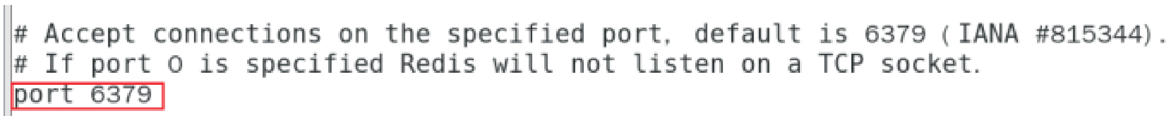
```
    (error) DENIED Redis is running in protected mode because protected mode is
enabled, no bind address was specified, no authentication password is requested
to clients. In this mode connections are only accepted from the loopback
interface. If you want to connect from external computers to Redis you may adopt
one of the following solutions: 1) Just disable protected mode sending the
command 'CONFIG SET protected-mode no' from the loopback interface by connecting
to Redis from the same host the server is running, however MAKE SURE Redis is not
publicly accessible from internet if you do so. Use CONFIG REWRITE to make this
change permanent. 2) Alternatively you can just disable the protected mode by
editing the Redis configuration file, and setting the protected mode option to
'no', and then restarting the server. 3) If you started the server manually just
for testing, restart it with the '--protected-mode no' option. 4) Setup a bind
address or an authentication password. NOTE: You only need to do one of the above
things in order for the server to start accepting connections from the outside.
```



```
# Protected mode is a layer of security protection, in order to avoid that
# Redis instances left open on the internet are accessed and exploited.
#
# When protected mode is on and if:
#
# 1) The server is not binding explicitly to a set of addresses using the
#    "bind" directive.
# 2) No password is configured.
#
# The server only accepts connections from clients connecting from the
# IPv4 and IPv6 loopback addresses 127.0.0.1 and ::1, and from Unix domain
# sockets.
#
# By default protected mode is enabled. You should disable it only if
# you are sure you want clients from other hosts to connect to Redis
# even if no authentication is configured, nor a specific set of interfaces
# are explicitly listed using the "bind" directive.
protected-mode yes
```

```
默认是 protected-mode yes，即开启保护模式，这里改为 no 关闭
```

设置端口

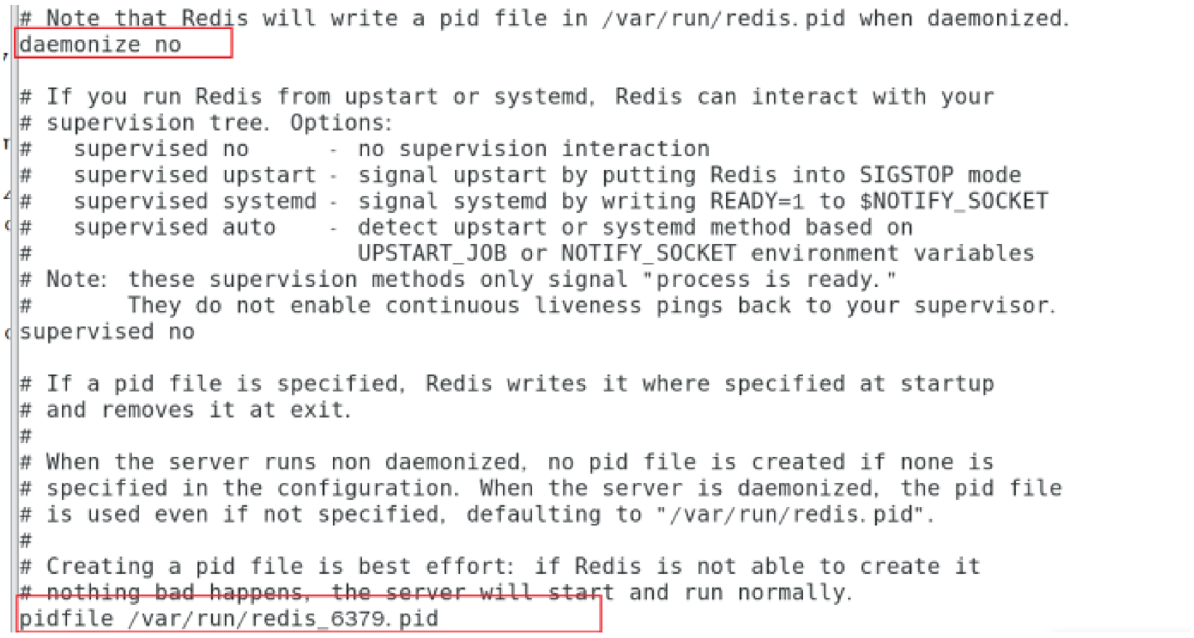


```
# Accept connections on the specified port, default is 6379 (IANA #815344).
# If port 0 is specified Redis will not listen on a TCP socket.
port 6379
```

通用设置

守护进程

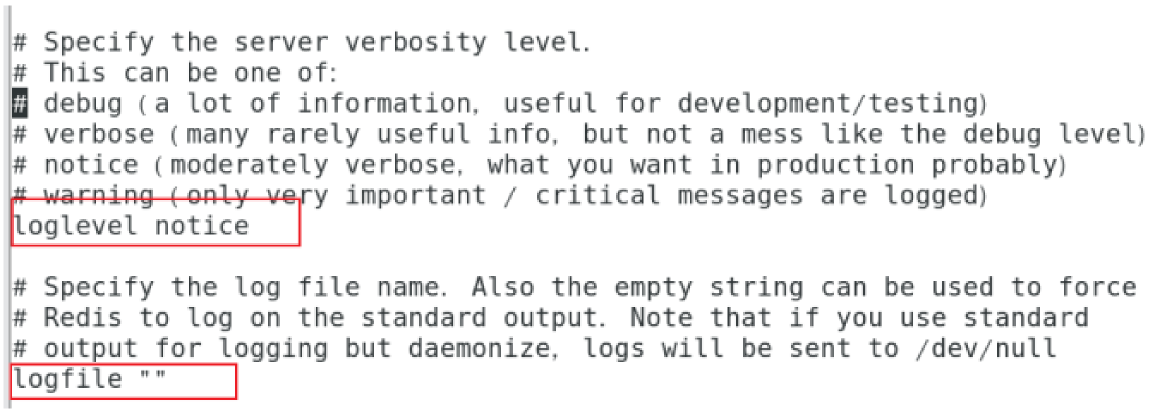
```
是否以守护进程方式运行 （默认是no）
如果以守护进程方式运行，需要指定一个PID进程文件
```



```
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize no

# If you run Redis from upstart or systemd, Redis can interact with your
# supervision tree. Options:
#   supervised no      - no supervision interaction
#   supervised upstart - signal upstart by putting Redis into SIGSTOP mode
#   supervised systemd - signal systemd by writing READY=1 to $NOTIFY_SOCKET
#   supervised auto    - detect upstart or systemd method based on
#                        UPSTART_JOB or NOTIFY_SOCKET environment variables
# Note: these supervision methods only signal "process is ready."
#       They do not enable continuous liveness pings back to your supervisor.
supervised no

# If a pid file is specified, Redis writes it where specified at startup
# and removes it at exit.
#
# When the server runs non daemonized, no pid file is created if none is
# specified in the configuration. When the server is daemonized, the pid file
# is used even if not specified, defaulting to "/var/run/redis.pid".
#
# Creating a pid file is best effort: if Redis is not able to create it
# nothing bad happens, the server will start and run normally.
pidfile /var/run/redis_6379.pid
```

loglevel

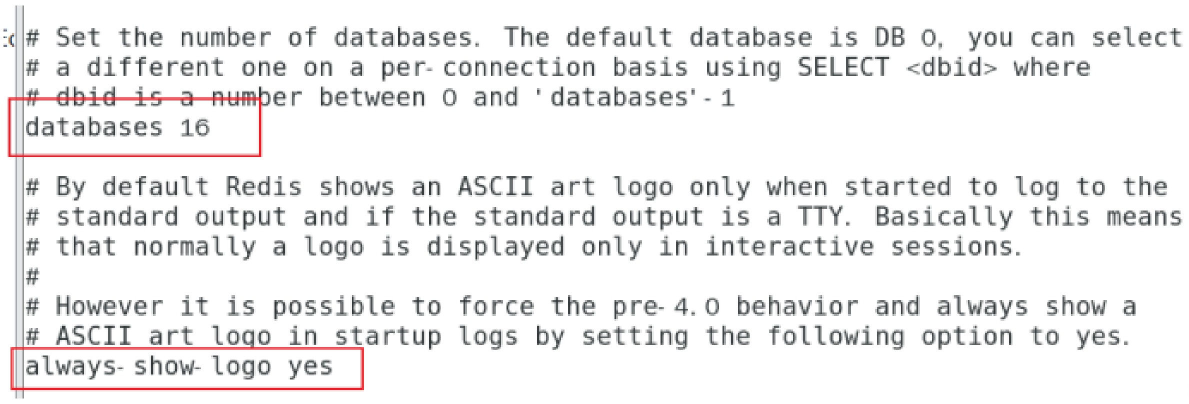
```
日志 debug测试环境使用 notice生产环境使用，日志生成的文件位置名，如果空只是控制台输出
```



```
# Specify the server verbosity level.
# This can be one of:
# debug (a lot of information, useful for development/testing)
# verbose (many rarely useful info, but not a mess like the debug level)
# notice (moderately verbose, what you want in production probably)
# warning (only very important / critical messages are logged)
loglevel notice

# Specify the log file name. Also the empty string can be used to force
# Redis to log on the standard output. Note that if you use standard
# output for logging but daemonize, logs will be sent to /dev/null
logfile ""
```

数据库数量

```
数据库数量和是否显示redis的logo
```

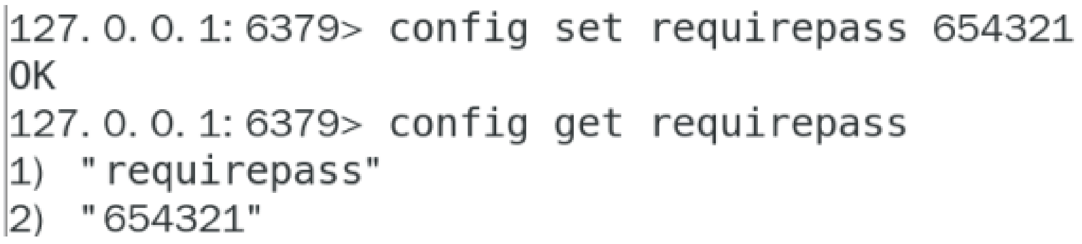


```
# Set the number of databases. The default database is DB 0, you can select
# a different one on a per-connection basis using SELECT <dbid> where
# dbid is a number between 0 and 'databases'-1
databases 16

# By default Redis shows an ASCII art logo only when started to log to the
# standard output and if the standard output is a TTY. Basically this means
# that normally a logo is displayed only in interactive sessions.
#
# However it is possible to force the pre-4.0 behavior and always show a
# ASCII art logo in startup logs by setting the following option to yes.
always-show-logo yes
```

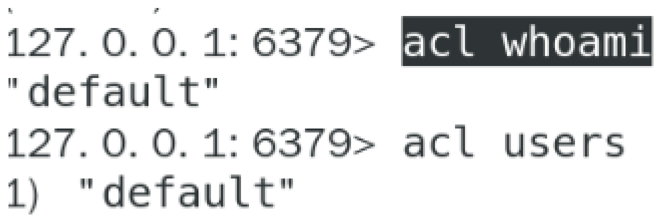
安全配置

连接密码

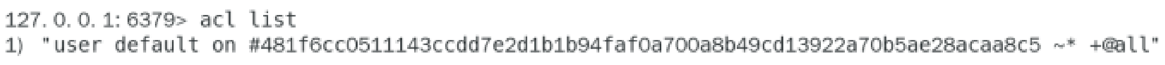
```
    连接密码，默认是不需要密码的，可以通过“requirepass 123456”设置连接密码为“123456”
    可以通过命令设置密码，但设置的密码在服务重新启动之后就失效啦  没有指定用户， 进入都是
defautl 用户  acl list 可以看所有权限
    Config set requirepass  123456
    Config get requirepass
    Auth 123456
```



```
127.0.0.1:6379> config set requirepass 654321
OK
127.0.0.1:6379> config get requirepass
1) "requirepass"
2) "654321"
```

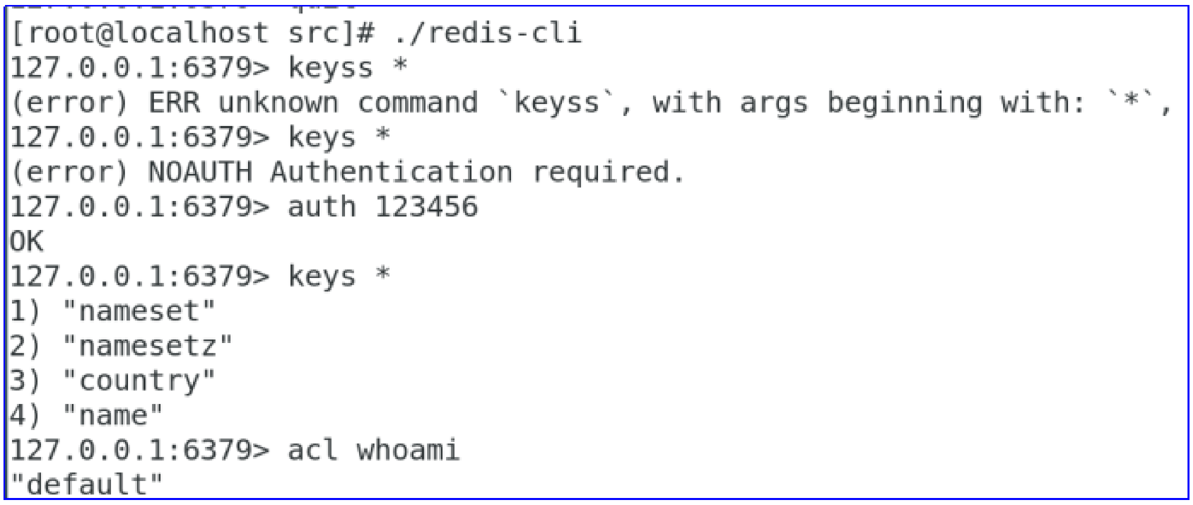


```
127.0.0.1:6379> acl whoami
"default"
127.0.0.1:6379> acl users
1) "default"
```



```
127.0.0.1:6379> acl list
1) "user default on #481f6cc0511143ccdd7e2d1b1b94faf0a700a8b49cd13922a70b5ae28acaa8c5 ~* +@all"
```

```
    永久配置redis的密码 通过更改配置文件来执行，注意修改配置文件之后，需要重新启动服务
    Shutdown   退出redis服务
    ./redis-server ../redis.conf
    设置完毕密码之后在退出redis-cli后再次登录需要验证密码
```



```
[root@localhost src]# ./redis-cli
127.0.0.1:6379> keyss *
(error) ERR unknown command `keyss`, with args beginning with: `*`,
127.0.0.1:6379> keys *
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth 123456
OK
127.0.0.1:6379> keys *
1) "nameset"
2) "namesetz"
3) "country"
4) "name"
127.0.0.1:6379> acl whoami
"default"
```

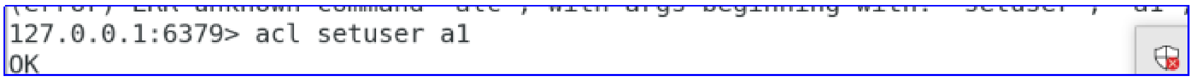
ACL简介

```
    在Redis6.0之前的版本中，登陆Redis Server只需要输入密码（前提配置了密码 requirepass ）即可，不需要输入用户名，而且密码也是明文配置到配置文件中，安全性不高。并且应用连接也使用该密码，导致应用有所有权限处理数据，风险也极高。在Redis6.0有了ACL之后，终于解决了这些不安全的因素，可以按照不同的需求设置相关的用户和权限
```

ACL命令

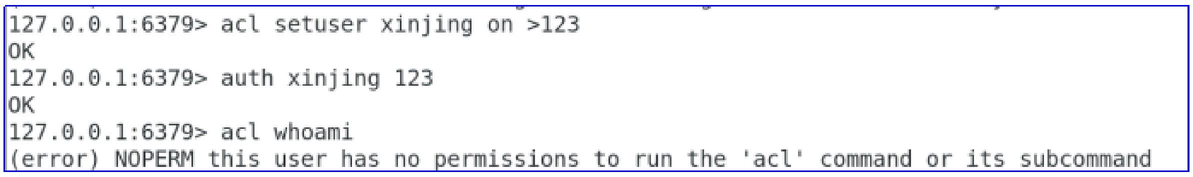
```
> ACL help
 1) ACL <subcommand> arg arg ... arg. Subcommands are:
 2) LOAD                             -- 从ACL文件中重新载入用户信息.
 3) SAVE                             -- 保存当前的用户配置信息到ACL文件.
 4) LIST                             -- 以配置文件格式显示用户详细信息.
 5) USERS                            -- 列出所有注册的用户名.
 6) SETUSER <username> [attribs ...] -- 创建或则修改一个用户.
 7) GETUSER <username>               -- 得到一个用户的详细信息.
 8) DELUSER <username> [...]         -- 删除列表中的用户.
 9) CAT                              -- 列出可用的类别.
10) CAT <category>                   -- 列出指定类别中的命令.
11) GENPASS [<bits>]                 -- 生成一个安全的用户密码.
12) WHOAMI                           -- 返回当前的连接用户.
13) LOG [<count> | RESET]            -- 显示ACL日志条目.
```

> 新建用户
> 不带密码



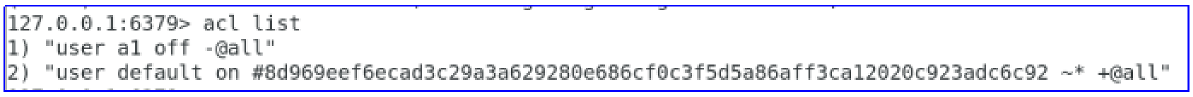
```
127.0.0.1:6379> acl setuser a1
OK
```

> 带密码的
> 并切换用户，但是没有任何权限访问被拒绝

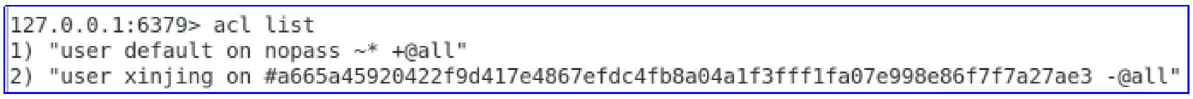


```
127.0.0.1:6379> acl setuser xinjing on >123
OK
127.0.0.1:6379> auth xinjing 123
OK
127.0.0.1:6379> acl whoami
(error) NOPERM this user has no permissions to run the 'acl' command or its subcommand
```

> 列出所有用户



```
127.0.0.1:6379> acl list
1) "user a1 off -@all"
2) "user default on #8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92 ~* +@all"
```

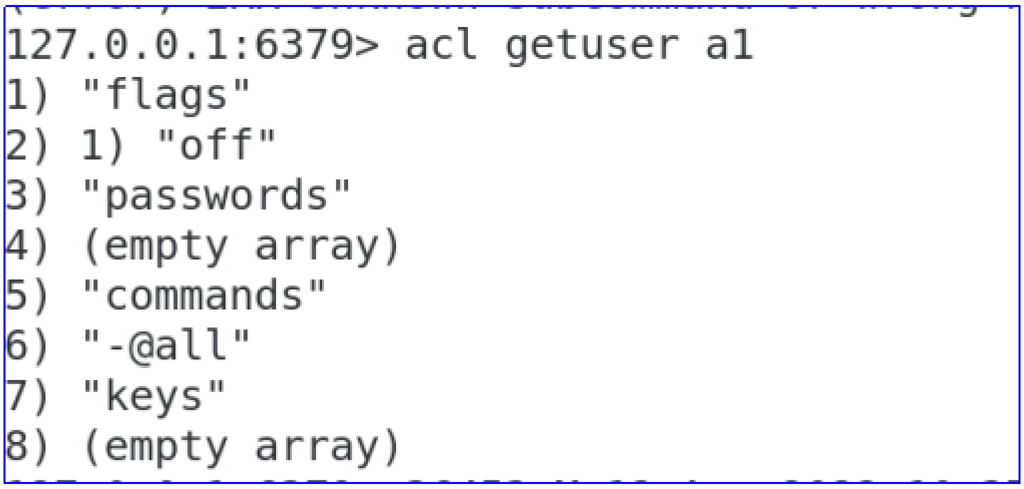


```
127.0.0.1:6379> acl list
1) "user default on nopass ~* +@all"
2) "user xinjing on #a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3 -@all"
```

> `user` 代表是用户
> `default` 代表默认用户（反之 为自己创建的用户）
> `on` 代表激活（反之`off`,默认新增的为`off`）
> `nopass` 代表不需要密码
> `~*` 代表可以访问的`key`
> `+@all` 代表可以操作的`command`

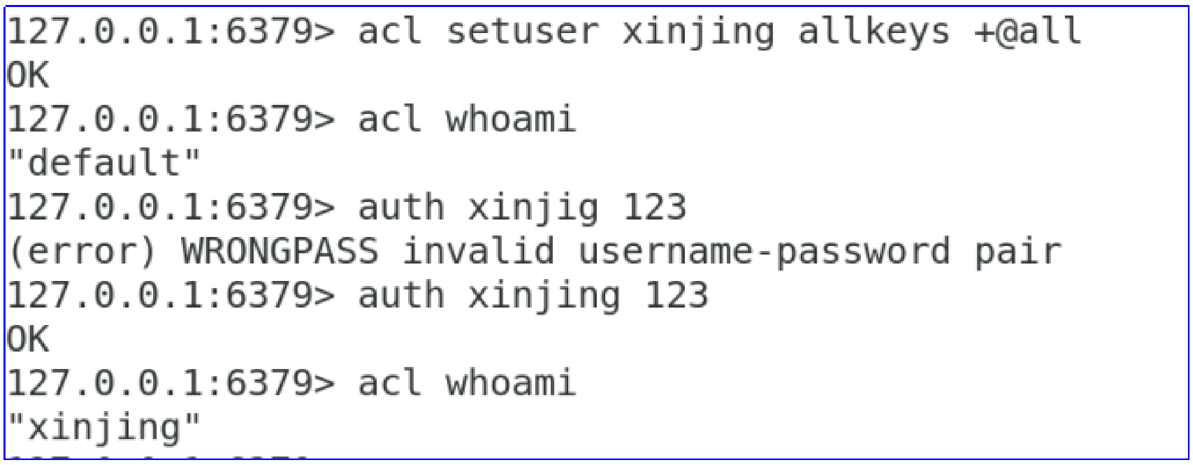
Key的说明

```
得到一个用户详细信息
```



```
127.0.0.1:6379> acl getuser a1
1) "flags"
2) 1) "off"
3) "passwords"
4) (empty array)
5) "commands"
6) "-@all"
7) "keys"
8) (empty array)
```

```
给用户赋权限
acl setuser xinjing allkeys +@all
acl setuser a1 +@string
Acl setuser a2 ~z* +@string -@hash
```

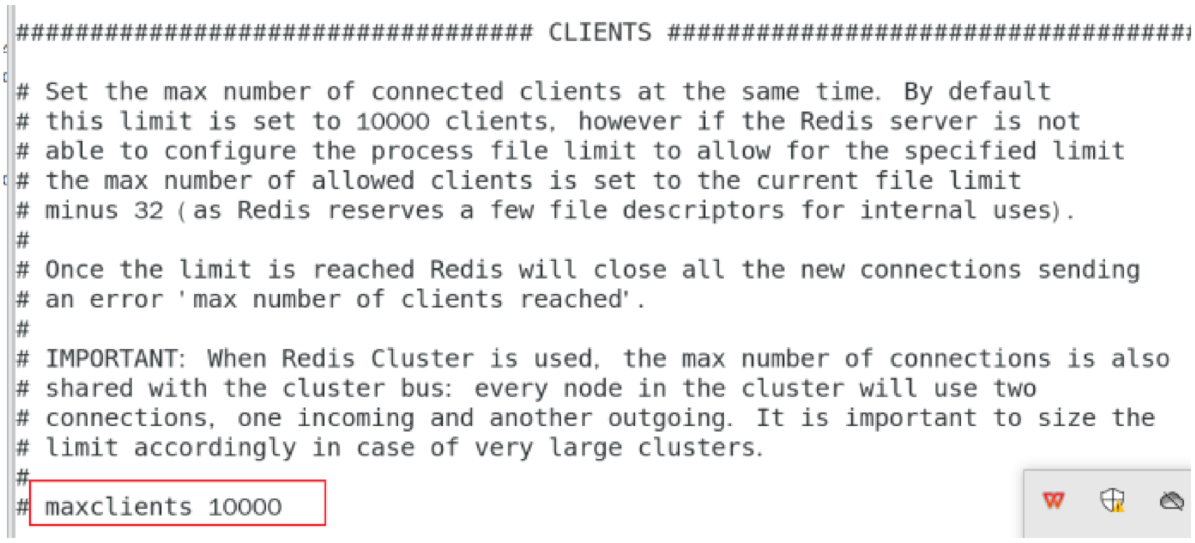


```
127.0.0.1:6379> acl setuser xinjing allkeys +@all
OK
127.0.0.1:6379> acl whoami
"default"
127.0.0.1:6379> auth xinjig 123
(error) WRONGPASS invalid username-password pair
127.0.0.1:6379> auth xinjing 123
OK
127.0.0.1:6379> acl whoami
"xinjing"
```

```
    +<command>：将命令添加到用户可以调用的命令列表中，如+@hash
    -<command>: 将命令从用户可以调用的命令列表中移除
    +@<category>: 添加一类命令，如：@admin, @set, @hash ... 可以ACL CAT 查看具体的操作指令。特殊类别@all表示所有命令，包括当前在服务器中存在的命令，以及将来将通过模块加载的命令
    -@<category>: 类似+@<category>，从客户端可以调用的命令列表中删除命令
    +<command>|subcommand: 允许否则禁用特定子命令。注意，这种形式不允许像-DEBUG | SEGFAULT那样，而只能以“ +”开头
    allcommands：+@all的别名，允许所有命令操作执行。注意，这意味着可以执行将来通过模块系统加载的所有命令。
    nocommands：-@all的别名，不允许所有命令操作执行。
```

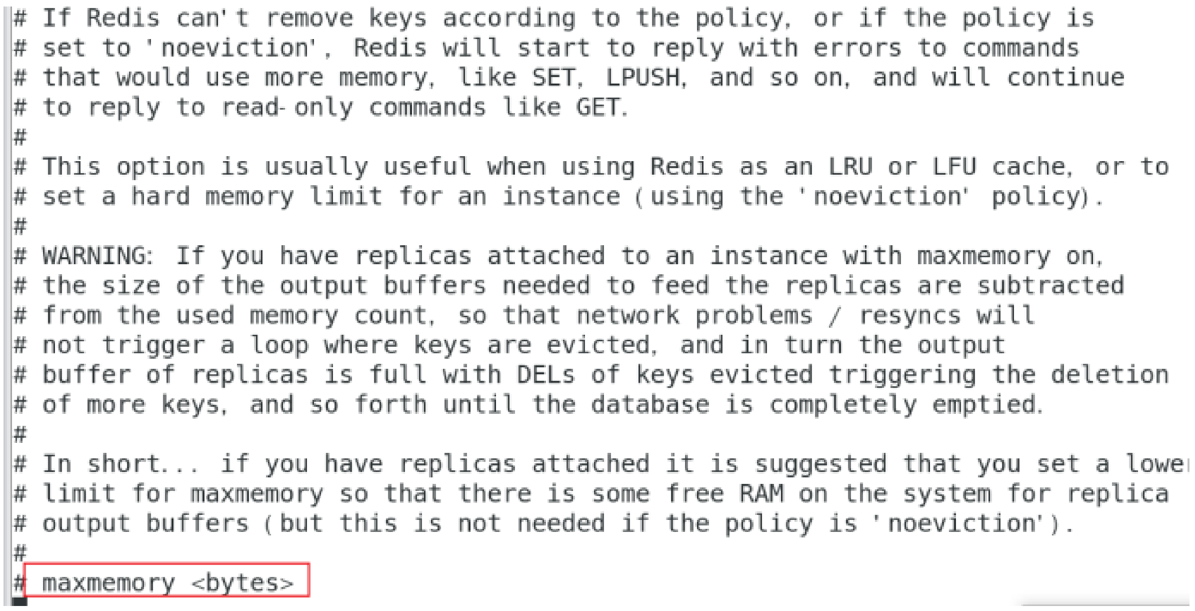
最大客户端连接数

```
Maxclients
```

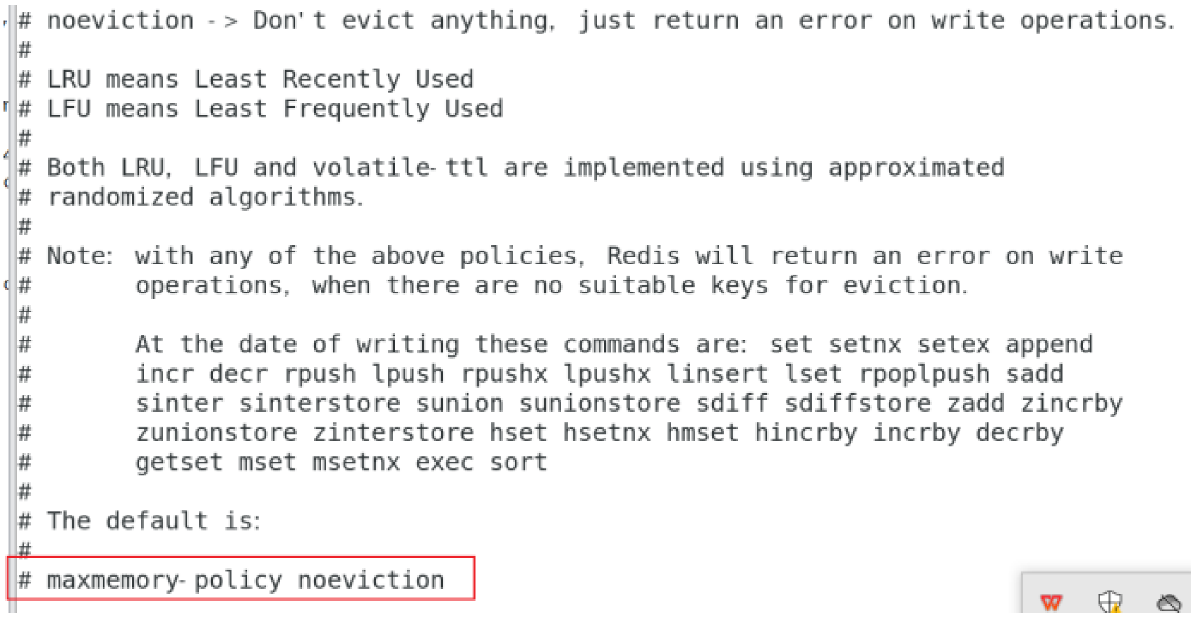

```
################################### CLIENTS ####################################

# Set the max number of connected clients at the same time. By default
# this limit is set to 10000 clients, however if the Redis server is not
# able to configure the process file limit to allow for the specified limit
# the max number of allowed clients is set to the current file limit
# minus 32 (as Redis reserves a few file descriptors for internal uses).
#
# Once the limit is reached Redis will close all the new connections sending
# an error 'max number of clients reached'.
#
# IMPORTANT: When Redis Cluster is used, the max number of connections is also
# shared with the cluster bus: every node in the cluster will use two
# connections, one incoming and another outgoing. It is important to size the
# limit accordingly in case of very large clusters.
#
# maxclients 10000
```

内存设置

```
redis最大内存容量
配置文件中 Maxmemory
```


```
# If Redis can't remove keys according to the policy, or if the policy is
# set to 'noeviction', Redis will start to reply with errors to commands
# that would use more memory, like SET, LPUSH, and so on, and will continue
# to reply to read-only commands like GET.
#
# This option is usually useful when using Redis as an LRU or LFU cache, or to
# set a hard memory limit for an instance (using the 'noeviction' policy).
#
# WARNING: If you have replicas attached to an instance with maxmemory on,
# the size of the output buffers needed to feed the replicas are subtracted
# from the used memory count, so that network problems / resyncs will
# not trigger a loop where keys are evicted, and in turn the output
# buffer of replicas is full with DELs of keys evicted triggering the deletion
# of more keys, and so forth until the database is completely emptied.
#
# In short... if you have replicas attached it is suggested that you set a lowe
# limit for maxmemory so that there is some free RAM on the system for replica
# output buffers (but this is not needed if the policy is 'noeviction').
#
# maxmemory <bytes>
```

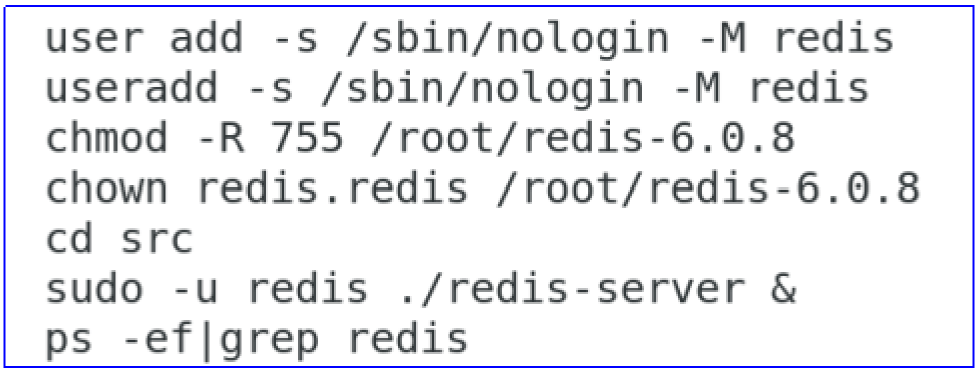
```
    内存达到上限的处理策略
    常见的内存替换策略
    - LRU (Least recently used) 最近最少使用，如果数据最近被访问过，那么将来被访问的几率也更高。
    - LFU (Least frequently used) 最不经常使用，如果一个数据在最近一段时间内使用次数很少，那么在将来一段时间内被使用的可能性也很小。
    - FIFO (Fist in first out) 先进先出， 如果一个数据最先进入缓存中，则应该最早淘汰掉。

    volatile-lru：只对设置了过期时间的key进行LRU（默认值）
    allkeys-lru ： 删除lru算法的key
    volatile-random：随机删除即将过期key
    allkeys-random：随机删除
    volatile-ttl ： 删除即将过期的
    noeviction ： 永不过期，返回错误
```



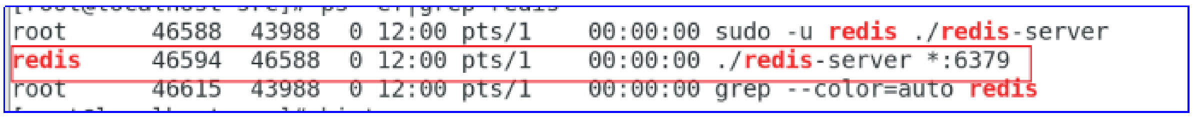
```
# noeviction -> Don't evict anything, just return an error on write operations.
#
# LRU means Least Recently Used
# LFU means Least Frequently Used
#
# Both LRU, LFU and volatile-ttl are implemented using approximated
# randomized algorithms.
#
# Note: with any of the above policies, Redis will return an error on write
#       operations, when there are no suitable keys for eviction.
#
#       At the date of writing these commands are: set setnx setex append
#       incr decr rpush lpush rpushx lpushx linsert lset rpoplpush sadd
#       sinter sinterstore sunion sunionstore sdiff sdiffstore zadd zincrby
#       zunionstore zinterstore hset hsetnx hmset hincrby incrby decrby
#       getset mset msetnx exec sort
#
# The default is:
#
# maxmemory-policy noeviction
```

创建普通用户启动 Redis

```
cd redis-6.0.8/
useradd -s /sbin/nologin -M redis
chmod -R 755 /root/redis-6.0.8
chown redis.redis /root/redis-6.0.8
cd src
sudo -u redis ./redis-server &
ps -ef|grep redis
```



```
user add -s /sbin/nologin -M redis
useradd -s /sbin/nologin -M redis
chmod -R 755 /root/redis-6.0.8
chown redis.redis /root/redis-6.0.8
cd src
sudo -u redis ./redis-server &
ps -ef|grep redis
```



```
root      46588  43988  0 12:00 pts/1    00:00:00 sudo -u redis ./redis-server
redis     46594  46588  0 12:00 pts/1    00:00:00 ./redis-server *:6379
root      46615  43988  0 12:00 pts/1    00:00:00 grep --color=auto redis
```