

# 一、安装redis

## 1、下载Redis安装包

官网下载：

<https://redis.io/download>  
<http://www.redis.cn/download.html>

百度云下载：

链接：<https://pan.baidu.com/s/1TJQqj7fQyXpkNRF1iHNF8A>

提取码：xt76

wget下载：

```
wget http://download.redis.io/releases/redis-6.0.1.tar.gz
```

## 2、解压压缩包

将下载的压缩包通过winSCP上传到服务器，运行如下命令解压：

```
tar -zxvf redis-6.0.1.tar.gz
```

## 3、安装 gcc 编译

因为后面安装redis的时候需要编译，所以事先得先安装gcc编译。阿里云主机已经默认安装了 gcc，如果是自己安装的虚拟机，那么需要先安装一下 gcc：

### 能上网

```
yum install gcc-c++
```

### 不能上网

gcc是linux下的一个编译程序，是C程序的编译工具。

GCC(GNU Compiler Collection) 是 GNU(GNU's Not Unix) 计划提供的编译器家族，它能够支持 C, C++, Objective-C, Fortran, Java 和 Ada 等等程序设计语言前端，同时能够运行在 x86, x86-64, IA-64, PowerPC, SPARC 和 Alpha 等等几乎目前所有的硬件平台上。鉴于这些特征，以及 GCC 编译代码的高效性，使得 GCC 成为绝大多数自由软件开发编译的首选工具。虽然对于程序员们来说，编译器只是一个工具，除了开发和维护人员，很少有人关注编译器的发展，但是 GCC 的影响力是如此之大，它的性能提升甚至有望改善所有的自由软件的运行效率，同时它的内部结构的变化也体现出现代编译器发展的新特征。

下载依赖包：

官网: <https://pkgs.org/>

百度云下载 (已下载好所有依赖包):

链接: [https://pan.baidu.com/s/1B\\_4et9GsH51LKWKTD5bitQ](https://pan.baidu.com/s/1B_4et9GsH51LKWKTD5bitQ)

提取码: 82b4

使用WinSCP上传到服务器后, 运行如下命令:

```
rpm -Uvh *.rpm --nodeps --force
```






安装好, 查询下:

```
gcc -v
```

```
[root@localhost gcc]# gcc -v
使用内建系统。
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
目标: x86_64-redhat-linux
配置为: ./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstrap --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-cxx-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enable-languages=C,C++,ObjC,ObjC++,Java,Fortran,Ada,Go,Lto --enable-plugin --enable-initfini-ar-y --disable-lto --with-isl=/build/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/isl-install --with-cloog=/build/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/cloog-install --enable-gnu-indirect-function --with-tune=generic --with-arch=32-x86-64 --build=x86_64-redhat-linux
线程模型: posix
gcc 版本 4.8.5 20150623 (Red Hat 4.8.5-39) (GCC)
[root@localhost gcc]#
```

离线升级gcc为9.3, 编译安装章节中有在线升级。下载gcc升级包:

电脑 > 工作 (D:) > Tools > codeTools > gcc > 升级9.3

名称	修改日期	类型	大小
 gcc-9.3.0.tar.gz	2020/5/8 10:51	360压缩	121,231 KB
 gmp-6.1.0.tar.bz2	2020/5/8 11:06	360压缩	2,328 KB
 isl-0.18.tar.bz2	2020/5/8 11:06	360压缩	1,620 KB
 mpc-1.0.3.tar.gz	2020/5/8 11:05	360压缩	655 KB
 mpfr-3.1.4.tar.bz2	2020/5/8 11:05	360压缩	1,250 KB

解压gcc-9.3.0.tar.gz:

```
tar -zxvf gcc-9.3.0.tar.gz
```

将另外四个上传到解压目录gcc-9.3.0下。

进入gcc-9.3.0目录下:

```
cd gcc-9.3.0/
./contrib/download_prerequisites
```

若出现找不到wget或tar: bzip2: 无法 exec: 没有那个文件或目录错误, 需要安装wget和bzip2:

```
yum install wget
yum install -y bzip2
```

离线安装可自行百度

```
#创建预编译目录
mkdir build && cd build
```

```
#设置编译选项并编译
```

```
../configure --prefix=/usr/local/gcc-9.3.0 --enable-bootstrap --enable-checking=release --enable-languages=c,c++ --disable-multilib
```

#安装

#编译生成makefile文件

make

#安装GCC

make install

#安装后的设置

#设置环境变量

touch /etc/profile.d/gcc.sh

sudo chmod 777 /etc/profile.d/gcc.sh

sudo echo -e '\nexport PATH=/usr/local/gcc-9.3.0/bin:\$PATH\n' >> /etc/profile.d/gcc.sh && source /etc/profile.d/gcc.sh

#设置头文件

sudo ln -sv /usr/local/gcc/include/ /usr/include/gcc

#设置库文件

touch /etc/ld.so.conf.d/gcc.conf

sudo chmod 777 /etc/ld.so.conf.d/gcc.conf

sudo echo -e "/usr/local/gcc/lib64" >> /etc/ld.so.conf.d/gcc.conf

#加载动态连接库

sudo ldconfig -v

ldconfig -p |grep gcc

#测试版本号

gcc -v

## 4、编译安装

然后将解压的文件夹 redis-6.0.1 放到 /usr/local/ 下，一般安装软件都放在 /usr/local 下。然后进入 /usr/local/redis-6.0.1/ 文件夹下，执行 make 命令即可完成安装。

```
mv /home/redis/redis-6.0.1 /usr/local
cd /usr/local/redis-6.0.1/
make MALLOC=libc
```

二次make;

Jemalloc/jemalloc.h: 没有那个文件或目录。运行make distclean之后再make MALLOC=libc:

# 编译出错时，清出编译生成的文件

make distclean

```
cc -o luac luac.o print.o liblua.a -lm
make[3]: 离开目录"/usr/local/redis-6.0.1/deps/lua/src"
make[2]: 离开目录"/usr/local/redis-6.0.1/deps"
make[1]: *** 没有规则可以创建"adlist.o"需要的目标"/usr/local/redis-6.0.1/deps/jemalloc/include/jemalloc/jemalloc.h"。 停止。
make[1]: 离开目录"/usr/local/redis-6.0.1/src"
make: *** [all] 错误 2
```

报错: server.c:xxxx:xx: error: 'xxxxxxx' has no member named 'xxxxx'

```

server.c:5101:19: error: 'struct redisServer' has no member named 'sofd'
    if (server.sofd > 0)
        ^
server.c:5102:94: error: 'struct redisServer' has no member named 'unixsocket'
    serverLog(LL_NOTICE, "The server is now ready to accept connections at %s", server.unixsock
        ^
server.c:5103:19: error: 'struct redisServer' has no member named 'supervised_mode'
    if (server.supervised_mode == SUPERVISED_SYSTEMD) {
        ^
server.c:5104:24: error: 'struct redisServer' has no member named 'masterhost'
    if (!server.masterhost) {
        ^
server.c:5117:15: error: 'struct redisServer' has no member named 'maxmemory'
    if (server.maxmemory > 0 && server.maxmemory < 1024*1024) {
        ^
server.c:5117:39: error: 'struct redisServer' has no member named 'maxmemory'

```

解决办法：

```

# 查看gcc版本是否在5.3以上，centos7.6默认安装4.8.5
gcc -v
# 升级gcc到5.3及以上，如下：
升级到gcc 9.3：
yum -y install centos-release-scl
yum -y install devtoolset-9-gcc devtoolset-9-gcc-c++ devtoolset-9-binutils
scl enable devtoolset-9 bash
需要注意的是scl命令启用只是临时的，退出shell或重启就会恢复原系统gcc版本。
如果要长期使用gcc 9.3的话：

echo "source /opt/rh/devtoolset-9/enable" >>/etc/profile
这样退出shell重新打开就是新版的gcc了
以下其他版本同理，修改devtoolset版本号即可。

```

make完成后继续执行make install：

```

# 编译安装到指定目录下
make PREFIX=/usr/local/redis-6.0.1 install
# 卸载
make uninstall

```

```

[root@localhost redis-6.0.1]# make PREFIX=/usr/local/redis-6.0.1 install
cd src && make install
make[1]: 进入目录"/usr/local/redis-6.0.1/src"
CC Makefile.dep
make[1]: 离开目录"/usr/local/redis-6.0.1/src"
make[1]: 进入目录"/usr/local/redis-6.0.1/src"

Hint: It's a good idea to run 'make test' ;)

INSTALL install
INSTALL install
INSTALL install
INSTALL install
INSTALL install
make[1]: 离开目录"/usr/local/redis-6.0.1/src"
[root@localhost redis-6.0.1]# ll
总用量 236
-rw-rw-r--. 1 root root 53405 5月 2 06:10 00-RELEASENOTES
drwxr-xr-x. 2 root root 134 5月 8 18:03 bin
-rw-rw-r--. 1 root root 53 5月 2 06:10 BUGS
-rw-rw-r--. 1 root root 2381 5月 2 06:10 CONTRIBUTING
-rw-rw-r--. 1 root root 1487 5月 2 06:10 COPYING
drwxr-xr-x. 6 root root 192 5月 8 17:56 deps
-rw-rw-r--. 1 root root 11 5月 2 06:10 INSTALL
-rw-rw-r--. 1 root root 151 5月 2 06:10 Makefile
-rw-rw-r--. 1 root root 6888 5月 2 06:10 MANIFESTO
-rw-rw-r--. 1 root root 20812 5月 2 06:10 README.md
-rw-rw-r--. 1 root root 81476 5月 2 06:10 redis.conf
-rwxr-xr-x. 1 root root 275 5月 2 06:10 runtest
-rwxr-xr-x. 1 root root 280 5月 2 06:10 runtest-cluster
-rwxr-xr-x. 1 root root 679 5月 2 06:10 runtest-moduleapi
-rwxr-xr-x. 1 root root 281 5月 2 06:10 runtest-sentinel
-rw-rw-r--. 1 root root 10743 5月 2 06:10 sentinel.conf
drwxr-xr-x. 3 root root 8192 5月 8 18:03 src
drwxr-xr-x. 11 root root 182 5月 2 06:10 tests
-rw-rw-r--. 1 root root 3167 5月 2 06:10 TLS.md
drwxr-xr-x. 9 root root 4096 5月 2 06:10 utils
[root@localhost redis-6.0.1]# cd bin/
[root@localhost bin]# ll
总用量 18376
-rwxr-xr-x. 1 root root 797120 5月 8 18:03 redis-benchmark
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-check-aof
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-check-rdb
-rwxr-xr-x. 1 root root 1120352 5月 8 18:03 redis-cli
lrwxrwxrwx. 1 root root 12 5月 8 18:03 redis-sentinel -> redis-server
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-server
[root@localhost bin]#

```

redis-benchmark:性能测试工具，可以在自己本子运行，看看自己本子性能如何， 服务启动起来后执行。

redis-check-aof: 修复有问题的AOF文件，rdb和aof后面讲

redis-check-dump: 修复有问题的dump.rdb文件

redis-cli: 客户端，操作入口

redis-sentinel: redis集群使用

redis-server: Redis服务器启动命令

## 二、启动redis的三种方式

先切换到redis的bin目录下：

```

[root@localhost redis-6.0.1]# cd bin/
[root@localhost bin]# ll
总用量 18376
-rwxr-xr-x. 1 root root 797120 5月 8 18:03 redis-benchmark
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-check-aof
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-check-rdb
-rwxr-xr-x. 1 root root 1120352 5月 8 18:03 redis-cli
lrwxrwxrwx. 1 root root 12 5月 8 18:03 redis-sentinel -> redis-server
-rwxr-xr-x. 1 root root 5630272 5月 8 18:03 redis-server
[root@localhost bin]#

```

### 1、直接启动redis

```
./redis-server
```

```
[root@localhost bin]# ./redis-server
16491:C 08 May 2020 18:09:13.234 # o000o000o000o Redis is starting o000o000o000o
16491:C 08 May 2020 18:09:13.234 # Redis version=6.0.1, bits=64, commit=00000000, modified=0, pid=16491, just started
16491:C 08 May 2020 18:09:13.234 # Warning: no config file specified, using the default config. In order to specify a config file use ./redis-server /path/to/redis.conf
16491:M 08 May 2020 18:09:13.235 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 6.0.1 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 16491

http://redis.io

16491:M 08 May 2020 18:09:13.237 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
16491:M 08 May 2020 18:09:13.237 # Server initialized
16491:M 08 May 2020 18:09:13.237 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then run
'tl vm.overcommit_memory=1' for this to take effect.
16491:M 08 May 2020 18:09:13.237 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command
'/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
16491:M 08 May 2020 18:09:13.237 * Ready to accept connections
```

## 2、以后台进程方式启动redis

### (1) 修改redis.conf文件

打开 redis 配置文件：

```
cd /usr/local/redis-6.0.1
vim redis.conf
```

在命令模式下输入 /bind 来查找 bind 配置，按enter后，再按 n 来查找下一个，找到配置后，将 bind 配置成

0.0.0.0，允许任意服务器来访问 redis，即：

```
bind 0.0.0.0
```

使用同样的方法，将 daemonize 改成 yes（默认为 no），允许 redis 在后台执行。

将 requirepass 注释打开，并设置密码为 123456（密码自己设置）。

### (2) 指定redis.conf文件启动

```
./redis-server /usr/local/redis-6.0.1/redis.conf
```

```
[root@localhost redis-6.0.1]# cd bin/
[root@localhost bin]# ll
总用量 18380
-rw-r--r--. 1 root root    92 5月  8 18:09 dump.rdb
-rwxr-xr-x. 1 root root 797120 5月  8 18:03 redis-benchmark
-rwxr-xr-x. 1 root root 5630272 5月  8 18:03 redis-check-aof
-rwxr-xr-x. 1 root root 5630272 5月  8 18:03 redis-check-rdb
-rwxr-xr-x. 1 root root 1120352 5月  8 18:03 redis-cli
lrwxrwxrwx. 1 root root    12 5月  8 18:03 redis-sentinel -> redis-server
-rwxr-xr-x. 1 root root 5630272 5月  8 18:03 redis-server
[root@localhost bin]# ./redis-server /usr/local/redis-6.0.1/redis.conf
16893:C 08 May 2020 18:17:19.847 # o000o000o000o Redis is starting o000o000o000o
16893:C 08 May 2020 18:17:19.847 # Redis version=6.0.1, bits=64, commit=00000000, modified=0, pid=16893, just started
16893:C 08 May 2020 18:17:19.847 # Configuration loaded
[root@localhost bin]# ps -ef|grep redis
root      16894      1  0 18:17 ?        00:00:00 ./redis-server 0.0.0.0:6379
root      16938    14862  0 18:18 pts/0    00:00:00 grep --color=auto redis
[root@localhost bin]#
```

## 3、设置redis开机自启动

### (1) 在/etc目录下新建redis目录

```
cd /etc
mkdir redis
```

## (2) 将/usr/local/redis-6.0.1/redis.conf 文件复制一份到/etc/redis目录下，并命名为6379.conf

```
cp /usr/local/redis-6.0.1/redis.conf /etc/redis/6379.conf
```

## (3) 将redis的启动脚本复制一份放到/etc/init.d目录下

```
cp /usr/local/redis-6.0.1/utils/redis_init_script /etc/init.d/redisd
```

```
vim redisd
```

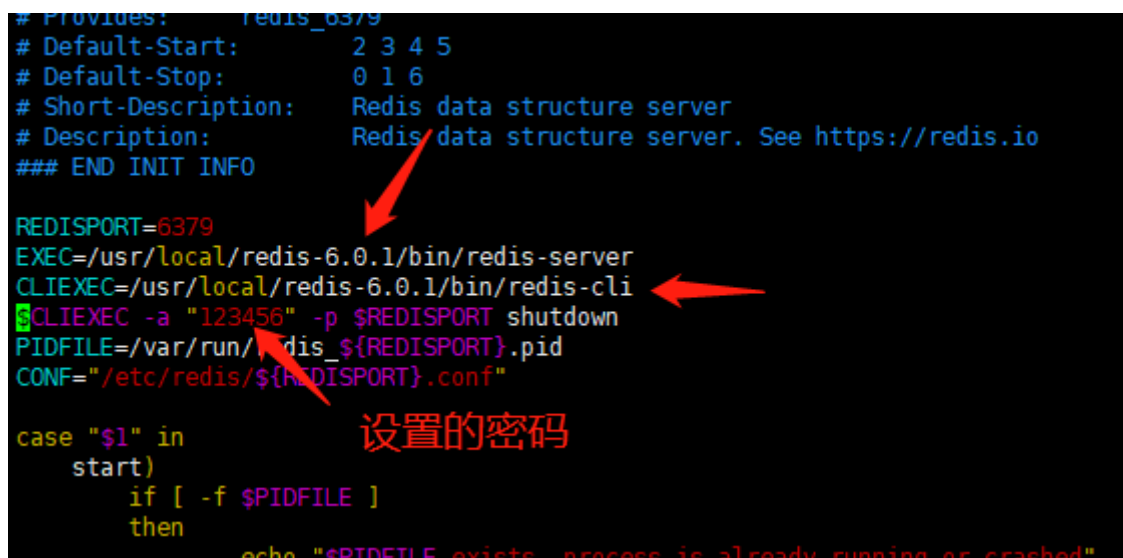
#修改如下

```
EXEC=/usr/local/redis-6.0.1/bin/redis-server
```

```
CLIEXEC=/usr/local/redis-6.0.1/bin/redis-cli
```

```
$CLIEXEC -a "123456" -p $REDISPORT shutdown
```

# 这里设置密码是因为修改了配置文件redis.conf里面设置了redis密码，后面所有命令中的-a 123456，其实是给操作redis授权



```
# Provides:      redis_6379
# Default-Start: 2 3 4 5
# Default-Stop:  0 1 6
# Short-Description: Redis data structure server
# Description:    Redis data structure server. See https://redis.io
### END INIT INFO

REDISPORT=6379
EXEC=/usr/local/redis-6.0.1/bin/redis-server
CLIEXEC=/usr/local/redis-6.0.1/bin/redis-cli
$CLIEXEC -a "123456" -p $REDISPORT shutdown
PIDFILE=/var/run/redis_${REDISPORT}.pid
CONF="/etc/redis/${REDISPORT}.conf"

case "$1" in
    start)
        if [ -f $PIDFILE ]
        then
            echo "$PIDFILE exists, process is already running or crashed"
```

## (4) 设置redis开机自启动

先切换到/etc/init.d目录下

```
cd /etc/init.d
```

然后执行自启命令

```
chkconfig redisd on
```

如果报：service redisd does not support chkconfig。看结果是redisd不支持chkconfig

解决方法：

使用vim编辑redisd文件，在第一行加入如下两行注释，保存退出

```
# chkconfig: 2345 90 10
# description: Redis is a persistent key-value database
```

注释的意思是，redis服务必须在运行级2，3，4，5下被启动或关闭，启动的优先级是90，关闭的优先级是10。

```
#!/bin/sh
# chkconfig: 2345 90 10
# description: Redis is a persistent key-value database
```

再次执行开机自启命令，成功

```
chkconfig redisd on
```

现在可以直接以服务的形式启动和关闭redis了

启动：

```
service redisd start
```

关闭：

方法1: `service redisd stop`

方法2：

#单实例关闭：

```
./redis-cli -a 123456 shutdown
```

#多实例关闭，指定端口关闭：

```
./redis-cli -p 6379 -a 123456 shutdown
```

```
[root@localhost bin]# ./redis-cli -a 123456 shutdown
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
[root@localhost bin]# ps -ef|grep redis
root      18563  14862  0 18:44 pts/0    00:00:00 grep --color=auto redis
```

## (5) 使用：永远的helloworld

```
[root@localhost ~]# cd /usr/local/redis-6.0.1/bin/
[root@localhost bin]#
[root@localhost bin]# ./redis-cli
127.0.0.1:6379> set key1 hellword
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth 123456
OK
127.0.0.1:6379> set key1 hellword
OK
127.0.0.1:6379> get key1
"hellword"
127.0.0.1:6379> quit
[root@localhost bin]#
```



# 三、 防火墙开放6379端口

```
firewall-cmd --zone=public --add-port=6379/tcp --permanent
firewall-cmd --reload #重新加载
```

