cpp\_redis测试

一.简介

cpp\_redis是一个基于C++11实现的轻量级客户端，它支持同步，异步，线程安全，管道操作，还有Redis Sentinel,支持多平台，编译过程只需要C++11的配置环境，同时还提供了详细的API文档（https://cylix.github.io/cpp\_redis/html/classcpp\_\_redis\_1\_1client.html）

源码地址：https://github.com/cylix/cpp\_redis

二.安装（在终端窗口中用shell命令操作）

1.从github上获取安装文件

git clone https://github.com/Cylix/cpp\_redis.git

2.进入下载下来的文件夹

cd cpp\_redis

3.获取tacopie模块

git submodule init && git submodule update

4.创建一个build文件夹

mkdir build && cd build

5.生成Makefile文件

cmake .. -DCMAKE\_BUILD\_TYPE=Release

6.编译

make

7.安装

make install

然后就可以使用使用<cpp\_redis/cpp\_redis>与cpp\_redis，tacopie库在工程中调用客户端的API

三.测试get,set命令

1.同步

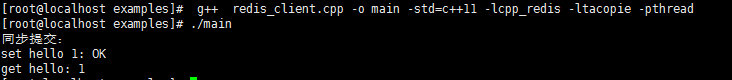
1. #include <cpp\_redis/cpp\_redis>
2. #include <iostream>
3. **using** **namespace** std;
4. **int** main()
5. {
6. cpp\_redis::active\_logger = std::unique\_ptr<cpp\_redis::logger>(**new** cpp\_redis::logger);
8. cpp\_redis::client client;
10. //连接到客户端，因为是在本机测试，且服务端端口设置为6379
11. client.connect("127.0.0.1", 6379, [](**const** std::string& host, std::**size\_t** port, cpp\_redis::client::connect\_state status) {
12. **if** (status == cpp\_redis::client::connect\_state::dropped) {
13. std::cout << "client disconnected from " << host << ":" << port << std::endl;
14. }
15. });
17. client.set("hello", "1", [](cpp\_redis::reply& reply) {
18. std::cout << "set hello 1: " << reply << std::endl;
19. });
20. client.get("hello", [](cpp\_redis::reply& reply) {
21. std::cout << "get hello: " << reply << std::endl;
23. });
24. cout<<"同步提交："<<endl;
25. client.sync\_commit();

28. **return** 0;
30. }

编译：

g++ redis\_client.cpp -o main -std=c++11 -lcpp\_redis -ltacopie -pthread

运行结果



2. 异步

通过API介绍

[**commit()**](https://cylix.github.io/cpp_redis/html/classcpp__redis_1_1client.html#a36a48d61a4900e88fd67795ca59cbea3) works asynchronously: it returns immediately after sending the queued requests and replies are processed asynchronously.

可知commit是异步的

1. #include <cpp\_redis/cpp\_redis>
2. #include <iostream>
3. **using** **namespace** std;
4. **int** main()
5. {
6. cpp\_redis::active\_logger = std::unique\_ptr<cpp\_redis::logger>(**new** cpp\_redis::logger);
8. cpp\_redis::client client;
10. client.connect("127.0.0.1", 6379, [](**const** std::string& host, std::**size\_t** port,     cpp\_redis::client::connect\_state status) {
11. **if** (status == cpp\_redis::client::connect\_state::dropped) {
12. std::cout << "client disconnected from " << host << ":" << port << std::endl;
13. }
14. });
15. client.set("hello", "1", [](cpp\_redis::reply& reply) {
16. std::cout << "set hello 1: " << reply << std::endl;
17. });
18. client.get("hello", [](cpp\_redis::reply& reply) {
19. std::cout << "get hello: " << reply << std::endl;
21. });
22. cout<<"异步提交："<<endl;
23. client.commit();
24. **while**(1){};
25. //因为是异步提交，不会等待服务器处理完才返回，若没有循环，则没有输出，因为可能来不及返回，程序就return退出了
27. **return** 0;
29. }

编译输出：

