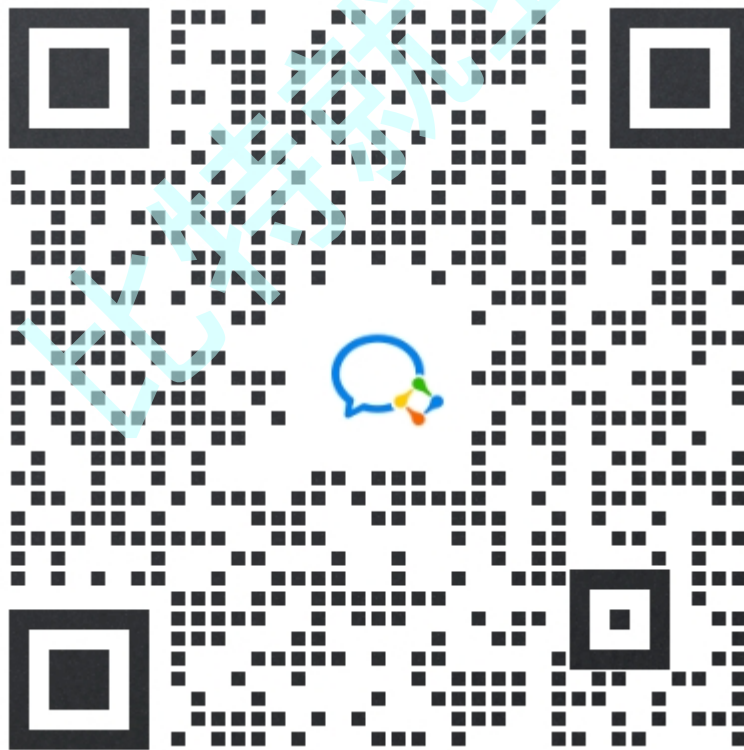


Redis C++使用 样例列表

版权说明

本“比特就业课”课程（以下简称“本课程”）的所有内容，包括但不限于文字、图片、音频、视频、软件、程序、数据库、设计、布局、界面等，均由本课程的开发者或授权方拥有版权。我们鼓励个人学习者使用本课程进行学习和研究。在遵守相关法律法规的前提下，个人学习者可以下载、浏览、学习本课程的内容，并为了个人学习、研究或教学目的而使用其中的材料。但请注意，未经我们明确授权，个人学习者不得将本课程的内容用于任何商业目的，包括但不限于销售、转让、许可或以其他方式从中获利。此外，个人学习者也不得擅自修改、复制、传播、展示、表演或制作本课程内容的衍生作品。任何未经授权的使用均属侵权行为，我们将依法追究法律责任。如果您希望以其他方式使用本课程的内容，包括但不限于引用、转载、摘录、改编等，请事先与我们联系，获取书面授权。感谢您对“比特就业课”课程的关注与支持，我们将持续努力，为您提供更好的学习体验。特此说明。比特就业课版权所有方

对比特课程感兴趣，可以联系这个微信。



代码 & 板书链接

<https://gitee.com/HGtz2222/bitproject/tree/master/redis>

安装 redis-plus-plus

C++ 操作 redis 的库有很多. 咱们此处使用 redis-plus-plus.

这个库的功能强大, 使用简单.

Github 地址: <https://github.com/sewnew/redis-plus-plus>

安装 hiredis

redis-plus-plus 是基于 hiredis 实现的.

| hiredis 是一个 C 语言实现的 redis 客户端.

因此需要先安装 hiredis. 直接使用包管理器安装即可.

Ubuntu

```
1 apt install libhiredis-dev
```

Centos

```
1 yum install hiredis-devel.x86_64
```

下载 redis-plus-plus 源码

```
1 git clone https://github.com/sewnew/redis-plus-plus.git
```

编译/安装 redis-plus-plus

Ubuntu

使用 cmake 构建

```
1 cd redis-plus-plus
2
```

```
3 mkdir build
4
5 cd build
6
7 cmake ..
8
9 make
10
11 make install    # 这一步操作需要管理员权限。如果是非 root 用户，使用 sudo make
                  install 执行。
```

构建成功后, 会在 `/usr/local/include/` 中多出 `sw` 目录, 并且内部包含 redis-plus-plus 的一系列头文件.

会在 `/usr/local/lib/` 中多出一系列 `libredis` 库文件.

Centos

Centos 自带的 cmake 版本较低, 需要先安装 cmake3

```
1 yum install cmake3
```

然后使用 cmake3 构建项目.

```
1 cd redis-plus-plus
2
3 mkdir build
4
5 cd build
6
7 cmake3 ..
8
9 make
10
11 make install    # 这一步操作需要管理员权限。如果是非 root 用户，使用 sudo make
                  install 执行。
```

构建成功后, 会在 `/usr/local/include/` 中多出 `sw` 目录, 并且内部包含 redis-plus-plus 的一系列头文件.

会在 `/usr/local/lib/` 中多出一系列 `libredis` 库文件.

详细 API

参考：

Github 地址：<https://github.com/sewenew/redis-plus-plus/blob/master/src/sw/redis++/redis.h>

同步到 gitee 后的地址：<https://gitee.com/peixincheng2/redis-plus-plus/blob/master/src/sw/redis++/redis.h>

源码结构

```
1 .
2 |— Makefile
3 |— src
4     |— generic.cc
5     |— list.cc
6     |— set.cc
7     |— string.cc
8
9 1 directory, 5 files
```

Makefile

由于不同系统中, 安装好的库所在位置会存在差异. 因此要注意文件路径是否存在.

Ubuntu

```
1 all: generic string list set
2
3 generic: src/generic.cc
4     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-
        linux-gnu/libhiredis.a -pthread
5
6 string: src/string.cc
7     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-
        linux-gnu/libhiredis.a -pthread
8
9 list: src/list.cc
10    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-
        linux-gnu/libhiredis.a -pthread
11
```

```
12 set: src/set.cc
13     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-
    linux-gnu/libhiredis.a -pthread
14
15 .PHONY: clean
16 clean:
17     rm -rf generic
18     rm -rf string
19     rm -rf list
20     rm -rf set
```

Centos

```
1 all: generic string list set
2
3 generic: src/generic.cc
4     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a
    /usr/local/lib/libhiredis.a -pthread
5
6 string: src/string.cc
7     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a
    /usr/local/lib/libhiredis.a -pthread
8
9 list: src/list.cc
10    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a
    /usr/local/lib/libhiredis.a -pthread
11
12 set: src/set.cc
13     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a
    /usr/local/lib/libhiredis.a -pthread
14
15 .PHONY: clean
16 clean:
17     rm -rf generic
18     rm -rf string
19     rm -rf list
20     rm -rf set
```

src/generic.cc

```

1 #include <sw/redis++/redis++.h>
2 #include <cstdio>
3 #include <string>
4 #include <vector>
5 #include <unordered_map>
6 #include <chrono>
7
8
9 void testGenericCommands(sw::redis::Redis& redis) {
10     printf("Generic 系列命令:\n");
11     printf("=====\n");
12     {
13         printf("EXISTS 命令\n\n");
14         redis.flushdb();
15
16         bool b;
17         b = redis.set("key1", "Hello");
18         printf("redis < SET key1 \"Hello\"\n");
19         printf("redis > %s\n", b ? "OK" : "(nil)");
20
21         long long e;
22         e = redis.exists("key1");
23         printf("redis < EXISTS key1\n");
24         printf("redis > %lld\n", e);
25
26         e = redis.exists("nosuchkey");
27         printf("redis < EXISTS nosuchkey\n");
28         printf("redis > %lld\n", e);
29
30         b = redis.set("key2", "World");
31         printf("redis < SET key2 \"World\"\n");
32         printf("redis > %s\n", b ? "OK" : "(nil)");
33
34         e = redis.exists({"key1", "key2", "nosuchkey"});
35         printf("redis < EXISTS key1 key2 nosuchkey\n");
36         printf("redis > %lld\n", e);
37         printf("=====\n");
38     }
39
40     {
41         printf("DEL 命令\n\n");
42         redis.flushdb();
43         bool b;
44         b = redis.set("key1", "Hello");
45         printf("redis < SET key1 \"Hello\"\n");
46         printf("redis > %s\n", b ? "OK" : "(nil)");
47

```

```

48     b = redis.set("key2", "World");
49     printf("redis < SET key2 \"World\\n\\n");
50     printf("redis > %s\\n", b ? "OK" : "(nil)");
51
52     int d;
53     d = redis.del({"key1", "key2", "key3"});
54     printf("redis < DEL key1 key2 key3\\n");
55     printf("redis > %lld\\n", d);
56     printf("=====\\n");
57 }
58
59 {
60     printf("KEYS 命令\\n\\n");
61     redis.flushdb();
62     std::unordered_map<std::string, std::string> kvs1 = {
63         {"firstname", "Jack"},
64         {"lastname", "Stuntman"},
65         {"age", "35"}
66     };
67     redis.mset(kvs1.begin(), kvs1.end());
68     printf("redis < MSET firstname Jack lastname Stuntman age 35\\n");
69     printf("redis > OK\\n");
70
71     std::vector<std::string> keys;
72     std::insert_iterator<std::vector<std::string>> ins =
std::inserter(keys, keys.begin());
73     redis.keys("*name*", ins);
74     printf("redis < KEYS *name*\\n");
75     int n = 1;
76     for (std::vector<std::string>::iterator it = keys.begin(); it !=
keys.end(); ++it) {
77         printf("redis > %d) %s\\n", n++, it->c_str());
78     }
79
80     keys.clear();
81     ins = std::inserter(keys, keys.begin());
82     redis.keys("a??", ins);
83     printf("redis < KEYS a??\\n");
84     n = 1;
85     for (std::vector<std::string>::iterator it = keys.begin(); it !=
keys.end(); ++it) {
86         printf("redis > %d) %s\\n", n++, it->c_str());
87     }
88
89     keys.clear();
90     ins = std::inserter(keys, keys.begin());
91     redis.keys("*", ins);

```

```

92     printf("redis < KEYS *\n");
93     n = 1;
94     for (std::vector<std::string>::iterator it = keys.begin(); it !=
keys.end(); ++it) {
95         printf("redis > %d) %s\n", n++, it->c_str());
96     }
97
98     printf("=====\n");
99 }
100
101 {
102     printf("EXPIRE 命令\n");
103     printf("TTL 命令\n\n");
104     redis.flushdb();
105     bool b;
106     b = redis.set("mykey", "Hello");
107     printf("redis < SET mykey \"Hello\"\n");
108     printf("redis > %s\n", b ? "OK" : "(nil)");
109
110     b = redis.expire("mykey", std::chrono::seconds(10));
111     printf("redis < EXPIRE mykey 10\n");
112     printf("redis > %d\n", b ? 1 : 0);
113
114     long long ttl = redis.ttl("mykey");
115     printf("redis < TTL mykey\n");
116     printf("redis > %lld\n", ttl);
117
118     printf("=====\n");
119 }
120
121 {
122     printf("PTTL 命令\n\n");
123     redis.flushdb();
124     bool b;
125     b = redis.set("mykey", "Hello");
126     printf("redis < SET mykey \"Hello\"\n");
127     printf("redis > %s\n", b ? "OK" : "(nil)");
128
129     b = redis.expire("mykey", std::chrono::seconds(10));
130     printf("redis < EXPIRE mykey 10\n");
131     printf("redis > %d\n", b ? 1 : 0);
132
133     long long pttl = redis.pttl("mykey");
134     printf("redis < PTTL mykey\n");
135     printf("redis > %lld\n", pttl);
136
137     printf("=====\n");

```



```

138     }
139
140     {
141         printf("TYPE 命令\n\n");
142         redis.flushdb();
143         bool b;
144         b = redis.set("key1", "value");
145         printf("redis < SET key1 \"value\"\n");
146         printf("redis > %s\n", b ? "OK" : "(nil)");
147
148         long long c = redis.lpush("key2", "value");
149         printf("redis < LUSH key2 \"value\"\n");
150         printf("redis > %d\n", c);
151
152         c = redis.sadd("key3", "value");
153         printf("redis < SADD key3 \"value\"\n");
154         printf("redis > %d\n", c);
155
156         std::string type = redis.type("key1");
157         printf("redis < TYPE key1\n");
158         printf("redis > \"%s\"\n", type.c_str());
159
160         type = redis.type("key2");
161         printf("redis < TYPE key2\n");
162         printf("redis > \"%s\"\n", type.c_str());
163
164         type = redis.type("key3");
165         printf("redis < TYPE key3\n");
166         printf("redis > \"%s\"\n", type.c_str());
167
168         printf("=====\n");
169     }
170 }
171
172 int main() {
173     sw::redis::Redis redis("tcp://127.0.0.1:6379");
174
175     // Generic commands
176     testGenericCommands(redis);
177
178     return 0;
179 }

```

generic 测试结果

```
1 $ make generic
2 g++ -std=c++17 -o generic src/generic.cc /usr/local/lib64/libredis++.a
   /usr/local/lib/libhiredis.a -pthread
3 $ ./generic
4 Generic 系列命令:
5 =====
6 EXISTS 命令
7
8 redis < SET key1 "Hello"
9 redis > OK
10 redis < EXISTS key1
11 redis > 1
12 redis < EXISTS nosuchkey
13 redis > 0
14 redis < SET key2 "World"
15 redis > OK
16 redis < EXISTS key1 key2 nosuchkey
17 redis > 2
18 =====
19 DEL 命令
20
21 redis < SET key1 "Hello"
22 redis > OK
23 redis < SET key2 "World"
24 redis > OK
25 redis < DEL key1 key2 key3
26 redis > 2
27 =====
28 KEYS 命令
29
30 redis < MSET firstname Jack lastname Stuntman age 35
31 redis > OK
32 redis < KEYS *name*
33 redis > 1) firstname
34 redis > 2) lastname
35 redis < KEYS a??
36 redis > 1) age
37 redis < KEYS *
38 redis > 1) firstname
39 redis > 2) age
40 redis > 3) lastname
41 =====
42 EXPIRE 命令
43 TTL 命令
44
45 redis < SET mykey "Hello"
46 redis > OK
```

```

47 redis < EXPIRE mykey 10
48 redis > 1
49 redis < TTL mykey
50 redis > 10
51 =====
52 PTTL 命令
53
54 redis < SET mykey "Hello"
55 redis > OK
56 redis < EXPIRE mykey 10
57 redis > 1
58 redis < PTTL mykey
59 redis > 10000
60 =====
61 TYPE 命令
62
63 redis < SET key1 "value"
64 redis > OK
65 redis < LPUSH key2 "value"
66 redis > 1
67 redis < SADD key3 "value"
68 redis > 1
69 redis < TYPE key1
70 redis > "string"
71 redis < TYPE key2
72 redis > "list"
73 redis < TYPE key3
74 redis > "set"
75 =====

```

src/string.cc

```

1  #include <sw/redis++/redis++.h>
2  #include <cstdio>
3  #include <string>
4  #include <chrono>
5  #include <thread>
6
7
8  void testStringCommands(sw::redis::Redis& redis) {
9      printf("String 系列命令:\n");
10     printf("=====\n");
11     {
12         printf("SET 命令\n\n");
13

```

```
14     redis.flushdb();
15
16     long long e = redis.exists("mykey");
17     printf("redis < EXISTS mykey\n");
18     printf("redis > %lld\n", e);
19
20     bool b = redis.set("mykey", "Hello");
21     printf("redis < SET mykey \"Hello\"\n");
22     printf("redis > %s\n", b ? "OK" : "(nil)");
23
24     sw::redis::OptionalString v = redis.get("mykey");
25     printf("redis < GET mykey\n");
26     if (bool(v)) {
27         printf("redis > \"%s\"\n", v->c_str());
28     } else {
29         printf("redis > (nil)\n");
30     }
31
32     b = redis.set("mykey", "World", std::chrono::milliseconds(0),
sw::redis::UpdateType::NOT_EXIST);
33     printf("redis < SET mykey \"World\" NX\n");
34     printf("redis > %s\n", b ? "OK" : "(nil)");
35
36     long long d = redis.del("mykey");
37     printf("redis < DEL mykey\n");
38     printf("redis > %lld\n", d);
39
40     e = redis.exists("mykey");
41     printf("redis < EXISTS mykey\n");
42     printf("redis > %lld\n", e);
43
44     b = redis.set("mykey", "World", std::chrono::seconds(0),
sw::redis::UpdateType::EXIST);
45     printf("redis < SET mykey \"World\" XX\n");
46     printf("redis > %s\n", b ? "OK" : "(nil)");
47
48     v = redis.get("mykey");
49     printf("redis < GET mykey\n");
50     if (bool(v)) {
51         printf("redis > \"%s\"\n", v->c_str());
52     } else {
53         printf("redis > (nil)\n");
54     }
55
56     b = redis.set("mykey", "World", std::chrono::seconds(0),
sw::redis::UpdateType::NOT_EXIST);
57     printf("redis < SET mykey \"World\" NX\n");
```

```

58     printf("redis > %s\n", b ? "OK" : "(nil)");
59
60     v = redis.get("mykey");
61     printf("redis < GET mykey\n");
62     if (bool(v)) {
63         printf("redis > \"%s\"\n", v->c_str());
64     } else {
65         printf("redis > (nil)\n");
66     }
67
68     b = redis.set("mykey", "Will expire in 10s", std::chrono::seconds(10));
69     printf("redis < SET mykey \"Will expire in 10s\" EX 10\n");
70     printf("redis > %s\n", b ? "OK" : "(nil)");
71
72     v = redis.get("mykey");
73     printf("redis < GET mykey\n");
74     if (bool(v)) {
75         printf("redis > \"%s\"\n", v->c_str());
76     } else {
77         printf("redis > (nil)\n");
78     }
79
80     for (int i = 10; i > 0; --i) {
81         printf("Waiting %ds\n", i);
82         std::this_thread::sleep_for(std::chrono::seconds(1));
83     }
84
85     v = redis.get("mykey");
86     printf("redis < GET mykey\n");
87     if (bool(v)) {
88         printf("redis > \"%s\"\n", v->c_str());
89     } else {
90         printf("redis > (nil)\n");
91     }
92
93     printf("=====\n");
94 }
95
96 {
97     printf("GET 命令\n\n");
98
99     redis.flushdb();
100
101     sw::redis::OptionalString v = redis.get("nonexisting");
102     printf("redis < GET nonexisting\n");
103     if (bool(v)) {
104         printf("redis > \"%s\"\n", v->c_str());

```

```

105     } else {
106         printf("redis > (nil)\n");
107     }
108
109     bool b = redis.set("mykey", "Hello");
110     printf("redis < SET mykey \"Hello\"\n");
111     printf("redis > %s\n", b ? "OK" : "(nil)");
112
113     v = redis.get("mykey");
114     printf("redis < GET mykey\n");
115     if (bool(v)) {
116         printf("redis > \"%s\"\n", v->c_str());
117     } else {
118         printf("redis > (nil)\n");
119     }
120
121     long long d = redis.del("mykey");
122     printf("redis < DEL mykey\n");
123     printf("redis > %lld\n", d);
124
125     long long e = redis.exists("mykey");
126     printf("redis < EXISTS mykey\n");
127     printf("redis > %lld\n", e);
128
129     b = redis.hset("mykey", "name", "Bob");
130     printf("redis < HSET mykey name Bob\n");
131     printf("redis > %d\n", b ? 1 : 0);
132
133     try {
134         printf("redis < GET mykey\n");
135         v = redis.get("mykey");
136         if (bool(v)) {
137             printf("redis > \"%s\"\n", v->c_str());
138         } else {
139             printf("redis > (nil)\n");
140         }
141     } catch (const sw::redis::ReplyError& exc) {
142         printf("redis > %s\n", exc.what());
143     }
144
145     printf("=====\n");
146 }
147
148 {
149     printf("APPEND 命令\n\n");
150
151     redis.flushdb();

```

```

152
153     long long e = redis.exists("mykey");
154     printf("redis < EXISTS mykey\n");
155     printf("redis > %lld\n", e);
156
157     long long c = redis.append("mykey", "Hello");
158     printf("redis < APPEND mykey \"Hello\"\n");
159     printf("redis > %lld\n", c);
160
161     sw::redis::OptionalString v = redis.get("mykey");
162     printf("redis < GET mykey\n");
163     if (bool(v)) {
164         printf("redis > \"%s\"\n", v->c_str());
165     } else {
166         printf("redis > (nil)\n");
167     }
168
169     c = redis.append("mykey", " World");
170     printf("redis < APPEND mykey \" World\"\n");
171     printf("redis > %lld\n", c);
172
173     v = redis.get("mykey");
174     printf("redis < GET mykey\n");
175     if (bool(v)) {
176         printf("redis > \"%s\"\n", v->c_str());
177     } else {
178         printf("redis > (nil)\n");
179     }
180
181     printf("=====\n");
182 }
183
184 {
185     printf("GETRANGE 命令\n\n");
186
187     redis.flushdb();
188
189     bool b = redis.set("mykey", "This is a string");
190     printf("redis < SET mykey \"This is a string\"\n");
191     printf("redis > %s\n", b ? "OK" : "(nil)");
192
193     std::string s = redis.getrange("mykey", 0, 3);
194     printf("redis < GETRANGE mykey 0 3\n");
195     printf("redis > %s\n", s.c_str());
196
197     s = redis.getrange("mykey", -3, -1);
198     printf("redis < GETRANGE mykey -3 -1\n");

```

```

199     printf("redis > %s\n", s.c_str());
200
201     s = redis.getrange("mykey", 0, -1);
202     printf("redis < GETRANGE mykey 0 -1\n");
203     printf("redis > %s\n", s.c_str());
204
205     s = redis.getrange("mykey", 10, 100);
206     printf("redis < GETRANGE mykey 10 100\n");
207     printf("redis > %s\n", s.c_str());
208
209     printf("=====\n");
210 }
211
212 {
213     printf("SETEX 命令\n\n");
214
215     redis.flushdb();
216
217     redis.setex("mykey", std::chrono::seconds(10), "Hello");
218     printf("redis < SETEX mykey 10 \"Hello\"\n");
219     printf("redis > OK\n");
220
221     long long ttl = redis.ttl("mykey");
222     printf("redis < TTL mykey\n");
223     printf("redis > %lld\n", ttl);
224
225     sw::redis::OptionalString v = redis.get("mykey");
226     printf("redis < GET mykey\n");
227     if (bool(v)) {
228         printf("redis > \"%s\"\n", v->c_str());
229     } else {
230         printf("redis > (nil)\n");
231     }
232
233     for (int i = 10; i > 0; --i) {
234         printf("Waiting %ds\n", i);
235         std::this_thread::sleep_for(std::chrono::seconds(1));
236     }
237
238     v = redis.get("mykey");
239     printf("redis < GET mykey\n");
240     if (bool(v)) {
241         printf("redis > \"%s\"\n", v->c_str());
242     } else {
243         printf("redis > (nil)\n");
244     }
245

```



```

246     printf("=====\n");
247 }
248
249 {
250     printf("PSETEX 命令\n\n");
251
252     redis.flushdb();
253
254     redis.psetex("mykey", std::chrono::milliseconds(10000), "Hello");
255     printf("redis < PSETEX mykey 10000 \"Hello\"\n");
256     printf("redis > OK\n");
257
258     long long pttl = redis.pttl("mykey");
259     printf("redis < PTTL mykey\n");
260     printf("redis > %lld\n", pttl);
261
262     sw::redis::OptionalString v = redis.get("mykey");
263     printf("redis < GET mykey\n");
264     if (bool(v)) {
265         printf("redis > \"%s\"\n", v->c_str());
266     } else {
267         printf("redis > (nil)\n");
268     }
269
270     for (int i = 10; i > 0; --i) {
271         printf("Waiting %ds\n", i);
272         std::this_thread::sleep_for(std::chrono::seconds(1));
273     }
274
275     v = redis.get("mykey");
276     printf("redis < GET mykey\n");
277     if (bool(v)) {
278         printf("redis > \"%s\"\n", v->c_str());
279     } else {
280         printf("redis > (nil)\n");
281     }
282
283     printf("=====\n");
284 }
285
286 {
287     printf("SETNX 命令\n\n");
288
289     redis.flushdb();
290
291     long long r = redis.setnx("mykey", "Hello");
292     printf("redis < SETNX mykey \"Hello\"\n");

```

```

293     printf("redis > %d\n", r ? 1 : 0);
294
295     r = redis.setnx("mykey", "World");
296     printf("redis < SETNX mykey \"World\"\n");
297     printf("redis > %d\n", r ? 1 : 0);
298
299     sw::redis::OptionalString v = redis.get("mykey");
300     printf("redis < GET mykey\n");
301     if (bool(v)) {
302         printf("redis > \"%s\"\n", v->c_str());
303     } else {
304         printf("redis > (nil)\n");
305     }
306
307     printf("=====\n");
308 }
309
310 {
311     printf("DECR 命令\n\n");
312
313     redis.flushdb();
314
315     long long e = redis.exists("mykey");
316     printf("redis < EXISTS mykey\n");
317     printf("redis > %lld\n", e);
318
319     long long v = redis.decr("mykey");
320     printf("redis < DECR mykey\n");
321     printf("redis > %lld\n", v);
322
323     bool b = redis.set("mykey", "10");
324     printf("redis < SET mykey \"10\"\n");
325     printf("redis > %s\n", b ? "OK": "(nil)");
326
327     v = redis.decr("mykey");
328     printf("redis < DECR mykey\n");
329     printf("redis > %lld\n", v);
330
331     b = redis.set("mykey", "234293482390480948029348230948");
332     printf("redis < SET mykey \"234293482390480948029348230948\"\n");
333     printf("redis > %s\n", b ? "OK": "(nil)");
334
335     try {
336         v = redis.decr("mykey");
337         printf("redis < DECR mykey\n");
338         printf("redis > %lld\n", v);
339     } catch (const sw::redis::ReplyError& exc) {

```

```

340         printf("redis > %s\n", exc.what());
341     }
342
343     b = redis.set("mykey", "not a number");
344     printf("redis < SET mykey \"not a number\"\n");
345     printf("redis > %s\n", b ? "OK": "(nil)");
346
347     try {
348         v = redis.decr("mykey");
349         printf("redis < DECR mykey\n");
350         printf("redis > %lld\n", v);
351     } catch (const sw::redis::ReplyError& exc) {
352         printf("redis > %s\n", exc.what());
353     }
354
355     printf("=====\n");
356 }
357
358 {
359     printf("DECRBY 命令\n\n");
360
361     redis.flushdb();
362
363     long long e = redis.exists("mykey");
364     printf("redis < EXISTS mykey\n");
365     printf("redis > %lld\n", e);
366
367     long long v = redis.decrby("mykey", 3);
368     printf("redis < DECRBY mykey 3\n");
369     printf("redis > %lld\n", v);
370
371     bool b = redis.set("mykey", "10");
372     printf("redis < SET mykey \"10\"\n");
373     printf("redis > %s\n", b ? "OK": "(nil)");
374
375     v = redis.decrby("mykey", 3);
376     printf("redis < DECRBY mykey 3\n");
377     printf("redis > %lld\n", v);
378
379     printf("=====\n");
380 }
381
382 {
383     printf("INCR 命令\n\n");
384
385     redis.flushdb();
386

```

```

387     long long e = redis.exists("mykey");
388     printf("redis < EXISTS mykey\n");
389     printf("redis > %lld\n", e);
390
391     long long v = redis.incr("mykey");
392     printf("redis < INCR mykey\n");
393     printf("redis > %lld\n", v);
394
395     bool b = redis.set("mykey", "10");
396     printf("redis < SET mykey \"10\"\n");
397     printf("redis > %s\n", b ? "OK": "(nil)");
398
399     v = redis.incr("mykey");
400     printf("redis < INCR mykey\n");
401     printf("redis > %lld\n", v);
402
403     b = redis.set("mykey", "234293482390480948029348230948");
404     printf("redis < SET mykey \"234293482390480948029348230948\"\n");
405     printf("redis > %s\n", b ? "OK": "(nil)");
406
407     try {
408         v = redis.incr("mykey");
409         printf("redis < INCR mykey\n");
410         printf("redis > %lld\n", v);
411     } catch (const sw::redis::ReplyError& exc) {
412         printf("redis > %s\n", exc.what());
413     }
414
415     b = redis.set("mykey", "not a number");
416     printf("redis < SET mykey \"not a number\"\n");
417     printf("redis > %s\n", b ? "OK": "(nil)");
418
419     try {
420         v = redis.incr("mykey");
421         printf("redis < INCR mykey\n");
422         printf("redis > %lld\n", v);
423     } catch (const sw::redis::ReplyError& exc) {
424         printf("redis > %s\n", exc.what());
425     }
426
427     printf("=====\n");
428 }
429
430 {
431     printf("INCRBY 命令\n\n");
432
433     redis.flushdb();

```

```

434
435     long long e = redis.exists("mykey");
436     printf("redis < EXISTS mykey\n");
437     printf("redis > %lld\n", e);
438
439     long long v = redis.incrby("mykey", 3);
440     printf("redis < INCRBY mykey 3\n");
441     printf("redis > %lld\n", v);
442
443     bool b = redis.set("mykey", "10");
444     printf("redis < SET mykey \"10\"\n");
445     printf("redis > %s\n", b ? "OK": "(nil)");
446
447     v = redis.incrby("mykey", 3);
448     printf("redis < INCRBY mykey 3\n");
449     printf("redis > %lld\n", v);
450
451     printf("=====\n");
452 }
453
454 {
455     printf("INCRBYFLOAT 命令\n\n");
456
457     redis.flushdb();
458
459     bool b = redis.set("mykey", "10.50");
460     printf("redis < SET mykey \"10.50\"\n");
461     printf("redis > %s\n", b ? "OK": "(nil)");
462
463     double v = redis.incrbyfloat("mykey", 0.1);
464     printf("redis < INCRBYFLOAT mykey 0.1\n");
465     printf("redis > \"%.2lf\"\n", v);
466
467     v = redis.incrbyfloat("mykey", -5);
468     printf("redis < INCRBYFLOAT mykey -5\n");
469     printf("redis > \"%.2lf\"\n", v);
470
471     b = redis.set("mykey", "5.0e3");
472     printf("redis < SET mykey 5.0e3\n");
473     printf("redis > %s\n", b ? "OK": "(nil)");
474
475     v = redis.incrbyfloat("mykey", 2.0e2);
476     printf("redis < INCRBYFLOAT mykey 2.0e2\n");
477     printf("redis > \"%.0lf\"\n", v);
478
479     printf("=====\n");
480 }

```

```

481
482     {
483         printf("MGET 命令\n\n");
484
485         redis.flushdb();
486
487         bool b = redis.set("key1", "Hello");
488         printf("redis < SET key1 \"Hello\"\n");
489         printf("redis > %s\n", b ? "OK": "(nil)");
490
491         b = redis.set("key2", "World");
492         printf("redis < SET key2 \"World\"\n");
493         printf("redis > %s\n", b ? "OK": "(nil)");
494
495         std::vector<sw::redis::OptionalString> values;
496         std::insert_iterator<std::vector<sw::redis::OptionalString>> ins =
std::inserter(values, values.begin());
497         redis.mget({"key1", "key2", "nonexisting"}, ins);
498
499         printf("redis < MSET key1 key2 nonexisting\n");
500         int n = 1;
501         for (std::vector<sw::redis::OptionalString>::iterator it =
values.begin(); it != values.end(); ++it) {
502             if (bool(*it)) {
503                 printf("redis > %d) %s\n", n++, (*it)->c_str());
504             } else {
505                 printf("redis > %d) (nil)\n", n++);
506             }
507         }
508
509         printf("=====\n");
510     }
511
512     {
513         printf("MSET 命令\n\n");
514
515         redis.flushdb();
516
517         std::vector<std::pair<std::string, std::string>> kvs = {
518             {"key1", "Hello"},
519             {"key2", "World"}
520         };
521         redis.mset(kvs.begin(), kvs.end());
522         printf("redis < MSET key1 \"Hello\" key2 \"World\"\n");
523         printf("redis > OK\n");
524
525         sw::redis::OptionalString v = redis.get("key1");

```

```

526     printf("redis < GET key1\n");
527     if (bool(v)) {
528         printf("redis > \"%s\"\n", v->c_str());
529     } else {
530         printf("redis > (nil)\n");
531     }
532
533     v = redis.get("key2");
534     printf("redis < GET key2\n");
535     if (bool(v)) {
536         printf("redis > \"%s\"\n", v->c_str());
537     } else {
538         printf("redis > (nil)\n");
539     }
540
541     printf("=====\n");
542 }
543
544 {
545     printf("MSETNX 命令\n\n");
546
547     redis.flushdb();
548
549     std::vector<std::pair<std::string, std::string>> kvs1 = {
550         {"key1", "Hello"},
551         {"key2", "there"}
552     };
553     bool b = redis.msetnx(kvs1.begin(), kvs1.end());
554     printf("redis < MSETNX key1 \"Hello\" key2 \"there\"\n");
555     printf("redis > %d\n", b ? 1 : 0);
556
557     std::vector<std::pair<std::string, std::string>> kvs2 = {
558         {"key2", "new"},
559         {"key3", "world"}
560     };
561     b = redis.msetnx(kvs1.begin(), kvs1.end());
562     printf("redis < MSETNX key2 \"new\" key3 \"world\"\n");
563     printf("redis > %d\n", b ? 1 : 0);
564
565     std::vector<sw::redis::OptionalString> values;
566     std::insert_iterator<std::vector<sw::redis::OptionalString>> ins =
std::inserter(values, values.begin());
567     redis.mget({"key1", "key2", "key3"}, ins);
568
569     printf("redis < MSET key1 key2 key3\n");
570     int n = 1;

```

```

571     for (std::vector<sw::redis::OptionalString>::iterator it =
values.begin(); it != values.end(); ++it) {
572         if (bool(*it)) {
573             printf("redis > %d) %s\n", n++, (*it)->c_str());
574         } else {
575             printf("redis > %d) (nil)\n", n++);
576         }
577     }
578
579     printf("=====\n");
580 }
581
582 {
583     printf("SETRANGE 命令\n\n");
584
585     redis.flushdb();
586
587     bool b = redis.set("key1", "Hello World");
588     printf("redis < SET key1 \"Hello World\"\n");
589     printf("redis > %s\n", b ? "OK" : "(nil)");
590
591     long long c = redis.setrange("key1", 6, "Redis");
592     printf("redis < SETRANGE key1 6 \"Redis\"\n");
593     printf("redis > %lld\n", c);
594
595     sw::redis::OptionalString v = redis.get("key1");
596     printf("redis < GET key1\n");
597     if (bool(v)) {
598         printf("redis > \"%s\"\n", v->c_str());
599     } else {
600         printf("redis > (nil)\n");
601     }
602
603     printf("=====\n");
604 }
605
606 {
607     printf("SETRANGE 命令\n\n");
608
609     redis.flushdb();
610
611     bool b = redis.set("mykey", "Hello World");
612     printf("redis < SET mykey \"Hello World\"\n");
613     printf("redis > %s\n", b ? "OK" : "(nil)");
614
615     long long c = redis.strlen("mykey");
616     printf("redis < STRLEN mykey\n");

```



```

617         printf("redis > %lld\n", c);
618
619         c = redis.strlen("nonexisting");
620         printf("redis < STRLEN nonexisting\n");
621         printf("redis > %lld\n", c);
622
623         printf("=====\n");
624     }
625 }
626
627 int main() {
628     sw::redis::Redis redis("tcp://127.0.0.1:6379");
629
630     // String commands
631     testStringCommands(redis);
632
633     return 0;
634 }

```

string 测试结果

```

1 $ make string
2 g++ -std=c++17 -o string src/string.cc /usr/local/lib64/libredis++.a
  /usr/local/lib/libhiredis.a -pthread
3 $ ./string
4 String 系列命令:
5 GET 命令
6
7 redis < GET nonexisting
8 redis > (nil)
9 redis < SET mykey "Hello"
10 redis > OK
11 redis < GET mykey
12 redis > "Hello"
13 redis < DEL mykey
14 redis > 1
15 redis < EXISTS mykey
16 redis > 0
17 redis < HSET mykey name Bob
18 redis > 1
19 redis < GET mykey
20 redis > WRONGTYPE Operation against a key holding the wrong kind of value
21 =====
22 APPEND 命令
23

```

```
24 redis < EXISTS mykey
25 redis > 0
26 redis < APPEND mykey "Hello"
27 redis > 5
28 redis < GET mykey
29 redis > "Hello"
30 redis < APPEND mykey " World"
31 redis > 11
32 redis < GET mykey
33 redis > "Hello World"
34 =====
35 GETRANGE 命令
36
37 redis < SET mykey "This is a string"
38 redis > OK
39 redis < GETRANGE mykey 0 3
40 redis > This
41 redis < GETRANGE mykey -3 -1
42 redis > ing
43 redis < GETRANGE mykey 0 -1
44 redis > This is a string
45 redis < GETRANGE mykey 10 100
46 redis > string
47 =====
48 SETNX 命令
49
50 redis < SETNX mykey "Hello"
51 redis > 1
52 redis < SETNX mykey "World"
53 redis > 0
54 redis < GET mykey
55 redis > "Hello"
56 =====
57 DECR 命令
58
59 redis < EXISTS mykey
60 redis > 0
61 redis < DECR mykey
62 redis > -1
63 redis < SET mykey "10"
64 redis > OK
65 redis < DECR mykey
66 redis > 9
67 redis < SET mykey "234293482390480948029348230948"
68 redis > OK
69 redis > ERR value is not an integer or out of range
70 redis < SET mykey "not a number"
```

```
71 redis > OK
72 redis > ERR value is not an integer or out of range
73 =====
74 DECRBY 命令
75
76 redis < EXISTS mykey
77 redis > 0
78 redis < DECRBY mykey 3
79 redis > -3
80 redis < SET mykey "10"
81 redis > OK
82 redis < DECRBY mykey 3
83 redis > 7
84 =====
85 INCR 命令
86
87 redis < EXISTS mykey
88 redis > 0
89 redis < INCR mykey
90 redis > 1
91 redis < SET mykey "10"
92 redis > OK
93 redis < INCR mykey
94 redis > 11
95 redis < SET mykey "234293482390480948029348230948"
96 redis > OK
97 redis > ERR value is not an integer or out of range
98 redis < SET mykey "not a number"
99 redis > OK
100 redis > ERR value is not an integer or out of range
101 =====
102 INCRBY 命令
103
104 redis < EXISTS mykey
105 redis > 0
106 redis < INCRBY mykey 3
107 redis > 3
108 redis < SET mykey "10"
109 redis > OK
110 redis < INCRBY mykey 3
111 redis > 13
112 =====
113 INCRBYFLOAT 命令
114
115 redis < SET mykey "10.50"
116 redis > OK
117 redis < INCRBYFLOAT mykey 0.1
```

```
118 redis > "10.60"
119 redis < INCRBYFLOAT mykey -5
120 redis > "5.60"
121 redis < SET mykey 5.0e3
122 redis > OK
123 redis < INCRBYFLOAT mykey 2.0e2
124 redis > "5200"
125 =====
126 MGET 命令
127
128 redis < SET key1 "Hello"
129 redis > OK
130 redis < SET key2 "World"
131 redis > OK
132 redis < MSET key1 key2 nonexisting
133 redis > 1) Hello
134 redis > 2) World
135 redis > 3) (nil)
136 =====
137 MSET 命令
138
139 redis < MSET key1 "Hello" key2 "World"
140 redis > OK
141 redis < GET key1
142 [root@VM-52-199-opencloudos use-redis-cxx]# make
143 g++ -std=c++17 -o string src/string.cc /usr/local/lib64/libredis++.a
    /usr/local/lib/libhiredis.a -pthread
144 [root@VM-52-199-opencloudos use-redis-cxx]# ./string
145 String 系列命令:
146 =====
147 SET 命令
148
149 redis < EXISTS mykey
150 redis > 0
151 redis < SET mykey "Hello"
152 redis > OK
153 redis < GET mykey
154 redis > "Hello"
155 redis < SET mykey "World" NX
156 redis > (nil)
157 redis < DEL mykey
158 redis > 1
159 redis < EXISTS mykey
160 redis > 0
161 redis < SET mykey "World" XX
162 redis > (nil)
163 redis < GET mykey
```

```
164 redis > (nil)
165 redis < SET mykey "World" NX
166 redis > OK
167 redis < GET mykey
168 redis > "World"
169 redis < SET mykey "Will expire in 10s" EX 10
170 redis > OK
171 redis < GET mykey
172 redis > "Will expire in 10s"
173 Waiting 10s
174 Waiting 9s
175 Waiting 8s
176 Waiting 7s
177 Waiting 6s
178 Waiting 5s
179 Waiting 4s
180 Waiting 3s
181 Waiting 2s
182 Waiting 1s
183 redis < GET mykey
184 redis > (nil)
185 =====
186 GET 命令
187
188 redis < GET nonexisting
189 redis > (nil)
190 redis < SET mykey "Hello"
191 redis > OK
192 redis < GET mykey
193 redis > "Hello"
194 redis < DEL mykey
195 redis > 1
196 redis < EXISTS mykey
197 redis > 0
198 redis < HSET mykey name Bob
199 redis > 1
200 redis < GET mykey
201 redis > WRONGTYPE Operation against a key holding the wrong kind of value
202 =====
203 APPEND 命令
204
205 redis < EXISTS mykey
206 redis > 0
207 redis < APPEND mykey "Hello"
208 redis > 5
209 redis < GET mykey
210 redis > "Hello"
```

```
211 redis < APPEND mykey " World"
212 redis > 11
213 redis < GET mykey
214 redis > "Hello World"
215 =====
216 GETRANGE 命令
217
218 redis < SET mykey "This is a string"
219 redis > OK
220 redis < GETRANGE mykey 0 3
221 redis > This
222 redis < GETRANGE mykey -3 -1
223 redis > ing
224 redis < GETRANGE mykey 0 -1
225 redis > This is a string
226 redis < GETRANGE mykey 10 100
227 redis > string
228 =====
229 SETEX 命令
230
231 redis < SETEX mykey 10 "Hello"
232 redis > OK
233 redis < TTL mykey
234 redis > 10
235 redis < GET mykey
236 redis > "Hello"
237 Waiting 10s
238 Waiting 9s
239 Waiting 8s
240 Waiting 7s
241 Waiting 6s
242 Waiting 5s
243 Waiting 4s
244 Waiting 3s
245 Waiting 2s
246 Waiting 1s
247 redis < GET mykey
248 redis > (nil)
249 =====
250 PSETEX 命令
251
252 redis < PSETEX mykey 10000 "Hello"
253 redis > OK
254 redis < PTTL mykey
255 redis > 10000
256 redis < GET mykey
257 redis > "Hello"
```

```
258 Waiting 10s
259 Waiting 9s
260 Waiting 8s
261 Waiting 7s
262 Waiting 6s
263 Waiting 5s
264 Waiting 4s
265 Waiting 3s
266 Waiting 2s
267 Waiting 1s
268 redis < GET mykey
269 redis > (nil)
270 =====
271 SETNX 命令
272
273 redis < SETNX mykey "Hello"
274 redis > 1
275 redis < SETNX mykey "World"
276 redis > 0
277 redis < GET mykey
278 redis > "Hello"
279 =====
280 DECR 命令
281
282 redis < EXISTS mykey
283 redis > 0
284 redis < DECR mykey
285 redis > -1
286 redis < SET mykey "10"
287 redis > OK
288 redis < DECR mykey
289 redis > 9
290 redis < SET mykey "234293482390480948029348230948"
291 redis > OK
292 redis > ERR value is not an integer or out of range
293 redis < SET mykey "not a number"
294 redis > OK
295 redis > ERR value is not an integer or out of range
296 =====
297 DECRBY 命令
298
299 redis < EXISTS mykey
300 redis > 0
301 redis < DECRBY mykey 3
302 redis > -3
303 redis < SET mykey "10"
304 redis > OK
```

```
305 redis < DECRBY mykey 3
306 redis > 7
307 =====
308 INCR 命令
309
310 redis < EXISTS mykey
311 redis > 0
312 redis < INCR mykey
313 redis > 1
314 redis < SET mykey "10"
315 redis > OK
316 redis < INCR mykey
317 redis > 11
318 redis < SET mykey "234293482390480948029348230948"
319 redis > OK
320 redis > ERR value is not an integer or out of range
321 redis < SET mykey "not a number"
322 redis > OK
323 redis > ERR value is not an integer or out of range
324 =====
325 INCRBY 命令
326
327 redis < EXISTS mykey
328 redis > 0
329 redis < INCRBY mykey 3
330 redis > 3
331 redis < SET mykey "10"
332 redis > OK
333 redis < INCRBY mykey 3
334 redis > 13
335 =====
336 INCRBYFLOAT 命令
337
338 redis < SET mykey "10.50"
339 redis > OK
340 redis < INCRBYFLOAT mykey 0.1
341 redis > "10.60"
342 redis < INCRBYFLOAT mykey -5
343 redis > "5.60"
344 redis < SET mykey 5.0e3
345 redis > OK
346 redis < INCRBYFLOAT mykey 2.0e2
347 redis > "5200"
348 =====
349 MGET 命令
350
351 redis < SET key1 "Hello"
```



```
352 redis > OK
353 redis < SET key2 "World"
354 redis > OK
355 redis < MSET key1 key2 nonexisting
356 redis > 1) Hello
357 redis > 2) World
358 redis > 3) (nil)
359 =====
360 MSET 命令
361
362 redis < MSET key1 "Hello" key2 "World"
363 redis > OK
364 redis < GET key1
365 redis > "Hello"
366 redis < GET key2
367 redis > "World"
368 =====
369 MSETNX 命令
370
371 redis < MSETNX key1 "Hello" key2 "there"
372 redis > 1
373 redis < MSETNX key2 "new" key3 "world"
374 redis > 0
375 redis < MSET key1 key2 key3
376 redis > 1) Hello
377 redis > 2) there
378 redis > 3) (nil)
379 =====
380 SETRANGE 命令
381
382 redis < SET key1 "Hello World"
383 redis > OK
384 redis < SETRANGE key1 6 "Redis"
385 redis > 11
386 redis < GET key1
387 redis > "Hello Redis"
388 =====
389 SETRANGE 命令
390
391 redis < SET mykey "Hello World"
392 redis > OK
393 redis < STRLEN mykey
394 redis > 11
395 redis < STRLEN nonexisting
396 redis > 0
397 =====
```

```

1  #include <sw/redis++/redis++.h>
2  #include <cstdio>
3  #include <string>
4  #include <chrono>
5  #include <thread>
6  #include <iterator>
7
8
9  void testListCommands(sw::redis::Redis& redis) {
10     printf("List 系列命令:\n");
11     printf("=====\n");
12     {
13         printf("LPUSH 命令\n\n");
14
15         redis.flushdb();
16
17         long long c = redis.lpush("mylist", "world");
18         printf("redis < LPUSH mylist \"world\"\n");
19         printf("redis > %lld\n", c);
20
21         c = redis.lpush("mylist", "hello");
22         printf("redis < LPUSH mylist \"hello\"\n");
23         printf("redis > %lld\n", c);
24
25         c = redis.lpush("mylist", {"Redis", "MySQL", "MongoDB"});
26         printf("redis < LPUSH mylist \"Redis\" \"MySQL\" \"MongoDB\"\n");
27         printf("redis > %lld\n", c);
28
29         std::vector<std::string> elements;
30         std::back_inserter_iterator<std::vector<std::string>> ins =
31         std::back_inserter(elements);
32         redis.lrange("mylist", 0, -1, ins);
33
34         printf("redis < LRANGE mylist 0 -1\n");
35         int n = 1;
36         for (auto it = elements.begin(); it != elements.end(); ++it) {
37             printf("redis > %d) %s\n", n++, it->c_str());
38         }
39
40         printf("=====\n");
41     }
42     {
43         printf("LPUSHX 命令\n\n");

```

```

44
45     redis.flushdb();
46
47     long long c = redis.lpush("mylist", "World");
48     printf("redis < LPUSH mylist World\n");
49     printf("redis > %lld\n", c);
50
51     c = redis.lpushx("mylist", "Hello");
52     printf("redis < LPUSHX mylist Hello\n");
53     printf("redis > %lld\n", c);
54
55     c = redis.lpushx("myotherlist", "Hello");
56     printf("redis < LPUSHX myotherlist Hello\n");
57     printf("redis > %lld\n", c);
58
59     std::vector<std::string> elements;
60     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
61     redis.lrange("mylist", 0, -1, ins);
62
63     printf("redis < LRANGE mylist 0 -1\n");
64     int n = 1;
65     for (auto it = elements.begin(); it != elements.end(); ++it) {
66         printf("redis > %d) %s\n", n++, it->c_str());
67     }
68
69     elements.clear();
70     ins = std::back_inserter(elements);
71     redis.lrange("myotherlist", 0, -1, ins);
72
73     printf("redis < LRANGE myotherlist 0 -1\n");
74     if (elements.size() == 0) {
75         printf("redis < (empty array)\n");
76     } else {
77         int n = 1;
78         for (auto it = elements.begin(); it != elements.end(); ++it) {
79             printf("redis > %d) %s\n", n++, it->c_str());
80         }
81     }
82
83     printf("=====\n");
84 }
85
86 {
87     printf("RPUSH 命令\n\n");
88
89     redis.flushdb();

```

```

90
91     long long c = redis.rpush("mylist", "world");
92     printf("redis < RPush mylist \"world\"\n");
93     printf("redis > %lld\n", c);
94
95     c = redis.rpush("mylist", "hello");
96     printf("redis < RPush mylist \"hello\"\n");
97     printf("redis > %lld\n", c);
98
99     c = redis.rpush("mylist", {"Redis", "MySQL", "MongoDB"});
100    printf("redis < RPush mylist \"Redis\" \"MySQL\" \"MongoDB\"\n");
101    printf("redis > %lld\n", c);
102
103    std::vector<std::string> elements;
104    std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
105    redis.lrange("mylist", 0, -1, ins);
106
107    printf("redis < LRANGE mylist 0 -1\n");
108    int n = 1;
109    for (auto it = elements.begin(); it != elements.end(); ++it) {
110        printf("redis > %d) %s\n", n++, it->c_str());
111    }
112
113    printf("=====\n");
114 }
115
116 {
117     printf("RPushX 命令\n\n");
118
119     redis.flushdb();
120
121     long long c = redis.rpush("mylist", "World");
122     printf("redis < RPush mylist World\n");
123     printf("redis > %lld\n", c);
124
125     c = redis.rpushx("mylist", "Hello");
126     printf("redis < RPushX mylist Hello\n");
127     printf("redis > %lld\n", c);
128
129     c = redis.rpushx("myotherlist", "Hello");
130     printf("redis < RPushX myotherlist Hello\n");
131     printf("redis > %lld\n", c);
132
133     std::vector<std::string> elements;
134     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);

```

```

135     redis.lrange("mylist", 0, -1, ins);
136
137     printf("redis < LRANGE mylist 0 -1\n");
138     int n = 1;
139     for (auto it = elements.begin(); it != elements.end(); ++it) {
140         printf("redis > %d) %s\n", n++, it->c_str());
141     }
142
143     elements.clear();
144     ins = std::back_inserter(elements);
145     redis.lrange("myotherlist", 0, -1, ins);
146
147     printf("redis < LRANGE myotherlist 0 -1\n");
148     if (elements.size() == 0) {
149         printf("redis < (empty array)\n");
150     } else {
151         int n = 1;
152         for (auto it = elements.begin(); it != elements.end(); ++it) {
153             printf("redis > %d) %s\n", n++, it->c_str());
154         }
155     }
156
157     printf("=====\n");
158 }
159
160 {
161     printf("LRANGE 命令\n\n");
162
163     redis.flushdb();
164
165     long long c = redis.rpush("mylist", "one");
166     printf("redis < RPush mylist one\n");
167     printf("redis > %lld\n", c);
168
169     c = redis.rpush("mylist", "two");
170     printf("redis < RPush mylist two\n");
171     printf("redis > %lld\n", c);
172
173     c = redis.rpush("mylist", "three");
174     printf("redis < RPush mylist three\n");
175     printf("redis > %lld\n", c);
176
177     std::vector<std::string> elements;
178     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
179     redis.lrange("mylist", 0, 0, ins);
180

```

```
181     printf("redis < LRANGE mylist 0 0\n");
182     if (elements.size() == 0) {
183         printf("redis > (empty array)\n");
184     } else {
185         int n = 1;
186         for (auto it = elements.begin(); it != elements.end(); ++it) {
187             printf("redis > %d) %s\n", n++, it->c_str());
188         }
189     }
190
191     elements.clear();
192     ins = std::back_inserter(elements);
193     redis.lrange("mylist", -3, 2, ins);
194
195     printf("redis < LRANGE mylist -3 2\n");
196     if (elements.size() == 0) {
197         printf("redis > (empty array)\n");
198     } else {
199         int n = 1;
200         for (auto it = elements.begin(); it != elements.end(); ++it) {
201             printf("redis > %d) %s\n", n++, it->c_str());
202         }
203     }
204
205     elements.clear();
206     ins = std::back_inserter(elements);
207     redis.lrange("mylist", -100, 100, ins);
208
209     printf("redis < LRANGE mylist -100 100\n");
210     if (elements.size() == 0) {
211         printf("redis > (empty array)\n");
212     } else {
213         int n = 1;
214         for (auto it = elements.begin(); it != elements.end(); ++it) {
215             printf("redis > %d) %s\n", n++, it->c_str());
216         }
217     }
218
219     elements.clear();
220     ins = std::back_inserter(elements);
221     redis.lrange("mylist", 5, 10, ins);
222
223     printf("redis < LRANGE mylist 5 10\n");
224     if (elements.size() == 0) {
225         printf("redis > (empty array)\n");
226     } else {
227         int n = 1;
```

```

228         for (auto it = elements.begin(); it != elements.end(); ++it) {
229             printf("redis > %d) %s\n", n++, it->c_str());
230         }
231     }
232
233     printf("=====\n");
234 }
235
236 {
237     printf("LPOP 命令\n\n");
238
239     redis.flushdb();
240
241     long long c = redis.rpush("mylist", {"one", "two", "three", "four",
"five"});
242     printf("redis < RPush mylist one two three four five\n");
243     printf("redis > %lld\n", c);
244
245     sw::redis::OptionalString v = redis.lpop("mylist");
246     printf("redis < LPOP mylist\n");
247     if (v) {
248         printf("redis > %s\n", v->c_str());
249     } else {
250         printf("redis > (nil)\n");
251     }
252
253     v = redis.lpop("mylist");
254     printf("redis < LPOP mylist\n");
255     if (v) {
256         printf("redis > %s\n", v->c_str());
257     } else {
258         printf("redis > (nil)\n");
259     }
260
261     v = redis.lpop("mylist");
262     printf("redis < LPOP mylist\n");
263     if (v) {
264         printf("redis > %s\n", v->c_str());
265     } else {
266         printf("redis > (nil)\n");
267     }
268
269     std::vector<std::string> elements;
270     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
271     redis.lrange("mylist", 0, -1, ins);
272

```

```

273     printf("redis < LRANGE mylist 0 -1\n");
274     if (elements.size() == 0) {
275         printf("redis > (empty array)\n");
276     } else {
277         int n = 1;
278         for (auto it = elements.begin(); it != elements.end(); ++it) {
279             printf("redis > %d) %s\n", n++, it->c_str());
280         }
281     }
282
283     printf("=====\n");
284 }
285
286 {
287     printf("BLPOP 命令\n\n");
288
289     redis.flushdb();
290
291     long long e = redis.exists({"list1", "list2"});
292     printf("redis < EXISTS list1 list2\n");
293     printf("redis > %lld\n", e);
294
295     long long c = redis.rpush("list1", {"a", "b", "c"});
296     printf("redis < RPush list1 a b c\n");
297     printf("redis > %lld\n", c);
298
299     sw::redis::OptionalStringPair p = redis.blpop({"list1", "list2"},
std::chrono::seconds(0));
300     printf("redis < BLPOP list1 list2 0\n");
301     if (p) {
302         printf("redis > 1) %s\n", p->first.c_str());
303         printf("redis > 2) %s\n", p->second.c_str());
304     } else {
305         printf("redis > (nil)\n");
306     }
307
308     p = redis.blpop({"list1", "list2"}, std::chrono::seconds(0));
309     printf("redis < BLPOP list1 list2 0\n");
310     if (p) {
311         printf("redis > 1) %s\n", p->first.c_str());
312         printf("redis > 2) %s\n", p->second.c_str());
313     } else {
314         printf("redis > (nil)\n");
315     }
316
317     p = redis.blpop({"list1", "list2"}, std::chrono::seconds(0));
318     printf("redis < BLPOP list1 list2 0\n");

```



```

319     if (p) {
320         printf("redis > 1) %s\n", p->first.c_str());
321         printf("redis > 2) %s\n", p->second.c_str());
322     } else {
323         printf("redis > (nil)\n");
324     }
325
326     p = redis.blpop({"list1", "list2"}, std::chrono::seconds(5));
327     printf("redis < BLOP list1 list2 5\n");
328     if (p) {
329         printf("redis > 1) %s\n", p->first.c_str());
330         printf("redis > 2) %s\n", p->second.c_str());
331     } else {
332         printf("redis > (nil)\n");
333     }
334
335     printf("=====\n");
336 }
337
338 {
339     printf("RPOP 命令\n\n");
340
341     redis.flushdb();
342
343     long long c = redis.rpush("mylist", {"one", "two", "three", "four",
"five"});
344     printf("redis < Rpush mylist one two three four five\n");
345     printf("redis > %lld\n", c);
346
347     sw::redis::OptionalString v = redis.rpop("mylist");
348     printf("redis < RPOP mylist\n");
349     if (v) {
350         printf("redis > %s\n", v->c_str());
351     } else {
352         printf("redis > (nil)\n");
353     }
354
355     v = redis.rpop("mylist");
356     printf("redis < RPOP mylist\n");
357     if (v) {
358         printf("redis > %s\n", v->c_str());
359     } else {
360         printf("redis > (nil)\n");
361     }
362
363     v = redis.rpop("mylist");
364     printf("redis < RPOP mylist\n");

```

```

365     if (v) {
366         printf("redis > %s\n", v->c_str());
367     } else {
368         printf("redis > (nil)\n");
369     }
370
371     std::vector<std::string> elements;
372     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
373     redis.lrange("mylist", 0, -1, ins);
374
375     printf("redis < LRANGE mylist 0 -1\n");
376     if (elements.size() == 0) {
377         printf("redis > (empty array)\n");
378     } else {
379         int n = 1;
380         for (auto it = elements.begin(); it != elements.end(); ++it) {
381             printf("redis > %d) %s\n", n++, it->c_str());
382         }
383     }
384
385     printf("=====\n");
386 }
387
388 {
389     printf("BRPOP 命令\n\n");
390
391     redis.flushdb();
392
393     long long e = redis.exists({"list1", "list2"});
394     printf("redis < EXISTS list1 list2\n");
395     printf("redis > %lld\n", e);
396
397     long long c = redis.rpush("list1", {"a", "b", "c"});
398     printf("redis < RPUSH list1 a b c\n");
399     printf("redis > %lld\n", c);
400
401     sw::redis::OptionalStringPair p = redis.brpop({"list1", "list2"},
std::chrono::seconds(0));
402     printf("redis < BRPOP list1 list2 0\n");
403     if (p) {
404         printf("redis > 1) %s\n", p->first.c_str());
405         printf("redis > 2) %s\n", p->second.c_str());
406     } else {
407         printf("redis > (nil)\n");
408     }
409

```

```

410     p = redis.brpop({"list1", "list2"}, std::chrono::seconds(0));
411     printf("redis < BRPOP list1 list2 0\n");
412     if (p) {
413         printf("redis > 1) %s\n", p->first.c_str());
414         printf("redis > 2) %s\n", p->second.c_str());
415     } else {
416         printf("redis > (nil)\n");
417     }
418
419     p = redis.brpop({"list1", "list2"}, std::chrono::seconds(0));
420     printf("redis < BRPOP list1 list2 0\n");
421     if (p) {
422         printf("redis > 1) %s\n", p->first.c_str());
423         printf("redis > 2) %s\n", p->second.c_str());
424     } else {
425         printf("redis > (nil)\n");
426     }
427
428     p = redis.brpop({"list1", "list2"}, std::chrono::seconds(5));
429     printf("redis < BRPOP list1 list2 5\n");
430     if (p) {
431         printf("redis > 1) %s\n", p->first.c_str());
432         printf("redis > 2) %s\n", p->second.c_str());
433     } else {
434         printf("redis > (nil)\n");
435     }
436
437     printf("=====\n");
438 }
439
440 {
441     printf("LINDEX 命令\n\n");
442
443     redis.flushdb();
444
445     long long c = redis.lpush("mylist", "World");
446     printf("redis < LPUSH mylist World\n");
447     printf("redis > %lld\n", c);
448
449     c = redis.lpush("mylist", "Hello");
450     printf("redis < LPUSH mylist Hello\n");
451     printf("redis > %lld\n", c);
452
453     sw::redis::OptionalString v = redis.lindex("mylist", 0);
454     printf("redis < LINDEX mylist 0\n");
455     if (v) {
456         printf("redis > %s\n", v->c_str());

```

```

457     } else {
458         printf("redis > (nil)\n");
459     }
460
461     v = redis.lindex("mylist", -1);
462     printf("redis < LINDEX mylist -1\n");
463     if (v) {
464         printf("redis > %s\n", v->c_str());
465     } else {
466         printf("redis > (nil)\n");
467     }
468
469     v = redis.lindex("mylist", 3);
470     printf("redis < LINDEX mylist 3\n");
471     if (v) {
472         printf("redis > %s\n", v->c_str());
473     } else {
474         printf("redis > (nil)\n");
475     }
476
477     printf("=====\n");
478 }
479
480 {
481     printf("LINSERT 命令\n\n");
482
483     redis.flushdb();
484
485     long long c = redis.rpush("mylist", "Hello");
486     printf("redis < RPUSH mylist Hello\n");
487     printf("redis > %lld\n", c);
488
489     c = redis.rpush("mylist", "World");
490     printf("redis < RPUSH mylist World\n");
491     printf("redis > %lld\n", c);
492
493     c = redis.linsert("mylist", sw::redis::InsertPosition::BEFORE,
"World", "There");
494     printf("redis < LINSERT mylist BEFORE World There\n");
495     printf("redis > %lld\n", c);
496
497     std::vector<std::string> elements;
498     std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
499     redis.lrange("mylist", 0, -1, ins);
500
501     printf("redis < LRANGE mylist 0 -1\n");

```

```

502     int n = 1;
503     for (auto it = elements.begin(); it != elements.end(); ++it) {
504         printf("redis > %d) %s\n", n++, it->c_str());
505     }
506
507     printf("=====\n");
508 }
509
510 {
511     printf("LLEN 命令\n\n");
512
513     redis.flushdb();
514
515     long long c = redis.rpush("mylist", "Hello");
516     printf("redis < Rpush mylist Hello\n");
517     printf("redis > %lld\n", c);
518
519     c = redis.rpush("mylist", "World");
520     printf("redis < Rpush mylist World\n");
521     printf("redis > %lld\n", c);
522
523     c = redis.llen("mylist");
524     printf("redis < LLEN mylist\n");
525     printf("redis > %lld\n", c);
526
527     printf("=====\n");
528 }
529 }
530
531 int main() {
532     sw::redis::Redis redis("tcp://127.0.0.1:6379");
533
534     // List commands
535     testListCommands(redis);
536
537     return 0;
538 }

```

list 测试结果

```

1 $ make list
2 g++ -std=c++17 -o list src/list.cc /usr/local/lib64/libredis++.a
   /usr/local/lib/libhiredis.a -pthread
3 $ ./list
4 List 系列命令:

```

```
5 =====
6 LPUSH 命令
7
8 redis < LPUSH mylist "world"
9 redis > 1
10 redis < LPUSH mylist "hello"
11 redis > 2
12 redis < LPUSH mylist "Redis" "MySQL" "MongoDB"
13 redis > 5
14 redis < LRANGE mylist 0 -1
15 redis > 1) MongoDB
16 redis > 2) MySQL
17 redis > 3) Redis
18 redis > 4) hello
19 redis > 5) world
20 =====
21 LPUSHX 命令
22
23 redis < LPUSH mylist World
24 redis > 1
25 redis < LPUSHX mylist Hello
26 redis > 2
27 redis < LPUSHX myotherlist Hello
28 redis > 0
29 redis < LRANGE mylist 0 -1
30 redis > 1) Hello
31 redis > 2) World
32 redis < LRANGE myotherlist 0 -1
33 redis < (empty array)
34 =====
35 RPUSH 命令
36
37 redis < RPUSH mylist "world"
38 redis > 1
39 redis < RPUSH mylist "hello"
40 redis > 2
41 redis < RPUSH mylist "Redis" "MySQL" "MongoDB"
42 redis > 5
43 redis < LRANGE mylist 0 -1
44 redis > 1) world
45 redis > 2) hello
46 redis > 3) Redis
47 redis > 4) MySQL
48 redis > 5) MongoDB
49 =====
50 RPUSHX 命令
51
```

```
52 redis < RPUSH mylist World
53 redis > 1
54 redis < RPUSHX mylist Hello
55 redis > 2
56 redis < RPUSHX myotherlist Hello
57 redis > 0
58 redis < LRANGE mylist 0 -1
59 redis > 1) World
60 redis > 2) Hello
61 redis < LRANGE myotherlist 0 -1
62 redis < (empty array)
63 =====
64 LRANGE 命令
65
66 redis < RPUSH mylist one
67 redis > 1
68 redis < RPUSH mylist two
69 redis > 2
70 redis < RPUSH mylist three
71 redis > 3
72 redis < LRANGE mylist 0 0
73 redis > 1) one
74 redis < LRANGE mylist -3 2
75 redis > 1) one
76 redis > 2) two
77 redis > 3) three
78 redis < LRANGE mylist -100 100
79 redis > 1) one
80 redis > 2) two
81 redis > 3) three
82 redis < LRANGE mylist 5 10
83 redis < (empty array)
84 =====
85 LPOP 命令
86
87 redis < RPUSH mylist one two three four five
88 redis > 5
89 redis < LPOP mylist
90 redis > one
91 redis < LPOP mylist
92 redis > two
93 redis < LPOP mylist
94 redis > three
95 redis < LRANGE mylist 0 -1
96 redis > 1) four
97 redis > 2) five
98 =====
```

```
99  BLPOP 命令
100
101 redis < EXISTS list1 list2
102 redis > 0
103 redis < RPUSH list1 a b c
104 redis > 3
105 redis < BLPOP list1 list2 0
106 redis > 1) list1
107 redis > 2) a
108 redis < BLPOP list1 list2 0
109 redis > 1) list1
110 redis > 2) b
111 redis < BLPOP list1 list2 0
112 redis > 1) list1
113 redis > 2) c
114 redis < BLPOP list1 list2 5
115 redis > (nil)
116 =====
117 RPOP 命令
118
119 redis < RPUSH mylist one two three four five
120 redis > 5
121 redis < RPOP mylist
122 redis > five
123 redis < RPOP mylist
124 redis > four
125 redis < RPOP mylist
126 redis > three
127 redis < LRange mylist 0 -1
128 redis > 1) one
129 redis > 2) two
130 =====
131 BRPOP 命令
132
133 redis < EXISTS list1 list2
134 redis > 0
135 redis < RPUSH list1 a b c
136 redis > 3
137 redis < BRPOP list1 list2 0
138 redis > 1) list1
139 redis > 2) c
140 redis < BRPOP list1 list2 0
141 redis > 1) list1
142 redis > 2) b
143 redis < BRPOP list1 list2 0
144 redis > 1) list1
145 redis > 2) a
```



```

146 redis < BRPOP list1 list2 5
147 redis > (nil)
148 =====
149 LINDEX 命令
150
151 redis < LPUSH mylist World
152 redis > 1
153 redis < LPUSH mylist Hello
154 redis > 2
155 redis < LINDEX mylist 0
156 redis > Hello
157 redis < LINDEX mylist -1
158 redis > World
159 redis < LINDEX mylist 3
160 redis > (nil)
161 =====
162 LINSERT 命令
163
164 redis < RPUSH mylist Hello
165 redis > 1
166 redis < RPUSH mylist World
167 redis > 2
168 redis < LINSERT mylist BEFORE World There
169 redis > 3
170 redis < LRANGE mylist 0 -1
171 redis > 1) Hello
172 redis > 2) There
173 redis > 3) World
174 =====
175 LLEN 命令
176
177 redis < RPUSH mylist Hello
178 redis > 1
179 redis < RPUSH mylist World
180 redis > 2
181 redis < LLEN mylist
182 redis > 2
183 =====

```

src/set.cc

```

1 #include <sw/redis++/redis++.h>
2 #include <cstdio>
3 #include <string>
4 #include <chrono>

```

```

5 #include <thread>
6 #include <iterator>
7 #include <unordered_set>
8
9
10 void testSetCommands(sw::redis::Redis& redis) {
11     printf("Set 系列命令:\n");
12     printf("=====\n");
13     {
14         printf("SADD 命令\n\n");
15         printf("SMEMBERS 命令\n\n");
16
17         redis.flushdb();
18
19         long long c = redis.sadd("myset", "Hello");
20         printf("redis < SADD myset Hello\n");
21         printf("redis > %lld\n", c);
22
23         c = redis.sadd("myset", "World");
24         printf("redis < SADD myset World\n");
25         printf("redis > %lld\n", c);
26
27         c = redis.sadd("myset", "World");
28         printf("redis < SADD myset World\n");
29         printf("redis > %lld\n", c);
30
31         std::unordered_set<std::string> elements;
32         redis.smembers("myset", std::inserter(elements, elements.begin()));
33         printf("redis < SMEMBERS myset\n");
34         if (elements.size() == 0) {
35             printf("redis < (empty array)\n");
36         } else {
37             int n = 1;
38             for (const std::string& e : elements) {
39                 printf("redis > %d) %s\n", n++, e.c_str());
40             }
41         }
42
43         printf("=====\n");
44     }
45
46     printf("=====\n");
47     {
48         printf("SISMEMBER 命令\n\n");
49         redis.flushdb();
50
51         long long c = redis.sadd("myset", "one");

```

```

52     printf("redis < SADD myset one\n");
53     printf("redis > %lld\n", c);
54
55     bool e = redis.sismember("myset", "one");
56     printf("redis < SISMEMBER myset one\n");
57     printf("redis > %d\n", e ? 1 : 0);
58
59     e = redis.sismember("myset", "two");
60     printf("redis < SISMEMBER myset two\n");
61     printf("redis > %d\n", e ? 1 : 0);
62
63     printf("=====\n");
64 }
65
66 printf("=====\n");
67 {
68     printf("SCARD 命令\n\n");
69     redis.flushdb();
70
71     long long c = redis.sadd("myset", "Hello");
72     printf("redis < SADD myset Hello\n");
73     printf("redis > %lld\n", c);
74
75     c = redis.sadd("myset", "World");
76     printf("redis < SADD myset World\n");
77     printf("redis > %lld\n", c);
78
79     c = redis.scard("myset");
80     printf("redis < SCARD myset\n");
81     printf("redis > %lld\n", c);
82
83     printf("=====\n");
84 }
85
86 printf("=====\n");
87 {
88     printf("SPOP 命令\n\n");
89     redis.flushdb();
90
91     long long c = redis.sadd("myset", "one");
92     printf("redis < SADD myset one\n");
93     printf("redis > %lld\n", c);
94
95     c = redis.sadd("myset", "two");
96     printf("redis < SADD myset two\n");
97     printf("redis > %lld\n", c);
98

```

```

99     c = redis.sadd("myset", "three");
100    printf("redis < SADD myset three\n");
101    printf("redis > %lld\n", c);
102
103    sw::redis::OptionalString v = redis.spop("myset");
104    printf("redis < SPOP myset\n");
105    if (v) {
106        printf("redis > %s\n", v->c_str());
107    } else {
108        printf("redis > (nil)\n");
109    }
110
111    std::unordered_set<std::string> elements;
112    redis.smembers("myset", std::inserter(elements, elements.begin()));
113    printf("redis < SMEMBERS myset\n");
114    if (elements.size() == 0) {
115        printf("redis < (empty array)\n");
116    } else {
117        int n = 1;
118        for (const std::string& e : elements) {
119            printf("redis > %d) %s\n", n++, e.c_str());
120        }
121    }
122
123    c = redis.sadd("myset", "four");
124    printf("redis < SADD myset four\n");
125    printf("redis > %lld\n", c);
126
127    c = redis.sadd("myset", "five");
128    printf("redis < SADD myset five\n");
129    printf("redis > %lld\n", c);
130
131    std::unordered_set<std::string> popped;
132    redis.spop("myset", 3, std::inserter(popped, popped.begin()));
133    printf("redis < SPOP myset 3\n");
134    if (popped.size() == 0) {
135        printf("redis < (empty array)\n");
136    } else {
137        int n = 1;
138        for (const std::string& e : popped) {
139            printf("redis > %d) %s\n", n++, e.c_str());
140        }
141    }
142
143    elements.clear();
144    redis.smembers("myset", std::inserter(elements, elements.begin()));
145    printf("redis < SMEMBERS myset\n");

```

```

146     if (elements.size() == 0) {
147         printf("redis < (empty array)\n");
148     } else {
149         int n = 1;
150         for (const std::string& e : elements) {
151             printf("redis > %d) %s\n", n++, e.c_str());
152         }
153     }
154
155     printf("=====\n");
156 }
157
158 printf("=====\n");
159 {
160     printf("SMOVE 命令\n\n");
161     redis.flushdb();
162
163     long long c = redis.sadd("myset", "one");
164     printf("redis < SADD myset one\n");
165     printf("redis > %lld\n", c);
166
167     c = redis.sadd("myset", "two");
168     printf("redis < SADD myset two\n");
169     printf("redis > %lld\n", c);
170
171     c = redis.sadd("myotherset", "three");
172     printf("redis < SADD myotherset three\n");
173     printf("redis > %lld\n", c);
174
175     bool r = redis.smove("myset", "myotherset", "two");
176     printf("redis < SMOVE myset myotherset two\n");
177     printf("redis > %d\n", r ? 1 : 0);
178
179     std::unordered_set<std::string> myset;
180     redis.smembers("myset", std::inserter(myset, myset.begin()));
181     printf("redis < SMEMBERS myset\n");
182     if (myset.size() == 0) {
183         printf("redis < (empty array)\n");
184     } else {
185         int n = 1;
186         for (const std::string& e : myset) {
187             printf("redis > %d) %s\n", n++, e.c_str());
188         }
189     }
190
191     std::unordered_set<std::string> myotherset;
192     redis.smembers("myotherset", std::inserter(myotherset, myset.begin()));

```

```

193     printf("redis < SMEMBERS myotherset\n");
194     if (myotherset.size() == 0) {
195         printf("redis < (empty array)\n");
196     } else {
197         int n = 1;
198         for (const std::string& e : myotherset) {
199             printf("redis > %d) %s\n", n++, e.c_str());
200         }
201     }
202
203     printf("=====\n");
204 }
205
206 {
207     printf("SREM 命令\n\n");
208     redis.flushdb();
209
210     long long c = redis.sadd("myset", "one");
211     printf("redis < SADD myset one\n");
212     printf("redis > %lld\n", c);
213
214     c = redis.sadd("myset", "two");
215     printf("redis < SADD myset two\n");
216     printf("redis > %lld\n", c);
217
218     c = redis.sadd("myset", "three");
219     printf("redis < SADD myset three\n");
220     printf("redis > %lld\n", c);
221
222     long long r = redis.srem("myset", "one");
223     printf("redis < SREM myset one\n");
224     printf("redis > %lld\n", r);
225
226     r = redis.srem("myset", "four");
227     printf("redis < SREM myset four\n");
228     printf("redis > %lld\n", r);
229
230     std::unordered_set<std::string> elements;
231     redis.smembers("myset", std::inserter(elements, elements.begin()));
232     printf("redis < SMEMBERS myset\n");
233     if (elements.size() == 0) {
234         printf("redis < (empty array)\n");
235     } else {
236         int n = 1;
237         for (const std::string& e : elements) {
238             printf("redis > %d) %s\n", n++, e.c_str());
239         }

```

```

240     }
241
242     printf("=====\n");
243 }
244
245 {
246     printf("SINTER 命令\n\n");
247     redis.flushdb();
248
249     long long c = redis.sadd("key1", "a");
250     printf("redis < SADD key1 a\n");
251     printf("redis > %lld\n", c);
252
253     c = redis.sadd("key1", "b");
254     printf("redis < SADD key1 b\n");
255     printf("redis > %lld\n", c);
256
257     c = redis.sadd("key1", "c");
258     printf("redis < SADD key1 c\n");
259     printf("redis > %lld\n", c);
260
261     c = redis.sadd("key2", "c");
262     printf("redis < SADD key2 c\n");
263     printf("redis > %lld\n", c);
264
265     c = redis.sadd("key2", "d");
266     printf("redis < SADD key2 d\n");
267     printf("redis > %lld\n", c);
268
269     c = redis.sadd("key2", "e");
270     printf("redis < SADD key2 e\n");
271     printf("redis > %lld\n", c);
272
273     std::unordered_set<std::string> elements;
274     redis.sinter({"key1", "key2"}, std::inserter(elements,
elements.begin()));
275     printf("redis < SINTER key1 key2\n");
276     if (elements.size() == 0) {
277         printf("redis < (empty array)\n");
278     } else {
279         int n = 1;
280         for (const std::string& e : elements) {
281             printf("redis > %d) %s\n", n++, e.c_str());
282         }
283     }
284
285     printf("=====\n");

```

```

286     }
287
288     {
289         printf("SINTERSTORE 命令\n\n");
290         redis.flushdb();
291
292         long long c = redis.sadd("key1", "a");
293         printf("redis < SADD key1 a\n");
294         printf("redis > %lld\n", c);
295
296         c = redis.sadd("key1", "b");
297         printf("redis < SADD key1 b\n");
298         printf("redis > %lld\n", c);
299
300         c = redis.sadd("key1", "c");
301         printf("redis < SADD key1 c\n");
302         printf("redis > %lld\n", c);
303
304         c = redis.sadd("key2", "c");
305         printf("redis < SADD key2 c\n");
306         printf("redis > %lld\n", c);
307
308         c = redis.sadd("key2", "d");
309         printf("redis < SADD key2 d\n");
310         printf("redis > %lld\n", c);
311
312         c = redis.sadd("key2", "e");
313         printf("redis < SADD key2 e\n");
314         printf("redis > %lld\n", c);
315
316         c = redis.sinterstore("key", {"key1", "key2"});
317         printf("redis < SINTERSTORE key key1 key2\n");
318         printf("redis > %lld\n", c);
319
320         std::unordered_set<std::string> elements;
321         redis.smembers("key", std::inserter(elements, elements.begin()));
322         printf("redis < SMEMBERS key\n");
323         if (elements.size() == 0) {
324             printf("redis < (empty array)\n");
325         } else {
326             int n = 1;
327             for (const std::string& e : elements) {
328                 printf("redis > %d) %s\n", n++, e.c_str());
329             }
330         }
331
332         printf("=====\n");

```



```

333     }
334
335     {
336         printf("SUNION 命令\n\n");
337         redis.flushdb();
338
339         long long c = redis.sadd("key1", "a");
340         printf("redis < SADD key1 a\n");
341         printf("redis > %lld\n", c);
342
343         c = redis.sadd("key1", "b");
344         printf("redis < SADD key1 b\n");
345         printf("redis > %lld\n", c);
346
347         c = redis.sadd("key1", "c");
348         printf("redis < SADD key1 c\n");
349         printf("redis > %lld\n", c);
350
351         c = redis.sadd("key2", "c");
352         printf("redis < SADD key2 c\n");
353         printf("redis > %lld\n", c);
354
355         c = redis.sadd("key2", "d");
356         printf("redis < SADD key2 d\n");
357         printf("redis > %lld\n", c);
358
359         c = redis.sadd("key2", "e");
360         printf("redis < SADD key2 e\n");
361         printf("redis > %lld\n", c);
362
363         std::unordered_set<std::string> elements;
364         redis.union({"key1", "key2"}, std::inserter(elements,
elements.begin()));
365         printf("redis < SUNION key1 key2\n");
366         if (elements.size() == 0) {
367             printf("redis < (empty array)\n");
368         } else {
369             int n = 1;
370             for (const std::string& e : elements) {
371                 printf("redis > %d) %s\n", n++, e.c_str());
372             }
373         }
374
375         printf("=====\n");
376     }
377
378     {

```

```

379     printf("SUNIONSTORE 命令\n\n");
380     redis.flushdb();
381
382     long long c = redis.sadd("key1", "a");
383     printf("redis < SADD key1 a\n");
384     printf("redis > %lld\n", c);
385
386     c = redis.sadd("key1", "b");
387     printf("redis < SADD key1 b\n");
388     printf("redis > %lld\n", c);
389
390     c = redis.sadd("key1", "c");
391     printf("redis < SADD key1 c\n");
392     printf("redis > %lld\n", c);
393
394     c = redis.sadd("key2", "c");
395     printf("redis < SADD key2 c\n");
396     printf("redis > %lld\n", c);
397
398     c = redis.sadd("key2", "d");
399     printf("redis < SADD key2 d\n");
400     printf("redis > %lld\n", c);
401
402     c = redis.sadd("key2", "e");
403     printf("redis < SADD key2 e\n");
404     printf("redis > %lld\n", c);
405
406     c = redis.sunionstore("key", {"key1", "key2"});
407     printf("redis < SUNIONSTORE key key1 key2\n");
408     printf("redis > %lld\n", c);
409
410     std::unordered_set<std::string> elements;
411     redis.smembers("key", std::inserter(elements, elements.begin()));
412     printf("redis < SMEMBERS key\n");
413     if (elements.size() == 0) {
414         printf("redis < (empty array)\n");
415     } else {
416         int n = 1;
417         for (const std::string& e : elements) {
418             printf("redis > %d) %s\n", n++, e.c_str());
419         }
420     }
421
422     printf("=====\n");
423 }
424
425 {

```

```

426     printf("SDIFF 命令\n\n");
427     redis.flushdb();
428
429     long long c = redis.sadd("key1", "a");
430     printf("redis < SADD key1 a\n");
431     printf("redis > %lld\n", c);
432
433     c = redis.sadd("key1", "b");
434     printf("redis < SADD key1 b\n");
435     printf("redis > %lld\n", c);
436
437     c = redis.sadd("key1", "c");
438     printf("redis < SADD key1 c\n");
439     printf("redis > %lld\n", c);
440
441     c = redis.sadd("key2", "c");
442     printf("redis < SADD key2 c\n");
443     printf("redis > %lld\n", c);
444
445     c = redis.sadd("key2", "d");
446     printf("redis < SADD key2 d\n");
447     printf("redis > %lld\n", c);
448
449     c = redis.sadd("key2", "e");
450     printf("redis < SADD key2 e\n");
451     printf("redis > %lld\n", c);
452
453     std::unordered_set<std::string> elements;
454     redis.sdiff({"key1", "key2"}, std::inserter(elements,
elements.begin()));
455     printf("redis < SDIFF key1 key2\n");
456     if (elements.size() == 0) {
457         printf("redis < (empty array)\n");
458     } else {
459         int n = 1;
460         for (const std::string& e : elements) {
461             printf("redis > %d) %s\n", n++, e.c_str());
462         }
463     }
464
465     printf("=====\n");
466 }
467
468 {
469     printf("SDIFFSTORE 命令\n\n");
470     redis.flushdb();
471

```

```

472     long long c = redis.sadd("key1", "a");
473     printf("redis < SADD key1 a\n");
474     printf("redis > %lld\n", c);
475
476     c = redis.sadd("key1", "b");
477     printf("redis < SADD key1 b\n");
478     printf("redis > %lld\n", c);
479
480     c = redis.sadd("key1", "c");
481     printf("redis < SADD key1 c\n");
482     printf("redis > %lld\n", c);
483
484     c = redis.sadd("key2", "c");
485     printf("redis < SADD key2 c\n");
486     printf("redis > %lld\n", c);
487
488     c = redis.sadd("key2", "d");
489     printf("redis < SADD key2 d\n");
490     printf("redis > %lld\n", c);
491
492     c = redis.sadd("key2", "e");
493     printf("redis < SADD key2 e\n");
494     printf("redis > %lld\n", c);
495
496     c = redis.sdiffstore("key", {"key1", "key2"});
497     printf("redis < SDIFFSTORE key key1 key2\n");
498     printf("redis > %lld\n", c);
499
500     std::unordered_set<std::string> elements;
501     redis.smembers("key", std::inserter(elements, elements.begin()));
502     printf("redis < SMEMBERS key\n");
503     if (elements.size() == 0) {
504         printf("redis < (empty array)\n");
505     } else {
506         int n = 1;
507         for (const std::string& e : elements) {
508             printf("redis > %d) %s\n", n++, e.c_str());
509         }
510     }
511
512     printf("=====\n");
513 }
514 }
515
516 int main() {
517     sw::redis::Redis redis("tcp://127.0.0.1:6379");
518

```

```

519 // Set commands
520 testSetCommands(redis);
521
522 return 0;
523 }

```

set 测试结果

```

1 $ make set
2 g++ -std=c++17 -g -O0 -o set src/set.cc /usr/local/lib64/libredis++.a
  /usr/local/lib/libhiredis.a -pthread
3 $ ./set
4 Set 系列命令:
5 =====
6 SADD 命令
7
8 SMEMBERS 命令
9
10 redis < SADD myset Hello
11 redis > 1
12 redis < SADD myset World
13 redis > 1
14 redis < SADD myset World
15 redis > 0
16 redis < SMEMBERS myset
17 redis > 1) World
18 redis > 2) Hello
19 =====
20 =====
21 SISMEMBER 命令
22
23 redis < SADD myset one
24 redis > 1
25 redis < SISMEMBER myset one
26 redis > 1
27 redis < SISMEMBER myset two
28 redis > 0
29 =====
30 =====
31 SCARD 命令
32
33 redis < SADD myset Hello
34 redis > 1
35 redis < SADD myset World
36 redis > 1

```

```
37 redis < SCARD myset
38 redis > 2
39 =====
40 =====
41 SPOP 命令
42
43 redis < SADD myset one
44 redis > 1
45 redis < SADD myset two
46 redis > 1
47 redis < SADD myset three
48 redis > 1
49 redis < SPOP myset
50 redis > one
51 redis < SMEMBERS myset
52 redis > 1) two
53 redis > 2) three
54 redis < SADD myset four
55 redis > 1
56 redis < SADD myset five
57 redis > 1
58 redis < SPOP myset 3
59 redis > 1) five
60 redis > 2) four
61 redis > 3) two
62 redis < SMEMBERS myset
63 redis > 1) three
64 =====
65 =====
66 SMOVE 命令
67
68 redis < SADD myset one
69 redis > 1
70 redis < SADD myset two
71 redis > 1
72 redis < SADD myotherset three
73 redis > 1
74 redis < SMOVE myset myotherset two
75 redis > 1
76 redis < SMEMBERS myset
77 redis > 1) one
78 redis < SMEMBERS myotherset
79 redis > 1) three
80 redis > 2) two
81 =====
82 SREM 命令
83
```

```
84 redis < SADD myset one
85 redis > 1
86 redis < SADD myset two
87 redis > 1
88 redis < SADD myset three
89 redis > 1
90 redis < SREM myset one
91 redis > 1
92 redis < SREM myset four
93 redis > 0
94 redis < SMEMBERS myset
95 redis > 1) two
96 redis > 2) three
97 =====
98 SINTER 命令
99
100 redis < SADD key1 a
101 redis > 1
102 redis < SADD key1 b
103 redis > 1
104 redis < SADD key1 c
105 redis > 1
106 redis < SADD key2 c
107 redis > 1
108 redis < SADD key2 d
109 redis > 1
110 redis < SADD key2 e
111 redis > 1
112 redis < SINTER key1 key2
113 redis > 1) c
114 =====
115 SINTERSTORE 命令
116
117 redis < SADD key1 a
118 redis > 1
119 redis < SADD key1 b
120 redis > 1
121 redis < SADD key1 c
122 redis > 1
123 redis < SADD key2 c
124 redis > 1
125 redis < SADD key2 d
126 redis > 1
127 redis < SADD key2 e
128 redis > 1
129 redis < SINTERSTORE key key1 key2
130 redis > 1
```

```
131 redis < SMEMBERS key
132 redis > 1) c
133 =====
134 SUNION 命令
135
136 redis < SADD key1 a
137 redis > 1
138 redis < SADD key1 b
139 redis > 1
140 redis < SADD key1 c
141 redis > 1
142 redis < SADD key2 c
143 redis > 1
144 redis < SADD key2 d
145 redis > 1
146 redis < SADD key2 e
147 redis > 1
148 redis < SUNION key1 key2
149 redis > 1) e
150 redis > 2) b
151 redis > 3) d
152 redis > 4) c
153 redis > 5) a
154 =====
155 SUNIONSTORE 命令
156
157 redis < SADD key1 a
158 redis > 1
159 redis < SADD key1 b
160 redis > 1
161 redis < SADD key1 c
162 redis > 1
163 redis < SADD key2 c
164 redis > 1
165 redis < SADD key2 d
166 redis > 1
167 redis < SADD key2 e
168 redis > 1
169 redis < SUNIONSTORE key key1 key2
170 redis > 5
171 redis < SMEMBERS key
172 redis > 1) e
173 redis > 2) b
174 redis > 3) d
175 redis > 4) c
176 redis > 5) a
177 =====
```



```
178 SDIFF 命令
179
180 redis < SADD key1 a
181 redis > 1
182 redis < SADD key1 b
183 redis > 1
184 redis < SADD key1 c
185 redis > 1
186 redis < SADD key2 c
187 redis > 1
188 redis < SADD key2 d
189 redis > 1
190 redis < SADD key2 e
191 redis > 1
192 redis < SDIFF key1 key2
193 redis > 1) a
194 redis > 2) b
195 =====
196 SDIFFSTORE 命令
197
198 redis < SADD key1 a
199 redis > 1
200 redis < SADD key1 b
201 redis > 1
202 redis < SADD key1 c
203 redis > 1
204 redis < SADD key2 c
205 redis > 1
206 redis < SADD key2 d
207 redis > 1
208 redis < SADD key2 e
209 redis > 1
210 redis < SDIFFSTORE key key1 key2
211 redis > 2
212 redis < SMEMBERS key
213 redis > 1) a
214 redis > 2) b
215 =====
```

src/hash.cc

```
1 #include <sw/redis++/redis++.h>
2 #include <cstdio>
```

```
3 #include <string>
4 #include <chrono>
5 #include <sw/redis++/redis.h>
6 #include <sw/redis++/utils.h>
7 #include <thread>
8 #include <iterator>
9 #include <vector>
10
11 using std::string;
12 using std::vector;
13 using sw::redis::Redis;
14 using sw::redis::OptionalString;
15
16 void testHashCommand(Redis& redis) {
17     printf("Hash 系列命令:\n");
18     printf("=====\n");
19     {
20         printf("hset 和 hget\n\n");
21         redis.flushdb();
22
23         redis.hset("key", "name", "zhangsan");
24         printf("redis < hset key name zhangsan\n");
25         redis.hset("key", "age", "20");
26         printf("redis < hset key age 20\n");
27
28         OptionalString name = redis.hget("key", "name");
29         printf("redis < hget key name\n");
30         printf("redis > name = %s\n", name->c_str());
31         OptionalString age = redis.hget("key", "age");
32         printf("redis < hget key age\n");
33         printf("redis > age = %s\n", age->c_str());
34
35         printf("=====\n");
36     }
37
38     {
39         printf("hexists 和 hdel\n\n");
40         redis.flushdb();
41
42         redis.hset("key", "name", "zhangsan");
43         printf("redis < hset key name zhangsan\n");
44
45         bool ok = redis.exists("key", "name");
46         printf("redis < hexists key name\n");
47         printf("redis > ok = %d\n", ok);
48
49         long long n = redis.hdel("key", "name");
```

```
50     printf("redis < hdel key name\n");
51     printf("n = %lld\n", n);
52
53     ok = redis.hexists("key", "name");
54     printf("redis < hexists key name\n");
55     printf("redis > ok = %d\n", ok);
56
57     printf("=====\n");
58 }
59
60 {
61     printf("hkeys 和 hvals\n\n");
62     redis.flushdb();
63
64     redis.hset("key", "name", "zhangsan");
65     printf("redis < hset key name zhangsan\n");
66     redis.hset("key", "age", "20");
67     printf("redis < hset key age 20\n");
68
69     vector<string> keys;
70     auto itKeys = std::back_inserter(keys);
71     redis.hkeys("key", itKeys);
72     printf("redis < hkeys key\n");
73     for (const auto& key : keys) {
74         printf("redis > key %s\n", key.c_str());
75     }
76
77     vector<string> values;
78     auto itValues = std::back_inserter(values);
79     redis.hvals("key", itValues);
80     printf("redis < hvals key\n");
81     for (const auto& val : values) {
82         printf("redis > val %s\n", val.c_str());
83     }
84
85     printf("=====\n");
86 }
87
88 {
89     printf("hmget\n\n");
90     redis.flushdb();
91
92     redis.hset("key", "name", "zhangsan");
93     printf("redis < hset key name zhangsan\n");
94     redis.hset("key", "age", "20");
95     printf("redis < hset key age 20\n");
96 }
```

```

97     vector<OptionalString> values;
98     auto it = std::back_inserter(values);
99     vector<string> fields = {"name", "age"};
100    // 参数的多个 field 需要通过一对迭代器来表示.
101    redis.hmget("key", fields.begin(), fields.end(), it);
102    printf("redis < hmget key\n");
103    for (const auto& val : values) {
104        printf("redis > val %s\n", val->c_str());
105    }
106
107    printf("=====\n");
108 }
109
110 {
111     printf("hlen\n\n");
112     redis.flushdb();
113
114     redis.hset("key", "name", "zhangsan");
115     printf("redis < hset key name zhangsan\n");
116     redis.hset("key", "age", "20");
117     printf("redis < hset key age 20\n");
118
119     long long len = redis.hlen("key");
120     printf("redis < hlen key\n");
121     printf("redis > len=%lld\n", len);
122
123     printf("=====\n");
124 }
125
126 {
127     printf("hincrby 和 hincrbyfloat\n\n");
128     redis.flushdb();
129
130     redis.hset("key", "age", "20");
131     printf("redis < hset key age 20\n");
132
133     long long age = redis.hincrby("key", "age", 10);
134     printf("redis > hincrby key age 10\n");
135     printf("redis > age=%lld\n", age);
136
137     double ageDouble = redis.hincrbyfloat("key", "age", 0.5);
138     printf("redis > hincrbyfloat key age 0.5\n");
139     printf("redis > age=%lf\n", ageDouble);
140
141     printf("=====\n");
142 }
143 }

```

```

144
145 int main() {
146     Redis redis("tcp://127.0.0.1:6379");
147     testHashCommand(redis);
148     return 0;
149 }
150

```

hash 测试结果

```

1 Hash 系列命令:
2 =====
3 hset 和 hget
4
5 redis < hset key name zhangsan
6 redis < hset key age 20
7 redis < hget key name
8 redis > name = zhangsan
9 redis < hget key age
10 redis > age = 20
11 =====
12 hexists 和 hdel
13
14 redis < hset key name zhangsan
15 redis < hexists key name
16 redis > ok = 1
17 redis < hdel key name
18 n = 1
19 redis < hexists key name
20 redis > ok = 0
21 =====
22 hkeys 和 hvals
23
24 redis < hset key name zhangsan
25 redis < hset key age 20
26 redis < hkeys key
27 redis > key name
28 redis > key age
29 redis < hvals key
30 redis > val zhangsan
31 redis > val 20
32 =====
33 hmget

```

```

34
35 redis < hset key name zhangsan
36 redis < hset key age 20
37 redis < hmget key
38 redis > val zhangsan
39 redis > val 20
40 =====
41 hlen
42
43 redis < hset key name zhangsan
44 redis < hset key age 20
45 redis < hlen key
46 redis > len=2
47 =====
48 hincrby 和 hincrbyfloat
49
50 redis < hset key age 20
51 redis > hincrby key age 10
52 redis > age=30
53 redis > hincrbyfloat key age 0.5
54 redis > age=30.500000
55 =====

```

src/zset.cc

```

1 #include <sw/redis++/command_options.h>
2 #include <sw/redis++/redis++.h>
3 #include <cstdio>
4 #include <string>
5 #include <chrono>
6 #include <sw/redis++/redis.h>
7 #include <sw/redis++/utils.h>
8 #include <thread>
9 #include <iterator>
10 #include <vector>
11
12 using std::string;
13 using std::vector;
14 using sw::redis::Redis;
15 using sw::redis::OptionalString;
16 using sw::redis::OptionalLongLong;
17 using sw::redis::OptionalDouble;
18

```

```

19 void testZsetCommand(Redis& redis) {
20     printf("Zset 系列命令:\n");
21     printf("=====\n");
22     {
23         printf("zadd 和 zrange\n\n");
24         redis.flushdb();
25
26         redis.zadd("key", "吕布", 100);
27         printf("redis < zadd key 100 吕布\n");
28         redis.zadd("key", "赵云", 98);
29         printf("redis < zadd key 98 赵云\n");
30         redis.zadd("key", "典韦", 95);
31         printf("redis < zadd key 95 典韦\n");
32         redis.zadd("key", "关羽", 92);
33         printf("redis < zadd key 92 关羽\n");
34         redis.zadd("key", "刘备", 70);
35         printf("redis < zadd key 70 刘备\n");
36
37         vector<OptionalString> members;
38         auto it = std::back_inserter(members);
39         redis.zrange("key", 0, 4, it);
40         printf("redis < zrange key 0 4\n");
41         for (const auto& member : members) {
42             printf("redis > member=%s\n", member->c_str());
43         }
44
45         vector<std::pair<string, double>> membersWithScore;
46         auto itWithScore = std::back_inserter(membersWithScore);
47         redis.zrange("key", 0, 4, itWithScore);
48         printf("redis < zrange key 0 4 withscores\n");
49         for (const auto& member : membersWithScore) {
50             printf("redis > member=%s, score=%lf\n", member.first.c_str(),
member.second);
51         }
52
53         printf("=====\n");
54     }
55
56     {
57         printf("zrem 和 zcard\n\n");
58         redis.flushdb();
59
60         redis.zadd("key", "吕布", 100);
61         printf("redis < zadd key 100 吕布\n");
62         redis.zadd("key", "赵云", 98);
63         printf("redis < zadd key 98 赵云\n");
64         redis.zadd("key", "典韦", 95);

```

```

65     printf("redis < zadd key 95 典韦\n");
66     redis.zadd("key", "关羽", 92);
67     printf("redis < zadd key 92 关羽\n");
68     redis.zadd("key", "刘备", 70);
69     printf("redis < zadd key 70 刘备\n");
70
71     long long n = redis.zcard("key");
72     printf("redis < zcard key\n");
73     printf("redis > n=%lld\n", n);
74
75     n = redis.zrem("key", "吕布");
76     printf("redis < zrem key 吕布\n");
77     printf("redis > n=%lld\n", n);
78
79     n = redis.zcard("key");
80     printf("redis < zcard key\n");
81     printf("redis > n=%lld\n", n);
82
83     printf("=====\n");
84 }
85
86 {
87     printf("zcount\n\n");
88     redis.flushdb();
89
90     redis.zadd("key", "吕布", 100);
91     printf("redis < zadd key 100 吕布\n");
92     redis.zadd("key", "赵云", 98);
93     printf("redis < zadd key 98 赵云\n");
94     redis.zadd("key", "典韦", 95);
95     printf("redis < zadd key 95 典韦\n");
96     redis.zadd("key", "关羽", 92);
97     printf("redis < zadd key 92 关羽\n");
98     redis.zadd("key", "刘备", 70);
99     printf("redis < zadd key 70 刘备\n");
100
101     // 这里可以手动设定区间是开区间还是闭区间。
102     long long n = redis.zcount("key", sw::redis::BoundedInterval<double>
(92, 100, sw::redis::BoundType::CLOSED));
103     printf("redis < zcount key 92 100\n");
104     printf("redis > n=%lld\n", n);
105
106     printf("=====\n");
107 }
108
109 {
110     printf("zpopmax 和 zpopmin\n\n");

```



```

111     redis.flushdb();
112
113     redis.zadd("key", "吕布", 100);
114     printf("redis < zadd key 100 吕布\n");
115     redis.zadd("key", "赵云", 98);
116     printf("redis < zadd key 98 赵云\n");
117     redis.zadd("key", "典韦", 95);
118     printf("redis < zadd key 95 典韦\n");
119     redis.zadd("key", "关羽", 92);
120     printf("redis < zadd key 92 关羽\n");
121     redis.zadd("key", "刘备", 70);
122     printf("redis < zadd key 70 刘备\n");
123
124     auto result = redis.zpopmax("key");
125     printf("redis < zpopmax key\n");
126     printf("redis > member=%s, score=%lf\n", result->first.c_str(), result-
>second);
127
128     result = redis.zpopmin("key");
129     printf("redis < zpopmin key\n");
130     printf("redis > member=%s, score=%lf\n", result->first.c_str(), result-
>second);
131
132     printf("=====\n");
133 }
134
135 {
136     printf("zrank\n\n");
137     redis.flushdb();
138
139     redis.zadd("key", "吕布", 100);
140     printf("redis < zadd key 100 吕布\n");
141     redis.zadd("key", "赵云", 98);
142     printf("redis < zadd key 98 赵云\n");
143     redis.zadd("key", "典韦", 95);
144     printf("redis < zadd key 95 典韦\n");
145     redis.zadd("key", "关羽", 92);
146     printf("redis < zadd key 92 关羽\n");
147     redis.zadd("key", "刘备", 70);
148     printf("redis < zadd key 70 刘备\n");
149
150     // n 可能是无效值
151     OptionalLongLong n = redis.zrank("key", "吕布");
152     printf("redis < zrank key 吕布\n");
153     printf("redis > n=%lld\n", n.value());
154
155     n = redis.zrevrank("key", "吕布");

```

```

156     printf("redis < zrevrank key 吕布\n");
157     printf("redis > n=%lld\n", n.value());
158
159     printf("=====\n");
160 }
161
162 {
163     printf("zscore\n\n");
164     redis.flushdb();
165
166     redis.zadd("key", "吕布", 100);
167     printf("redis < zadd key 100 吕布\n");
168
169     OptionalDouble score = redis.zscore("key", "吕布");
170     printf("redis < zscore key 吕布\n");
171     printf("redis > score=%lf\n", score.value());
172
173     printf("=====\n");
174 }
175
176 {
177     printf("zincrby\n\n");
178     redis.flushdb();
179
180     redis.zadd("key", "吕布", 100);
181     printf("redis < zadd key 100 吕布\n");
182
183     double score = redis.zincrby("key", 10, "吕布");
184     printf("redis < zincrby key 10 吕布\n");
185     printf("redis > score=%lf\n", score);
186
187     score = redis.zincrby("key", -20, "吕布");
188     printf("redis < zincrby key -20 吕布\n");
189     printf("redis > score=%lf\n", score);
190
191     printf("=====\n");
192 }
193
194 {
195     printf("zinterstore\n\n");
196     redis.flushdb();
197
198     redis.zadd("key1", "吕布", 100);
199     printf("redis < zadd key1 100 吕布\n");
200     redis.zadd("key1", "赵云", 98);
201     printf("redis < zadd key1 98 赵云\n");
202     redis.zadd("key1", "典韦", 95);

```

```

203     printf("redis < zadd key1 95 典韦\n");
204
205     redis.zadd("key2", "吕布", 100);
206     printf("redis < zadd key2 100 吕布\n");
207     redis.zadd("key2", "赵云", 98);
208     printf("redis < zadd key2 98 赵云\n");
209     redis.zadd("key2", "关羽", 92);
210     printf("redis < zadd key2 92 关羽\n");
211
212     long long n = redis.zinterstore("key3", {"key1", "key2"});
213     printf("redis < zinterstore key3 key1 key2\n");
214     printf("redis > n=%lld\n", n);
215
216     vector<std::pair<string, double>> members;
217     auto it = std::back_inserter(members);
218     redis.zrange("key3", 0, -1, it);
219     for (const auto& member : members) {
220         printf("redis > member=%s, score=%lf\n", member.first.c_str(),
member.second);
221     }
222
223     printf("=====\n");
224 }
225
226 {
227     printf("zunionstore\n\n");
228     redis.flushdb();
229
230     redis.zadd("key1", "吕布", 100);
231     printf("redis < zadd key1 100 吕布\n");
232     redis.zadd("key1", "赵云", 98);
233     printf("redis < zadd key1 98 赵云\n");
234     redis.zadd("key1", "典韦", 95);
235     printf("redis < zadd key1 95 典韦\n");
236
237     redis.zadd("key2", "吕布", 100);
238     printf("redis < zadd key2 100 吕布\n");
239     redis.zadd("key2", "赵云", 98);
240     printf("redis < zadd key2 98 赵云\n");
241     redis.zadd("key2", "关羽", 92);
242     printf("redis < zadd key2 92 关羽\n");
243
244     long long n = redis.zunionstore("key3", {"key1", "key2"});
245     printf("redis < zunionstore key3 key1 key2\n");
246     printf("redis > n=%lld\n", n);
247
248     vector<std::pair<string, double>> members;

```

```

249     auto it = std::back_inserter(members);
250     redis.zrange("key3", 0, -1, it);
251     for (const auto& member : members) {
252         printf("redis > member=%s, score=%lf\n", member.first.c_str(),
member.second);
253     }
254
255     printf("=====\n");
256 }
257 }
258
259 int main() {
260     Redis redis("tcp://127.0.0.1:16379");
261     testZsetCommand(redis);
262     return 0;
263 }

```

zset 测试结果

```

1 Zset 系列命令:
2 =====
3 zadd 和 zrange
4
5 redis < zadd key 100 吕布
6 redis < zadd key 98 赵云
7 redis < zadd key 95 典韦
8 redis < zadd key 92 关羽
9 redis < zadd key 70 刘备
10 redis < zrange key 0 4
11 redis > member=刘备
12 redis > member=关羽
13 redis > member=典韦
14 redis > member=赵云
15 redis > member=吕布
16 redis < zrange key 0 4 withscores
17 redis > member=刘备, score=70.000000
18 redis > member=关羽, score=92.000000
19 redis > member=典韦, score=95.000000
20 redis > member=赵云, score=98.000000
21 redis > member=吕布, score=100.000000
22 =====
23 zrem 和 zcard
24

```

```
25 redis < zadd key 100 吕布
26 redis < zadd key 98 赵云
27 redis < zadd key 95 典韦
28 redis < zadd key 92 关羽
29 redis < zadd key 70 刘备
30 redis < zcard key
31 redis > n=5
32 redis < zrem key 吕布
33 redis > n=1
34 redis < zcard key
35 redis > n=4
36 =====
37 zcount
38
39 redis < zadd key 100 吕布
40 redis < zadd key 98 赵云
41 redis < zadd key 95 典韦
42 redis < zadd key 92 关羽
43 redis < zadd key 70 刘备
44 redis < zcount key 92 100
45 redis > n=4
46 =====
47 zpopmax 和 zpopmin
48
49 redis < zadd key 100 吕布
50 redis < zadd key 98 赵云
51 redis < zadd key 95 典韦
52 redis < zadd key 92 关羽
53 redis < zadd key 70 刘备
54 redis < zpopmax key
55 redis > member=吕布, score=100.000000
56 redis < zpopmin key
57 redis > member=刘备, score=70.000000
58 =====
59 zrank
60
61 redis < zadd key 100 吕布
62 redis < zadd key 98 赵云
63 redis < zadd key 95 典韦
64 redis < zadd key 92 关羽
65 redis < zadd key 70 刘备
66 redis < zrank key 吕布
67 redis > n=4
68 redis < zrevrank key 吕布
69 redis > n=0
70 =====
71 zscore
```

```
72
73 redis < zadd key 100 吕布
74 redis < zscore key 吕布
75 redis > score=100.000000
76 =====
77 zincrby
78
79 redis < zadd key 100 吕布
80 redis < zincrby key 10 吕布
81 redis > score=110.000000
82 redis < zincrby key -20 吕布
83 redis > score=90.000000
84 =====
85 zinterstore
86
87 redis < zadd key1 100 吕布
88 redis < zadd key1 98 赵云
89 redis < zadd key1 95 典韦
90 redis < zadd key2 100 吕布
91 redis < zadd key2 98 赵云
92 redis < zadd key2 92 关羽
93 redis < zinterstore key3 key1 key2
94 redis > n=2
95 redis > member=赵云, score=196.000000
96 redis > member=吕布, score=200.000000
97 =====
98 zunionstore
99
100 redis < zadd key1 100 吕布
101 redis < zadd key1 98 赵云
102 redis < zadd key1 95 典韦
103 redis < zadd key2 100 吕布
104 redis < zadd key2 98 赵云
105 redis < zadd key2 92 关羽
106 redis < zunionstore key3 key1 key2
107 redis > n=4
108 redis > member=关羽, score=92.000000
109 redis > member=典韦, score=95.000000
110 redis > member=赵云, score=196.000000
111 redis > member=吕布, score=200.000000
112 =====
```

访问集群

只需要使用 `RedisCluster` 类代替 `Redis` 类即可.

构造方法中填入集群中的任意节点地址.

`RedisCluster` 提供了和 `Redis` 类一样的接口, 基本都和 Redis 原生命令一致.

```
1 #include <sw/redis++/redis++.h>
2 #include <cstdio>
3 #include <string>
4 #include <vector>
5
6 using std::string;
7 using sw::redis::RedisCluster;
8 using sw::redis::OptionalString;
9
10 int main() {
11     RedisCluster cluster("tcp://127.0.0.1:6379");
12
13     cluster.set("k1", "111");
14     auto value = cluster.get("k1");
15     printf("value: %s\n", value->c_str());
16
17     return 0;
18 }
```

修改 makefile

原有内容保持不变.

```
1 all: generic string list set hash zset cluster
2
3 ...
4
5 cluster: src/cluster.cc
6     g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-
    linux-gnu/libhiredis.a -pthread
7
8 .PHONY: clean
9 clean:
10     ...
11     rm -rf cluster
```

执行结果

```
1 $ ./cluster  
2 value: 111
```

比特就业课