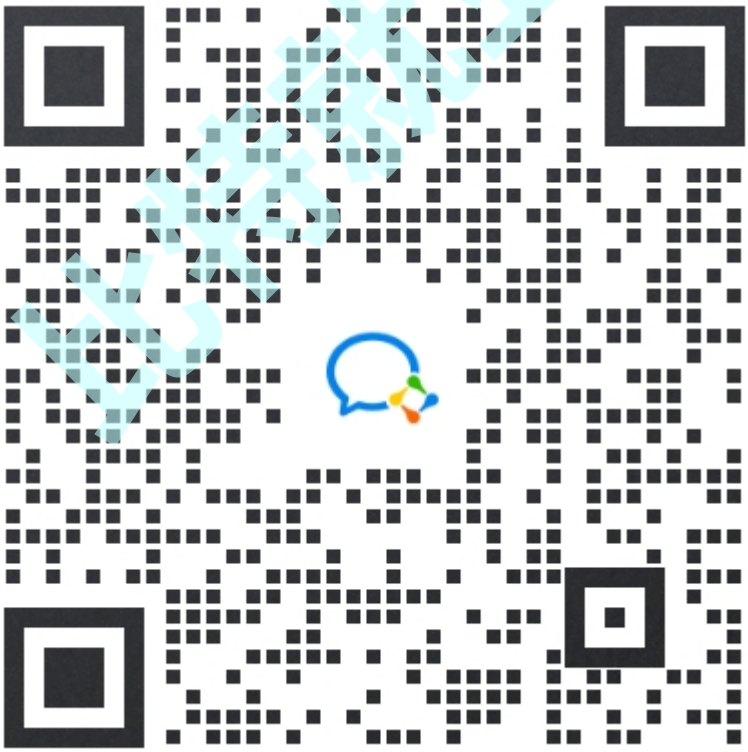
Redis C++使用 样例列表

版权说明

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/sewenew/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/sewenew/redis-plus-plus.git
```

编译/安装 redis-plus-plus

Ubuntu

使用 cmake 构建

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

```
mkdir build

cd build

cmake ..

make

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

```
yum install cmake3
```

然后使用 cmake3 构建项目.

```
cd redis-plus-plus

mkdir build

cd build

cmake3 ..

make

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/peixinchen2/redis-plus-plus/blob/master/src/sw/redis++/redis.h

源码结构

```
.
├── Makefile
└── src
    ├── generic.cc
    ├── list.cc
    ├── set.cc
    └── string.cc

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

```
all: generic string list set

generic: src/generic.cc
    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-linux-gnu/libhiredis.a -pthread

string: src/string.cc
    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-linux-gnu/libhiredis.a -pthread

list: src/list.cc
    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-linux-gnu/libhiredis.a -pthread

```

```
set: src/set.cc
    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-linux-gnu/libhiredis.a -pthread

.PHONY: clean
clean:
    rm -rf generic
    rm -rf string
    rm -rf list
    rm -rf set
```

Centos

```
all: generic string list set

generic: src/generic.cc
        g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a /usr/local/lib/libhiredis.a -pthread

string: src/string.cc
        g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a /usr/local/lib/libhiredis.a -pthread

list: src/list.cc
        g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a /usr/local/lib/libhiredis.a -pthread

set: src/set.cc
        g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib64/libredis++.a /usr/local/lib/libhiredis.a -pthread

.PHONY: clean
clean:
        rm -rf generic
        rm -rf string
        rm -rf list
        rm -rf set
```

src/generic.cc

```cpp
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <vector>
#include <unordered_map>
#include <chrono>


void testGenericCommands(sw::redis::Redis& redis) {
    printf("Generic 系列命令:\n");
    printf("============================================\n");
    {
        printf("EXISTS 命令\n\n");
        redis.flushdb();

        bool b;
        b = redis.set("key1", "Hello");
        printf("redis < SET key1 \"Hello\"\n");
        printf("redis > %s\n", b ? "OK" : "(nil)");

        long long e;
        e = redis.exists("key1");
        printf("redis < EXISTS key1\n");
        printf("redis > %lld\n", e);

        e = redis.exists("nosuchkey");
        printf("redis < EXISTS nosuchkey\n");
        printf("redis > %lld\n", e);

        b = redis.set("key2", "World");
        printf("redis < SET key2 \"World\"\n");
        printf("redis > %s\n", b ? "OK" : "(nil)");

        e = redis.exists({"key1", "key2", "nosuchkey"});
        printf("redis < EXISTS key1 key2 nosuchkey\n");
        printf("redis > %lld\n", e);
        printf("============================================\n");
    }

    {
        printf("DEL 命令\n\n");
        redis.flushdb();
        bool b;
        b = redis.set("key1", "Hello");
        printf("redis < SET key1 \"Hello\"\n");
        printf("redis > %s\n", b ? "OK" : "(nil)");

```

```
48          b = redis.set("key2", "World");
49          printf("redis < SET key2 \"World\"\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");
```

```
    }

    {
        printf("TYPE 命令\n\n");
        redis.flushdb();
        bool b;
        b = redis.set("key1", "value");
        printf("redis < SET key1 \"value\"\n");
        printf("redis > %s\n", b ? "OK" : "(nil)");

        long long c = redis.lpush("key2", "value");
        printf("redis < LPUSH key2 \"value\"\n");
        printf("redis > %d\n", c);

        c = redis.sadd("key3", "value");
        printf("redis < SADD key3 \"value\"\n");
        printf("redis > %d\n", c);

        std::string type = redis.type("key1");
        printf("redis < TYPE key1\n");
        printf("redis > \"%s\"\n", type.c_str());

        type = redis.type("key2");
        printf("redis < TYPE key2\n");
        printf("redis > \"%s\"\n", type.c_str());

        type = redis.type("key3");
        printf("redis < TYPE key3\n");
        printf("redis > \"%s\"\n", type.c_str());

        printf("==========================================\n");
    }
}

int main() {
    sw::redis::Redis redis("tcp://127.0.0.1:6379");

    // Generic commands
    testGenericCommands(redis);

    return 0;
}
```

generic 测试结果

```
$ make generic
g++ -std=c++17 -o generic src/generic.cc /usr/local/lib64/libredis++.a
/usr/local/lib/libhiredis.a -pthread
$ ./generic
Generic 系列命令:
==========================================
EXISTS 命令

redis < SET key1 "Hello"
redis > OK
redis < EXISTS key1
redis > 1
redis < EXISTS nosuchkey
redis > 0
redis < SET key2 "World"
redis > OK
redis < EXISTS key1 key2 nosuchkey
redis > 2
==========================================
DEL 命令

redis < SET key1 "Hello"
redis > OK
redis < SET key2 "World"
redis > OK
redis < DEL key1 key2 key3
redis > 2
==========================================
KEYS 命令

redis < MSET firstname Jack lastname Stuntman age 35
redis > OK
redis < KEYS *name*
redis > 1) firstname
redis > 2) lastname
redis < KEYS a??
redis > 1) age
redis < KEYS *
redis > 1) firstname
redis > 2) age
redis > 3) lastname
==========================================
EXPIRE 命令
TTL 命令

redis < SET mykey "Hello"
redis > OK
```

```
redis < EXPIRE mykey 10
redis > 1
redis < TTL mykey
redis > 10
==========================================
PTTL 命令

redis < SET mykey "Hello"
redis > OK
redis < EXPIRE mykey 10
redis > 1
redis < PTTL mykey
redis > 10000
==========================================
TYPE 命令

redis < SET key1 "value"
redis > OK
redis < LPUSH key2 "value"
redis > 1
redis < SADD key3 "value"
redis > 1
redis < TYPE key1
redis > "string"
redis < TYPE key2
redis > "list"
redis < TYPE key3
redis > "set"
==========================================
```

src/string.cc

```cpp
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <chrono>
#include <thread>


void testStringCommands(sw::redis::Redis& redis) {
    printf("String 系列命令:\n");
    printf("==========================================\n");
    {
        printf("SET 命令\n\n");

```

```
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");
```

```
        printf("redis > %lld\n", c);

        c = redis.strlen("nonexisting");
        printf("redis < STRLEN nonexisting\n");
        printf("redis > %lld\n", c);

        printf("============================================\n");
    }
}

int main() {
    sw::redis::Redis redis("tcp://127.0.0.1:6379");

    // String commands
    testStringCommands(redis);

    return 0;
}
```

string 测试结果

```
$ make string
g++ -std=c++17 -o string src/string.cc /usr/local/lib64/libredis++.a
/usr/local/lib/libhiredis.a -pthread
$ ./string
String 系列命令:
GET 命令

redis < GET nonexisting
redis > (nil)
redis < SET mykey "Hello"
redis > OK
redis < GET mykey
redis > "Hello"
redis < DEL mykey
redis > 1
redis < EXISTS mykey
redis > 0
redis < HSET mykey name Bob
redis > 1
redis < GET mykey
redis > WRONGTYPE Operation against a key holding the wrong kind of value
============================================
APPEND 命令

```

```
redis < EXISTS mykey
redis > 0
redis < APPEND mykey "Hello"
redis > 5
redis < GET mykey
redis > "Hello"
redis < APPEND mykey " World"
redis > 11
redis < GET mykey
redis > "Hello World"
================================================
GETRANGE 命令

redis < SET mykey "This is a string"
redis > OK
redis < GETRANGE mykey 0 3
redis > This
redis < GETRANGE mykey -3 -1
redis > ing
redis < GETRANGE mykey 0 -1
redis > This is a string
redis < GETRANGE mykey 10 100
redis > string
================================================
SETNX 命令

redis < SETNX mykey "Hello"
redis > 1
redis < SETNX mykey "World"
redis > 0
redis < GET mykey
redis > "Hello"
================================================
DECR 命令

redis < EXISTS mykey
redis > 0
redis < DECR mykey
redis > -1
redis < SET mykey "10"
redis > OK
redis < DECR mykey
redis > 9
redis < SET mykey "234293482390480948029348230948"
redis > OK
redis > ERR value is not an integer or out of range
redis < SET mykey "not a number"
```

```
redis > OK
redis > ERR value is not an integer or out of range
==========================================
DECRBY 命令

redis < EXISTS mykey
redis > 0
redis < DECRBY mykey 3
redis > -3
redis < SET mykey "10"
redis > OK
redis < DECRBY mykey 3
redis > 7
==========================================
INCR 命令

redis < EXISTS mykey
redis > 0
redis < INCR mykey
redis > 1
redis < SET mykey "10"
redis > OK
redis < INCR mykey
redis > 11
redis < SET mykey "234293482390480948029348230948"
redis > OK
redis > ERR value is not an integer or out of range
redis < SET mykey "not a number"
redis > OK
redis > ERR value is not an integer or out of range
==========================================
INCRBY 命令

redis < EXISTS mykey
redis > 0
redis < INCRBY mykey 3
redis > 3
redis < SET mykey "10"
redis > OK
redis < INCRBY mykey 3
redis > 13
==========================================
INCRBYFLOAT 命令

redis < SET mykey "10.50"
redis > OK
redis < INCRBYFLOAT mykey 0.1
```

```
redis > "10.60"
redis < INCRBYFLOAT mykey -5
redis > "5.60"
redis < SET mykey 5.0e3
redis > OK
redis < INCRBYFLOAT mykey 2.0e2
redis > "5200"
==========================================
MGET 命令

redis < SET key1 "Hello"
redis > OK
redis < SET key2 "World"
redis > OK
redis < MSET key1 key2 nonexisting
redis > 1) Hello
redis > 2) World
redis > 3) (nil)
==========================================
MSET 命令

redis < MSET key1 "Hello" key2 "World"
redis > OK
redis < GET key1
[root@VM-52-199-opencloudos use-redis-cxx]# make
g++ -std=c++17 -o string src/string.cc /usr/local/lib64/libredis++.a
/usr/local/lib/libhiredis.a -pthread
[root@VM-52-199-opencloudos use-redis-cxx]# ./string
String 系列命令:
==========================================
SET 命令

redis < EXISTS mykey
redis > 0
redis < SET mykey "Hello"
redis > OK
redis < GET mykey
redis > "Hello"
redis < SET mykey "World" NX
redis > (nil)
redis < DEL mykey
redis > 1
redis < EXISTS mykey
redis > 0
redis < SET mykey "World" XX
redis > (nil)
redis < GET mykey
```

```
redis > (nil)
redis < SET mykey "World" NX
redis > OK
redis < GET mykey
redis > "World"
redis < SET mykey "Will expire in 10s" EX 10
redis > OK
redis < GET mykey
redis > "Will expire in 10s"
Waiting 10s
Waiting 9s
Waiting 8s
Waiting 7s
Waiting 6s
Waiting 5s
Waiting 4s
Waiting 3s
Waiting 2s
Waiting 1s
redis < GET mykey
redis > (nil)
==========================================
GET 命令

redis < GET nonexisting
redis > (nil)
redis < SET mykey "Hello"
redis > OK
redis < GET mykey
redis > "Hello"
redis < DEL mykey
redis > 1
redis < EXISTS mykey
redis > 0
redis < HSET mykey name Bob
redis > 1
redis < GET mykey
redis > WRONGTYPE Operation against a key holding the wrong kind of value
==========================================
APPEND 命令

redis < EXISTS mykey
redis > 0
redis < APPEND mykey "Hello"
redis > 5
redis < GET mykey
redis > "Hello"
```

```
redis < APPEND mykey " World"
redis > 11
redis < GET mykey
redis > "Hello World"
==============================================
GETRANGE 命令

redis < SET mykey "This is a string"
redis > OK
redis < GETRANGE mykey 0 3
redis > This
redis < GETRANGE mykey -3 -1
redis > ing
redis < GETRANGE mykey 0 -1
redis > This is a string
redis < GETRANGE mykey 10 100
redis > string
==============================================
SETEX 命令

redis < SETEX mykey 10 "Hello"
redis > OK
redis < TTL mykey
redis > 10
redis < GET mykey
redis > "Hello"
Waiting 10s
Waiting 9s
Waiting 8s
Waiting 7s
Waiting 6s
Waiting 5s
Waiting 4s
Waiting 3s
Waiting 2s
Waiting 1s
redis < GET mykey
redis > (nil)
==============================================
PSETEX 命令

redis < PSETEX mykey 10000 "Hello"
redis > OK
redis < PTTL mykey
redis > 10000
redis < GET mykey
redis > "Hello"
```

```
Waiting 10s
Waiting 9s
Waiting 8s
Waiting 7s
Waiting 6s
Waiting 5s
Waiting 4s
Waiting 3s
Waiting 2s
Waiting 1s
redis < GET mykey
redis > (nil)
================================================
SETNX 命令

redis < SETNX mykey "Hello"
redis > 1
redis < SETNX mykey "World"
redis > 0
redis < GET mykey
redis > "Hello"
================================================
DECR 命令

redis < EXISTS mykey
redis > 0
redis < DECR mykey
redis > -1
redis < SET mykey "10"
redis > OK
redis < DECR mykey
redis > 9
redis < SET mykey "2342934823904809480293482309 48"
redis > OK
redis > ERR value is not an integer or out of range
redis < SET mykey "not a number"
redis > OK
redis > ERR value is not an integer or out of range
================================================
DECRBY 命令

redis < EXISTS mykey
redis > 0
redis < DECRBY mykey 3
redis > -3
redis < SET mykey "10"
redis > OK
```

```
redis < DECRBY mykey 3
redis > 7
==========================================
INCR 命令

redis < EXISTS mykey
redis > 0
redis < INCR mykey
redis > 1
redis < SET mykey "10"
redis > OK
redis < INCR mykey
redis > 11
redis < SET mykey "234293482390480948029348230948"
redis > OK
redis > ERR value is not an integer or out of range
redis < SET mykey "not a number"
redis > OK
redis > ERR value is not an integer or out of range
==========================================
INCRBY 命令

redis < EXISTS mykey
redis > 0
redis < INCRBY mykey 3
redis > 3
redis < SET mykey "10"
redis > OK
redis < INCRBY mykey 3
redis > 13
==========================================
INCRBYFLOAT 命令

redis < SET mykey "10.50"
redis > OK
redis < INCRBYFLOAT mykey 0.1
redis > "10.60"
redis < INCRBYFLOAT mykey -5
redis > "5.60"
redis < SET mykey 5.0e3
redis > OK
redis < INCRBYFLOAT mykey 2.0e2
redis > "5200"
==========================================
MGET 命令

redis < SET key1 "Hello"
```

```
redis > OK
redis < SET key2 "World"
redis > OK
redis < MSET key1 key2 nonexisting
redis > 1) Hello
redis > 2) World
redis > 3) (nil)
==========================================
MSET 命令

redis < MSET key1 "Hello" key2 "World"
redis > OK
redis < GET key1
redis > "Hello"
redis < GET key2
redis > "World"
==========================================
MSETNX 命令

redis < MSETNX key1 "Hello" key2 "there"
redis > 1
redis < MSETNX key2 "new" key3 "world"
redis > 0
redis < MSET key1 key2 key3
redis > 1) Hello
redis > 2) there
redis > 3) (nil)
==========================================
SETRANGE 命令

redis < SET key1 "Hello World"
redis > OK
redis < SETRANGE key1 6 "Redis"
redis > 11
redis < GET key1
redis > "Hello Redis"
==========================================
SETRANGE 命令

redis < SET mykey "Hello World"
redis > OK
redis < STRLEN mykey
redis > 11
redis < STRLEN nonexisting
redis > 0
==========================================
```

src/list.cc

```cpp
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <chrono>
#include <thread>
#include <iterator>


void testListCommands(sw::redis::Redis& redis) {
    printf("List 系列命令:\n");
    printf("============================================\n");
    {
        printf("LPUSH 命令\n\n");

        redis.flushdb();

        long long c = redis.lpush("mylist", "world");
        printf("redis < LPUSH mylist \"world\"\n");
        printf("redis > %lld\n", c);

        c = redis.lpush("mylist", "hello");
        printf("redis < LPUSH mylist \"hello\"\n");
        printf("redis > %lld\n", c);

        c = redis.lpush("mylist", {"Redis", "MySQL", "MongoDB"});
        printf("redis < LPUSH mylist \"Redis\" \"MySQL\" \"MongoDB\"\n");
        printf("redis > %lld\n", c);

        std::vector<std::string> elements;
        std::back_insert_iterator<std::vector<std::string>> ins =
std::back_inserter(elements);
        redis.lrange("mylist", 0, -1, ins);

        printf("redis < LRANGE mylist 0 -1\n");
        int n = 1;
        for (auto it = elements.begin(); it != elements.end(); ++it) {
            printf("redis > %d) %s\n", n++, it->c_str());
        }

        printf("============================================\n");
    }

    {
        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 < BLPOP 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");
```

```
        int n = 1;
        for (auto it = elements.begin(); it != elements.end(); ++it) {
            printf("redis > %d) %s\n", n++, it->c_str());
        }

        printf("============================================\n");
    }

    {
        printf("LLEN 命令\n\n");

        redis.flushdb();

        long long c = redis.rpush("mylist", "Hello");
        printf("redis < RPUSH mylist Hello\n");
        printf("redis > %lld\n", c);

        c = redis.rpush("mylist", "World");
        printf("redis < RPUSH mylist World\n");
        printf("redis > %lld\n", c);

        c = redis.llen("mylist");
        printf("redis < LLEN mylist\n");
        printf("redis > %lld\n", c);

        printf("============================================\n");
    }
}

int main() {
    sw::redis::Redis redis("tcp://127.0.0.1:6379");

    // List commands
    testListCommands(redis);

    return 0;
}
```

list 测试结果

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

```
==========================================
LPUSH 命令

redis < LPUSH mylist "world"
redis > 1
redis < LPUSH mylist "hello"
redis > 2
redis < LPUSH mylist "Redis" "MySQL" "MongoDB"
redis > 5
redis < LRANGE mylist 0 -1
redis > 1) MongoDB
redis > 2) MySQL
redis > 3) Redis
redis > 4) hello
redis > 5) world
==========================================
LPUSHX 命令

redis < LPUSH mylist World
redis > 1
redis < LPUSHX mylist Hello
redis > 2
redis < LPUSHX myotherlist Hello
redis > 0
redis < LRANGE mylist 0 -1
redis > 1) Hello
redis > 2) World
redis < LRANGE myotherlist 0 -1
redis < (empty array)
==========================================
RPUSH 命令

redis < RPUSH mylist "world"
redis > 1
redis < RPUSH mylist "hello"
redis > 2
redis < RPUSH mylist "Redis" "MySQL" "MongoDB"
redis > 5
redis < LRANGE mylist 0 -1
redis > 1) world
redis > 2) hello
redis > 3) Redis
redis > 4) MySQL
redis > 5) MongoDB
==========================================
RPUSHX 命令

```

```
redis < RPUSH mylist World
redis > 1
redis < RPUSHX mylist Hello
redis > 2
redis < RPUSHX myotherlist Hello
redis > 0
redis < LRANGE mylist 0 -1
redis > 1) World
redis > 2) Hello
redis < LRANGE myotherlist 0 -1
redis < (empty array)
==============================================
LRANGE 命令

redis < RPUSH mylist one
redis > 1
redis < RPUSH mylist two
redis > 2
redis < RPUSH mylist three
redis > 3
redis < LRANGE mylist 0 0
redis > 1) one
redis < LRANGE mylist -3 2
redis > 1) one
redis > 2) two
redis > 3) three
redis < LRANGE mylist -100 100
redis > 1) one
redis > 2) two
redis > 3) three
redis < LRANGE mylist 5 10
redis > (empty array)
==============================================
LPOP 命令

redis < RPUSH mylist one two three four five
redis > 5
redis < LPOP mylist
redis > one
redis < LPOP mylist
redis > two
redis < LPOP mylist
redis > three
redis < LRANGE mylist 0 -1
redis > 1) four
redis > 2) five
==============================================
```

```
BLPOP 命令

redis < EXISTS list1 list2
redis > 0
redis < RPUSH list1 a b c
redis > 3
redis < BLPOP list1 list2 0
redis > 1) list1
redis > 2) a
redis < BLPOP list1 list2 0
redis > 1) list1
redis > 2) b
redis < BLPOP list1 list2 0
redis > 1) list1
redis > 2) c
redis < BLPOP list1 list2 5
redis > (nil)
==============================================
RPOP 命令

redis < RPUSH mylist one two three four five
redis > 5
redis < RPOP mylist
redis > five
redis < RPOP mylist
redis > four
redis < RPOP mylist
redis > three
redis < LRANGE mylist 0 -1
redis > 1) one
redis > 2) two
==============================================
BRPOP 命令

redis < EXISTS list1 list2
redis > 0
redis < RPUSH list1 a b c
redis > 3
redis < BRPOP list1 list2 0
redis > 1) list1
redis > 2) c
redis < BRPOP list1 list2 0
redis > 1) list1
redis > 2) b
redis < BRPOP list1 list2 0
redis > 1) list1
redis > 2) a
```

```
redis < BRPOP list1 list2 5
redis > (nil)
==========================================
LINDEX 命令

redis < LPUSH mylist World
redis > 1
redis < LPUSH mylist Hello
redis > 2
redis < LINDEX mylist 0
redis > Hello
redis < LINDEX mylist -1
redis > World
redis < LINDEX mylist 3
redis > (nil)
==========================================
LINSERT 命令

redis < RPUSH mylist Hello
redis > 1
redis < RPUSH mylist World
redis > 2
redis < LINSERT mylist BEFORE World There
redis > 3
redis < LRANGE mylist 0 -1
redis > 1) Hello
redis > 2) There
redis > 3) World
==========================================
LLEN 命令

redis < RPUSH mylist Hello
redis > 1
redis < RPUSH mylist World
redis > 2
redis < LLEN mylist
redis > 2
==========================================
```

src/set.cc

```
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <chrono>
```

```cpp
#include <thread>
#include <iterator>
#include <unordered_set>


void testSetCommands(sw::redis::Redis& redis) {
    printf("Set 系列命令:\n");
    printf("============================================\n");
    {
        printf("SADD 命令\n\n");
        printf("SMEMBERS 命令\n\n");

        redis.flushdb();

        long long c = redis.sadd("myset", "Hello");
        printf("redis < SADD myset Hello\n");
        printf("redis > %lld\n", c);

        c = redis.sadd("myset", "World");
        printf("redis < SADD myset World\n");
        printf("redis > %lld\n", c);

        c = redis.sadd("myset", "World");
        printf("redis < SADD myset World\n");
        printf("redis > %lld\n", c);

        std::unordered_set<std::string> elements;
        redis.smembers("myset", std::inserter(elements, elements.begin()));
        printf("redis < SMEMBERS myset\n");
        if (elements.size() == 0) {
            printf("redis < (empty array)\n");
        } else {
            int n = 1;
            for (const std::string& e : elements) {
                printf("redis > %d) %s\n", n++, e.c_str());
            }
        }

        printf("============================================\n");
    }

    printf("============================================\n");
    {
        printf("SISMEMBER 命令\n\n");
        redis.flushdb();

        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
```

```
        c = redis.sadd("myset", "three");
        printf("redis < SADD myset three\n");
        printf("redis > %lld\n", c);

        sw::redis::OptionalString v = redis.spop("myset");
        printf("redis < SPOP myset\n");
        if (v) {
            printf("redis > %s\n", v->c_str());
        } else {
            printf("redis > (nil)\n");
        }

        std::unordered_set<std::string> elements;
        redis.smembers("myset", std::inserter(elements, elements.begin()));
        printf("redis < SMEMBERS myset\n");
        if (elements.size() == 0) {
            printf("redis < (empty array)\n");
        } else {
            int n = 1;
            for (const std::string& e : elements) {
                printf("redis > %d) %s\n", n++, e.c_str());
            }
        }

        c = redis.sadd("myset", "four");
        printf("redis < SADD myset four\n");
        printf("redis > %lld\n", c);

        c = redis.sadd("myset", "five");
        printf("redis < SADD myset five\n");
        printf("redis > %lld\n", c);

        std::unordered_set<std::string> poped;
        redis.spop("myset", 3, std::inserter(poped, poped.begin()));
        printf("redis < SPOP myset 3\n");
        if (poped.size() == 0) {
            printf("redis < (empty array)\n");
        } else {
            int n = 1;
            for (const std::string& e : poped) {
                printf("redis > %d) %s\n", n++, e.c_str());
            }
        }

        elements.clear();
        redis.smembers("myset", std::inserter(elements, elements.begin()));
        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.sunion({"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

```
$ make set
g++ -std=c++17 -g -O0 -o set src/set.cc /usr/local/lib64/libredis++.a
/usr/local/lib/libhiredis.a -pthread
$ ./set
Set 系列命令:
============================================
SADD 命令

SMEMBERS 命令

redis < SADD myset Hello
redis > 1
redis < SADD myset World
redis > 1
redis < SADD myset World
redis > 0
redis < SMEMBERS myset
redis > 1) World
redis > 2) Hello
============================================
============================================
SISMEMBER 命令

redis < SADD myset one
redis > 1
redis < SISMEMBER myset one
redis > 1
redis < SISMEMBER myset two
redis > 0
============================================
============================================
SCARD 命令

redis < SADD myset Hello
redis > 1
redis < SADD myset World
redis > 1
```

```
redis < SCARD myset
redis > 2
============================================
============================================
SPOP 命令

redis < SADD myset one
redis > 1
redis < SADD myset two
redis > 1
redis < SADD myset three
redis > 1
redis < SPOP myset
redis > one
redis < SMEMBERS myset
redis > 1) two
redis > 2) three
redis < SADD myset four
redis > 1
redis < SADD myset five
redis > 1
redis < SPOP myset 3
redis > 1) five
redis > 2) four
redis > 3) two
redis < SMEMBERS myset
redis > 1) three
============================================
============================================
SMOVE 命令

redis < SADD myset one
redis > 1
redis < SADD myset two
redis > 1
redis < SADD myotherset three
redis > 1
redis < SMOVE myset myotherset two
redis > 1
redis < SMEMBERS myset
redis > 1) one
redis < SMEMBERS myotherset
redis > 1) three
redis > 2) two
============================================
SREM 命令

```

```
redis < SADD myset one
redis > 1
redis < SADD myset two
redis > 1
redis < SADD myset three
redis > 1
redis < SREM myset one
redis > 1
redis < SREM myset four
redis > 0
redis < SMEMBERS myset
redis > 1) two
redis > 2) three
==========================================
SINTER 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SINTER key1 key2
redis > 1) c
==========================================
SINTERSTORE 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SINTERSTORE key key1 key2
redis > 1
```

```
redis < SMEMBERS key
redis > 1) c
==========================================
SUNION 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SUNION key1 key2
redis > 1) e
redis > 2) b
redis > 3) d
redis > 4) c
redis > 5) a
==========================================
SUNIONSTORE 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SUNIONSTORE key key1 key2
redis > 5
redis < SMEMBERS key
redis > 1) e
redis > 2) b
redis > 3) d
redis > 4) c
redis > 5) a
==========================================
```

```
SDIFF 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SDIFF key1 key2
redis > 1) a
redis > 2) b
==========================================
SDIFFSTORE 命令

redis < SADD key1 a
redis > 1
redis < SADD key1 b
redis > 1
redis < SADD key1 c
redis > 1
redis < SADD key2 c
redis > 1
redis < SADD key2 d
redis > 1
redis < SADD key2 e
redis > 1
redis < SDIFFSTORE key key1 key2
redis > 2
redis < SMEMBERS key
redis > 1) a
redis > 2) b
==========================================
```


src/hash.cc

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

```cpp
#include <string>
#include <chrono>
#include <sw/redis++/redis.h>
#include <sw/redis++/utils.h>
#include <thread>
#include <iterator>
#include <vector>

using std::string;
using std::vector;
using sw::redis::Redis;
using sw::redis::OptionalString;

void testHashCommand(Redis& redis) {
    printf("Hash 系列命令:\n");
    printf("==========================================\n");
    {
        printf("hset 和 hget\n\n");
        redis.flushdb();

        redis.hset("key", "name", "zhangsan");
        printf("redis < hset key name zhangsan\n");
        redis.hset("key", "age", "20");
        printf("redis < hset key age 20\n");

        OptionalString name = redis.hget("key", "name");
        printf("redis < hget key name\n");
        printf("redis > name = %s\n", name->c_str());
        OptionalString age = redis.hget("key", "age");
        printf("redis < hget key age\n");
        printf("redis > age = %s\n", age->c_str());

        printf("==========================================\n");
    }

    {
        printf("hexists 和 hdel\n\n");
        redis.flushdb();

        redis.hset("key", "name", "zhangsan");
        printf("redis < hset key name zhangsan\n");

        bool ok = redis.hexists("key", "name");
        printf("redis < hexists key name\n");
        printf("redis > ok = %d\n", ok);

        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
```

```
redis < hset key name zhangsan
redis < hset key age 20
redis < hmget key
redis > val zhangsan
redis > val 20
==========================================
hlen

redis < hset key name zhangsan
redis < hset key age 20
redis < hlen key
redis > len=2
==========================================
hincrby 和 hincrbyfloat

redis < hset key age 20
redis > hincrby key age 10
redis > age=30
redis > hincrbyfloat key age 0.5
redis > age=30.500000
==========================================
```

src/zset.cc

```cpp
#include <sw/redis++/command_options.h>
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <chrono>
#include <sw/redis++/redis.h>
#include <sw/redis++/utils.h>
#include <thread>
#include <iterator>
#include <vector>

using std::string;
using std::vector;
using sw::redis::Redis;
using sw::redis::OptionalString;
using sw::redis::OptionalLongLong;
using sw::redis::OptionalDouble;

```

```
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;
```

```
        auto it = std::back_inserter(members);
        redis.zrange("key3", 0, -1, it);
        for (const auto& member : members) {
            printf("redis > member=%s, score=%lf\n", member.first.c_str(), member.second);
        }

        printf("============================================\n");
    }
}

int main() {
    Redis redis("tcp://127.0.0.1:16379");
    testZsetCommand(redis);
    return 0;
}
```

zset 测试结果

```
Zset 系列命令:
============================================
zadd 和 zrange

redis < zadd key 100 吕布
redis < zadd key 98 赵云
redis < zadd key 95 典韦
redis < zadd key 92 关羽
redis < zadd key 70 刘备
redis < zrange key 0 4
redis > member=刘备
redis > member=关羽
redis > member=典韦
redis > member=赵云
redis > member=吕布
redis < zrange key 0 4 withscores
redis > member=刘备, score=70.000000
redis > member=关羽, score=92.000000
redis > member=典韦, score=95.000000
redis > member=赵云, score=98.000000
redis > member=吕布, score=100.000000
============================================
zrem 和 zcard

```

```
redis < zadd key 100 吕布
redis < zadd key 98 赵云
redis < zadd key 95 典韦
redis < zadd key 92 关羽
redis < zadd key 70 刘备
redis < zcard key
redis > n=5
redis < zrem key 吕布
redis > n=1
redis < zcard key
redis > n=4
==========================================
zcount

redis < zadd key 100 吕布
redis < zadd key 98 赵云
redis < zadd key 95 典韦
redis < zadd key 92 关羽
redis < zadd key 70 刘备
redis < zcount key 92 100
redis > n=4
==========================================
zpopmax 和 zpopmin

redis < zadd key 100 吕布
redis < zadd key 98 赵云
redis < zadd key 95 典韦
redis < zadd key 92 关羽
redis < zadd key 70 刘备
redis < zpopmax key
redis > member=吕布, score=100.000000
redis < zpopmin key
redis > member=刘备, score=70.000000
==========================================
zrank

redis < zadd key 100 吕布
redis < zadd key 98 赵云
redis < zadd key 95 典韦
redis < zadd key 92 关羽
redis < zadd key 70 刘备
redis < zrank key 吕布
redis > n=4
redis < zrevrank key 吕布
redis > n=0
==========================================
zscore
```

```
redis < zadd key 100 吕布
redis < zscore key 吕布
redis > score=100.000000
==========================================
zincrby

redis < zadd key 100 吕布
redis < zincrby key 10 吕布
redis > score=110.000000
redis < zincrby key -20 吕布
redis > score=90.000000
==========================================
zinterstore

redis < zadd key1 100 吕布
redis < zadd key1 98 赵云
redis < zadd key1 95 典韦
redis < zadd key2 100 吕布
redis < zadd key2 98 赵云
redis < zadd key2 92 关羽
redis < zinterstore key3 key1 key2
redis > n=2
redis > member=赵云, score=196.000000
redis > member=吕布, score=200.000000
==========================================
zunionstore

redis < zadd key1 100 吕布
redis < zadd key1 98 赵云
redis < zadd key1 95 典韦
redis < zadd key2 100 吕布
redis < zadd key2 98 赵云
redis < zadd key2 92 关羽
redis < zunionstore key3 key1 key2
redis > n=4
redis > member=关羽, score=92.000000
redis > member=典韦, score=95.000000
redis > member=赵云, score=196.000000
redis > member=吕布, score=200.000000
==========================================
```

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.

```
#include <sw/redis++/redis++.h>
#include <cstdio>
#include <string>
#include <vector>

using std::string;
using sw::redis::RedisCluster;
using sw::redis::OptionalString;

int main() {
    RedisCluster cluster("tcp://127.0.0.1:6379");

    cluster.set("k1", "111");
    auto value = cluster.get("k1");
    printf("value: %s\n", value->c_str());

    return 0;
}
```


修改 makefile

> 原有内容保持不变.

```
all: generic string list set hash zset cluster

...

cluster: src/cluster.cc
    g++ -std=c++17 -g -O0 -o $@ $^ /usr/local/lib/libredis++.a /lib/x86_64-linux-gnu/libhiredis.a -pthread

.PHONY: clean
clean:
    ...
    rm -rf cluster
```

执行结果

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