

## 一、项目：基于TCP的在线词典

### 1.功能演示

### 2.功能说明

### 3.流程图

### 4.功能实现

### 5.代码实现

## 一、项目：基于TCP的在线词典

### 1.功能演示

```
*****
* 1: register  2: login  3: quit *
*****
please choose : 1
input your name:money
input your password:123456
register : OK
*****
* 1: register  2: login  3: quit *
*****
please choose : 1
input your name:money
input your password:111222
register : user money already exist!!!
*****
* 1: register  2: login  3: quit *
*****
please choose : █
```

```
*****
* 1: register  2: login  3: quit *
*****
please choose : 2
input your name:money
input your password:123
login : name or password is wrongy!!!
*****
* 1: register  2: login  3: quit *
*****
please choose : 2
input your name:money
input your password:123456
login : OK
*****
* 1: query 2: history 3: quit *
*****
please choose : █
```

```

*****
* 1: query  2: history  3: quit *
*****
please choose : 1
-----
input word : happy
      adj. ~ (about/in/with sth/sb) feeling or expressing pleasure, contentment, satisfaction, etc

input word : a
      indef art one

input word : quit
      v. go away from (a place); leave

input word : asdfasd
      not found
input word : █

```

```

input word : #
*****
* 1: query  2: history  3: quit *
*****
please choose : 2
2021-9-10 10:7:35 : happy
2021-9-10 10:7:38 : a
2021-9-10 10:7:39 : quit
2021-9-10 10:8:53 : break
2021-9-10 10:8:56 : hello
*****
* 1: query  2: history  3: quit *
*****

```

## 2.功能说明

大方向一共四个功能：

注册

登录

查询单词

查询历史记录

单词和解释保存在文件中，单词和解释只占一行，

**一行最多300个字节，单词和解释之间至少有一个空格**

也可以先写个程序，将文件中的内容先都插入到数据库中。

```
1 fgets---> sqlite3_exec("INSERT INTO dict VALUES('word','解释')");
```

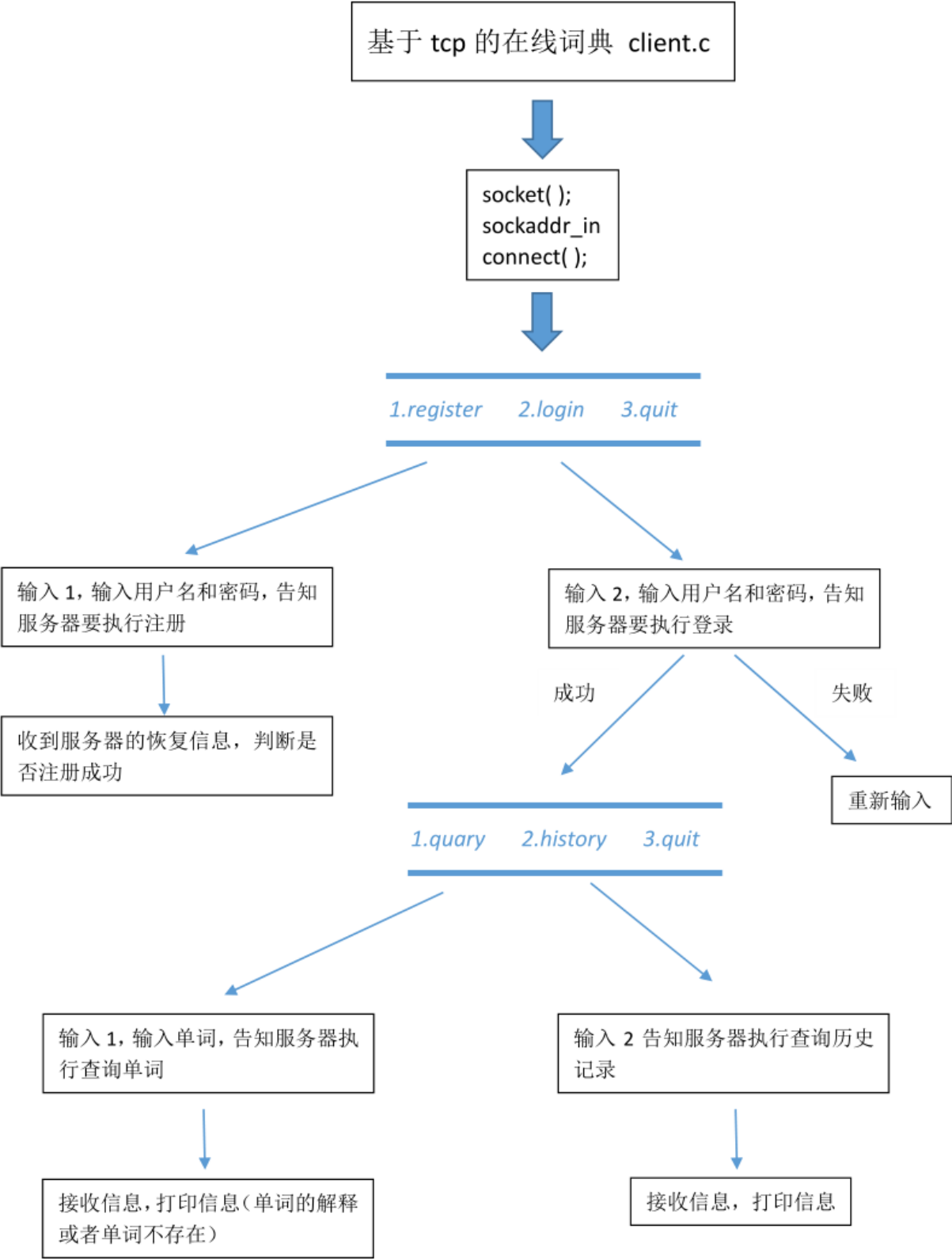
实现TCP并发 --多进程 多线程 io多路复用 均可

建表语句：--在sqlite3终端执行即可

```
CREATE TABLE usr (name TEXT PRIMARY KEY, pass TEXT);
```

```
CREATE TABLE record (name TEXT, date TEXT, word TEXT);
```

3.流程图



# 基于 tcp 的在线词典 server.c

```
sqlite3_open( );  
socket( );  
sockaddr_in  
bind( );  
listen( );  
accept();  
fork( );//实现并发
```

接收数据，做出相应的判断

注册

将接收到的用户  
和密码

将信息插入到数  
据库中

判断是否能够插  
入成功,成功,返  
回 ok, 失败返回  
“用户名已存在”

登录

将接收到的用户  
和密码

判断数据是否存  
在

如果 nrow=0, 则  
不存在, 发送“用  
户名或者密码有  
误”, nrow!=0, 信  
息存在, 发送 ok

查询  
单词

接收到单词

打开文件, 每次  
读取一行 (一行  
300 字节)

对比单词, 找到  
解释, 并将找到  
的单词以及用户  
名和时间保存在  
数据库里面, 发  
送解释给客户  
端, 如果单词不  
存在, 发送“not  
found”

查询  
历史  
记录

根据接收到的用  
户名, 从数据库  
中找到当前用户  
的历史记录, 保  
存并发送给客户  
端

## 4.功能实现

- 1.搭建程序框架
- 2.实现注册和登录功能
- 3.查单词
- 4.查历史记录

## 5.代码实现

服务器：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <sqlite3.h>
6  #include <signal.h>
7  #include <time.h>
8  #include <sys/socket.h>
9  #include <netinet/in.h>
10 #include <arpa/inet.h>
11 #include <sys/wait.h>
12
13 #define N 16
14 #define R 1 // user register
15 #define L 2 // user login
16 #define Q 3 // query word
17 #define H 4 // history record
18
19 #define DATABASE "my.db"
20 typedef struct
21 {
22     int type;
23     char name[N];
24     char data[256]; // password or word
25 } MSG;
26
27 void do_register(int connectfd, MSG *msg, sqlite3 *db);
28 void do_login(int connectfd, MSG *msg, sqlite3 *db);
29 void do_query(int connectfd, MSG *msg, sqlite3 *db);
30 void do_history(int connectfd, MSG *msg, sqlite3 *db);
```

```

31 void do_client(int connectfd, sqlite3 *db);
32 int do_searchword(int connectfd, MSG *msg);
33 void getdata(char data[]);
34 int history_callback(void *arg, int f_num, char **f_value, char **f_name);
35
36 void handler(int sig)
37 {
38     wait(NULL);
39 }
40
41 int main(int argc, char *argv[])
42 {
43     int listenfd, connectfd;
44     struct sockaddr_in server_addr;
45     pid_t pid;
46     sqlite3 *db;
47
48     if (argc < 3)
49     {
50         printf("Usage : %s <ip> <port>\n", argv[0]);
51         exit(-1);
52     }
53     if (sqlite3_open(DATABASE, &db) != SQLITE_OK)
54     {
55         printf("error : %s\n", sqlite3_errmsg(db));
56         exit(-1);
57     }
58     if ((listenfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
59     {
60         perror("fail to socket");
61         exit(-1);
62     }
63
64     memset(&server_addr, 0, sizeof(server_addr));
65     server_addr.sin_family = AF_INET;
66     server_addr.sin_addr.s_addr = inet_addr(argv[1]);
67     server_addr.sin_port = htons(atoi(argv[2]));
68

```

```

69     if (bind(listenfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)

```

```

70     {
71         perror("fail to bind");
72         exit(-1);
73     }
74
75     if (listen(listenfd, 5) < 0)
76     {
77         perror("fail to listen");
78         exit(-1);
79     }
80
81     signal(SIGCHLD, handler); //处理僵尸进程
82
83     while ( 1 )
84     {
85         if ((connectfd = accept(listenfd, NULL, NULL)) < 0)
86         {
87             perror("fail to accept");
88             exit(-1);
89         }
90
91         if ((pid = fork()) < 0)
92         {
93             perror("fail to fork");
94             exit(-1);
95         }
96         else if(pid == 0) //子进程执行处理代码
97         {
98             do_client(connectfd, db);
99         }
100         else //父进程负责连接
101         {
102             close(connectfd);
103         }
104     }
105     return 0;
106 }
107

```

```

108 void do_client(int connectfd, sqlite3 *db)

```

```

109 {
110     MSG msg;
111     while (recv(connectfd, &msg, sizeof(MSG), 0) > 0) // receive request
112     {
113         printf("type = %d\n", msg.type);
114         printf("type = %s\n", msg.data);
115         switch ( msg.type )
116         {
117             case R :
118                 do_register(connectfd, &msg, db);
119                 break;
120             case L :
121                 do_login(connectfd, &msg, db);
122                 break;
123             case Q :
124                 do_query(connectfd, &msg, db);
125                 break;
126             case H :
127                 do_history(connectfd, &msg, db);
128                 break;
129         }
130     }
131     printf("client quit\n");
132     exit(0);
133     return;
134 }
135
136 void do_register(int connectfd, MSG *msg, sqlite3 *db)
137 {
138     char sqlstr[512] = {0};
139     char *errmsg;
140
141     //使用sqlite3_exec函数调用插入函数判断是否能够插入成功
142     //由于用户名设置为主键，所以如果用户名已经存在就会报错
143     sprintf(sqlstr, "insert into usr values('%s', '%s')", msg->name, msg->data);
144     if(sqlite3_exec(db, sqlstr, NULL, NULL, &errmsg) != SQLITE_OK)
145     {
146         sprintf(msg->data, "user %s already exist!!!", msg->name);
147     }

```



```
148     else
149     {
150         strcpy(msg->data, "OK");
151     }
152
153     send(connectfd, msg, sizeof(MSG), 0);
154
155     return;
156 }
157
158 void do_login(int connectfd, MSG *msg, sqlite3 *db)
159 {
160     char sqlstr[512] = {0};
161     char *errmsg, **result;
162     int nrow, ncolumn;
163
164     //通过sqlite3_get_table函数查询记录是否存在
165     sprintf(sqlstr, "select * from usr where name = '%s' and pass = '%s'", msg->name, msg->pass);
166     if(sqlite3_get_table(db, sqlstr, &result, &nrow, &ncolumn, &errmsg) != SQLITE_OK)
167     {
168         printf("error : %s\n", errmsg);
169     }
170     //通过nrow参数判断是否能够查询到记录，如果值为0，则查询不到，如果值为非0，则查询到
171     if(nrow == 0)
172     {
173         strcpy(msg->data, "name or password is wrongy!!!");
174     }
175     else
176     {
177         strncpy(msg->data, "OK", 256);
178     }
179
180     send(connectfd, msg, sizeof(MSG), 0);
181
182     return;
183 }
184
185 void do_query(int connectfd, MSG *msg, sqlite3 *db)
186 {
```

```

187     char sqlstr[128], *errmsg;
188     int found = 0;
189     char date[128], word[128];
190
191     strcpy(word, msg->data);
192
193     //通过found保存查询结果
194     found = do_searchword(connectfd, msg);
195
196     //如果执行成功，还需要保存历史记录
197     if(found == 1)
198     {
199         //获取时间
200         getdata(date);
201         //通过sqlite3_exec函数插入数据
202         sprintf(sqlstr, "insert into record values('%s', '%s', '%s')", msg->name, date, word);
203         if(sqlite3_exec(db, sqlstr, NULL, NULL, &errmsg) != SQLITE_OK)
204         {
205             printf("error : %s\n", errmsg);
206         }
207     }
208
209     send(connectfd, msg, sizeof(MSG), 0);
210
211     return;
212 }
213
214 int do_searchword(int connectfd, MSG *msg)
215 {
216     FILE *fp;
217     char temp[300];
218     char *p;
219     int len, result;
220     //保存单词的长度
221     len = strlen(msg->data);
222     //打开保存单词的文件
223     if((fp = fopen("dict.txt", "r")) == NULL)
224     {
225         strcpy(msg->data, "dict can not open");

```

```

226     send(connectfd, msg, sizeof(MSG), 0);
227 }
228 //printf("query word is %s len = %d\n", msg->data, len);
229
230 //每次读取一行内容
231 int flags = 0;
232 while(fgets(temp, 300, fp) != NULL)
233 {
234     //比较单词
235     result = strncmp(msg->data, temp, len);
236
237     if(result == 0 && temp[len] == ' ')
238     {
239         //p保存单词后面第一个空格的首地址
240         p = temp + len;
241
242         //移动p, 让p保存解释的第一个字符的首地址
243         while(*p == ' ')
244         {
245             p++;
246         }
247
248         //将解释保存在data里面
249         memcpy(msg->data, p, strlen(p));
250
251         fclose(fp);
252         return 1;
253     }
254 }
255
256 strcpy(msg->data, "not found");
257 fclose(fp);
258 return 0;
259 }
260
261 void getdata(char *data)
262 {
263     time_t t;
264
265     struct tm *tp;

```

```

265     time(&t);
266     tp = localtime(&t);
267     sprintf(data, "%d-%d-%d %d:%d:%d", 1900+tp->tm_year, 1+tp->tm_mon, tp->tm_mday, \
268             tp->tm_hour, tp->tm_min, tp->tm_sec);
269 }
270
271 void do_history(int connectfd, MSG *msg, sqlite3 *db)
272 {
273     char sqlstr[128], *errmsg;
274
275     //查询历史表
276     sprintf(sqlstr, "select * from record where name = '%s'", msg->name);
277     if (sqlite3_exec(db, sqlstr, history_callback, (void *)&connectfd, &errmsg) != SQLITE_
278     {
279         printf("error : %s\n", errmsg);
280         sqlite3_free(errmsg);
281     }
282
283     //发送结束标志
284     strcpy(msg->data, "**OVER**");
285     send(connectfd, msg, sizeof(MSG), 0);
286
287     return;
288 }
289
290 //通过回调函数发送时间和单词
291 int history_callback(void *arg, int f_num, char **f_value, char **f_name)
292 {
293     int connectfd;
294     MSG msg;
295     connectfd = *(int *)arg;
296     sprintf(msg.data, "%s : %s", f_value[1], f_value[2]);
297     send(connectfd, &msg, sizeof(msg), 0);
298     return 0;
299 }

```

客户端:

```

1 #include <stdio.h>

```

```

 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 #include <string.h>
 5 #include <sqlite3.h>
 6 #include <sys/socket.h>
 7 #include <netinet/in.h>
 8 #include <arpa/inet.h>
 9
10 #define N 16
11 #define R 1 // user register
12 #define L 2 // user login
13 #define Q 3 // query word
14 #define H 4 // history record
15
16 #define DATABASE "my.db"
17
18 typedef struct
19 {
20     int type;
21     char name[N];
22     char data[256]; // password or word or remark
23 } MSG;
24
25 void do_register(int sockfd, MSG *msg);
26 int do_login(int sockfd, MSG *msg);
27 void do_query(int sockfd, MSG *msg);
28 void do_history(int sockfd, MSG *msg);
29
30 int main(int argc, char *argv[])
31 {
32     int sockfd ;
33     struct sockaddr_in server_addr;
34     MSG msg;
35     if(argc < 3){
36         printf("Usage : %s <serv_ip> <serv_port>\n", argv[0]);
37         exit(-1);
38     }
39     if(-1 == (sockfd = socket(PF_INET, SOCK_STREAM, 0))){
40         perror("fail to socket");

```

```
41     exit(-1);
42 }
43
44 memset(&server_addr, 0, sizeof(server_addr));
45 server_addr.sin_family = PF_INET;
46 server_addr.sin_addr.s_addr = inet_addr(argv[1]);
47 server_addr.sin_port = htons(atoi(argv[2]));
48
49 if(-1 == connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr))) {
50     perror("fail to connect");
51     exit(-1);
52 }
53
54 int choose = 0;
55 while(1)
56 {
57     printf("*****\n");
58     printf("* 1: register   2: login   3: quit *\n");
59     printf("*****\n");
60     printf("please choose : ");
61
62     if(scanf("%d", &choose) <= 0) {
63         perror("scanf");
64         exit(-1);
65     }
66
67     switch(choose) {
68         case 1:
69             do_register(sockfd, &msg);
70             break;
71         case 2:
72             // 执行登录函数，执行完毕后通过返回值决定是否要跳转到下一个菜单
73             if(do_login(sockfd, &msg) == 1) {
74                 goto next;
75             }
76             break;
77         case 3:
78             close(sockfd);
79             exit(0);
```

```

80     }
81 }
82 next:
83 while(1){
84     printf("*****\n");
85     printf("* 1: query   2: history   3: quit *\n");
86     printf("*****\n");
87     printf("please choose : ");
88
89     if(scanf("%d", &choose) <= 0){
90         perror("scanf");
91         exit(-1);
92     }
93     switch(choose){
94         case 1:
95             do_query(socketfd, &msg);
96             break;
97         case 2:
98             do_history(socketfd, &msg);
99             break;
100        case 3:
101            close(socketfd);
102            exit(0);
103    }
104 }
105 return 0;
106 }
107
108 void do_register(int sockfd, MSG *msg){
109     //指定操作码
110     msg->type = R;
111     //输入用户名
112     printf("input your name:");
113     scanf("%s", msg->name);
114     //输入密码
115     printf("input your password:");
116     scanf("%s", msg->data);
117     //发送数据
118     send(sockfd, msg, sizeof(MSG), 0);

```

```

119 //接收数据并输出
120     recv(socketfd, msg, sizeof(MSG), 0);
121     printf("register : %s\n", msg->data);
122
123     return;
124 }
125
126 int do_login(int sockfd, MSG *msg){
127     //设置操作码
128     msg->type = L;
129     //输入用户名
130     printf("input your name:");
131     scanf("%s", msg->name);
132     //输入密码
133     printf("input your password:");
134     scanf("%s", msg->data);
135     //发送数据给服务器
136     send(sockfd, msg, sizeof(MSG), 0);
137     //接收服务器发送的数据
138     recv(sockfd, msg, sizeof(MSG), 0);
139
140     //判断是否登录成功
141     if(strncmp(msg->data, "OK", 3) == 0){ //用3 可以防止 OK 和 OKkshdfkj
142         //登录成功返回1
143         printf("login : OK\n");
144         return 1;
145     }
146
147     //登录失败返回0
148     printf("login : %s\n", msg->data);
149     return 0;
150 }
151
152 void do_query(int sockfd, MSG *msg){
153     msg->type = Q;
154     puts("-----");
155
156     while(1){
157         printf("input word (if # is end): ");

```



```

158     scanf("%s", msg->data);
159
160     //如果输入的是#, 返回上一级
161     if(strcmp(msg->data, "#") == 0){
162         break;
163     }
164
165     send(socketfd, msg, sizeof(MSG), 0);
166
167     recv(socketfd, msg, sizeof(MSG), 0);
168     printf(">>> %s\n", msg->data);
169 }
170 return;
171 }
172
173 void do_history(int sockfd, MSG *msg){
174     msg->type = H;
175     send(socketfd, msg, sizeof(MSG), 0);
176
177     while(1){
178         recv(socketfd, msg, sizeof(MSG), 0);
179
180         if(strcmp(msg->data, "***OVER**") == 0){
181             break;
182         }
183
184         printf("%s\n", msg->data);
185     }
186
187     return;
188 }
189

```

自己写一个能将词典文件中的内容导入数据库的程序

```

1 #include <stdio.h>
2 #include <errno.h>
3 #include <string.h>

```

```

4  #include <stdlib.h>
5  #include <sqlite3.h>
6  #include <unistd.h>
7
8  #define DATABASE "my.db"
9
10 int main(int argc, char const *argv[])
11 {
12     //把文件导入数据库中
13     sqlite3 *db;
14     FILE *fp;
15     if(SQLITE_OK !=sqlite3_open(DATABASE,&db)){
16         perror("sqlite err");
17         exit(1);
18     }
19     fp = fopen("dict.txt","r");
20     if(fp==NULL){
21         perror("err");
22         exit(1);
23     }
24     char str[300]={0};
25     char word[50]={0};
26     char introduct[250]={0};
27     char *errmsg;
28     char sql[500]={0};
29     sprintf(sql,"drop table dict");
30     if(SQLITE_OK !=sqlite3_exec(db,sql,NULL,NULL,&errmsg)){
31         printf("drop table dict error!!!\n");
32     }else{
33         printf("drop table dict yes!!!\n");
34     }
35     sprintf(sql,"create table if not exists dict(word text,translation text)");
36     if(SQLITE_OK !=sqlite3_exec(db,sql,NULL,NULL,&errmsg)){
37         printf("表已经存在!!!\n");
38     }else{
39         printf("表创建成功!!!\n");
40     }
41     int count=0;
42     while ((fgets(str,300,fp))!=NULL)

```

```

43     {
44         //usleep(10000);
45         memset(word, 0, 300);
46         int i=0;
47         char *p=str;
48         str[strlen(str)-1] = '\0';
49         //printf("str = [%s]\n",str);
50         while (*p!=' ')
51         {
52             word[i]=*p;
53             p++;
54             i++;
55         }
56         word[i]='\0';
57         p++;
58         while(*p==' ' && *p != '\0')
59         {
60             p++;
61         }
62
63
64         //处理 两个字段值中的 单引号  sqlite3 数据库 text 字段
65         //不能插入带有单引号的字符串 如 one's 转换成 one.s 再插入
66         char *temp = p;
67         while(*temp != '\0'){
68             if(*temp == '\'){
69                 *temp = '.';
70             }
71             temp++;
72         }
73         temp = word;
74         while(*temp != '\0'){
75             if(*temp == '\'){
76                 *temp = '.';
77             }
78             temp++;
79         }
80

```

```

82     strcpy(introduct, p);
83     introduct[strlen(introduct)-1]='\0';
84     printf("insert count = [%d]\tword:[%s]\t\ttranslation:[%s]\n", count, word, introduct);
85
86     //插入数据
87     sprintf(sql,"INSERT INTO dict (word,translation) VALUES('%s','%s')",word,introduct);
88     int ret =-1;
89     if(SQLITE_OK !=(ret=sqlite3_exec(db,sql,NULL,NULL,&errmsg))){
90         printf("sql err : %s\n", errmsg);
91         exit(1);
92     }
93     count++;
94     memset(str, 0, 300);
95 }
96
97 printf("sum count = %d , process end.....\n", count);
98 return 0;
99 }

```

作业:

- 1.自己把项目的代码梳理一遍, 要求每行都得看懂
- 2.看懂之后, 自己重新实现一次(使用select实现), 代码写的严谨一些
- 3.自己写一个能将词典文件中的内容导入数据库的程序