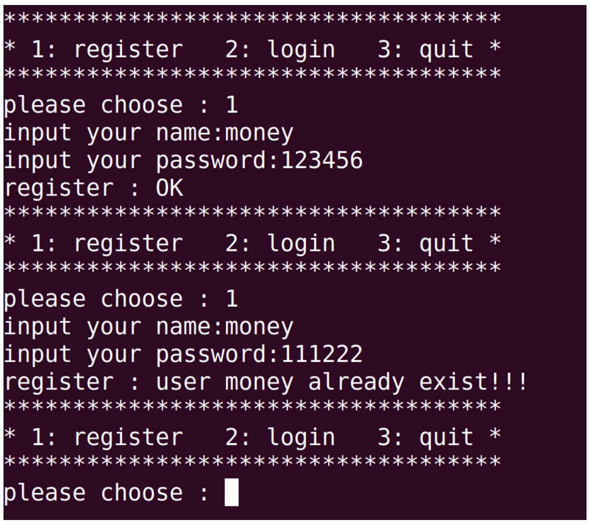
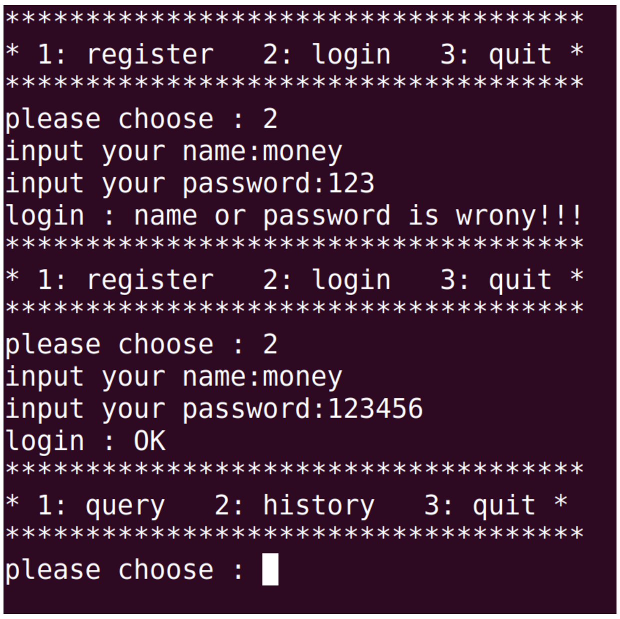
一、项目：基于TCP的在线词典

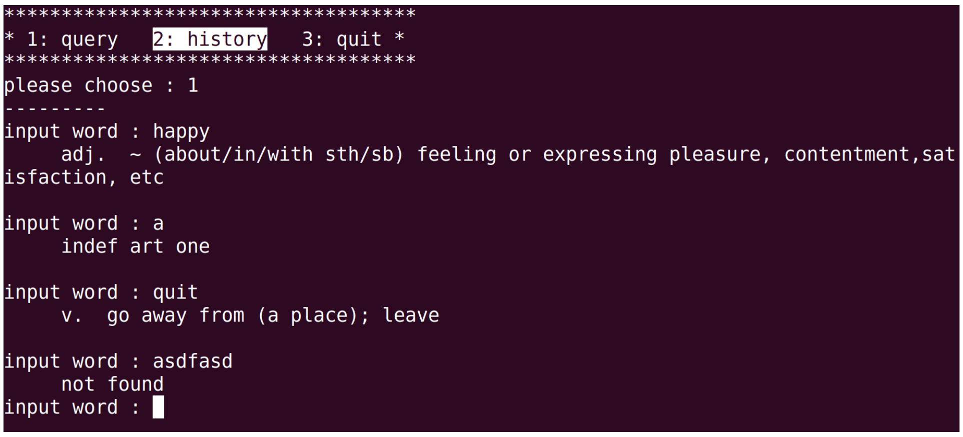
1.功能演示



```
************************************
* 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 wrony!!!
************************************
* 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 : 
```

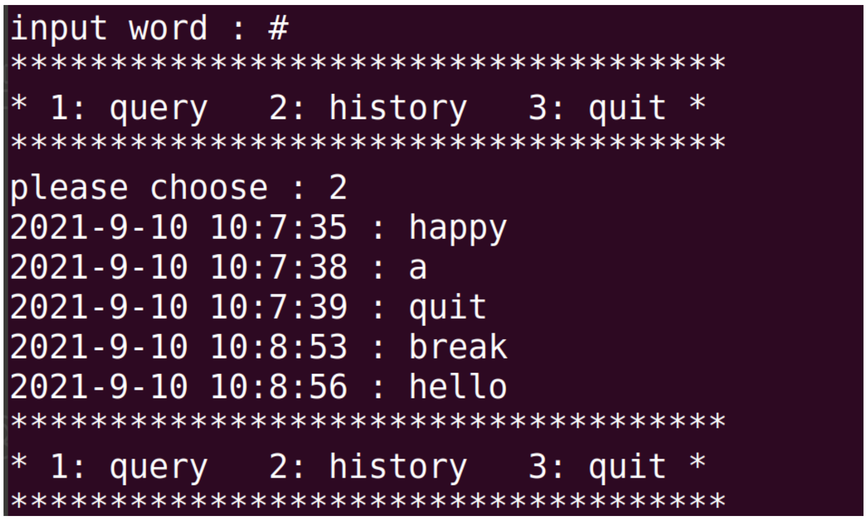


```
************************************
* 1: query   2: history   3: quit *
************************************
please choose : 1
---------
input word : happy
     adj.  ~ (about/in/with sth/sb) feeling or expressing pleasure, contentment,sat
isfaction, etc

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。

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

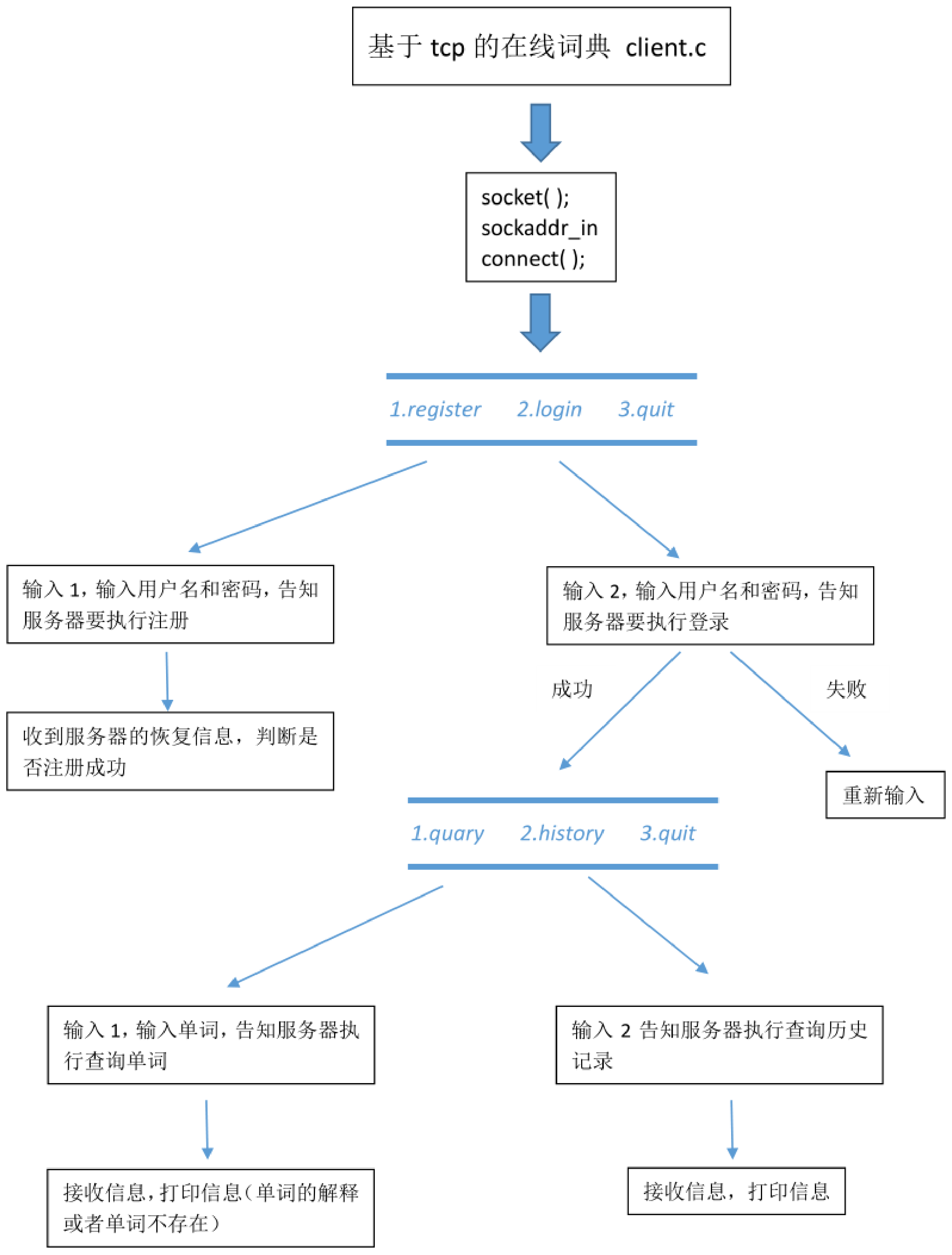
实现TCP并发 --多进程 多线程 io多路复用 均可

建表语句：--在sqlite3终端执行即可

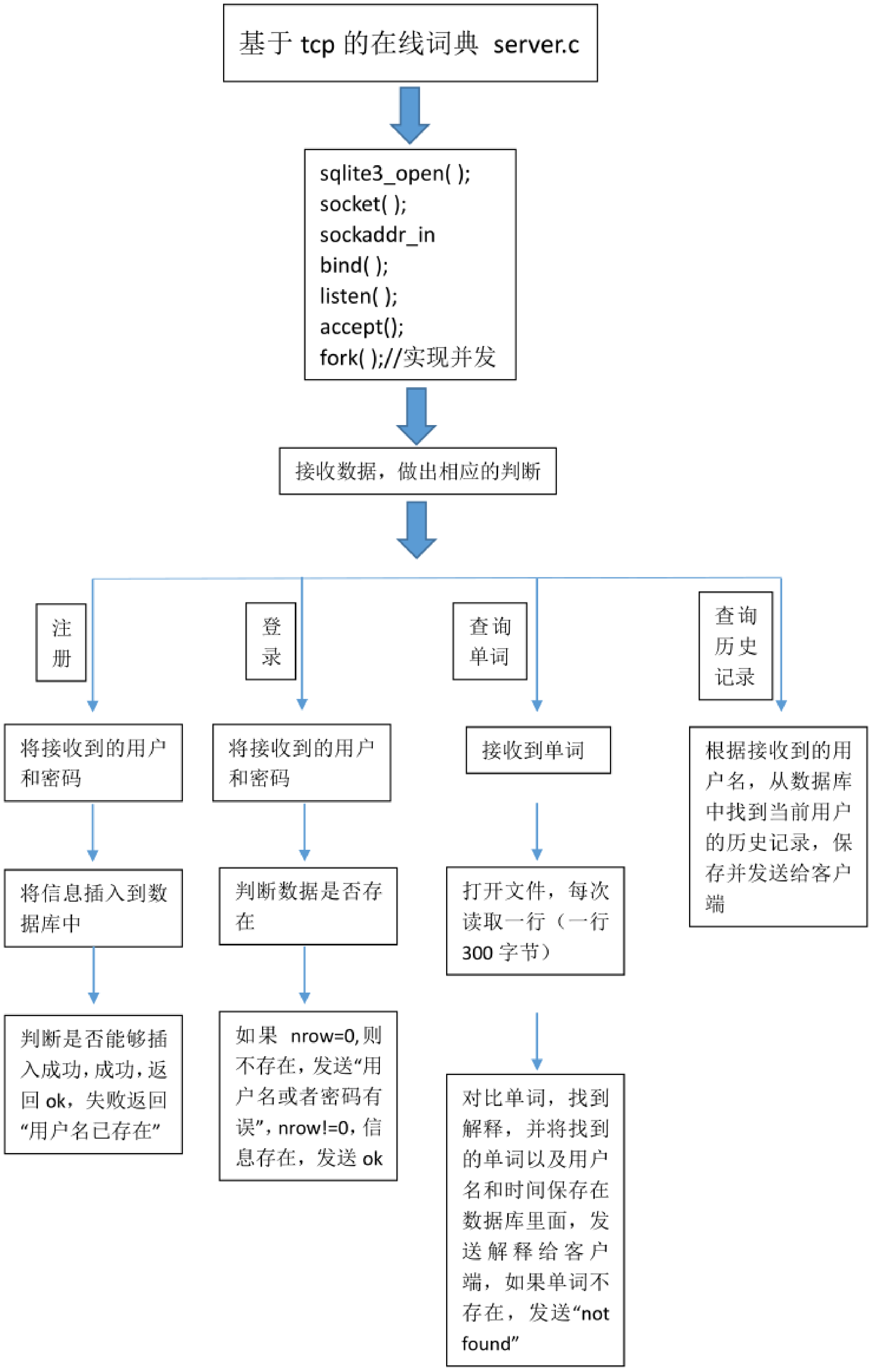
　　CREATE TABLE usr (name TEXT PRIMARY KEY, pass TEXT);

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

3.流程图



基于 tcp 的在线词典 client.c
socket( ); sockaddr_in connect( );
1.register 2.login 3.quit
输入 1，输入用户名和密码，告知服务器要执行注册
收到服务器的恢复信息，判断是否注册成功
输入 2，输入用户名和密码，告知服务器要执行登录
成功
失败
重新输入
1.quary 2.history 3.quit
输入 1，输入单词，告知服务器执行查询单词
接收信息，打印信息（单词的解释或者单词不存在）
输入 2 告知服务器执行查询历史记录
接收信息，打印信息


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
登录
查询单词
查询历史记录
将接收到的用户和密码
将接收到的用户和密码
接收到单词
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
将信息插入到数据库中
判断数据是否存在
打开文件，每次读取一行（一行 300 字节）
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”
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
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”

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:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sqlite3.h>
#include <signal.h>
#include <time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>

#define  N  16
#define  R  1   //  user register
#define  L  2   //  user login
#define  Q  3   //  query word
#define  H  4   //  history record

#define DATABASE "my.db"
typedef struct
{
   int type;
   char name[N];
   char data[256];   // password or word
} MSG;

void do_register(int connectfd, MSG *msg, sqlite3 *db);
void do_login(int connectfd, MSG *msg, sqlite3 *db);
void do_query(int connectfd, MSG *msg, sqlite3 *db);
void do_history(int connectfd, MSG *msg, sqlite3 *db);
void do_client(int connectfd, sqlite3 *db);
int  do_searchword(int connectfd, MSG *msg);
void getdata(char data[]);
int history_callback(void *arg, int f_num, char **f_value, char **f_name);

void handler(int sig)
{
     wait(NULL);
}

int main(int argc, char *argv[])
{
    int listenfd, connectfd;
    struct sockaddr_in server_addr;
    pid_t pid;
    sqlite3 *db;

     if (argc < 3)
    {
        printf("Usage : %s <ip> <port>\n", argv[0]);
        exit(-1);
    }
    if (sqlite3_open(DATABASE, &db) != SQLITE_OK)
    {
        printf("error : %s\n", sqlite3_errmsg(db));
        exit(-1);
    }
    if ((listenfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("fail to socket");
        exit(-1);
    }

    memset(&server_addr,0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(argv[1]);
    server_addr.sin_port = htons(atoi(argv[2]));

```

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

```
   {
       perror("fail to bind");
       exit(-1);
   }

    if (listen(listenfd, 5) < 0)
   {
       perror("fail to listen");
       exit(-1);
   }

    signal(SIGCHLD, handler);//处理僵尸进程

    while ( 1 )
   {
       if ((connectfd = accept(listenfd, NULL, NULL)) < 0)
       {
           perror("fail to accept");
           exit(-1);
       }

        if ((pid = fork()) < 0)
       {
           perror("fail to fork");
           exit(-1);
       }
       else if(pid == 0) //子进程执行处理代码
       {
           do_client(connectfd, db);
       }
        else  //父进程负责连接
        {
           close(connectfd);
        }
   }
   return 0;
}

```

```
void do_client(int connectfd, sqlite3 *db)
```

```c
{
    MSG msg;
    while (recv(connectfd, &msg, sizeof(MSG), 0) > 0)  // receive request
    {
        printf("type = %d\n", msg.type);
        printf("type = %s\n", msg.data);
        switch ( msg.type )
        {
        case R :
            do_register(connectfd, &msg, db);
            break;
        case L :
            do_login(connectfd, &msg, db);
            break;
        case Q :
            do_query(connectfd, &msg, db);
            break;
        case H :
            do_history(connectfd, &msg, db);
            break;
        }
    }
    printf("client quit\n");
    exit(0);
    return;
}

void do_register(int connectfd, MSG *msg, sqlite3 *db)
{
    char sqlstr[512] = {0};
    char *errmsg;

     //使用sqlite3_exec函数调用插入函数判断是否能够插入成功
     //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
    sprintf(sqlstr, "insert into usr values('%s', '%s')", msg->name, msg->data);
    if(sqlite3_exec(db, sqlstr, NULL, NULL, &errmsg) != SQLITE_OK)
    {
        sprintf(msg->data, "user %s already exist!!!", msg->name);
```

```c
    }
    else
    {
        strcpy(msg->data, "OK");
    }

    send(connectfd, msg, sizeof(MSG), 0);

    return;
}

void do_login(int connectfd, MSG *msg, sqlite3 *db)
{
    char sqlstr[512] = {0};
    char *errmsg, **result;
    int nrow, ncolumn;

     //通过sqlite3_get_table函数查询记录是否存在
    sprintf(sqlstr, "select * from usr where name = '%s' and pass = '%s'", msg->name, msg
    if(sqlite3_get_table(db, sqlstr, &result, &nrow, &ncolumn, &errmsg) != SQLITE_OK)
    {
        printf("error : %s\n", errmsg);
    }
     //通过nrow参数判断是否能够查询到疾记录，如果值为0，则查询不到，如果值为非0，则查询到
    if(nrow == 0)
    {
        strcpy(msg->data, "name or password is wrony!!!");
    }
    else
    {
        strncpy(msg->data, "OK", 256);
    }

    send(connectfd, msg, sizeof(MSG), 0);

    return;
}

void do_query(int connectfd, MSG *msg, sqlite3 *db)
{
    char sqlstr[128], *errmsg;
    int found = 0;
    char date[128], word[128];

    strcpy(word, msg->data);

    //通过found保存查询结果
    found = do_searchword(connectfd, msg);

    //如果执行成功，还需要保存历史记录
    if(found == 1)
    {
        //获取时间
        getdata(date);
        //通过sqlite3_exec函数插入数据
        sprintf(sqlstr, "insert into record values('%s', '%s', '%s')", msg->name, date, w
        if(sqlite3_exec(db, sqlstr, NULL, NULL, &errmsg) != SQLITE_OK)
        {
            printf("error : %s\n", errmsg);
        }
    }

    send(connectfd, msg, sizeof(MSG), 0);

    return;
}

int do_searchword(int connectfd, MSG *msg)
{
    FILE *fp;
    char temp[300];
    char *p;
    int len, result;
    //保存单词的长度
     len = strlen(msg->data);
     //打开保存单词的文件
    if((fp = fopen("dict.txt", "r")) == NULL)
    {
        strcpy(msg->data, "dict can not open");
        send(connectfd, msg, sizeof(MSG), 0);
    }
    //printf("query word is %s len = %d\n", msg->data, len);

     //每次读取一行内容
    int flags = 0;
    while(fgets(temp, 300, fp) != NULL)
    {
        //比较单词
        result = strncmp(msg->data, temp, len);

        if(result == 0 && temp[len] == ' ')
         {
            //p保存单词后面第一个空格的首地址
            p = temp + len;

            //移动p，让p保存解释的第一个字符的首地址
            while(*p == ' ')
            {
                p++;
            }

            //将解释保存在data里面
            memcpy(msg->data, p,strlen(p));

            fclose(fp);
            return 1;
         }
    }

    strcpy(msg->data, "not found");
    fclose(fp);
    return 0;
}

void getdata(char *data)
{
    time_t t;
```

```c
    struct tm *tp;
    time(&t);
    tp = localtime(&t);
    sprintf(data, "%d-%d-%d %d:%d:%d",  1900+tp->tm_year,  1+tp->tm_mon,  tp->tm_mday, \
                    tp->tm_hour, tp->tm_min, tp->tm_sec);
}

void do_history(int connectfd, MSG *msg, sqlite3 *db)
{
    char sqlstr[128], *errmsg;

    //查询历史表
    sprintf(sqlstr, "select * from record where name = '%s'", msg->name);
    if (sqlite3_exec(db, sqlstr, history_callback, (void *)&connectfd, &errmsg) != SQLITE_
    {
        printf("error : %s\n", errmsg);
        sqlite3_free(errmsg);
    }

    //发送结束标志
    strcpy(msg->data, "**OVER**");
    send(connectfd, msg, sizeof(MSG), 0);

    return;
}

//通过回调函数发送时间和单词
int history_callback(void *arg, int f_num, char **f_value, char **f_name)
{
    int connectfd;
    MSG msg;
    connectfd = *(int *)arg;
    sprintf(msg.data, "%s : %s", f_value[1], f_value[2]);
    send(connectfd, &msg, sizeof(msg), 0);
    return 0;
}
```

客户端:

```c
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sqlite3.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define  N  16
#define  R  1   //  user register
#define  L  2   //  user login
#define  Q  3   //  query word
#define  H  4   //  history record

#define DATABASE "my.db"

typedef struct
{
   int type;
   char name[N];
   char data[256];   // password or word or remark
} MSG;

void do_register(int socketfd, MSG *msg);
int do_login(int socketfd, MSG *msg);
void do_query(int socketfd, MSG *msg);
void do_history(int socketfd, MSG *msg);

int main(int argc, char *argv[])
{
   int socketfd ;
   struct sockaddr_in server_addr;
   MSG msg;
   if(argc < 3){
       printf("Usage : %s <serv_ip> <serv_port>\n", argv[0]);
       exit(-1);
   }
   if(-1 == (socketfd = socket(PF_INET, SOCK_STREAM, 0))){
       perror("fail to socket");
```

```c
        exit(-1);
    }

    memset(&server_addr,0, sizeof(server_addr));
    server_addr.sin_family = PF_INET;
    server_addr.sin_addr.s_addr = inet_addr(argv[1]);
    server_addr.sin_port = htons(atoi(argv[2]));

    if(-1 == connect(socketfd, (struct sockaddr *)&server_addr, sizeof(server_addr))){
        perror("fail to connect");
        exit(-1);
    }

    int choose = 0;
    while(1)
    {
        printf("************************************\n");
        printf("* 1: register   2: login   3: quit *\n");
        printf("************************************\n");
        printf("please choose : ");

        if(scanf("%d", &choose) <= 0){
            perror("scanf");
            exit(-1);
        }

        switch(choose){
            case 1:
                do_register(socketfd, &msg);
                break;
            case 2:
                 //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
                if(do_login(socketfd, &msg) == 1){
                    goto next;
                }
                break;
            case 3:
                close(socketfd);
                exit(0);
        }
    }
next:
    while(1){
        printf("************************************\n");
        printf("* 1: query   2: history   3: quit  *\n");
        printf("************************************\n");
        printf("please choose : ");

        if(scanf("%d", &choose) <= 0){
            perror("scanf");
            exit(-1);
        }
        switch(choose){
            case 1:
                do_query(socketfd, &msg);
                break;
            case 2:
                do_history(socketfd, &msg);
                break;
            case 3:
                close(socketfd);
                exit(0);
        }
    }
    return 0;
}

void do_register(int socketfd, MSG *msg){
     //指定操作码
    msg->type = R;
     //输入用户名
    printf("input your name:");
    scanf("%s", msg->name);
    //输入密码
     printf("input your password:");
    scanf("%s", msg->data);
    //发送数据
     send(socketfd, msg, sizeof(MSG), 0);
```

```c
    //接收数据并输出
     recv(socketfd, msg, sizeof(MSG), 0);
    printf("register : %s\n", msg->data);

     return;
}

int do_login(int socketfd, MSG *msg){
     //设置操作码
    msg->type = L;
    //输入用户名
     printf("input your name:");
    scanf("%s", msg->name);
    //输入密码
     printf("input your password:");
    scanf("%s", msg->data);
    //发送数据给服务器
     send(socketfd, msg, sizeof(MSG), 0);
    //接收服务器发送的数据
     recv(socketfd, msg, sizeof(MSG), 0);

     //判断是否登录成功
     if(strncmp(msg->data, "OK", 3) == 0){  //用3  可以防止  OK 和 OKkshdfkj
         //登录成功返回1
        printf("login : OK\n");
        return 1;
    }

     //登录失败返回0
    printf("login : %s\n", msg->data);
    return 0;
}

void do_query(int socketfd, MSG *msg){
    msg->type = Q;
    puts("-----------------------------");

    while(1){
        printf("input word (if # is end): ");
```

```
       scanf("%s", msg->data);

        //如果输入的是#，返回上一级
       if(strcmp(msg->data, "#") == 0){
           break;
       }

       send(socketfd, msg, sizeof(MSG), 0);

       recv(socketfd, msg, sizeof(MSG), 0);
       printf(">>> %s\n", msg->data);
   }
   return;
}

void do_history(int socketfd, MSG *msg){
   msg->type = H;
   send(socketfd, msg, sizeof(MSG), 0);

   while(1){
       recv(socketfd, msg, sizeof(MSG), 0);

       if(strcmp(msg->data, "**OVER**") == 0){
            break;
        }

       printf("%s\n", msg->data);
   }

   return;
}

```

自己写一个能将词典文件中的内容导入数据库的程序

```
#include <stdio.h>
#include <errno.h>
```

```
#include <string.h>
```

```c
#include <stdlib.h>
#include <sqlite3.h>
#include <unistd.h>

#define DATABASE "my.db"

int main(int argc, char const *argv[])
{
    //把文件导入数据库中
    sqlite3 *db;
    FILE *fp;
    if(SQLITE_OK !=sqlite3_open(DATABASE,&db)){
        perror("sqlite err");
        exit(1);
    }
    fp = fopen("dict.txt","r");
    if(fp==NULL){
        perror("err");
        exit(1);
    }
    char str[300]={0};
    char word[50]={0};
    char introduct[250]={0};
    char *errmsg;
    char sql[500]={0};
    sprintf(sql,"drop table dict");
    if(SQLITE_OK !=sqlite3_exec(db,sql,NULL,NULL,&errmsg)){
        printf("drop table dict error!!!\n");
    }else{
        printf("drop table dict yes!!!\n");
    }
    sprintf(sql,"create table if not exists dict(word text,translation text)");
    if(SQLITE_OK !=sqlite3_exec(db,sql,NULL,NULL,&errmsg)){
        printf("表已经存在!!!\n");
    }else{
        printf("表创建成功!!!\n");
    }
    int count=0;
```

```c
    while ((fgets(str,300,fp))!=NULL)
    {
        //usleep(10000);
        memset(word, 0, 300);
         int i=0;
         char *p=str;
        str[strlen(str)-1] = '\0';
         //printf("str = [%s]\n",str);
         while (*p!=' ')
         {
             word[i]=*p;
             p++;
             i++;
         }
         word[i]='\0';
         p++;
         while(*p==' ' && *p != '\0')
         {
             p++;
         }


        //处理 两个字段值中的 单引号  sqlite3 数据库  text 字段
        //不能插入带有单引号的字符串   如   one's  转换成   one.s 再插入
        char *temp = p;
        while(*temp != '\0'){
            if(*temp == '\''){
                *temp = '.';
            }
            temp++;
        }
        temp = word;
        while(*temp != '\0'){
            if(*temp == '\''){
                *temp = '.';
            }
            temp++;
        }

        strcpy(introduct, p);
        introduct[strlen(introduct)-1]='\0';
        printf("insert count = [%d]\tword:[%s]\t\ttranslation:[%s]\n", count, word, intr

        //插入数据
        sprintf(sql,"INSERT INTO dict (word,translation) VALUES('%s','%s')",word,introdu
        int ret =-1;
        if(SQLITE_OK !=(ret=sqlite3_exec(db,sql,NULL,NULL,&errmsg))){
            printf("sql err : %s\n", errmsg);
            exit(1);
        }
        count++;
       memset(str, 0, 300);
    }

    printf("sum count = %d , process end......................\n", count);
    return 0;
}
```

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