مستندات پروژه مبانی برنامه نویسی (چت اپلیکیشن)

علی مهدوی فر
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف
بهمن ماه ۱۳۹۸

سمت کاربر (کلاینت)

- توابع فایل *client_header.c* :

```
01 int make_socket ();
```

تابع یک اتصال سوکت به سرور ایجاد می‌کند. آی‌پی *127.0.0.1* و پورت *12345* تنظیم شده. در صورت بروز خطا، متن خطا چاپ می‌شود و برنامه به اتمام می‌رسد. وگرنه، مقدار *client_socket* برگردانده می‌شود.

```
02 void set_color (int k);
```

به کمک توابع کتابخانه ویندوز، رنگ کنسول را طبق مقدار k تنظیم می‌کند.

```
03 void reset_color();
```

به کمک توابع کتابخانه ویندوز، رنگ کنسول را به حالت پیش‌فرض تنظیم می‌کند.

```
04 int cprintf (int k, const char *format, ...);
```

همانند تابع *printf* عمل می‌کند، با این تفاوت که رنگ متن چاپ‌شده طبق مقدار k خواهد بود.

```
05 void gotoline(int n);
```

به کمک تابع *gotoxy* به ابتدای خط n-اُم کنسول می‌رود.

```
06 char make_menu (char* title, int n, ...);
```

یک رشته عنوان، یک عدد n و سپس n رشته گزینه‌ها را می‌گیرد. کاربر می‌تواند به کمک کلیدهای بالا و پایین بین گزینه‌ها جابجا شود و یا با عدد، گزینه مورد نظر را انتخاب کند. گزینه انتخاب شده خروجی تابع است. به عنوان مثال اگر گزینه اول انتخاب شود، مقدار *'1'* باز گردانده می‌شود.

```
07 void _send (const char *format, ...);
```

ابتدا به کمک *sprintf* ورودی‌های فرمت‌شده را در یک رشته می‌ریزد. سپس سوکتی می‌سازد و رشته ساخته شده را برای سرور می‌فرستد.

```
08 char* _recv ();
```

از سرور رشته‌ای دریافت می‌کند و اشاره‌گر به ابتدای رشته را برمی‌گرداند.

- توابع فایل *client.c* :

```
01 int main();
```

تابع *main* یک متغیر *state* با مقدار اولیه ۱ نگه می‌دارد. در یک حلقه بی‌پایان، بنابر مقدار *state* به منوی مربوطه می‌رود و *state* جدید را برابر با خروجی آن منو می‌گذارد.

```
02 int account_menu();      // 1
```

منوی حساب کاربری، که سه گزینه ثبت‌نام، ورود، و خروج از برنامه دارد.

```
03 int register_menu();     // 1.1
```

منوی ثبت‌نام، نام کاربری و پسورد را می‌گیرد و به سرور ارسال می‌کند و پاسخ مناسب را به کاربر می‌دهد و به منوی حساب بر می‌گردد.

```
04 int login_menu();        // 1.2
```

منوی ورود، نام کاربری و پسورد را می‌گیرد و به سرور ارسال می‌کند و پاسخ مناسب را به کاربر می‌دهد. اگر توانست وارد شود به منوی اصلی و درغیراینصورت به منوی حساب می‌رود.

```
05 int main_menu();         // 2
```

منوی اصلی، سه گزینه ساخت کانال یا ورود به کانال یا خروج از حساب را دارد.

```
06 int creat_channel();     // 2.1
```

منوی ساخت کانال، نام کانال را می‌گیرد و به سرور می‌فرستد. در صورت موفقیت وارد کانال شده و به منوی چت می‌رود. وگرنه به منوی اصلی برمی‌گردد.

```
07 int join_channel();      // 2.2
```

منوی ورود به کانال، نام کانال را می‌گیرد و به سرور می‌فرستد. در صورت موفقیت وارد کانال شده و به منوی چت می‌رود. وگرنه به منوی اصلی برمی‌گردد.

```
08 int logout_menu();       // 2.3
```

از حساب کاربری خارج شده و به منوی حساب می‌رود.

```
09 int chat_menu();         // 3
```

منوی چت، چهار گزینه دارد: ارسال پیام، دریافت پیام‌ها، اعضای کانال، خروج از کانال.

```
10 int send_message();      // 3.1
```

پیام را از کاربر دریافت و به سرور ارسال می‌کند. به منوی چت برمی‌گردد.

```
11 int refresh();           // 3.2
```

پیام‌های جدید را از سرور دریافت می‌کند و نمایش می‌دهد. به منوی چت برمی‌گردد.

```
12 int members();           // 3.3
```

اعضای کانال فعال را نشان می‌دهد. به منوی چت برمی‌گردد.

```
13 int search_menu();       // 3.4
```

منوی جستجو، دو گزینه دارد: جستجو بین کاربران، جستجو بین پیام‌ها.

```
14 int search_user();       // 3.4.1
```

یک نام کاربری از ورودی می‌گیرد. با ارسال درخواست و دریافت پاسخ از سرور، اعلام می‌کند که نام کاربری پرسیده‌شده عضو کانال فعال کاربر است یا نه. سپس به منوی چت برمی‌گردد.

```
15 int search_message();    // 3.4.2
```

یک کلمه از ورودی می‌گیرد. با ارسال درخواست و دریافت پاسخ از سرور، پیام‌هایی از کانال که حاوی آن کلمه هستند را نمایش می‌دهد. سپس به منوی چت برمی‌گردد.

```
16 int leave_channel();     // 3.5
```

به اطلاع سرور می‌رساند که کاربر از کانال خارج می‌شود، به منوی اصلی می‌رود.

سمت سرور

- توابع فایل *server_header.c* :

```c
int make_socket();
char* _recv();
void _send (const char *format, ...);
```

این توابع مشابه هِدِر کلاینت تعریف شده اند. با این تفاوت که *make_socket* اینجا *bind* و *listen* هم انجام می‌دهد.

- فایل *server.c* :

استراکت کاربر:

```c
typedef struct {
    char name[100], char pass[100],token[100];
    bool on;     // if user is online/logged in
    cJSON* chnl; // NULL if no chnl
    int seen;
} USER;
```

برای راحتی کار، اطلاعات هر کاربر را در یک استراکت ذخیره می‌کنیم. شامل نام، پسورد، آنلاین بودن، توکن (مشروط به آنلاین بودن)، اشاره‌گر به کانال، و تعداد پیام‌های خوانده شده.

متغییرهای گلوبال فایل *server.c* :

```c
USER   users[MAX];             // array of users
cJSON* chnls[MAX];             // array of channels
int    uMAX = 0  ,  cMAX = 0; // Number of existing users and chnls
char*  req;                    // Pointer to client request
char   token[100];             // token of last user sending request
cJSON* ans;                    // json which will be sent to client
```

توابع فایل *server.c* :

```
01 int main();
```

تابع *main* در سرور ابتدا توسط توابع ۲ و ۳ و ۴ این فایل، دیتابیس سرور را می‌خواند. سپس در یک حلقه بی‌پایان درخواست‌های کلاینت‌ها را در *req* دریافت می‌کند و متناسب با کلیدواژه‌ی درخواست، تابع مناسب را صدا می‌زند (توابع ۹ تا ۱۷). سپس جیسون *ans* را به رشته تبدیل کرده و برای کلاینت می‌فرستد.

```
02 void read_data();
03 void read_user ();
04 void read_chnl ();
```

این توابع، دیتابیش ذخیره‌شده سرور را از دایرکتوری مربوط می‌خوانند و اطلاعات کاربرها و کانال‌ها را در آرایه‌های گلوبال ذخیره می‌کنند.

```
05 void make_token (int i);
```

یک توکن رندوم و منحصر به فرد به طول ۹۹ کاراکتر برای *i*-اُمین کاربر تولید و در استراکت آن ذخیره می‌کند.

```
06 USER* find_token ();
```

به دنبال کاربری می‌گردد که توکن آن برابر با آرایه توکن درخواستی باشد که در متغییر گلوبال ذخیره شده. درصورت یافتن، اشاره‌گر به آن را برمی‌گرداند. اگر کاربر یافت نشد، اشاره‌گر *NULL* برمی‌گرداند.

```
07 void update_channel (cJSON* chnl);
```

فایل دیتابیس کانال داده شده را آپدیت می‌کند، تا درصورت قطع سرور پیام‌ها از بین نروند.

```
08 void add_to_channel (cJSON* chnl, char* sender, char* content);
```

به جیسون مربوط به کانال داده شده، پیامی اضافه می‌کند. رشته‌ی فرستنده و رشته‌ی پیام هم در ورودی آمده است. سپس تابع *update_channel* را فراخوانی می‌کند. این تابع هم فاقد خروجی است.

```
09 bool find (char* str, char* word);
```

یک رشته و یک کلمه می‌گیرد. کلمات رشته را به کمک تابع *strtok* می‌خواند و اگر با کلمه‌ای یکسان با *word* مواجه شد مقدار *true* و در غیر اینصورت *false* را برمی‌گرداند.

```
10 void register_user ();  // 1.1
11 void login_user ();     // 1.2
12 void create_channel (); // 2.1
13 void join_channel ();   // 2.2
14 void logout_user ();    // 2.3
15 void send_message ();   // 3.1
16 void refresh ();        // 3.2
17 void members ();        // 3.3
18 void search_user ();    // 3.4.1
19 void search_message (); // 3.4.2
20 void leave_channel ();  // 3.5
```

این توابع عملکردی مشابه تابع نظیرشان در سرور دارند. به طوری که با خواندنِ *req* ، درخواست کلاینت را تفسیر می‌کنند و به فراخور درخواست، پاسخ مناسب را در جیسون *ans* آماده می‌کنند.

۳. جیسون خانگی!

این بخش به‌گونه‌ای پیاده‌سازی شده که اسامی توابع و استراکت و اجزای آن منطبق بر کتابخانه‌ی داده‌شده باشد تا بدون نیاز به تغییر کدهای سابق بتوان از کتابخانه‌ی جیسونی که خودمان پیاده‌سازی کرده‌ایم بهره ببریم. کتابخانه‌ی شخصی، مشابهاً در دو فایل $cJSON.h$ و $cJSON.c$ قرار داده شده. فایل h. حاوی تعریف استراکت، امضای توابع، و ماکروها هستند. پیاده‌سازی کامل توابع در فایل c. آمده است.

۲۱ استراکت جیسون:

```
22 typedef struct cJSON {
23     struct cJSON* next;
24     struct cJSON* prev;
25 
26     struct cJSON* child;
27 
28     int type;
29 
30     char* valuestring;
31 
32     char* name;
33 } cJSON;
```

یک جیسون دارای این اجزاء است: اشاره‌گری به جیسون قبل و بعد و فرزند خود، یک عدد که نشان‌دهنده‌ی نوع جیسون است (رشته، آرایه، یا شیء)، رشته‌ی مقدار و رشته‌ی نام (درصورت وجود.)

۳۴ ماکروها:

```
35 #define cJSON_String (1 << 4)
36 #define cJSON_Array  (1 << 5)
37 #define cJSON_Object (1 << 6)
```

اگر جزء $type$ از یک جیسون برابر با هر عدد فوق باشد، به این معناست که آن جیسون به ترتیب یک رشته، یک آرایه، یا یک شیء است.

۳۸ توابع:

```
01 cJSON* cJSON_CreateObject (void);
```

این تابع به کمک *malloc* یک جیسون ساخته و اشاره‌گر به آن را باز می‌گرداند. نوع جیسون، شیء خواهد بود.

```
02 cJSON* cJSON_CreateArray (void);
```

به کمک *malloc* یک جیسون ساخته و اشاره‌گر به آن را باز می‌گرداند. نوع جیسون، آرایه خواهد بود.

```
03 cJSON* cJSON_CreateString (char* string);
```

به کمک *malloc* یک جیسون ساخته و اشاره‌گر به آن را باز می‌گرداند. نوع جیسون، رشته خواهد بود.

```
04 cJSON* cJSON_AddItemToObject (cJSON* object, char* name, cJSON* item);
```

ورودی‌ها به ترتیب: جیسون والد، نام جیسون فرزند، و جیسون فرزند. تابع جیسون دومی را فرزند جیسون والد قرار می‌دهد، و اشاره‌گرها را به صورت مناسب تنظیم می‌کند. خروجی، اشاره‌گر به جیسون فرزند است.

```
05 cJSON* cJSON_AddItemToArray(cJSON *array, cJSON *item);
```

مشابه تابع ۴، یک جیسون را فرزند جیسون آرایه‌ای قرار می‌دهد. خروجی، اشاره‌گر به جیسون فرزند است.

```
06 cJSON* cJSON_AddStringToObject (cJSON* object, char* name, char* string);
```

ابتدا یک جیسون با نوع رشته می‌سازد. سپس آن را به عنوان فرزند جیسون والد تنظیم می‌کند. خروجی، اشاره‌گر به جیسون فرزند است.

```
07 cJSON* cJSON_AddArrayToObject (cJSON* object, char* name);
```

ابتدا یک آرایه با نام داده شده و تهی می‌سازد. سپس آن را به عنوان فرزند جیسون والد تنظیم می‌کند. خروجی، اشاره‌گر به آرایه است.

```
08 cJSON* add_child (cJSON* parent, cJSON* item);
```

تابع کمکی که اشاره‌گرها را به گونه‌ای تنظیم می‌کند که جیسون دوم، به فرزندان جیسون اول اضافه شود. خروجی، جیسون فرزند است.

```
09 cJSON* add_brother (cJSON* prev, cJSON* item);
```

تابع کمکی که جیسون دوم را به برادران جیسون اول اضافه می‌کند. خروجی، جیسون دوم است.

```
10 cJSON* cJSON_GetObjectItem (cJSON* object, char* name);
```

درصورتی که جیسون *object* دارای فرزندی با رشته‌ی نامِ *name* بود اشاره‌گر به آن را برمی‌گرداند. وگرنه، اشاره‌گر *NULL* برمی‌گرداند.

```
cJSON* cJSON_GetArrayItem (cJSON *array, int index);
```

اشاره‌گرِ عضو با شماره اندیس *index* از جیسونِ آرایه را برمی‌گرداند. اگر چنین عضوی وجود نداشت، اشاره‌گر *NULL* برمی‌گرداند.

```
int cJSON_GetArraySize (cJSON* array);
```

تعداد اعضای یک جیسون را با شمارش فرزندانش می‌دهد.

```
char* cJSON_Print (cJSON* item);
```

یک استراکت جیسون دریافت می‌کند و رشته‌ی متنی متناظر را در خروجی می‌دهد.

```
void addprint (const char * format, ...);
char* print (cJSON* item);
```

توابع درونی که عمل تبدیل جیسون به رشته را به عهده دارند.

```
cJSON* cJSON_Parse (char* str);
```

با دریافت رشته‌ی متنی، استراکت جیسون متناظر با آن را تولید می‌کند و در خروجی، اشاره‌گر به آن را می‌دهد.

```
cJSON* parse (char* str);
```

تابع درونی که عمل تبدیل رشته‌ی متنی به جیسون را به عهده دارد.

پایان!

شکر که این نامه به عنوان رسید

پیشتر از عمر به پایان رسید

:D