شرح پروژه مبانی برنامه سازی ۱۳۹۸

چت اپلیکیشن تحت C ، فاز اول، دوم و سوم

محمدآرمان سلیمانی

شماره دانشجویی: ۹۸۱۰۵۸۳۵

مقدمه:

هدف از ساخت این پروژه در قدم اول آشنایی با توانایی های C در زمینه شبکه و برقراری ارتباط تحت اینترنت، آشنایی با روش کار با فایل و عملکرد دیتابیس-گونه‌ی کد C و همچنین برقراری ارتباط تحت json میان یک سرور و کلاینت بوده و در قدم دوم جمع‌بندی موارد آموخته شده در درس مبانی برنامه‌سازی می‌باشد.

در این مستند به بررسی عملکرد کلی کلاینت (در زمینه تعامل با کاربر) و نحوه پیاده‌سازی کلاینت و سرور و همچنین کتابخانه شخصی json یعنی فاز سوم می‌پردازیم.

قابلیت های امتیازی افزوده شده در فاز سوم در این مستند بررسی نمی‌گردند چرا که جزو بخش اصلی پروژه محسوب نمی‌شوند.

فاز اول، کلاینت چت

کلاینت چت به‌صورت منو به منو عمل می‌کند و کاربر بسته به منویی که در آن قرار دارد می‌تواند کارهایی را انجام دهد که برخی منجر به عوض شدن منوی او می‌شوند.

از آنجایی که برای هر منو یک اسم در نظر گرفته شده و یک تابع جداگانه برای آن منو وجود دارد، توضیح هر منو در قسمت توضیح تابع آن قرار می‌گیرد.

توابع به ترتیب برخورد کاربر با عملکرد آنها نگاشته می‌شوند و ترتیب آمدن آنها در کد الزاما همین ترتیب نیست.

void initialize()

این تابع یک سوکت برای ارتباط با سرور درست می‌کند و امکان ارتباط با سرور را فراهم می‌کند. چون سرور ساخته شده Stateless است لازم است هربار برای تبادل پیام میان کلاینت و سرور این تابع صدا زده شود.

void account_menu()

در این تابع، کاربر در منوی اول قرار دارد. بسته به ورودی گرفته شده، کاربر می‌تواند از این منو یک حساب کاربری ایجاد کند یا وارد حساب خود شود. برای هریک از این دو کار یک تابع جدا وجود دارد و تابع منوی اکانت یکی از آن دو تابع را صدا می‌زند.

void register_user()

این تابع با دریافت نام کاربری و رمز مورد نظر کاربر، درخواست ساخت حساب را به سرور منتقل می‌کند و پاسخ سرور را به کاربر اطلاع می‌دهد. در صورت موفقیت یا عدم موفقیت، کاربر به منوی اکانت بازگردانده می شود.

void login()

این تابع نام کاربری و رمز عبور کاربر را دریافت می کند، درخواست ورود به حساب را به سرور انتقال می‌دهد و چنانچه ورود موفقیت‌آمیز باشد کاربر را به منوی اصلی انتقال می‌دهد. در صورت عدم موفقیت کاربر به منوی اکانت بازگردانده می‌شود.

void main_menu()

منوی اصلی کلاینت است. از اینجا کاربر می‌تواند درخواست ساخت یا ورود به کانال، خروج از حساب کاربری و یا مشاهده یک جمله حکیمانه را داشته باشد.

با زدن کلید C کاربر می‌تواند نام کانال مورد نظر را وارد کند و درخواست ساخت کانال به سرور ارسال می‌شود. در صورت موفقیت در ساخت کانال، کاربر به کانالی که ساخته است انتقال داده می‌شود.

با زدن کلید J امکان ورود به کانال وجود دارد.

با زدن کلید L کاربر از حساب خود خارج می‌شود.

با زدن کلید Q یکی از ۶ جمله موجود برای کاربر ارسال می‌شود (جملات در کد کلاینت ذخیره شده‌اند و این دستور تنها دستوری‌است که ارتباطی با سرور ندارد)

چنانچه کاربر وارد کانال شود به منوی چت انتقال می‌یابد و در غیر این صورت در این منو باقی‌می ماند. در صورت خروج از حساب، کاربر به منوی اکانت بازگردانده می شود.

void chat_menu()

در صورتی که کاربر در کانالی قرار داشته باشد، این منو برای او قابل مشاهده است. در این منو کاربر می‌تواند افراد حاضر در کانال خود و پیام‌هایی که در کانال تبادل می‌شوند را مشاهده کند و در کانال پیام بفرستد یا از آن خارج شود. در صورتی که کاربر بخواهد پیام بفرستد کلید S را وارد می‌کند و پیام خود را تایپ کرده و با زدن اینتر آنرا ارسال می‌کند.

با خروج از کانال، کاربر به منوی اصلی بازگردانده می‌شود.

void refresh()

این تابع لیست پیام هایی که کاربر از کانال فعلی خود ندیده است را پرینت می‌کند. برای این کار، رشته اطلاعات آمده از سرور را به تابع qTjson_printRefresh(char array[]) پاس می‌دهد و آن تابع خودش کار پرینت پیام ها و نام فرستنده را انجام می‌دهد.

void members_list()

این تابع نیز مانند تابع رفرش، از یک تابع دیگر استفاده می‌کند. این تابع رشته اطلاعات آمده از سرور را به تابع ثانویه qTjson_printMemList(char array[]) پاس می‌دهد و آن تابع لیست اعضا را پرینت می‌کند.

فاز دوم، سرور چت

از آنجایی که این فاز عملا تعاملی با کاربر ندارد به بررسی توابع آن می‌پردازیم.

void response(char buffer[])

این تابع یک رشته را دریافت می‌کند و آنرا برای کلاینت ارسال می‌کند و آنرا در پنجره سرور نیز نمایش می‌دهد. برای اینکه ارسال صورت گیرد باید تابع initialize قبل آن صدا زده شود.

void initialize()

این تابع ابتدا آخرین سوکت ایجاد شده را می بندد سپس یک سوکت جدید برای برقراری ارتباط با کلاینت ایجاد می‌کند و سرور را روی حالت listen قرار می‌دهد.

void read_command()

این تابع در یک while(true) دستورات کلاینت را دریافت کرده و بسته به اینکه دستور چیست، تابع موردنظر را صدا می‌زند. کل دستور ارسال شده از کلاینت به تابع مورد نظر داده می‌شود تا اطلاعات مورد نظر را آن تابع استخراج کند.

void register_user(char buffer[])

نام کاربری و رمز عبور را از رشته دریافتی جدا می‌کند و به آدرس مورد نظر برای تولید یک فایل json برای کاربر می‌رود. چنانچه فایل وجود داشته باشد ارور می دهد – زیرا چنین کاربری وجود دارد.

در غیر این صورت فایل json برای کاربر ایجاد می‌کند و نام کاربری و رمز عبور او را ذخیره می‌کند.

void login_user(char buffer[])

نام کاربری و رمز عبور را از رشته دریافتی جدا کرده و به آدرس فایل کاربر مورد نظر می رود.

اگر فایل کاربر پیدا نشود یا رمز عبور با رمز ثبت شده در فایل مطابقت نداشته باشد ارور می‌دهد ، در غیر این صورت تابع make_auth_token() را صدا می‌زند و از طریق یک رشته گلوبال، یک توکن منحصر به فرد برای کاربر تولید می‌کند و برای او ارسال می‌کند.

همچنین لیست کاربر های آنلاین هنگام لاگین بررسی می‌شود، اگر کاربر آنلاین باشد سرور ارور می‌دهد.

void create_channel(char buffer[])

این تابع نام کانال را از رشته دریافتی خارج می کند و چنانچه token دریافتی نامعتبر باشد ارور می‌دهد.

اگر هم فایل json آن کانال را پیدا کند ارور می‌دهد زیرا آن کانال از قبل وجود دارد. اگر مشکلی پیش نیاید، فایل json کانال را ایجاد کرده و دو پیام در آن می‌فرستد، که اولی پیام ساخته شدن کانال و دومی پیام ملحق شدن عضو اول (سازنده) است.

void join_channel(char buffer[])

این تابع پس از بررسی اعتبار token و وجود کانال، کاربر را به کانال اضافه می‌کند و در فایل json کانال، خبر افزوده شدن کاربر به کانال را اضافه می‌کند.

void send_msg(char buffer[])

این تابع پیام ارسالی کاربر را از رشته دریافتی جدا کرده و پس از بررسی صحت token او و اینکه کاربر در چه کانالی است، یک جیسون دو آیتمه ایجاد می‌کند که در آن نام کاربر به عنوان ارسال کننده و پیام او به عنوان محتوا قرار دارند.

void refresh(char buffer[])

این تابع پس از بررسی اعتبار token کاربر و کانال او، پیام‌هایی که او از کانال ندیده است را چاپ می‌کند.

برای اینکه بدانیم هر کاربر چه پیام هایی را دیده است، یک عدد صحیح به عنوان تعداد پیام‌های دیده شده برای اون ذخیره می‌کنیم و هنگام ارسال پیام‌های کانال به کلاینت، از ارسال پیام هایی با شماره کوچکتر از تعداد پیام‌های دیده شده کاربر، اجتناب می شود.

void members_list(char buffer[])

از آنجایی که لیستی از اینکه هر کانال چه اعضایی دارد در دسترس نیست، برای معرفی اعضای کانال ابتدا دانه‌دانه اعضای آنلاین روی سرور جاروب می‌شوند. هرکدام که عضو کانال مورد نظر باشند، اسمشان به یک آرایه json اضافه می‌شود و در نهایت لیست اعضا به کلاینت ارسال می‌شود.

همانند سایر توابع، این اتفاقات پس از بررسی اعتبار token ارسالی و عضویت یا عدم عضویت کاربر در کانال رخ می دهند.

void leave(char buffer[])

این تابع با بررسی کانال کاربر و نام او، او را از همه متغیرهایی که ذخیره‌کننده کاربران کانال (در واقع کانال هر کاربر) هستند حذف می‌کند و پیام جدایی او از کانال را در فایل کانال چاپ می‌کند. همچنین یک json بیانگر موفقیت به کلاینت ارسال می‌گردد. چنانچه token نامعتبر باشد یا کاربر در کانالی عضو نباشد سرور ارور می‌دهد.

void logout(char buffer[])

این تابع هم مانند leave عمل می‌کند با این تفاوت که کاربر را عملا از همه متغیر های موقت مانند لیست اعضای آنلاین، لیست توکن های آنلاین و ... حذف می‌کند ولی از آنجایی که (طبق شیوه ساخت کلاینت) لازمه اینکه کاربر logout کند این است که در کانالی نباشد، نیازی به چاپ پیام نداریم و صرفا موفقیت این‌کار به کلاینت اعلام می‌شود.

void make_auth_token()

این تابع با استفاده از توابع رندوم موجود در زبان C یک توکن ۳۲ رقمی ایجاد می‌کند و در هنگام لاگین کاربر صدا زده می‌شود تا توکن ایجاد کند. احتمال این که دو توکن ایجاد شده یکسان باشند حدودا برابر ۶۲ به توان منفی ۳۲ است که حدودا برابر ۱۰ به توان منفی ۵۸ است.

int main()

هدف از بررسی main بررسی اتفاقاتی است که در آن رخ می‌دهد.

ابتدا سرور با mkdir فولدر های مورد نظر را تولید می‌کند و سپس تابع read_command صدا زده می‌شود.

فاز سوم، کتابخانه qTjson

البته کتابخانه تولید شده در این فاز عملا برخورد دقیقی با اشیا json ندارد و در واقع به شکل ad hoc برای فاز اول و دوم ساخته شده است (یعنی مخصوص این اپلیکیشن است و قابلیت های آن برای این اپلیکیشن هستند)

کتابخانه qTjson برخی کارها را به شکل استرینگ انجام می‌دهد (که این کارها مخصوص این سرور هستند) ولی امکان ساخت یک linked list برای برخورد شی‌گونه با json را نیز داراست.

توابعی که برخورد stringگونه **دارند و مخصوص این اپلیکیشن هستند:**

char * qTjson_twinMessage(char name1[],char content1[],char name2[],char content2[])

این تابع با دریافت چهار رشته معادل دو item جیسون، یک رشته متناظر با object جیسون دو آیتمی تولید کرده و return می‌کند. مزیت این شیوه این است که بر خلاف نگرش object گونه، می توان با یک خط کد، پیغام خطا، موفقیت و حتی پیام های لیست اعضا و پیام های کانال را تولید و ارسال کرد.

عملکرد تابع با sprint است.

char * qTjson_twinMessageWithArray(char name1[],char content1[],char name2[],char array[])

این تابع مانند تابع قبل است با این تفاوت که برای آخرین رشته ورودی، گیومه در نظر نمی‌گیرد تا با شیوه نگارشjson مطابق باشد.

char * qTjson_createArray()

یک آرایه json خالی در استرینگ تولید کرده و خروجی می‌دهد.

Char * qTjson_appendToArray(char array[],char item[])

یک آرایه و یک آیتم دریافت کرده و با strcat و strcpy آیتم را به انتهای آرایه json اضافه می‌کند و آرایه جدید را Return می‌کند.

Char * qTjson_parseTwin(char array[], int * type)

یک اینت و یک رشته (که با فرمت json دارای دو item باشد) دریافت می‌کند و بصورت pass by reference مقدار اینت را تغییر می‌دهد. چنانچه نوع پیامی که در رشته است error باشد عدد ۰ ، اگر success باشد عدد ۱ و اگر authtoken باشد عدد ۲ روی اینت دریافتی ریخته می‌شود. در واقع این کار برای این است که از تابع به نحوی دو خروجی دریافت کنیم.

برای سایر انواع عدد ۳ ریخته می‌شود.

همچنین content آیتم دوم به عنوان خروجی اصلی Return می‌شود.

(پیاده‌سازی ad hoc در اینجا واضح است، زیرا در دنیای واقعی هر چهار عنصر رشته می‌توانند اهمیت داشته باشند ولی در نحوه پیاده‌سازی کلاینت و سرور، عنصری که باید به طور خاص مشخص باشد content آیتم دوم است و بنابراین سایر عناصر جزو خروجی نیستند.)

void qTjson_printMemList(char array[])

این تابع در واقع پیام سرور را دریافت کرده و بر اساس محل قرارگیری علائم مانند { و ... اعضا کانال را بر روی ترمینال چاپ می‌کند.

Void qTjson_printRefresh(char array[])

این تابع هم مانند تابع قبل، پاسخ سرور را دریافت کرده و بر اساس محل علائم (یعنی شیوه نگاریش json) ابتدا فرستنده هر پیام و سپس پیام او را مقابل نام او چاپ می‌کند.

توابع جامع‌تر که برخورد شی‌گونه با json **دارند**

Typedef struct { ... } qTjson;

این نوع استراکت برای برخورد شی‌گونه با json را مهیا می‌کند. در آن یک اشاره‌گر به بچه‌ی هر استراکت یعنی عنصر بعدی زنجیر وجود دارد و دو رشته که مربوط به اطلاعات همان item هستند. یعنی در این حالت برای اینکه تعداد item هر object ما الزاما ۲ تا نباشد ، یک linked list تشکیل داده‌ایم.

Void qTjson_addItemToObject(qTjson **root, char type_inp[],char content_inp[])

این تابع یک اشاره‌گر دوتایی به شی json دریافت می‌کند (زیرا قصد داریم خود اشاره‌گر را به شکل ارجاعی پاس بدهیم به تابع) و دو رشته مربوط به itemای که قرار است افزوده شود.

در واقع هر شی جیسون برای ما، اشاره‌گر به اولین عنصر linked list مربوط به آن شی است و پاس دادن شی ما به صورت رفرنس در واقع پاس دادن اشاره‌گری به آن اشاره‌گر است.

چنانچه اولین عنصر لیست خالی باشد، آیتم جدید همانجا ریخته می‌شود و در غیر این صورت بچه آن را بررسی می کنیم. اگر بچه خالی باشد در آن ریخته می‌شود و اگر خالی نباشد بچه‌ی بچه بررسی می‌شود و این روند ادامه پیدا می‌کند تا یک عنصر خالی پیدا شود.

Char * qTjson_getObjectItem(qTjson *object , char itemName[])
این تابع از عنصر اول تا قبل اولین جایی که NULL باشد یعنی زنجیر تمام شود را بررسی می‌کند و اولین جایی که یک item پیدا کند که اسم (یا همان type) آن برابر itemName باشد، محتوا (content) آن آیتم را return می‌کند.

qTjson * qTjson_deleteObjectItem(qTjson *root,char itemName[])
این تابع آیتمی که اسمی مانند itemName داشته باشد را پیدا می‌کند و اشاره‌گر به بچه آن را جایگزین اشاره‌گر عنصر پدر آن می‌کند (تا حذف شدن یک حلقه از زنجیر linked list باعث گسستگی آن نشود)