به نام خدا

دانشگاه صنعتی شریف

عنوان:

مستند پروژه مبانی برنامه نویسی

بخش سرور

نویسنده: سید علی نجیبی

بهمن ۱۳۹۸

Functions List

این فایل مستند پروژه مبانی برنامه نویسی میباشد و در آن توابع به کار رفته در قسمت server این برنامه شرح داده شده اند.

۱. General information

این سرور با توجه به کلاینتی که برای آن ساخته شده بود، ساخته شده و عملکردی متناسب با آن دارد. در ابتدا لازم به ذکر است که هر یوسر از یک struct است که در آن اطلاعات: یوسرنیم، پسورد و نام کانالی که یوسر عضو آن است ذخیره میشود که این Struct به وسیله ی توابعی که در زیر شرح داده خواهند شد ذخیره سازی میشود. هر کانال نیز Struct است که تنها شامل نام کانال است که رشته ای با طول معین است که طول آن در ابتدای برنامه define شده است. Struct به نام online تعریف شده است که در آن دو رشته با نام های Token و name و یک Boolean وجود دارد که آرایه ای از این Struct تشکیل دهنده ی لیست کاربران آنلاین ما میباشد. یک Struct دیگر با نام channelInfo نیز تعریف شده که در در آن اطلاعاتی از قبیل تعداد کاربران آنلاین آن کانال، آرایه ای که نام هرکدام از عضوهای کانال در آن ذخیره میشوند و نام کانال وجود دارد. یک struct دیگر نیز با نام DataBase وجود دارد که در آن اطلاعات اصلی دیتابیس از قبیل تعداد یوسرهای ایجاد شده و تعداد کانال های ایجاد شده ذخیره میشود.

دو متغیر گلوبال در برنامه وجود دارد که یکی برای ذخیره ی تعداد یوسر های آنلاین و دیگری برای ذخیره ی کانال های آنلاین[1] تعبیه شده اند. دو آرایه ی گلوبال یکی از جنس online[2] و دیگری از جنس channelInfo میباشند. که با توجه به موارد شرح داده شده، کاربرد هرکدام مشخص است.

۲. Main ()

کار در تابع main با تابع dataBaseCreator شروع میشود. در متد main پس از dataBaseCreator یک سوکت ایجاد میشود و سپس در یک حلقه ی بینهایت هردفعه یکبار سرور به یک client_socket متصل م پس از اتمام receive و send سوکت را میبندد. در این حلقه ی بینهایت با استفاده از تابع Strstr و الگوی هر درخواست، تابع مورد نیاز آن فراخوانی میشود.

۳. dataBaseCreator ()

این تابع در آدرس داده شده به آن جستجو کرده و در صورتی که فایلmainInfo.txt را پیدا کند mainDatabase را با توجه به آن میسازد، در غیر این صورت برنامه نتیجه میگیرد که سرور تابه حال اجرا نشده و یا در آن هیچ یویزری ساخته نشده و mainDatabase را بر اساس آن میسازد.

---

[1] به کانالی اتلاق میشود که حداقل ۱ کاربرآنلاین داشته باشد چراکه هر کانال در زمان بسته شدن سرور آفلاین شده و تمام کاربران پیشین آن حذف میشوند.

[2] منظور **data type** میباشد

۴. **void login (char* request, char response[150000])**

در این تابع با استفاده از pattern که برای login در ابتدای برنامه تعریف شده و تابع sscanf یوسرنیم و پسورد را از request که به سرور ارسال شده استخراج کرده و توابع زیر فراخوانی میشوند و پاسخ مناسب در رشته ی response ذخیره میشود. پس از اطمینان از این که این یوسر وجود دارد که با تابع userExists انجام میشود، برای اطمینان از اینکه یوسر قبلاlogin نکرده تابع findOnlineUserByName فراخوانی میشود. سپس اگر پسورد وارد شده درست بود در آرایه یonlineUsers نام یوسر ذخیره شده و یوسر login میشود و یک authToken به او تعلق میگیرد که برای او ارسال میشود. همچنین فایلی که حاوی پیام هایی است که یوسر زمانی که در آخرین کانال بود refresh نکرده بود پاکسازی شده و آرایه ای خالی در آن قرار میگیرد.

۵. **bool userExists(const char*name)**

این تابع در پوشه ی Resources\Users به دنبال فایلی با یوسرنیم موردنظر میگردد و اگر وجود داشت true و در غیر این صورت false برمیگرداند.

۶. **int findOnlineUserByName(const char* name)**

این تابع در آرایه onlineUsers جستجو کرده و درصورتی که یوسر را پیدا کند که شماره ی خانه آرایه را برمیگرداند و در غیر اینصورت 1- برمیگرداند.

۷. **User findUserByName(const char* name)**

از روی فایلی با نام name اطلاعات یوسر را خوانده و یک User برمیگرداند.

۸. **Void authTokenCreator ()**

با استفاده از آرایه ای که از قبل تعریف شده و اعضای آن حروف a-z,A-Z و اعداد هستند، یک عدد رندم از 0 تا 62 ساخته و عضو با آن شماره را به رشته ای که نمایانگر token است نسبت میدهد و این کار را ۳۲ بار تکرار میکند و این token را به یوسر با شماره یonlineUsercount نسبت داده و این عدد را یک واحد افزایش میدهد.

۹. **void registerUser(char* request, char response[150000])**

این تابع نیز مانند تابع login در ابتدا بررسی میکند که یوسر وجود نداشته باشد و در صورتی که وجود نداشت با استفاده از تابع createNewUser یک یوسر جدید ایجاد میکند. و سپس تعداد یوسرهای ساخته شده را در mainDatabase یک واحد افزایش میدهد و سپس آن را با تابع syncDataBase آپدیت میکند.

۱۰. **void createNewUser(User user)**

user را در فایلی با نام یوسرنیمی که متعلق به آن است درResources/Users/$user.userName.txt ذخیره میکند.

۱۱. **void syncDataBase (DataBase dataBase)**
در فایلResources/MainInfo/mainInfo.txt اطلاعات mainDataBase را ذخیره میکند.

۱۲. **Void createChannel (char* request, char response [150000]**
در ابتدا بررسی میکند با استفاده از توابع بالا که یوسر وجود داشته باشد و AuthToken که به سرور ارسال شده نیز درست باشد و مربوط به یک یوسر آنلاین باشد. سپس با تابع channelExists که عملکردی مشابهuserExists دارد عدم وجود چنین کانالی را بررسی میکند. سپس با استفاده از توابع cJSON آرایه ای ساخته و با استفاده از تابع makeNewMessage یک مسیج به آن اضافه میکند که یوسر مورد نظر این کانال را ایجاد کرده است. سپس با تابع SaveChannelMessages مسیج مورد نظر را ذخیره میکند. و همان مسیج را با تابع saveUserMessages در فایل مربوط به هر یوسر ذخیره میکند. و یوسر و دیتا بیس را آپدیت میکند.

۱۳. **void saveUsersMessages(const char* userName, const char* list)**
در فایلی با نام $username_messages list را ذخیره میکند.

۱۴. **void saveChannelMessages(const char* channelName, const char* list)**
در فایلی با نام $channelName_messages list را ذخیره میکند.

۱۵. **void logout(char* request, char response[150000])**
در این تابع ابتدا موارد مورد نیار بررسی میشوند سپس با استفاده از تابع deleteOnilneUser فرایند logout کردن انجام میشود.

۱۶. **void deleteOnlineUser(int userIndexL)**
در این تابع در آرایه ی onlineUsers ، یوسری که شماره آن برابر userIndexL است را پیدا کرده و یک واحد یوسرهای بعد از آن را شیفت میدهد.

۱۷. **void sendMessage (char* request, char response[150000])**
در این تابع پس از بررسی صحت اطلاعات یوسر، در ابتدا با استفاده از تابع getChannelMessages مسیج های حال حاضر کانال را گرفته پیام مورد نظر را به آن اضافه میکند و با saveChannelMessages آن را ذخیره میکند. سپس با تابع addMessageToMembers آن پیام را به پیامهای هرکدام از کاربران (پیامهای رفرش نشده) اضافه میکند.

۱۸. **void getChannelMessages(const char* name, char buffer[150000])**
فایل مشخصی را باز کرده و از آن پیامهای فرستاده شده در کانال موردنظر را میخواند و در buffer ذخیره میکند.

۱۹. **void addMessageToMembers(const char* message, const char* channelName)**
در یک حلقه که به تعداد کاربران آنلاین کانال انجام میشود، هربار پیامهای مربوط به کاربر با میگیرد و به آن پیام مورد نظر را اضافه کرده و دوباره در فایل مشخصی ذخیره میکند.

۲۰. **const char* makeNewMessage (const char* senderName, const char * contentString)**
با استفاده از cJSON پیام و فرستنده ی آن را به صورت قالبی مشخص درآورده و سپس به صورت یک رشته باز میگرداند.

۲۱. **void leave (char* request, char response[150000])**
با استفاده از توابعی که گفته شد پیغام خارج شدن از گروه را برای تمام اعضا و در کانال ذخیره کرده و سپس با تابع deletMemberFromChannel عضو را از گروه حذف میکند.

۲۲. **void channelMembers(char* request, char response[150000])**
اعضای کانال را با استفاده از یک cJSON به صورت قالبی خاص درآورده و به صورت رشته در response ذخیره میکند.

۲۳. **void joinChannel(char* request, char response[150000])**
مانند createChannel عمل میکند با این تفاوت که پیغام جوین شدن را به تمامی اضا و خود کانال اضافه میکند.

۲۴. **void refresh (char* request, char response[150000])**
ابتدا پیام های موجود در فایل هر کاربر را میگیرد و سپس آن را در یک cJSON_Object اضافه کرده و سپس آن را به صورت رشته در response ذخیره میکند.