

## به نام خدا

فاز یک:

توضیح کدهای مربوط به زیبا سازی در فاز 3 آمده است.

کل این فاز از 3 تابع `make socket` و `chat` و `main` تشکیل شده است، در ابتدای تابع `main` یک بار سوکت ساخته می شود و به سرور متصل میگردد. بعد از آن در قسمت `while` ، با توجه به متغیر `marhale` که مشخص می کند که `client` در کدام مرحله قرار دارد، منوی مربوط به آن قسمت نمایش داده میشود. مرحله یک مربوط به منوی ثبت نام است، مرحله دو مربوط به مرحله ساخت یا ... چنل هاست و مرحله سه هم مربوط به ایجاد و یا دیدن `message` ها است.

از کاربر ورودی `user` که به معنای این است کاربر کدام گزینه را انتخاب کرده گرفته می شود. و تابع `chat` که عملیات فرستادن پیام به سرور و دریافت پیام سرور و تفسیر آن صورت می گیرد.

در تابع `chat` ابتدا با توجه به 2 متغیر `marhale` و `user` موقعیت کلاینت تشخیص داده می شود تا ورودی مورد نیاز از کاربر گرفته شود. بعد از گرفتن ورودی در همان قسمت رشته ای که قرار است کلاینت به سرور بفرستد با توجه به فرمتی که در داک مشخص شده بود ساخته می شود.

به دلیل اینکه برای هر دفعه فرستادن پیام به سرور باید یک سوکت جدید ساخته شود در اینجا تابع `make_socket` صدا زده می شود تا سوکت جدید ساخته شده و به سرور متصل گردد. به از ساخت سوکت، پیام با فرمت مشخص شده به سرور فرستاده می شود و به وسیله تابع کتابخانه ای `recv` پیام سرور دریافت میگردد.

بعد از دریافت پیام توسط کلاینت، با تشخیص موقعیت کلاینت (مثلا اگر مرحله برابر 1 و `user` برابر 2 بود به معنای این است که سرور پاسخ قسمت `login` را داده است) با توجه به الگویی که متن پاسخ سرور که به صورت سی جیسون است دارد و موقعیتی که کلاینت قرار دارد پاسخ سرور تفسیر می شود و متغیر `marhale` دچار تغییر خواهد شد.

بعد از انجام این مراحل برنامه دوباره وارد حلقه در تابع `main` میشود و با توجه به تغییراتی که در متغیر `marhale` داده شده منو مرتبط نشان داده می شود.

فاز دو :

این فاز در مجموع 15 تابع دارد.

آرایه `users` که خانه های آن از یک `struct` به نام `user` به صورت گلوبال تعریف شده که در آن اطلاعات مربوط به کاربران آنلاین نگه داری می شود. رشته `ans` هم به صورت گلوبال تعریف شده که آن در آخر در `buffer` کپی می شود و `buffer` به عنوان پاسخ سرور به کلاینت به آن فرستاده می شود.

تابع `main` :

در این تابع یک حلقه ی درست (`while(1)`) وجود دارد و داخل آن سوکت سرور و کلاینت ساخته می شود. بعد از ساخت آن تابع `chat` صدا زده می شود.

تابع chat : در تابع چت پیام سرور به کلاینت ارسال می شود و همچنین پیام کلاینت را در این تابع دریافت می کند. (هم پیام دریافت شده و هم پیامی که فرستاده می شود در رشته **buffer** وجود دارند) بعد از دریافت پیام کلاینت با توجه به فرمتی که در داک گفته شده بود سرور پیام کلاینت را تفسیر میکند و دستوری که کلاینت به سرور داده است را داخل متغیر **Dastoor** می ریزد. در ادامه با توجه به رشته داخل دستور توابع مختلف صدا زده می شوند.

تابع reg : این تابع **buffer** را به عنوان ورودی میگیرد که همان رشته ای است که کلاینت به سرور فرستاده و از روی آن **user name** و پسورد شخص را جدا میکند. در ادامه در فایلی که تمام **username** ها هستند، نام کاربری شخص را جستجو کرده و اگر چنین فردی وجود داشت سی جیسون مربوط به آن خطا ساخته می شود و داخل متغیر **ans** که آن هم داخل **buffer** ریخته می شود، قرار میگیرد. عملیات ریختن سی جیسن داخل رشته و فرستادن آن به کلاینت در همه ی توابع مانند یکدیگر است و دیگر در توضیح بقیه توابع تکرار نمی شود. اگر هم همچنین نامی موجود نبود سی جیسون مربوط به موفقیت آمیز بودن عملیات ساخته می شود.

تابع login : این تابع مانند قبلی **buffer** را ورودی می گیرد و رمز و نام کاربری شخص را داخل دو آرایه کاراکتر نگه می دارد. فایل مربوط به نام کاربری را باز می کند و نام کاربری شخص را جستجو میکند اگر همچنین نامی وجود داشت و رمز آن با رمز شخص مطابقت داشت تابعی که **auth token** می سازد صدا زده و سی جیسون مربوطه ساخته می شود و در ادامه در آرایه **users** که مربوط به کاربران آنلاین است خانه ای برای آن کاربر پر می شود که شامل نام کاربری، رمز، **auth token** و نام چنلی که در آن عضو هست و دو عضو که نشان میدهند که آیا کاربر آنلاین است و اینکه در چنلی عضو هست یا نه، میشود. در غیر این صورت سی جیسون مربوط به خطا ساخته می شود.

تابع autt : این تابع برای ساختن **auth token** استفاده می شود که در آن به صورت رندم از میان 72 حرف موجود در رشته **auth** یک حرف را انتخاب میکند و داخل رشته **auth\_token** میگذارد و این عملیات 32 بار انجام میگیرد تا یک رشته 32 حرفی از کاراکترهای تصادفی ساخته می شود.

تابع create : این تابع **buffer** را به عنوان ورودی می گیرد و از روی آن نام کانال و **auth token** را جدا میکند در ابتدا در تمامی توابع از اینجا به بعد ابتدا چک میکنند که این **auth token** معتبر است یا خیر و در صورت اعتبار آن ادامه دستورات اجرا می شود در غیر این صورت سی جیسون مربوط به خطا ساخته می شود. سپس تابع چک میکند در فایلی که نام چنل ها نوشته شده آیا همچنین نامی وجود دارد یا نه اگر وجود داشت سی جیسون خطا را می سازد در غیر این صورت نام چنل را اضافه میکند و سی جیسون مربوط را می سازد.

تابع join : این تابع **buffer** را به عنوان ورودی دریافت می کند و نام چنل را از روی **buffer** جدا می کند سپس معتبر بودن نام چنل بررسی می شود اگر معتبر بود کاربر وارد آن کانال شده و اطلاعات مربوط به کانال داخل خانه مربوط به آن کاربر در آرایه پر می شود. در فایلی که با نام کانال وجود دارد عضو شدن این کاربر را می نویسد.

تابع send : این تابع **buffer** را به عنوان ورودی می گیرد و از روی آن **message** کاربر را جدا میکند و داخل فایلی با نام کانال و فرمت مشخص قرار می دهد.

تابع refresh : این تابع **buffer** را به عنوان ورودی می گیرد. سپس از روی فایلی با نام کانال **message** های موجود با فرستنده های آن را می خواند و داخل یک سی جیسون قرار می دهد.

تابع channel members : در این تابع از روی آرایه **users** کاربرانی را که در کانال عضو هستند را پیدا می کند و داخل سی جیسون قرار می دهد.

تابع `leave channel` : در این تابع کاربر از کانال خارج می شود و در خانه ی مربوط به آن کاربر در آرایه نام کانال را حذف می کند و `boolean` مربوط به عضو بودن در کانال را صفر می کند. و در فایلی که با نام کانال وجود دارد خارج شدن این کاربر را اضافه می کند.

تابع `logout` : در این تابع اطلاعات خانه مربوط به کاربر در آرایه از بین می رود.

فاز 3:

در این توابع کتابخانه ای سی جیسون که در فاز 2 استفاده شده بود دیگر استفاده نشده یعنی به صورت دستی رشته ای که باید به صورت فرمت مشخص شده ای به کلاینت فرستاده بشود ساخته می شود که از 4 تابع تشکیل شده است.

تابع `obj` : این تابع ورودی یک پوینتر کاراکتر می گیرد و داخل رشته `{ }` قرار می دهد.

تابع `arr` : این تابع ورودی یک پوینتر کاراکتر میگیرد . داخل رشته `[ ]` قرار می دهد.

تابع `additemtoobj` : این تابع 3 ورودی رشته میگیرد و در اولی با فرمت سی جیسون دو رشته بعدی را اضافه میکند. به طوری که ابتدا چک می کند ببیند که `item` که همان ورودی سوم است باید به صورت یک رشته با آن برخورد شود و " " نیاز دارد یا نه(در سی جیسون برای اضافه کردن رشته از " " استفاده می کند در کد فاز 3 `object` و `array` هم رشته هستند اما نحوه ی اضافه شدن آن به یک `object` یا `array` متفاوت است به همین دلیل این شرط نیاز است) بعد از آن چک می شود که آیا رشته اول 2 کاراکتر دارد که همان `{ }` یا بیشتر برای اینکه اگر بیشتر از 2 کاراکتر باشد باید یک ' ' هم اضافه بشود در ادامه رشته دوم و سوم درون رشته ی اول و درون `{ }` اضافه می شوند.

تابع `additemtoarr` : کاملاً مشابه `additemtoobj` است با این تفاوت که رشته ها باید داخل `[ ]` اضافه بشوند.

قسمت امتیازی:

برای زیبا سازی کد از تابع `Clearconsoleto color` استفاده شده است که کپی شده از اینترنت است و رنگ پس زمینه ی ترمینال و رنگ نوشتار به توسط آن تغییر می کند. برای استفاده از `arrow keys` هم از تابع `gotoxy` استفاده شده که توسط این تابع موقعیت `cursor` تعیین می شود و با فشردن کلید های بالا و پایین، علامت ← در موقعیت مربوطه `print` می شود و جای علامت قبلی " " گذاشته می شود. علاوه بر این دو تابع، از توابع کتابخانه ای `conio.h` و `windows.h` استفاده شده است.

برای دو قابلیت `search message` و `search members` در ترکیب فاز دو و سه، دو تابع با همین نام ها تعریف شده اند که تابع `send message` از دو حلقه همیشه درست تشکیل شده که در اولی پیام های درون فایل خوانده می شوند و در حلقه ی دومی کلمات درون هر پیام که اگر کلمه ای با کلمه ی جستجو شده یکسان بود، آن پیام به سی جیسون در حال ساخت اضافه شود.

در تابع `search members` ابتدا کاربر از روی `auth token` تشخیص داده می شود. سپس افرادی که با آن فرد در یک کانال هستند مشخص شده و در اینجا چک می شود آیا آن افراد با کاربری که کلاینت به سرور فرستاده شده یکی هستند یا خیر و با توجه به آن سرور پاسخ مناسب را بر می گرداند.