مستند پروژه مبانی برنامه نویسی

محمد جواد هزاره
ترم پاییز ۹۸

۱۴ بهمن ۱۳۹۸

فهرست مطالب

۱ توابع مورد استفاده در برنامه

۱.۱ توابع کلاینت

void CreatSocket()

این تابع ورودی و خروجی ندارد، تنها کاری که انجام می دهد ایجاد سوکت کلاینت و اتصال آن به سوکت سرور می باشد. در حقیقت پکیجی از توابع ساخت و اتصال سوکت می باشد.

۲.۱ توابع سرور

به علت تعداد زیاد توابع مورد استفاده در سرور، فقط توابع مهم به اختصار توضیح داده شده و مابقی توابع فقط اسم شان آورده شده است.

void Start()

وظیفه این تابع ایجاد سوکت سرور و پذیرش کلاینت می باشد.

---

void RemoveCama(char* a)

---

int IsThisValid(char* root ,char* item)

از مهم ترین توابع سرور است، وردی آن دو رشته هستند که بررسی می کند آیا رشته ی دوم در مسیر رشته ی اول وجود دارد یا خیر root یا Users است یا Channels که تابع در مسیر Resources/root جست و جو می کند تا فایلی به نام item پیدا کند. جست و جو به این صورت انجام می شود که سعی میکند تا فایلی با نام داده شده را در آدرس مورد نظر به حالت r باز کند، اگر موفق به باز کردن فایل شد می توان نتیجه گرفت که فایل در آن آدرس موجود بوده ولی اگر نتوانست فایلی با نام مورد نظر را باز کند، به این معناست که فایل در آن آدرس موجود نبوده. در حالت دوم اگر فایل موجود نباشد، خود تابع آن فایل را دوباره به حالت w باز می کند تا فایل مورد نظر ایجاد شود. از این تابع برای بررسی وجود کاربر یا کانالی در لیست کاربر ها یا کانال ها استفاده می شود. اگر کاربر یا کانال وجود داشت خروجی ۰ در غیر این صورت خروجی ۱ خواهد بود.

---

int IsUserAndPassValid(char* username, char* pass)

برای بررسی صحت اطلاعات کاربری که قصد ورود دارد مورد استفاده قرار می گیرد. ورودی های آن نام کاربر و پسورد آن می باشد. در اول بررسی می کند که کاربری با این نام وجود دارد یا خیر (انجام این کاربر مطابق روش تابع بالا صورت می گیرد)، سپس اگر وجود داشت فایل آن کاربر که به نام خود او ذخیره شده را باز کرده و این فایل حاوی پسورد او می باشد، این پس.ردی که از فایل خوانده را با پسورد داده شده به تابع مقایسه می کند و اگر این قسمت نیز صحیح بود، تابع چک می کند که کاربر در حال حاضر آنلاین نباشد (از تابع دیگری برای این کار استفاده شده که در زیر آمده است). اگر اطلاعات مورد قبول بود خروجی ۱ در غیر این صورت ۰ خواهد بود.

---

int CheckIsOnline(char* u)

ورودی تابع رشته ای است که نام کاربر می باشد و این تابع چک می کند که کاربر آنلاین است یا خیر. برای این کار روی آرایه ای از استراکت ها که برای کاربران آنلاین ایجاد شده جست و جو می کند و اگر نام کاربری با نام داده شده به تابع تطابق داشت ۱ خروجی می دهد در غیر این صورت ۰.

---

int IsChannelExist(char* channel)

وقتی کاربر می خواهد در کانالی عضو شود مورد استفاده قرار میگیرد، ورودی آن اسم کانال مورد نظر می باشد. الگوریتم جست و جوی آن مانند تابع IsThisValid بوده اما با این تفاوت که اگر کانال وجود داشت خروجی ۱ و در غیر این صورت خروجی ۰ خواهد بود و کانال جدیدی ایجاد نخواهد شد.

---

void SetPass(char* user, char* pass)

پسورد کاربر را در فایلی به نام کاربر ذخیره می کند.

---

void SetUser (char* u, char* t)

اطلاعات کاربر را به آرایه کاربران آنلاین اضافه می کند. این اطلاعات شامل توکن و نام کاربری وی می باشد.

---

void SetUserChannel (char* ch, char* t, char* s)

نام کانالی که کاربر در آن عضو شده یا آن را ایجاد کرده است را به اطلاعاتش در آرایه کاربران آنلاین اضافه میکند. ورودی های این تابع اسم کانال، توکن و رشته ی دیگری است که مشخص می کند کاربر این کانال را ایجاد کرده یا در آن عضو شده است. علاوه بر اضافه کردن اسم کانال به اطلاعات کاربر، پیامی نیز در این کانال با این مضمون که این کاربر به کانال عضو شد یا آن را ایجاد کرد نیز به پیام های کانال مورد نظر می افزاید.

---

void Shift(int i)

---

void SetUserOffline(char* t)

توکن را از کاربر گرفته، در آرایه کاربران آنلاین، استراکت حاوی این توکن را پیدا می کند و تمامی اعضای آرایه که بعد از آن قرار گرفته اند را با تابع Shift یکی به سمت چپ شیفت میدهد. به این ترتیب کاربر مورد نظر از کاربران آنلاین خارج می شود.

---

int CheckToken(char* token)

---

int CheckChannel(char* t)

ورودی آن توکن کاربر است و چک می کند که آیا کانال کاربر در پوشه ی مربوط به کانال ها وجود دارد یا خیر، اگر وجود داشت خروجی ۱ و در غیر این صورت خروجی ۰ خواهد بود.

---

void LeaveChannel(char* t)

توکن کاربر را گرفته، در آرایه کاربران آنلاین او را پیدا میکند و در اطلاعات آن، اسم کانال وی را به NoChannel تغییر می دهد.

---

void GetMyMessage(char* m, char* t)

این تابع برای جدا کردن پیام و توکن استفاده می شود. ورودی های آن دو اشاره گر به کاراکتر هستند که اولی به اولین خانه ی رشته ی پیام و دومی به اولین خانه ی رشته ی توکن اشاره می کند.
روش کار به این صورت است که ابتدا از آخر رشته ی بافر شروع به بررسی می کند تا به کاراکتر ، (کالن) برسد. حال از خانه ی پنجم بافر (چون قالب بافر به صورت send message, token می باشد پیام از خانه ی شماره ۵ تا خانه ای که کالن دارد ادامه پیدا میکند) تا خانه ای که از for بالا بدست آورد را در رشته ای که m به اول آن اشاره می کند کپی می کند.
برای توکن نیز از خانه ای که کالن داشت تا ۳۰ خانه جلوتر (توکن ۳۰ حرفی است) را در رشته ای که t به اولین خانه آن اشاره می کرد کپی می کند.

---

void SendMessageToChannel(char* m, char* t)

با استفاده از توکن، نام کانالی که کاربر در آن عضو است و نام خود کاربر را بدست می آورد و سپس در فایل مربوط به این کانال پیامی به قالب (پیام : فرستنده) ذخیره میکند.

---

void MyJson(char* type, char* content)

یک آبجکت جیسون درست می کند که محتوای تایپ این آبجکت، رشته ی type و محتوای کانتنت آن، رشته ی content خواهد بود.

---

void MakeToken(char* token)

توکن را می سازد. با استفاده از تابع رند عددی بدست آورده، اگر باقی مانده آن به سه برابر ۰ باشد، باقی مانده آن را به ۲۶ گرفته و با ۶۵ جمع میکند به این ترتیب یک حرف بزرگ تولید می شود، اگر باقی مانده به سه برابر ۱ باشد، باقی مانده به ۲۶ گرفته شده و با ۹۷ جمع می شود که در این صورت حرفی کوچک تولید می شود، و اگر باقی مانده بر سه ۲ باشد، باقی مانده به ۱۰ گرفته شده و با ۴۸ جمع می شود که در این صورت یک رقم تولید می شود. این فرآیند ۳۰ بار انجام می شود تا توکن بدست آید.

---

void ShowMessages(char* t)

آبکت حاوی پیام ها را تولید می کند و در بافر ذخیره می کند. با استفاده از توکن به اطلاعات کاربر دسترسی پیدا میکند و از آنجا line که شماره آخرین خطی است که کاربر دیده را پیدا میکند. حال فایل کانال مورد نظر را باز کرده و ابتدا به تعداد line از فایل خط می خواند و در یک رشته بی خودی ذخیره می کند، از آن به بعد هر خط که میخواند را در یک رشته ذخیره می کند، سپس اطلاعات درون این خط که عبارتند از sender و message را جدا کرده و در دو رشته ی جدید ذخیره می کند. حال با این رشته ها یک آبجکت درست کرده و آن را آرایه اضافه می کند و در آخر پس از اتمام فایل این آرایه را به یک آبجکت کلی می افزاید.

---

void ShowMembers(char* t)

آبجکت حاوی اعضای کانال را می سازد و در بافر ذخیره می کند. ابتدا با توکن، نام کانالی که فرد در آن عضو است را بدست می آورد، سپس در آرایه کاربران آنلاین جست و جو می کند و اگر فردی کانالش با نام کانال بدست آمده یکی بود، اسم آن کاربر را به آرایه ای اضافه می کند. در آخر این ارایه را به آبجکتی افزوده و آن را در بافر کپی می کند.

۳.۱ توابع کتابخانه جیسون

۱.۳.۱ توابع کلی

void Delete_MyJSON(cJSON* a)

برای آزاد کردن حافظه اختصاص داده شده به استراکت که در حافظه هیپ قرار دارد استفاده می شود.

---

int Find_State(char* f)

از این تابع برای تعیین حالت بعد از : استفاده می شود. با یک مثال کارکرد آن را توضیح می دهم:
"content" : [whatever] "content" : "whatever"
دو حالت بالا را در نظر بگیرد، اگر حالت اول رخ دهد یعنی بعد از :، " (دابل کوتیشن) بیاید، خروجی تابع ۱ و اگر حالت دوم رخ دهد، خروجی تابع ۲ خواهد بود. ورودی تابع باید پوینتری باشد که به اولین خانه ی رشته یعنی اولین " اشاره کند. کاربرد آن در تعین این است که وقتی می خواهیم یک آیتم از یک آبجکت را دریافت کنیم، for ما باید تا کجا ادامه پیدا کند (توضیح دقیق این که کاربردش چیست در عمل و در توابع بخش کلاینت جیسون آشکار می شود.)

۲.۳.۱ توابع کلاینت جیسون

cJSON* cJSON_Parse(char* t)

این تابع یک استراکت از نوع cJSON درست کرده و رشته ی ورودی را در عضو text این استراکت کپی می کند و اشاره گر به این استراکت را بر میگرداند.

---

cJSON* cJSON_GetObjectItem(cJSON* ptr, char* t)

دو ورودی دارد، اولی اشاره گر به یک استراکت cJSON است و دومی اشاره گر به کاراکتر. در محتویات عضو text استراکت داده شده به آن، به دنبال زیر رشته ای که به صورت "رشته داده شده" می باشد می گردد، سپس استراکتی جدید ساخته و با پیدا شدن این زیر رشته، اگر value این آیتم از آبجکت ما خود نیز رشته باشد، رشته ی value را بدون دابل کوتیشن هایش در valuestring استراکت جدید ذخیره کرده (اصولا این کار را چه value رشته باشد چه آبجکت انجام می دهد ولی اگر آبجکت باشد، valuestring مورد استفاده قرار نمی گیرد) و کل آیتم را در text این استراکت ذخیره می کند. پیدا کردن زیر رشته با تابع strstr انجام شده است. این جا مهم است که value این آیتم خود رشته است یا آرایه، برای فهمیدن این موضوع از تابع Find_State استفاده می شود، اگر رشته بود تا زمانی که به خط بعد برود for را ادامه میدهیم و اگر آرایه باشد تا زمان رسیدن به کاراکتر [.

---

cJSON* cJSON_GetArrayItem(cJSON* arr, int i)

این تابع عضو امi آرایه داده شده را در text یک استراکت جدید ذخیره می کند. اگر عضو امi خود از نوع رشته باشد، متن درون رشته را نیز در valuestring این استراکت جدید که ساخته، ذخیره می کند (اصولا این کار را چه value رشته باشد چه آبجکت انجام می دهد ولی اگر آبجکت باشد، valuestring مورد استفاده قرار نمی گیرد). روش کار به این صورت است که دو اشاره گر داریم یک find و دیگری move که اولی به ابتدای عضو امi و دومی به انتهای این عضو اشاره میکند. هدف بدست آوردن این دو اشاره گر است. ابتدا با strstr اول آرایه را پیدا کرده و در find می ریزیم، در همین جا مقدار move را هم همین مقدار قرار می دهیم. سپس با یک for روی کاراکتر های آرایه حرکت میکنیم move) را یکی یکی زیاد میکنیم) تا به (،) برسیم، البته اگر اعضای آرایه خود آبجکت باشند باید چک کنیم که قبل از این کاراکتر حتما کاراکتر (کروشه بسته) آمده باشد و اگر اعضا رشته باشند نیازی به این بررسی نیست. با رسیدن به (،) مطابق آنچه گفته شد، یک شمارنده که از ــ۱ شروع شده را یک زیاد میکنیم، حال اگر این شمارنده با i برابر بود که find و move را پیدا کرده ایم و اما اگر برابر نبود، مقدار move+۱ را در find ریخته و به حرکت خود روی کاراکتر های آرایه ادامه می دهیم تا زمانی که یا شرط برابر شدن شمارنده با i برقرار شود و یا به انتهای آرایه (کاراکتر براکت بسته) برسیم.

---

int cJSON_GetArraySize(cJSON* arr)

روش کار اینگونه است که ابتدا باید تعین کنیم که آرایه چه دارد؛ رشته، آبجکت یا کلن خالی است. برای این کار اولین کاراکتر (براکت باز) را پیدا کرده و با توجه به کاراکتر بعد از آن می توان فهمید این آرایه چه دارد! اگر خالی بود که ۰ ریترن می شود. اگر رشته بود تعداد ، ها را شمرده و تعداد آنها به اضافه یک ریترن می شود. اگر آبجکت باشد نیز تعداد ، ها شمرده می شود اما چک می شود که قبل از این کاراکتر حتما کاراکتر (کروشه بسته) آمده باشد، در این صورت جزو شمارش حساب می شود، در غیر اینصورت خیر. پس از شمارش تعداد ، به اضافه یک ریترن می شود.

۳.۳.۱ توابع سرور جیسون

cJSON* cJSON_CreateObject()

این تابع ورودی ندارد، یک استراکت از نوع cJSON می سازد، اولین حرف عضو text آن را کاراکتر } گذاشته و پوینتر به این استراکت را خروجی می دهد.

---

cJSON* cJSON_CreateArray()

ورودی نداشته، یک استراکت از نوع cJSON ساخته و اولین کاراکتر text آن را کاراکتر (براکت باز) گذاشته و پوینتر به این استراکت را خروجی می دهد.

---

cJSON* cJSON_CreateString(char* str)

این تابع اشاره گر به کاراکتر (رشته) ورودی گرفته، استراکتی از نوع cJSON درست میکند. سپس رشته ای به قالب "str" ساخته و آن را در text استراکت ساخته شده کپی می کند و پوینتر به این استراکت را خروجی می دهد.

---

void EndObject(cJSON* a)

انتهای آبجکت کاراکتر (کروشه بسته) می گذارد. سپس کاراکتر قبلی آن را که ، می باشد به بک اسلش n تغییر می دهد.

---

void EndArray(cJSON* b)

انتهای آرایه کاراکتر(براکت بسته) می گذارد. سپس این کاراکتر را یکی به چپ شیفت می دهد تا به جای ، قرار بگیرد.

---

void EndObjectwithArray(cJSON* a)

این تابع انتهای آبجکتی را می بندد که دارای آرایه باشد. روش کار به این صورت است که کاراکتر (براکت بسته) را پیدا کرده و کاراکتر بعد آن را (بک اسلش n) گذاشته و کاراکتر بعد از این کاراکتر را (کروشه بسته) می گذارد.

---

void cJSON_AddItemToObject(cJSON* a, char* item, cJSON* b)

این تابع یک آیتم با نام "item" را به آبجکت a اضافه میکند. value این آیتم محتوای text اشاره گر b خواهد بود. روش کار استفاده از تابع strcat است. به این صورت که اول قالب "item" ساخته شده، سپس : به آن افزوده می شود و در مرحله آخر محتوای text اشاره گر b به انتهای آن افزوده می شود.

---

void cJSON_AddItemToArray(cJSON* b, cJSON* a)

محتوای text اشاره گر a را با استفاده از strcat به انتهای محتوای trxt اشاره گر b می افزاید.

---

char* cJSON_Print(cJSON* a)

خروجی آن یک اشاره گر به کاراکتر است که همان اشاره گر به اولین کاراکتر عضو text استراکت a خواهد بود.