گزارش پروژه اول

شبکه های کامپیوتری

امیر فراهانی ۸۱۰۱۹۸۵۷۲

امید پنکاری ۸۱۰۱۹۸۵۲۸

مقدمه

در این پروژه به طراحی یک سرور FTP با استفاده از برنامه‌نویسی ساکت (Socket) می‌پردازیم. پروژه از دو بخش اصلی Client و Server تشکیل شده است که این دو بخش با استفاده از دو کانال ساکت به یکدیگر متصل شده‌اند. یکی از کانال‌ها برای انتقال دستورهای کاربر از client به server و برگرداندن پاسخ استفاده می‌شود و کانال دیگر نیز برای انتقال دادن داده‌های مورد نیاز استفاده می‌شود.

Client

وظیفه client گرفتن دستورها از کاربر و بررسی آن‌ها و سپس صدا کردن متد مناسب از server با استفاده از socket است. پس از آن server نتیجه‌ی اجرای دستور را با استفاده از socket برای client می فرستد و client نیز با توجه به آن خروجی مناسب را به کاربر نمایش می‌دهد. در ادامه این کار ها با جزییات بیشتر توضیح داده می‌شوند.

وصل شدن به server

در ابتدا وقتی برنامه client اجرا می‌شود باید به server متصل شود. برای این کار ابتدا دو socket ساخته می‌شود که هر دو از نوع TCP هستند و یکی برای دستورات و یکی برای داده است. اطلاعات مورد نیاز برای وصل شدن به server مانند port مربوط به دو socket گفته شده ، از فایل config.json که در client قرار دارد خوانده می‌شوند و این کار توسط کلاس JsonSerializer و در درون constructor کلاس clientانجام می‌شود. سپس با استفاده از این اطلاعات و دو socket ساخته شده و فراخوانی سیستمی connect به socket های سمت سرور متصل می‌شویم. پس از وصل شدن به server از هر دو کانال، باید به نحوی server را مطلع کنیم که هردوی کانال‌های باز شده مربوط به یک client هستند به همین علت بلافاصله پس از وصل شدن server یک کد را از طریق کانال دستورات برای client می‌فرستد که clientنیز همان کد را دوباره برای server ولی از طریق کانال داده ارسال می‌کند. به این نحو server می‌فهمد که هردوی این کانال‌ها از طرف یک client هستند. در اینجا Client به طور کامل به server متصل شده است.

اجرای دستورات

پس از گرفتن دستورات از کاربر در تابع HandleCommand دستور داده شده بررسی می‌شود تا نوع آن مشخص شود. پس از مشخص شدن نوع دستور تابع مربوطه به آن صدا می‌شود. حال باید متد مربوط به این دستور را با پارامترهای مورد نیاز آن از سمت serverصدا کنیم. برای این کار با استفاده از کلاس JsonSerializer یک رشته‌ی Json ساخته می‌شود که دارای یک فیلد method که تابع سمت server برای اجرا را مشخص می‌کند و همچنین پارامترهای مورد نیاز برای اجرای آن متد می‌باشد. سپس این رشته‌ی ساخته شده توسط فراخوانی سیستمی send به server فرستاده می‌شود و پاسخ آن نیز با استفاده از فراخوانی سیستمی recv دریافت می‌شود. پاسخ دریافت شده نیز به مانند درخواست یک رشته‌ی json می‌باشد. این رشته توسط JsonSerializer باز می‌شود. در پاسخ آمده از سمت server یک فیلد code وجود دارد که نتیجه‌ی عملیات را نشان می‌دهد. اطلاعات مورد نیاز دیگر (سایز داده‌های ارسالی و مسیر کاری و...) نیز درون این پاسخ فرستاده شده موجود می‌باشند. Client پس از بررسی کد فرستاده شده خروجی مناسب مربوط به آن کد را به کاربر نمایش می‌دهد.

دریافت داده

اگر دستوری که کاربر وارد کرده دستور retr (دستور مربوط به دریافت فایل) یا ls(گرفتن لیست فایل‌ها و پوشه‌ها در مسیر فعلی) باشد، client پس از دریافت پاسخ server کد آن را چک می‌کند و اگر با توجه به کد عملیات موفقیت آمیز باشد، سایز داده نیز برای آن فرستاده شده بنابراین ما باید داده‌ای به آن اندازه از کانال داده دریافت کند. سپس با توجه به سایز داده شده به تعداد مناسب packet از سمت server دریافت می‌کند تا جایی که مجموع سایز packetهای ارسال شده برابر سایز داده شود. در نهایت نیز اگر داده از نوع فایل باشد، فایل را در پوشه‌ی download در کنار client ذخیره می‌کند.

Server

Server خود از دوکلاس ServerApi و ServerCore تشکیل شده است. ServerApi نوعی لایه اولیه است که وظیفه‌ی ارتباط با clientها و تعامل با آن ها را دارد. ServerCore نیز یک لایه‌ی مرکزی است عملیات‌های اصلی را انجام می‌دهد.

ServerApi

ساخت socketها

کلاس ServerApi در ابتدا باید socketهای مورد نیاز را بسازد. به مانند client، socketها ساخته می‌شوند. سپس هر socket با توجه به portای که از فایل config.json خوانده می‌شود و با استفاده از فراخوانی سیستمی bind به آن port وصل می‌شود. سپس هردو socket با استفاده از فراخوانی سیستمی listen شروع به گوش کردن به port داده شده برای گرفتن درخواست‌ها می‌کنند.

گرفتن درخواست‌ها و ارسال پاسخ

پس از آن که دو socket شروع به گوش کردن کردند، دو fd_set به نام‌های readSet و writeSet ساخته می‌شوند که هر کدام شامل مجموعه‌ای از fdهای (File Descriptor) مربوط به Clientها هستند. readSet برای گرفتن پیغام از client از طریق fdهای درون آن استفاده می‌شود و writeSet نیز در جهت نوشتن برروی یک fd استفاده می‌شود. در ابتدا تنها fd مربوط به خود دو Socket ساخته شده در server درون readSet قرار دارد که درخواست‌های مربوط به اتصال به این socketها از طرف clientها دریافت شود. سپس دو fd_set ساخته شده به فراخوانی سیستمی select داده می‌شوند. این تابع برنامه را تا زمانی که یکی از fdهای درون readSet اماده خواندن باشد و یا یکی از fdهای درون writeSet آماده گرفتن اطلاعات باشد برنامه را بلاک می‌کند و به این صورت برنامه برای خواندن از یک fd که آماده نوشتن نیست یا بالعکس، معطل نمی‌شود. پس از آن با استفاده از تابع FD_ISSET می‌توانیم چک کنیم که کدام fdها آماده هستند.

در اینجا ۳ حالت به وجود می‌آید:

- اگر یک درخواست برای خود fdهای server برای اتصال آمده باشد، با استفاده از فراخوانی سیستمی accept برای client جدید fd ساخته می‌شود و آن client به server متصل می‌شود. سپس اگر این client به Socket دستورات وصل شده باشد، ابتدا fd آن را به writeSet اضافه می‌کنیم تا بتوانیم کد fd آن را برای خود client بفرستیم که بتواند طبق فرآیند گفته شده در قسمت client آن را برای ما از طریق کانال داده بفرستد. به طور مشابه اگر درخواست اتصال برای کانال داده بود، fdساخته شده را به readSet اضافه می‌کنیم تا بتوانیم کدی را که برای ما می‌فرستد را دریافت کنیم. برای ذخیره کردن اطلاعات clientهای متصل شده از یک داده‌ساختار به نام Client استفاده می‌شود که دارای اطلاعاتی مثل fd کانال داده، پیامی که قرار است برای Client فرستاده شود و بافر دانلود می‌شود.
- ممکن است یک درخواست از طرف fd مربوط به client ها برای server آمده باشد. اگر این درخواست برای کانال دستورات باشد به این معنا است که یک دستور آمده است. دستورات در فرمت json فرستاده می‌شود. تابع HandleRequest با نگاه کردن به فیلد method از این json نوع درخواست را می‌فهمد و تابع مورد نیاز آن را از ServerCore صدا می‌کند. سپس نتیجه‌ی نهایی آن تابع را به حالت یک رشته‌ی json در می‌آورد و client fd را در writeSet قرار می‌دهد تا پاسخ را برای آن بفرستد. اگر دستور آمده برای گرفتن داده باشد علاوه بر این کارها داده مورد نظر به صورت تعدادی packet در می‌آید و در بافر آن کلاینت قرار می‌گیرد و همچنین fd مربوط به کانال داده‌ی این client در writeSet قرار می‌گیرد. اگر درخواست داده شده برای کانال داده باشد یعنی آن که clientکد ابتدایی را فرستاده است.
- ممکن است که یکی از Clientها آماده خواندن باشد که در این صورت اگر این نوشتن برای کانال دستورات باشد،پیامی که در داده‌ساختار Client ذخیره شده برای آن فرستاده می‌شود. اگر نوشتن مربوط به کانال داده باشد، آخرین packetای که درون بافر client باشد برایش فرستاده می‌شود.

ServerCore

این کلاس در اصل هسته‌ی مرکزی server است و تمام دستورات در اصل در آن اجرا می‌شود. به ازای هر کدام از عملیات‌ها یک متد در درون این کلاس وجود دارد که کار آن را انجام می‌دهد. هر کدام از متدها یک کد را به عنوان خروجی بر می‌گردانند که نتیجه‌ی اجرای

آن عملیات را مشخص می‌کند. متدهایی که خروجی های دیگر نیز دارند، خروجی خود را در قالب یک داده سختار می‌دهند که علاوه بر کد گفته شده خروجی‌های دیگر را نیز در بر می‌گیرد.

برای مدیریت کاربرهای سیستم از یک داده ساختار به نام User استفاده می‌شود که اطلاعات کاربر مانند نام‌کاربری، پسورد، ادمین بودن و حجم مجاز کاربر را شامل می‌شود. این کاربرها در ابتدای کار از فایل config.json خوانده می‌شوند. علاوه بر آن یک داده‌ساختار به نام OnlineUser نیز ساخته شده که برای مدیریت کاربران آنلاین استفاده می‌شود. درحقیقت هربار یک Client در server با یک نام‌کاربری و پسورد وارد می‌شود یک OnlineUser برای آن ساخته می‌شود. کاربرهای آنلاین با استفاده از عدد fd مربوط به Client که در ServerApi به دست می‌آید شناخته می‌شوند.

سایر کلاس‌ها

JsonSerializer

این کلاس وظیفه‌ی خواندن فایل‌های json و یا رشته‌ی json دریافت اطلاعات از آن‌ها یا بالعکس را دارد. اطلاعات در قالب یک map استخراج می‌شوند. همچنین برای ساخت json نیز باید یک map ساخته شود.

Logger

وظیفه این کلاس نوشت لاگ‌ها در فایل گفته شده است. متد Log از این کلاس یک متن را دریافت می‌کند و بعد از اضافه کردن زمان و تاریخ به آن، آن را در فایل لاگ می‌نویسد. از این کلاس در تمام طول برنامه استفاده شده است.

Utility

وظیفه این کلاس در بر داشتن تابع‌های کمکی است که در دیگر کلاس‌ها استفاده شده‌اند. به مانند تابع split که در جهت جداسازی یک رشته از آن استفاده می‌شود و تابع ToString است که اعداد را به رشته تبدیل می‌کند.