پروژه ی اول درس شبکه های کامپیوتری

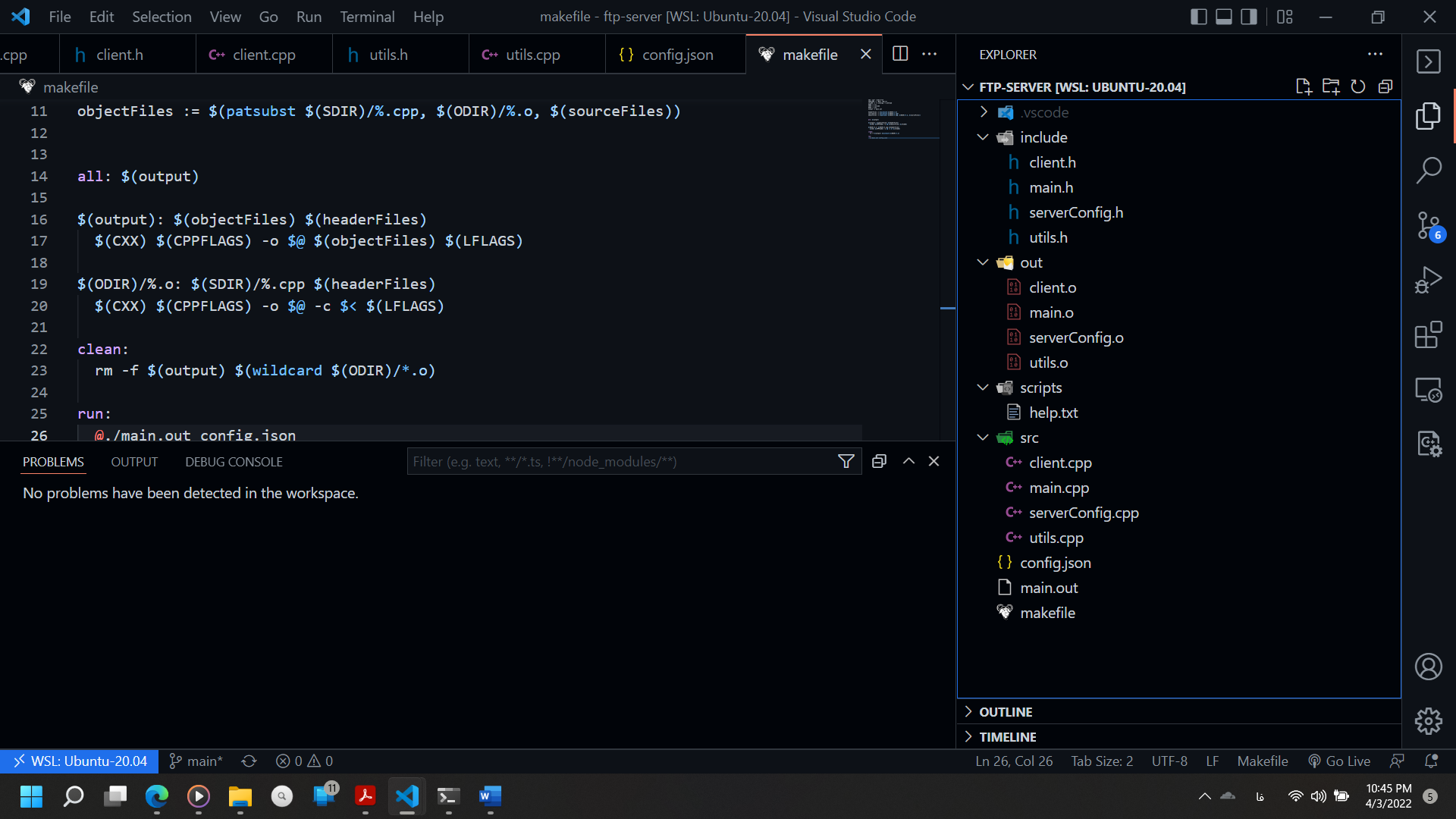
طراحی یک سرور FTP

علی لطف اللهی ۸۱۰۱۹۷۵۷۳

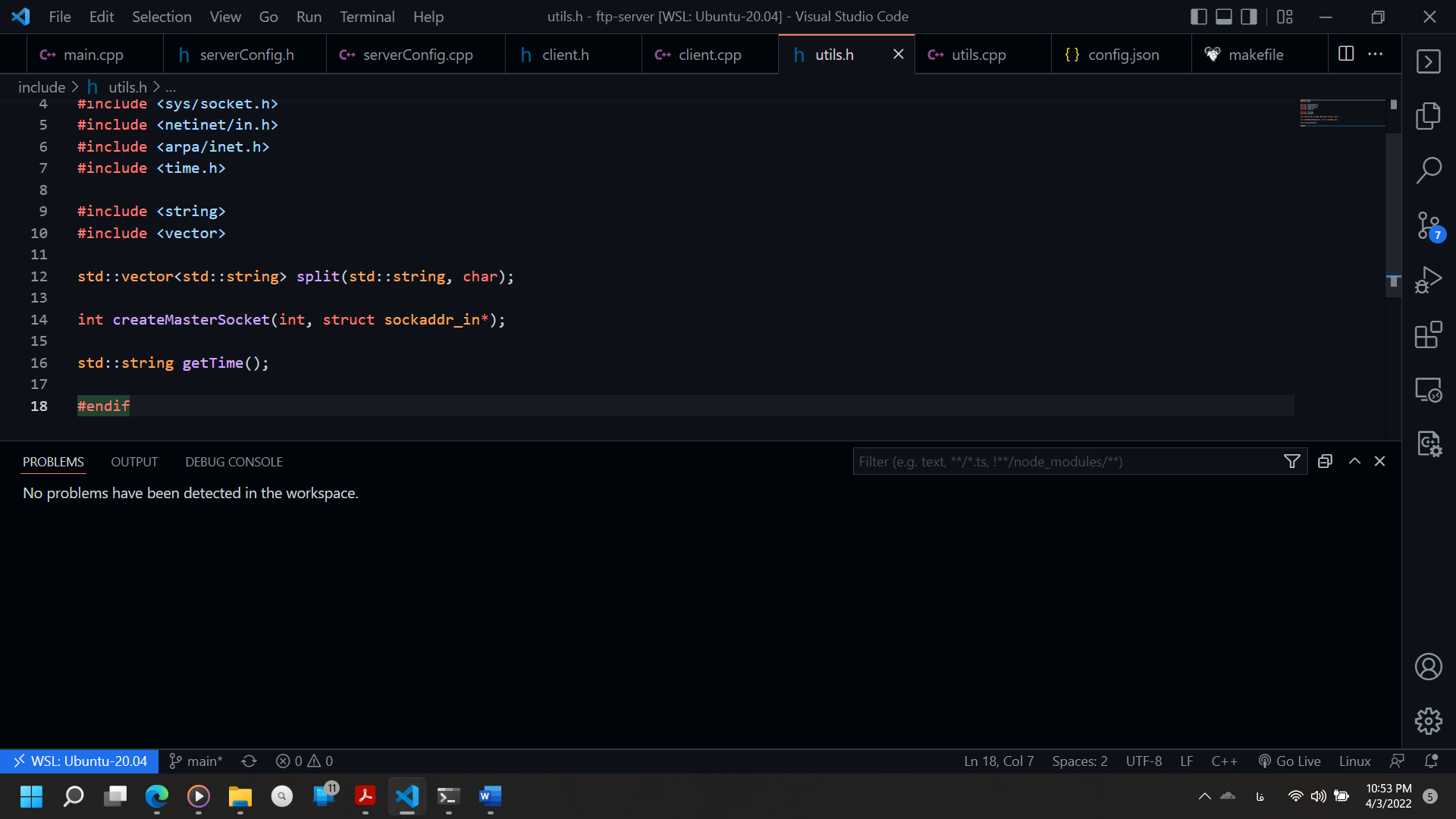
در این پروژه هدف ساخت یک سرور FTP است. این سرور با این هدف ایجاد شده است که به کاربرانش امکان استفاده از فایل هایی را بدهد که روی آن قرار دارند. این سرور از طریق دو socket با کاربرانش ارتباط برقرار می کند. یکی از این socket ها برای انتقال دستورات از کلاینت ها به سرور استفاده می شود و دیگری برای انتقال داده ها از سرور به کلاینت هاست.

سرور:

فایل های سرور به صورت زیر در پوشه ی پروژه ذخیره شده اند. همان طور که می توان دید این فایل ها در ۴ پوشه ی جداگانه دسته بندی شده اند. فایل های header در پوشه ی include، فایل های object در پوشه ی out و فایل های cpp در پوشه ی src جای گرفته اند. فایل help.txt نیز در پوشه ی scripts جای دارد که استفاده ی آن را خواهیم دید.

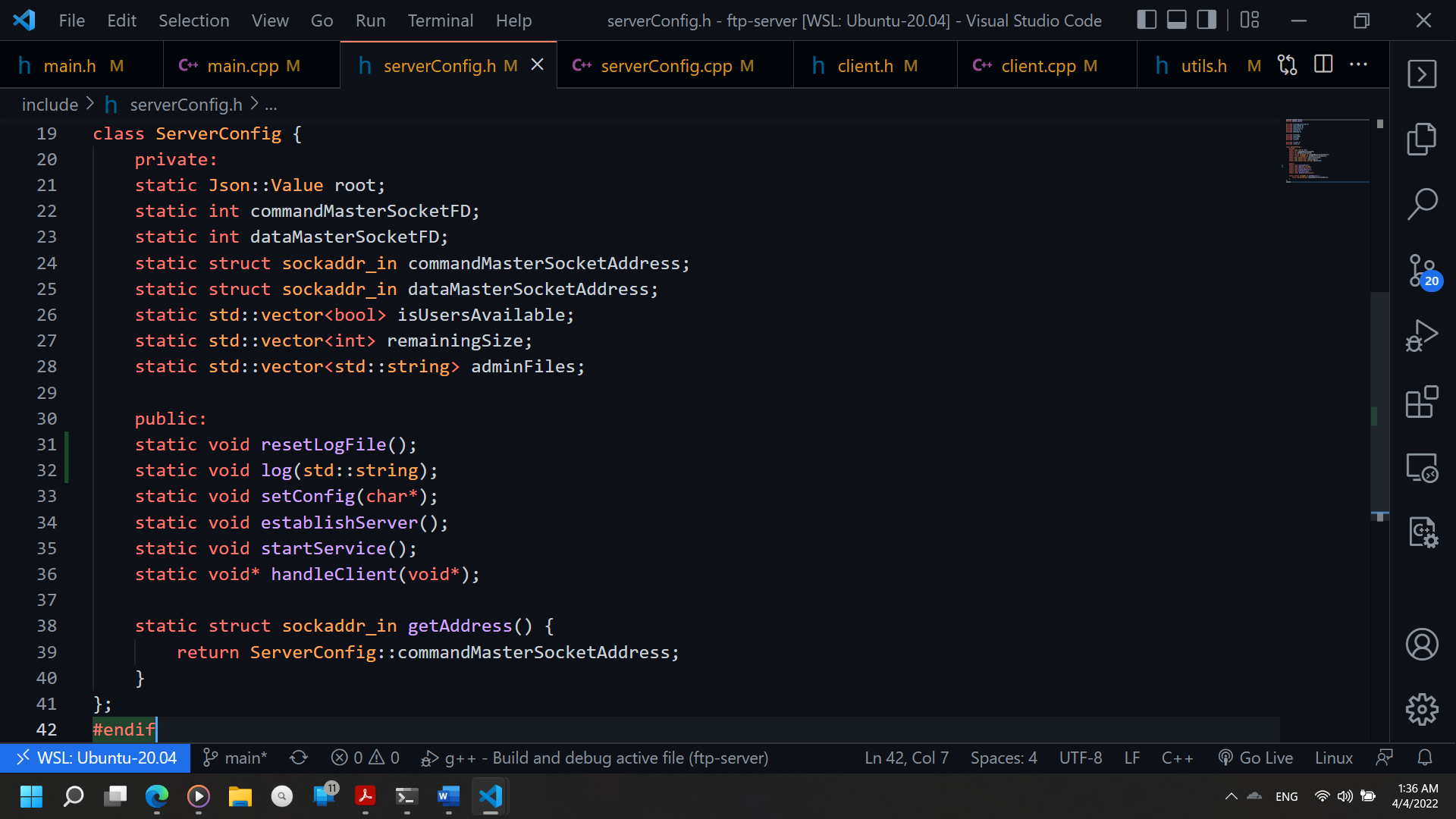


در فایل utils توابعی پیاده سازی شده اند که در طول برنامه در هر جایی از آن می توانند کاربردی باشند.



* تابع split یک داده ی string را بر اساس یک separator می شکند.
* تابع createMasterSocket یک socket در آدرس و پورت داده شده می سازد.
* تابع getTime زمان و تاریخ را به صورت string برمی گرداند.

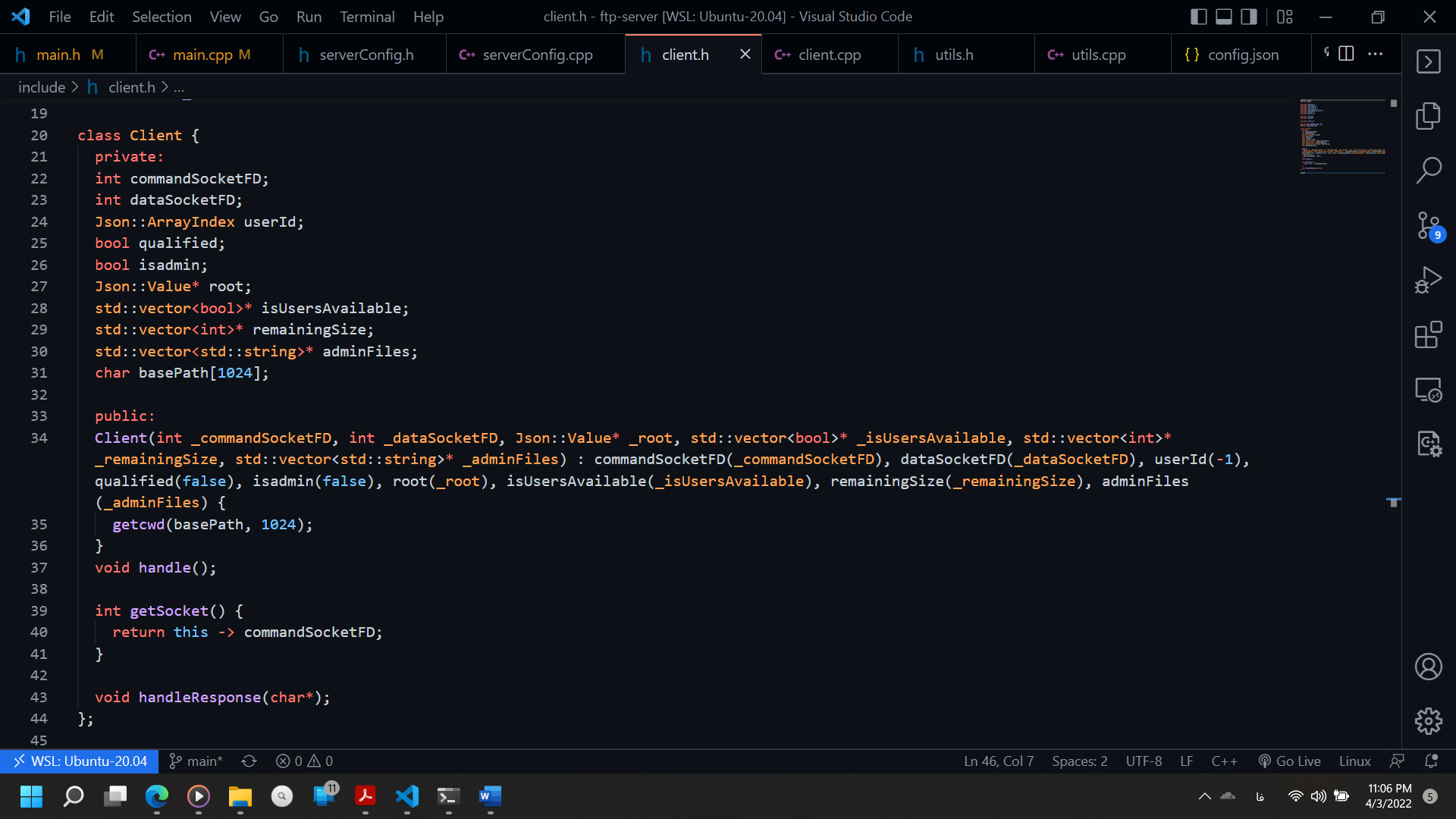
در فایل serverConfig تمامی توابعی که برای شروع به کار سرور نیاز هستند قرار گرفته اند.



این توابع به در قالب یک کلاس و به عنوان static method های آن پیاده سازی شده اند.

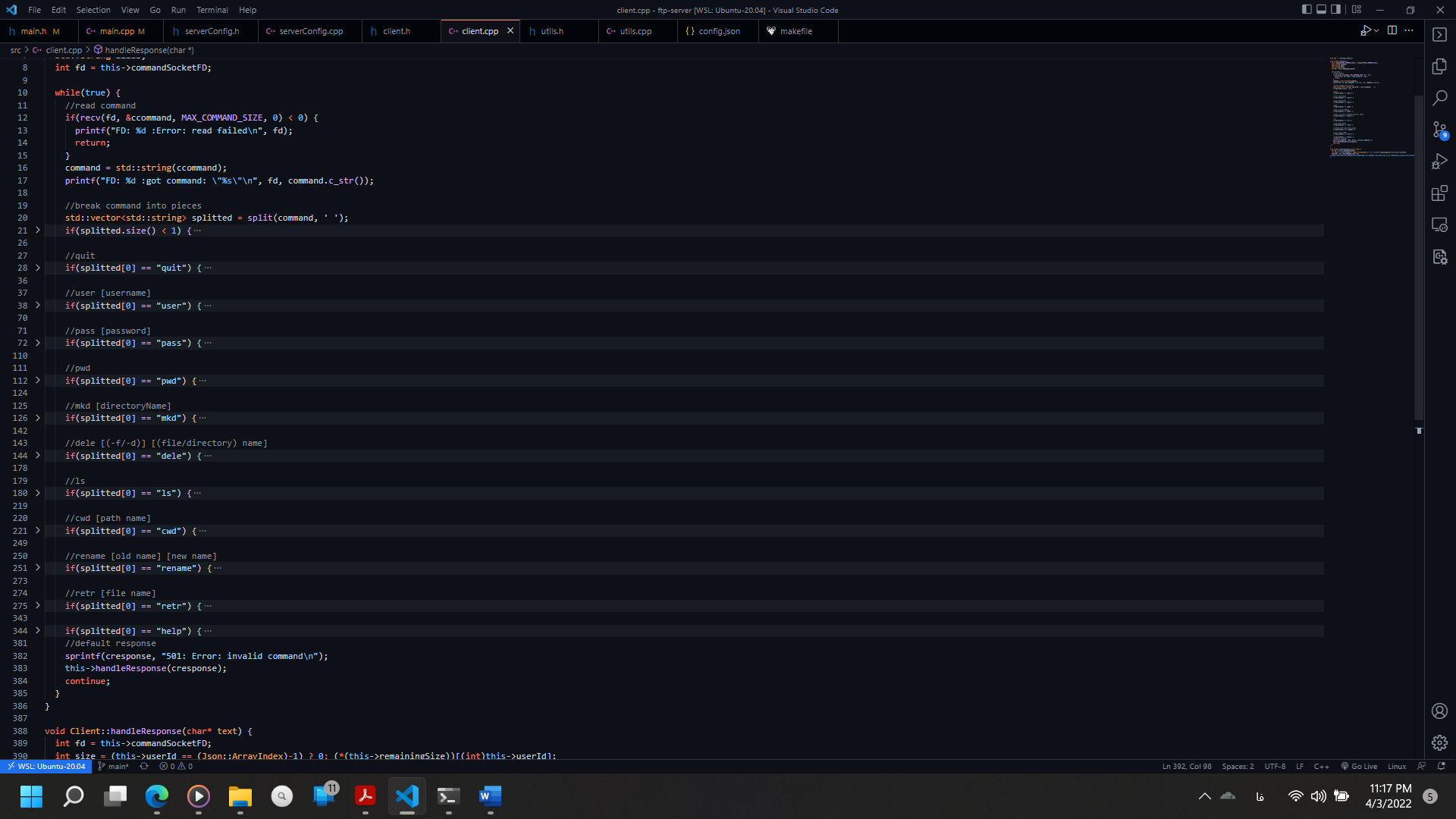
* تابع resetLogFile فایل log.txt را delete می کند.
* تابع log یک string را هم در کنسول و هم در log.txt چاپ می کند.
* تابع setConfig آدرس فایل config.json را مشخص می کند.
* تابع establishServer هر دو socket اولیه را می سازد.
* تابع startService قبول کردن کلاینت ها را شروع کرده و سپس برای هرکدام از آن ها یک thread می سازد.
* این thread تابع handleClient را اجرا می کند که این تابع، methodِ handleِ آن instance از client را اجرا می کند.

در فایل client تمامی اطلاعات مربوط به کلاس کلاینت که قرار است به طور جداگانه تک تک کلاینت ها را handle کند، قرار می گیرد.



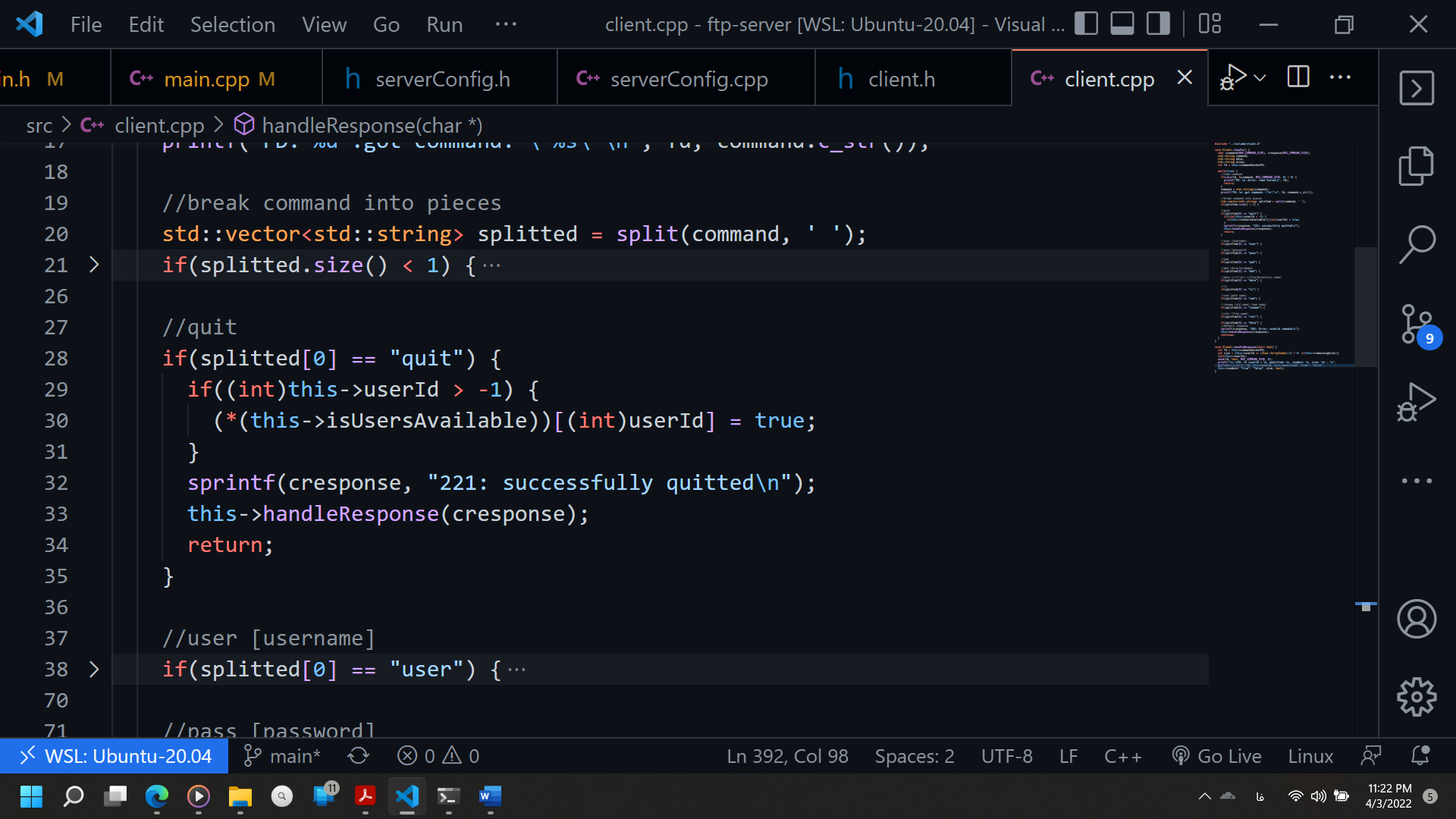
* این کلاس اطلاعات زیر را از serverConfig گرفته و به صورت by reference نگه می دارد:
* کدام user ها لاگین نیستند و available اند.
* برای هر user چقدر حجم باقی مانده است.
* User های معمولی به کدام فایل ها دسترسی ندارند.
* بخش بزرگی از کار سرور در methodِ handle انجام می شود که در ادامه به آن می پردازیم.

شکل کلی این فایل به صورت زیر است:

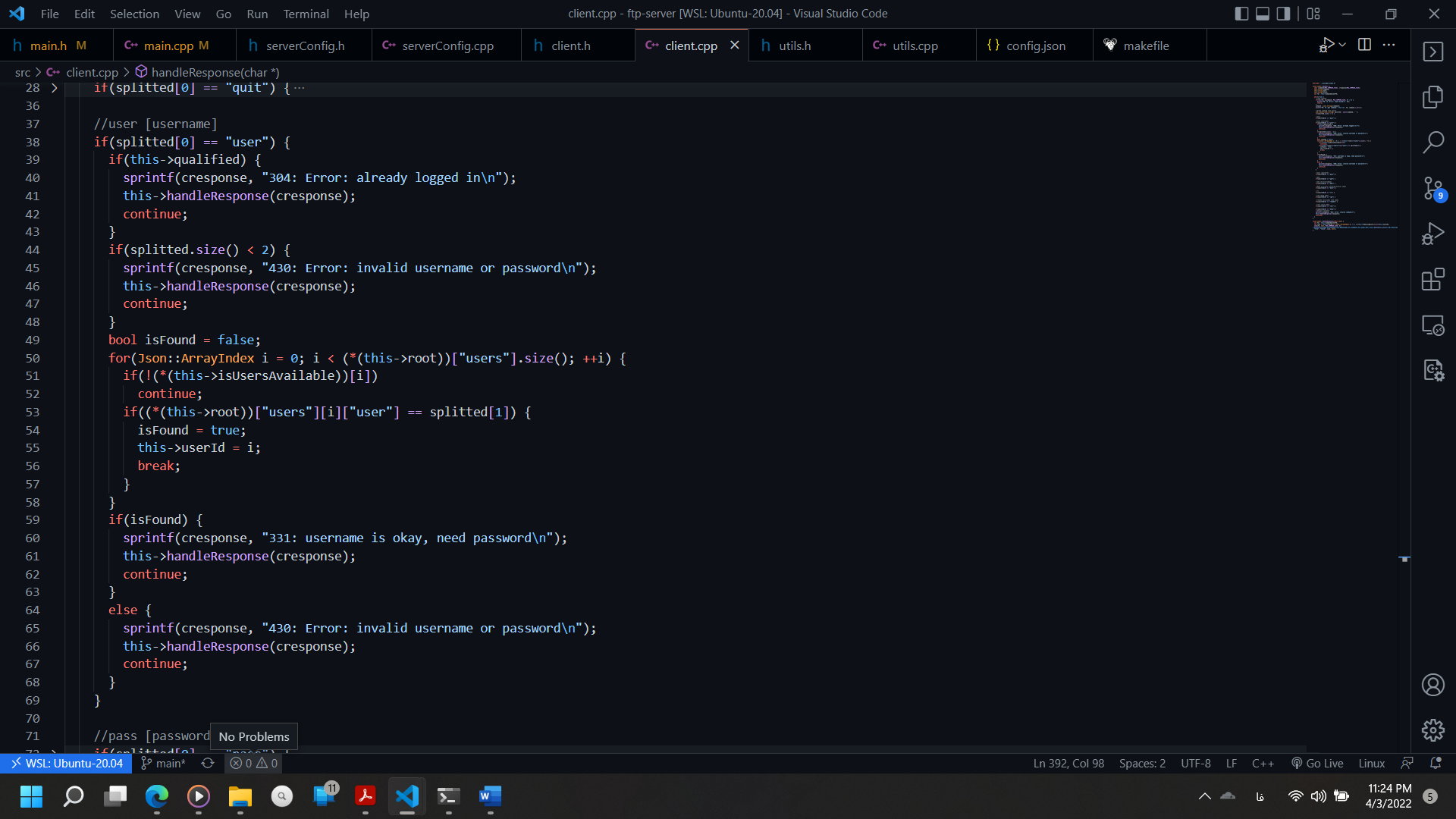


این method از یک while(true) تشکیل شده است که هر loop آن شامل handle کردن یک command می باشد. سپس command گرفته شده را می شکند و اجزای تشکیل دهنده ی آن را جدا می کند.

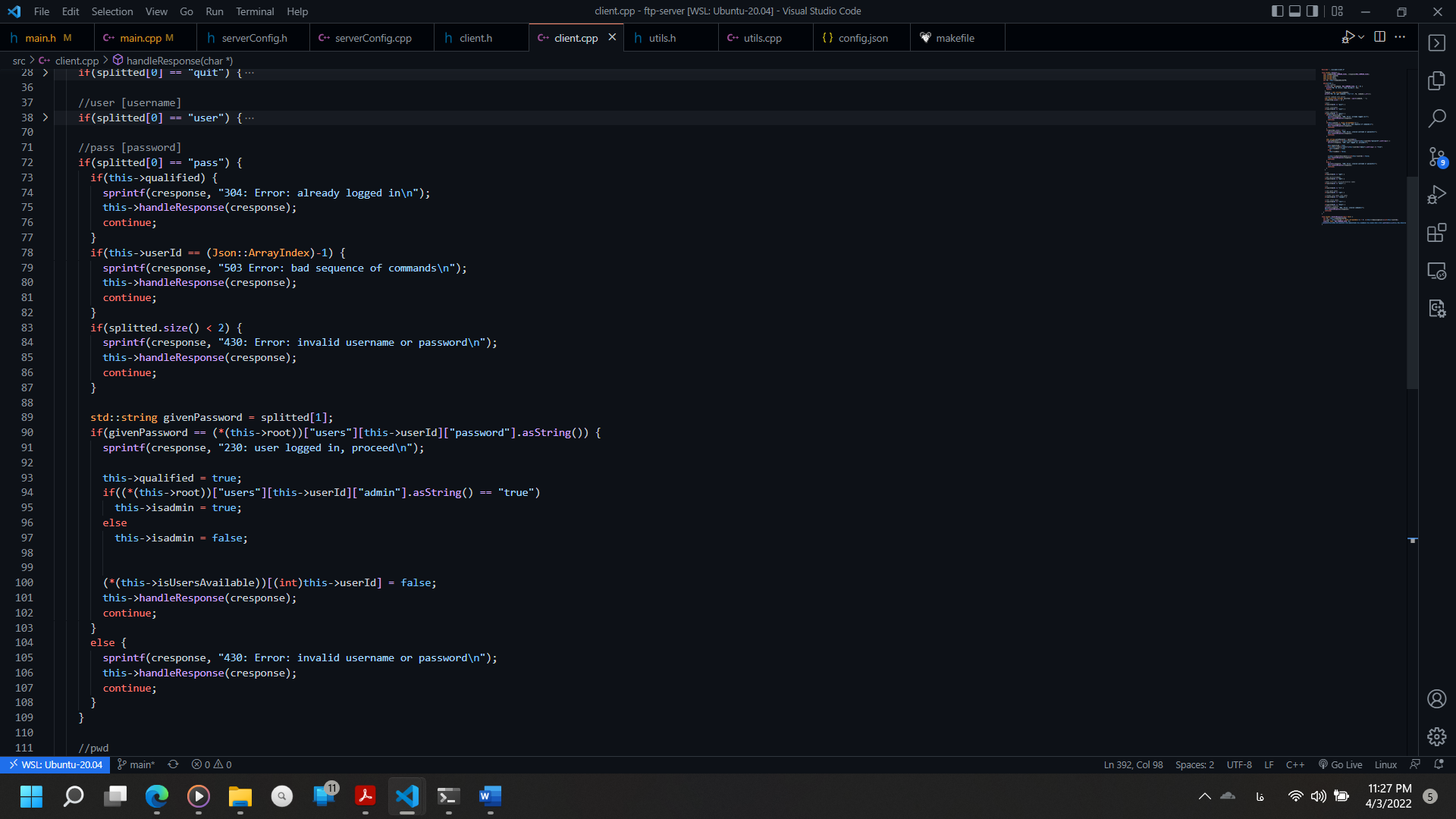
با توجه به بخش اول دستور را شناسایی کرده و کد مربوط به آن را اجرا می کند.



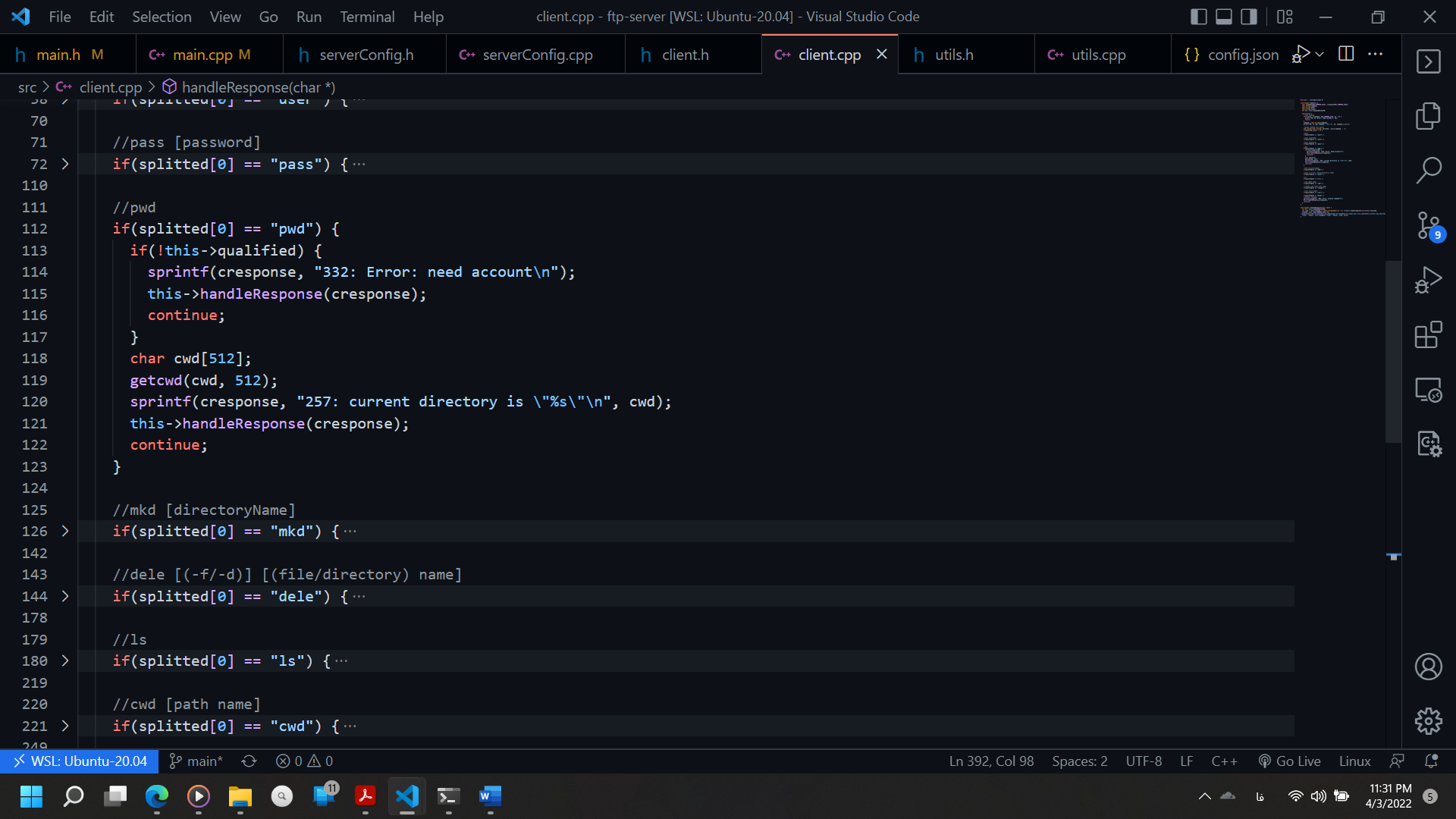
دستور بالا مربوط به دستور quit است که کاربر را ابتدا خارج کرده و سپس حلقه را پایان می دهد.



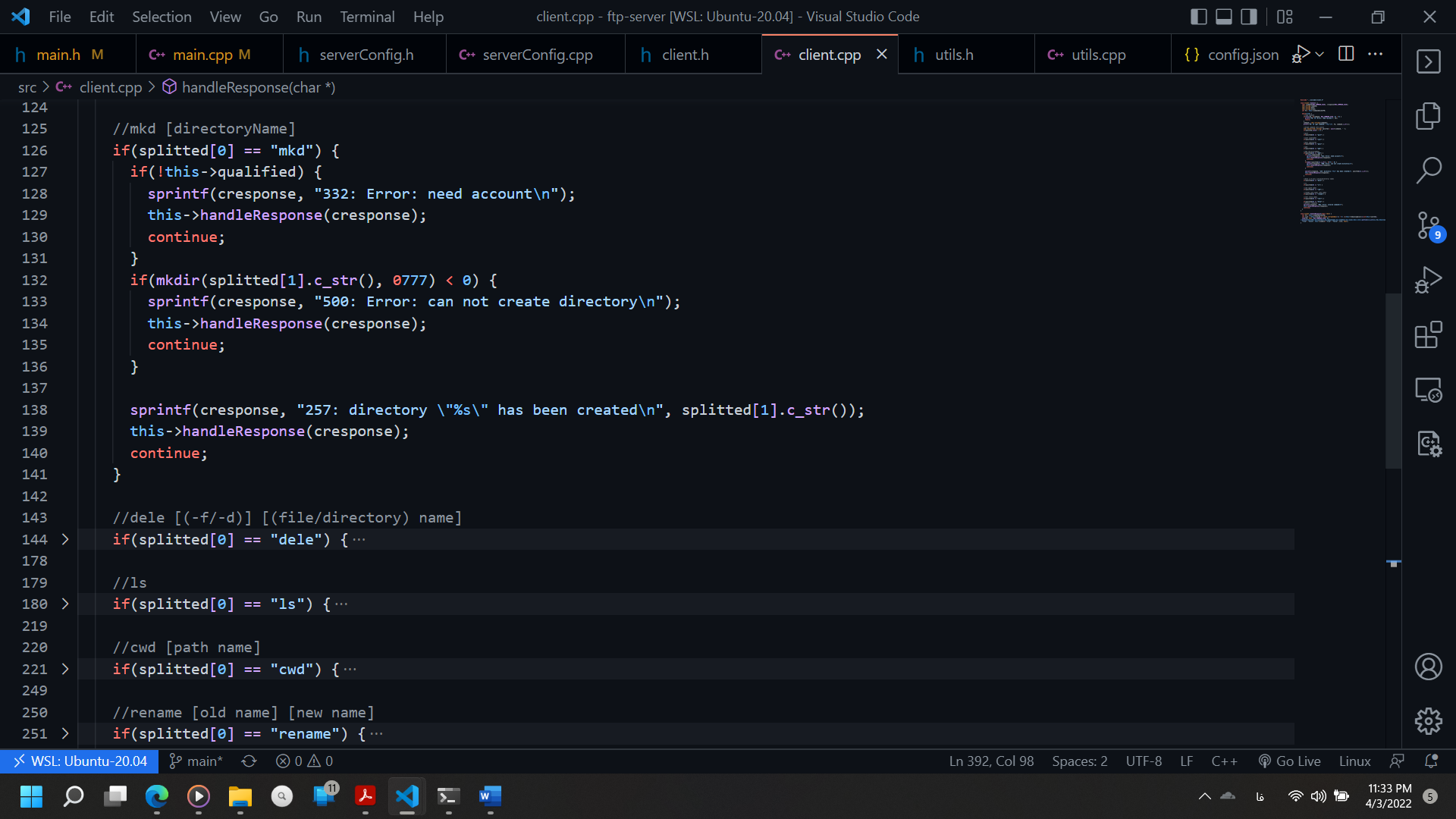
دستور user ابتدا چک می کند آیا کاربری وارد شده است یا خیر. سپس چک می کند آیا این username بین کاربران available وجود دارد یا خیر. در صورتی که تمامی شروط برقرار بود، شناسه ی آن user را ذخیره کرده و منتظر password می شود.



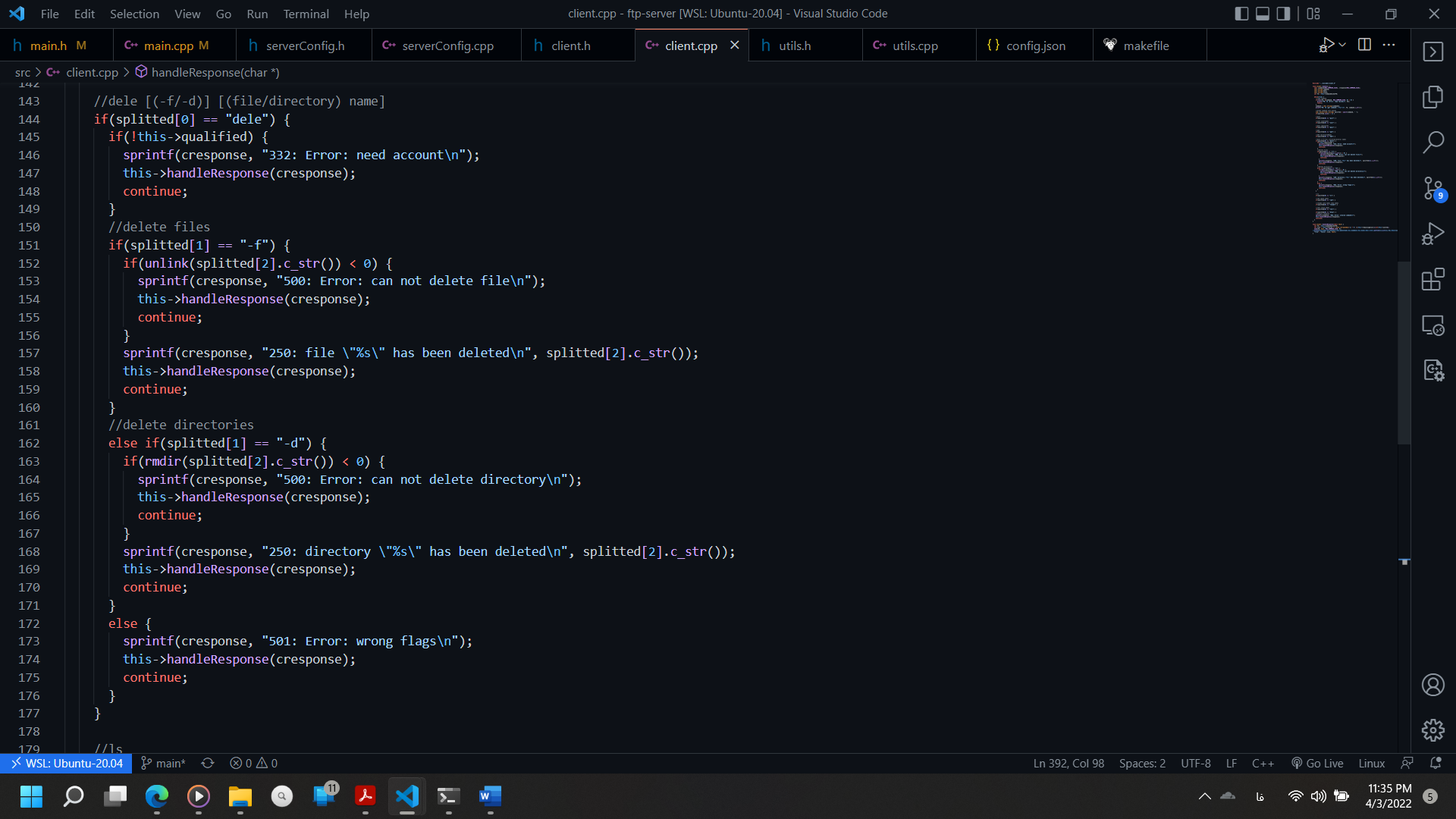
دستور pass ابتدا چک می کند که کاربر احراز هویت نشده باشد. سپس چک می کند که شناسه ی کاربری وجود داشته باشد (یعنی قبل از این دستور لزوما یک دستور user موفق بوده باشد). سپس password را چک کرده و اگر درست بود، کاربر وارد می شود.



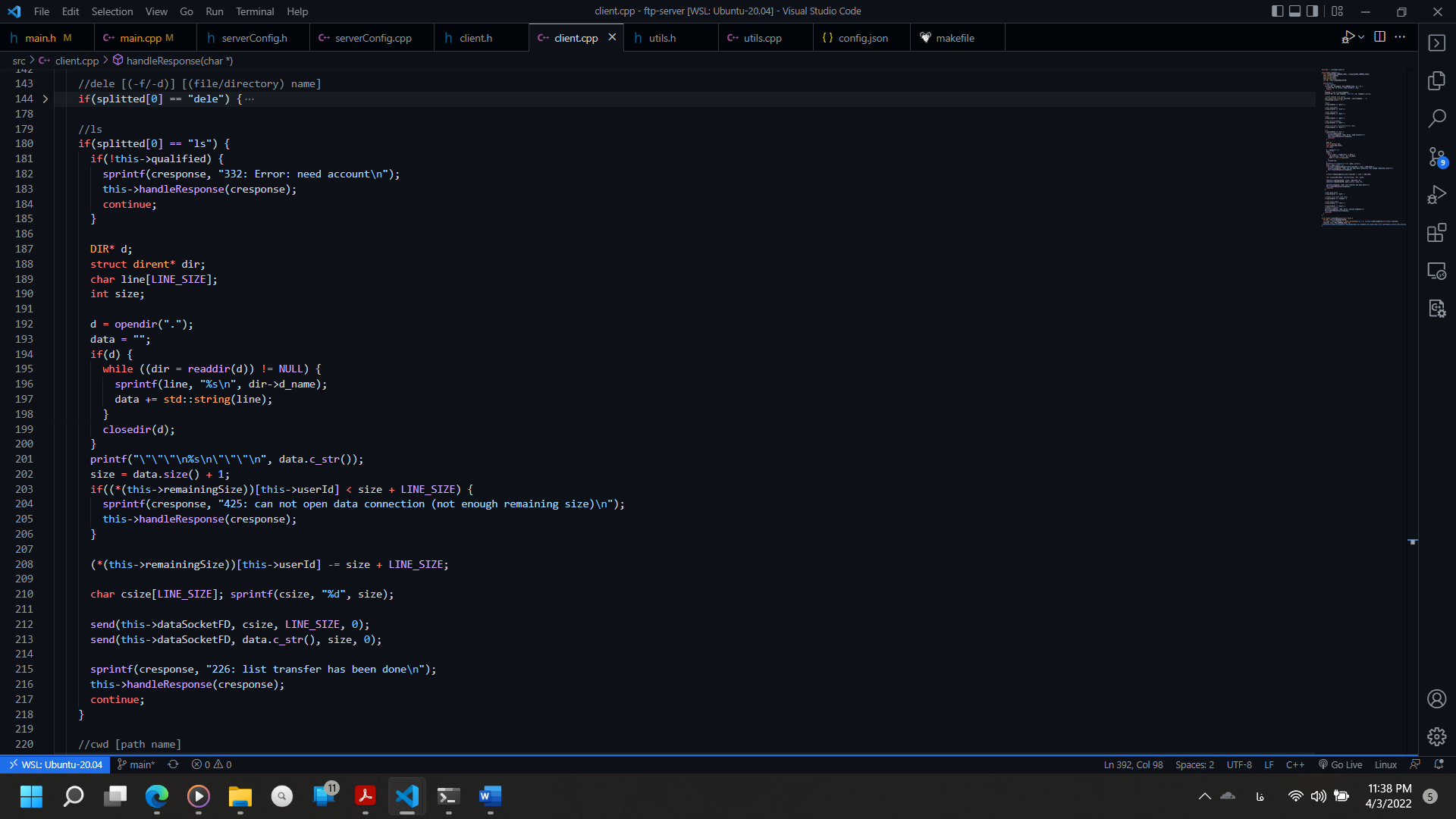
دستور pwd ابتدا چک می کند که کاربر وارد شده باشد و سپس اگر وارد شده بود، پوشه ی فعلی را بر می گرداند.



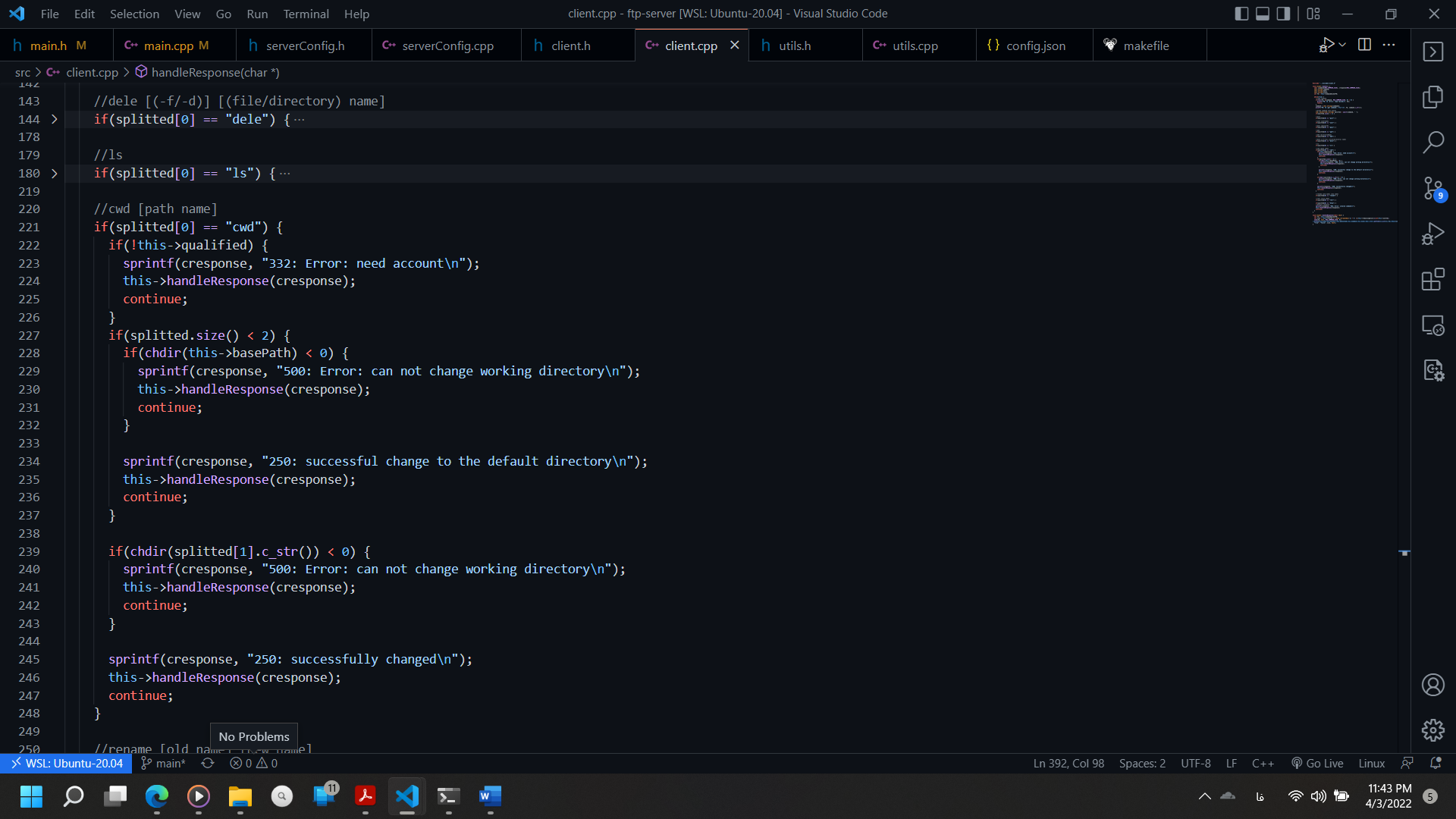
دستور mkd بعد از این که چک کرد کاربر وارد شده باشد،‌ یک directory با نام داده شده می سازد. اگر خود آن عملیات خطا داشت، خطا را باز می گرداند.



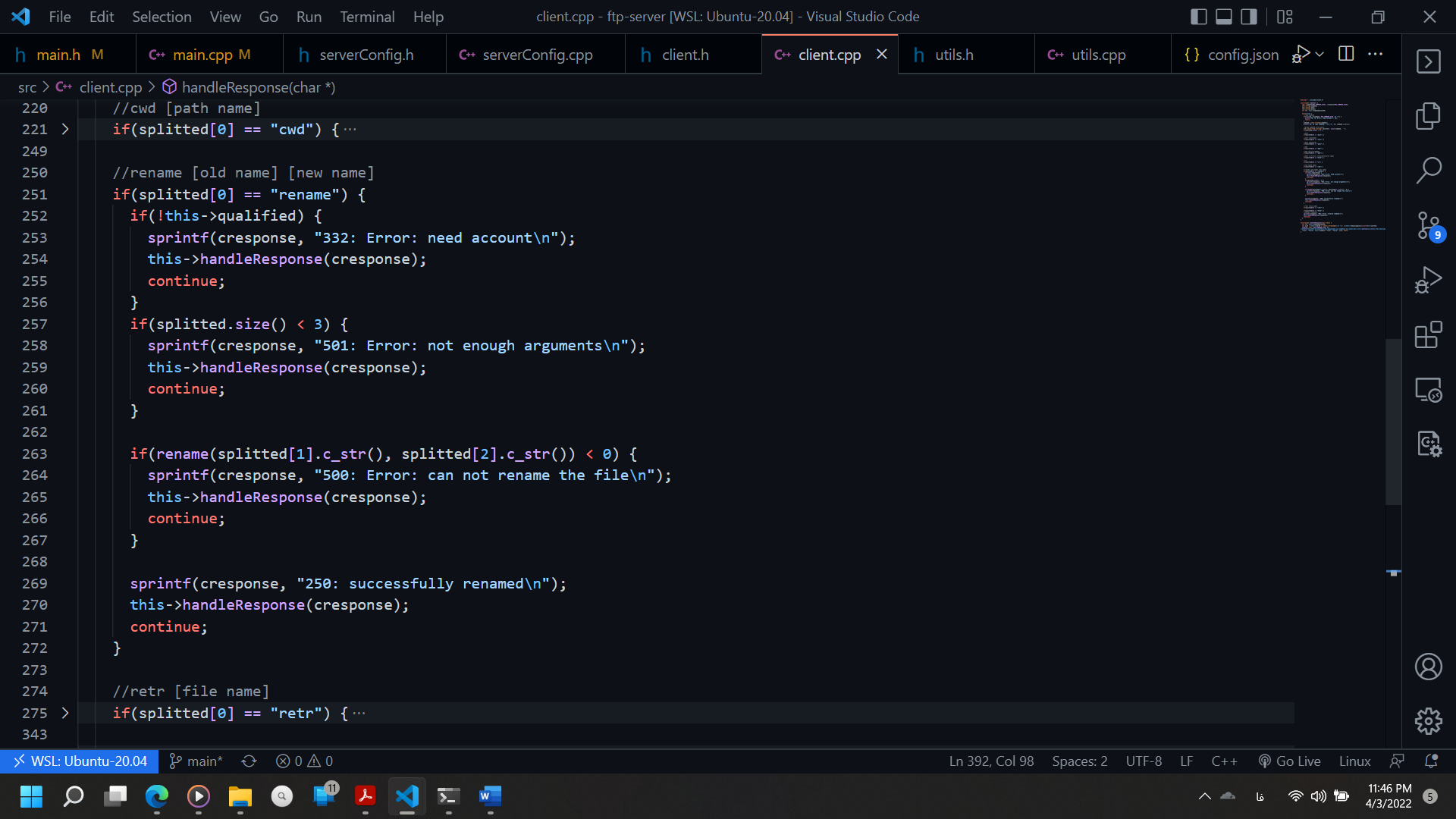
برای دستور dele ابتدا مانند دستورات پیشین، برنامه چک می کند که کاربر وارد شده باشد. سپس با توجه به flag داده شده مشخص می کند که چه چیزی باید پاک شود و آن را پاک می کند. اگر در هر مرحله خطایی رخ دهد آن را برمی گرداند.



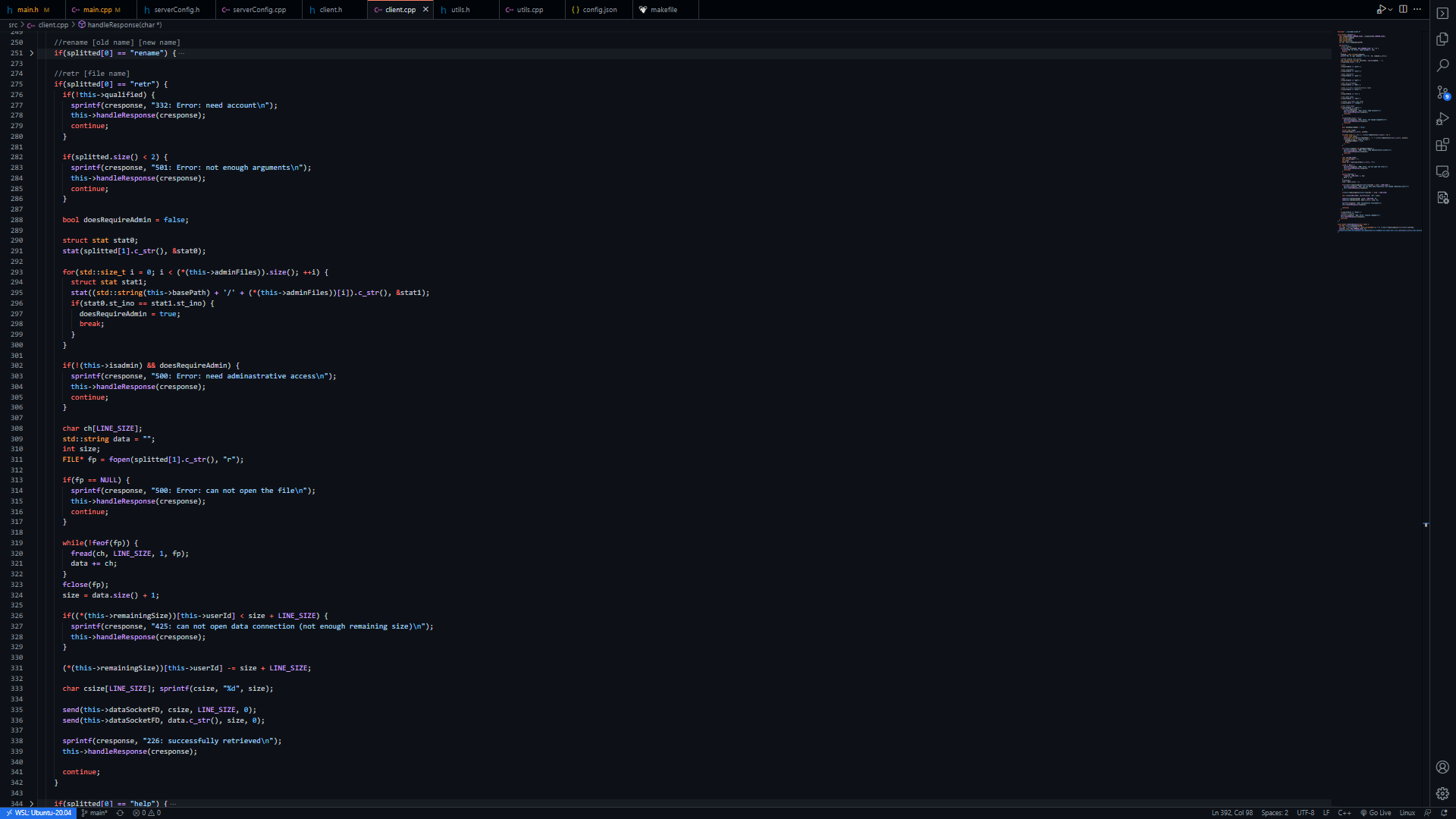
دستور ls ابتدا چک می کند ( مثل دستورات قبل!) که کاربر وارد شده باشد. سپس تمامی فایل های در directory فعلی را در یک string ذخیره کرده و سپس ابتدا سایز آن و بعد خود آن string را روی socket مربوط به داده قرار می دهد. از حجم مصرفی کاربر حجم این فایل کم خواهد شد.



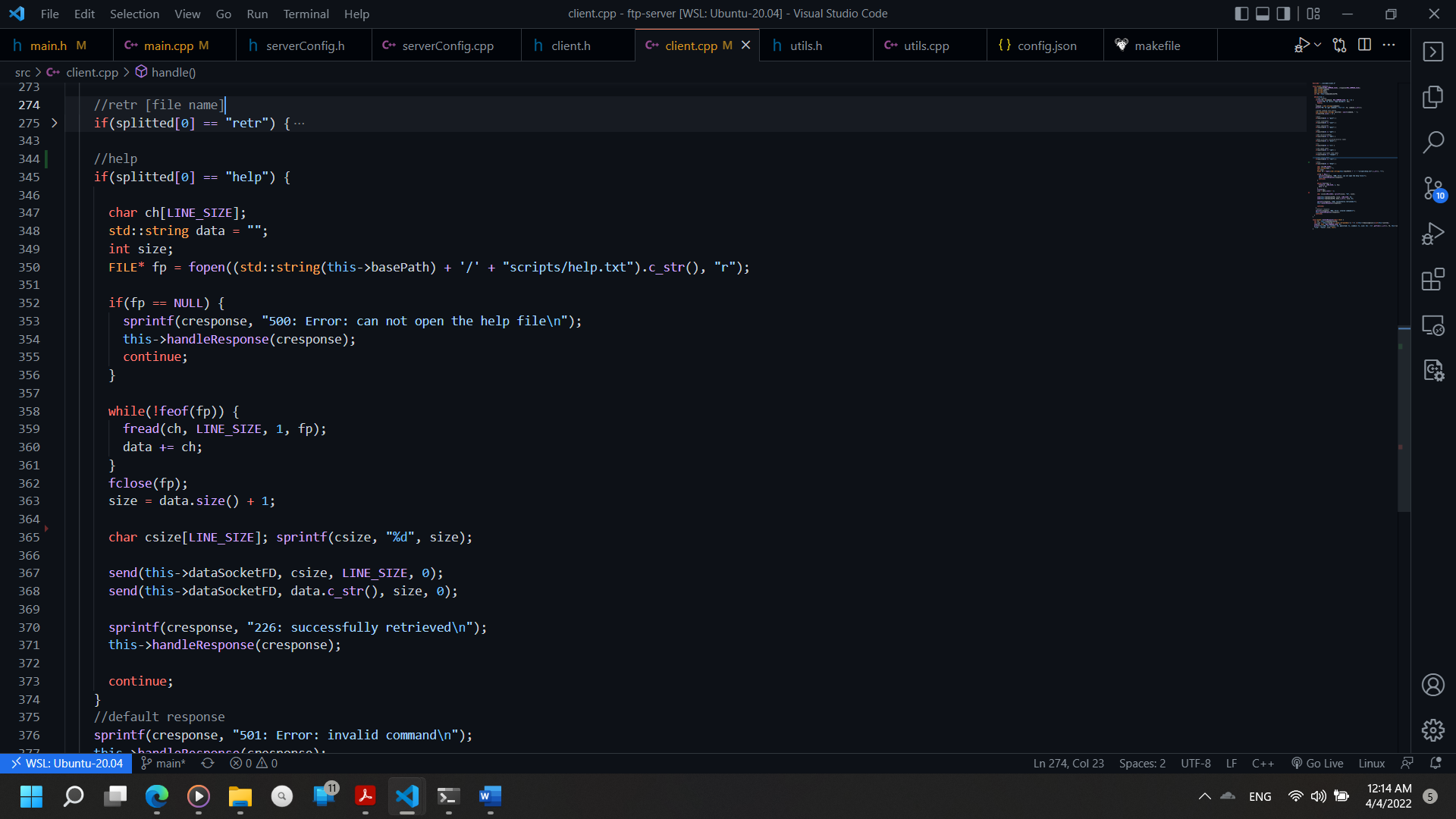
دستورcwd مانند بقیه ی دستور ها ابتدا چک می کند که آیا کلاینت احراز هویت شده است و یا خیر. سپس working directory را به directory خواسته شده تغییر می دهد. اگر آرگومانی نداشت به directory پایه می رود.



دستور بعدی دستور rename است. این دستور در صورتی که کاربر احراز هویت شده باشد نام فایل مشخص شده را به نام خواسته شده تغییر می دهد.



دستور retr که use case اصلی سیستم ما را ارضا می کند، ابتدا چک می کند که کاربر احراز هویت کرده باشد. سپس چک می کند که اگر کاربر یک کاربر معمولی است، فایل مورد نظر فایل غیر قابل دسترسی نباشد. در صورتی که همه چیز درست بود، ابتدا سایز فایل و سپس خود فایل را روی socket مربوط به داده قرار می دهد. پیش از ارسال فایل حجم کاربر مدیریت می شود.



در نهایت دستور help نیز محتویات فایل scripts/help.txt را روی socket داده قرار می دهد. تنها جایی که روی این socket داده قرار می گیرد ولی از حجم کسی کم نمی شود اینجاست.

برای پیاده سازی این برنامه از library های زیر استفاده شده است:

* Sys/socket.h
* Sys/stat.h
* Sys/type.h
* Unistd.h
* Fnctl.h
* Dirent.h
* Time.h
* String.h
* Netinet.h
* Pthread.h
* Arpa/inet.h
* Jsoncpp
* String
* Vector

Logging:

برای این برنامه log روی کنسول و فایل log.txt چاپ می شود. (این فایل در آدرس پایه ی سرور قرار دارد)

کلاینت:

پیاده سازی کلاینت به مراتب ساده تر است. در نتیجه تمامی کد را در یک فایل main جای داده ایم. این فایل شامل یک تابع main و دو تابع split و initializeConnection است. تابع اول به مانند سرور، command وارد شده توسط کلاینت را نسبت به space هایش جدا می کند. تابع initializeConnection، یک socket را ساخته و به سرور وصل می کند.

در تابع main پس از راه اندازی اتصالات، یک while(true) ساخته و در هر loop یک command را از کاربر گرفته و آن را به سرور می فرستیم.

* اگر دستور quit بود، از برنامه خارج می شویم.
* اگر دستور ls و یا help بود محتوا را از socket داده گرفته و چاپ می کنیم.
* اگر دستور retr بود محتوا را از socket داده گرفته و در فایلی با نام داده شده ذخیره می کنیم.