گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

روال کلی برنامه:

دو برنامه اجرایی سرور و کلاینت داریم:

- در سرور در یک حلقه (با select) منتظر کانکت شدن کلاینت‌ها به سوکت دستور و یا دریافت دستور جدید از طرف کلاینت‌های کانکت شده می‌مانیم. وقتی دستوری دریافت می‌کنیم آن‌را parse می کنیم و با توجه به نوع دستور آن را اجرا می‌کنیم و سپس پاسخ برای کلاینت ارسال می‌کنیم. اتصال به سوکت داده، هنگامی که یک کلاینت با دستور pass وارد می‌شود انجام می‌شود. هنگام ارسال فایل هم در سرور از fork استفاده می‌شود تا درصورت بزرگ بودن فایل، سرور بلاک نشود.
- کلاینت هم در یک حلقه اصلی از ورودی استاندارد دستور دریافت می‌کند و آن را برای سرور ارسال می‌کند. سپس منتظر پاسخ می‌ماند و پس از دریافت آن در صورتی که دستور عمل خاصی از جمله برای کانکت شدن به سوکت داده یا دریافت فایل و ... در آن بود آن را انجام می‌دهد و در غیر این صورت پاسخ را در خروجی استاندارد چاپ می‌کند.

در ادامه به توضیح کلاس های طراحی شده و method ها و field های آن ها می پردازیم.

سرور:

در main یک instance از کلاس سرور ساخته می‌شود و سپس متد run() آن صدا زده می‌شود.

- کلاس سرور:

کانستراکتور:

در کانستراکتور کلاس سرور ابتدا ۳ کلاس database و logger و command handler ساخته می‌شوند که هر کدام پایین تر توضیح داده خواهند شد. سپس متد initial_socket صدا زده می‌شود تا سوکت دستور را برای سرور بسازد و در فیلد آن ذخیره کند.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

متد initial_socket:

در این متد، فیلد استراکت sockaddr_in مقدار دهی می شود. و در ادامه سوکت دستور را می سازیم و در فیلد مربوطه ذخیره می کنیم. هر کجای این متد اگر مشکلی پیش در ساخت سوکت پیش بیاید، یک exception مربوط به آن فرستاده می شود و در کانستراکتور این کلاس دریافت و error مربوط به آن چاپ می شود.

متد run:

این متد لوپ اصلی سرور است که در آن از کلاینت‌ها دستور دریافت می‌کند و سپس پاسخ را برای آن‌ها ارسال می‌کند.

در این متد در یک حلقه بی‌نهایت، از select برای سرویس‌دهی هم‌زمان به چند کلاینت استفاده شده است. برای select یک master_set داریم که لیست فایل‌دیسکریپتورهایی است که باید روی آن‌ها منتظر دریافت اطلاعات باشیم (سوکت‌ها). و یک working_set داریم که در select استفاده می‌کنیم تا لیست فایل‌دیسکریپتور هایی که روی آن‌ها اطلاعات جدیدی دریافت شده را داشته باشیم.

در یک حلقه داخلی‌تر همه فایل‌دیسکریپتورهارا چک می‌کنیم که در working_set هستند یا نه، اگر باشند یعنی اطلاعات جدیدی برای خواندن از آن‌ها وجود دارد.

اگر این فایل‌دیسکریپتور همان سوکت سرور باشد یعنی یک کلاینت جدید می‌خواهد کانکت شود، پس آن را accept می‌کنیم و به master_set اضافه می‌کنیم. اگر سوکت سرور نباشد یعنی یکی از کلاینت‌ها که قبلا کانکت شده بود پیام جدیدی فرستاده است؛ پس آن‌را می‌خوانیم. و اگر طول این پیام صفر باشد یعنی EOF دریافت کرده‌ایم و آن کلاینت دیسکانکت شده پس سوکت او را می‌بندیم و فایل‌دیسکریپتورهای او را از جاهایی که نگه داشته بودیم پاک می‌کنیم و اگر کاربر آن لاگین بود او را لاگ‌اوت می‌کنیم.

اگر پیام دریافتی صفر نبود در نتیجه یک پیام جدید داریم و آن‌را با فایل‌دیسکریپتور متناظر با آن کلاینت به کلاس command_handler می‌فرستیم تا پاسخ آن را برگرداند و سپس پاسخ آن‌را به آن کلاینت می‌فرستیم.

- کلاس دیتابیس:

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

وظیفه کلی این کلاس نگه‌داری و بازیابی اطلاعات کاربران و فایل‌های محافظت شده‌است.

کانستراکتور:

با استفاده از متد read File، فایل config.json را با استفاده از یک کتابخانه می‌خواند و آن‌را parse می‌کند و سپس از روی اطلاعات آن آبجکت کاربرها را می‌سازد و آن‌هارا در وکتوری ذخیره می‌کند. همچنین لیستی از فایل‌های محافظت شده‌را نیز نگه می‌دارد.

نگه‌داری اطلاعات کاربران:

در کلاس دیتابیس دو ساختارداده map وجود دارد که یکی از آن‌ها از int به User* است. این int درواقع همان فایل‌دیسکریپتور سوکت کاربران هستند. در این مپ سوکت مربوط به دستور را نگه می‌داریم تا بدانیم هر کدام مربوط به کدام کاربر است. کلاینت وقتی که بعد از کانکت شدن دستور USER را بزند، این مپ آپدیت می‌شود و سوکت او را با یوزر match می‌کند.

مپ دیگر که int به int است، سوکت دستور کاربر را به سوکت داده او match می‌کند. این اتصال بعد از وارد کردن پسورد صحیح اتفاق می‌افتد.

و همینطور یک وکتور برای نگه‌داری همه کاربران وجود دارد.

متدها:

تعداد متد هم به عنوان getter وجود دارند تا بتوان کاربر را براساس یوزرنیم یا سوکت او دریافت کرد یا اینکه سوکت داده یک کاربر را از روی سوکت دستور او دریافت کرد تا بتوان برای او فایلی ارسال کرد.

تعدادی setter هم برای اضافه کردن زوج مرتبی به مپ‌های توضیح داده شده وجود دارد.

همچنین دو متد برای پاک کردن سوکت‌ها از مپ‌ها وجود دارد که بعد از خروج کاربران، از آن‌ها استفاده می‌شود.

یک متد هم is_restricted() برای چک کردن اینکه اسم فایلی در لیست فایل‌های محافظت شده است یا خیر وجود دارد.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

- کلاس کامند هندلر:

وظیفه کلی این کلاس parse کردن دستور ارسال شده توسط کاربران و انجام کار مربوط به آن و بازگردانی پاسخ مناسب آن و یا ارسال فایل به کاربران است.

کانستراکتور:

در این جا، instance های مربوط به کلاس های کامند هندلر و لاگر را از کلاس سرور دریافت کرده و در فیلد های مربوط ذخیره می کنیم. و اینکه سوکت داده ی سرور در اینجا ساخته می‌شود؛ چون در این کلاس ارسال فایل‌ها انجام می‌شود.

متد run_command_handler:

این متدی است که توسط کلاس سرور صدا زده می‌شود و پیام دریافت شده را به این کلاس می‌دهد. پیام را در قالب "رشته" دریافت می‌کند و آن را توکنایز می‌کند (توسط متد separate) و سپس متد handle_command را صدا می‌کند و دستور دریافت شده بررسی شود و انجام شود و نتیجه آن را بازمی‌گرداند.

همچنین دستورات این تابع در بلاک try هستند تا در صورت بروز اروری در انجام دستورها، پیام مناسب بازگردانده شود.

متد handle_command:

ابتدا با استفاده از فایل‌دیکسریپتور سوکتی که دستور از او آمده، از دیتابیس، کاربر مربوط به آن را دریافت می‌کنیم تا بدانیم دستور از طرف کدام کاربر است. اگر کاربر هنوز یوزرنیم وارد نکرده باشد nullptr دریافت می‌شود پس بعدا با مقایسه این متغیر با nullptr می‌توانیم بفهمیم کاربری وجود دارد یا خیر.

سپس در چندین ساختار if و else if بررسی می‌شود که دستور وارد شده (کلمه اول) کدام دستور است و براساس آن متد مناسب آن دستور صدا زده می‌شود. اگر هیچ‌کدام از دستورات نبود، ارور مناسب داده می‌شود.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

متدهای دستورها:

:USER

متد user_command که در آن در ابتدا ارور ها بررسی می‌شوند؛ اگر بعد از دستور، یوزرنیم وارد نشده باشد و یا کلاینت قبلا لاگین کرده باش ارور می‌دهیم.

در غیر این صورت در دیتابیس فایل‌دیسکریپتور کلاینت را با یوزر داده شده match می‌کنیم (متد set_user_fd).

:PASS

متد pass_command که در آن بررسی می‌شود که اگر بعد از دستور، رشته پسورد نبود و یا کاربر nullptr بود (یعنی دستور USER قبل از این دستور وارد نشده بود) و یا کلاینت قبلا لاگین کرده بود، ارور می‌دهیم.

اگر این ارورها وجود نداشت با پسورد داده شده روی آبجکت کاربر متد login را صدا می‌کنیم که اگر پسورد غلط باشد اکسپن می‌دهد. سپس در این مرحله که کاربر شناسایی شده سوکت داده را وصل می‌کنیم و متد create_data_connection را صدا می‌کنیم. در این متد به کلاینت رشته "connect" را می‌فرستیم تا بفهمد به پورت سرور برای سوکت داده متصل شود و در اینجا یک سوکت جدید را اکسپت می‌کنیم و آن را ریترن می‌کنیم. حال این فایل‌دیسکریپتور جدید را در دیتابیس ذخیره می‌کنیم تا بدانیم مربوط به کدام کلاینت است.

:PWD

بررسی می‌کنیم که کاربر وارد شده باشد و اروری نداشته باشیم. و سپس از آبجکت آن کاربر، محل فعلی او را می‌گیریم و ریترن می‌کنیم.

:MKD

متد mkd که در آن بررسی می‌کنیم کاربر وارد شده باشد و اسم دیرکتوری جدید را داده باشد و اروری نداشته باشیم. سپس از آبجکت کاربر، محل فعلی او را می‌گیریم و در آن با سیستم‌کال mkdir یک دیرکتوری جدید می‌سازیم.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

:DELE

متد dele که ابتدا بررسی های اولیه برای لاگین بودن و وارد کردن فلگ و اسم فایل انجام می‌شود تا اروری نداشته باشیم.

سپس بررسی می‌شود که فلگ -f است یا -d. اگر -f بود بررسی می‌شود اسم فایل داده شده، در لیست فایل‌های محافظت شده نباشد و یا اگر هست کاربر دسترسی ادمینی داشته باشد. سپس با سیستم کال stat و ماکرو S_ISREG بررسی می‌شود که فایل داده شده وجود داشته باشد و نوع آن فایل باشد و اگر بود با سیستم‌کال remove آن‌را پاک می‌کنیم.

اگر فلگ -d بود دوباره با سیستم کال stat و ماکرو S_ISDIR بررسی می‌کنیم دیرکتوری باشد. اگر بود با اجرای rm -rf آن را پاک می‌کنیم.

:LS

متد ls که در آن اول بررسی می‌شود کاربر وارد شده باشد و اروری نداشته باشیم. سپس سوکت داده کاربر را از دیتابیس می‌گیریم. به کلاینت رشته "ls" را می‌فرستیم تا متوجه شود قرار است پاسخ ls از طریق سوکت داده به او ارسال شود. حال با کمک سیستم‌کال های opendir و readdir در یک حلقه لیست همه فایل‌های موجود در آدرس فعلی کاربر را در یک وکتور می‌ریزیم و سپس اعضای وکتور را مرتب می‌کنیم و در یک رشته می‌ریزیم. حالا با استفاده از fork یک پراسس جدید می‌سازیم تا این رشته را با سوکت داده به کلاینت ارسال کند. دلیل فورک کردن هم این است که اگر ارسال طولانی شد سرور بلاک نشود. (این کار بیشتر برای دستور retr مهم است.)

:CWD

متد cwd که ابتدا بررسی می‌کند کاربر لاگین باشد و اروری نداشته باشیم. سپس اگر دستور تنها و بدون آرگومان بود، environment variable به نام PWD از طریق سیستم‌کال getenv دریافت می‌شود که در واقع محل اولیه سرور را می‌دهد و به عنوان محل فعلی کاربر ست می‌شود.

اگر آرگومان داشت به کمک stat و S_ISDIR بررسی می‌کنیم دیرکتوری باشد و سپس آن‌را به عنوان محل فعلی کاربر ست می‌کنیم.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

:RENAME

متد rename_command که بررسی می‌کند کاربر لاگین باشد، آرگومان ها به تعداد باشد و فایل داده شده محافظت شده نباشد و اگر هست کاربر دسترسی ادمین داشته باشد و اروری نداشته باشیم.

سپس با سیستم‌کال rename نام فایل را تغییر می‌دهیم.

:RETR

متد retr که ابتدا بررسی می‌کند کاربر لاگین باشد، آرگومان ها به تعداد باشد و فایل داده شده محافظت شده نباشد و اگر هست کاربر دسترسی ادمین داشته باشد و اروری نداشته باشیم.

سپس با سیستم‌کال stat و S_ISREG بررسی می‌کند نام داده شده فایل باشد.

با متد can_download از آبجکت کاربر بررسی می‌شود که کاربر حجم کافی برای دانلود فایل دارد یا نه. سایز فایل را هم از استراکت stat که در بررسی قبل گرفته شده بود، می‌گیریم.

اگر همه بررسی ها مثبت بودند سایز فایل را با متد subtract size از حجم باقی‌مانده کاربر کم می‌کنیم. فایل را با open باز می‌کنیم و فایل‌دیسکریپتور آن‌را نگه می‌داریم.

حال می‌خواهیم فایل را بفرستیم. سوکت داده کاربر را از دیتابیس می‌گیریم و به کلاینت یک پیام قراردادی به شکل زیر می‌فرستیم:

dl NAME#SIZE$

از دو حرف اول کلاینت می‌فهمد که قرار است پاسخ فایل دستور retr بیاید و با NAME می‌فهمد که فایل گرفته شده با چه نامی باید ذخیره شود و از SIZE می‌فهمد که چقدر باید از سوکت داده بخواند.

حالا برای ارسال فایل فورک می‌کنیم تا درصورت بزرگ بودن فایل سرور بلاک نشود. با سیستم‌کال sendfile فایل‌دیسکریپتور فایل و سوکت داده کلاینت و حجم را می‌دهیم تا فایل ارسال شود. سمت کلاینت هم اندازه سایز فرستاده شده از سوکت داده می‌خواند و ذخیره می‌کند.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

:HELP

متد help که رشته آماده را دارد و به کلاینت پیام قراردادی $hp SIZE را می‌فرستد تا سایز پیام را بداند تا در صورتی که متن هلپ بزرگ‌تر از بافر بود، بتواند همه آن‌را بخواند. سپس رشته برای کلاینت فرستاده می‌شود.

:QUIT

متد quit که بررسی می‌کند اگر کاربر لاگین نبود ارور دهد.

متد logout کاربر را صدا می‌کند و سپس فایل‌دیسکریپتور های سوکت‌های مربوط به کاربر را از دیتابیس پاک می‌کند.

- کلاس لاگر:

فایل log.txt را در کانستراکتور در حالت append باز می‌کند.

با متد save_log پیام آرگومان را به همراه زمان فعلی (به کمک متد find time) در فایل می‌نویسد.

در بین کدها هم بعد از اجرای دستورات یا اتفاقات پیام مناسب با این متد در لاگ نوشته می‌شود.

- کلاس کاربر:

اطلاعاتی از جمله یوزرنیم، پسورد، دسترسی ادمینی داشتن، وضعیت لاگین بودن، حجم باقی‌مانده و مکان فعلی کاربر را نگه‌می‌دارد.

کانستراکتور:

اطلاعات داده شده را ذخیره می‌کند. مکان اولیه را از روی متغیر محیطی PWD ست می‌کنیم و سایز داده شده در ۱۰۲۴ ضرب می‌کنیم که تبدیل به بایت شود.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

متدها:

تعدادی getter و setter برای وضعیت ادمینی، لاگین بودن و یا محل فعلی وجود دارد.

متد login که پسورد داده شده را بررسی می‌کند و درصورت درست بودن، وضعیت لاگین بودن را تغییر می‌دهد و اگر غلط باشد اکسپشن می‌دهد.

متد logout هم برای خروج استفاده می‌شود.

متد can_download بررسی می‌کند با توجه به حجم باقی مانده، امکان دانلود فایلی با سایز داده شده است یا خیر.

متد subtract_size هم سایز فایل داده شده را از حجم باقی‌مانده کم‌میکند.

کلاینت:

در main یک اینستنس از کلاس کلاینت ساخته می‌شود و متد run() آن صدا می‌شود.

- کلاس کلاینت:

سوکت‌های داده و دستور را نگه می‌دارد و با متدهایی با سرور ارتباط برقرار می‌کند.

کانستراکتور:

متد connect_to_server را صدا می‌کند که در آن سوکت دستور را به سرور کانکت می‌کند و یک سوکت داده می‌سازد (سوکت داده را فعلا متصل نمی‌کند).

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

متد run:

حلقه اصلی برنامه این متد است که در آن از ورودی استاندارد، یک دستور می‌گیرد و آن‌را برای سرور می‌فرستد و پاسخ آن‌را به متد handle_response می‌دهد.

متد handle_response:

رشته پاسخ را بررسی می‌کند که ببیند رشته‌های قراردادی را دارد یا نه؛ در ادامه این رشته های قراردادی را بررسی می کنیم:

"connect":

یعنی باید به سوکت داده ی سرور متصل شود. و بعد از آن پاسخ بعدی سرور را از سوکت دستور دریافت می‌کند.

"ls":

یعنی قرار است پاسخ ls از سوکت داده بیاید. پس، از سوکت داده پاسخ سرور را می‌خوانیم و آن‌را در خروجی چاپ می‌کنیم. سپس باقی پاسخ سرور را یا چاپ می‌کنیم یا اگر باقی‌ای نداشت، دوباره از سوکت دستور، پاسخ می‌گیریم.

"dl":

یعنی قرار است از سرور، فایل در پاسخ retr بیاید. متد receive file from server را صدا می‌کنیم. سپس باقی پاسخ سرور را یا چاپ می‌کنیم یا اگر باقی‌ای نداشت دوباره از سوکت دستور، پاسخ می‌گیریم.

"221":

این کد در پاسخ quit می‌آید، پس یعنی که کاربر خارج شده در نتیجه سوکت داده را می‌بندیم و یک سوکت جدید می‌سازیم.

گزارش پروژه اول درس شبکه های کامپیوتری

حمیدرضا خدادادی 810197499

محمدعلی زارع 810197626

"hp":

یعنی پاسخ help قرار است دریافت شود. پس به اندازه سایز دریافت شده از سرور از سوکت دستور می‌خوانیم تا کل پاسخ را دریافت کنیم. دلیل این کار امکان بزرگ‌تر بودن متن از سایز بافر است که مجبور می‌شویم چندین بار بخوانیم. با دانستن سایز، به اندازه می‌خوانیم و روی سیستم‌کال recv بلاک نمی‌شویم.

متد receive_file_from_server:

ابتدا یک دیرکتوری جدید به اسم downloads، درصورتی که وجود نداشت می‌سازیم تا فایل را در آن ذخیره کنیم. سپس نام و سایز فایل‌را از رشته دریافت شده استخراج می‌کنیم. یک فایل با "نام دریافتی" open می‌کنیم و می‌سازیم. حال در یک حلقه تا زمانی که اندازه سایز فایل از سرور دریافت کنیم، از سوکت داده می‌خوانیم و آن را در فایلی که open کردیم می‌نویسیم (دلیل این‌کار احتمال بزرگ‌تر بودن سایز فایل از بافر و اطمینان نداشتن دریافت کل فایل در یک بار صدا کردن recv است). هنگامی که به اندازه خواندیم و نوشتیم فایل را close می‌کنیم.

- کلاس های ارور هندلر:

این مجموعه کلاس ها که از کلاس std::exception ارث بری کرده اند، به گونه ای پیاده سازی شده اند که به ازای هر یک از ارور های موجود، یک کلاس ساخته ایم. در کانستراکتور هر یک از این کلاس ها، پیغام مربوطه در فیلد کلاس ذخیره می شود. و با متد what، به هنگام صدا زده شدن کلاس، پیغام ذخیره شده در فیلد مربوطه ریترن می شود.