گزارش پروژه اول درس شبکه‌های کامپیوتری

آتیه آرمین(۸۱۰۱۹۷۶۴۸) – معین شیردل(۸۱۰۱۹۷۵۳۵)

کلاس server

در ابتدا و در تابع main برنامه، یک instance از سرور ساخته می شود. این سرور وارد تابع خواننده ی فایل کانفیگ می شود. در این تابع از فایل json اطلاعات خوانده می شوند. یک object برای هر یوزر ساخته می شوند و وارد یک بردار (vector) از یوزر ها می شوند. پورت های کانال ارسال داده و ارسال دستورات نیز به همراه نام فایل هایی که محدودیت دسترسی دارند ذخیره می شوند. در نهایت وارد تابع start این سرور می شویم. در اینجا، دو سوکت روی پورت های مشخص شده در فایل کانفیگ ساخته می شود و منتظر اتصال کلاینت ها می مانند. به هر کلاینتی که اتصالش به هر دو سوکت accept شد، یک thread اختصاص می دهیم تا دستورات آن کلاینت را اجرا کنند. دو سوکت داریم که به ازای هر یک، یک file descriptor وجود دارد. آن ها را نیز به کمک یک struct به thread مخصوص آن کلاینت پاس می دهیم (به کمک cast کردن به void*). سپس در سرور منتظر ارسال یک دستور از سمت کلاینت می شویم و هر گاه دستوری وارد شد، به کمک instance ای از کلاس CommandHandler که در کلاس سرور وجود دارد، به آن دستور رسیدگی می کنیم. در نهایت نیز پاسخ متنی هر دستور را از طریق کانال دستور به کلاینت ارسال می کنیم و اگر کلاینت به واسطه آن دستور فایلی درخواست کرده بود، آن فایل را نیز از طریق کانال دیتا برای کلاینت ازسال می کنیم.
در تمام این مراحل، لاگ های مربوط به سرور از طریق تابع record_log موجود در این کلاس، ثبت می شوند و در انتهای کار، فایل سوکت ها بسته می شوند و thread های مشغول join می شوند و برنامه خاتمه می یابد.

کلاس client

در تابع main مربوط به کلاینت نیز یک instance از آن ساخته می شود. سپس پورت های اتصال را از فایل کانفیگ می خوانیم و وارد تابع start مخصوص کلاینت می شویم. سوکت های مربوط به آن را می سازیم و درخواست اتصال به سرور را می دهیم. سرور باید توسط یک ترمینال دیگر start شده باشد تا کلاینت بتواند به آن متصل شود. پس از برقراری ارتباط، دستوراتی که در ترمینال مربوط به کلاینت وارد می شوند از طریق کانال دیتا (کانالی که به کمک سوکت دیتا ساخته شده است) به سرور ارسال می شود و پاسخ متنی آن از همین کانال دریافت می شود و در ترمینال چاپ می شود. در صورتی که کلاینت درخواست دانلود فایلی را داشت هم این فایل را می تواند از کانال دیتا دریافت کند.
نحوه دریافت همزمان دیتا و پاسخ کامند ها نیز به این صورت است که از ۲ thread استفاده شده است. پس از کانکت شدن هر دو سوکت، به یک thread وظیفه دریافت محتوای فایل ها را می سپاریم که رشته ورودی را در یک بافر ذخیره می کند و در انتها این بافر را درون یک فایل جدید در کنار client.out ذخیره می کند. thread دیگر دستور ها را می فرستد و پاسخشان را دریافت می کند و در ترمینال چاپ می کند.

کلاس command handler

در این کلاس سعی بر این است که command هایی که کاربر وارد میکند، مدیریت شوند. این کلاس شامل ۲متد command parser و handle و یک رشته command که دستور را نگه میدارد، یک بردار args که argument هایی دستور را نگه میدارد، آدرس server، آدرس client، یک Boolean برای اینکه بدانیم user ای وارد سامانه شده است یا نه و یک instance از کلاس login که امور مربوط به login را مدیریت میکند، است. ابتدا در تابع connect درون کلاس server، تابع command parser صدا زده میشود تا دستور وارد شده مشخص شود. در این تابع دستور فرستاده شده از سمت client، جداسازی میشود، اصل دستور در رشته command و argument های دستور در بردار args ریخته می‌شود. پس از اتمام کار این متد، در تابع connect در کلاس server، یک ساختار try–catch داریم که در آن تابع handle کلاس command handler صدا زده میشود و exception هایی که در این تابع فرستاده می‌شوند را میگیرد و به عنوان پاسخ به client میفرستد. در متد handle هم

یک ساختار try¯catch استفاده شده است. در قسمت try مجموعه‌ای از if ها داریم تا تشخیص دهیم دستور اصلی چه دستوری است و بعد به ازای هر دستور عملیات مخصوص به آن انجام می‌شود.

دستور ها:

- User : این دستور همراه با یک username وارد می‌شود. سیستم باید چک کند آیا این username در سیستم وجود دارد یا نه. اگر وجود داشت منتظر ورود password میماند. اگر نه خطای Invalid username or password به client فرستاده شود. برای چک کردن درستی username از کلاس login و تابع find_username استفاده می‌کنیم.
- Pass : این دستور به همراه password وارد می‌شود. سیستم باید چک کند آیا passwordوارد شده با password کاربری که username آن وارد شده، یکی است یا خیر و اگر یکی بود user وارد سیستم می‌شود. دستور ورود به سیستم توسط تابع login در کلاس login انجام می‌شود.
- Pwd : این دستور برای یافتن دایرکتوری‌ای است که در آن هستیم. برای این کار از تابع ()system استفاده می‌کنیم. این تابع برای پاس دادن یک command به command processor یا terminal است که سپس command را پس از اتمام برمیگرداند. در واقع اگر در ورودی این تابع، command ای که می توان در terminal زد را وارد کنیم، این تابع آن کار را برایمان در terminal انجام می‌دهد. در اینجا ورودی ()system را pwd >> temp.txt داده‌ایم که این کار باعث می‌شود آدرس دایرکتوری‌ای که در آن هستیم را در یک فایل موقت می‌ریزد. سپس این فایل را باز می‌کنیم و محتوای آن را به عنوان exception برمیگردانیم. فایل temp.txt را پس از خواندن، remove می‌کنیم.
- Mkd : این دستور به همراه یک آدرس دایرگتوری وارد می‌شود. در این جا هم از تابع ()system استفاده میکنیم و دستور <mkdir ¯p <path را به آن می دهیم. در اینجا تابع system این دایرکتوری را برای ما می‌سازد.
- Dele : این دستور دو حالت دارد. در حالت ¯f باید فایل مورد نظر که نام آن هم وارد می‌شود پاک شود. برای این کار دستور <rm <filename را به تابع ()system می‌دهیم. در حالت دوم که ¯d است، باید دایرکتوری‌ای که آدرس آن داده شده است پاک شود. برای انجام این کار دستور <rmdir <directory را به تابع ()system می دهیم و این کار برای ما انجام داده می‌شود.
- Ls : این دستور برای نشان دادن لیست فایل ها و فولدر هایی است که در دایرکتوری کنونی وجود دارند، است. برای این کار هم دستور ls >> temp.txt را به تابع ()system می دهیم که باعث می‌شود نتیجه ls که لیست فایل ها هست را برای ما در فایل موقت temp.txt بریزد. محتوای این فایل را می‌خوانیم و آن را به عنوان exception برمیگردانیم تا به client ارسال شود. پس از خواندن از فایل temp آن را remove میکنیم.
- Cwd : این دستور مانند دستور cd در ترمینال عمل می‌کند. در اینجا نمی‌توانیم از تابع ()system استفاده کنیم زیرا این تابع یک process درست میکند و دستور مورد نظر را اجرا می‌کند. در اینجا این دستور نتیجه‌ای نخواهد داشت که بتوانیم برگردانیم به همین دلیل استفاده از ()systemباعث می‌شود اتفاقی نیافتد. برای انجام این دستور از تابع ()chdir استفاده می‌کنیم که ورودی دایرکتوری مورد نظر را می‌گیرد و دایرکتوری را به آن تغییر می‌دهد. در اینجا با یک چالش رو به رو هستیم. اگر دو client داشته باشیم که هردو به یک server متصل هستند باید حواسمان باشد که با وارد کردن دستور cwd در یک client، دایرکتوری کنونی client دوم نباید تغییر کند. برای کنترل این چالش، دو مقدار server_path و client_directory را در کلاس commandhandler نگه می‌داریم. پیش از انجام تمام دستورات،دایرکتوری را به client_directory تغییر می‌دهیم. هنگام وارد کردن دستور cwd بدون ورودی به server_pat، chdir انجام می‌دهیم. و برای انجام cwd با هر ورودی دیگری ابتدا یک chdir به client_directory و سپس chdir به دایرکتوری مورد نظر را انجام می دهیم و client_directory را با گرفتن pwd، update میکنیم.
- Rename : این تابع با دو ورودی from و to داده می‌شود. تابعی به نام rename استفاده می‌شود که نام قدیم و جدید فایل را می‌گیرد و نام را تغییر می‌دهد.

- Retr : این تابع برای دانلود فایل است و به همراه نام فایل مورد نظر می‌آید. فایل مورد نظر را ابتدا باز میکنیم و سپس توسط تابع sendfile، این فایل را به client میفرستیم. ورودی های این تابع، file descriptor سوکت داده ها، file descriptor فایل باز شده، جایی که می‌خواهیم فایل از آن جا خوانده شود و سایز فایل است. در اینجا باید توجه داشته باشیم که اگر حجم فایل از میزان حجمی که کاربر می‌تواند دانلود کند بیشتر باشد، خطای مربوطه به عنوان یک exception برگردانده می‌شود. همینطور پس از دانلود فایل، حجم فایل از میزان حجم مجاز دانلود کاربر کم می‌شود.
- Help : این دستور برای نشان دادن نحوه استفاده دستورات است. برای این کار یک فایل help.txt در فولدز server قرار دارد. این فایل در این قسمت خوانده می‌شود و محتوای آن باexception بازمیگردد تا به client ارسال شود.
- Quit : این دستور برای خارج شدن کاربر از سیستم است که توسط تابع quit در کلاس login انجام می‌شود.

نکته قابل توجه این است که در ۳ دستور retr، rename و delete باید چک کنیم که آیا فایل مورد نظر جرو فایل های سیستم است یا خیر. اگر بود باید چک کنیم که آیا کاربر ادمین سیستم است یا نه. اگر بود می‌توانیم به کاربر اجازه دسترسی دهیم. برای ان کار از دو تابع admin در کلاس user و in_system_files در کلاس command handler استفاده می‌کنیم.

کلاس login

این کلاس برای انجام کار های مربوط به login و logout است. در متد find_username، usernameای که با دستور user وارد شده را پیدا می‌کنیم و آن را در user object درون کلاس login نگه‌میداریم. سپس در تابع login چک می شود که passwordداده شده با دستور pass با password کاربری که نگه‌داشته‌ایم یکی باشد. اگر بود flag logged_in مقدار true را می‌گیرد. در تابع quit()، flag logged_in مقدار false را میگیرد.

کلاس user

این کلاس object های مربوط به کاربران را ارائه می دهد و با خواندن مشخصات کابران در ابتدای کار، برای هر یک از آن ها یک object از این کلاس می سازیم و در آن نام کاربری، رمز عبور، ادمین بودن یا نبودن کاربر و میزان حجم دانلود باقی مانده برای آن کاربر را نگهداری می کنیم. در صورتی که این کاربر فایلی را نیز دریافت کرد، به کمک تابع reduce_download_size حجم باقی مانده آن کاربر را کاهش می دهیم.