تمرین برنامه نویسی اول

SOCKET PROGRAMMING

FTP Server/Client

نحوه اجرا

با وارد کردن دستور زیر در پوشه اصلی پروژه ، فایل اجرایی کلاینت و سرور در محل فعلی ایجاد می‌شود .

```
make
```

ابتدا نیاز داریم سرور را اجرا کرده و سپس به سراغ کلاینت برویم . هر دو برنامه نیاز به آدرس فایل کانفیگ دارند که باید به صورت Programmer Argument به برنامه ها داده شوند . برای مثال اجرا برنامه‌ها به صورت زیر است .

```
./server.out Source/config.json
./client.out Source/config.json
```

در صورتی که آدرس فایل کانفیگ وارد نشده یا اشتباه وارد شده باشد ، برنامه اروری به کاربر نمایش می‌دهد و خاتمه پیدا می‌کند .

ساختار

کدهای برنامه‌ها در پوشه Source قرار گرفته است . این پوشه دارای سه پوشه دیگر است :

- Client : محتوی این پوشه شامل فایل‌های مورد نیاز برای ایجاد برنامه client می‌باشد
- Common : برخی فایل‌های مشترک میان برنامه server و client در این پوشه جای گرفته است
- Server : فایل‌های مورد نیاز برنامه server در این پوشه قرار دارد

شرح کدهای سرور و عملکرد آن‌ها

کلاس Command

این کلاس وظیفه بررسی دستورات وارد شده توسط کلاینت و پاسخ به آن‌ها را دارد .

- bool verify(string msg, bool count) : در این تابع به طور کلی صحت ورودی دستور ورودی بررسی می‌شود . این بررسی در دو حالت انجام می‌گیرد . حالت اول بررسی موجود بودن دستور بدون توجه به دیگر پارامترهای ورودی است . و در حالت دوم دستور با تعداد آرگومان‌های آن بررسی می‌شود که آیا تعداد آرگومان‌ها با نوع دستور همخوانی دارند یا خیر .
- bool verify(string msg, string cmd, int count) : این تابع بررسی صحت دستور ورودی با دستوری که ما انتظار داریم را ارزیابی می‌کند . همچنین تعداد آرگومان دستور در این تابع بررسی خواهد شد .

- bool verify(string msg, string cmd, string branch, int count) : این تابع نیز همانند تابع قبلی صحت ورودی را بررسی می‌کند اما برای دستوراتی مانند دستور dele که دارای دو شاخته هستند کاربر بیشتری دارد .
- void enterCredential(string msg, User *user) : در صورتی که یکی از دستورات user یا pass وارد شود ، این تابع فرخوانی خواهد شد و با ارسال دستور و کاربر مربوطه به این تابع ، مقادیر نام کاربری و کلمه عبور به آن کاربر اضافه خواهد شد .
- void response(int fd, int code) : این تابع وظیفه ارسال پاسخ دستوراتی که پاسخ آن‌ها ثابت است را دارد . با دریافت File Descriptor کاربر و کد دستور ، پاسخ دستور به کاربر ارسال می‌شود.
- void response(int fd, int code, string branch) : این تابع پاسخ دو دستور ls و retr را به دلیل مشابهت در کد پاسخ آن‌ها به کاربر ارسال می‌کند .
- void response(int fd, int code, string branch, string name) : در این تابع پاسخ به دستورات mkd ، pwd ، cwd ، rename و dele ارسال می‌شود . این دستورات در پاسخ خود نام فایل یا پوشه‌ای دارند که باید در پاسخ باشد . برای همین این تابع یک ورودی نام دارد .
- string getPath(string msg, int i) : به طوری کلی این تابع هر آرگومان از دستور را جدا می‌کند و می‌توان با دادن شماره آن را از این تابع دریافت کرد .

کلاس CommandExecutor

در این کلاس دستوراتی که سیستمی هستند و با فایل‌ها سر و کار دارد انجام می‌شود .

- string pwd() : به کمک توابع کتابخانه ای لینوکس ، مسیر فعلی را به صورت یک رشته برمی‌گرداند
- string cwd(string currentPath, string destination) : با گرفتن مسیر فعلی به مسیر مقصد می‌رود و مسیر جدید را به صورت یک رشته برمی‌گرداند . در صورتی که مسیر مقصد .. باشد به پوشه بالایی می‌رود و این کار را تا زمانی انجام می‌دهد که به مسیر / برسد . در صورتی که مسیر وارد شده .. نباشد بررسی می‌کند آیا مسیر وارد شده صحیح است یا خیر ، درصورت صحیح بودن مسیر جدید را برمی‌گرداند و در غیر این صورت رشته ثابت ارور را می‌دهد .
- bool mkd(string currentPath, string path) : این تابع در مسیر فعلی پوشه‌ای یجاد می‌کند . در صورتی که عملیات موفقیت آمیز باشد true و در غیر این صورت false می‌دهد .
- bool dele(string currentPath, string branch, string name) : این تابع فایل‌ها را به کمک توابع سیستمی لینوکس پاک می‌کند که درصورت موفقت true و در غیر این صورت false برمی‌گرداند . برای پاک کردن یک پوشه ابتدا به کمک تابع cwd از صحت وجود پوشه اطمینان

حاصل می‌کنم ، سپس به کمک تابع system و اجرا دستور rm با شاخه rf- در سیستم لینوکس ، پوشه مد نظر را پاک می‌کنیم .

- string ls(string currentPath) : لیست فایل‌ها و پوشه‌های مسیر فعلی را در قالب یک رشته برمی‌گرداند .
- bool rename(string currentPath, string currentName, string afterName) : در مسیر فعلی ، نام فایلی را تغییر می‌دهد و در صورت موفقیت true و در غیر این صورت false برمی‌گرداند .
- string help() : راهنما برنامه را در قالب یک رشته برمی‌گرداند .
- string getFileName(string path) : در صورتی که مسیری توسط کاربر وارد شده باشد که انتهای آن نام فایلی باشد ، این تابع نام فای را جدا کرده و برمی‌گرداند .
- long getFileSize(string path) : این تابع حجم یک فایل را بر مبنی بایت برمی‌گرداند . در صورتی که فایل وجود نداشته باشد ، ۱- برگردانده می‌شود .
- string getFileContent(string path) : در صورتی که نیاز به ارسال فایل به کلاینت باشد ، این تابع محتوی فایل را به صورت یک رشته در آورده و برمی‌گرداند .

کلاس Data

وظیفه ارسال داده در کانال داده به کاربران بر عهده این کلاس است .

- void response(int fd, string data) : این تابع File Descriptor کانال داده کاربر و دیتا مربوطه را گرفته و به کاربر ارسال می‌کند .

کلاس Logger

این کلاس وظیفه نمایش و ذخیره کردن رخدادهای برنامه را دارد . یک فایل با نام log.txt ساخته شده و در آن ذخیره می‌شود .

- string getDateTime() : این تابع تاریخ و ساعت فعلی را در قالب یک رشته برمی‌گرداند .
- void log(string username, string msg) : این تابع یک رشته رخداد تولید کرده و آن را در فایل رخدادها ذخیره کرده و آن را در کنسول سرور نمایش می‌دهد . رخداد تولید شده دارای تاریخ و ساعت ، نام کاربری و عملیات انجام شده است .
- void log(string username, string msg, string argument) : این تابع یک رشته رخداد تولید کرده و آن را در فایل رخدادها ذخیره کرده و آن را در کنسول سرور نمایش می‌دهد . رخداد تولید شده دارای تاریخ و ساعت ، نام کاربری ، عملیات انجام شده و نام پوشه یا فایل است .

کلاس Server

این کلاس ، کلاس کلی و مدیریت کننده سرور ما می‌باشد . پیاده سازی این کلاس به صورت Singletone است ، چرا که ما نباید بیش از یک نمونه از کلاس سرور در برنامه داشته باشیم .

در این کلاس کاربرها و فایل‌های ادمین وجود دارد .

- int acceptClientCommand() : این تابع یک File Descriptor برای ارتباط در کانال دستور به کاربر اختصاص می‌دهد .
- int acceptClientData() : این تابع یک File Descriptor برای ارتباط در کانال داده به کاربر اختصاص می‌دهد .
- User *findUser(int fd, fileDescriptor type, vector<User *> _users_) : این تابع به دنبال کاربری با File Descriptor مشخصی در لیست ورودی می‌گردد .
- User *findUser(string username, vector<User *> _users_) : این تابع به دنبال کاربری با نام‌کاربری مشخص در لیست ورودی می‌گردد .
- User *findUser(string username, string password, vector<User *> _users_): این تابع برای احراز هویت کاربر انجام می‌شود . در این تابع بررسی می‌کینم آیا کاربری با نام‌کاربری و رمزعبور داده شده وجود دارد یا خیر .
- void removeUser(int fd, fileDescriptor type) : زمانی که کاربر از سامانه خارج شود ، کاربر به کمک این تابع از لیست حذف می‌شود .
- bool canAccess(User *user, string name) : در این تابع بررسی می‌کینم آیا کاربر اجازه دسترسی به فایل مورد نظر را دارد یا خیر . ادمین بودن یا نبودن کاربر در این بررسی اهمیت دارد .
- void init(string path) : این کلاس با دریافت مسیر فایل کانفیگ ، داده‌های مورد نیاز را به کمک کتابخانه متن باز json خوانده و در کلاس سرور ذخیره می‌کند .
- void startServer() : در این کلاس دو کانال ارتباطی ، یکی برای دستور و دیگری برای داده ، ایجاد می‌شود و File Descriptor آن‌ها برای استفاده‌های بعدی ذخیره می‌شود .
- void listenCommand() : این کلاس نقطه ارتباط کانال دستور با کلاینت‌ها است . در این تابع منتظر دریافت دستور از کاربران می‌مانیم ، در صورتی که کاربر هنوز وارد نشده باشد ، در لیست newUsers قرار گرفته و File Descriptor به آن اختصاص داده می‌شود و پس از ورودی موفقیت آمیز به لیستی اصلی کاربرها منتقل می‌شود . این تابع به کمک فراخوانی سیستمی select ارتباط با چندید کلاینت را مدیریت می‌کند . هر کاربر در یک مرحله است که جلوتر به توضیح مراحل خواهیم پرداخت . بر اساس مرحله کاربر ، پاسخ‌ها و عملکرد سرور متفاوت خواهد بود . در صورتی که

کاربر هنوز وارد نشده باشد ، پاسخ به بسیاری از دستورات او با ارور مواجه می‌شود . به ازای هر دستور یک پاسخ ارسال و یک رخداد در سمت سرور ایجاد می‌شود . نحوه عملکر ارسال پاسخ و ذخیره رخداد پیشتر توضیح داده شده است .

ارسال داده با توجه قراردادی بودن این ارتباط به این شرح است . خط اول داده نوع آن را مشخص می‌کند . این نوع می‌تواند نمایش در کنسول ، ذخیره به صورت فایل یا ارور باشد که این نوع در سمت کلاینت مورد استفاده قرار خواهد گرفت . در صورتی که نوع فایل باشد ، محل ذخیره و نام فایل در خط دوم خواهد آمد . در ادامه داده ارسالی ، فرستاده می‌شود .

- void listenData() : این تابع منتظر برقراری ارتباط برای کانال داده است . سوکت کانال داده زمانی شکل می‌گیرد که کاربر با موفقیت وارد شده باشد . قبل از این ، ارتباط کانال داده برقرار نیست. عملکر این تابع نیز مشابه تابع قبل است .

کلاس User

این کلاس به عنوان یک کلاس جهت نگه‌داری داده‌های هر کاربر استفاده می‌شود .

این کلاس حاوی نام‌کاربری ، کلمه عبور ، ادمین بودن یا نبودن ، حجم کاربر ، مسیر فعلی کاربر ، مرحله کاربر و File Descriptor دستورات و داده‌ها است .

مرحه یک کاربر عبارت است از :

- ENTER_USER : این مرحله ، مرحله ابتدایی هر کاربر است که منتظر ورودی نام‌کاربری هستیم
- ENTER_PASSWORD : این مرحله پس از ورود نام‌کاربری است و منتظر وارد کردن کلمه عبور هستیم . در صورتی که نام‌کاربری و رمز عبور درست باشد به مرحله LOGGED_IN می‌رویم ، در غیر این صورت به مرحه ENTER_USER می‌رویم .
- LOGGED_IN : در این مرحه کاربر وارد سامانه شده است و می‌تواند بقیه دستورات را وارد کند . کاربر با وارد کردن دستور quit به مرحله ENTER_USER می‌رود .

فایل main

در این فایل تابع init سرور صدا می‌شود تا سرور پیکربندی شود سپس در صورت پیکربندی صحیح ، سرور شروع به کار می‌کند . در این تابع دو Thread جدید به برنامه اضافه می‌شود . یکی برای دریافت دستورات کانال دستور و دیگری برای دریافت درخواست اتصال کانال داده . دلیل استفاده از دو Threadدریافت همزمان دستور و درخواست اتصال کانال داده است .

شرح کدهای کلاینت و عملکرد آن‌ها

کلاس Client

پیاده سازی این کلاس به صورت Singletone است ، چراکه نباید بیش از یک نمونه از کلاس کلاینت در برنامه موجود باشد . در این کلاس نام‌کاربری و کلمه عبور کاربر به همراه File Descriptor کانال داده و دستور ذخیره می‌شود .

- int connectServer(int port) : این تابع با دریافت پورت ، به سرور مربوطه وصل می‌شود و File Descriptor را برمی‌گرداند .
- bool responseCode(string msg, int code) : این تابع کد پاسخ کانال دستور را با کد مورد نظر مقایسه و نتیجه برابر بودن یا نبودن را برمی‌گرداند .
- void receiveDataResponse() : این تابع پاسخ کانال داده را دریافت و تفسیر می‌کند . در صورتی که نوع داده نمایش در کنسول باشد ، در کنسول نمایش می‌دهد و در صورتی که فای باشد، فایلی در مسیر فعلی با نام مشخص ساخته و محتوی دریافتی را در آن ذخیره می‌کند . سپس به دریافت پاسخ از کانال دستور می‌پردازد .
- void init(string path) : این تابع با ورودی گرفتن آدرس فایل کانفیگ پورت اتصال کانال داده و دستور را به کمک کتابخانه متن باز json خوانده و ذخیره می‌کند .
- void startCommand() : این تابع ، تابع connectServer را با پورت کانال دستور صدا می‌زند و File Descriptor کانال دستور را ذخیره می‌کند .
- void startData() : این تابع زمانی صدا زده می‌شود که نام‌کاربری و کلمه‌عبور کاربر تاییده شده باشد . این تابع نام‌کاربری و کلمه‌عبور کاربر را به کانال دستور ارسال می‌کند تا ارتباط برقرار شود .
- void sendCommand() : این تابع ، ورودی کاربر را از کنسول خوانده و به سرور ارسال می‌کند، این تابع همچنان نام‌کاربری و کلمه‌عبور کاربر را ذخیره می‌کند . در انتها برای دستوراتی که نیاز به پاسخ در کانال داده باشد ، ادامه کار را به تابع receiveDataResponse محول می‌کند .
- void receiveCommandResponse() : این تابع پاسخ دریافتی در کانال دستور را از سرور دریافت و در کنسول چاپ می‌کند . همچنین با بررسی پاسخ دستور ، در صورت صحت نام‌کاربری و کلمه‌عبور ارتباط برای برقراری ارتباط کانال داده اقدام می‌کند و در صورت دریافت پاسخ دستور quit برنامه را خاتمه می‌دهد .

فایل main

در تابع اصلی این فایل پس از صحت پیکربندی ، حلقه از ارسال و دریافت دستورات شکل می‌گیرد .