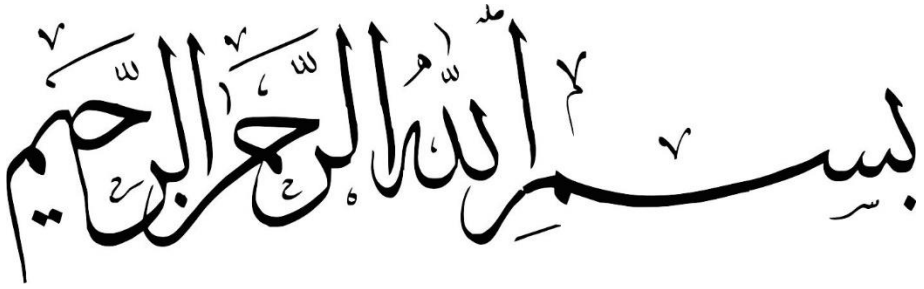

تمرین برنامه نویسی شماره ۰۱

عنوان: پروژه برنامه نویسی- شماره ۰۱

درس: شبکه‌های کامپیوتری

استاد راهنما: دکتر ناصر یزدانی[۱]

رشته: مهندسی کامپیوتر

دستیاران آموزشی: اسامه ایراندوست[۲]

نیمسال اول سال تحصیلی ۰۲-۱۴۱

[۱] نشانی پست الکترونیکی: yazdani@ut.ac.ir

[۲] نشانی پست الکترونیکی: osameh.irandoust@ut.ac.ir

عنوان پروژه

این تمرین شامل ۳ مینی پروژه است که هدف آشنایی با مفاهیم اولیه Socket programming و نحوه ارتباط چندین برنامه است که همه بر روی یک Local Host ولی با Portهای مختلف هستند.

- **FTP سرور**: ورژن ساده شده‌ی FTP که از دو بخش کلاینت و سرور ایجاد شده. هدف آپلود و دانلود فایل تا حجم مشخصی است.
- **وب سرور**: یک وب سرور ساده که هدف از انجام آن آشنایی با وب و سروری است که در بک اند مدیریت می‌شود. در این پروژه نمایش یک سری type از contentهای مختلف را یاد خواهید گرفت.
- **چت سرور**: Chatroom ساده‌ای که هدف از انجام آن آشنایی با یک سرور و چند کلاینت به صورت همزمان است.

FTP سرور

در این پروژه شما به پیاده سازی یک ورژن ساده شده‌ی FTP می‌پردازید. این برنامه از دو بخش کلاینت(Client) و سرور(Server) تشکیل شده است که سرور وظیفه‌ی خدمت رسانی فایل‌ها را به کلاینت دارد. FTP به ارتباط دو کانال مختلف برای ارسال و دریافت داده‌ها نیاز دارد. کانال اول که کانال دستور نامیده می‌شود کانالی است که دستورها و پاسخ‌ها از آن رد می‌شود. کانال دیگر که کانال داده نامیده می‌شود، کانالیست که وظیفه جابه‌جایی داده را دارد. هدف از انجام این پروژه آپلود و دانلود فایل از طریق buffer تا حجم 1024 Kbytes می‌باشد.

۱- مقدمه

شما باید در این پروژه یک برنامه‌ی سرور با دو کانال ارتباطی ایجاد کنید. این دو کانال ارتباطی می‌توانند روی دو پورت دلخواه که از طریق فایل کانفیگ به برنامه‌ی سرور داده میشوند اجرا شوند. سپس شما به پیاده‌سازی یک کلاینت که از طریق دو پورت دلخواه به این کانالها متصل می‌شود می‌پردازید.

توجه داشته باشید که سرور وظیفه سرو کردن فایل‌های کامپیوتری که در آن در حال اجراست را دارد.همین‌طور محلی که برنامه قرار دارد و اجرا می‌شود به عنوان دایرکتوری اولیه در نظر گرفته می‌شود.

۲- احراز هویت[۳] و مدیریت دسترسی[۴]

برای احراز هویت در سرور باید از فایل کانفیگ سرور، اطلاعات کاربران سیستم را خوانده و با استفاده از پسورد آن‌ها را شناسایی و احراز هویت کنند. فایل کانفیگ JSON است و می‌توانید از کتابخونه‌های open source برای parse کردن دیتای آن استفاده کنید.

```
{
"users": [
  {
      "user": "Mohsen",
      "password": "1234"
  },
  {
      "user": "Ali",
      "password": "1234"
  }
 ]
}
```

در کلاینت برای اینکه کاربر وارد شود ابتدا نام کاربری خود را مشابه دستور زیر وارد کند:

```
user <username>
```

[۳] Authentication
[۴] Authorization

در صورتی که نام کاربری قابل قبول بود پاسخ زیر داده می‌شود:

```
331: User name okay, need password.
```

سپس سرور منتظر رمزعبور کاربر می‌ماند. در این حالت کلاینت باید رمز عبور را برای سرور ارسال نماید:

```
pass <password>
```

در صورتی که کلاینت قبلا درخواست ورود نداده بود پاسخ زیر داده می‌شود:

```
503: Bad sequence of commands.
```

در صورتی که کاربر با موفقیت وارد شود پاسخ زیر داده می‌شود:

```
230: User logged in, proceed. Logged out if appropriate.
```

در صورتی که هرکدام از رمزعبور یا نام کاربری ایراد داشتند و قابل قبول نبودند پاسخ زیر داده می‌شود:

```
430: Invalid username or password
```

در ادامه‌ی این بخش قصد داریم یک مکانیزم مدیریت دسترسی پیاده‌سازی کنیم به طوری که دو نوع کاربر در سیستم وجود داشته باشند:

۱- ادمین ۲- کاربر عادی.

کاربران عادی تنها مجاز به دانلود فایل هستند و نمی‌توانند چیزی را آپلود نمایند و این درحالی است که ادمین سیستم می‌تواند به همه‌ی فایل‌ها دسترسی داشته باشد. در فایل کانفیگ سرور، لیست فایل‌هایی که فقط ادمین‌های سیستم به آن‌ها دسترسی دارند آمده است. در صورتی که از سوی کاربر سیستم دسترسی غیر مجازی انجام شد، پیغام مبنی بر عدم در دسترس بودن فایل به شکل زیر نمایش داده شود:

```
550: File unavailable.
```

۳- دانلود فایل

این دستور فایل گفته شده را در صورتی که موجود باشد دانلود می‌کند:

```
retr <name>
```

فایل از طریق کانال داده برای کلاینت ارسال می‌شود و پس از آن که انتقال فایل کامل شد پاسخ زیر از طریق کانال دستور ارسال می‌شود:

```
226: Successful Download.
```

۴- آپلود فایل

این دستور فایل داده شده(که می‌تواند در هر Locationی باشد) را در دایرکتوری اصلی برنامه و در پوشه‌ای به نام Files بارگذاری می‌کند. حتما دقت داشته باشید که حجم فایل از 1024 Kbytes بیشتر نشود.

```
Upload <name>
```

فایل از طریق کانال داده برای کلاینت ارسال می‌شود و پس از آن که انتقال فایل کامل شد پاسخ زیر از طریق کانال دستور ارسال می‌شود:

```
226: Successful Download.
```

۵- راهنما

این دستور، دستورات موجود در سرور را به همراه راهنمای استفاده از آن‌ها به کاربر نمایش می‌دهد:

```
help
```

پاسخ این دستور در کانال دستور ارسال می‌شود و به عنوان مثال مانند شکل زیر است. شما باید همه دستورات را مانند دستور زیر در پاسخ این دستور ارسال کنید:

```
214
USER [name], Its argument is used to specify the user's string. It is used for user authentication.
```

۶- خارج شدن از سرور

این دستور کاربر فعلی را از سیستم خارج می‌نماید:

```
quit
```

پاسخ این دستور در کانال دستور به شکل زیر ارسال می‌شود:

```
221: Successful Quit.
```

۷- مدیریت خطاها

در تمامی حالات اگر کاربری هنوز وارد نشده بود و دستورات را وارد کرد ، سرور باید پاسخ زیر را برگرداند:

```
332: Need account for login.
```

در تمامی حالات اگر ایراد نگارشی در پارامترهای یک دستور وجود داشت باید پاسخ زیر داده شود:

```
501: Syntax error in parameters or arguments.
```

در صورتی که خطای دیگری رخ داد پاسخ زیر را در کانال دستور ارسال کنید:

```
500: Error
```

۸- مدیریت حجم کاربران

در فایل کانفیگ سرور حجم مجاز مصرفی[۵] هر کاربر آورده شده‌است.(واحد آن Kbytes می‌باشد) در هنگام ورود دستور دانلود یک فایل در صورتی که کاربر حجم کافی برای دانلود آن را داشت، فایل را دانلود کرده و مقدار حجم فایل از حجم مجاز کاربر کم می‌شود، و در در غیر این صورت پیغامی مبنی بر کافی نبودن حجم کاربر از سرور پاسخ داده می‌شود:

```
425: Can't open data connection.
```

۹- کلاینت

شما باید یک کلاینت پیاده سازی کنید تا با استفاده از دستورات گفته شده با سرورتان بتواند ارتباط برقرار کند و این دستورات را انجام دهد.

همین طور توجه داشته باشید برای راحتی پاسخ‌هایی که از طریق کانال داده ارسال می‌شود می‌تواند به صورت قراردادی بین سرور و کلاینت از هر فرمتی به دلخواه خودتان پیروی کند.

۱۰- لاگ فایل

سرور شما در زمان اجرا باید یک فایل log در کنار خود ایجاد کند و تمامی اطلاعات را با تاریخ و ساعت وقوع در آن ذخیره کند. اطلاعاتی مانند افرادی که وارد سیستم شده‌اند؛ فایل‌هایی که ساخته، پاک کرده و یا دانلود کرده‌اند.

دقت کنید با هر بار اجرای سرور اگر فایل log وجود نداشت آن را بسازید و در صورتی که از قبل این فایل وجود داشته باشد، در ادامه‌ی آن شروع به نوشتن کنید.

جمع بندی و نکات نهایی مینی پروژه اول

- برنامه شما باید درخواست‌ها را به صورت همزمان پاسخ دهد و چندین کلاینت بتوانند با کاربران مختلف به سرور وصل شوند. (می‌توانید از thread یا select() system call به دلخواه خودتان استفاده کنید)

[۵] بر اساس KBytes

وب سرور

در این مینی پروژه، می‌خواهیم یک Web Server ایجاد کنیم. هدف آشنایی شما با مباحث وب و سرورهای محلی است.

۱- مقدمه

پروژه شامل دو قسمت A و B است. که هرکدام از آن‌ها به شرح زیر هستند:

- Part A: یک وب سرور را implement کنید که request messageها را به console می‌فرستد. این به شما کمک می‌کند تا نحوه عملکرد HTTP را مشاهده کنید. بنابراین یک مروگر را اجرا کنید (می‌توانید از هر مرورگری استفاده نمایید). سپس به سرور خود متصل شوید و سرور درخواست شما را می‌شنود و در محیط کنسول نمایش می‌دهد.
- Part B: بر اساس برنامه‌ای که در قسمت قبل نوشتید یک فانکشن دیگر به آن اضافه نمایید. اینبار درخواستی که سرور دریافت می‌کند را باید بتوانید در مرورگر که نقش کلاینت را برعهده دارد نمایش دهید. پس در این قسمت شما باید یک HTTP response message داشته باشید که بتواند آدرس درخواستی را دریافت کند و مستقیما به کلاینت(مرورگز) بفرستد.

۲- دستورالعمل‌ها

به موارد زیر دقت کنید:

- وب سرور شما فقط command line زیر را خواهد داشت:

```
%your_server:<Port number>
```

Hint: <port number> is a port number of your web server for the listening socket. This way you can input your server port number when you start the web server

- اگر سرور و مرورگر شما بر روی همان سیستم در حال اجرا است می‌توانید از ۱۲۷.۰.۰.۱ یا localhost به جای %your_server استفاده کنید.
- مطمئن شوید که فانکشن Content_type شما حداقل فایل‌های html را پشتیبانی می‌کند. بعدا در مورد این قسمت مواردی را اضافه خواهیم کرد.
- بعد از اتمام دو قسمت اصلی این پروژه، شما نیاز است سرور خو درا تست کنید. ابتدا شما فایل index.html خود را در محل پروژه و کدهای خود کپی کنید. سپس از طریق مرورگر خود و وارد کردن دستور زیر آن را تست کنید:

```
http://%your_server:<Port number>/index.html
```

- حال برای بهتر کردن پروژه با اجرای فرمان زیر redirect صورت بگیرد و فایل index.html نمایش داده شود:

```
http://%your_server:<Port number>
```

- مدیریت خطا را به آن اضافه کنید. یک فایل 404.html بنویسید که در صورت واردکردن آدرس اشتباه آن را نمایش دهد.
- برگردیم به Content-typeها، حال به سرور خود چند نوع داده دیگر اضافه نمایید. شما باید فرمت‌های GIF، JPEG، MP3 و PDF را نیز نمایش دهید و از هرکدام از این فرمت‌ها حداقل یک نمونه به نمایش بگذارید.

جمع بندی و نکات نهایی مینی پروژه دوم

- به جای استفاده از پورت‌های معمول مانند ۸۰ یا ۸۰۸۰ یا ۶۷۸۹ برای گوش دادن به سوکت، شما می‌توانید از سایر پورت‌هایی که توسط سیستم اشغال نشده اند استفاده نمایید تا به مشکلی برخورد نکنید. شما باید این پورت سرور را به عنوان ورودی در دستور اجرای وب سرور خود قرار دهید. همچنین پیشنهاد می‌شود از پورت ۰ تا ۱۰۲۴ استفاده نکنید.

چت سرور

در این قسمت سرور که یک Chatroom ساده است، به همراه نحوه ارتباط با آن در اختیار شما قرار می‌گیرد و شما یک کلاینت برای آن پیاده سازی می‌کنید. هر کلاینت یک ارتباط TCP با سرور برقرار می‌کند و این ارتباط باید تا انتهای برنامه باز باشد.

۱- مقدمه

ساختار کلی پیام‌ها و پاسخ‌های سرور به صورت زیر است:

Message Type (4b)	Message ID (4b)	Length (1B)
Payload		

در این ساختار Message Type می‌تواند مقادیر زیر را که در ادامه هر کدام از آن‌ها نیز معرفی خواهند شد داشته باشد. Message ID یک عدد متفاوت برای هر پیام است که امکان می‌دهد کانکشن TCP بین پیام‌های مختلف به اشتراک گذاشته شود. اگر پس از هر درخواست منتظر پاسخ سرور می‌مانید مقدار Message ID اهمیتی ندارد. Length هم طول کل پیام را مشخص می‌کند.

Message Type	Value
CONNECT	1
CONNACK	2
LIST	3
LISTREPLY	4
INFO	5
INFOREPLY	6
SEND	7
SENDREPLY	8
RECEIVE	9
RECEIVEREPLY	10

کلاینت پس از اتصال به سرور اسم خود را با استفاده از پیام CONNECT برای سرور ارسال می‌کند. در پاسخ سرور پیام CONNACK را ارسال می‌کند. در صورت عدم ارسال این دستور کانکشن توسط سرور بسته می‌شود. تا زمان بازبودن ارتباط کلاینت این نام در سرور معتبر باقی خواهد ماند.

CONNECT	Message ID	2+length(name)
name		
CONNACK	Message ID	2

هر کلاینت می‌تواند با پیام LIST شناسه تمام کاربران را از سرور درخواست کند. در پاسخ به این دستور سرور لیست تمام کاربران را برای کلاینت با پیام LISTREPLY ارسال می‌کند. هر عضو این لیست شامل یک عدد ۲ بایتی به عنوان شناسه کاربر است.

LIST	Message ID	2
LISTREPLAY	Message ID	2+2*n
User ID (2B) * n		

هر کلاینت می‌تواند با پیام INFO شناسه کاربر را به سرور ارسال کند و نام کاربر را بدست آورد. در پاسخ سرور پیام INFOREPLY را برای کلاینت ارسال می‌کند و در Payload آن نام کاربر قرار می‌گیرد. اگر طول Payload صفر باشد یعنی کاربر در سرور وجود ندارد.

USERINFO	Message ID	2 + 2
User ID		
USERINFOREPLY	Message ID	2+length(user name)
User Name		

کلاینت برای ارسال پیام از دستور SEND استفاده می‌کند. این پیام شامل شناسه کاربری که باید پیام برایش ارسال شود و متن پیام است. سرور با پیام SENDREPLY پاسخ می‌دهد. وضعیت ارسال پیام در پاسخ وجود دارد.

SEND	Message ID	2 + 2+length(message)
User ID		
Message		
SENDREPLY	Message ID	2+1
Status (0: success, 1: failure)		

کلاینت برای دریافت پیام‌هایی که برایش ارسال شده‌اند از پیام RECEIVE استفاده می‌کند. سرور در پاسخ از RECEIVEREPLY استفاده می‌کند و پیامی که برای کاربر ارسال شده است را برای کلاینت می‌فرستد. اگر پیامی وجود نداشته باشد Sender ID برابر با صفر و Payload خالی خواهد بود.

RECEIVE	Message ID	2
RECEIVEREPLY	Message ID	2+2+length(message)
Sender ID		
Message		

۲- انتظارات از کلاینت

برنامه کلاینت در زمان اجرا آدرس سرور (با فرمت Host:Port) و نام کاربر را به عنوان آرگومان می‌گیرد.

برنامه کلاینت باید ۳ دستور را در اختیار کاربر قرار دهد:

- List: کلاینت از سرور لیست تمامی کاربران را دریافت می‌کند و نمایش می‌دهد.

```
>> list
   - <User Name 1>
   - <User Name 2>
   ...
```

- Send: پیغامی برای یکی از کلاینت‌ها ارسال کند. این دستور دو ورودی دارد که یکی نام گیرنده و دیگری متن پیام است.

```
>> send <User Name> <Message>
```

- Exit: از برنامه خارج می‌شود.

همچنین در هر زمان که پیامی از دیگران برای کلاینت ارسال شد برنامه باید آن را به شکل زیر نمایش دهد:

```
<< <User Name>: <Message>
```

در ادامه نیز مثالی برای تفهیم بیشتر آورده شده است:

```
$ ./client localhost:9000 ali
>> list
    - sirous
>> send sirous Hello
<< sirous: Hi
>> exit
```

نکات پایانی

- **مهلت انجام پروژه:** ۱۰ آذرماه ۱۴۰۱
- برای پیاده سازی تمامی پروژه‌ها از C یا C++ استفاده کنید.
- پروژه در گروه‌های ۲ نفره انجام می‌شود.
- هر ۲ نفر می‌بایست کار را تقسیم کنند. همچنین از Git برای ساختن branch و تقسیم issueها استفاده نمایید.(با استفاده از commit ها و تعیین issueها میزان مشارکت هر نفر تعیین می‌شود) پروژه اصلی شامل ۳ ماژول اصلی خواهد بود که هر ماژول یکی از مینی پروژه ها را شامل می‌شود.(از CMake می‌توانید استفاده کنید.) و در هر فولدر نیز کدهای مربوط به آن مینی پروژه قرارخواهد گرفت. بعد از انجام این کار کدها را در یک repository به نام CN_CHomeworks1 در اکانت‌های github/gitlab خود قرار دهید(به صورت public باشد). همچنین در یک فایل readme.md نیز می‌توانید report و document خود را کامل کنید و همراه repository قرار دهید. در نهایت فقط لینک مربوط به همین repository را در محل پاسخ تکلیف ارسال کنید. (از ارسال فایل به صورت زیپ خودداری نمایید)
- سیستم عامل مورد استفاده Linux باشد.
- ساختار صحیح و تمیزی کد برنامه، بخشی از نمره‌ی این پروژه شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- برای هر قسمت کد، گزارش دقیق و شفاف بنویسید. کدهای ضمیمه شده بدون گزارش مربوطه نمره‌ای نخواهند داشت.
- دقت کنید گزارش نهایی می‌بایست همانند یک Document باشد و شامل توضیح کد و ساختار کد، همچنین نتیجه نهایی اجرای کد و اسکرین شات‌های دقیق از تمام مراحل باشد (در فایل readme.md کنار فایل‌های اصلی خود و در repository مربوطه قرار دهید). این نکته حائز اهمیت است که pdf مورد قبول نیست.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، مطابقت سیاست درس با گروه متقلب و تقلب دهنده برخورد خواهد شد.
- سؤالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن بهره‌مند شوند. در صورتی که قصد مطرح کردن سؤال خاص‌تری دارید، از طریق ایمیل زیر ارتباط برقرار کنید. توجه داشته باشید که سایر شبکه‌های اجتماعی راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی مؤظف به پاسخگویی در محیط‌های غیررسمی نیستند.
  - Osameh.irandoust@ut.ac.ir

موفق باشید - ایراندوست