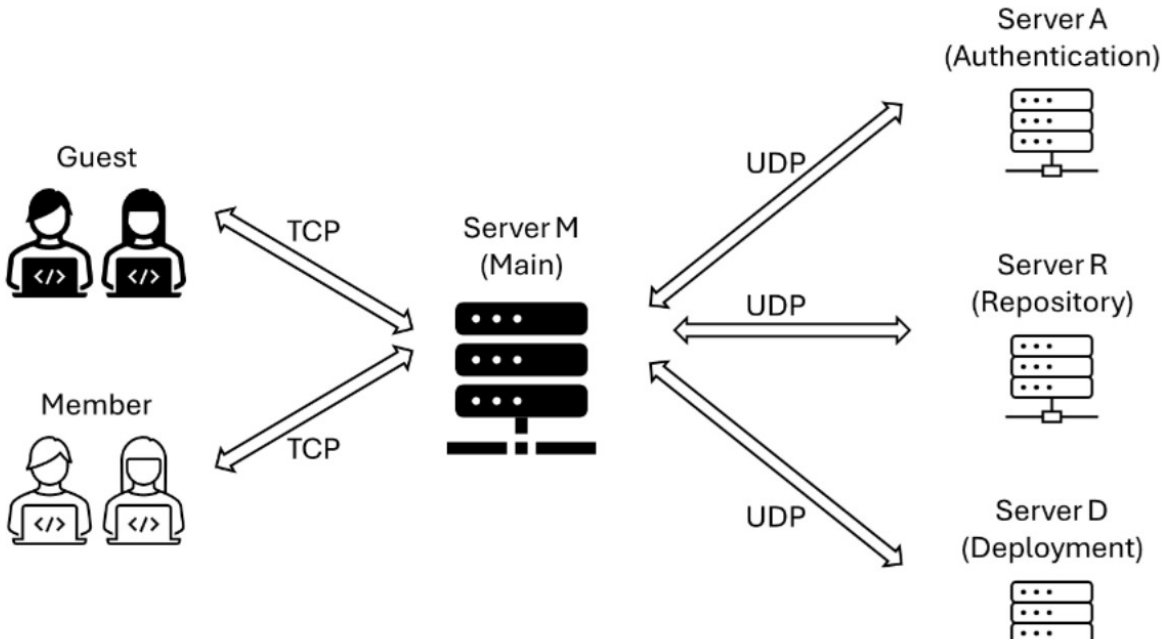
پروژه

گیت‌هاب یک پلتفرم مدیریت کد است که بر اساس سیستم کنترل نسخه گیت ساخته شده و عمدتاً برای توسعه نرم‌افزار و کنترل نسخه استفاده می‌شود. این پلتفرم همکاری را تسهیل کرده، تغییرات کد را پیگیری می‌کند، پروژه‌ها را مدیریت می‌کند، مخزن‌ها را به اشتراک می‌گذارد و جریان‌های کاری را اجرا می‌کند. ما یک نسخه طراحی خواهیم کرد که به‌طور مؤثر فایل‌ها را مدیریت خواهد کرد. نسخه Git450 به احراز هویت ساده کاربران، مدیریت مخزن‌ها (با این فرض که هر کاربر فقط یک مخزن دارد) و اجرای استقرار کمک خواهد کرد. علاوه بر این، از آنجایی که امنیت کاربران بسیار مهم است و برای اطمینان از ایمنی کاربران، نام کاربری و رمز عبور ورود به سیستم را رمزگذاری خواهد کرد تا محیطی ایمن، قابل اعتماد و کاربرپسند فراهم کند.

برای پروژه Git450، ما به سه سرور بک‌اند نیاز داریم: سرور A، سرور R و سرور D. کاربران قادر خواهند بود تا از طریق رابط کاربری مختلف اقداماتی مانند احراز هویت، ارسال، استقرار، جستجو و حذف را انجام دهند و این اقدامات از طریق سرور اصلی به سرورهای بک‌اند مربوطه ارسال خواهند شد.

- کلاینت: رابط Git450 به کاربران این امکان را می‌دهد تا اقداماتی مانند احراز هویت، ارسال فایل، استقرار، جستجو و حذف را انجام دهند.
  - اعضای سایت: می‌توانند وارد شوند، فایل‌ها را ارسال کنند، برنامه‌ها را استقرار دهند، فایل‌ها را جستجو کنند و آنها را حذف کنند.
  - مهمان‌ها: فقط می‌توانند نام فایل‌های یک عضو خاص را که در سرور R ذخیره شده است، جستجو کنند. همه فایل‌ها در مخزن به‌طور پیش‌فرض عمومی فرض می‌شوند. برای ورود، مهمان‌ها باید "guest" را به‌عنوان نام کاربری و رمز عبور وارد کنند.

- سرور اصلی: تمامی اقدامات مهمان و عضو را مدیریت کرده و درخواست‌ها را به سرورهای بک‌اند مربوطه ارسال می‌کند.
- سرورهای بک‌اند: سرور A، سرور R و سرور D برای احراز هویت کاربران، مدیریت مخزن و اجرای استقرار به‌طور جداگانه استفاده می‌شوند. هنگامی که سیستم شروع به کار می‌کند، سرور A فایل member.txt را برای احراز هویت اعضا می‌خواند و سرور R مخزن را با استفاده از فایل filenames.txt راه‌اندازی می‌کند.



شکل 1. تصویر شبکه

فایل کدها:

اجرای شما باید شامل فایل‌های زیر باشد، برای هر جز از سیستم:

1. کلاینت: نام این کد باید client.c یا client.cc یا client.cpp (همه حروف کوچک) باشد و فایل هدر (اگر وجود دارد؛ الزامی نیست) باید client.h (همه حروف کوچک) نامگذاری شود.

2. سرور اصلی: شما باید فایل کد خود را با نام serverM.c یا serverM.cc یا serverM.cpp (همه حروف کوچک به‌جز "M") نامگذاری کنید. همچنین، باید فایل هدر مربوطه (اگر وجود دارد؛ الزامی نیست) به نام serverM.h (همه حروف کوچک به‌جز "M") گنجانده شود.

3. سرورهای بک‌اند A، R، D: شما باید سه فایل مجزا ایجاد کنید که از الگوهای نام‌گذاری زیر استفاده کنند: server#.c یا server#.cc یا server#.cpp. نام فایل باید یکی از این فرمت‌ها را به‌کار گیرد، با جایگزینی "#" با شناسه سرور خاص (مانند "A" یا "R" یا "D") تا نشان دهد که این سرور نماینده کدام است، نتیجه نام‌گذاری‌هایی مانند serverA.c، serverA.cc، serverA.cpp، serverR.c، serverR.cc یا serverR.cpp (توجه کنید که نام باید به جز حرف جایگزین "#" به‌صورت تمام حروف کوچک باشد). در صورت امکان، باید یک فایل هدر مربوطه با نام server#.h که مطابق با همان قوانین نام‌گذاری است، گنجانده شود.

توجه: شما مجاز به استفاده از یک برنامه اجرایی برای همه چهار سرور نیستید (یعنی پیاده‌سازی مبتنی بر "fork" مجاز نیست).

فایل‌های ورودی

- "members.txt" : این فایل در سرور A (سرور احراز هویت) قرار دارد که اطلاعات کاربری (نام‌های کاربری و رمزهای عبور به همین ترتیب) را نگه‌داری می‌کند. فرمت این فایل شامل دو ستون، یکی برای نام‌های کاربری و دیگری برای رمزهای عبور است که با یک فاصله " " از هم جدا شده‌اند. اطلاعات هر کاربر در یک خط جدید ذخیره می‌شود. فایل members.txt برای احراز هویت قبل از اعطای مجوزهای استقرار، ارسال و حذف به کاربران استفاده می‌شود.

- "filenames.txt" : سرور R (سرور مخزن) فایل filenames.txt را نگه‌داری می‌کند که شامل مجموعه‌ای از نام فایل‌های کاربران است. این فایل دارای دو ستون است: یکی برای نام‌های کاربری و دیگری برای نام فایل‌ها. هر زمان که کاربر یک فایل جدید ارسال کند، متا دیتای آن (نام کاربری و نام فایل) در یک خط جدید به این فایل اضافه می‌شود.

توضیحات بیشتر:

فاز 1A: (20 امتیاز)

لطفاً به بخش "فرآیند" مراجعه کنید تا برنامه‌های خود را به ترتیب زیر شروع کنید: سرور اصلی (serverM)، سرور A، سرور R، سرور D و حداقل دو کلاینت (یک کلاینت عضو و یک کلاینت مهمان). برنامه‌های شما باید به همین ترتیب آغاز شوند. هر یک از سرورها و کلاینت‌ها پیام‌های شروع دارند که باید روی صفحه نمایش داده شوند. لطفاً برای اطلاعات بیشتر به بخش پیام‌های روی صفحه مراجعه کنید. هنگامی که هر سه سرور بک‌اند (سرور A، سرور R و سرور D) در حال کار و فعال هستند، می‌توانید از هر ساختار داده‌ای که نیازهای شما را برآورده می‌کند استفاده کنید. باید پیام‌های صحیح را روی صفحه نمایش دهید تا نشان‌دهنده موفقیت این عملیات‌ها در سرور اصلی و سرورهای بک‌اند باشد که در فاز 2 و در بخش "پیام‌های روی صفحه" شرح داده خواهند شد.

فاز 1B: (20 امتیاز)

احراز هویت:

فرمت فرمان: ./client <نام‌کاربری> <رمزعبور>

در این فاز، از کاربر خواسته می‌شود تا نام‌کاربری و رمز عبور را در ترمینال وارد کند. برای احراز هویت، کاربر باید فرمان را به همان شکل که در ابتدای این بخش توضیح داده شده است اجرا کند.

این سیستم دو نوع کاربر دارد:

- **مهمان**: این نوع کاربر باید کلمه "guest" را به عنوان نام‌کاربری و رمزعبور وارد کند. پس از ارائه این اطلاعات، کاربر به عملیات جستجو دسترسی خواهد داشت، همانطور که در "پروژه" توضیح داده شده است.

- **عضو**: پس از ارائه اطلاعات کاربری به سرور اصلی، سرور اصلی درخواست احراز هویت را به سرور A (سرور احراز هویت) ارسال می‌کند. سرور A فایل member.txt را دارد و بررسی می‌کند که آیا خطی در این فایل شامل <نام‌کاربری> و <رمزعبور> ارائه شده وجود دارد یا خیر. اگر خطی شامل نام کاربری و رمز عبور مورد نظر پیدا شود، احراز هویت کامل شده تلقی شده و کاربر می‌تواند به عملیات‌های بعدی ادامه دهد.

اگر اطلاعات ارائه‌شده تحت هیچ‌یک از موارد فوق قرار نگیرد، سرور A، سرور M و کلاینت باید پیام خطای مناسبی ارائه دهند (در بخش "پیام‌های روی صفحه" توضیح داده شده) و فرآیند باید خاتمه یابد. اگر کاربر بخواهد دوباره تلاش کند، باید دوباره فرمان را اجرا کند.

برای اعضاء، این سیستم یک طرح رمزگذاری برای رمزهای عبور احراز هویت فراهم خواهد کرد. طرح رمزگذاری به شکل زیر است:

- هر کاراکتر و/یا عدد را با 3 واحد جابجا کنید.
- **کاراکتر**: به صورت دورانی از نظر حروف الفبای انگلیسی (A-Z، a-z) به‌روزرسانی شده و در صورت سرریز به ابتدای حروف برمی‌گردد.
- **عدد**: به صورت دورانی از نظر ارقام 0-9 به‌روزرسانی شده و در صورت سرریز به ابتدای ارقام بازمی‌گردد.
- طرح رمزگذاری به حروف بزرگ و کوچک حساس است.
- کاراکترهای ویژه (از جمله فضاها و نقطه اعشار) رمزگذاری یا تغییر داده نمی‌شوند.

جدول 1. مثال‌های رمزگذاری

مثال	متن اصلی	متن رمزگذاری‌شده
#1	Welcome to EE450!	Zhofrph wr HH783!
#2	199xyz@$	422abc@$
#3	0.27#&	3.50#&

محدودیت‌ها:

- رمز عبور به حروف بزرگ و کوچک حساس خواهد بود (طول 5 تا 50 کاراکتر).

مشاهده: اطلاعات رمز عبور که در فایل members.txt ذکر شده، به نسخه رمزگذاری‌شده رمز عبور اشاره دارد. از اعضا خواسته می‌شود که هنگام ارائه اطلاعات کاربری، نام‌کاربری و رمز عبور اصلی (غیر رمزگذاری‌شده) را وارد کنند. یک فهرست کوتاه از نام‌کاربری‌ها و رمزهای عبور اصلی به عنوان یک فایل اضافی برای مرجع ("original.txt") ارائه خواهد شد.

فاز 2: (60 امتیاز)

جستجو:

فرمت فرمان: `lookup <نام‌کاربری>`

این فرمان فهرستی از اسناد موجود در مخزن یک نام‌کاربری خاص را بازیابی می‌کند و از طریق سرور M به سرور R ارسال می‌شود. کلاینت‌های مهمان می‌توانند از فرمان "lookup <نام‌کاربری>" برای دسترسی به مخزن هر کاربری استفاده کنند. کلاینت‌های عضو نیز می‌توانند به فایل‌های خود دسترسی داشته باشند (با یا بدون مشخص‌کردن نام‌کاربری) یا مخزن کاربر دیگری را با ارائه نام‌کاربری مربوطه مشاهده کنند.

ارسال:

فرمت فرمان: `push <نام‌فایل>`

این فرمان باید همراه با یک نام فایل باشد؛ در غیر این صورت، پیام خطا نمایش داده خواهد شد. درخواست از طریق سرور M به سرور R ارسال می‌شود تا بررسی کند که آیا نام فایل به مخزن کاربر فعلی در سرور R مرتبط است یا خیر.

در اینجا با دو سناریو مواجه خواهیم شد:

- اگر نام فایل در سیستم وجود نداشته باشد، فایل مستقیماً در مخزن عضو ذخیره می‌شود (نام فایل و نام کاربری به "filenames.txt" در سرور R اضافه می‌شوند).

- اگر فایل از قبل وجود داشته باشد، هیچ تغییری در filenames.txt مورد نیاز نیست.

پاسخ از سرور R به سرور M و سپس به کلاینت ارسال می‌شود تا بررسی کند که آیا عضو می‌خواهد فایل موجود با همان نام را بازنویسی کند یا خیر. کاربر باید Y یا N را (حساس به حروف بزرگ و کوچک نیست) برای تصمیم‌گیری در مورد بازنویسی فایل وارد کند. پیام پاسخ کلاینت از طریق سرور M به سرور R ارسال می‌شود، و سرور R پاسخ را دریافت می‌کند.

لازم نیست که محتوای کامل فایل را در سرور R ذخیره کنید، زیرا این پروژه نسخه‌ای ساده‌شده است. تنها نام فایل (و نه کل مسیر) باید در سرور R ذخیره شود. سرور R فقط باید نام فایل‌ها را در مخزن عضو بررسی کند تا ببیند آیا فایل وجود دارد یا خیر.

استقرار (Deploy):

فرمت فرمان: `deploy`

این فرمان ابتدا سرور مخزن R را بررسی می‌کند، تمام نام فایل‌ها را از کاربر خاص از فایل filenames.txt بازیابی کرده و آن‌ها را در فایل deployed.txt که توسط سرور D نگه‌داری می‌شود، مستقر می‌کند. توجه داشته باشید که فایل deployed.txt ارائه نشده است و سرور D این فایل را زمانی که اولین استقرار رخ دهد، ایجاد می‌کند.

- اگر هیچ نام فایلی در مخزن سرور R یافت نشد، پیام خطا نمایش داده خواهد شد؛ در غیر این صورت، فایل‌ها مستقر شده و به فایل deployed.txt اضافه می‌شوند و پیام‌های مربوط به "پیام‌های روی صفحه" نمایش داده می‌شوند.

- توجه داشته باشید که برای فرمان deploy، نیازی به مشخص کردن نام فایل نداریم؛ بنابراین، تمام فایل‌های مربوط به یک کاربر خاص از سرور R بازیابی و در سرور D مستقر می‌شوند.

نیازی به ذخیره محتوای فایل در سرور D ندارید؛ فقط نام فایل‌ها کافی است. برای اهداف تست و ارزیابی، ما فقط فرمان deploy را برای هر کاربر حداکثر یک بار اجرا خواهیم کرد.

حذف (Remove):

فرمت فرمان: `remove <نام‌فایل>`

هدف این فرمان حذف فایلی است که کاربر در مخزن خود قرار داده است. با استفاده از این فرمان، نام فایل از لیست فایل‌های ذخیره‌شده در سرور مخزن (سرور R) حذف می‌شود. ورودی مربوط به کاربر احراز هویت شده و نام فایل برای حذف، از فایل filenames.txt حذف خواهد شد. اگر نام فایل در سرور R یافت نشد، در نظر بگیرید که پیام خطایی مطابق با بخش "پیام‌های روی صفحه" نوشته شود.

توجه: اگر کاربر یک فرمان نامعتبر وارد کند، سیستم از کاربر می‌خواهد که فرمان را دوباره وارد کند بعد از نمایش پیام خطا. برای مثال، اگر عضو `push` را بدون مشخص کردن نام فایل وارد کند، ابتدا پیام

"filename is not specified."

نمایش داده شود. سپس از کاربر خواسته می‌شود که فرمان را دوباره وارد کند و پیام زیر نمایش داده شود:

"Please enter the command: <lookup <username>> , <push <filename> > , <remove <filename> > , <deploy> , <log>."

امتیاز اضافی (10 امتیاز):

فرمت فرمان: `log`

در این بخش، شما یک عملیات دیگر به نام log را پیاده‌سازی خواهید کرد. سرور اصلی می‌تواند فرمان log را دریافت و اجرا کند تا تاریخچه اقدامات انجام‌شده توسط یک عضو را نمایش دهد. به عنوان مثال، کاربر Yvette وارد حساب کاربری خود می‌شود و احراز هویت را انجام می‌دهد. او می‌خواهد تاریخچه عملیات خود را مشاهده کند، بنابراین فرمان log را اجرا می‌کند، و سپس

سرور اصلی نتیجه را نمایش می‌دهد و تاریخچه را به ترتیب زمانی صعودی (از قدیمی‌ترین به جدیدترین) روی ترمینال کلاینت نمایش می‌دهد. سرور اصلی باید یک رکورد پایدار از تمام عملیات‌ها را نگه‌داری کند تا این داده‌ها حتی در صورت قطع و اجرای مجدد برنامه توسط عضو، قابل دسترسی باشد. پیام‌های دقیق روی صفحه در زیر نشان داده شده‌اند. برای کسب امتیاز اضافی، لطفاً در فایل readme.txt اشاره کنید که این بخش امتیاز اضافی را پیاده‌سازی کرده‌اید.

تخصیص شماره پورت مورد نیاز

پورت‌هایی که باید توسط کلاینت‌ها و سرورها برای این تمرین استفاده شوند، در جدول زیر مشخص شده‌اند:

توجه: xxx = 649 است. برای مثال، باید از پورت 21000+649 = 21649 برای سرور Backend (A) استفاده کنید. این عدد **21000649** نخواهد بود.

فرآیند	پورت‌های پویا	پورت‌های ایستا
serverA	-	1 UDP, 21000+xxx
serverR	-	1 UDP, 22000+xxx
serverD	-	1 UDP, 23000+xxx
serverM	-	1 UDP, 24000+xxx
	1 TCP 25000 ,با کلاینت+xxx	
client	2 TCPs	-

پیام‌های روی صفحه

جدول 3. پیام‌های روی صفحه برای سرور A (سرور احراز هویت)

Event	On Screen Message
Booting Up (Only while starting):	“Server A is up and running using UDP on port <port number>”.
Upon Receiving the auth request:	“ServerA received username <USERNAME> and password ****** ”
If auth request is member user:	“Member <USERNAME> has been authenticated”
If either username or password incorrect:	“The username <USERNAME> or password ****** is incorrect”

جدول 4. پیام‌های روی صفحه برای سرور R (سرور مخزن)

Event	On Screen Message
Booting Up (Only while starting):	“Server R is up and running using UDP on port <port number>.”
lookup request	
Upon receiving a lookup request from the main server	“Server R has received a lookup request from the main server.”
After returning the list of documents in the repository	“Server R has finished sending the response to the main server.”
push request	
Upon receiving a push request from the main server	“Server R has received a push request from the main server.”
After checking the filename and finding a duplicate within the member's repository	“<filename> exists in <username>'s repository; requesting overwrite confirmation."
Upon receiving a Y response to the overwrite confirmation	“User requested overwrite; overwrite successful.”
Upon receiving a N response to the overwrite confirmation	“Overwrite denied”
After checking the filename and finding no duplicates, the file is stored directly.	“<filename> uploaded successfully.”
remove request	
Upon receiving remove request	“Server R has received a remove request from the main server.”
deployment request	
Upon receiving a deploy request from the main server	“Server R has received a deploy request from the main server.”
After returning the list of documents in the repository	“Server R has finished sending the response to the main server.”

جدول 5. پیام‌های روی صفحه برای سرور D (سرور استقرار)

Event	On Screen Message
Booting Up (Only while starting):	"Server D is up and running using UDP on port <port number>."
deployment request	
Upon receiving a deploy request from the main server	"Server D has received a deploy request from the main server."
Upon completing the deploy request from the main server	"Server D has deployed the user <username>'s repository."

جدول 6. پیام‌های روی صفحه برای سرور M (سرور اصلی)

Event	On Screen Message
Booting Up (Only while starting):	"Server M is up and running using UDP on port <port number>."
Upon Receiving the authentication request from member	"Server M has received username <USERNAME> and password ****."
Upon sending the authentication request to server A	"Server M has sent authentication request to Server A"
Upon receiving the authentication response from server A	"The main server has received the response from server A using UDP over <Main Server UDP port number>"
Upon sending the authentication response to client	"The main server has sent the response from server A to client using TCP over port <main server TCP port number>."
lookup request	
Upon receiving a lookup request from a guest	The main server has received a lookup request from Guest to lookup <username>'s repository using TCP over port <main server TCP port number>.
Upon receiving a lookup request from a member	The main server has received a lookup request from <member's username> to lookup <username>'s repository using TCP over port <main server TCP port number>.
After forwarding the lookup request to server R	The main server has sent the lookup request to server R.
After receiving the response from server R	The main server has received the response from server R using UDP over <Main Server UDP port number>
After forwarding the response to the client	The main server has sent the response to the client.
push request	

Upon receiving push request from a client	The main server has received a push request from <username>, using TCP over port <main server TCP port number>.
Upon receiving overwrite confirmation response from client	The main server has received the overwrite confirmation response from <username> using TCP over port <main server TCP port number>
Upon receiving a push request response from Server R (excluding overwrite confirmation requests)	The main server has received the response from server R using UDP over <Main Server UDP port number>
Upon receiving response from server R asking for overwrite confirmation	The main server has received the response from server R using UDP over <Main Server UDP port number>, asking for overwrite confirmation
After forwarding the response to the client (excluding overwrite confirmation requests)	The main server has sent the response to the client.
After forwarding the push request to server R (excluding overwrite confirmation response)	The main server has sent the push request to server R.
After forwarding the overwrite confirmation request to the client	The main server has sent the overwrite confirmation request to the client.
After forwarding the overwrite confirmation response to server R	The main server has sent the overwrite confirmation response to server R.
remove request	
Upon receiving a remove request from member user	"The main server has received a remove request from member <username> TCP over port <main server TCP port number>."
Upon receiving a remove request done by server R	"The main server has received confirmation of the remove request done by the server R"
deployment request	
Upon receiving a deploy request from the Member User	The main server has received a deploy request from member <username> TCP over port <main server TCP port number>.
After forwarding the lookup request to server R	The main server has sent the lookup request to server R.
After receiving the response to lookup request to server R	The main server received the lookup response from server R.
After sending the server R response to request deploy to server D	The main server has sent the deploy request to server D.
After receiving a confirmation response from server D	The user <username>'s repository has been deployed at server D.
log request	
Upon receiving a log request	The main server has received a log request from member <username> TCP

from the Member User	over port <main server TCP port number>.
After sending the response to client	The main server has sent the log response to the client.

جدول 7. پیام‌های روی صفحه برای کلاینت (عضو)

Event	On Screen Message
Booting Up	"The client is up and running."
If client is a guest and wrote correct credentials	"You have been granted guest access."
fail login	"The credentials are incorrect. Please try again."
If client is a member and wrote correct credentials	"You have been granted member access"
Asking for commands (please note that the lookup command accepts <username>, and so on so forth)	"Please enter the command: <lookup <username>> <push <filename>> <remove <filename>> <deploy> <log>" *Menu can be printed in horizontal or vertical format.
Asking for commands (if you implement extra credit)	"Please enter the command: <lookup <username>> , <push <filename> > , <remove <filename> > , <deploy> , <log>."
lookup request	
No username is specified	"Username is not specified. Will lookup <member's username>."
Upon sending a lookup request	"<username> sent a lookup request to the main server."
After receiving the response from the main server (the username exists)	"The client received the response from the main server using TCP over port <client port number>. <filename1.js> <filename2.html> … ----Start a new request----"
After receiving the response from the main server (the username	"The client received the response from the main server using TCP over port <client port number>.

does not exist)	<username> does not exist. Please try again. ——Start a new request——"
After receiving the response from the main server (the repository is empty)	"The client received the response from the main server using TCP over port <client port number>. Empty repository. ——Start a new request——"
push request	
No filename is specified	"Error: Filename is required. Please specify a filename to push."
Filename is invalid or unable to read	"Error: Invalid file: <filename> ——Start a new request——"
Filename is exist in the member's repository and server R request for confirmation	"<filename> exists in <username>'s repository, do you want to overwrite (Y/N)? "
Upon receiving a successful response from Server R	"<filename> pushed successfully"
Upon receiving a fail response from Server R	"<filename> was not pushed successfully."
remove request	
Upon sending a remove request	"<username> sent a remove request to the main server."
After receiving the confirmation from main server	"The remove request was successful."
deployment request	
Upon sending a deploy request	<username> sent a lookup request to the main server.
After receiving the response from the main server (the username exists)	The client received the response from the main server using TCP over port <client port number>. The following files in his/her repository have been deployed. <filename1.js> <filename2.html> … ——Start a new request——
(extra credit) log request	
Upon sending a log request	<username> sent a log request to the main server.

After receiving the response from the main server	The client received the response from the main server using TCP over port <client port number>. 1. push <filename> 2. push <filename> 3. lookup <username> 4. remove <filename> 5. deploy 6. log ... ----Start a new request----

جدول 8. پیام‌های روی صفحه برای کلاینت (مهمان)

Event	On Screen Message
Booting Up	"The client is up and running."
Asking for commands (please note that the lookup command needs <username>)	Please enter the command: <lookup <username>>
Invalid command (guest could only use the lookup command)	Guests can only use the lookup command
lookup request	
No username is specified	"Error: Username is required. Please specify a username to lookup. ——Start a new request——"
Upon sending a lookup request	"Guest sent a lookup request to the main server."
After receiving the response from the main server (the username exists)	"The client received the response from the main server using TCP over port <client port number>. <filename1.js> <filename2.html> … ——Start a new request——"
After receiving the response from the main server (the username does not exist)	"The client received the response from the main server using TCP over port <client port number>. <username> does not exist. Please try again. ——Start a new request——"
After receiving the response from the main server (the repository is empty)	"The client received the response from the main server using TCP over port <client port number>. Empty repository. ——Start a new request——"

فرضیات:

1. شما باید فرآیندها را به این ترتیب راه‌اندازی کنید: serverM، سپس serverA، serverR، serverD، و در نهایت client. اگر نیاز به فایل‌های کد بیشتری نسبت به آنچه در اینجا ذکر شده است دارید، لطفاً از نام‌های معنی‌دار و تمام حروف کوچک استفاده کنید و تمامی آن‌ها را در فایل README خود ذکر کنید.

2. شما مجاز به استفاده از بخش‌هایی از کد موجود در راهنمای برنامه‌نویسی سوکت بیج (Beej's guide to network programming) هستید. با این حال، لازم است قسمت‌های کپی شده را در کد خود علامت بزنید.

3. هنگام اجرای کد خود، اگر با پیغام "port already in use" یا "address already in use" مواجه شدید، ابتدا بررسی کنید که آیا پروسه‌های زامبی دارید یا خیر. اگر چنین پروسه‌هایی ندارید و همچنان این پیغام را دریافت می‌کنید، سعی کنید شماره پورت UDP یا TCP استاتیکی که این پیغام خطا را دارد تغییر دهید (تمامی شماره پورت‌های زیر 1024 رزرو شده‌اند و نباید استفاده شوند). اگر مجبور به تغییر شماره پورت هستید، لطفاً این موضوع را در فایل README خود ذکر کرده و دلایل آن را توضیح دهید.

4. ممکن است هنگام آزمایش کدهای خود پروسه‌های زامبی ایجاد کنید، لطفاً مطمئن شوید که هر بار قبل از اجرای کد خود آن‌ها را از بین می‌برید. برای مشاهده لیستی از تمام پروسه‌های زامبی، این فرمان را اجرا کنید: `ps –aux | grep ee450`. پروسه‌های زامبی و شماره پروسه آن‌ها را شناسایی کرده و با تایپ این فرمان در خط فرمان آن‌ها را از بین ببرید:

`kill -9 processNumber`.

الزامات:

1. شماره پورت‌های TCP یا UDP که به صورت پویا به‌دست می‌آیند را به صورت ثابت کدنویسی نکنید. به جدول 1 برای مشاهده پورت‌های تعریف شده به صورت استاتیک و پورت‌هایی که به صورت پویا تخصیص داده می‌شوند، مراجعه کنید. از تابع

getsockname()

برای بازیابی شماره پورت محلی استفاده کنید هر جا که پورت‌ها به صورت پویا تخصیص داده شده‌اند، به صورت زیر:

```
/*Retrieve the locally-bound name of the specified socket and
store it in the sockaddr structure*/
Getsock_check=getsockname(TCP_Connect_Sock,(struct sockaddr
*)&my_addr, (socklen_t *)&addrlen);
//Error checking
if (getsock_check== -1) {
perror("getsockname");
```

```
exit(1);
}
```

2. نام میزبان (host name) باید به صورت ثابت به عنوان localhost (127.0.0.1) در تمامی کدها قرار داده شود.

3. کلاینت‌ها و سرورهای بک‌اند باید در حالت اجرا باقی بمانند و منتظر درخواست دیگری باشند تا زمانی که فرمان Ctrl+C آن‌ها را خاتمه دهد. اگر آن‌ها قبل از این خاتمه یابند، امتیازاتی از دست خواهید داد.

4. تمامی قواعد نامگذاری و پیام‌های روی صفحه باید مطابق با قوانین قبلاً ذکر شده باشند.

5. اجازه ندارید هیچ پارامتر، مقدار، رشته یا کاراکتری را به عنوان آرگومان خط فرمان ارسال کنید، مگر هنگام اجرای کلاینت در فاز 1.

6. تمام پیام‌های روی صفحه باید دقیقاً مطابق با توضیحات پروژه باشند. نباید پیام‌های اضافی به آن‌ها اضافه کنید. اگر نیاز به اضافه کردن پیام‌های اضافی برای دیباگینگ دارید، باید تمامی پیام‌های اضافی را قبل از ارسال پروژه کامنت کنید.

7. لطفاً به یاد داشته باشید که پس از اتمام استفاده از سوکت، آن را ببندید و اتصال را خاتمه دهید.

پلتفرم برنامه‌نویسی و محیط:

1. تمامی کدهای ارسالی شما باید به خوبی بر روی virtual machine Ubuntu که در اختیار قرار داده شده کار کنند.

2. تمامی ارزیابی‌ها فقط بر روی Ubuntu ارائه شده انجام خواهد شد. مصحح ها هیچ به‌روزرسانی یا تغییری در ماشین مجازی انجام نخواهند داد. این مسئولیت شماست که مطمئن شوید کدتان به خوبی روی Ubuntu ارائه شده کار می‌کند.

3. ارسال شما باید دارای یک Makefile باشد. لطفاً الزامات را در بخش "قوانین ارسال" که در ادامه آمده است رعایت کنید.

زبان‌های برنامه‌نویسی و کامپایلرها:

شما باید تنها از زبان‌های C/C++ بر روی سیستم عامل UNIX و دستورات و توابع برنامه‌نویسی سوکت UNIX استفاده کنید. در اینجا منابع مربوط به راهنمای Beej برای برنامه‌نویسی C و شبکه (برنامه‌نویسی سوکت) آورده شده است.

شما می‌توانید از ویرایشگر متنی یونیکس مانند emacs برای نوشتن کد خود استفاده کنید و سپس از کامپایلرهایی مانند g++ (برای C++) و gcc (برای C) که به طور پیش‌فرض در اوبونتو نصب شده‌اند، برای کامپایل کد خود استفاده کنید. باید از دستورات و سوئیچ‌های زیر برای کامپایل فایل‌های `yourfile.c` یا `yourfile.cpp` استفاده کنید. این دستورات یک فایل اجرایی با نام "yourfileoutput" تولید خواهند کرد:

```

**gcc -o yourfileoutput yourfile.c**

**g++ -o yourfileoutput yourfile.cpp**

لطفاً فراموش نکنید که قوانین نامگذاری اجباری که قبلاً ذکر شده‌اند را رعایت کنید!

همچنین در داخل کد خود باید این فایل‌های هدر را علاوه بر هر فایل هدر دیگری که ممکن است نیاز داشته باشید، وارد کنید:

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <sys/wait.h>
```

## قوانین ارسال:

1. همراه با فایل‌های کد خود، یک فایل README و یک فایل Makefile قرار دهید. در فایل README موارد زیر را بنویسید:
   - **الف.** نام کامل شما طبق لیست کلاسی
   - **ب.** شماره دانشجویی شما
   - **پ.** آنچه در تکلیف انجام داده‌اید. اگر بخش اختیاری را تکمیل کرده‌اید، آن را قید کنید (اگر ذکر نشده باشد، در نظر گرفته نمی‌شود).
```

- **ت.** فایل‌های کد شما چه هستند و هر کدام از آنها چه کاری انجام می‌دهند. (لطفاً توضیحات پروژه را تکرار نکنید، فقط نام فایل‌های کد خود را بنویسید و به طور خلاصه ذکر کنید که چه کاری انجام می‌دهند).
- **ث.** فرمت تمام پیام‌های مبادله شده.
- **ج.** هر گونه ویژگی خاص پروژه شما. توضیح دهید که پروژه تحت چه شرایطی شکست می‌خورد، اگر چنین باشد.
- **چ.** کد بازاستفاده شده: آیا از جایی کدی برای پروژه خود استفاده کرده‌اید؟ اگر نه، بگویید نه. اگر بله، بگویید که کدام توابع و از کجا. (همچنین در کد منبع خود با یک کامنت آن را مشخص کنید).

توجه: ارسال‌هایی که فاقد فایل README و Makefile باشند، نمره‌دهی نخواهند شد.

جدول ۹. فرآیند اجرای تست

`make all`	Compiles **all** your files and creates executables
`./serverM`	Runs Main server
`./serverA`	Runs backend server A
`./serverR`	Runs backend server R
`./serverD`	Runs backend server D
`./client <USERNAME> <PASSWORD>`	Runs the client

2. ابتدا، برای نمره دهی تمام کدها را با استفاده از دستور `make all` کامپایل خواهند کرد. سپس، آن‌ها ۶ پنجره ترمینال مختلف باز می‌کنند. در ۴ ترمینال، سرورها M، A، R و D را با دستورات

`./serverM`، `./serverA`، `./serverR` و `./serverD`

اجرا خواهند کرد. در ۲ ترمینال دیگر، دو کلاینت (یکی به عنوان عضو و دیگری به عنوان مهمان) را با استفاده از

`./client`

اجرا می‌کنند. به خاطر داشته باشید که سرورها پس از شروع باید همیشه فعال باشند. کلاینت‌ها می‌توانند دوباره و دوباره با دستورات و مقادیر ورودی مختلف متصل شوند. خروجی‌ها را برای مقادیر ورودی متعدد بررسی خواهند شد. ترمینال‌ها باید پیام‌های ذکر شده در بخش پیام‌های روی صفحه را نمایش دهند.

تمام فایل‌های خود، شامل فایل README، را به یک "فایل فشرده tar" تبدیل کرده. دستورالعمل‌ها به صورت زیر است:

- روی ماشین مجازی خود، به دایرکتوری‌ای که تمام فایل‌های پروژه شما در آن قرار دارند بروید. تمام فایل‌های اجرایی و سایر فایل‌های غیرضروری را حذف کنید. تنها فایل‌های کد منبع مورد نیاز، Makefile و فایل README را وارد کنید.