Mini-projet Programmation Système et Réseaux sous Unix

1. Contexte du Sujet:

Le but essentiel du mini-projet consiste à élaborer et mettre en œuvre deux applications client/serveur en langage C, en exploitant l'environnement Unix (Linux) et en utilisant les sockets comme protocole de communication. Le projet se divise en deux parties distinctes, à savoir le développement d'applications fonctionnant sur les protocoles TCP et UDP, chacune apportant ses propres spécificités et caractéristiques en termes de communication et de fonctionnalités.

1. Partie 1: Communication UDP
2. Objectif:

Dans cette première partie du projet, nous avons mis en œuvre la communication entre un client et un serveur en utilisant le protocole de datagramme utilisateur (UDP). L'objectif était d'envoyer un entier "n" du client au serveur, qui renverrait ensuite un tableau de "n" nombres triés au client.

1. Architecture et Communication

Client UDP:

* Le client UDP prend en paramètres l'adresse IP du serveur et le port sur lequel le serveur écoute.
* Il crée une socket UDP.
* Le client génère un nombre aléatoire "n" et l'envoie au serveur via la fonction `sendto`.
* Il attend ensuite de recevoir le tableau trié du serveur via la fonction `recvfrom`.
* Enfin, le tableau trié est affiché à l'utilisateur.

Serveur UDP :

* Le serveur UDP crée une socket et se lie à un port spécifié.
* Il écoute en permanence pour recevoir des requêtes des clients via la fonction `recvfrom`.
* Lorsqu'il reçoit un nombre "n" du client, le serveur génère "n" nombres aléatoires, les trie, puis renvoie le tableau trié au client via la fonction `sendto`.

1. Partie 2: Communication TCP
2. Objectif:

La deuxième partie du projet a implémenté une communication client-serveur via le protocole de contrôle de transmission (TCP), avec une couche d'authentification. L'objectif était de fournir diverses fonctionnalités telles que l'obtention de la date et de l'heure, la récupération de la liste des fichiers dans un répertoire, l'accès au contenu d'un fichier spécifique, et le calcul du temps écoulé depuis la connexion.

1. Architecture et Communication

Client TCP:

* Le client TCP prend en paramètres l'adresse IP du serveur et le port sur lequel le serveur est en écoute.
* L'utilisateur fournit un nom d'utilisateur et un mot de passe, qui sont envoyés au serveur pour authentification. La fonction authenticate gère ce processus. Dans cette implémentation, les identifiants prédéfinis sont "user" pour le nom d'utilisateur et "password" pour le mot de passe. La réussite de l'authentification dépend de ces valeurs.
* En cas de succès de l'authentification, le client peut choisir parmi différentes options du menu pour accéder aux services du serveur. La fonction displayMenuAndGetChoice facilite la sélection des services.
* Les résultats de chaque service sont affichés sur le client. La fonction receiveAndDisplayResult assure cette étape.

Serveur TCP :

* Le serveur TCP est en écoute sur un port spécifié.
* À chaque connexion d'un client, un processus enfant est créé pour gérer la communication avec ce client. La fonction handleClient gère ce processus.
* Le serveur effectue une authentification en demandant un nom d'utilisateur et un mot de passe au client. Si les informations sont correctes, l'accès est autorisé. La fonction authenticate gère cette étape. Dans cette implémentation, les identifiants valides sont prédéfinis comme "user" pour le nom d'utilisateur et "password" pour le mot de passe.
* Un menu d'options est proposé au client, qui peut choisir parmi différents services. La fonction handleServiceSelection gère la sélection des services.
* Les services incluent l'obtention de la date et de l'heure, la récupération de la liste des fichiers, l'accès au contenu d'un fichier, et le calcul du temps écoulé depuis la connexion. Les fonctions telles que sendDateTime, sendFileList, sendFileContent, et sendElapsedTime assurent l'exécution de ces services.
* Les résultats de chaque service sont renvoyés au client via la fonction send du côté serveur. La connexion reste active jusqu'à ce que l'utilisateur choisisse de se déconnecter, offrant une expérience continue où le menu est réaffiché après chaque service rendu.

1. Conclusion

En conclusion, ce mini-projet nous a offert une expérience approfondie dans le développement d'applications client/serveur en utilisant les protocoles TCP et UDP. L'implémentation réussie de la gestion d'accès multiple en TCP renforce la fiabilité de notre solution. Ces réalisations enrichissent notre compréhension des concepts de communication réseau et de sécurité, tout en mettant en évidence les différences entre les protocoles TCP et UDP dans un environnement Unix.