Lakshya Selvakumar 12202210

Rapport

Projet Wikicomptines

18/05/2024

Dans le cadre de ce projet, j’ai implémenté des fonctions afin de permettre à des utilisateurs de partager des comptines sans sortir de leur terminal, en facilitant une communication fluide et sans interruption. Ainsi, ce rapport détaille les modifications spécifiques apportées au protocole, les nouvelles fonctionnalités implémentées, et les divers problèmes rencontrés au cours du processus de développement.

Pour offrir une vision claire et complète du travail accompli, ce rapport présente également les efforts déployés pour résoudre les obstacles techniques rencontrés. De plus, il mentionne les ressources externes utilisées pour guider et enrichir mon travail.

En somme, ce rapport constitue une synthèse complète et réfléchie des efforts fournis. Il offre une vue d’ensemble des progrès réalisés et des défis surmontés.

Description détaillée des ajouts:

Au départ, chaque ligne de notre comptine pouvait contenir au maximum 256 octets, y compris le caractère de fin de ligne `'\n'`. Par la suite, le format a été modifié pour accepter des lignes de plus grande taille. Pour gérer ce changement, j'ai défini tous mes tampons mémoire (buffers) à l'aide de la macro `#define BUFSIZE 1024`. Ainsi, la taille des buffers est passée de 256 octets à 1024 octets.

J'ai ensuite modifié le code pour que le serveur soit multithreadé, permettant ainsi de servir plusieurs clients simultanément. Pour cela, dans ma fonction `main()`, j'ai initialisé les threads ainsi que les informations nécessaires dans une structure spécialement conçue à cet effet. Ces informations sont passées en argument à la fonction `void *f(void *args)`, qui se charge d'envoyer les comptines demandées par chaque client.

Enfin, j'ai modifié la fonction main du fichier `wcp_clt` en ajoutant une boucle infinie, permettant aux clients de demander plusieurs

comptines successivement. Si le client saisit une commande invalide ou un numéro hors intervalle, la requête est terminée.

Malheureusement, par manque de temps, je n'ai pas pu implémenter certaines fonctionnalités, comme le téléversement de nouvelles comptines par un client.

J'avais commencé à travailler sur cette fonctionnalité, mais je n'ai pas pu la terminer. Voici quelques idées que j'ai eues pour implémenter le téléversement de nouvelles comptines :

1. Côté serveur : J'ai ajouté une fonction pour construire un chemin complet en utilisant le paramètre dirname, qui ressemblera à `dirname/base_name`.
2. Cette fonction initialise une structure de type comptine pour la nouvelle comptine à ajouter.

Cependant, j'ai commis une erreur due à une mauvaise compréhension de la tâche. Après l'initialisation, la fonction crée simplement un fichier dans le chemin créé, mais cela ne correspond pas à un téléversement de fichier.

Ainsi, bien que la création du chemin et l'initialisation de la structure soient correctes, la fonction ne gère actuellement que la création de fichier et non le téléversement complet des données de la comptine.

```
struct comptine *ajouter_comptine(const char *dirname, const char *base_name) {
    char chemin[strlen(dirname) + strlen(base_name) + 2];
    strcpy(chemin, dirname);
    strcat(chemin, "/");
    strcat(chemin, base_name);

    struct comptine *ctn = init_cpt_depuis_fichier(dirname, base_name);

      /* On ajoute le fichier dans le repertoire dirname, open échoue si le fichier existe déjà */
    int fd = open(chemin, O_CREAT | O_EXCL, 0644);
    if (fd < 0) {
        perror("open");
        exit(99);
    }

    return ctn;
}
```

Ensuite, dans la fonction `f` qui s'occupe d'envoyer la comptine au client j'ai ajouté un `switch` qui permet de choisir entre lire une comptine et en ajouter une.

```
switch(choix_clt) {
    case 1:
        envoyer_liste(clt->socket, c);

        uint16_t m = recevoir_num_comptine(clt->socket);
        envoyer_comptine(clt->socket, clt->dirname, c, m);
        break;
    case 2:
        liberer_catalogue(c);

        printf("Saisissez le nom de la comptine que vous souhaitez ajouter: \n");
        fgets(buf, BUFSIZE, stdin);
        ajouter_comptine(clt->dirname, buf);

        creer_catalogue(clt->dirname);
        envoyer_liste(clt->socket, c);

        uint16_t n = recevoir_num_comptine(clt->socket);
        envoyer_comptine(clt->socket, clt->dirname, c, n);
        break;
}
```

Cependant, cette implémentation est incorrecte. Par exemple, le `printf` ne sert à rien puisqu'il s'affiche sur le terminal du serveur alors que nous souhaitons l'afficher chez le client. Par conséquent, cette méthode ne fonctionne pas.

Je pense qu'il aurait fallu créer deux fonctions distinctes: une pour envoyer la requête du client et une autre pour réceptionner cette requête côté serveur. Ensuite, il aurait fallu modifier le main dans `wcp_clt` en fonction du choix du client et ajuster la fonction `f` pour réagir en conséquence.

Ces fonctions n'apparaissent pas dans la version finale. Néanmoins, je souhaitais tout de même montrer le début de mon implémentation.

Référence externes:

Pour me guider dans mon travail, j'ai consulté des ressources externes, principalement des sites web, pour des rappels ponctuels. Le

cours magistral 8 m'a également beaucoup aidé à mieux comprendre le fonctionnement des sockets.
Ces ressources m'ont permis d'approfondir mes connaissances sur certaines fonctions que nous connaissions déjà, comme les arguments de la fonction open, par exemple.

Vous trouverez ci-dessous un tableau listant les ressources qui m'ont aidé.

Rappel sur les fonctions de bases	Koor.fr
Rappel sur les appels systèmes	Manpagesfr.free.fr
Utilisation du scanf avec un uint16_t	Stackoverflow
Conversion de données	Manpages.ubuntu

En conclusion, ce projet m'a permis d'approfondir mes connaissances en programmation réseau. J'ai appris à mettre en place un serveur capable de gérer plusieurs clients simultanément, à utiliser les sockets de manière efficace, et à manipuler des fichiers et des répertoires pour gérer les comptines. Bien que certaines fonctionnalités, comme le téléversement de nouvelles comptines par les clients, n'aient pas été entièrement implémentées, ce travail m'a offert une bonne occasion de comprendre les défis et les complexités liés à la conception et à la mise en œuvre de telles fonctionnalités.