Rapport du projet de Sytèmes et Réseaux

MEHAMLI Samy - 12213639

May 18, 2024

1 Introduction

Ce document explique les nouvelles fonctionnalités ajoutées à l'application serveur-client pour la gestion des comptines. Les ajouts incluent un menu interactif pour le client, la possibilité d'afficher le catalogue de comptines, d'afficher une comptine spécifique et d'ajouter une nouvelle comptine au catalogue. Chaque fonctionnalité est accompagnée d'une description du code utilisé.

2 Menu interactif

Un menu interactif a été ajouté pour améliorer l'expérience utilisateur. Ce menu permet au client de choisir parmi les options suivantes :

- Quitter le menu et arrêter le processus
- Afficher le catalogue des comptines
- Afficher une comptine spécifique
- Ajouter une comptine au catalogue

2.1 Coté serveur

Le code pour envoyer le menu au client est le suivant :

```
1  void envoyer_menu(int fd, struct catalogue *c) {
2      char *menu = "\t\t MENU\n 0. Quitter le menu et arreter le processus\n 1.
           Afficher catalogue comptines.\n 2. Afficher une comptines.\n 3. Ajouter
           comptines au catalogue.\nChoisisez une proposition  : ";
3      write(fd, menu, strlen(menu) + 1);
4
5      uint16_t choix = recevoir_num_comptine(fd);
6      switch (choix) {
7          case 0:
8              return;
9          case 1:
10             envoyer_liste(fd, c);
11             envoyer_menu(fd, c);
12             break;
13         case 2:
14             envoyer_liste(fd, c);
15             uint16_t ic = recevoir_num_comptine(fd);
16             envoyer_comptine(fd, "comptines", c, ic);
17             break;
18         case 3:
19             ajouter_comptine(fd, c, "comptines");
20             envoyer_menu(fd, c);
21             break;
22         default:
23             break;
24     }
25 }
```

La fonction `envoyer_menu` a pour rôle d'envoyer un menu interactif au client, de recevoir une réponse du client, et de traiter cette réponse en appelant les fonctions appropriées selon le choix effectué.

1. **Envoi du menu** :
   - Le menu interactif est défini sous forme de chaîne de caractères et est envoyé au client via le descripteur de fichier `fd` en utilisant la fonction `write`.
   - Ce menu propose quatre options : quitter le menu, afficher le catalogue des comptines, afficher une comptine spécifique, ou ajouter une comptine au catalogue.
2. **Réception du choix du client** :
   - La fonction `recevoir_num_comptine` est appelée pour recevoir le choix du client (un numéro correspondant à une option du menu).
3. **Traitement du choix** :
   - La structure `switch` est utilisée pour traiter le choix du client :
     - **Case 0** : Si le client choisit 0, la fonction retourne simplement, ce qui termine le processus actuel.
     - **Case 1** : Si le client choisit 1, la fonction `envoyer_liste` est appelée pour envoyer le catalogue des comptines au client. Ensuite, la fonction `envoyer_menu` est appelée à nouveau pour afficher le menu.
     - **Case 2** : Si le client choisit 2, le catalogue des comptines est d'abord envoyé au client pour qu'il puisse voir les comptines disponibles. Ensuite, le serveur reçoit le numéro de la comptine spécifique que le client souhaite afficher et utilise la fonction `envoyer_comptine` pour envoyer cette comptine.
     - **Case 3** : Si le client choisit 3, la fonction `ajouter_comptine` est appelée pour permettre au client d'ajouter une nouvelle comptine au catalogue. Ensuite, le menu est envoyé à nouveau pour permettre au client de faire un nouveau choix.
     - **Default** : Si une option non valide est reçue, aucune action n'est effectuée.

En résumé, cette fonction envoie un menu interactif au client, attend le choix du client, et exécute les actions appropriées en fonction de ce choix, offrant ainsi une interaction fluide et structurée entre le client et le serveur.

2.2 Afficher le catalogue des comptines :

lorque le client envoie 1 comme choix, le serveur fait appel a la fonction envoyer_liste:

```
1 void envoyer_liste(int fd, struct catalogue *c){
2     int i;
3     for (i = 0; i < c->nb ; i++) {
4         dprintf(fd, "%6d %s", i, c->tab[i]->titre);
5     }
6     dprintf(fd, "\n");
7 }
```

Et juste aprés, le serveur renvoie le menu, car logiquement on veut voir le catalogue pour effectuer une opération dessus.

2.2.1 Afficher une comptine

Si le client choisit 1, la fonction `envoyer_liste` est appelée pour envoyer le catalogue des comptines au client. Ensuite, la fonction `envoyer_menu` est appelée à nouveau pour afficher le menu.

```c
void envoyer_comptine(int fd, const char *dirname, struct catalogue *c, uint16_t ic
    ) {
    char *path = malloc(strlen(dirname) + strlen(c->tab[ic]->nom_fichier) + 2);
    strcpy(path, dirname);
    strcat(path, "/");
    strcat(path, c->tab[ic]->nom_fichier);

    int fout = open(path, O_RDONLY);
    char buffer[BUFSIZ];
    int bytes;
    while((bytes = read(fout, buffer, BUFSIZ)) != 0){
        write(fd, buffer, bytes);
    }
    write(fd, "\n\n", 2);
    close(fout);
    free(path);
}
```

2.3 Ajout d'une comptine au catalogue

La fonction `ajouter_comptine` permet d'ajouter une nouvelle comptine au catalogue à partir des données envoyées par le client via le descripteur de fichier `fd`. Voici le déroulement de la fonction :

1. **Réception du nom de fichier** :
   - La fonction `read_until_nl` est utilisée pour recevoir le nom de fichier de la nouvelle comptine envoyé par le client via la socket. Ce nom est stocké dans le tableau `nom_fichier`.
2. **Création du chemin complet du fichier** :
   - Le chemin complet du nouveau fichier est construit en concaténant le nom du répertoire (`dir_name`) et le nom du fichier (`nom_fichier`). Cela garantit l'unicité du chemin.
3. **Création du nouveau fichier** :
   - Un nouveau fichier est créé en utilisant la fonction `open`, avec les indicateurs `O_WRONLY` pour écriture, `O_CREAT` pour création et `O_EXCL` pour garantir l'unicité du fichier. Le fichier est ouvert avec les permissions `0644`.
4. **Écriture du titre de la comptine** :
   - Le titre de la comptine est reçu du client via la socket et écrit comme première ligne dans le fichier nouvellement créé.
5. **Lecture et écriture du contenu de la comptine** :
   - Le contenu de la comptine est lu du client via la socket jusqu'à ce que deux sauts de ligne consécutifs soient rencontrés.
   - Ce contenu est ensuite écrit dans le fichier nouvellement créé.
6. **Fermeture du fichier** :
   - Une fois que le contenu de la comptine est entièrement écrit dans le fichier, celui-ci est fermé en utilisant la fonction `close`.
7. **Mise à jour du catalogue** :
   - Le catalogue existant est libéré en appelant la fonction `liberer_catalogue`.
   - Ensuite, un nouveau catalogue est créé à partir du répertoire (`dir_name`) en utilisant la fonction `creer_catalogue`. Cela assure que le catalogue est mis à jour avec la nouvelle comptine ajoutée.

```c
void ajouter_comptine(int fd, struct catalogue *c, char *dir_name) {
    char buffer[BUFSIZ];
    char nom_fichier[256];
    read_until_nl(fd, nom_fichier);
    // Creation du chemin complet du fichier
    char path[strlen(nom_fichier) + strlen(dir_name) + 2];
    strcpy(path, dir_name);
    strcat(path, "/");
    strcat(path, nom_fichier);

    int nv_fd = open(path, O_WRONLY | O_CREAT | O_EXCL, 0644);
    if (nv_fd == -1) {
        if (errno == EEXIST) {
            fprintf(stderr, "Erreur: le fichier '%s' existe déjà.\n", nom_fichier);
        } else {
            perror("Erreur lors de la création du fichier");
        }
        return;
    }

    // Demander le titre de la comptine au client
    read_until_nl(fd, buffer);
    write(nv_fd, buffer, strlen(buffer));
    write(nv_fd, "\n", 1);

    int ligne_vide = 0;
    // Lire le contenu de la comptine
    while (1) {
        int bytes = read_until_nl(fd, buffer);
        if (bytes > 0) {
            ligne_vide = 1;
            buffer[bytes] = '\n';
            write(nv_fd, buffer, bytes + 1);
        } else {
            if (ligne_vide) {
                // Ajouter une nouvelle ligne vide dans le fichier
                write(nv_fd, "\n", 1);
                ligne_vide = 0;
            } else {
                break;
            }
        }
    }
    close(nv_fd);

    liberer_catalogue(c);
    c = creer_catalogue(dir_name);
    if (c == NULL) {
        perror("Erreur lors de la création du catalogue après l'ajout de la
            nouvelle comptine.");
        return;
    }
}
```

N.B: Cette fonction ne marche pas malheureusement, je n'ai pas terminé le deboggage, il ya un problème au niveau de la lecture du contenu de la comptiné sur la socket.

2.4 Coté client

La fonction `recevoir_menu` permet de recevoir le menu envoyé par le serveur et de l'afficher sur le terminal du client.

```c
void recevoir_menu(int fd)
{
    char buffer[BUFSIZ];
    int bytes = read(fd, buffer, BUFSIZ);
    if (bytes <= 0) {
        perror("read menu non effectué");
        exit(2);
```

```
    }
    printf("%s", buffer);
}
```

2.5 Envoyer le choix du menu

La fonction `envoyer_choix_menu` permet à l'utilisateur de saisir un choix à partir du menu et de l'envoyer au serveur.

```
void envoyer_choix_menu(int fd)
{
    uint16_t choix;
    choix = saisir_num_comptine(4); // 4 options dans le menu
    envoyer_num_comptine(fd, choix);
    switch (choix) {
        // Traiter le choix du menu
    }
}
```

2.6 Recevoir et afficher la liste des comptines

La fonction `recevoir_liste_comptines` permet de recevoir la liste numérotée des comptines envoyée par le serveur et de l'afficher sur le terminal du client.

```
uint16_t recevoir_liste_comptines(int fd)
{
    int bytes;
    char buf[258]; // Pour stocker une ligne de la liste
    uint16_t nb = 0; // Compteur de comptines
    while ((bytes = read_until_nl(fd, buf)) != 0) {
        buf[bytes + 1] = '\0';
        printf("%s", buf); // Afficher la comptine
        nb++; // Incrémenter le compteur
    }
    return nb; // Retourner le nombre de comptines reçues
}
```

2.7 Saisir et envoyer le numéro de comptine

La fonction `envoyer_num_comptine` permet d'envoyer le numéro de la comptine choisie par l'utilisateur au serveur.

```
void envoyer_num_comptine(int fd, uint16_t nc)
{
    nc = htons(nc); // Conversion en network byte order
    write(fd, &nc, sizeof(nc)); // Envoi du numéro de comptine
}
```

2.8 Afficher une comptine

La fonction `afficher_comptine` permet de recevoir et d'afficher le contenu d'une comptine envoyée par le serveur.

```
void afficher_comptine(int fd)
{
    char buffer[BUFSIZ];
    int bytes;
    int ligne_vide = 0; // Pour indiquer si une ligne est vide
    while (1) {
        bytes = read_until_nl(fd, buffer); // Lecture jusqu'au saut de ligne
        if (bytes > 0) {
            ligne_vide = 1;
            buffer[bytes] = '\n'; // Ajout du saut de ligne manquant
```

```
            buffer[bytes + 1] = '\0';
            printf("%s", buffer); // Afficher la ligne
        } else {
            if (ligne_vide) {
                printf("\n"); // Afficher une ligne vide
                ligne_vide = 0;
            } else {
                return; // Fin de la comptine
            }
        }
    }
}
```

2.9 Ajouter une comptine

La fonction `ajouter_comptine` permet à l'utilisateur d'ajouter une nouvelle comptine au catalogue en saisissant le nom du fichier, le titre et le contenu de la comptine.

```
void ajouter_comptine(int fd, char *dir_name){
        char buffer[BUFSIZ];
        do{
                printf("Entrer le nom du nouveau fichier comptine : ");
                scanf("%s", buffer);
                printf("\n");
        }while(est_nom_fichier_comptine(buffer) == 0);
        //envoyer le nom du nouveau fichier
        write(fd,buffer, strlen(buffer));
        printf("le titre de la comptine : ");
        scanf("%s", buffer);
        printf("\n");
        //envoyer le titre de la comptine
        write(fd, buffer, strlen(buffer));

        //demander le contenu de la comptine
        printf("Entrez le contenu de la comptine. Terminez avec deux sauts de ligne
             consécutifs :\n");
        int bytes;
    int ligne_vide = 0; // Pour indiquer si une ligne est vide et cela aide à arrê
        ter la boucle (2 sauts de lignes consécutifs)
    while ((bytes = read(fd, buffer, BUFSIZ)) > 0) {
        buffer[bytes] = '\0';
        printf("%s", buffer);
        if (bytes >= 2 && buffer[bytes - 1] == '\n' && buffer[bytes - 2] == '\n') {
            ligne_vide = 1;
        } else {
            ligne_vide = 0;
        }
    }
    if (!ligne_vide) {
        printf("\n");
    }
}
```

3 Conclusion

En conclusion, ce projet a été une expérience enrichissante. Malgré le temps limité, nous avons pu ajouter des fonctionnalités significatives au serveur et au client. Cependant, nous aurions aimé avoir plus de temps pour explorer des améliorations telles que rendre le serveur multithreadé et maintenir un fichier de log. Ces idées restent des pistes à explorer dans le futur. Personnellement, je compte continuer à développer ce projet pour le rendre encore plus robuste. Dans l'ensemble, cette expérience a été stimulante et m'a donné envie d'approfondir mes connaissances dans ce domaine.