ElevatorSimulation

> Simulation d'un ascenseur
> Auteur : ELKHALKI Yassine
> Contact :
>
> - yassine.elkhalki@outlook.fr
> - yassine.elkhalki.auditeur@lecnam.net

Sommaire

Consignes : Simulation d’un ascenseur (threads et mutex)

Contexte :

- Implémentez un système d’ascenseur pour un bâtiment de 5 étages.
- Les utilisateurs (threads) demandent à monter ou descendre.

Fonctionnalités à implémenter :

- Gestion des requêtes d’appel et de destination.
- Simulation du mouvement de l’ascenseur.
- Synchronisation des utilisateurs pour accéder à l’ascenseur.

Objectifs pédagogiques :

- Synchronisation avancée des threads.
- Gestion des événements concurrents.

Implémentation

> L'ascenseur implémenté est un petit ascenseur. Il ne peut prendre qu'une personne à la fois.

Cette version repose sur le modèle Producteur-Consommateur décrit dans le cours. J'utilise donc :

- Deux mutex (***pthread_mutex_t***). Un pour garantir l'exclusion mutuelle lors de l'accès au buffer de requête partagé. Un autre pour s'assurer que les requêtes de tous les passagers ont été satisfaites.
- Deux sémaophores (***sem_t***).
    - **empty :** pour dire si l'ascenceur est libre ou non
    - **full :** pour dire qu'un passager est prêt à être transporté

La simulation propose un affichage pas à pas dans le terminal pour suivre l'ascenceur à travers les 5 étages. Voici un aperçu du rendu :

```
=== Lancement simulation Ascenseur avec 4 usagers ===


     RDC | 1  | 2  | 3  | 4  |   ETAT ASCENSEUR
    ---------------------------------------------
    [ E ]  |    |    |    |     <-- En attente...
[PASSAGER 0] Je veux aller de 4 à 2.
[ASCENSEUR] PRISE EN CHARGE PASSAGER 0 !
[PASSAGER 1] Je veux aller de 1 à 1.
[PASSAGER 1] Demande annulée. Je suis déjà à l'étage 1, pas besoin
d'ascenseur.
[PASSAGER 2] Je veux aller de 4 à 0.
      |  [ E ]  |    |    |     <-- Je vais chercher P0 à l'étage 4
[PASSAGER 3] Je veux aller de 2 à 2.
[PASSAGER 3] Demande annulée. Je suis déjà à l'étage 2, pas besoin
d'ascenseur.
      |    |  [ E ]  |    |     <-- Je vais chercher P0 à l'étage 4
      |    |    |  [ E ]  |     <-- Je vais chercher P0 à l'étage 4
      |    |    |    |  [ E ]   <-- Je vais chercher P0 à l'étage 4
      |    |    |    |  [ E ]   <-- Ouverture des portes ...
      |    |    |    |  [ P0]   <-- Passager P0 est monté
      |    |    |    |  [ P0]   <-- Fermeture des portes ...
      |    |    |  [ P0]  |     <-- Transport P0 -> 2
      |    |  [ P0]  |    |     <-- Transport P0 -> 2
      |    |  [ P0]  |    |     <-- Ouverture des portes ...
      |    |  [ E ]  |    |     <-- Passager P0 est descendu
      |    |  [ E ]  |    |     <-- En attente...
[ASCENSEUR] PRISE EN CHARGE PASSAGER 2 !
      |    |    |  [ E ]  |     <-- Je vais chercher P2 à l'étage 4
      |    |    |    |  [ E ]   <-- Je vais chercher P2 à l'étage 4
      |    |    |    |  [ E ]   <-- Ouverture des portes ...
      |    |    |    |  [ P2]   <-- Passager P2 est monté
      |    |    |    |  [ P2]   <-- Fermeture des portes ...
      |    |    |  [ P2]  |     <-- Transport P2 -> 0
      |    |  [ P2]  |    |     <-- Transport P2 -> 0
      |  [ P2]  |    |    |     <-- Transport P2 -> 0
    [ P2]  |    |    |    |     <-- Transport P2 -> 0
    [ P2]  |    |    |    |     <-- Ouverture des portes ...
    [ E ]  |    |    |    |     <-- Passager P2 est descendu
    [ E ]  |    |    |    |     <-- En attente...


=== Fin simulation ===
```

Architecture technique

Code source

Ce dépot est décomposé en deux sous-dossiers :

- **doc** : Qui contient la documentation (le sujet ainsi que le compte rendu)
- **src** : Qui contient le code source du projet dont :
  - **simulation.h** : Fichier qui contient la définition des prototypes et de la structure Passager
  - **simulation.c** : Fichier qui contient toute la logique métier notamment les fonctions des threads (ascenseur/usager), la gestion des sémaphores/mutex et la fonction d'affichage graphique
  - **main.c** : Point d'entré du programme qui gère la réccuperation des arguments (le nombre d'usager) et lance la simulation

Bibliothèques

La projet repose sur des bibliothèques standards du langage C et des bibliothuèques POSIX pour la gestion des threads:

- **<pthread.h>** : Fournit les fonctions nécessaires à la création et à la gestion des threads ainsi que les mutex
- **<semaphore.h>** : Utilisée pour contrôler l'accès aux ressources partagées et synchroniser les flux producteurs-consommateurs
- **<fnctl.h>** : Car développement sur macOS pour l'utilisation des constante lors de l'initialisation des sémaphores nommés
- **<unistd.h>** : permet l'utilisation des fonctions sleep et usleep pour simuler les temps de trajets

Usage

Pour compiler le projet, il faut avoir le compilateur **gcc** d'installé sur votre machine. Pour compiler et obtenir l'éxecutable sur Linux et macOS, rendez vous à la racine du projet, c'est à dire ici puis éxecutez la commande suivante :

```
gcc -Wall -pthread src/main.c src/simulation.c -o ElevatorSimulation
```

un exécutable du nom de `ElevatorSimulation` sera créé et il attend en paramètre le nombre d'usagers. Pour exécuter :

```
./ElevatorSimulation 4
```

Finalement pour supprimer l'exécutable :

```
rm ElevatorSimulation
```

Tests