HMIN105M - projet
Espace blanc collaboratif

A déposer au plus tard le lundi 2 décembre 2019 à 23h59

Instructions : Ce projet est à réaliser en trinôme. Lire attentivement l'énoncé avant de commencer le travail. Pour la mise en place des connexions et communications distantes, vous utiliserez le protocole TCP/IP et les fonctions C utilisées en cours. Le plagiat est strictement interdit.

L'idée de ce projet s'inspire du concept "Whiteboard". Il a pour objectif de mettre en place un espace partagé permettant à des clients distants d'échanger des informations ou des idées, de collaborer pour rédiger un article, de jouer, etc. Les usages possibles sont variés, il vous reviendra de définir quel usage vous en ferez, à condition qu'il soit totalement cohérent avec la spécification qui suit.

L'application à réaliser doit répondre à des contraintes bien déterminées. Elle est constituée d'un serveur concurrent et de plusieurs clients. Le rôle du serveur est de mettre en place l'espace partagé, de gérer les accès clients à cet espace, de maintenir ce dernier dans un état cohérent et d'informer les clients de toute modification effectuée. Le rôle des clients est d'échanger avec le serveur pour demander à modifier et/ou à visualiser le contenu de l'espace partagé.

Plus précisément :

1. Le serveur concurrent est constitué d'un processus parent qui est responsable de la mise en place de l'espace partagé sous forme d'un segment de mémoire partagé (IPC) et d'attendre les connections des clients. Le traitement de chaque client est délégué à un processus fils qui lui sera exclusivement dédié. Ce processus fils, dès son lancement, sera responsable de l'envoi du contenu de l'espace partagé à son client, puis d'attendre toute modification demandée par ce client, de mettre à jour le contenu de l'espace et de déclencher une diffusion du nouvel état de l'espace à tous les clients. En parallèle, ce processus fils doit pouvoir envoyer à son client, toutes les modifications effectuées par les autres clients. Les threads sont à utiliser pour mettre en oeuvre le comportement parallèle au sein d'un processus fils (nécessaire donc pour le traitement de son propre client). Remarque : un processus fils ne peut communiquer qu'avec son propre client. Il sera nécessaire d'utiliser des tableaux de sémaphores pour les communications entre processus fils du serveur. Il sera aussi nécessaire de veiller à avoir un état cohérent du contenu de l'espace partagé en gérant les accès concurrents (aussi avec les tableaux de sémaphores). L'utilisation des files de messages n'est pas utile dans ce projet.
2. Le client doit avoir la possibilité de visualiser le contenu de l'espace après connexion au serveur, et en boucle, d'envoyer des modifications saisies au clavier, de recevoir et d'afficher le résultat après traitement par le serveur. En parallèle, le client peut recevoir les mises à jour effectuées par les autres clients. Ce comportement parallèle sera mis en oeuvre en utilisant les threads.
3. Deux clients ne doivent pas avoir la possibilité de modifier en même temps une même zone de l'espace partagé (on parlera de zone à écriture exclusive). Ils peuvent toutefois modifier simultanément des zones différentes. Ce comportement doit être visible à l'exécution et il vous revient de faire le bon choix pour gérer les demandes de modifications simultanées. Vous avez aussi le choix d'autoriser ou pas des lectures en même temps qu'une écriture.

4. L'usage que vous aurez choisi de cette application aura un impact sur le contenu de l'espace partagé, la définition de ce qu'est une zone à écriture exclusive et la façon de traiter les demandes des clients. Faites simple. Dans tous les cas, votre choix ne devra pas affecter le comportement général du serveur et du client décrit ci-dessus.

Le projet est organisé en étapes progressives. Chaque étape mérite d'être bien réfléchie avant d'être mise en oeuvre, sans oublier de faire le lien avec les autres étapes.

1. Définir un usage de l'espace partagé, de la structure/contenu de cet espace, des zones à écriture exclusive, du type d'actions à réaliser et des données à transférer lors d'une modification. Remarque : après chaque modification, il est possible d'envoyer le contenu de tout l'espace aux clients (vous avez la possibilité de commencer par cela). Toutefois, il serait plus efficace, si possible, d'avoir une stratégie permettant de n'envoyer que les données modifiées (travail optionnel qui sera pris en compte en bonus).
2. Réaliser une version centralisée / sans communications distantes et sans lien de parenté au sein du processus serveur : dans cette partie, il s'agit de mettre en place une architecture concurrente du serveur. Le comportement du client sera intégré dans des processus serveurs dédiés. Plus précisément, vous aurez à mettre en place :
   - Un programme serveur dit "maître" qui mettra en place l'espace collaboratif et tout ce qui est nécessaire pour la gestion de plusieurs clients.
   - Un programme client intégré à un serveur dédié. Ce programme aura pour rôle d'afficher le contenu de l'espace partagé à son lancement, et en boucle, d'attendre une saisie de l'utilisateur, de mettre à jour le contenu de l'espace et de notifier cette modification à tous les autres processus clients pour qu'ils affichent le nouveau contenu à leur propre utilisateur. Bien entendu, en parallèle, un processus doit pouvoir afficher à son utilisateur, toute modification effectuée par un autre client/processus.
3. Une fois la première étape terminée et testée, garder une copie. Modifier maintenant le code pour mettre en place l'architecture père-fils du serveur et des clients distants, où :
   - le serveur parent aura à créer un serveur fils à chaque connexion d'un nouveau client.
   - une demande (via une saisie au clavier) de modification du contenu de l'espace partagé nécessitera maintenant un échange de messages avec le serveur fils dédié.
   - un processus serveur fils aura à envoyer des messages pour que le processus client puisse visualiser le contenu de l'espace partagé.
   - etc.

Quelques remarques :

- la diffusion d'une mise à jour du contenu de l'espace partagé implique des envois parallèles de cette mise à jour aux différents clients.
- la gestion des erreurs est indispensable. Exemples : si un objet IPC est supprimé, les processus doivent quitter proprement ; si un processus fils est tué, l'exécution doit pouvoir se poursuivre sans erreurs (le traitement des autres clients doit pouvoir se poursuivre, sauf éventuelles exceptions) ; en cas de déconnexion ou fermeture d'une socket, un processus concerné doit quitter proprement.
- Bien prendre en compte l'ergonomie de votre application (elle sera installée et exécutée par des utilisateurs qui n'auront pas à lire le code avant de comprendre les fonctionnalités réalisées)
- Il n'est pas demandé de réaliser une interface graphique.

Dépôt de votre travail

1. Le code source à remettre sera le code le plus fonctionnel (soit l'application centralisée, soit l'application distribuée, soit l'application distribuée avec bonus). Remarque : le bonus ne sera pris en compte que si le reste est complètement fonctionnel. Le code source doit être positionné dans un répertoire "projet", incluant aussi un Makefile.

2. Créer une archive projet.tgz contenant votre répertoire "projet" et un fichier README décrivant le travail effectué et donnant les instructions nécessaires pour la compilation et l'exécution.
3. Désigner un membre de votre trinôme pour déposer l'archive sur Moodle : HMIN105M : atelier projet (donc un seul dépôt par trinôme). Le dépôt doit être anonyme : le contenu de vos fichiers, les noms des fichiers, etc. ne doivent contenir aucune information permettant d'identifier les propriétaires du travail déposé.
4. Le dépôt est possible avant lundi 02/12 à 23h59 pour procéder à une évaluation par les pairs le mardi 03/12 de 15h à 18h15.