HMIN105M - projet
Espace blanc collaboratif

A déposer au plus tard le lundi 4 décembre 2018 à 12h

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/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 dans le cadre de ce projet.

L'application à réaliser doit toutefois répondre à des contraintes définies. 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 modifier et/ou 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, sera responsable de l'envoi du contenu de l'espace partagé au client à son lancement, d'attendre toute modification demandée par son 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, le 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 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. L'usage que vous aurez choisi de cette application aura un impact sur le contenu de l'espace partagé 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 plusieurs étapes. 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, 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 : dans cette partie, il s'agit de mettre en place l'architecture concurrente du serveur. Le comportement du client sera intégré dans les processus fils. Plus précisément :
   - Le serveur parent prendra en paramètre le nombre de fils à créer.
   - chaque fils aura pour rôle d'afficher le contenu de l'espace partagé, 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 processus pour qu'ils affichent le nouveau contenu à leur utilisateur. Bien entendu, en parallèle, le processus fils doit pouvoir afficher à son utilisateur, toutes les modifications effectuées par les autres utilisateurs/processus fils.
3. Une fois la première étape terminée et testée, garder une copie. Modifier maintenant le code pour mettre en place des clients distants :
   - le serveur parent créera un fils à chaque connexion d'un nouveau client (le nombre de client n'est pas connu à l'avance et n'est donc plus un paramètre du programme)
   - la saisie au clavier d'une modification du contenu de l'espace se fait par le programme client et non plus par le processus fils (transformer la saisie dans le processus fils par une réception de message).
   - l'affichage du contenu de l'espace se fait dans le programme client (transformer les affichages dans le processus fils par des envois de messages).
   - 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 (traitement des autres clients) ; 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 remis.
4. Le dépôt est possible avant mardi 04/12 à 12h pour procéder à une évaluation par les pairs le mardi 04/12 de 15h à 18h15.