HMIN105M - TP noté 1

Durée : 1h15 (**à déposer impérativement avant 16h15)**

13 octobre 2020

Dans un jeu, N joueurs doivent suivre un parcours constitué de X étapes. Chaque étape ne peut supporter qu'un nombre limité de joueurs simultanément (ce nombre est entre 1 et N et peut être différent d'une étape à l'autre). Pour une étape Ei, si cette limite est dépassée, un joueur qui arrive à cette étape doit attendre que d'autres joueurs dépassent Ei et libèrent par la même occasion des places pour Ei. Si plusieurs joueurs sont en attente, un seul au hasard sera choisi à chaque fois qu'une place est libérée. Enfin, tous les joueurs doivent suivre toutes les étapes du jeu dans le même ordre et doivent aller jusqu'au bout du jeu.

Exemple : soit $E1$, $E2$ et $E3$ les étapes du jeu et NbJoeursEtape1 le nombre maximum de joueurs à pouvoir jouer $E1$ en même temps (NbJoeursEtape2 pour $E2$ et NbJoeursEtape3 pour $E3$). Le schéma algorithmique global d'un joueur sera :

Joueur :
si moins de NbJoeursEtape1 sont à $E1$, progresser, sinon attendre ;
jouer $E1$;
libérer une place pour $E1$;
si moins de NbJoeursEtape2 sont à $E2$, progresser, sinon attendre ;
jouer $E2$;
libérer une place pour $E2$;
si moins de NbJoeursEtape3 sont à $E3$, progresser, sinon attendre ;
jouer $E3$;
libérer une place pour $E3$;
arrivée à la fin du parcours.

Pour mettre en oeuvre la possibilité d'avoir un nombre maximum de joueurs par étape, nous mettons en place un système de jetons, via une structure de synchronisation appelée *JetonsEtape*. Cette structure permet à k ($1 \leq k \leq N$) joueurs d'être présents simultanément à une étape, mais pas plus de k joueurs à la fois. Chaque joueur est représenté par un thread. Pour utiliser des *JetonsEtape*, on dispose des 4 fonctions suivantes :

- **int jetonsEtape_init(JetonsEtape * v, int k)** pour initialiser la structure *JetonsEtape*. k est le nombre maximum de threads pouvant être simultanément dans une section critique.
- **int demande_jeton(JetonsEtape * v)** pour demander l'entrée dans la section critique.
- **int liberer_jeton(JetonsEtape * v)** pour avertir de la sortie de la section critique.
- **int jetonsEtape_destroy(JetonsEtape * v)** pour détruire les éléments de la structure.

L'ensemble de ces fonctions renvoie 0 en cas de succès, -1 en cas d'échec.

L'objectif de cet exercice est d'implémenter ces fonctions. Le reste (implémentation et création des threads joueurs, l'attente de leur terminaison, la gestion des paramètres, etc) est fait.

- Télécharger les fichiers fournis. Positionnez vous dans le répertoire srcTP. Vous y trouverez un squelette de code à étudier et à compléter (afaire.c), le reste du programme au format binaire (à ne pas ré-implémenter) et un Makefile pour compiler.
- Avant de commencer à travailler, vous pouvez compiler (commande "make") et exécuter le programme (commande "./bin/jeu"). Ainsi vous saurez comment démarrer.
- Etudiez attentivement le contenu des fichiers "commun.h" et "afaire.c" avant de procéder à la modification de ce dernier.

— La trace d'exécution du programme contient des indications vous permettant de savoir si vous n'êtes pas sur la bonne voix.

Les règles indispensables à respecter :

— Le code fourni produit des traces d'exécution suffisantes. N'ajoutez pas d'autres traces dans la version que vous déposerez.

— Testez votre programme avec différents paramètres (partant de cas les plus simples aux plus complexes). Vous remarquerez que la durée d'une étape peut atteindre jusqu'à 15 secondes.

Dépôt de votre travail Vous devez déposer un seul fichier source (le fichier afaire.c sans changer le nom). Tout fichier supplémentaire ou avec un autre nom sera automatiquement refusé. Votre dépôt doit être anonyme : votre nom/prénom/pseudo ne doit pas figurer ni dans le contenu du fichier, ni dans la description du dépôt.