TP № 5: Mémoire Partagée

Samir Aknine, Antoine Gréa & **Rémy Grünblatt** 22/04/2020

Compétences à acquérir:

- Savoir créer un espace de mémoire partagée
- Savoir gérer une ressource critique

Information: Il est extrement conseillé d'avoir recourir à une machine sous GNU/Linux (par exemple, Ubuntu). Il est possible d'utiliser `https://shell.univ-lyon1.fr` en secours, en ouvrant plusieurs terminaux (notamment dans le cas du processus zombie), mais cela ne devrait pas être la première option pour vous.

Important: Tous les TPs feront l'objet de rendus, deux semaines après le TP, et seront notés; les modalités de rendu seront précisées par email ou sur le chat discord de la classe. Il convient donc de bien noter les commandes exécutées, par exemple dans un fichier texte ou markdown, et leur résultats, afin de préparer le rendu du TP.

Exercice 1: Expérimentation

Question 1: 22, vla les ...

Étudier le fichier « `shmid.c` » et le commenter. À quoi servent les opérations de la ligne 19 ? (première ligne exécutée uniquement par le père). Le code présente il une situation de « *race condition* » ? Proposer une correction simple le cas échéant.

Question 2: Tri

Modifier le père afin qu'il stocke des valeurs aléatoires dans le tableau, puis implémenter un algorithme de tri dans le fils puis afficher les valeurs triées dans le père.

Exercice 2: Morpion

Le but de cet exercice est d'implémenter un jeu de morpion entre processus. Chaque processus joue chacun à son tour sur un plateau de jeu placé dans un segment de mémoire partagée. Plusieurs types de joueurs vont être implémentés:

Question 1: Arbitre

L'arbitre est le processus père. Il crée deux processus enfants, attribue un type de jeton pour chaque enfant, et gère les tours de jeu. Enfin, il déclare le vainqueur (ou « ex æquo » le cas échéant). Implémenter le programme arbitre. Dans l'ordre, ce programme:

- Crée un segment de mémoire partagée de la bonne taille (qui n'est pas la taille du plateau, mais plus pour gérer la synchronisation);
- Décide de l'attribution des pions (blanc ou noir, rouge ou bleu, zéro ou un…) pour chaque joueur;
- Crée deux processus enfants (qui seront les joueurs), et conserve leurs PIDs en mémoire;
- Vérifie si le jeu est fini, et tant que le jeu n'est pas fini, décide tour par tour qui doit jouer;

Question 2: le débutant

Le joueur « Débutant » est un joueur qui place ses pions au hasard, et joue uniquement sur des cases vides. Créer une fonction « `joueur_debutant` » qui pourra être exécutée par l'un des processus enfant de l'arbitre.

Question 3: le malin

Le joueur « Malin » est un joueur qui tente de bloquer la victoire de l'adversaire. Il joue toujours sur les cases adjacentes au dernier coup du joueur précédent, et jouera pour éviter la défaite et prioriser la victoire. créer une fonction « `joueur_malin` » qui pourra être exécutée par l'un des processus enfant de l'arbitre.

Question 4: le tricheur

Le joueur « Tricheur » est un joueur qui triche et s'autorise à jouer sur des cases déjà prise. Créer une fonction « `joueur_tricheur` » qui pourra être exécutée par l'un des processus enfant de l'arbitre.

Question 5: Game Over

Faire combattre ces joueurs l'un contre l'autre plusieurs centaines de fois, automatiquement. Quelles sont les probabilités de chacun de gagner?

Question Bonus: le polytech

Le joueur polytech, c'est vous. Créer une fonction qui vous permet de joueur contre les autres joueurs, et qui pourra être exécutée par l'un des processus enfant de l'arbitre.