TP1 : Course de petits chevaux

but : vous faire manipuler les sémaphores et les segments de mémoire partagée

validation : lors de cette séance de TP

dossier : le code source du programme `cheval.c` commenté. À rendre au plus tard le Vendredi de cette semaine

1 Cadre du TP

On souhaite faire participer plusieurs processus (appelés dans la suite les *chevaux*) dans une course de petits chevaux se déroulant de la manière suivante :

TQ le cheval n'a pas franchit la ligne d'arrivée **FRE**

- le joueur lance un dé pour faire avancer son cheval. Le dé donne un nombre N compris entre 1 et 6.
- le joueur déplace son cheval sur la piste de N cases : il s'élance de la case de départ, effectue son saut pendant un certain temps et atterrit dans la case d'arrivée. Si la case d'arrivée est déjà occupée par un autre cheval, alors ce dernier est "décanillé" (il est éjecté de la piste).
- le cheval fait une pause avant de recommencer à jouer

FTQ

Les processus *chevaux* courent bien sûr sur la même piste.

2 Remarques

Voici quelques remarques avant de vous donner le travail à faire dans ce TP :

1. Les processus *chevaux* courent sur une piste qui est un segment de mémoire partagée dont la clé externe vous sera donnée lors de la séance de TP.
2. Ceci implique que tous les processus *chevaux* doivent être exécutés sur la même machine. Le nom de cette machine vous sera donné lors de la séance de TP également.
3. La ressource critique de la piste est protégée par un sémaphore Unix dont la clé externe est la même que celle du segment de mémoire partagée. Il est composé d'un ensemble de sémaphores individuels d'exclusion mutuelle protégeant chacun une case de la piste. Un sémaphore

à le même numéro que celui de la case qu'il protège (S_i protège la case $n^o i$) : par exemple, le sémaphore S_0 protège la case $n^o 0$ de la piste. La FIGURE 1 montre les structures relatives à la gestion de la piste.

4. Quand un cheval veut participer à la course, il doit s'enregistrer dans une liste. Cette liste contient pour chaque cheval son état : `EN_COURSE`, `ARRIVE` ou `DECANILLE`. Donc un cheval "décanille" un autre cheval en mettant à jour l'état de ce dernier dans la liste. Un cheval sait qu'il est décanillé en consultant à chaque tour de jeu son état. Ainsi, cette liste est aussi une ressource partagée protégée par un autre ensemble de sémaphores Unix. Cet ensemble est ici composé d'un seul sémaphore individuel d'exclusion mutuelle protégeant l'ensemble de la liste. Cette liste est sur la même machine que la piste et la clé externe de son segment de mémoire partagée et de son sémaphore vous sera donnée également lors de ce TP. La FIGURE 1 montre les structures relatives à la gestion de la liste.
5. Pour éviter qu'un cheval soit "décanillé" en plein saut (c'est à dire entre sa case de départ et d'arrivée), on protège un cheval par un sémaphore. Ainsi lorsqu'il se déplace, il doit prendre son sémaphore. Si un cheval C_1 veut décaniller un autre cheval C_2, c'est à dire que C_1 veut mettre l'état de C_2 à `DECANILLE` alors C_1 devra d'abord prendre le sémaphore de C_2.
6. Les TDA `piste` et `cell` donnent les structures de la piste et des cases ainsi que les fonctions qui les manipulent (gestion des affichages, des pauses entre 2 tours de jeu, affectation de cases...). Les TDA `liste` et `elem` donnent les structures et les fonctions relatives à la liste. Ceux ci peuvent être trouvés à :
   — `/info/tmp/AnnexesTPM1_177UD02/TP_Petits_Chevaux`
   — ou depuis ma page *Enseignements*
7. le fichier `test_piste.c` contient des exemples d'appel des fonctions du TDA piste.
8. le fichier `test_liste.c` contient des exemples d'appel des fonctions du TDA liste.
9. une proposition d'un "patron" du programme `cheval` est donné dans le source `cheval_etu.c`
10. un fichier `Makefile_etu` contient les règles de compilation de ce TP. Pour compiler l'ensemble de votre TP, vous pouvez utiliser la commande suivante

    ```
    make -f Makefile_etu all
    ```

11. Le processus qui gère la piste et la liste est en principe lancé au début de la séance de TP. C'est lui qui est chargé de la création des deux segments de mémoire partagée et des deux ensembles de sémaphores.

3 Sujet du TP

Le but de ce TP est d'écrire une commande `cheval` appelable depuis un shell sous la forme

`cheval` *clé_externe_piste clé_externe_liste marque*

qui exécute un processus cheval qui sera repéré sur la piste par le caractère *marque*, qui courra sur le segment partagé de la piste de clé externe *clé_externe_piste* et sera enregistré sur la liste de clé externe *clé_externe_liste* jusqu'à ce qu'il ait franchi la ligne d'arrivée ou qu'il soit décanillé. Un joueur peut éventuellement lancer plusieurs chevaux.

4 Schéma du système

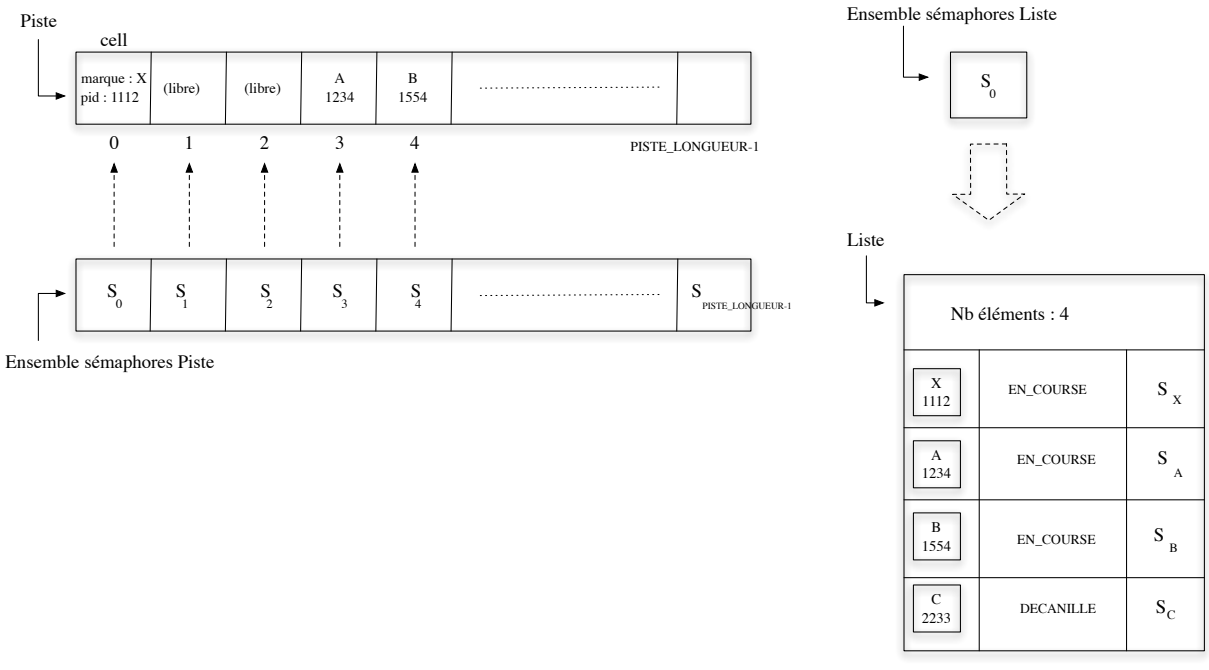

FIGURE 1 – Structures de la piste et de la liste et leur ensemble de sémaphores