Sujet des TP1 & 2: Jeux de Bataille Navale

Bruno Jacob, LIUM , IUPMIME, L3, Inf310A

1 Présentation générale

On désire modéliser un jeu de bataille navale entre plusieurs bateaux. Un bateau se déplace dans la mer et tire un boulet sur un autre bateau. Si le bateau adverse a suffisamment d´énergie, alors il maintient activé son bouclier de protection qui fait "rebondir" le boulet sur sa coque ; sinon le boulet s'incruste dans le corps du bateau et celui-ci coule dans la mer.
A chaque déplacement, le bateau consomme 5% de son énergie. On supposera que si son énergie est inférieure à une constante `SEUIL_BOUCLIER`, alors il n'en a pas assez pour activer son bouclier.
Le dernier bateau à encore naviguer dans la mer est déclaré vainqueur.

Pour réaliser ce jeu, on considère que :

- On utilise le système d'exploitation Unix vu en cours
- La Mer est vu comme un tableau à deux dimensions de `L`×`C` cellules contenant un caractère. Le tableau est implémenté par un fichier dont la référence est connue de tous les bateaux. On suppose que cette référence est contenue dans la constante globale `FICHIER_MER`.
- Chaque bateau est un processus indépendant. Il possède une longueur fixe de `LONGUEUR_BATEAU` cellules dans la mer. Il se déplace dans les cellules adjacentes à celles de sa position courante. Le temps entre 2 déplacements est aléatoire. La direction dans laquelle il se déplace est également aléatoire.
- Un bateau signale la position qu'il occupe en écrivant dans les cellules du tableau de la mer une marque distinctive (un 'X' ou un 'O' par exemple).

2 Sujet du TP

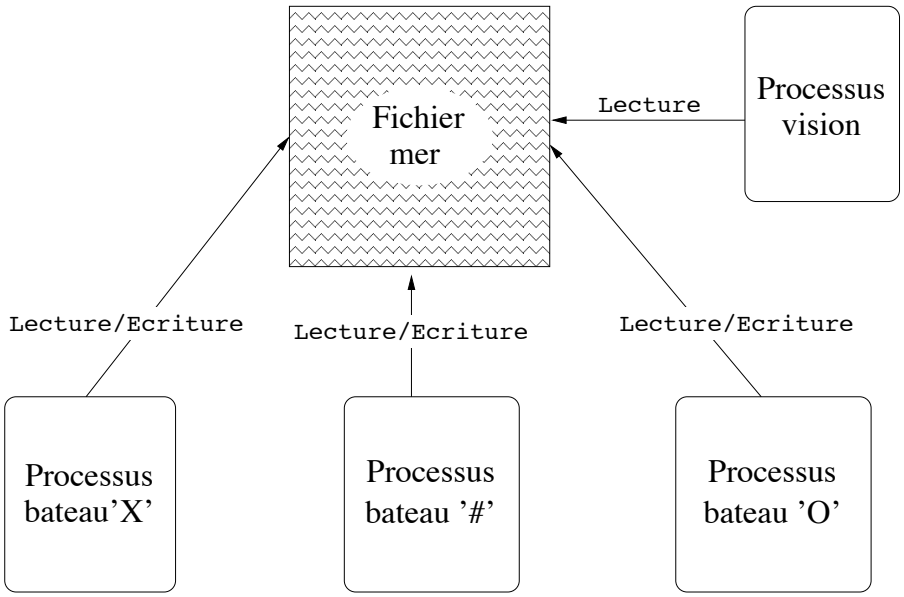
Le fichier mer dans lequel naviguent les bateaux est une ressource critique. Afin de résoudre les problèmes d'accès concurrents, deux solutions sont envisagées pour réaliser ce jeu :

1. Les processus 'bateaux" écrivent directement dans le fichier "mer".
2. Les processus "bateaux" communiquent avec un processus central "bateau amiral" qui, lui, met à jour le fichier "mer".

Le but de ce TP est de réaliser deux "sous-TPs" qui correspondent à l'avancement des bateaux dans les deux solutions.

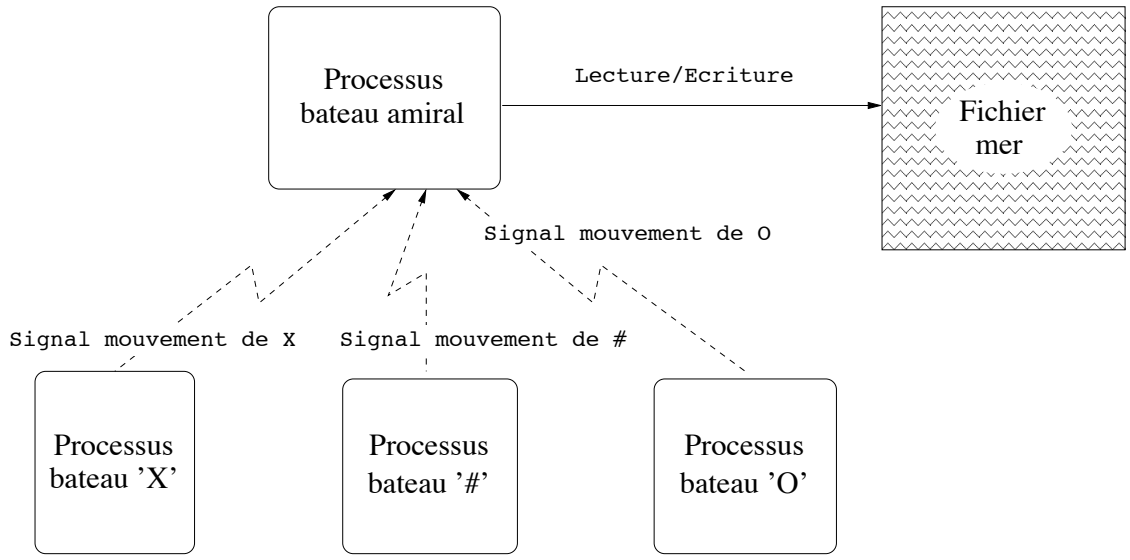
2.1 Solution 1

Cette solution revient à être dans la même configuration que celle du TP1. Plusieurs processus de type `bateau` peuvent écrire en même temps dans une ressource critique qui est le fichier représentant la mer. Un seul processus de type `vision` est lancé pour visualiser la mer quand il y a un changement dans la situation du jeu.



2.2 Solution 2

Cette solution revient à être dans la même configuration que celle du TP2. Un seul processus `bateau amiral` centralise toutes les opérations. Dans ce TP cela revient à ce qu'il soit le seul à écrire et à lire dans la ressource critique. Il règle ainsi lui-même les problèmes d'accès concurrents. Les processus de type `bateau` signalent simplement au processus `bateau amiral` qu'ils veulent se déplacer dans le fichier mer.



Si vous le souhaitez, vous pouvez utiliser le TDA des bateaux pour initialiser la communication entre les processus de type `bateau` et `bateau amiral`. Ce TDA offre des primitives de gestion (Création/Ajout/Consultation) d'un fichier contenant, entre autres, les pid du processus `bateau amiral` et ceux des processus `bateau`.

3 Organisation du code

Dans la programmation du sujet de ce TP, vous pourrez utiliser les définitions des TDA qui sont définis dans ce paragraphe. Les codes sources des définitions et des réalisations (les fichiers *.{h,c}) de ces TDA peuvent être trouvés :

- sur le compte `/info/tmp/AnnexesTPL3_Inf310A/TP_Bataille_Navale`
- sur le Web à `http ://www-ic2.univ-lemans.fr/~jacob/enseignement.html`