# Sujet des TP1 & 2: Jeux de Bataille Navale

*Bruno Jacob, IC2, L3 SPI*

## 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.

## Présentation technique

Pour réaliser ce jeu, on considère que :

- 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 processus existants dans le jeu.
- Chaque navire est un processus indépendant. La taille d'un navire est de longueur fixe: `BATEAU_TAILLE` 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).

# Sujet du TP

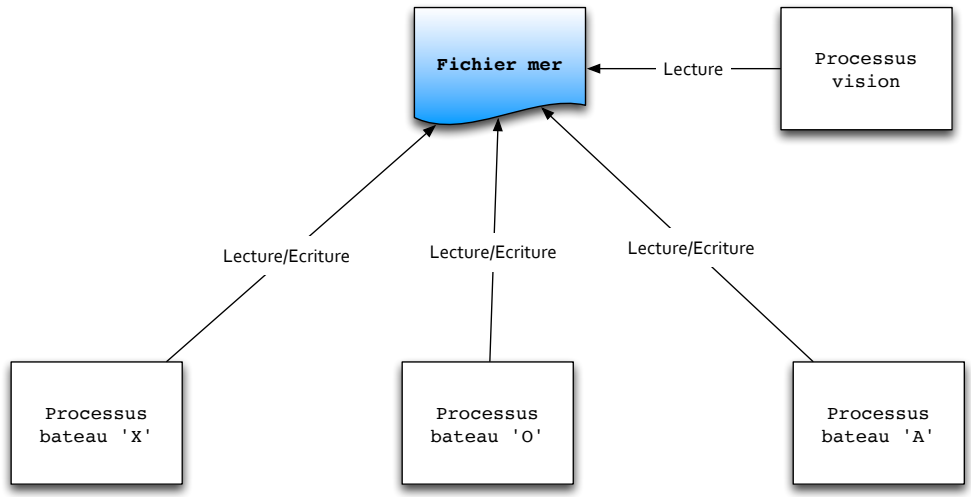
Le fichier mer dans lequel naviguent les bateaux est une ressource critique. Afin de résoudre les problèmes d'accès concurrents, deux solutions sont envisagées pour réaliser ce jeu :

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 ces TP est de programmer ces 2 réalisations qui correspondent à l'avancement des bateaux dans les deux solutions.

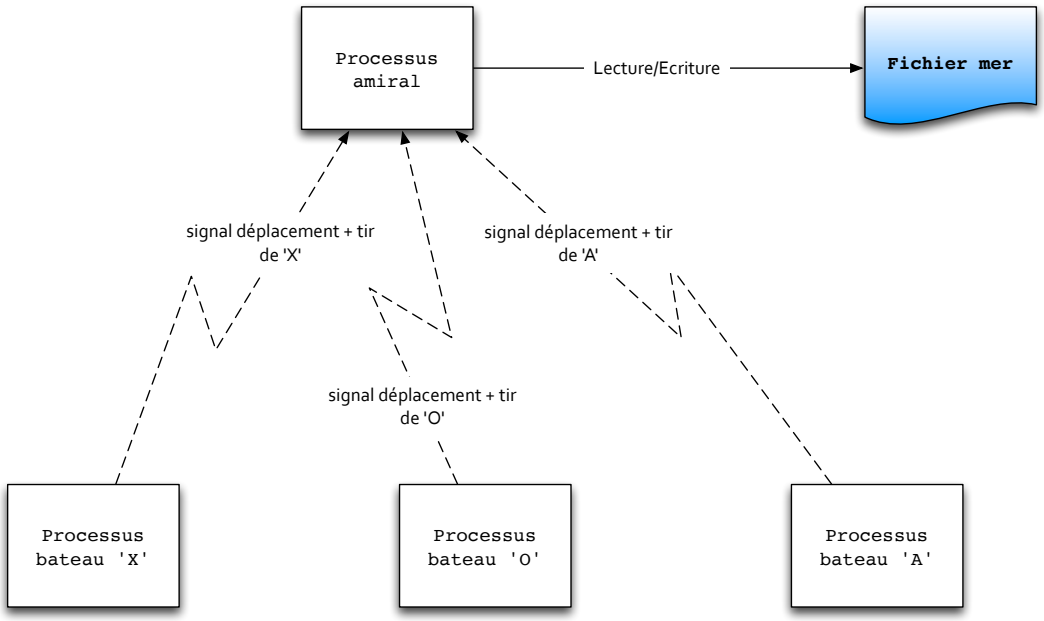
## Solution 1

C'est le sujet du TP1. Plusieurs processus de type `navire` 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.




## Solution 2

C'est le sujet du TP2. Un seul processus `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 `navire` signalent simplement au processus `amiral` qu'ils veulent se déplacer et faire un tir dans le fichier mer.




Si vous le souhaitez, vous pouvez utiliser le TDA des bateaux pour gérer la communication entre les processus de type `navire` et `amiral`. Ce TDA offre des primitives de gestion (Création/Ajout/Consultation...) d'une liste de bateaux.

Pour que `amiral` puisse identifier un `navire` par son signal on peut envisager deux solutions:

- chaque `navire` à son propre signal mais cela limite le nombre de vers au nombre de signaux utilisables sur votre machine
- tous les `navire` ont le même signal et `amiral` récupère le pid du processus émetteur grace
  - à une structure `siginfo_t` passée en paramètre du handler du signal
  - en positionnant `SA_SIGINFO` dans le champ `sa_flags` de la structure `sigaction` lors de la capture du signal par la fonction `sigaction`.

  Faites un `man sigaction` pour avoir plus d'informations

**Contrainte imposée dans ce TP** : l'énergie et le seuil du bouclier doivent rester des caractéristiques propres à chaque `navire` , `amiral` ne les connait pas.

# Téléchargement

1. Récupérez l'archive `TP_Bataille_Navale.tar.gz` depuis
   - soit le répertoire `/info/tmp/AnnexesTPL3_175EN007/TP_Bataille_Navale/`
   - soit la page `http://www-info.univ-lemans.fr/~jacob/enseignement.html`
2. Décompresser cette archive par les commandes :

   ```
   gunzip TP_Bataille_Navale.tar.gz
   tar xvf TP_Bataille_Navale.tar
   ```

   Normalement vous devriez obtenir une arborescence de fichiers expliquée dans le fichier `README`
3. Compilez tout votre projet depuis sa racine par

   ```
   cd TP_Bataille_Navale
   make -f Makefile all
   ```

4. Testez éventuellement par `make -f Makefile tests`
5. Par la suite, vous pourrez compiler les programmes de vos 2 TPs en vous situant dans leur répertoire respectif (`Verrous` ou `Signaux`) et en faisant la commande

   ```
   make -f Makefile all
   ```

# Validation

- il y aura un test sur machine
- vous donnerez les sources que vous avez produits, c'est à dire
  - les fichiers `navire.c` et `vision.c` pour le TP1
  - les fichiers `navire.c` et `amiral.c` pour le TP2
- et un petit rapport sur comment vous avez utilisé
  - les verrous (pour le TP1)
  - et les signaux (pour le TP2)