Programmation Système :
TP1 – Bataille Navale par Verrous

FROGER Adrien – L3 info G2

Emploi des verrous :

Rappel :

exclusif → incompatible avec la pose de tout autre verrou
partagé → compatible avec la pose d'un verrou de même type

bloquant → le processus est mis en attente si impossible de verrouiller jusqu'à verrouillage
non bloquant → la fonction fcntl renvoie une erreur si impossible de verrouiller et poursuite de l'exécution

- **Dans navire.c**

► Initialisation d'un bateau dans la mer

Type de verrou : écriture (exclusif)
Mode : bloquant
Portée : tout le fichier mer

J'ai opté pour un verrou exclusif car les bateaux déjà initialisés dans la mer ne doivent pas pouvoir se déplacer durant l'initialisation d'un nouveau bateau. En effet, ce dernier pouvant s'initialiser n'importe où dans la mer (positionnement aléatoire), il ne faudrait pas qu'un autre bateau tente d'écrire à l'endroit qu'il aura choisi au même moment. C'est aussi pour le fait que le positionnement soit aléatoire que la portée du verrou concerne la totalité du fichier mer.

► Déplacement d'un bateau dans la mer

Type de verrou : écriture (exclusif)
Mode : non bloquant
Portée : toutes les cases libres voisines au bateau qui souhaite se déplacer

J'ai opté pour un verrou en écriture car il est exclusif : sa pose est incompatible avec tout autre verrou. Ainsi, deux navires qui convoiteraient la même place ne peuvent pas verrouiller tous les deux cette place pour s'y déplacer. Le mode est non bloquant pour éviter le dead-locking. En cas d'échec de pose d'un verrou sur une des cases voisines, la fonction renverra ERREUR (-1) et le déplacement sera alors considéré comme impossible et sera abandonné.

► Bouclier d'un bateau

Type de verrou : lecture
Mode : bloquant
Portée : les cases constituant le corps du bateau

J'ai opté pour un verrou en lecture afin de pouvoir permettre le verrouillage de l'ensemble du fichier mer en lecture lorsqu'aucun déplacement n'est en cours. Ainsi, si l'on décide que vision.c verrouille l'ensemble de la mer en lecture lorsqu'il affiche la mer, on en a la possibilité (même si ce n'est pas ce que j'ai fait)
Le verrou en lecture n'est compatible qu'avec avec la pose d'autre verrous de type lecture. Ainsi, si un navire souhaite tirer sur une case avec bouclier, lorsqu'il essaiera de verrouiller en écriture cette derrière, cela renverra un code d'erreur. Le tir sera alors considéré comme impossible.
Le mode choisi est bloquant : le bateau attend d'avoir activé son bouclier pour poursuivre le jeu.

► Verrouillage d'une cible

Type de verrou : écriture (exclusif)
Mode : non bloquant
Portée : la case visée par le tir

Lorsqu'un bateau souhaite tirer sur un autre, il tente de poser un verrou en écriture sur la case qu'il vise. Si le bateau dispose d'un bouclier, le verrou ne pourra alors être posé car le verrou en lecture symbolisant le bouclier est incompatible avec la pose d'un verrou en écriture. Le mode est non bloquant de sorte que lorsque le bateau essaie de verrouiller une cible protégée par un bouclier, la fonction fcntl renverra -1. Le tir sera alors considéré comme impossible.
On ne peut pas utiliser un mode bloquant sinon le tir s'effectuerait lorsque le bateau s'est déplacé à son ancienne position, autrement dit, dans la mer.

► Verrouillage du header du fichier mer

Type de verrou : lecture
Mode : bloquant
Portée : le header du fichier mer

Le but de ce verrouillage est de notifier le déplacement en cours d'un bateau. Lorsqu'un bateau souhaite se déplacer, il place un verrou en lecture sur le header du fichier mer pour signaler qu'il est en cours de manœuvre. Lorsqu'il a terminé, il lève ce verrou. Plusieurs bateaux peuvent se déplacer simultanément, j'ai donc opté pour un verrou en lecture (compatible avec d'autres verrous du même type). Ainsi, plusieurs navires peuvent verrouiller le header en même temps. Le but de ce signalement est de ne pas afficher un bateau en cours de manœuvre dans vision.c qui s'assurera alors qu'aucun verrou n'est placé sur le header.
Le mode est bloquant : si il existe trop de verrous sur le header, le processus attend la levée d'un verrou pour verrouiller à son tour avant de poursuivre.

- **Dans vision.c**

Vision.c utilise F_GETLK pour tester la présence d'un verrouillage sur le header du fichier mer. Si un verrouillage est en cours, c'est qu'au moins une manœuvre est en cours. Il faut attendre qu'elle(s) se termine(nt) pour afficher la carte de la mer.

Réflexion :

J'ai opté pour le choix de verrouiller le header du fichier mer en lecture lors d'un déplacement afin d'avoir un moyen de m'assurer qu'aucun déplacement n'est en cours lors d'un affichage dans vision.c. L'autre solution aurait consisté à verrouiller l'ensemble de la mer en lecture dans vision.c à chaque affichage. A y réfléchir, cette seconde solution est peut être meilleure car elle évite l'attente active dans vision.c (`while(verif_verrou_header(fd_mer))`) mais elle est peut être moins intéressante dans le cas d'un fichier mer de grande taille où le verrouillage/déverrouillage intégral serait peut être également coûteux.

D'autre part, pour l'initialisation d'un bateau dans la mer, je verrouille l'intégralité du fichier mer alors que j'aurais pu ne pas verrouiller le header (grace à l'utilisation du champ l_strat). J'ai en effet trouvé cette option plus commode et cela évite tout risque d'erreur (par exemple, au cas ou la taille indiquée pour le header serait erronée).

Enfin, il y a un risque potentiel d'attaque entre le déverrouillage du voisinage post-déplacement et la réactivation du bouclier pour un bateau. En effet, le bateau pourrait être touché entre ces deux étapes (bien qu'elle se fassent juste à la suite l'une de l'autre). Il aurait peut être été plus judicieux de passer un paramètre supplémentaire de type booléen à la fonction navire_deverrouiller_voisinage qui indique si le bateau peut disposer ou non d'un bouclier et, en fonction de ce booléen, déverrouiller la totalité des cases contenues dans liste_voisins ou la totalité des cases contenues dans liste_voisins SAUF les cases où s'est déplacé le bateau (emplacement qu'il occupe après son déplacement).