BONNET Ludivine
MOLINER Emma

Compte rendu Projet

I- Introduction

Dans le cadre de notre projet, nous devons créer en C un jeu de bataille navale en utilisant nos connaissances acquises en cours de Système d'exploitation. Ce jeu doit être réalisé à partir du fichier d'entrée qui comporte la caractérisation de la carte, son type, sa taille, ainsi que la caractérisation de la partie, c'est-à-dire le nombre de joueurs avec leurs valeurs initiales de coque et de kérosène et le nombre de tours maximum joués.

Le jeu se termine lorsqu'il reste un seul joueur sur la carte navale et que tous les autres bateaux ont coulés. Un bateau coule lorsque la valeur de son kérosène ou de sa coque est inférieure à 0. Chaque joueur a la possibilité de choisir entre 5 actions disponibles : l'attaque, le déplacement, le radar, la charge et la réparation. Le but de chaque équipe est de faire couler les bateaux des équipes adverses. Ainsi, l'attaque constitue l'action essentielle de ce jeu.

II- Etape 1 : Gestion de système de fichiers

Nous avons utilisé les appels systèmes *open, read* et *close.* Pour ouvrir le fichier on utilise *open* qui prend en argument le chemin du fichier à ouvrir et le *flags* de mode d'accès (en lecture dans notre cas). Cette fonction renvoie le descripteur de fichier qui sera utilisé ensuite pour la fonction *read.* Si le descripteur est égal à -1, on sort de la fonction. Pour lire le fichier, on parcourt le fichier caractère par caractère et chaque caractère qui est lu est stocké dans un tableau de caractères. C'est pour cela que notre fonction *read* s'arrête quand elle rencontre un « ; ». Il faut ensuite fermer correctement le fichier en utilisant la fonction *close* qui prend en paramètre le descripteur de fichier à fermer.

III- Stockage des bateaux

Pour garder en mémoire l'état d'un bateau, nous avons créé une structure de données nommée « bateau » où l'on stocke la valeur de la coque, du kérosène et l'ID du bateau. Pour stocker les bateaux de tous les joueurs, nous avons fait choix de créer un tableau de *struct bateau*. La fonction *init_bateau* de notre programme se charge d'allouer de la mémoire pour le tableau de structure en fonction du nombre de joueurs qu'il y a sur la carte navale. Elle s'occupe également d'initialiser tous les bateaux avec les valeurs de coque et de kérosène initiales que l'on a fourni dans le fichier d'entrée. La fonction *free_bateau* s'occupe de libérer la mémoire du tableau que l'on avait alloué précédemment.

IV- Etape 2 : Les joueurs (Processus)

Pour effectuer les actions, le serveur agit dans le processus père. Celui-ci tient compte de la prochaine action qui a été déterminée dans chaque processus fils. Le processus fils détermine la

prochaine action à effectuer à l'aide de l'algorithme naïf donné dans l'énoncé du projet. Pour communiquer entre le processus père et le processus fils, nous avons utilisé des tubes anonymes qui permettent au processus père de lire l'information dans le tube et ensuite d'appliquer l'action.

La fonction radar permet de détecter le bateau le plus proche et de récupérer ses informations telles que ses coordonnées, ainsi que les valeurs de sa coque et de son kérosène.

Afin de déterminer l'ordre des actions à enchaîner, on implémente un algorithme « naïf » qui prend en compte le fait qu'on connaisse ou non la position adverse. Si la position adverse est non connue ou alors que le radar est trop vieux alors on effectue l'action radar. Un radar est considéré trop vieux tous les 4 tours. Si la position adverse est connue, il y a plusieurs possibilités :
- soit elle est à portée d'attaque
- soit elle est à portée de charge
- soit elle n'est ni à portée d'attaque, ni à portée de charge, dans ce cas-là on se déplace
Un adversaire est considéré à porter d'attaque d'un joueur lorsque l'adversaire est situé à une distance entre 2 et 4 de la case du joueur. Celui-ci est considéré à portée de charge lorsque l'adversaire est situé à une distance entre 4 et 5 de la case du joueur et qu'il est aligné soit horizontalement, soit verticalement avec ce joueur.

Il existe aussi une fonction de réparation qui est utilisée lorsque la valeur de la coque du joueur est comprise entre 20C et 40C. Celle-ci permet de restaurer la coque mais le joueur perd 20K.

V- Etape 3 : Equipe (Threads)

Pour implémenter un mode équipe contre équipe, on crée des threads. Pour cela, chaque processus fils va donc représenter une équipe qui va contenir les joueurs représentés par les threads. Nous ne sommes pas parvenus à achever cette partie, toutefois nous avons fait de notre mieux pour pouvoir le faire, nous avons mis quelques éléments entre commentaires dans notre code qui sont ce que nous avons commencé à faire

Conclusion

Ce projet nous a permis de développer notre capacité à travailler en équipe, à s'organiser et à concevoir un projet. Il s'agit d'un exercice très différent de ceux que nous avons pu faire en C jusqu'ici car c'est une méthode de programmation qui utilise le noyau du système d'exploitation Linux.
Cependant, nous avons rencontrés quelques difficultés notamment dans la création des tubes et dans l'implémentation des threads et des processus dans notre code. D'ailleurs, nous n'avons pas réussi à inclure les threads dans notre code, nous avons commencé une partie que l'on a laissé entre commentaires mais nous n'avons réussi à achever cette partie.

Source

BONNET Ludivine
MOLINER Emma

- Linux, Programmation système et réseau (Livre)
- OpenClassroom (Site internet pour divers sujets)
- ManPages (Site internet pour nous aider à utiliser les fonctions)