vendredi 26 avril 2024 Lou BELL

Rapport de projet Programmation Concurrente

Exercice 1 :

Dans cet exercice, j'ai choisi d'utiliser des threads pour représenter les oisillons et l'oiseau parent, un sémaphore pour contrôler l'accès au plat de nourriture, afin que tous les oisillons mangent chacun leurs tours, un à la fois, puis que l'oiseau parent remplisse le plat lorsque celui-ci est vide.

Dans le code, cela se traduit par une fonction pour chaque type d'individu, l'oisillon attend de recevoir le jeton du semaphore (que le plat soit disponible), il mange, puis se rendort et libère le jeton.

Le parent quant à lui ajoute P portions de nourriture dans le plat et attend que celui-ci se vide pour le remplir à nouveau.

Exercice 2 :

La version linéaire répondant au problème est très simple je ne vais donc pas me pencher dessus et directement aborder la suite.

Pour modéliser le problème de réduction à l'aide de forks et pipes, j'ai utilisé des threads, un processus fils pour deux valeurs du tableau, chacun est responsable de lire une valeur du tableau et de la placer dans un pipe. Le processus parent lit ensuite les valeurs des pipes, trouve le maximum local et l'envoie aux processus fils pour enfin trouver le maximum global par réduction.

Dans le code, ceci se traduit dans un premier temps par la création des pipes, et des processus fils. À leurs création, chaque fils va avoir la tâche de trouver son maximum local et de l'écrire dans son pipe attitré, puis une fois toutes les valeurs écrites dans les pipes, le processus parent va lire ces pipes et retourner la valeur max globale.