TP Système – Allocateur de mémoire

Thomas MEDARD
Antoine MERCIER-PRONCHERY

Introduction

De manière générale, la mémoire est gérée à l'aide d'une liste chaînée de bloc libres. Une fois un bloc alloué, la taille de ce bloc allouée est stockée juste avant ce bloc. Un bloc est toujours alloué à la fin d'un bloc libre.

Généralité et gestion de la mémoire

➔ Initialisation du programme

L'initialisation de l'allocateur va initialiser toutes les variables globales avec les informations sur la mémoire qui lui sont fournis.
Elle va notamment enregistrer la taille de la mémoire, son début et sa fin.

➔ Demande d'allocation

La fonction de recherche de bloc libre doit respecter certaines contraintes :

- Le bloc libre doit avoir une taille supérieur à celle demandée (qui est une addition entre la taille utilisateur et la taille d'un size_t)

- Le bloc doit soit avoir juste assez de place pour être complètement remplis, soit qui doit permettre l'existence de la structure décrivant le bloc libre.

Si la taille demandée fait la même taille que celle du bloc libre, alors la structure sera supprimée.

➔ Libération mémoire

Lors de la libération de la mémoire par l'utilisateur, notre programme va être capable de chaîner correctement, si nécessaire, la liste des blocs libres.
De plus, si l'utilisateur tente de libérer une zone déjà libre, notre programme sera capable de s'en apercevoir.

Il y a plusieurs cas particuliers que nous avons pris en compte dont la liste non exhaustive pourrait contenir :

- Libération d'une zone mémoire pleine
- Libération d'une zone accolée à une zone libre (avant et/ou après).

En effet en fonction de quel espace mémoire est libérée et de son emplacement certaines précautions doivent être prise afin d'assurer l'intégrité de la mémoire.

Ainsi notre algorithme peut correctement s'occuper de différents cas de libération de mémoire :

- Un bloc mémoire libéré entre deux blocs libres
- Un bloc mémoire libéré après un bloc libre mais juste avant un bloc alloué (aucun espace entre le bloc libéré et le prochain bloc)
- Un bloc mémoire libéré se situe après un bloc occupé et avant un bloc libre

Pour tous ces cas, des exceptions supplémentaires sont possible :

- Un bloc mémoire libéré est le seul bloc disponible de la mémoire
- Un bloc mémoire libéré se situe avant tous les autres blocs libres

Il nous a donc fallut réfléchir à ces différentes possibilités afin d'obtenir une méthode la libération de mémoire alloué la plus flexible possible.

Pour chaque cas, lors de la libération, nous recherchons au préalable le bloc précédent et le bloc suivant afin d'avoir toutes les informations pour effectuer les diverses opérations sur les pointeurs.

Parcours et recherche de la mémoire

Notre fonction de parcours mémoire se contente d'itérer à travers la liste des blocs libres après avoir parcouru les bloc alloués se situant avant la tête de liste.

À chaque itération sur un bloc libre, on sait que si ce n'est pas la fin de la mémoire, alors il est suivit d'un certain nombre de bloc occupés.

Enfin, nous montrons à l'utilisateur la taille totale stocké dans la mémoire à la fin des opérations.

Batterie de test

Test d'endurance

Ce test consiste à simuler la vie de l'allocateur mémoire dans un programme utilisateur.

Pour ça, le test va, de manière aléatoire, soit effectuer une allocation, soit effectuer une libération. Le choix est influencé par le nombre d'allocation déjà effectuées (plus on fait d'allocation et plus on a de chance de réaliser une libération).

Les tailles allouées sont aussi aléatoires.

Le nombre d'allocation à réussir est fixe.

Ce test est utile pour repérer des bugs non spécifiques qui peuvent arriver durant une longue période d'activité du programme. En revanche, nous avons du tester nous même les cas particuliers (notamment ceux décrit dans la partie sur la libération).