TP Système – Allocateur de mémoire

Thomas Medard

Antoine Mercier—pronchery

## Introduction

Ce Tp nous a permis de mieux nous familiariser avec le fonctionnement d’une mémoire interne. Nous avons mis au point un allocateur de mémoire robuste (selon nos test) capable de tenir sur la durée sans rencontrer de bug bloquant.

## Généralité et allocation de la mémoire

* Init du programe

Lors de l’allocation de la mémoire et afin d’imiter les fonctions du vrai Malloc, si la taille demandée à allouer est de zéro, notre programme retournera Null afin d’éviter les erreurs.

Si la taille est différente de Null notre programme va tenter de trouver des bloc disponible, arrondi au début a un multiple de sa structure, et, s’il n’en trouve aucun à la taille exact demandé par l’utilisateur.

* Rechain les block
* update free block si il y a encore un freeblock avant ou après

## Libération des espaces mémoire

Concernant la libération des espaces mémoire préalablement alloué, nous avons couvert la plupart des cas possible afin de recomposer des block vide valide. En effet en fonction de quel espace mémoire est libéré et de son emplacement certaine précaution 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 block mémoire libéré se situe entre deux blocs libres ;
* Un block mémoire libéré se situe après un bloc libre mais juste avant un bloc alloué (aucun espace en le bloc libéré et le prochain bloc)
* Un block mémoire libéré se situe après un bloc occupé et avant un bloc libre

Pour tous ces cas, des exceptions supplémentaires sont possible :

* Un block mémoire libéré est le seul bloc disponible de la mémoire
* Un block mémoire libéré se situe avant tous les autres blocks 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, nous stockons au préalable dans cette méthode, 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 notre structure. Lors de l'itération sur les blocks déjà occupé, notre programme retournait parfais des blocs de taille zéro. Nous les excluons donc temporairement puisqu’ils n’impact pas l’algorithme.

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

Nous n’avons pas réalisé de module supplémentaire, nous n’avons donc que la fonction fit first, occupant le premier bloc libre trouvé lors de la recherche. La fonction fit best aurait été intéressante car elle aurait permis de trouver, pour chaque allocation, la place la plus approprié dans la mémoire, réduisant ainsi les espaces de mémoires inoccupés, trop petit pour accueillir une nouvelle donnée.

## Batterie de test

### Test d’endurance

* La batterie de test (que je n’ai pas tout pigé)

### Test Initial

* Les tests du prof