KHELIFI Aziz
MACHENAUD Florian

Documentation Allocateur Mémoire :

Lors de ce projet de construction d'un allocateur mémoire, nous avions plusieurs critères et fonctionnalités à assurer. Nous allons donc ici présenter les différentes implémentations de nos fonctions, puis décrire les différents jeux de tests permettant de s'assurer du bon fonctionnement de notre allocateur mémoire.

Tout d'abord, nous allons décrire les structures présentes :

Nous avons donc 3 structures qui nous permettent la construction de notre allocateur mémoire. Tour d'abord, la structure *allocator_header* correspond au header principal de l'allocateur. Ensuite, nous retrouvons deux structures différentes : la structure *fb* correspond aux freeblocks. Elle contient une taille *size* et un pointeur vers le prochain *fb.* L'autre structure quant à elle est la structure *db*, qui correspond à la structure d'un datablock. Cette structure ne contient qu'une taille, qui correspond à la taille mémoire allouée.

Pour nos fonctions, nous avons conservé l'entièreté des fonctions données dans le fichier *mem.c.* Nous allons donc ici décrire chacune des fonctions propres à l'allocateur, c'est -à -dire la fonction d'initialisation, d'allocation et de libération.

Tout d'abord, pour l'initialisation, on récupère l'adresse de notre début d'allocateur. On vient ensuite créer un allocateur qui commence à cette adresse. On lui assigne une taille qui est récupérée depuis les arguments de la fonction. On définit ensuite un freeblock qui sera le seul de notre allocateur pour le moment. On lui donne une taille égale à la taille de l'espace mémoire, en veillant à retirer la taille occupée par la structure du header de l'allocateur. On définit l'élément suivant de ce freeblock comme étant nul, car c'est le seul freeblock de l'allocateur lors de cette fonction d'initialisation.

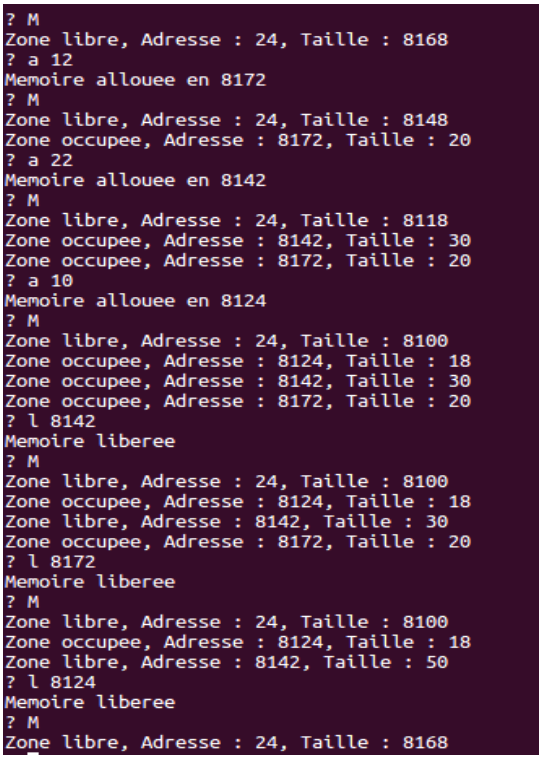
Ensuite, lors d'une allocation d'une taille choisie par l'utilisateur, on va créer un nouveau datablock, en veillant à bien le placer dans la mémoire pour qu'il puisse accueillir la taille souhaitée, ainsi que la taille du header de la structure db. une fois la taille allouée, on retourne l'adresse du nouveau datablock. Cette fonction n'est pas très longue. Avant toute allocation, on vérifie bien qu'il existe un freeblock disponible pouvant accueillir la taille passée en argument.

Enfin, pour la libération, on vérifie tout d'abord que l'adresse passée en argument correspond bien à une adresse d'un datablock. Si c'est le cas, on va tout d'abord récupérer l'adresse mémoire du dernier freeblock qui précède le bloc alloué que l'on va libérer. Cela nous servira afin de rediriger le dernier freeblock vers le nouveau que l'on va créer. Une fois ce freeblock trouvé, on crée un nouveau freeblock, qui aura comme taille l'ancienne taille du bloc alloué, en prenant bien en compte la taille du header qui va différer car la taille des structures freeblock et datablock sont différentes. Une fois la création du freeblock faite au bon endroit, on lui attribue la taille que contenait le bloc alloué. Reste à régler le problème des pointeurs sur les freeblocks. Ici, il nous suffit de définir le prochain freeblock comme étant le freeblock pointé par le dernier freeblock trouvé (celui trouvé un peu plus tôt dans l'explication). On définit ensuite le pointeur du dernier freeblock comme pointant sur le nouveau freeblock. Une fois toutes ces étapes réalisées, nous avons fini de libérer notre espace alloué. Il ne nous reste plus qu'à gérer la fusion de deux freeblocks contiguës. Pour

cela, On va parcourir les freeblocks, en vérifiant à chaque fois que l'adresse de départ + la taille du freeblock est différente de la prochaine adresse pointée par le freeblock. Si c'est le cas, alors cela signifie que le bloc suivant le freeblock est un bloc alloué. Dans ce cas, on ne touche à rien. Sinon, on modifie le freeblock pour lui ajouter la taille du freeblock suivant. Après cette modification effectuée, on détruit le freeblock suivant en lui réduisant sa taille à 0, et on récupère son pointeur pour le transférer à l'espace libre agrandi. Ainsi, il est très simple de fusionner les espaces libres de notre espace mémoire. On fait donc cette manipulation après chaque libération, pour s'assurer qu'il n'y a pas deux freeblocks qui s'enchaînent, ce qui pourrait empêcher une allocation possible.

Maintenant que les choix d'implémentation des fonctions principales ont été décrits, on peut passer aux tests :

Tout d'abord, nous allouons un bloc de taille 12, qui sera stocké à l'adresse 8172, puis un deuxième bloc de taille 22, stocké à l'adresse 8142, et enfin un troisième, stocké à l'adresse 8142. Nous pouvons remarquer d'après l'image ci-dessous que l'allocation a bien eu lieu sans erreur et que les blocs alloués sont bien placés sur le bloc total de notre allocateur mémoire.
Dans un deuxième temps, nous allons essayer de libérer certains blocs et de voir s'il y aurait bien une fusion des blocs libres.
Pour commencer, nous libérons le bloc stocké à l'adresse 8142 et aussi celui placé à l'adresse 8172. Nous remarquons d'après la même image que la libération de ces deux blocs a eu lieu sans problème, mais aussi que les blocs libérés forment bien un seul bloc libre. Donc, la fusion des blocs libérés est fonctionnelle.
Et pour plus de vérification sur la fusion des blocs, nous essayons de libérer le dernier bloc alloué qui est placé à l'adresse 8124. Nous nous retrouvons à la situation initiale, où nous n'avons qu'un seul bloc libre.



Document 1: Test de l'allocateur mémoire