AMESSEGHER AYMANE INFO3
MEMIL Dila 22/12/2023

TP4 Allocateur Mémoire

1. Les choix d'implémentation

Lors de la mise en œuvre de notre allocateur de mémoire dans le cadre du travail pratique, nous avons fait le choix d'insérer la taille de la structure fb dans la zone allouée par l'utilisateur. Lorsque l'utilisateur va utiliser la fonction mem_alloc pour allouer une certaine quantité de mémoire, notre allocateur va effectuer l'allocation en réservant l'espace requis par l'utilisateur avec un espace supplémentaire pour la structure de contrôle fb ainsi et l'alignement nécessaire. Dans notre implémentation nous utilisons une liste doublement chaînée, ce qui nous a permis de trouver plus facilement l'emplacement des zones.
Donc si l'utilisateur demande une allocation de 10, mem_alloc va en réalité réserver une zone de mémoire de taille égale à 10+A+B avec A qui représente la taille de la structure fb et B qui représente l'alignement. Mais l'utilisateur peu utiliser seulement la taille qu'il a demandé. Nous avons choisi d'implémenter notre fonction de cette façon pour pouvoir accéder aux métadonnées associées à chaque bloc de mémoire.
Ensuite pour cette fonction le but été de d'abord vérifier si la taille été valide, c'est à dire s'il n'était pas négatif, nul ou plus grand que la taille du système. Ensuite nous utilisons la fonction mem_fit de l'allocateur pour trouver le bon emplacement de la nouvelle zone qui sera occupée avec une taille T. Après avoir récupérer cette zone on regarde si sa taille est égale à T ou si elle est plus grande. Si c'est égale alors cette liste devient complètement occupée sinon la taille qui reste est inséré dans la liste chaînée au bon endroit.

La fonction mem_free est programmé pour traiter différents cas de libération. Lorsqu'un bloc à libérer se situe entre deux zones libres, la fonction effectue une fusion de ces trois blocs. La liste doublement chaînée offre un accès aisé aux blocs précédent et suivant, facilitant ainsi la fusion complète de l'espace libéré avec les zones adjacentes.
Si le bloc à libérer se trouve entre deux blocs déjà occupés, la fonction libère uniquement le bloc concerné sans effectuer de fusion.
Lorsqu'une zone libre précède le bloc à libérer, on va fusionner le bloc libéré avec la zone libre précédente. Et enfin si une zone libre suit le bloc à libérer, la fonction fusionne le bloc libéré avec la zone libre suivante.
La fonction mem_free a été conçue pour traiter uniquement les blocs de mémoire préalablement alloués par la fonction mem_alloc. Si l'utilisateur tente de libérer une zone déjà libre ou une adresse n'appartenant pas à un bloc alloué, la fonction mem_free détecte ce cas particulier et envoie un segmentation fault ou arrête le programme . Cela garantit que seuls les blocs de mémoire validement alloués sont soumis au processus de libération.

2. Le résultat obtenu

Nos fonctions mem_init et mem_show fonctionne correctement. La fonction mem_show nous a beaucoup aidé dans la compréhension du code. Grace à cette fonction nous avons pu rajouter des cas que nous n'avions pas pris en compte dès le début pour nos fonctions mem_alloc et mem_free.

Pour la partie des tests nous avons testé tous les cas possible que nous avons cité ci-dessus pour la fonction mem_free et on obtenait le bon résultat à chaque fois. Tandis qu'à la fonction mem_alloc ce n'est pas pareil, nous l'avons changé plusieurs fois. La fonction mem_alloc fonctionne bien quand la zone_occupée se trouve au début de la liste chaînée. Mais quand on doit mettre la zone occupée entre deux blocs occupées notre fonction envoie des résultats différents. Malheureusement nous n'avons pas pu résoudre ce problème. Mais notre allocateur mémoire fonctionnera quand même si on mettait des règles spécifique en plus qu'il faudrait respecter des le début pour la fonction mem_alloc.

3. Les tests

Nous avons mis en place deux tests automatique pour pouvoir tester nos deux fonctions mem_alloc et mem_free avec bien-sûr mem_init et mem_show qui va nous afficher le résultat pour chaque cas.
Pour le test de l'allocation nous avons testé avec plusieurs cas, tout d'abord avec une taille négative ou trop grande, la fonction envoie Null car ce n'est pas possible. Ensuite on essaye d'allouer plusieurs bloc consécutif l'un après l'autre de différents taille pour voir si la fonction fonctionne correctement.
Ensuite pour la fonction mem_free nous avons mis en place des tests qui vérifie chaque cas, c'est à dire quand la zone occupée se trouve au début ou à la fin ou au milieu. Nous avons testé les différents cas pour voir si les fusionnement fonctionné bien. Nous avons testé avec des adresse fausse ce qui envoyé des segmentation faults, ce qui est attendu.