Brunin Thomas
Pernet Alexis
groupe 3

Implémentation de la ListeChaine

Pour créer la liste chaînée circulaire, j'ai créer une structure qui a comme attribut :

-struct ListeChaine* next; qui est un pointeur vers le prochain bloc
-struct m_petitBloc* tab; qui vaux NULL par défaut mais il peut contenir un tableau de "m_petitBloc" utilisé pour les petit allocation avec mallopt
-char utilise; pour savoir si le bloc est libre (utilise vaut 0), utilise (utilise vaut 1) ou encore, si le bloc est un tableau de "m_petitBloc", il peut également être rempli, dans ce cas, utilisé vaut 2
-void* adresse; l'adresse a laquelle le bloc de mémoire commence

Pour la structure des petits bloc (m_petitBloc), c'est plus simple, il n'y a que 2 attributs :

-char utilise; pour savoir si la case est utilisée ou non
-void* adresse; l'adresse du début de la case

Informations supplémentaires sur les fonctions

ListeChaine* rechercheListeParTaille(size_t size)
concernant la condition "if(listeEnCours->taille >= size && listeEnCours->taille <= (size+sizeof(ListeChaine)) && listeEnCours->utilise == FALSE)"
j'ai choisi de faire que, si l'utilisateur demande à allouer un bloc qui, quand divisé, n'a plus la place restante pour créer une nouvelle ListeChaine, alors on renvoie directement le bloc entier, même si cela veux dire qu'il est possible de donner un peu plus d'espace a l'utilisateur que le nombre demandé

void* mrealloc(void* base,size_t size)
On commence par un "free" pour ensuite refaire un malloc. j'ai décidé de faire cela afin de "simplifier" le code, en effet, de cette manière, une fois le free et la fusion des bloc effectués, c'est le malloc qui va se charger de retourner un bloc de taille adapté, avec une chance que malloc retourne le même bloc mais avec une taille différente

void mmallopt(int cmd, unsigned int val)
j'ai utilisé une variable globale "utilisationPossibleMallopt" qui se met à FALSE dès l'allocation d'un "petit bloc". Ce n'est pas la méthode la plus élégante, mais elle a le mérite d'être simple et rapide

Remarques supplémentaire

Je n'ai pas créer de fonction "enlever_bloc" car je ne voyais pas son utilisé, étant donné qu'il est préférable de fusionner des bloc plutôt que de les supprimer

Améliorations apportées

- la fonction free détecte si une adresse qui lui a été rendue ne correspond à aucun bloc mémoire, en affichant simplement un message
  - A chaque allocation de mémoire, il y a 4 char rajoutés en fin de bloc afin de détecter les éventuels dépassement de mémoire qui seront détecté lors de l'appel a free, dans ce cas, j'ai pris la liberté de seulement affiché un simple message

Détails des tests

TEST MALLOC ET FREE SIMPLE : On fais un simple malloc qu'on libère instantanément, on peut bien voir que, en demandant un bloc de taille 40, malloc nous donne un bloc de taille 44, correspondant au 4 char en fin de chaque segment. Et on remarque également qu'après l'appel à free, les 2 blocs fusionnent

TEST MALLOC ET FREE PLUS COMPLEXE : On fais plusieurs malloc a la suite qu'on libère chaque leur tour ensuite, le test est concluant étant donné qu'on retrouve bien notre bloc initial de taille 100 après tous les free

TEST LIBERATION D'UNE ADRESSE INVALIDE : On fais un appel a free sur l'adresse 0, et celui-ci nous envoie un message d'erreur pour nous informer que cela ne correspond a aucune adresse retournée par malloc

TEST DEPASSEMENT DE MEMOIRE : On alloue un bloc de taille 3, et on en utilise 4 (car un int est de taille 4), une fois que l'on fais appel a free, ce dernier nous informe que l'on a dépassé la taille autorisée

TEST CALLOC ET REALLOC : On fais appel a calloc pour créer un tableau de 30 int. Après l'affichage, on remarque que calloc a bien initialisé toutes les cases à 0
On remplis ensuite ce tableau et on décide de modifier la taille à 10 cases uniquement grâce a un realloc.
Après la réallocation, on peut voir que les 10 premières cases ont conservé leur valeur, et que le bloc n'est plus que de taille 44 (10 int + les 4 char en fin de segment)
Ensuite, on augmente la taille à 20 cases à nouveau grâce a un realloc, et on remarque que les 10 première valeur sont toujours conservé, mais les 10 suivantes ne sont plus les même qu'au début (normal) et que le bloc est bien de taille 84 (20 int + les 4 char en fin de chaque segment)

TEST MALLOPT : On met les valeur de maxfast à 10 et numblks à 50
Ensuite, on fait 50 malloc de taille 10, qui va normalement remplir totalement le tableau de petit bloc, qu'on vérifie en imprimant l'entête
Ensuite on libère un petit bloc pour voir si la libération fonctionne, ce qui marche parfaitement
Après, on essaye de modifier la valeur de maxfast, qui ne marchera pas étant donné qu'un tableau de petit bloc a déjà été alloué, cela affiche un message d'erreur et la valeur de maxfast n'a pas été modifiée.
Enfin on fais 2 malloc supplémentaires de taille 10 et on remarque qu'un nouveau tableau de petit bloc a étais créé.