Lombard Myriam
Nguena Gloria

TD & TP Allocation Mémoire

Notre programme mem.c ne fonctionne pas, il réalise une erreur de segmentation dans notre fonction mem_alloc(size_t taille), nous rencontrons encore de nombreux bugs qui ne sont pas encore fixés. En revanche, la fonction mem_show fonctionne. On peut l'observer en lançant l'exécutable memshell.

Nous avons choisi d'utiliser une variable globale représentant notre liste chaînée de blocs libres appelée first_fb. Nous l'initialisons dans la fonction mem_init(void * mem, size_t taille). C'est le début du bloc mémoire, elle possède sa taille.

Une erreur de segmentation de notre programme se situe dans la fonction *mem_alloc(size_t taille), à la ligne 106. Nous avions prévu de l'implémenter ainsi :

-En début de fonction, on calcule la taille de la zone qui va être occupée dans notre zone mémoire en ajoutant les métadonnées et en alignant le tout.

-On place la liste des zones libres dans une nouvelle struct fb* que l'on va manipuler à notre guise

-On crée une variable afin de récupérer le résultat de mem_fit(). Cette fonction renvoie soit NULL si elle n'a rencontré aucun bloc libre qui corresponde à la taille recherchée, soit un pointeur vers le bloc correspondant le cas échéant.

-Si cette allocation échoue, on renvoie NULL

-Sinon, on se retrouve confronté à deux cas :

- Le bloc Libre contient assez de mémoire pour séparer ce bloc en un bloc occupé et en un bloc libre. Dans ce cas-ci, on sépare le bloc libre en un bloc occupé de la taille requise par l'utilisateur. La mémoire restante de ce bloc libre sera allouée à un nouveau bloc Libre, qui sera rajouté à la liste chaînée des blocs libres. On renvoie un pointeur vers le bloc alloué.
- Le bloc Libre contient juste assez de mémoire pour la taille requise par l'utilisateur. On alloue cet espace au bloc occupé et on le renvoie à l'utilisateur après l'avoir retiré de la liste chaînée des blocs libres. On renvoie un pointeur vers le bloc alloué.

Nous n'avons pas implémenté la fonction mem_free(void *mem) car le reste ne fonctionnait pas, mais nous l'imaginons ainsi :

-On parcourt la liste chaînée de zone libre. On compare l'adresse des pointeurs vers la zone libre avec l'adresse de la zone mémoire donnée en paramètre. Tant que l'adresse de la zone libre est plus petite ou que l'adresse suivante dans la liste est nulle, on boucle en passant à l'élément suivant de la liste. Dans cette boucle, on stocke dans une variable la valeur de la zone libre précédente.

-Une fois sortis de cette boucle, on regarde si l'adresse de la zone libre + sa taille est égale à notre adresse *mem. Si oui, nous sommes dans le cas de deux zones libres côte à côte dans la mémoire, il faut donc les fusionner.

-Ensuite, on regarde si la prochaine zone Libre est côte à côte avec notre zone mémoire libérée. Si oui, on fusionne les deux zones libres pour n'en former plus qu'une.

-Si on n'est rentrés dans aucune des boucles précédentes, on libère tout simplement la zone mémoire en créant un nouvel élément dans la liste chaînée des zones libres :

```
else
{
        newZL->size = mem->size;
        newZL->next = liste->next;
        liste->next = newZL;
}
```