Rapport du projet AISE

Réalisé par :
AAJLI Hamid
AZZAZ Hichem

Introduction

Le projet consiste à implémenter un ensemble de fonctions de gestion de la mémoire, à savoir l'allocation, la libération et la réallocation(malloc, free, realloc, calloc) des block mémoire en se basant sur des fonction système comme mmap et nmpam.

Structure du projet :

Le projet comporte 3 principaux fichiers :

Makefile : qui sert à automatiser la création de la bibliothèque dynamique, les tests et l'utilisation des fonctions dans un navigateur Web (Firefox dans notre cas). Il sert aussi à nettoyer le répertoire du projet

malloclib.c : est la bibliothèque qui regroupes l'ensemble de fonction pour l'allocateur mémoire : malloc, free, realloc, calloc; avec d'autres fonctions utiliser au sein de ces derniers tel que getbin, rechMem et mmapfct.

Malloclib.h : présente le fichier entête de la bibliothèque, avec les signature des fonction, la structure d'informations des blocks mémoires, des initialisation de variables globales ainsi qu'un mécanisme d'exclusion mutuelle.

Analyse de l'implémentation des fonctions :

rechMem :

```
void * rechMem(size_t size)
```

Cette fonction offre le block mémoire qui convient à la taille demandée.

Signature de la fonction :

```
void * rechMem(size_t size)
```

size_t size : la taille du block demandée

void * : le pointeur vers la structure strInformation

Algorithme :

1) Chercher dans la liste des str_Information (comportant des inforamtion sur les blocks mémoire libres) , le nœud dont sa taille qui convient le plus à la valeur size.
2) Une fois le nœud est trouvé, il sera supprimé de la liste des nœuds libres.
3) L'adresse du best fit est retourné par la fonction

mmapfct :

```
void * mmapfct(size_t size)
```

La fonction fait une allocation dans l'espace physique et retourne l'adresse réelle du bloc alloué

1) Incrémenter la taille totale que le thread a alloué en utilisant une exclusion mutuelle (la taille totale allouée est une variable critique).
2) Calculer le nombre de page correspondant à la taille demandée en divisant le paramètre size par la taille d'une page.
3) Allouer le nombre de page essentielle pour la taille demandée en utilisant la fonction mmap()
4) Une instance de la structure strInformation est rempli par la taille allouée et elle est copier au début du bloc mémoire physique en utilisant la fonction mmcpy()
5) L'adresse du début de la mémoire allouée (adresse début + déplacement d'une taille de la structure strInformation) est retournée par la fonction.

allocMem :

```
void *allocMem(size_t size)
```

La fonction cherche la bloc mémoire le plus adéquat avec la demande à travers la fonction mmRech() , ensuite fait son allocation grâce à la fonction mmapfct().

malloc() :

```
void* malloc(size_t size)
```

La fonction alloue un espace définie par le paramètre size en entrée et retourne une adresse de l'espace alloué si la size est un entier positif.
Un compteur global est utilisé en exclusion mutuelle afin de compter le nombre d'allocations faites. La fonction est juste utilisée comme interface qui utilise la fonction déjà présentée allocMem().

free() :

```
void free(void *memF)
```

La fonction a comme entrée l'adresse mémoire à libérer. Elle ajoute l'instance de strInformation lié au bloc adressé par memF à la liste des nœuds libres.

realloc() :

```
void *realloc(void *memRe, size_t size)
```

La fonction a comme paramètre memRe qui présente l'adresse mémoire déjà allouée ainsi que size, la taille souhaitée. Elle fait que libérer l'espace adressé par memRe et allouer un nouveau convenable à la taille demandée.
Elle retourne une nouvelle adresse de la mémoire allouée.

calloc() :

```
void *calloc(size_t nmemb, size_t size)
```

size_t nmemb: nombre de blocs contiguës à allouer
size_t size : la taille de chacun des blocs (ils ont la même taille)
La fonction utilise malloc() pour allouer un espace regroupant les nmemb blocks (taille totale = size*nmemb) et retourne l'adresse du premier bloc alloué .

Test du lib:

Notre lib il marche avec tous les programmes et les commandes linux et il peut exécuter firefox sans problème.
Avec un simple make dans le dossier du projet et :

```
#LD_PRELOAD=`pwd`/malloclib.so "prog Volu"
```

le seul problème qu'on a c'est que on peut pas afficher les valeurs de notre malloc.
On a fait une fonction pour faire l'affiche de nos stats mais ça nous a posé des problèmes est c'est pour ça qu'on a eu ce retard on a pensé que on va la régler mais malheureusement on a pas pu et on va continuer à travailler sur le projet après la correction pour qu'on le rend plus efficace.