Thomas LERCHUNDI
Mary-Andréa RAKOTOMANGA M1 Informatique

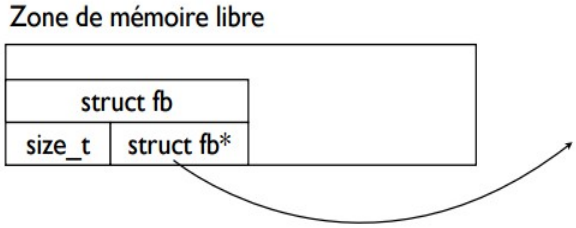
# Compte-rendu : TP – Allocation mémoire

*L'objectif de ce TP a été de réaliser un gestionnaire d'allocation mémoire.
Dans un premier temps, nous allons décrire la représentation de la mémoire que nous avons considéré ainsi que la structure choisie pour la gestion. Ensuite nous présenterons les fonctionnalités et les limites du code proposé. Nous finirons par une présentation des extensions apportées.*

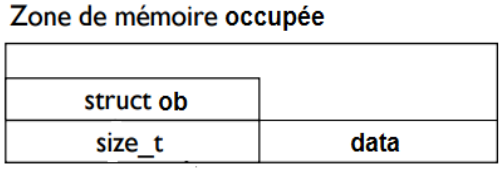
## Présentation de la structure pour la gestion de la mémoire

La mémoire est représentée par un tableau de taille fixée.
Pour la gestion de la mémoire, on distingue les zones utilisées, ou allouées, et les zones libres qui sont disponibles pour le système ou l'utilisateur qui demande une allocation. Dans notre structure, les zones libres sont représentées par une liste chaînée. Il y a un pointeur qui contient l'adresse vers la première zone libre, puis chaque zone libre a pour attribut sa taille et un lien de chaînage vers la zone libre suivante.

Les zones libres sont rangées par adresses croissantes dans la liste chaînée.




Chaque zone allouée est associée à un descripteur qui contient sa taille totale dans la mémoire.




Il n'est pas nécessaire d'avoir également une liste chaînée pour les zones occupées.
Pour savoir si une zone est occupée ou libre en parcourant la mémoire, il suffit de comparer son adresse avec celles présentent dans la liste chaînée des zones libres. Pour avancer, on utilise la taille de la zone.
A l'état initial, toute la mémoire est disponible. La liste chaînée des zones libres contient un élément ayant pour attribut la taille totale du tableau de mémoire.
Pour trouver une zone d'une certaine taille, on utilisera une fonction recherche qui renvoie l'adresse d'une zone libre assez grande selon l'algorithme utilisé, un pointeur nul sinon.

## Allocation d'une zone mémoire

Le gestionnaire permet d'allouer une zone d'une taille donnée quand il y a assez de place de disponible.
S'il n'y a pas de zone libre assez grande, la fonction de recherche renvoie un pointeur nul. Sinon la zone libre renvoyée devient une zone occupée. Il faut ensuite également mettre à jour la liste chaînée des zones libres.
On considère deux cas :

- la zone libre à allouer correspond au premier élément de liste chaînée :
Il faut mettre à jour l'adresse du pointeur du début de la liste chaînée des zones libres.

- la zone libre à allouer se situe après le premier élément de liste chaînée :

On récupère l'adresse de la zone libre précèdant celle que l'on va allouer pour pouvoir mettre à jour son lien de chaînage vers la prochaine zone.

Dans les deux cas, on vérifie s'il y a suffisamment de place pour créer une nouvelle zone libre après la nouvelle zone allouée. On aligne également les adresses si nécessaire. La fonction d'allocation renvoie ensuite l'adresse en la décalant de la taille de la structure d'une zone occupée.

## Libération d'une zone mémoire

Il est possible de libérer une adresse donnée de zone allouée dans la mémoire. Pour récupérer la structure de zone occupée associée, on recule l'adresse de la taille de la structure d'une zone occupée.
Plusieurs cas sont considérés :
-la zone occupée se trouve avant la première zone libre :
On vérifie si l'adresse de fin de la zone occupée correspond à l'adresse de la première zone libre. Si c'est le cas on les fusionne.
- la zone occupée se trouve après l'ensemble des zones libre :
On vérifie si l'adresse de la zone occupée correspond à l'adresse de fin de la dernière zone libre. Si c'est le cas on les fusionne.
-la zone occupée se trouve entre deux zones libres :
Même vérification que les deux au-dessus, avec la zone libre précédant la zone occupée à la place de la première zone libre et la zone libre suivant la zone occupée à la place de la dernière zone libre.
S'il n'y a pas de fusion, on crée une nouvelle zone libre à partir de la zone occupée en ajoutant les liens de chaînage correspondants.

## Réallocation d'une adresse de zone allouée

Pour remplacer le **malloc/free** standard avec notre implémentation, une fonction indiquant le nombre d'octets disponibles pour une adresse déjà allouée était nécessaire pour faire de la réallocation.
Pour avoir toute la taille disponible pour une adresse donnée, il faut regarder s'il y a une zone libre après la zone occupée. Si c'est le cas, il suffit de faire la somme des tailles des deux zones, sinon le nombre d'octets disponibles sont uniquement ceux déjà alloués.

## Extensions/ Test :

Les fonctions de recherche disponibles dans le gestionnaire d'allocation sont :
-Fit First :
Recherche de la première zone libre assez grande.
-Fit Best :
Recherche de la plus petite zone libre assez grande.
-Fit worst :
Recherche de la plus grande zone libre assez grande.

Nous avons ajouté un programme qui teste les differentes stragégies d'allocation (best fit et worst fit). Le test fait une série aléatoire de 50 allocations/libérations pour chacunes des stratégies.

Remarque: make tests lance successivement : (./memtest ; echo «... » ; cat Makefile et ./fittest)

## Limites

On suppose que l'utilisateur a une bonne utilisation du système. Notre code ne gère pas le Multithreading, bien que nous avons essayé d'utiliser des verrous d'exclusion mutuelle.