

LINFO1252 - SYSTÈMES INFORMATIQUES / 2024-2025

---

**Rapport Projet 0  
Allocation Dynamique de Mémoire  
Groupe 26**

---

*Auteurs :*

EPHRAIM TSHIEBUE - 2334  
2100  
ARTHUR LOUETTE - 5108 2100

*Professeur :*  
ÉTIENNE RIVIÈRE

# 1 Détails de l'Algorithmme utilisé

## 1.1 Structure de Donées utilisées

L'algorithme d'allocation de mémoire dynamique développé par nos soins utilise une liste doublement chaînée pour représenter la mémoire libre et allouée. On utilise une structure appelée *Block* pour représenter chaque élément contenu dans notre simulation de heap MY\_HEAP. Nous avons choisi d'implémenter une liste doublement chaînée car cela permet notamment d'insérer et retirer facilement des blocs de mémoire, fusionner des blocs adjacents, ainsi que d'itérer facilement vers l'avant ou l'arrière sur les blocs de la mémoire.

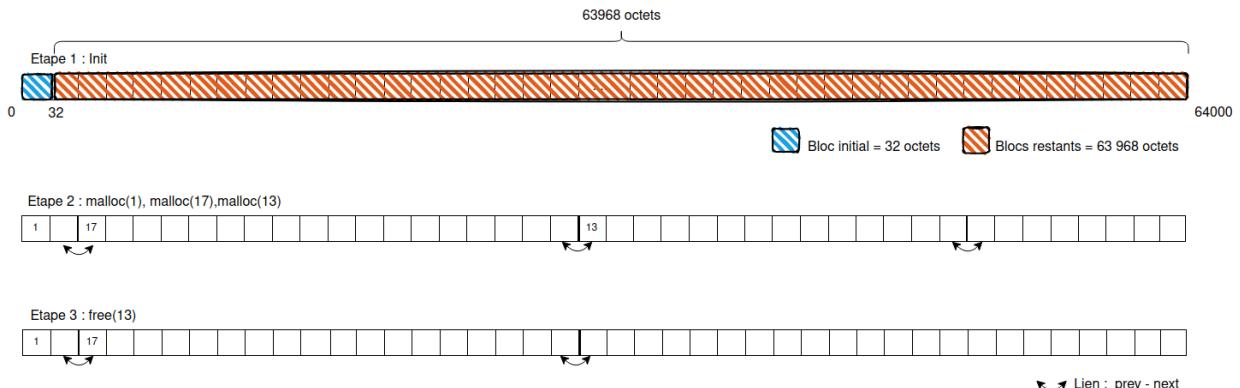
## 1.2 Métadonnées utilisées

La structure Block est composé de 4 champs distincts :

- `is_free` : indicateur (booléen) qui indique si le block est alloué ou libre. Le bloc est considéré comme libre si `is_free` vaut 1 et 0 dans le cas contraire.
- `size` : la taille en octets du Block, de type `size_t`.
- `prev` : pointeur de type `Block` qui pointe vers le bloc précédent.
- `next` : pointeur de type `Block` qui pointe vers le bloc suivant.

Ces métadonnées permettent de suivre l'état de chaque bloc dans la liste, de gérer efficacement les allocations et les libérations de mémoire, et de réaliser des opérations de fusion pour les blocs libres adjacents, optimisant ainsi l'utilisation de l'espace dans MY\_HEAP.

## 2 Schémas d'Exemple



Les 3 schémas ci-dessus représentent un exemple précis donné par le scénario suivant :

1. `init()`
2. `malloc` un bloc de taille 1 octet
3. `malloc` un bloc de taille 17 octets
4. `malloc` un bloc de taille 13 octets
5. `free` le bloc de 13 octets

Dans le premier schéma, on y retrouve un bloc de 32 bits, qui est la taille des métadonnées initiales de la structure Block. Les blocs restants représentent les 63 968 octets restants libres. Dans le second schéma, nous voyons les Blocks de 32 octets avec les tailles respectives allouées par la fonction `my_malloc()` et ensuite le nombre de blocs d'octets correspondant. On a respectivement, 1 bloc, 17 blocs et 13 blocs.

Dans le dernier, le bloc comportant la taille 13 et les octets correspondants sont supprimés. Le lien `prev - next` est donc réattribué entre les blocs correspondant à `my_malloc(17)` (`prev`) et l'ensemble des blocs libres restants (`next`).