Membre groupe :

Tidiani Ba Diop

Kemo Diaite

Abdou Aziz Diouf

Guide du développeur

1.Structure de donnée et variables globales

a.Structure de donnée

```
struct list{

    int taille;
    void *debutAlloc;
    void *finAlloc;
    struct list *allocSuivant;
};

typedef struct list listeAllocation;
```

taille correspond à la taille de l'allocation.

debutAlloc est un pointeur sur le début de l'allocation.

finAlloc est un pointeur correspondant à l'adresse de fin de la zone allouée.

allocSuivant est un pointeur sur l'allocation effectué par la suite.

b.Variables globales

```
listeAllocation *listeAllocations ;
```
Est la liste nous qui nous permet de gerer les allocations sur notre zone de mémoire .

```
listeAllocation *espaceLibres ;
```
Est la liste qui nous permet de gerer les espaces libres suite à une suppression dans notre zone.

```
void *espaceMemoireDebut;
```
Est le pointeur de debut de notre zone de mémoire fourni par la fonction initMemory().

```
void *espaceMemoireFin;
```

Est le pointeur de fin de notre zone mémoire fourni par la fonction initMemory() .

2. Algorithme Allocation et désallocation

a)Allocation

La mémoire étant organisé en octet, chaque adresse (pointeur) pointe sur un octet.

De plus l'addition d'un pointeur avec un entier donne un pointeur sur une adresse p incrémentée de i * sizeof(*p*).

Cas 1 : Aucune allocation n'a été faite

La première allocation aura comme adresse de début le pointeur espaceMemoireDebut et la fin de la zone allouée sera l'adresse de début + taille à allouer en octet pour pointer sur l'adresse se trouvant n octet après.

Cas 2 : une ou des allocations ont déjà été faites

La nouvelle allocation se fait à la suite de la dernière allocation.

L'adresse de début de la nouvelle allocation sera l'adresse fin de la dernière allocation +1.

Et l'adresse de fin sera l'adresse de début de l'allocation + taille à allouer en octet pour pointer sur l'adresse se trouvant n octet après.

Cas 3 : une ou des allocations ont déjà été faites et l'adresse de fin de la nouvelle allocation dépasse notre zone de mémoire

Dans ce cas on vérifie si de l'espace a été libérée dans les allocations précédentes, si oui on recherche le plus petit bloc pouvant recueillir la nouvelle allocation et on effectue l'allocation. Si de l'espace n'a pas été libéré ou que parmi les espaces libérées aucune ne peut contenir la nouvelle allocation, un message d'erreur est notifié à l'utilisateur.

NB : A chaque allocation, une vérification est faite pour vérifier qu'on ne dépasse pas notre zone de mémoire initiale.

b)desallocation

On recherche et on vérifie si l'adresse à libérer n'est pas NULL, si oui on libère la zone et on place cette mémoire libérée dans une liste tampon nous permettant de faire la gestion de l'espace libre.