TP SYSTÈME : ALLOCATION DE MÉMOIRE

1. Allocation de mémoire

Le système que nous nous proposons d'étudier possède un espace mémoire de taille fixée à l'initialisation. Cet espace correspond à la mémoire physique utilisable par le système ou par l'utilisateur.

Le problème qui se pose est de fournir un mécanisme de gestion de la mémoire.

La gestion de la mémoire signifie:

- connaître les zones de mémoire utilisées ainsi que celles libres, c'est-à-dire disponibles pour le système ou l'utilisateur lorsqu'ils en demandent l'allocation,
- allouer des zones de mémoire lorsque le système ou l'utilisateur les demande,
- libérer des zones de mémoire utilisées lorsque le système ou l'utilisateur n'en a plus besoin.

2. Mise en œuvre : chaînage des zones libres

Un algorithme de gestion de la mémoire repose sur le principe de chaînage des zones libres. À chaque zone libre est associé un descripteur qui contient sa taille et un lien de chaînage vers la zone libre suivante. Ce descripteur est placé dans la zone de mémoire libre elle-même.

Le principe de l'algorithme est le suivant:

Soit à trouver une zone de taille *tailleZone*. On explore la liste des zones libres; si on note *z* la zone courante, des critères de choix possibles sont les suivants:

- ***Première zone libre (first fit)***: on choisit la première zone *z* telle que
  taille(z) >= *tailleZone*. Ce choix vise à accélérer la recherche.
- ***Meilleur ajustement (best fit)***: on choisit la zone z donnant le plus petit résidu; autrement dit, on choisit *z* telle que *taille(z)* - *tailleZone* soit minimal, ce qui impose de parcourir toute la liste ou de la maintenir classée par tailles.
- ***Plus grand résidu (worst fit)***: on choisit la zone *z* telle que
  taille(z) - *tailleZone* soit maximal.

Lors de la libération d'une zone, celle-ci est réinsérée dans la liste et fusionnée, s'il y a lieu, avec la ou les zone(s) voisine(s). Cette fusion est facilitée si les zones sont rangées par adresses croissantes.

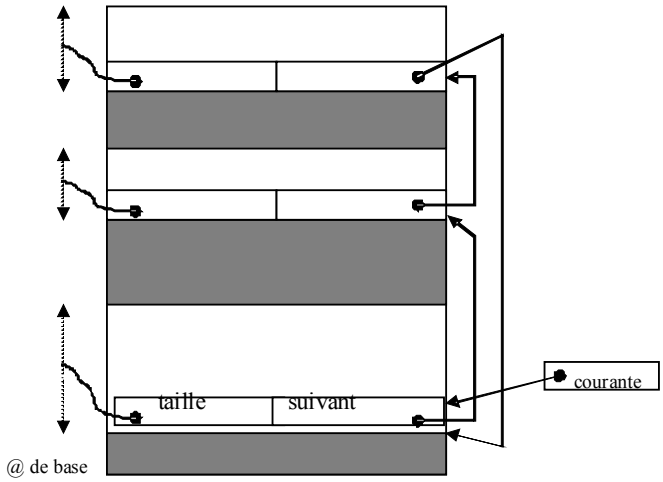

Figure 1 Représentation des zones libres chaînées

3. TRAVAIL DEMANDÉ

Il est demandé de réaliser le gestionnaire d'allocation mémoire avec l'algorithme de chaînage des zones libres dans l'ordre des adresses croissantes et le critère *«first fit »*. On déclare dans le programme un tableau de caractères de nom `mem_heap` et de taille `HEAP_SIZE` :

```
#define MAX_INDEX 20
#define HEAP_SIZE (1 << MAX_INDEX)
char mem_heap[HEAP_SIZE] ;
```

Au départ, la liste libre sera constituée d'une seule zone libre correspondant à l'ensemble du tableau `mem_heap`. La liste libre évoluera au fur et à mesure des allocations, mais les zones libres et occupées seront toujours à l'intérieur du tableau `mem_heap`. Le descripteur d'une zone libre, placé au début de la zone, peut être déclaré en C par :

```
struct fb {
    size_t size ;
    struct fb *next ;
    } ;
```

Le type `size_t`, défini dans la bibliothèque `stddef.h`, est synonyme de `unsigned long`.

L'allocateur comportera les fonctions suivantes :

```
void mem_init();
```

Cette procédure initialise la liste des zones libres avec une seule zone correspondant à l'ensemble du tableau mem_heap. Lorsqu'on appelle `mem_init` alors que des allocations et libérations ont déjà été effectuées, l'ensemble des structures de données est réinitialisé.

```
void *mem_alloc(size_t size);
```

Cette procédure reçoit en paramètre la taille `size` de la zone à allouer (cette taille est arrondie au plus petit multiple de la taille du descripteur de zone supérieur ou égal à `size`). Elle retourne un pointeur vers la zone allouée et `NULL` en cas d'allocation impossible.

```
void mem_free(void *zone, size_t size);
```

Cette procédure reçoit en paramètres l'adresse `zone` et la taille `size` de la zone à libérer. Elle met à jour la liste des zones libres avec fusion éventuelle des zones libres contiguës.

```
void mem_show(void (*print)(void *zone, size_t size));
```

Cette procédure reçoit en paramètre l'adresse d'une procédure `print` ayant elle-même deux paramètres, l'adresse d'une zone et une taille. Cette procédure `print` doit être utilisée pour afficher à l'écran la spécification d'une zone libre.

Exemple : soit la procedure

```
void imprimer (void *zone, size_t size)
    {
    printf("0x%X 0x%X\n", zone,size) ;
    }
```

On pourra appeler `mem_show` par `mem_show(&imprimer) ;`

Ces fonctions seront réalisées dans un fichier `alloc.c` dont l'interface `alloc.h` vous est donnée. On vous donne en outre, sous forme de code objet, un langage de commande dans le fichier `memshell.o`. Enfin, un fichier `makefile` permet de construire un fichier exécutable réunissant votre allocateur et le langage de commande. Les commandes disponibles sont les suivantes :

1) init: initialisation ou réinitialisation de l'allocateur

La taille peut être en décimal ou en hexadécimal (préfixe 0x)

2) alloc <taille> : allocation d'un bloc mémoire

La taille peut être en décimal ou en hexadécimal (préfixe 0x) ; alloc affiche en retour un identificateur de bloc et une adresse.

3) free <identificateur> : libération d'un bloc

4) show : affichage de la liste des blocs libres

5) help : affichage de ce manuel

6) exit : quitter le shell

Par défaut, au lancement, le shell appelle mem_init(MEM_HEAP) ; la constante MEM_HEAP est égale à 4096.