Travaux Dirigés n°6

Cours de Langage C

—Licence Informatique—

Ce TD porte sur l'allocation dynamique de mémoire. Nous voyons tout d'abord comment gérer une pile avec une allocation par bloc puis nous envisageons notre propre gestion de la mémoire pour limiter les inconvénients d'un nombre trop important d'appels à `malloc` et `free`.

▸ **Exercice 1.** (gestion d'une pile) *On souhaite gérer une pile, c'est-à-dire une structure de données de type LIFO (Last In First Out), dont la taille n'est, a priori, pas bornée. Pour optimiser l'allocation mémoire, on décide de représenter la pile par un tableau de cellules, alloué dynamiquement par bloc de* `TAILLE` *cellules. Pour cet exercice, on souhaite stocker un entier dans chaque cellule de notre pile :*

1. *Définir un type* `pile` *représentant une pile d'entiers.*
2. *Donner la fonction* `push` *permettant d'empiler un élément.*
3. *Donner la fonction* `pop` *permettant de dépiler l'élément du sommet de pile.*

▸ **Exercice 2.** (gestion d'une file) *Comment résoudre le problème dans le cas de la gestion d'une structure de type file, i.e LILO (Last In Last Out) ?*

▸ **Exercice 3.** (gestion améliorée de la mémoire) *L'inconvénient de la gestion de la mémoire utilisée dans l'exercice précédent vient du fait que l'on passe son temps à allouer et libérer la mémoire. Bien sûr, cette opération concerne des blocs de* `TAILLE` *cellules et l'on peut espérer ainsi réduire le nombre d'allocations et libérations perpétuées au cours d'une exécution.*

Pourtant, il peut être intéressant de limiter ce nombre d'allocations et de libérations en remarquant que les programmes passent en général leur temps à allouer et libérer des zones de même taille : d'ailleurs dans le cas de l'exercice précédent, ce phénomène est flagrant. La solution consiste à gérer sa propre mémoire afin de limiter les appels à `malloc` *et* `free` *par exemple :*

1. *Définir un type* `mémoire` *permettant de gérer sa propre mémoire.*
2. *Définir une fonction* `Malloc` *qui remplacera la fonction* `malloc`*.*
3. *Définir une fonction* `Free` *qui remplacera la fonction* `free`*.*
4. *Modifier en conséquence les fonctions* `push` *et* `pop`*.*
5. *Quel est l'inconvénient d'une gestion personnelle de la mémoire ?*