Cours n° B.4

Allocations mémoire

Problèmatique

La réservation d'espace via les définitions de variables (simple ou tableau) est appelée **allocation statique** :

la taille mémoire réservée n'est pas modifiable
au cours de l'exécution du programme.

Pour utiliser de l'espace mémoire dont on ne connait pas la taille avant le début de l'exécution du programme on doit trouver une solution.

La solution naïve c'est de :

- Réserver **statiquement** une grande zone
- Distribuer et Redistribuer **dynamiquement** des bouts de cette zone au cours de l'exécution

Allocation dynamique

La solution naïve a des inconvénients :

- ☞ sous-utilisation de la mémoire
  les limites de l'allocation statique s'applique et toute la mémoire de l'ordinateur (ou du processus) n'est pas utilisable (seul le système a accès à toute la mémoire de la machine)
- ☞ mauvaise utilisation de la mémoire
  toute la mémoire réservée est contigue, des trous peuvent se former et ne plus être utilisé (fragmentation)

Les systèmes d'exploitations modernes (au moins ceux respectant la norme POSIX) offres des fonctions d'**allocation mémoire dynamique** :

- ☞ l'espace mémoire n'est pas réservé avant l'appel système correspondant
- ☞ la taille de l'espace mémoire réservée peut-être modifiée au cours de l'exécution du programme sans perte de données
- ☞ le système gère seul la mémoire réservée (problème d'alignement, etc.)
- ☞ les problèmes de fragmentation sont toujours présents (mais en général limités par des algorithmes d'allocation efficaces)

➠ Fonctions de la bibliothèque standard du C (`stdlib.h`)

malloc

```
void * malloc (size_t size)
```

- ☞ Demande au système de réserver `size` octets contigus en mémoire
- ☞ Retourne l'adresse du début de la zone ou `NULL` si le système n'a pas pu trouver la place disponible
- ☞ Le pointer doit être converti en pointeur du type des données qui vont être stockées dans cette zone

plus de détails dans le manuel : `malloc(3)`

➠ `malloc` **n'initialise pas** la zone réservée

calloc

```
void *calloc(size_t nmemb, size_t size)
```

- Demande au système de réserver une zone de `nmemb` cases contigus en mémoire pouvant toutes contenir `size` octets. Toutes ces cases sont initialisées à 0.
- Retourne l'adresse du début de la zone ou `NULL` si le système n'a pas pu trouver la place disponible
- Le pointeur doit être converti en pointeur du type des données qui vont être stockées dans cette zone

plus de détails dans le manuel : `calloc(3)`

realloc

```
void *realloc(void *ptr, size_t size);
```

- ☞ Change la taille de la zone mémoire pointée par `ptr` pour qu'elle occupe `size` octets
- ☞ le contenu des cases mémoires est inchangé (du début du bloc au minimum de l'ancienne et de la nouvelle taille)
- ☞ le pointeur `ptr` doit être le résultat d'un appel précédent d'une fonction de type `malloc` ou `realloc` (ou être NULL)
- ☞ retourne un pointeur sur la nouvelle zone mémoire, ou NULL si une erreur est survenue ou si `size` valait 0
- ☞ en cas d'erreur, l'ancienne zone mémoire n'est pas modifiée.

plus de détails dans le manuel : `realloc(3)`

free

```
void free(void * ptr)
```

- ☞ Prévient le système que la zone mémoire débutant à l'adresse ptr n'est plus utilisée et que sa réservation peut être libérée
- ☞ La zone pointée par `ptr` doit avoir été préalablement allouée par un appel à `malloc` ou `calloc`

plus de détails dans le manuel : `free(3)`