LINFO1252 - Systèmes informatiques

Projet 0 : Allocation dynamique de mémoire

Ce premier projet vous permettra de comprendre le fonctionnement des librairies d'allocation dynamique de mémoire. Pour ce faire, vous allez développer par vous-même un algorithme de gestion de la mémoire en implémentant votre propre version des fonctions `malloc` et `free`. Celles-ci gèreront un simple tableau d'octets qui simulera le heap.

Le projet est à réaliser **en binôme** pour le jeudi **7 novembre** à **20:00**. Chaque binôme doit s'inscrire dans le même groupe libre dans le module de gestion de groupe sur Moodle. Ceci est nécessaire pour pouvoir soumettre votre rapport. Il est autorisé de travailler seul·e mais la notation sera identique. Il est alors quand même nécessaire de choisir un groupe libre.

Spécifications des fonctions `malloc` et `free`

Vos fonctions `malloc` et `free` devront gérer un tableau d'octets (`uint8_t`) déjà déclaré globalement, et simulant l'espace mémoire autorisé pour le heap :

```
uint8_t MY_HEAP[64000];
```

Vous disposez d'une fonction `init` à implémenter et qui est appelée à l'initialisation du programme. Elle vous permet de préparer un état initial de la mémoire cohérent pour vos algorithmes.

La fonction `malloc` porte la signature suivante :

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

Elle renvoie un pointeur vers une zone de mémoire d'une taille *minimum* de `size` octets issue de ce tableau. La zone mémoire commençant à l'adresse à laquelle renvoie le pointeur doit rester utilisable dans son entièreté pendant toute la durée de l'utilisation de celui-ci, càd jusqu'à sa libération avec la fonction `free`.

La fonction `free` porte la signature suivante :

```
void my_free(void *ptr);
```

Elle libère une zone mémoire précédemment réservée indiquée par le pointeur `ptr`.
Pour plus de détails concernant ces fonctions, consultez le syllabus.

Description de l'algorithme, travail à réaliser sur papier, sous forme de schéma :

Après avoir conçu votre algorithme en suivant la méthode adoptée dans le TD, vous devrez le décrire sous forme d'un schéma repris dans un rapport court. Pour vous aider, voici une liste non-exhaustive des questions auxquelles vous devez répondre :

- Quel type de structure de données votre algorithme utilise-t-il ?
- Quelles métadonnées sont nécessaires ? Où sont-elles stockées, à quel endroit de la mémoire, et sous quel format ?

Ensuite, vous appliquerez votre algorithme sur un exemple précis et représenté par le scénario suivant :

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

Vous devez produire des schémas représentants la mémoire dans la forme introduite au TD et dans les trois étapes suivantes :

1. Après exécution de la fonction d'initialisation, mais vierge de toute exécution de `malloc` ou de `free`.
2. Après les 3 exécutions de `malloc`, mais avant l'exécution de `free`. Dans ce cas, indiquez les adresses retournées par chaque exécution, en considérant que l'adresse du début du tableau porte la valeur 0.
3. Après l'exécution de `free`.

Chacun des schémas doit faire clairement état des données et métadonnées stockée en mémoire.

Implémentation des fonctions `malloc` et `free`

Ensuite, vous devez implémenter les fonctions `malloc` et `free` en C selon votre algorithme et tout en respectant leur spécification ci-dessus. Vous implémenterez aussi la fonction `init`, qui sera exécutée avant chaque suite de test de vos fonctions pour préparer la mémoire pour vos allocations. Naturellement, vous ne pouvez pas avoir recours aux fonctions `malloc` et `free` de la libraire standard C.
Le programme est à soumettre sur la tâche INGInious correspondante :
https://inginious.info.ucl.ac.be/course/LINFO1252/s3_malloc

Échéance et livrable

Le projet est à rendre pour le jeudi **07 novembre 2024** à **20:00**. Pour ce projet, le livrable est constitué de deux parties :

1. Sur Moodle : un rapport d'une page obligatoirement au format PDF comprenant les explications des principes de fonctionnement de vos algorithmes, ainsi que trois diagrammes représentant l'état de la mémoire dans les trois étapes décrits ci-dessus. N'hésitez pas à utiliser des annotations (par exemple, numéros) sur les diagrammes et à les référencer dans le texte pour supporter votre explication.
2. Sur INGInious : le code, en langage C, de vos fonctions `malloc`, `free` et `init`. **Attention**: Vous devez vous inscrire dans le groupe sur Inginious portant le même numéro que votre groupe Moodle.