LINFO1252 – Systèmes informatiques

Travail Dirigé (TD) S3 : Introduction au projet d'allocation dynamique de mémoire

Objectifs de ce TD : Vous apprendrez les enjeux du problème d'allocation dynamique et utiliserez une technique de réflexion graphique pour concevoir un algorithme résolvant ce problème. Les compétences acquises sont nécessaires pour réaliser le projet.

Compétences travaillées : Réflexion préalable à la programmation, illustrations et diagrammes

Prérequis : Ce TD nécessite un support (physique ou virtuel) de votre choix vous permettant de réaliser des illustrations (e.g., papier, iPad, draw.io, etc.). Vous aurez également besoin de l'énoncé du projet, disponible également sur Moodle.

Étape 1 : Définir un langage graphique (10 minutes)

Nous allons introduire le sujet en travaillant sur une représentation graphique de la mémoire et des opérations effectuées pour permettre d'allouer des portions de mémoire à un programme depuis un zone de mémoire préexistante, puis de les libérer.

Commencez par représenter cette mémoire préexistante comme une série de cases contiguës. Chaque case peut contenir 8 bits (un octet ou *byte*). Vous pouvez représenter ces bits les uns à la suite des autres ou utiliser une notation hexadécimale pour représenter les nibbles qu'ils forment. Illustrez une zone mémoire préexistante de 16 bytes dans un état encore inconnu.

Étape 2 : Initialiser la mémoire (5 minutes)

Nous allons considérer un cas simple de deux allocations successives de quatre bytes. Avant de les réaliser, représentez l'état de la mémoire après initialisation de votre algorithme. Encoder dans les cases les valeurs que votre algorithme initialise avant toute opération sur la zone mémoire. Utilisez une couleur ou grisez la case pour distinguer les métadonnées des données utilisateurs, contenues dans cette zone mémoire préexistante. Quel *facteur d'alignement* pouvez-vous utiliser ?

Déterminer l'état initial de la mémoire demande de réfléchir à votre algorithme d'allocation. Pour débuter, envisagez un algorithme permettant une seule allocation et déduisez l'état initial nécessaire pour représenter la présence de cette allocation. Essayez ensuite de représenter l'état nécessaire pour effectuer deux allocations successives.

Étape 3 : Allocation de la mémoire (5 minutes)

Représentez l'état de la zone mémoire, en distinguant métadonnées et données utilisateurs, après une allocation de 4 bytes. Notez également l'adresse retournée à l'utilisateur après l'allocation. Ensuite, représentez l'état de la zone mémoire après une allocation de 4 bytes supplémentaires.

Étape 4 : Désallocation de la mémoire (5 minutes)

Représentez l'état de la zone mémoire après la désallocation du premier bloc alloué. Dans cette opération, l'utilisateur passe à votre algorithme l'adresse que vous lui avez retourné lors de la première allocation.

Étape 5 : Amélioration incrémentale de votre algorithme (20 minutes)

Notez les différentes hypothèses que vous avez introduite lors de la conception de votre premier algorithme à chaque étape précédente, par ex. sur la taille des blocs et le nombre d'allocations maximal, ainsi que la représentation en mémoire de vos métadonnées. Améliorez votre algorithme pour supporter plus d'allocations de taille arbitraire, des alternances d'allocations et désallocations, et utiliser de façon efficace la mémoire disponible.