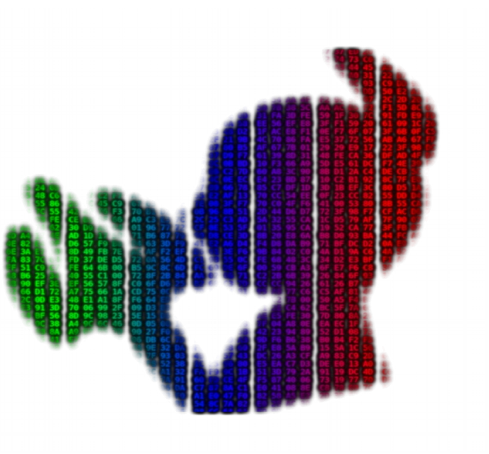

TP SPBRK

Gestion de la Memoires

- DaemonLab -
pedagogie@ecole-89.com

Table des matières

Détails administratifs

Votre travail doit être **envoyé sur la forge logicielle**. Le nom de votre dépôt doit être `2021_tp_ipc`. Un nom de dépôt erroné donnera lieu à un échec du TP.

Vous devez donner le droit en lecture à l'utilisateur `delivery-collector`, qui sera chargé de ramasser votre travail pour le corriger.

Vos identifiants de connexion à la forge sont votre mot de passe LDAP/UNIX, tel qu'il fonctionne sur les postes de l'école.

Médailles accessibles :

Rendu Git :
Vous avez réussi à rendre votre travail sur la forge logicielle

Prémisse

L'objectif de ce TP est de vous initier à la gestion de la mémoire et de vous faire découvrir le concept d'étendue de mémoire "memory range" et de tas "heap". Il est divisé en deux étapes : les bibliothèques partagées et les appels système de gestion de la mémoire.

Étape 1 – Les bibliothèques partagé

Remplacer une fonction d'un programme compilé par la votre :

- Écrivez un fichier .c qui contient les fonctions suivantes :
  - e89_write, qui affichera tout ce qu'elle recoit en majuscule, et qui dispose du même prototype que la fonction write.
- Compilez votre fichier en demandant à gcc de construire une bibliothéque partagée (shared object)
- Compilez votre fichier et spécifiez que vous voulez qu'il soit lié à votre bibliothèque partagée (avec (-l flag), comme pour les bibliothèques statiques).
- Pour charger les symboles (fonctions et variables globales) de votre bibliothèque et les appeler manuellement, vous devez utiliser les fonctions suivantes :
  - dlopen
  - dlclose
  - dlsym
  - Le flag RTLD_NEXT de dlsym dispose d'une fonctionnalité intéressante pour surcharger une fonction dont vous avez besoin dans cette surcharge… Il est accessible via les flags _GNU_SOURCE et __USE_GNU.

Vous pouvez définir la variable LD_PRELOAD pour précharger une bibliothèque dynamique pour l'ensemble des commandes qui suivront. Vous pouvez faire ca sur une seule ligne pour limiter son effet à une seule commande en faisant ainsi : LD_PRELOAD=./lib.so ls

Vous pouvez tester le fonctionnement de votre surcharge avec l'ensemble des commandes du shell. Attention, toutes ces commandes n'utilisent pas forcement write…

Comportement attendu :

Les bibliothèques partagées d'Unix, ou objets partagés, sont des collections de fonctions que vos programmes chargent dynamiquement pendant leur exécution.

Contrairement aux bibliothèques statiques, le contenu des fonctions appelées ne sera pas "copié" dans l'exécutable pendant la compilation.

Par conséquent, votre malloc doit être situé dans une bibliothèque partagée afin que le malloc du système puisse être remplacé sans avoir à recompiler les programmes que nous voulons tester avec malloc. Vous pourrez ainsi tester votre allocateur mémoire avec tous les programmes existants sans les modofier.

Étape 2 –Les Syscalls

Plongeons maintenant dans le vif du sujet en examinant les appels système liés à la gestion de la mémoire.les deux principaux appels permettent de manipuler le "break" (une sorte d'indicateur de la position supérieure de l'allocation). Sont les syscalls brk et sbrk.

Lire les manuels brk(2) et sbrk(2).

Voyons comment brk et sbrk fonctionnent.

- Utilisez sbrk et analysez la valeur de retour avec un paramètre positif, négatif ou nul.
- Prolongez le "break" et remplissez la mémoire nouvellement allouée pour voir si cela fonctionne. Ensuite essayer de remplir la mémoire au-delà du "break".

Vous remarquerez qu'il ne semble pas y avoir d'appel syscall pour "libérer" la mémoire. Alors, que pouvez-vous déduire du comportement de la fonction free ? Comment est-il possible de libérer de la mémoire en utilisant les appels système que vous venez de voir ?

"Bonus" -Pagination

Ce Bonus est une ouverture sur l'une des manières possibles d'optimiser les allocations de votre malloc.

- Lisez le manuel getpagesize(2).
- Réfléchissez à la façon d'utiliser les concepts de page pour améliorer les performances de votre malloc en limitant le nombre d'appels au système.