ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

3e année – IS – Systèmes et Réseaux

TP 1 – Segmentation mémoire et listes chaînées

C. BARÈS

Conseils généraux :

- *Ce TP doit être réalisé sous Linux (Installation ou VM).*
- *Vous êtes fortement encouragés à écrire **un** fichier par question (en copiant le fichier de la question précédente) et à utiliser un Makefile. Un Makefile générique qui compile chaque fichier .c est disponible sur moodle.*
- *Utilisez des commentaires pertinents (pas de : `i++; //incrémentation de i`);*
- *Voir pas de commentaires du tout !;*
- *De même, le découpage de votre programme en fonctions correctement nommées doit améliorer la lisibilité de votre code.*
- *Nommez vos constantes, n'utilisez pas de nombres « magiques » ;*

---

1 – Segmentation mémoire

En utilisant le champ "`%p`" de `printf` pour afficher une adresse, écrire un programme qui expose la localisation les segments mémoires suivants :

Data Données globales initialisées (stockées dans le fichier)
BSS Données globales initialisées à zéro (stockées dans le fichier)
Str Chaîne de caractères (stockées dans le fichier)
Heap Données allouées dynamiquement
Stack Données à portées limitées stockées dans la pile d'exécution
Main Function Zone mémoire code (.text)
LibC Function Zone mémoire librairie partagée
Mmap Zone mémoire allouées par « `mmap` »

Faites en sorte que votre programme fasse appel à la commande « `pmap -X PID` » pour afficher la carte mémoire de votre processus, et pouvoir ainsi vérifier les adresses des différents segments alloués.

2 – Projection de fichier en mémoire

Pour accélérer les accès aux fichiers volumineux, on préfère en général les mapper en mémoire plutôt que d'y accéder via les lectures/écritures standards. Le fichier mappé peut être accédé sous forme d'une zone mémoire accessible par un pointeur. Cette fonctionnalité est rendue possible par le mécanisme de mémoire virtuelle.

Pour mapper un fichier, on utilise la fonction « `mmap` » (et « `munmap` » pour clore le mapping mémoire).

1. Créez un fichier test.txt contenant du texte.
2. Ouvrez ce fichier (via « «`open`»), récupérez sa taille (via « «`stat`»).
3. Mappez l'intégralité du fichier en mémoire.
4. Inversez les octets du fichier (les octets de début de fichier se retrouvent à la fin).
5. Mettez fin au mapping mémoire.
6. Verifiez que le texte du fichier à bien été inversé (« `cat test.txt` »).

3 – Listes chaînées

1. Créez une liste chaînée contenant les n premiers entiers dans l'ordre croissant.
2. Créez une fonction qui renvoie la longueur d'une liste.
3. Créez une fonction qui affiche chaque élémet de votre liste sous la forme : <adresse du maillon> valeur du maillon.
4. Retirez le premier élément d'une liste.
5. Retirez le dernier élément d'une liste.
6. Ajoutez un élément à la fin d'une liste.
7. Ajoutez un élément au début d'une liste.
8. Écrivez une fonction qui concatène deux listes.
9. Écrivez une fonction qui retourne une nouvelle liste, après application d'une fonction à chaque élément (par exemple : élévation au carré).
10. Transformez votre liste en une liste doublement chaînée
11. Créez une liste doublement chaînée circulaire (adaptez les fonctions précédentes)