C1 - Unix & C Lab Seminar

CPE

Workshop Listes Chaînées & Tableaux de pointeurs sur fonctions

Les listes chaînées en s'amusant (c'est faux)!!!

1.0

Workshop Listes Chaînées & Tableaux de pointeurs sur fonctions

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

PART 1 : LISTES CHAINEES

Dans cette partie, vous devrez simuler une promotion (prénom, nom et âge de chaque étudiant). Le but de cet exercice est d'utiliser une liste chaînée où chaque maillon de la liste représente un étudiant. Étant donné qu'il s'agit d'un Workshop de renforcement, nous allons procéder étape par étape. Nous allons essayer de vous présenter le concept de la liste simplement chaînée. Si jamais vous êtes déjà capable de produire une liste chaînée, cet exercice ne devrait pas vous prendre longtemps. Vous êtes libre de passer sur la partie suivante, les tableaux de pointeurs sur fonctions.

+ EXO 1-1

Pour commencer, nous vous avons fourni un .h contenant la structure servant à produire votre liste chaînée.

Pensez à compléter ce .h si nécessaire.

Nous vous avons également transmis un .c contenant une fonction permettant de créer votre liste chaînée.

À partir de cette fonction, à vous de créer la fonction qui permet d'ajouter un élément à votre liste.

Son prototype est le suivant:

```
promo_t *add_elem(promo_t *list, char *name, char *lastname, int age);
```

+ Exo 1-2

Maintenant que vous êtes capable d'ajouter un élément à votre liste, il serait intéressant de pouvoir afficher le contenu de votre liste, ce qui vous permettrait donc de savoir si vous ajoutez vos éléments de la bonne manière.
Vous allez donc implémenter la fonction suivante:

```
void print_list(promo_t *list);
```

Cette dernière doit donc afficher le contenu de chaque élément de votre liste, c'est-à-dire le prénom, le nom et l'âge de chacun des étudiants présents dans votre liste.

+ Exo 1-3

Et la mémoire ?
Vous pouvez créer un élément et vérifier son contenu mais que devient-il quand on a plus besoin de lui ?
Eh bien c'est simple, on le détruit.
Rédigez une fonction permettant de détruire un élément de votre liste.

```
void destroy_elem(promo_t *list);
```

Vous devez évidemment détruire chaque élément de votre liste.

+ Exo 1-Bonus

Vous savez maintenant manipuler une liste simplement chaînée. Cependant, il est possible de créer des listes doublements chaînées. Ce concept consiste simplement à pouvoir se déplacer dans votre liste dans n'importe quel sens. Pour cela, il vous suffit de rajouter un élément permettant de naviguer dans le sens opposé à votre pointeur `next`.
Essayez donc d'afficher le contenu de votre liste mais dans le sens inverse. Il vous faut donc partir du dernier élément pour terminer par le premier…

PART 2

Cette partie concerne les tableaux de pointeurs sur fonctions. L'exercice consiste à prendre une chaîne de caractères passée en paramètre de votre programme ainsi que des entrées utilisateur. Ces dernières consisteront simplement à choisir comment afficher la dite chaîne de caractères. Si jamais l'exercice reste un peu abstrait à vos yeux, laissez vous guider par les étapes suivantes ou demandez de l'aide aux AERs.

L'AER roux n'est pas très agréable mais il ne mord pas.(C'est Lucas)

+ EXO 2-1

Récupérer simplement la chaîne de caractère passée en paramètre.

Ensuite, produisez une fonction permettant d'afficher la chaîne de caractères normalement, une faisant un affichage inversé (de droite à gauche) et une en majuscule.

+ EXO 2-2

Récupérer les entrées utilisateur.

En espérant que vous gardez des bons souvenirs de votre `get_next_line`.

+ EXO 2-3

Maintenant, il ne vous reste plus qu'à faire le tableau de pointeurs sur fonctions. Voilà bonne chance!

Non plus sérieusement, nous vous avons fourni un .h ainsi qu'un .c contenant la base permettant de réaliser cet exercice.