Système d'Exploitation- TP1
CERI / L3

A déposer dans l'ENT avant le début du TP suivant

Micro projet, organisation, librairies, etc...

L'objectif du TP est de réaliser un mini-projet en C, permettant d'afficher alternativement (toutes les secondes) des valeurs rangées dans une liste circulaire.

Pour cela, je vous demande d'utiliser la console de commandes et les commandes suivantes :

- **L'éditeur vi :** pour écrire les fichiers.
- **Le compilateur gcc :**

`gcc -c Main.c` : va générer le fichier Main.o
`gcc -o Projet Liste.o Main.o` : va générer le programme Projet avec les objets Liste.o et Main.o

Exercice 1 : Première Méthode

Le projet se composera de 3 fichiers contenant du code :

- **Liste.c** : contient les définitions des fonctions pour gérer la liste circulaire.
  Une liste circulaire est une liste qui n'a pas de tête ni de queue. Le maillon de tête est représenté par le maillon qui est en train d'être traité.
  - La fonction Suivant permet de remplacer le maillon courant de la liste (tête) par son suivant.
  - La fonction Supprimer supprime le maillon courant et passe automatiquement au Suivant (si il existe).
  - La fonction Ajouter ajoute un nouveau maillon avant le maillon courant (en dernier dans la file).

  Afin de faciliter les traitements, cette liste sera doublement chaînée (un pointeur vers le maillon suivant ET un vers le précédent).
- **Liste.h** : contient les déclarations des fonctions. **Aucune définition ne doit y figurer !!!**
- **Main.c** : contient le programme principal « main ». Ce programe affiche la valeur du maillon courant toutes les secondes (sleep) et attend une commande. Si la commande est :
  - (juste retour chariot) : on passe au maillon suivant,
  - S : le maillon courant est supprimé,
  - A 20 : un nouveau maillon avec la valeur 20 associée est crée,
  - X : fin du programme.

  Au début, la liste est vide et elle va croître puis décroître jusqu'à peut être redevenir vide puis se remplir à nouveau par la suite, etc.

Le projet sera organisé de la façon suivante :

Un répertoire TP0 contiendra 4 sous-répertoires : src, include, obj, bin.

- Le répertoire **src** contiendra les sources du projet (.c)
- Le répertoire **include** contiendra le fichier .h
- Le répertoire **obj** contiendra les fichiers .o
- Le répertoire **bin** contiendra l'exécutable **Liste**

Question 1 : Donner la liste des commandes permettant ces traitements (cp, mkdir, gcc, mv, etc…) :
Question 2 : Donner les 3 fichiers.

Exercice 2 : Librairie Liste.a

Nous souhaitons à présent faire une librairie nommée libListe.a contenant toutes les opérations que nous avons développées sur les listes.
Pour créer une librairie, il suffit de regrouper à l'aide de la commande **ar** les fichiers objets qui doivent y figurer (dans notre cas, uniquement Liste.o).

Dans ce cas, le projet sera réorganisé :

- Un repertoire libListe contiendra 4 sous répertoires :
  - src qui contient Liste.c
  - obj qui contient Liste.o
  - include qui contient Liste.h (regroupant les déclarations des fonctions figurant dans la librairie)
  - lib qui contient libListe.a
- Un repertoire TP0 contiendra 3 sous répertoires :
  - src qui contient Main.c
  - obj qui contient Main.o
  - bin qui contient Projet

Question 3 : Donner la liste des commandes permettant ces traitements (cp, mkdir, gcc, mv, etc…).

Exercice 3 : L'utilitaire Make

L'utilitaire Make permet de définir des dépendances entre des fichiers ainsi que les commandes à lancer lorsque les dates ne sont pas cohérentes.
Un fichier, nommé par défaut « makefile » permet de définir ces dépendances ainsi que les commandes à lancer. Ce fichier se compose de lignes du type :

```
Main.o : Main.c Liste.h
<tab>gcc –o Main.o –c Main.c

Executale : Main.o
<tab>gcc –o Executable Main.o
```

Dans cet exemple, si un des fichiers Main.c ou Liste.h est plus récent que le fichier Main.o, la commande « gcc –o Main.o –c Main.c » sera exécutée, etc….
Cet outil permet de n'exécuter que les commandes strictement nécessaires, sans se soucier de savoir quel fichier a été modifié ou pas, etc..
La commande : « make Executable » lancera les commandes nécessaires.

Question 4 : Faire un fichier makefile pour la construction de la librairie et un autre pour la construction du projet complet. Donner les contenus des 2 fichiers makefile.

Exercice 4 : Amélioration

Améliorer le programme pour qu'il affiche alternativement les valeurs que quand il y en a au moins 2. En effet, il est inutile d'utiliser la fonction sleep si la liste est vide ou bien ne contient qu'un élément.