PROGRAMMATION AVANCÉE
TP2
MANIPULATIONS DE FICHIERS

1. Entrées sorties Basiques

Exercice 1 Écrivez un programme qui prend en paramètre un fichier et le nom d'un second fichier, qui va copier le contenu du premier fichier dans le second. Si le second fichier n'existe pas il sera créé.
Exemple d'utilisation : `copie fichier.txt fichier2.txt`
On pourra vérifier le comportement correct de notre programme avec la commande :
`diff fichier1 fichier2` qui affiche les différences entre deux fichiers. Si la copie est correcte, rien ne doit être affiché.
Vous essaierez également de copie un fichier binaire (un programme par exemple).
Vous essaierez de copier le fichier exécutable de votre programme. Que se passe-t'il ?

Exercice 2 Même consigne que précédemment, mais en ajoutant la fin du fichier passé en second paramètre. Si le second fichier n'existe pas il sera créé.
Exemple d'utilisation :
`concatene fichier.txt fichier2.txt`

Exercice 3 Écrivez un programme qui prend en paramètre un fichier et qui supprime du fichier toutes les voyelles. Dans un premier temps on pourra avoir recours à un fichier auxiliaire.

Proposez une version de votre programme qui effectue toutes les opérations en place.

Exercice 4 Écrivez un programme qui remplit un tableau d'entiers `T` (disons 512 cases), avec la propriété suivante `T[i]` recevra la valeur i. Votre programme, à l'aide de la fonction `fwrite` écrira le contenu du tableau dans un fichier passé en paramètre.

On vérifiera la cohérence du fichier obtenu avec la commande `wc` qui compte, entre autres choses, le nombre de caractères et un éditeur hexadécimal (la commande `od` ou `hexdump` par exemple) pour déterminer si le contenu est en adéquation avec le contenu du tableau.

2. Codage/Décodage

Exercice 5 Écrivez un programme qui prend en paramètre deux fichiers. Le premier fichier représentera le fichier à coder et le second fichier contiendra une permutation.
Le programme effectuera les actions suivantes :

(1) Vérifiera et construira la permutation contenue dans le fichier permutation (*cf. TP1*). On notera par n le nombre d'éléments de la permutation.

(2) Pour chaque bloc de n caractères du fichier à coder, le programme effectuera une permutation des caractères selon la permutation présente dans le fichier permutation. Si à la fin on a moins de n caractères on laissera le bloc intact.

Exemple : permutation $= (2, 0, 1)$ et le contenu du fichier est le suivant :
`''il est 9h30''` on obtiendra le message suivant : `''l iste9h 30''`

Exercice 6 Écrivez un programme qui va décoder le fichier codé par le programme implémenté à l'exercice précédent. Le programme prendra en paramètre le fichier à décoder et le fichier qui contient la permutation

Exercice 7 Écrivez un prorgamme qui prend en paramètre un nom de fichier et un entier k (l'entier sera compris entre 0 et 255) et qui opère un décalage de k lettre (modulo 255) sur chacune des lettres du fichier.

Par exemple si le fichier contienr `hal` et le décalage donné est de 1. le fichier contiendra après décalage `ibm`.

Pour convertir la chaîne de caractères qui représente l'entier k en entier on pourra, à titre exceptionnel, utliser la fonction `atoi(char *)` (pour Ascii to Integer) qui convertit une chaîne de caractères qui représente un entier en entier.

3. Sérialisation

Exercice 8 Écrivez une fonction qui prend en paramètre une liste d'entiers (définie au TP1) et qui écrira son contenu dans un flux.

La syntaxe sera la suivante : `list: [n] element1, element2, ... , elementn` où n sera le nombre d'éléments. Le tout sera écrit sur une même ligne.
Pour la liste suivante $(1, 25, 9, 3, 6, 9, 8, 247, -15)$ on obtiendra la sortie suivante :
`list: [9] 1, 25, 9, 3, 6, 9, 8, 247, -15`

Exercice 9 Écrivez une fonction qui lit une ligne, depuis un flux, qui représente une liste (telle que définie à l'exercice précédent) et qui crée la liste correspondante.