Super-Encryptator

1 Objectif

Le but est d'écrire un jeu de programmes permettant de chiffrer et déchiffrer un fichier texte, en utilisant un dictionnaire. Plus précisément, il s'agit de créer les trois programmes suivants :

`key_generator` : créé un dictionnaire de chiffrement à partir d'un mot-clef et l'enregistre dans un fichier ;

`encrypt` : chiffre un fichier texte en utilisant le dictionnaire créé par `key_generator`, puis enregistre cette version chiffrée dans un nouveau fichier ;

`decrypt` : déchiffre un fichier texte, chiffré par `encrypt`, en utilisant le dictionnaire créé par `key_generator`, puis enregistre cette version déchiffrée dans un nouveau fichier.

2 Spécifications

Cette section décrit les spécifications des 3 programmes à réaliser. Toutes les fonctionnalités et caractéristiques indiquées dans cette section doivent donc être réalisées sans modification possible de votre part. À l'inverse, tout ce qui n'est pas spécifié reste à votre entière appréciation.

2.1 Dictionnaire de chiffrement, chiffrement et déchiffrement

Le principe du chiffrement est de créer un dictionnaire associant à chaque lettre de l'alphabet une autre lettre en se basant sur une clef secrète que donne l'utilisateur. Ensuite, pour chaque lettre de l'alphabet on associe, dans l'ordre alphabétique, une lettre de la clef.

Comme la clef est bien souvent plus petite que l'alphabet, les lettres restantes de l'alphabet sont associées aux lettres qui ne sont pas présentes dans la clef (ceci impose donc que la clef ne contienne pas de doublons). La figure 1 montre le dictionnaire de chiffrement obtenu avec la clef "CBOZD".

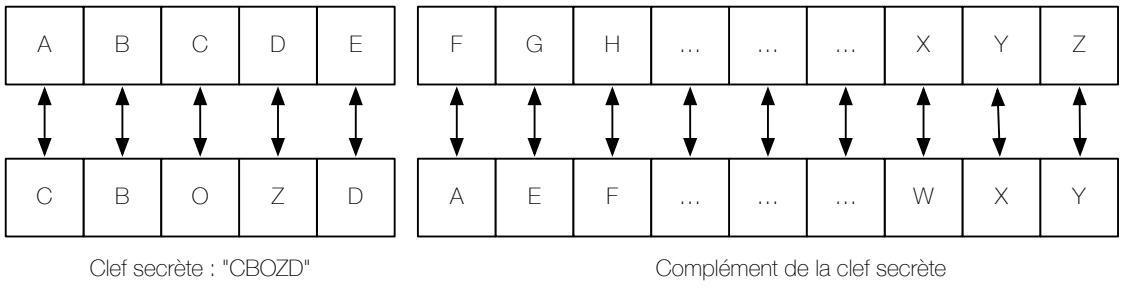

FIGURE 1 – Dictionnaire de chiffrement

Pour chiffrer un texte, il suffit de remplacer chaque lettre par son équivalent chiffré. Le déchiffrement se fait par application inverse. La figure 2 montre un exemple d'application de ce chiffrage avec le dictionnaire de la figure 1.

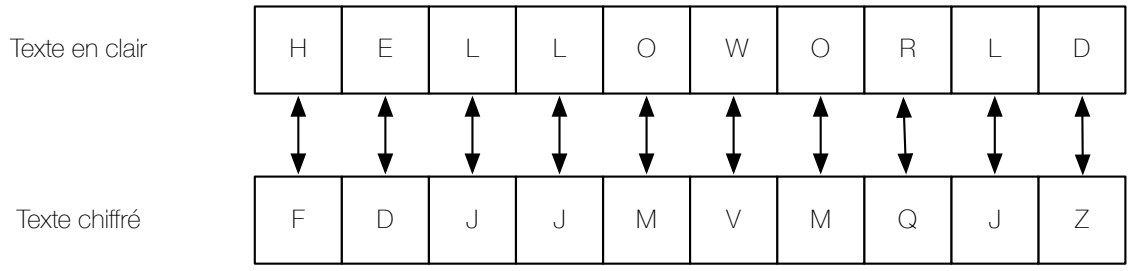

FIGURE 2 – Exemple de chiffrement et déchiffrement

2.2 Jeu de caractères à gérer

Vos programmes doivent gérer l'ASCII pour les caractères présents dans l'intervalle $[!, ~]$, soit en valeur décimale $[33, 126]$. L'ordre alphabétique sera défini par celui présent dans le standard ASCII.

2.3 `key_generator`

Ce programme doit prendre deux paramètres, dans cet ordre :

1. la clef secrète ;
2. le nom du fichier où enregistrer le dictionnaire de chiffrement.

Exemple : `key_generator CBOZD dict`.
Le programme devra vérifier que la clef secrète ne contient pas de doublons. Dans ce cas, aucun fichier ne sera généré et un message d'erreur clair sera affiché.

2.4 `encrypt`

Ce programme doit prendre trois paramètres, dans cet ordre :

1. le fichier contenant le dictionnaire de chiffrement ;
2. le nom du fichier à chiffrer ;
3. le nom du fichier où enregistrer le texte chiffré.

Exemple : `encrypt dict orig_file.txt enc_file.txt`.

2.5 `decrypt`

Ce programme doit prendre trois paramètres, dans cet ordre :

1. le fichier contenant le dictionnaire de chiffrement ;
2. le nom du fichier à déchiffrer ;
3. le nom du fichier où enregistrer le texte déchiffré.

Exemple : `encrypt dict enc_file.txt dec_file.txt`.

2.6 Gestion des fichiers

Dans tous les cas, tous les programmes doivent afficher un message clair si jmais il n'est pas possible de lire ou écrire les fichiers.

2.7 Rendu

Le programme doit être écrit avec le langage C. Il doit pouvoir être compilé à l'aide des compilateurs `gcc` et `clang`, avec les options `-Wall` et `-Werror`. De plus, il doit compiler et fonctionner sur les ordinateurs des salles F116 et F117 de l'ENSICA.

Le rendu se fera sur le LMS sous la forme d'une archive zip. Un fichier `Makefile` utilisable devra être fourni avec les sources.

2.8 Notation

Chacun des 3 programmes est noté sur 6 points (pour un total de 18 points), répartis comme suit :

— 4 points : Fonctionnalités remplies et tests qui passent

— 2 points : Qualité du code source (dont les commentaires et l'indentation)

Si un programme ne compile pas, la note sera de 0 pour celui-ci, sans aucune considération pour la qualité que le code correspondant pourrait avoir.

Les deux points restants porteront sur une appréciation générale (organisation des sources et du code, Makefile, etc.).

De plus, chaque fuite mémoire ou erreur indiquée par l'outil `valgrind` impliquera un malus de 0,5 point, pour un maximum de 5 points perdus.