TP 1 – Travail sur les fichiers et application à la stéganographie .

Q. Giorgi.

Recommandations

Lisez très attentivement le sujet

ATTENTION : La présentation, la lisibilité, ainsi que le **respect des consignes** seront prises en compte dans la notation.

Comme pour chaque TP, vous :

- Réaliserez un makefile pour la compilation des fichiers sources.
- Utiliserez les options de compilation -Wall -g
- Devrez rendre, selon les consignes de l'enseignant, un code source testé et qui se compile correctement!

Travail préparatoire, rappels et compléments de cours.

Deux exercices d'entrainement à réaliser avant le TP (exercices non notés, des indications seront données sur Chamilo pour ceux qui n'y arriveraient pas, il est très fortement recommandé de vous tester sur ces exercices qui normalement ne doivent vous prendre que quelques minutes (10 à 20 max).)

1. Faites un programme qui écrit dans un fichier nommé nombres_impairs.txt les 100 premiers nombres impairs positifs.
2. Ecrire un programme qui lit caractère par caractère un fichier dont le nom est passé par la ligne de commande et compte le nombre de voyelles (a,e,i,o,u,y majuscule et minuscule)

Lecture préparatoire :

- Lisez (et comprenez !) le document "Entrées-Sorties.pdf" fourni avant le TP.

- Si vous ne connaissez pas le sens du mot stéganographie, vous devez consulter cet article :

http ://fr.wikipedia.org/wiki/Steganographie

- Lisez cet article afin de connaitre le format d'un fichier bmp :

http ://fr.wikipedia.org/wiki/Windows_bitmap

Travail préparatoire à rendre en début de TP (ce travail sera évalué) :

- Ecrivez en quelques lignes de français l'algorithme qui permet de retrouver le fichier « source » à partir du fichier « transporteur.txt » (partie I)

Stéganographie

On appelera « source » le fichier à masquer, « originel » le fichier qui ne contient pas d'information cachée, et « transporteur » le fichier originel modifié incluant les informations masquées du fichier source.

I Une application un peu trop visible...

Ici la fonction de masquage de l'information dans le fichier « transporteur.txt » est un peu visible. Chaque caractère alphabetique du fichier « transporteur »(utilisez la fonction de la libc isalpha) est utilisé pour cacher un bit (Attention : pas un octet) du fichier source.[une minuscule 0, une majuscule 1].

Décodage :

Dans le fichier transporteur, si un caractère n'est pas alphabetique [a-z,A-Z] (ce qui peut être testé par la fonction « isalpha » de la bibliothèque standard C), il n'intervient pas dans le codage, on l'ignore.

Sinon si ce caractère est majuscule (isupper) ou minuscule (islower), il représente une information binaire qui permet de reconstituer des octest (8 bits par 8bits). Chaque octet resconstitué représente un octet du fichier source.

Question 1: Vous devez donc parcourir le fichier transporteur et reconstituer le fichier source

II Une application un peu moins visible...

La fonction de masquage est ici un peu moins visible.

Le fichier originel.bmp est une image au format bitmap ne contenant pas d'information cachée.

Le fichier « transporteur.bmp » contient lui de l'information cachée.

Si vous comparez les deux fichiers, la différence n'est pas (facilement) détectable à l'oeil nu car seul le bit de poids faible de chacune des couleurs de chaque pixels est modifié, ce qui à l'oeil nu ne présente que peu de variation. cependant chacun des bits de poids faible de chaque couleur correspondent aux bits (attention pas aux octets) du fichier source.

En parcourant chacun des pixels de l'image et en utilisant le bit de poids faible de chaque couleur de ce pixel, vous obtiendrez tous les bits du fichier source (que vous regrouperez bien sûr en octet).

Vous utiliserez le fichier bitmap.h

Question 1 : En lisant le fichier transporteur, remplissez tous les champs des structures indiquées dans le fichier bitmap.h. et affichez-les à l'écran.

Question 2 : Qui se cache derrière la barbe !!

Question 3 : Faites les fonctions de masquage qui à partir des fichiers source et originel produisent le fichier transporteur.