clement.rouvroy@ens.psl.eu

Stéganographie

Figure 1: XKCD - Un coeur en ASCII

Thème Manipulation de fichiers **Type** Centrale

Consignes Le candidat respectera le langage imposé (C) et ne devra pas utiliser de librairie hors-programme. Il est invité à **faire des schémas clairs et précis** de ses algorithmes lors des appels au correcteur, c'est à la fois un gain de temps pour les deux et une manière de montrer qu'il a compris ce qu'il manipule. Les preuves écrites doivent être formelles, sauf si la consigne précise que ce n'est pas nécessaire, la rigueur sera évaluée. L'examinateur ne débugera pas votre code, en revanche en cas de doute ("ai-je le droit à telle ou telle librairie ?", "je suis sortie de la VM, comment y revenir ?", "ai-je le droit à une indication pour cette question ?") n'hésitez pas à poser votre question. Enfin, si l'examinateur n'est pas à votre portée quand vous avez besoin d'une aide / d'une question orale, gardez le bras levé et lisez la suite, ne restez jamais passif.

Rendu Déposer votre code ici (mot de passe donné en colle). Sous forme d'archive `nom_prenom.extension` s.v.p (peu importe l'extension, mais rien d'exotique, tar,rar,zip).

Introduction du sujet

Dans ce TP, vous allez apprendre à faire de la stéganographie. La stéganographie consiste à cacher des messages dans des fichiers images. Nous allons le faire ici dans une version simplifiée, dans des images qui n'ont qu'une seule nuance de couleur, le gris.

Rappels Voici quelques fonctions C de la bibliothèque `stdio.h` (et des opérateurs génériques) qui peuvent vous êtres utiles, le programme précise que vous devez être capable de les maîtriser après rappel de la syntaxe.

- `fopen(filename: char*, mode: char*)` permet d'ouvrir un fichier. Le mode sera `r` (lecture), `r+` (lecture + écriture) ou `w` (écriture). Cette fonction renvoie un `FILE*`
- `fscanf(stream: FILE*, format: char*, arg1,..., argn)` effectue `scanf` sauf que son entrée n'est pas l'entrée standard mais le fichier passé en argument.
- `fprintf` même logique que `fscanf`
- `fclose(stream: FILE*)` permet de fermer un fichier
- `<<` est un left shift (un décalage à gauche) binaire. On l'utilise avec `x<<k` pour décaler `x` de `k` bits vers la gauche.
- `>>` est un right shift (comme pour `<<` mais vers la droite)

I - Lecture d'un message caché

On souhaite lire le contenu du fichier `secret.pgm`, l'avantage des fichiers Portable Grey Map est qu'ils sont très simples à manipuler avec la librairie `stdio` par leurs formats. Un fichier `pgm` a toujours cette tête:

```
MODE
MAX_X MAX_Y
MAX_GREY
CONTENU
```

Le champ `MODE` vaudra toujours `P2` dans ce TP (on le vérifiera dans le code, si ce n'est pas le cas on renverra une erreur).

Question 1 à l'écrit

Ouvrir le fichier `secret.pgm` et expliquer comment est encodé le contenu du fichier.

Question 2 code

Implémenter un type `image` qui contient ces différents champs:

- `haut`, un `int` (la hauteur)
- `larg`, un `int` (la largeur)
- `max_couleur`, un `int` (la couleur maximale)
- `pix`, un `uint16_t**` (les pixels de l'image)

Question 3 à l'écrit

Que signifie le `u` du type `uint16_t` ? Sur combien de bits est-il stocké ? Quel est l'entier maximal représentable ? L'entier minimal ?

Faire de même avec le typ `int16_t`.

Quand on développe un logiciel grand-public (nul doute que votre code de stéganographie va devenir très utilisé), on aime bien qu'il soit permissif, c'est-à-dire qu'on puisse compenser certaines erreurs de l'utilisateur. Ainsi, utiliserla librairie `string.h` pour répondre à cette question:

Question 4 code

Ecrire une fonction `char* verifie_url(char* url)` qui prend en entrée une url de fichier (par exemple `image.pgm`) et qui vérifie qu'elle finit bien par `.pgm`. Si ce n'est pas le cas, la fonction ajoute `.pgm` à la fin.

Question 5 code

Ecrire une fonction `image charger_image(char* fichier)` qui prend en entrée un nom de fichier et renvoie un type `image` qui contient les caractéristiques de l'image passée en argument. On supposera que les fichiers donnés en argument existent.

Pour coder un message, on utilise la méthode très simple suivante:

Les 8 premières valeurs de chaque ligne encodent un caractère en base 2, si la $i$ème valeur vaut $x_i, i \in [\![0; max\_couleur]\!]$, le $i$ème bit (en partant du bit de poids fort) vaut $x_i \bmod 2$. Par exemple 253 254 correspond à 10 donc 2 en base 10.

Question 6 code

Ecrire une fonction `void message(image img)` qui affiche le message encodé par `img`.

Avant de donner le résultat final, il serait bien de créer deux fichiers pour structurer le projet.

Question 7 code

Mettre à jour votre projet pour qu'il soit de la forme :

```
| stegano.h
| stegano.c
| main.c
```

et que `main.c` ne contienne que la fonction `main` qui sera par ailleurs créé pour ouvrir le fichier `secret.pgm` et utiliser le module `stegano.h` pour décoder le message.

Le programme ne précise pas si vous devez être capable de compiler ce genre de projets (mais vous devez être capable de manipuler les `.h`), par conséquent voici un bref cours à ce sujet (vous pouvez aussi faire `gcc main.c image.c` et faire semblant que vous lirez ce point de cours pour votre culture personnelle plus tard chez vous, je ne vous en tiendrais pas rigueur) :

Quand vous avez un fichier qui dépend d'un autre et que vous essayez de le compiler sans préciser où trouver les fonctions dont il a besoin, vous aurez une erreur (essayez `gcc main.c`). L'erreur n'apparaît en fait pas tout de suite ! Il y a deux étapes dans `gcc main.c`, la première produit un `main.o` à savoir un fichier "qui vous fait confiance", c'est-à-dire qu'il regarde dans votre stegano.h et voit que les types sont corrects, indépendamment de l'implémentation il va créer un fichier `main.o` qui n'a pas encore de liens, à savoir que les fonctions de `stegano.h` n'ont pas encore de corps, le fichier connaît uniquement

leurs signatures. Ensuite, il va essayer de faire les liens, et c'est ici qu'il n'y arrive pas, donc il renvoie une erreur.

La solution est de **forcer gcc à ne pas faire d'édition de liens** dans un premier temps, ce qui peut se faire très facilement par l'option `-c`. Ainsi, on commence par faire `gcc main.c -c` et `gcc stegano.c -c`. On peut alors par la suite ffectuer l'édition de lien en passant les `.o` à `gcc` : `gcc main.o stegano.o -o main`. Il ne reste plus qu'à lancer `./main` pour lancer le programme.

II - Ecriture d'un message

Vous êtes désormais capable de donner le message encodé par une image, voyons désor mais comment écrire un message dans une image donnée et comment la sauvegarder.

La première étape est de prendre un tableau de `uint16_t` de taille **au moins 8** (ce sera supposé) et de le modifier pour encoder le caractère voulu.

Question 8 code

Ecrire une fonction `void inserer_charactere(uint16_t* tab, char c)` qui encode `c` dans `tab`. On **utilisera aucune instruction if dans cette fonction**

Remarque: Si vous n'arrivez pas à faire votre fonction sans `if` pour une raison qui vous semble légitime, levez la main pour vérifier (il se peut que vous soyez juste en avance sur le sujet et donc c'est potentiellement très bien)

Question 9 code

En déduire une fonction `void cacher(image img, char* s)` qui cache `s` dans `img`

Question 10 code

Ecrire une fonction `void sauvegarder_image(char* fichier, image img)` qui met à jour `fichier` pour qu'il contienne désormais l'image `img`.

Question 11 à l'écrit

(Bonus) Qui est le monsieur en photo ?

III - Pour occuper les plus rapides

Déjà fini? Chercher ces deux exercices dans l'ordre que vous voulez. Lever la main une fois un exercice terminé pour me proposer votre solution:

Question 12 code

Vous avez fait une erreur sur votre fonction `inserer_charactere`. Corrigez-la. Vous avez cette fois le droit d'utiliser `if`. (l'erreur est volontairement présente sur la correction donc comparer votre code à celui de la correction n'aidera pas)

Remarque: Testez votre fonction sur des edge-cases de votre création peut aider...

Remarque: Vous avez possiblement évité le piège, si vous pensez être sûr de votre fonction appellez-moi. Cependant si l'erreur est présente ce sera retenu contre vous

Exercice 1 Elément majoritaire

1. Développer un algortihme en temps $O(n \log n)$ qui trouve l'élément majoritaire d'un tableau (apparaissant au moins $\lfloor \frac{n}{2} \rfloor$ fois). L'algorithme ne doit pas utiliser le tri.
2. De même en $O(n)$