
TD n° 4 - Manipulation d'images

Dans ce TD nous allons utiliser les appels systèmes d'entrée/sortie pour manipuler des fichiers représentant des images.

Tous les programmes écrits doivent gérer les éventuelles erreurs qui peuvent se produire lors de l'utilisation des appels systèmes, en particulier `open` (en cas d'erreur, affichez un message adapté à l'aide de `perror` et interrompez l'exécution du programme).

1 Images en niveaux de gris

Dans cette section, nous allons manipuler des images en niveaux de gris stockées au format *Portable Gray Map* (`pgm`). Un fichier `pgm` contient un en-tête suivi d'une partie de données binaires dans laquelle chaque point de l'image (pixel) est représenté par un octet du fichier, correspondant à son intensité en niveau de gris (0 : noir, 255 : blanc).

L'en-tête comporte au moins trois lignes :

- un code (ou *magic number* indiquant le format (« P5 » pour le `pgm`) ;
- la largeur de l'image, suivie de la hauteur. Les valeurs sont représentées par leurs représentations binaires (caractères ASCII) et sont séparées par au moins un espace ;
- la valeur maximale des niveaux de gris utilisés (en général 255), également en décimal ASCII. Cette valeur correspond à du blanc.

L'en-tête peut également contenir d'autres lignes de commentaires commençant par le caractère `#`. Pour simplifier les choses, on supposera dans ce TP que les en-têtes ne contiennent pas de commentaires et que le niveau de gris maximum est 255.

Par exemple, un fichier `pgm` pourrait commencer de la manière suivante :

```
P5
4320 3240
255
"+-6HB59;31.++35...
```

- 1 En utilisant la commande `convert` (vue au premier semestre), convertissez l'image de votre choix au format `pgm`. Renommez l'image produite en `image.pgm`.

Remarque : Pour visualiser les images `pgm`, vous pouvez utiliser le programme *gimp*.

- 2 Écrivez un programme `negative.c` qui prend en argument deux noms de fichiers, ouvre le premier (que l'on suppose être une image au format `pgm`) et écrit dans le second une image correspondant à une inversion d'intensité de chacun des pixels de l'image de départ.

2 Images en couleur

On s'intéresse maintenant aux images au format *Portable Pixel Map* (`ppm`). Tout comme pour le format `pgm`, les fichiers `ppm` commencent par un en-tête suivi de données binaires où chaque pixel de l'image est représenté par trois octets correspondant aux niveaux de rouge, vert et bleu de la couleur du pixel. Le format de l'en-tête est semblable à celui du format `pgm` :

- un *magic number* (« P6 » pour le format `ppm`) ;
- la largeur et la hauteur de l'image données en décimal ASCII ;

— le nombre de niveaux de chaque composante de couleur (en général 255).

3 Convertissez une image au format `ppm`, en la nommant `image.ppm`.

4 Écrivez un programme `permuté.c` qui permute les composantes de chaque pixel d'une image `ppm` (pour chaque pixel, `rgb` → `gbr`).

5 Écrivez un programme `intensite.c` qui modifie l'intensité de chaque composante des pixels d'une image en lui ajoutant une valeur constante passée en troisième argument (si la constante est de 15 par exemple, alors chaque composante doit être augmentée de 15).

Attention : La valeur de chaque composante doit rester entre 0 et 255.

Indication : Les arguments reçus par le programme sont tous des chaînes de caractères. Ceux qui représentent des entiers doivent donc être convertis en une variable de type `int`¹.

6 Écrivez un programme `miroir.c` qui permet de réaliser un effet de miroir vertical ou horizontal sur une image `pgm` ou `ppm`.

Le programme doit prendre en paramètre une option indiquant le type de symétrie à appliquer ainsi que les noms des fichiers en entrée et en sortie. Si la première option est `-v`, il faut appliquer une symétrie selon un axe vertical, si c'est `-h` l'axe est horizontal.

Indication : Il faut lire les données de l'image et les mémoriser dans un tableau (c'est plus simple pour le miroir vertical parce qu'on peut travailler ligne par ligne). Si l'image est grande, les tableaux ne peuvent pas être alloués de manière automatique et il faut alors utiliser l'allocation dynamique (`malloc`).

1. Vous pouvez utiliser les fonctions `strtol` (un peu compliquée mais recommandée) ou `atoi` (beaucoup plus simple mais ne détecte pas les erreurs).