

# Programmation séquentielle

## Série 10 - Fichiers et tokenisation

29.11.2022

### Traitement d'image rudimentaire

#### Buts

- Manipulation de pointeurs et chaînes de caractères.
- Lecture/écriture de fichiers.
- Tokenisation.
- Manipulation d'un format de fichier image.
- Utilisation de `git`.
- Utilisation de `make`.

#### Modalité

Vous devez créer un dépôt GIT pour héberger le code de cet exercice. Utilisez le fichier `.gitignore` pour éviter de commiter les fichiers exécutables.

Vous devez utiliser un `makefile` pour compiler votre code et utiliser au moins les paramètres suivants pour la compilation avec `gcc`:

```
-g -fsanitize=address
```

Le fichier contenant votre programme principal doit se nommer `symmetry.c` et l'exécutable généré doit se nommer `symmetry`.

#### Le format d'image PGM

Le format d'image PGM est un format très simple, disponible en version texte ou binaire. Nous utiliserons la version texte. Voici un exemple d'image encodé au format PGM :

```
P2
# Shows the word "FEEP" (example from Netpbm man page on PGM)
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

La première ligne contient le code `P2` qui désigne le type de fichier. La ligne suivante contient le nombre de colonnes et de lignes de l'image. Puis la valeur maximum que peut prendre chaque pixel. Ensuite, chaque ligne représente une série pixels de l'image, séparés par des espaces. Tout caractère à la suite du caractère `#` est un commentaire et n'est pas interprété.

Vous pouvez enregistrer cet exemple dans un fichier portant l'extension `.pgm` et l'ouvrir avec un éditeur d'image tel que Gimp.

## Enoncé

Le but de cet exercice est de lire un fichier PGM, d'appliquer une symétrie d'axe vertical sur l'image et d'enregistrer le résultat dans un nouveau fichier.

Pour cela vous devez :

- Lire l'entête du fichier original et le recopier dans la destination;
- Lire les lignes de l'image ligne par ligne;
- Tokeniser chaque ligne pour stocker la valeur de chaque pixel dans une structure de données (tableau ou pile).
- Parcourir la structure à l'inverse pour écrire la ligne renversée dans le fichier de destination.

Le nom du fichier d'entrée et de sortie doivent être donnés en argument de votre programme. Vous n'êtes pas obligé de gérer les commentaires ou les lignes vides dans le fichier source.

Par exemple, si le fichier `img.pgm` contient :

```
P2
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

L'appel :

```
./symmetry img.pgm img_out.pgm
```

Produira un fichier `img_out.pgm` contenant :

```
P2
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 15 15 15 15 0 0 11 11 11 11 0 0 7 7 7 7 0 0 3 3 3 3 0
0 15 0 0 15 0 0 0 0 0 11 0 0 0 0 0 7 0 0 0 0 0 3 0
0 15 15 15 15 0 0 0 11 11 11 0 0 0 7 7 7 0 0 0 3 3 3 0
0 0 0 0 15 0 0 0 0 0 11 0 0 0 0 0 7 0 0 0 0 0 3 0
0 0 0 0 15 0 0 11 11 11 11 0 0 7 7 7 7 0 0 0 0 0 3 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```