Initiation à la programmation en C

TP n°6

Antoine Miné

22 mars 2007

Site du cours: `http://www.di.ens.fr/~mine/enseignement/prog2006/`

Exercice 1. Commande `wc`.
Écrivez un programme qui compte le nombre de lignes d'un fichier passé en argument en ligne de commande (comme la commande Unix `wc` "word-count" avec l'option `-l`).

Modifiez ensuite votre programme pour qu'il accepte un nombre arbitraire de noms de fichiers en ligne de commande. Il faudra alors indiquer le nombre de lignes de chaque fichier ainsi que le nombre total de lignes.

Exercice 2. Lecture d'un entier.
Programmez à l'aide de la fonction `fgetc` (ou `getc`) une fonction `fgetint` qui lit un nombre décimal (*e.g.*, une suite de caractères `'0'` à `'9'`) dans un flux et le renvoie sous forme d'entier. Votre fonction devra donc avoir le prototype suivant.

```
int fgetint(FILE* stream);
```

Utilisez votre `fgetint` dans un programme de test qui se contente de lire un entier au clavier et de l'afficher à l'écran.

Modifiez votre fonction pour qu'elle accepte les nombres négatifs (commençant par un caractère `'-'` collé au premier chiffre).

Exercice 3. Manipulation d'images au format `.ppm` binaire.
Un fichier au format `ppm` binaire est un fichier contenant quelques lignes de texte (terminées par `\n`) suivies d'une suite d'octets :
- sur la première ligne, le mot `P6` (le nom du format),
- une ligne de commentaire (commençant par `#`),
- sur la troisième ligne, deux entiers séparés par un espace et représentant la largeur l et la hauteur h en pixels de l'image,
- sur la quatrième ligne, un entier (généralement 255) représentant l'intensité maximale de référence,
- ensuite, une séquence de $3 \times l \times h$ octets (sans espace ni retour à la ligne) représentant l'intensité de rouge, de vert, puis de bleu de chaque pixel de l'image.

Écrivez une fonction qui charge un fichier `.ppm` dans un tableau en mémoire ainsi qu'une fonction qui sauvegarde le contenu d'un tableau représentant une image dans un fichier. On supposera que les images sont de taille au plus 1024×1024. On propose les prototypes suivants pour ces fonctions :

```
void charge_image(const char* fichier, int* largeur, int* hauteur, unsigned char* image);
void sauve_image(const char* fichier, int largeur, int hauteur, unsigned char* image);
```

Ainsi, le tableau `image` où stocker la valeur des pixels est toujours passé par référence. De plus, lors de la lecture, la largeur et la hauteur de l'image lue (au plus 1024) sont retournés à l'appelant dans les variables `largeur` et `hauteur` passées par référence.
Écrivez ensuite une fonction qui réduit la résolution (largeur et hauteur) de l'image par deux. Pour cela, chaque pixel destination de coordonnées (x, y) sera calculé comme la moyenne des quatre pixels sources $(2x, 2y)$, $(2x + 1, 2y)$, $(2x, 2y + 1)$, $(2x + 1, 2y + 1)$.
Testez vos fonctions en écrivant un programme qui charge une image dont le nom est passé en ligne de commande, divise sa résolution par deux et sauvegarde le résultat dans un fichier passé également en ligne de commande.