Lecture de fichiers en C

Florian Bourse

Le but de ce TP est d'implémenter des fonctions basiques sur les fichiers textes grâce à un code source en C séparé en plusieurs fichiers.

Lecture de fichier (parsing.c)

Compléter le fichier `parsing.c` pour que la fonction `read_file` renvoie une chaîne de caractères de même contenu que le fichier dont le nom est passé en argument. La taille du fichier n'étant pas connu à l'avance, on ne peut pas utiliser de tableaux de taille statique pour contenir cette chaîne de caractères. Deux solutions sont proposées pour résoudre ce problème.

Lire deux fois le fichier

La première solution consiste à lire deux fois le fichier. La première fois pour compter le nombre de caractères, et ainsi savoir quelle est la taille requise pour le tableau qui contiendra la chaîne de caractères. La seconde fois pour inscrire les caractères lus dans ce tableau.

Redimensionner le tableau

Parfois, la solution précédente ne s'applique pas (e.g., si on veut stocker le contenu d'un flux comme `stdin`). Une autre solution consiste à commencer avec un tableau d'une certaine taille que l'on retient, et lorsque le tableau devient trop petit (si on cherche à écrire dans une case qui dépasse la dernière case du tableau), on créer un tableau plus grand, et on recopie les éléments du premier tableau dans le second. On continue ensuite avec le second tableau, et on libère la mémoire occupée par le premier. On peut être amenés à redimensionner plusieurs fois le tableau.

Outils

Pour s'occuper de cette partie, nous arons besoin des outils suivants :

`sizeof :` si on lui donne en argument un type, renvoie le nombre d'octets occupés par une valeur de ce type.

`malloc :` prend en argument un entier n et renvoie un pointeur vers un emplacement mémoire de n octets contigus. Si l'allocation de la mémoire échoue, renvoie la valeur particulière `NULL`.

`free :` prend en argument un pointeur p et libère l'emplacement mémoire vers lequel pointe p. Il faut que cet emplacement ait été alloué par un appel à `malloc`. On ne peut pas appeler `free` plusieurs fois sur le même emplacement mémoire.

`fopen :` prend en argument deux chaînes de caractères. La première désigne le nom d'un fichier et la seconde désigne les permissions utilisés pour ouvrir ce fichier : `"r"` désigne une ouverture du fichier pour la lecture uniquement. Cette fonction renvoie un descripteur de fichier, de type `FILE *`.

`fscanf :` prend en argument un descripteur de fichier, une chaîne de formattage, et éventuellement des pointeurs, selon la chaîne de formattage. Lit dans un fichier une chaîne de caractères qui correspond au format passé en entrée. Par exemple `"%c"` est une chaîne de formattage qui désigne 1 caractère. Le caractère remplacé par `"%c"` est recopié dans l'emplacement mémoire désigné par le pointeur donné en argument. La valeur de retour de `fscanf` est `EOF` si la lecture à échouée car le fichier est arrivé à sa fin. Équivalent le `scanf` mais en lisant dans un fichier plutôt que sur l'entrée standard.
Exemple d'utilisation : `scanf("n%d", &x);` lit un caractère `'n'` suivit de l'écriture décimale d'un nombre. Ce nombre est alors mis dans la variable `x`.

Si on alloue avec `malloc` un emplacement d'une taille correspondant à plusieurs fois la taille du type désigné par le pointeur, on accède au différents éléments avec des crochets, comme s'il s'agissait de tableau.

Analyse du fichier (main.c)

Le fichier `main.c` va utiliser la fonction définie dans le fichier `parsing.c` pour lire le texte présent dans un fichier. Le fichier fourni se contente de lire un fichier donné en argument du programme et de le recopier sur la sortie standard, mais on peut le modifier pour répondre à différent problèmes :

— Compter le nombre d'occurences d'un caractère.
— Déterminer si le parenthésage d'un fichier est cohérent.
— Vérifier qu'aucune ligne ne fasse plus de 80 caractères.
— Déterminer si un fichier est un palindrome.