TP2 : quelques exercices sur les opérateurs bit à bit, la manipulation de fichiers et les structures chaînées

1 Opérateurs bits a bits

1. Ecrire une instruction C qui permet de multiplier un entier par 2 sans utiliser l'opérateur de multiplication
2. Soient deux entiers non signés `a` et `b` ; créer un entier non signé `c` dont le contenu est le suivant : les 2 octets de poids faible de `c` sont les 2 octets de poids fort de `a` et les 2 octets de poids fort de `c` sont le complément à 1 des 2 octets de poids faible de `b`
3. Soient deux entiers non signés `a` et `b` qui diffèrent seulement par le bit 10. Ecrire l'instruction C qui permet de le vérifier.
4. Ecrire la fonction `int bitcount(int n)` qui donne le nombre de bits à 1 de l'entier `n`.

2 Manipulation de fichiers

Cet exercice vous permet d'une part de vous familiariser avec le traitement des fichiers, et plus particulièrement avec la lecture/écriture de lignes dans un fichier, et d'autre part d'étudier le passage de paramètres à la fonction `main` (ce qui correspond au traitement des options de la majorité des commandes Unix par exemple).

On désire réaliser un "équivalent" de la commande `split` du système Unix. Il s'agit donc de découper un fichier en un certain nombre de fichiers, chacun contenant un nombre fixe de lignes du fichier d'origine. Nous allons

ici réaliser plusieurs fonctions qui vont intervenir dans la réalisation de cette commande. **On suppose que l'on traite uniquement des fichiers texte**.

1. On suppose ici que l'on dispose d'un descripteur d'un fichier **précédemment ouvert en lecture**. On désire ici lire `nb_lignes` lignes dans ce fichier et les écrire dans un fichier que l'on crée. Le nom du fichier à créer, le nombre de lignes et le descripteur du fichier ouvert sont passés en paramètres. Vous devez donc réaliser la fonction :

   ```
   int EcritFichier(FILE * fich_lect, char * nom_fich_ecrit,
                    int nb_lignes)
   ```

   Cette fonction renvoie un entier. Cet entier doit indiquer si la fonction s'est exécutée correctement ou pas. Refléchissez aux différents cas.

2. Ecrivez maintenant une fonction qui ouvre un fichier en lecture et le découpe en plusieurs fichiers. Cette fonction a 3 paramètres :
   — le nom du fichier en lecture : `nom_fich_lect`
   — le nom de base de chaque fichier qui doit être écrit : `mon_fich_ecrit_base` ;
   chaque fichier écrit a ainsi pour nom :
   `nom_fich_ecrit_base.1, nom_fich_ecrit_base.2`, ...
   — le nombre de lignes à écrire dans chaque fichier.
   La fonction a donc pour définition :

   ```
   int CoupeFichier(char * nom_fich_lect,
                    char * nom_fich_ecrit_base,
                    int nb_lignes)
   ```

   Cette fonction renvoie un entier qui indique si la fonction s'est exécutée avec succès ou pas.

3. Ecrivez maintenant une fonction qui permet de saisir les différents noms décrits précédemment. Cette fonction a pour définition :

   ```
   int Saisie(char * nom_fich_lect, char * nom_fich_ecrit_base,
              int * nb_lignes)
   ```

4. Ecrivez la fonction `main` correspondante

5. On désire maintenant donner à l'utilisateur qui utilise votre programme la possibilité d'entrer les différentes données (qui sont en principe lues dans la fonction `Saisie`) depuis la ligne de commande. On désire en somme que l'utilisateur puisse appeler votre programme de la façon suivante (on suppose ici que votre programme s'intitule `mon_split`) :

`mon_split -l nb_lignes nom_fich_lect nom_fich_ecrit_base`

Ecrivez une fonction qui permet d'analyser les différents paramètres fournis à la fonction `main` et qui initialise `nom_fich_lect`, `mon_fich_ecrit_base`, `nb_lignes`.

`Note` : La fonction `main` doit être ainsi déclarée :

`int main(int argc, char *argv[])`, où `argc` est le nombre de paramètres fournis à la fonction (le nom du programme lui-même est compté dans le nombre de paramètres) et `argv` est un tableau de chaînes de caractères qui contient les paramètres fournis (le premier élément du tableau, `argv[0]`, est le nom du programme lui-même).

3 Structures chaînées

Nous proposons dans cet exercice de manipuler un ensemble de livres. Un livre est caractérisé par un auteur, une année et un titre. **On supposera pour simplifier que l'auteur et le titre sont représentés par un mot, sans espace**. Il est important qu'on puisse facilement afficher la liste de tous les livres quelque soit l'auteur, l'année ou le titre mais il est aussi important que l'on puisse facilement afficher la liste des livres d'un auteur particulier ainsi que la liste des livres d'une année particulière. Nous allons dans cet exercice utiliser une file pour gérer toutes ces contraintes.

3.1 Le livre

Proposez en langage C une structure de données permettant de représenter un livre. On vous impose de dimensionner statiquement les chaînes de caractères.

3.2 La file

Proposez en langage C, les structures de données permettant de représenter une file de livres, sachant que chaque cellule de la structure chaînée doit être liée à 1) la cellule suivante, 2) la prochaine cellule correspondant au même auteur et 3) la prochaine cellule correspondant à la même année.

3.3 Les opérations

1. Donnez le code C des fonctions `init_file` et `inserer_livre_file` qui permettent d'initialiser la file et d'insérer un nouveau livre dans la file, sans vous soucier des liens concernant les auteurs et les années pour le moment.
2. Donnez le code C de la fonction `classifie_par_auteur` qui prend en paramètre la file et un auteur et qui établit les liens reliant les cellules de cet auteur. Idem pour la fonction `classifie_par_date`.
3. Donnez le code C de la fonction `affiche_par_auteur` qui prend en paramètre la file et un auteur et qui affiche les livres correspondant à cet auteur, **en utilisant les liens établis ci-dessus**. Idem pour la fonction `affiche_par_date`.
4. Donnez le code de la fonction `init_livre` qui prend en paramètre un fichier ouvert et qui initialise une structure de données de type livre à partir d'une ligne du fichier, dont le format est : `Auteur Annee Titre`

3.4 Le main

Donnez le code C d'un programme principal typique :
— qui initialise une file de livres et insère dans cette file les livres décrits dans un fichier au format indiqué ci-dessus.
— qui effectue une classification et un affichage par un nom d'auteur quelconque, ainsi que par une année quelconque.