TP N°7
Manipulation de fichiers et algorithmique

Objectifs :

- ✓ Lecture et écriture de fichiers
- ✓ Manipulation de tableaux

Préambule

Après avoir ouvert virtualBox et vous être logué sous l'environnement CentOS5, on vous demande de créer un répertoire **tp7** dans votre répertoire de travail **algo.** Vous vous placerez ensuite dans le répertoire **tp7**. Vous allez créer un fichier nommé tp7.c qui contiendra le code source de votre TP (vous resterez dans ce fichier pour tout le TP).

On vous demande de concevoir et réaliser une application (en langage C) permettant de lire un fichier contenant une succession de points gagnés ou perdu par 2 joueurs de ping-pong, de calculer le score du match et d'écrire le résultat partiel ou final dans un fichier.

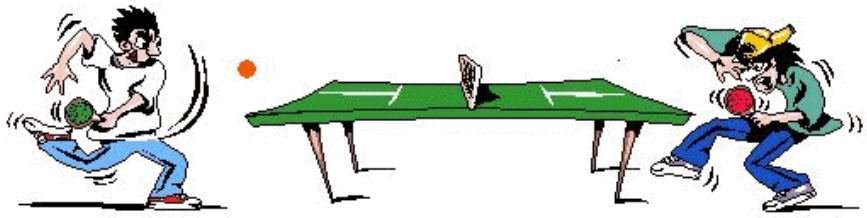

Figure 1 : match de ping-pong

Voici un exemple de fichier d'entrée : 0 correspond à un point marqué par le joueur A et 1 correspond à un point marqué par le joueur B.

```
101000101100
```

Dans le fichier de sortie on souhaite obtenir :

```
Score du 1 set: A-7      B-5
Match en cours
```

Un joueur gagne un set lorsqu'il est le premier à atteindre 21 points. Par simplicité on supposera qu'il n'y pas de notion de 2 points d'écarts pour gagner un set.

Un joueur gagne le match lorsqu'il est le premier à atteindre le nombre de set gagnants requis. Cette valeur sera paramétrable via la macro NB_SETS_GAGNANTS et on prendra comme valeur par défaut 2.

1. Lecture du fichier

On vous demande de créer la fonction suivante :

```
int* lirePoints(char* fichier, int* n)
```

Le but de cette fonction est d'ouvrir le fichier nommé « `fichier` », de lire caractère par caractère les données du fichier, de stocker les valeurs lues dans un tableau d'entier (contenant des 0 et des 1) et de retourner ce tableau en sortie de fonction. Le paramètre `n` est un argument de sortie représentant le nombre de points contenus dans le tableau. Afin de simplifier l'allocation mémoire on supposera que le nombre de points ne peut pas dépasser `41*(2*NB_SETS_GAGNANTS-1)`. Si le fichier contient davantage de points, on arrête la lecture !

Pour ouvrir un fichier en lecture, vous utiliserez la fonction `fopen` avec l'option « `r` » . Cette fonction retourne un pointeur de lecture sur le fichier ouvert (de type `FILE*`). Après avoir vérifié que le pointeur retourné n'était pas nul (le fichier existe bien), vous lirez le fichier caractère par caractère en utilisant la fonction `fgetc()`. Cette fonction retourne le code ASCII du caractère lu dans le fichier ou bien la valeur `EOF` (-1) signifiant qu'on est arrivé en fin de fichier.

En fin de lecture vous penserez à fermer le pointeur de lecture avec la fonction `fclose()`. Votre code devra être du type suivant :

```
FILE* pf = fopen(fichier,"r"); // ouverture du fichier
…
x = fgetc(pf);// lecture d'un caractère
…
fclose(pf); // fermeture du fichier
```

Après avoir compilé votre fonction, vous la testerez dans une fonction `main()` en utilisant les fichiers `input1.txt` et `input2.txt` fournis sur l'intranet.

2. Calcul du score

On vous demande à présent de créer la fonction :

```
struct score calculScore(int* points, int n)
```

retournant la structure suivante :

```
struct score{
      int joueurA[NB_SETS_GAGNANTS*2-1]; // points gagnés par A par set
      int joueurB[NB_SETS_GAGNANTS*2-1]; // points gagnés par B par set
      int setIdx;// index du set [0,1,…, NB_SETS_GAGNANTS*2-2]
      int vainqueur; // -1 en cours, 0 : A vainqueur, 1 : B vainqueur
};
```

Ainsi si on définit :

```
struct score s ;
```

et si on se trouve dans le 2ème set, alors `s->setIdx` vaut 1 et `s->joueurA[0]` est le score du joueur A dans le premier set et `s->joueurA[1]` est son score dans le 2ème set, etc… Cette fonction prend en entrée le tableau de points et calcul pour chaque joueur le nombre de points par set, l'index du set et si un joueur a gagné. Ces informations sont stockées dans la structure score et retournées en fin de fonction. L'algorithme s'arrête des qu'un joueur a gagné le match ou dès qu'on atteint la fin du tableau de points.

Après avoir compilé votre fonction, vous la testerez dans une fonction `main()` en utilisant les fichiers `input1.txt` et `input2.txt` fournis sur l'intranet.

3. Ecriture dans un fichier

On vous demande enfin de créer la fonction :

```
void ecrireScore(char* fichier, struct score s)
```

Cette fonction a pour but d'écrire le résultat du match dans le fichier nommé « fichier » sous la forme suivante :

```
Score du 1 set: A-7      B-21
…
Score du 3 set: A-5      B-21
Match termine, B vainqueur
```

Si le match n'est pas terminé on se reportera à l'affichage présenté dans le préambule.

Pour ouvrir un fichier en écriture, vous utiliserez la fonction `fopen` avec l'option « `w` » . Cette fonction retourne un pointeur d'écriture sur le fichier ouvert (de type `FILE*`). Après avoir vérifié que le pointeur retourné n'était pas nul (le fichier existe bien), vous écrirez dans le fichier au moyen de la fonction `fprintf()`. Cette fonction s'utilise de façon similaire que la fonction printf(), à ceci près que le pointeur d'écriture est à spécifier en premier argument.

En fin d'écriture, vous penserez à fermer le pointeur d'écriture avec la fonction `fclose()`. Votre code devra être du type suivant :

```
FILE* pf = fopen(fichier,"w"); // ouverture du fichier en écriture
…
fprintf(pf, "hello les amis\n" );// Ecriture d'une chaine de caractère
…
fclose(pf); // fermeture du fichier
```

Après avoir compilé votre fonction, vous la testerez dans une fonction `main()` en utilisant les fichiers `input1.txt` et `input2.txt` fournis sur l'intranet.