Préparation TP n°9
Lecture/écriture de fichiers
1h

On appelle fonctions I/O (*Input/Output*) les fonctions qui permettent de **lire ou d'écrire des données sur des périphériques externes** (en dehors de la mémoire de travail de l'ordinateur). Par exemple, la fonction `printf` que vous avez déjà vue, est une fonction d'I/O car elle écrit des flux de données vers le périphérique de sortie constitué par l'écran.

Tout transfert d'information depuis ou vers un périphérique externe (disque dur, écran, clavier...etc) est virtualisé sous la forme d'un flux entre le processus et ce périphérique. Ce flux circule à travers un canal que le processus va ouvrir et refermer. Le flux à travers ce canal va :

— du processus vers le périphérique si le processus écrit des informations,

— du périphérique vers le processus si le processus lit des informations.

La bibliothèque `stdio` nomme :

`stdin :` l'entrée standard, c'est-à-dire le clavier.

`stdout :` la sortie standard, c'est-à-dire l'écran.

`stderr :` la sortie d'erreur, qui est aussi l'écran.

Mais il est bien entendu possible de créer des flux d'information depuis (lecture) ou vers (écriture) des fichiers de données stockés sur le disque dur **Ouvrir un fichier, c'est donc ouvrir un canal entre la mémoire d'un processus et la zone de stockage du fichier sur disque dur, à travers lequel ce flux de données pourra circuler.**

Lorsque l'on demande l'accès à un fichier sur le disque dur, la bibliothèque C retourne un descripteur de fichier de type `FILE *` qui constitue le point d'ouverture du canal côté processus. Ce descripteur est donné en paramètre de toutes les fonctions d'entrée/sortie pour designer le canal qui doit être utilisé pour envoyer ou recevoir des informations.

L'accès aux données d'un fichier est séquentiel : si je veux accéder à une information située au milieu du fichier, je dois d'abord lire toutes les données stockées en amont. La position courante de la tête de lecture est stockée de manière cachée dans le descripteur de fichier (`FILE *`) pour le suivi de la lecture ou de l'écriture séquentielle.

Nous présentons les fonctions C, codées dans la bibliothèque `stdio`, permettant d'envoyer ou de recevoir des données par l'intermédiaire d'un canal depuis ou vers un fichier du disque dur (entre autres) :

`FILE *fopen(char *chemin_fichier, char *mode) :` ouvre un canal entre le processus et un fichier, en indiquant le chemin d'accès du fichier (absolu ou relatif, par défaut à partir du répertoire d'exécution du programme) et le mode d'ouverture permettant de préciser le sens des flux dans ce canal : `''r''` pour lecture (du fichier vers le processus), `''w''` pour écriture (du processus vers le fichier), `''a''` pour écriture à la suite (*append*). Elle renvoie un nouveau descripteur de fichier si tout s'est bien passé, et `NULL` sinon. A chaque ouverture du fichier, la tête, dont la position est stockée dans le descripteur de fichier, se repositionne en début de fichier car l'accès est séquentiel. Si le fichier n'existe pas, il est crée.

int fclose(FILE *f) : ferme le canal entre le processus et le fichier de descripteur `f`. Elle renvoie 0 si tout s'est bien passé, un code d'erreur sinon.

int feof(FILE *f) : cette fonction indique si la tête de lecture a atteint la fin du fichier en retournant la valeur 1, et 0 sinon.

int fprintf(FILE *f, char *format, ...) : cette fonction s'utilise comme `printf`, sauf que le canal n'est pas dirigé vers l'écran de l'ordinateur mais vers le fichier de descripteur `f`. Elle retourne le nombre caractères (octets) envoyés dans le canal au cours de l'instruction.

int fscanf(FILE *f, char *format, ...) : cette fonction permet de lire des informations depuis un fichier et de les stocker dans des variables dont on aura donné l'adresse. Elle retourne le nombre caractères (octets) reçus depuis le canal au cours de l'instruction.

Le code 1 ci-dessous montre un exemple d'utilisation pour l'écriture d'un fichier. Le code récupère une phrase donnée par l'utilisateur sur la ligne de commande et l'écrit dans le fichier `./toto.txt`, qu'il crée si nécessaire.

```c
#include <stdio.h>
#include <assert.h>

int main(int argc, char **argv)
{
  assert(argc == 2);

  char *phrase = argv[1];

  // ouverture d'un canal vers le fichier de chemin "./toto.txt"
  FILE *f = fopen("toto.txt", "w");
  if (f == NULL)
  {
    printf("Erreur lors de l'ouverture du fichier toto.txt en écriture\n");
    return 1;
  }

  // ecriture dans le fichier de descripteur f
  fprintf(f, "J'ai écrit cette phrase dans le terminal: %s\n", phrase);

  // on n'oublie pas de refermer le canal
  fclose(f);

  return 0;
}
```

Le code 2 ci-dessous montre un exemple d'utilisation pour la lecture d'un fichier. Le code compte le nombre de caractères du fichier texte ASCII `exemple.txt`.

```c
#include <stdio.h>
// Fichier exemple pour montrer l'utilisation des fonctions de la bibliothèque stdio
// Ici, on utilise les fonctions de lecture séquentielle d'un fichier
int main(void)
{
  // On ouvre un canal en lecture depuis le fichier exemple.txt
  FILE *f = fopen("exemple.txt", "r");
  if (f == NULL)
  {
    // si la fonction fopen renvoie un descripteur de fichier NULL
    // il y a eu un problème,
    //souvent c'est que le fichier n'existe pas chemin d'accès
    //qui consistue le nom du fichier n'est pas valide
    printf("Erreur lors de l'ouverture du fichier ./exemple.txt en lecture\n");
    return 1;
  }

  char caractere_lu;
  int compteur = 0;

  while (feof(f) == 0 )
  {
    // pour lire, on utilise un chaine format et on donne l'adresse
    // des variables dans lesquelles il faut stocker les valeurs lues
    // et formatées. Ici, on lit un char (%c) et on stocke le résultat
    // dans la variabel caractere_lu qui est bien de type char,
    //et dont on a passé l'adresse à la fonction fscanf
    fscanf(f, "%c", &caractere_lu);

    printf("Je viens de lire le caractère [%c]\n", caractere_lu);
    compteur = compteur + 1;
  }
  // On enlève 1 car emacs rajoute une ligne vide en fin de fichier
  printf("J'ai lu au total %d caractères\n", compteur-1);

  // On n'oublie pas de refermer le canal!
  fclose(f);

  return 0;
}
```

Exercice 1 (I/O).

1. Créez un dossier `TP9` dans votre répertoire d'informatique sur votre machine virtuelle.
2. Téléchargez le fichier `exemple.txt` disponible sur Moodle (clique-droit et sélectionner *Enregistrer la cible du lien sous...*). Dépla dans le répertoire `TP9`.
3. Dans le répertoire `TP9`, recopiez le code 1 dans un fichier `io1.c` (sans nécessairement rajouter les commentaires) et faites le fonctionner.
4. Dans le répertoire `TP9`, recopiez le code 2 dans un fichier `io2.c` (sans nécessairement rajouter les commentaires) et faites le fonctionner.
5. Bien comprendre ces codes et leur fonctionnement pour demain.
6. Copiez le dossier `TP9` sur votre clé USB dans votre répertoire d'informatique. Apportez votre clé USB demain.