TP9 - Manipulation de fichiers

Langage C (LC4)

Semaine du 2 avril 2012

1 Utilisation des redirections d'Unix

Question 1. Écrivez un programme `entree_sortie.c` qui :

1. affiche « Entrez 2 entiers : »,
2. passe à la ligne,
3. attend deux nombres entiers au clavier séparés par un retour à la ligne
4. et affiche leur somme à l'écran.

Vous utiliserez la fonction `scanf` pour récupérer les deux nombres, et la fonction `printf` pour afficher leur somme. En cas d'erreur (c'est-à-dire si `scanf` ne récupère pas deux entiers et ne retourne donc pas 2), vous afficherez un message d'erreur à l'aide de la fonction `perror`. Essayez d'entrer autre chose que des entiers pour voir le message d'erreur.

Solution.

```c
#include <stdio.h>

int main()
{
  int i, j;
  printf("Entrez 2 entiers :\n");
  if (scanf("%d%d", &i, &j) == 2)
    printf("La somme est %d.\n", i + j);
  else
    perror("Mauvais arguments de l'utilisateur");
  return 0;
}
```

Quand on exécute un programme en ligne de commande, on peut remplacer l'entrée standard (le clavier) par un fichier, à l'aide de l'opérateur <.

Question 2. Écrivez deux nombres entiers dans un fichier `nombres` et appliquez le programme `entree_sortie` dessus.

On peut aussi rediriger la sortie standard (qui est d'habitude affichée dans le terminal) vers un fichier, à l'aide de l'opérateur >.

Question 3. Redirigez la sortie du programme `entree_sortie` vers un fichier `somme`.

On peut aussi rediriger la sortie d'erreur standard (qui habituellement est également dirigée vers le terminal) vers un fichier, à l'aide de l'opérateur `2>`.

Question 4. Redirigez la sortie d'erreur du programme `entree_sortie` vers un fichier `erreur` (ne redirigez pas le reste) et donnez en entrée autre chose que des nombres.

Néanmoins, il n'est pas toujours possible d'utiliser les redirections d'Unix pour travailler sur des fichiers. C'est notamment le cas lorsqu'on veut lire (ou écrire) des données sur plusieurs fichiers. Mais le langage C dispose de fonctions spécifiques de manipulation de fichiers.

2 Utilisation de la bibliothèque `<stdio.h>` de C

2.1 Ouverture et fermeture de fichiers

Elles s'effectuent à l'aide des fonctions `fopen` et `fclose`. On peut faire une analogie avec `malloc` et `free` dans la mesure où `fopen` et `fclose` se chargent de créer et de détruire des pointeurs un peu particuliers de type `FILE *`. Un tel pointeur est appelé *flot* ou *canal* :

- `FILE *flot = fopen("foo.txt", "r");` permet d'ouvrir le fichier `foo.txt` en lecture (`"r"`) — pour écriture, ça serait `"w"`. Cette fonction renvoie le pointeur `NULL` en cas d'erreur. Il est prudent de vérifier que le flot retourné n'est justement pas `NULL` avant d'essayer de travailler dessus !
- `fclose(flot);` permet de fermer le canal et de libérer le pointeur `flot`.

2.2 Manipulation caractère par caractère

On utilise les fonctions `fgetc` pour lire et `fputc` pour écrire un caractère.

Question 5. Écrivez un programme `compte_caracteres.c` qui prend en argument le nom d'un fichier et affiche le nombre de caractères de ce fichier.

Solution.

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

int main(int argc, char *argv[])
{
  if (argc < 2) {
    printf("Veuillez donner un nom de fichier svp !\n");
    return EXIT_FAILURE;
  }
  FILE *f = fopen(argv[1], "r");
  if (f == NULL) {
    perror("Probl'eme 'a l'ouverture du fichier");
    return EXIT_FAILURE;
  }
  int compteur = 0;
  while (fgetc(f) != EOF)
    compteur++;
  printf("Ce fichier comporte %d caract'eres\n", compteur);
  fclose(f);
  return EXIT_SUCCESS;
}
```

Question 6. Écrivez un programme `melange_caracteres.c` qui prend en argument le nom de trois fichiers, et qui écrit sur le troisième fichier une alternance de caractères provenant des deux premiers fichiers. Lorsque la fin du fichier le plus court est atteinte, on continue avec ce qui reste de l'autre fichier. Testez votre programme.

Solution.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  if (argc < 4) {
    printf("Donnez 3 noms de fichiers !\n");
    return EXIT_FAILURE;
  }
  FILE *f1 = fopen(argv[1], "r");
  FILE *f2 = fopen(argv[2], "r");
  FILE *f3 = fopen(argv[3], "w");
  if (f1 == NULL || f2 == NULL || f3 == NULL) {
    perror("Probl'eme lors de l'ouverture d'un des fichiers.");
    return EXIT_FAILURE;
  }
  int c1 = fgetc(f1), c2 = fgetc(f2);
  while (1) {
    if (c1 == EOF && c2 == EOF) break;
    if (c1 != EOF) { fputc(c1, f3); c1 = fgetc(f1); }
    if (c2 != EOF) { fputc(c2, f3); c2 = fgetc(f2); }
  }
  fclose(f1), fclose(f2), fclose(f3);
  return EXIT_SUCCESS;
}
```

2.3 Manipulation ligne par ligne

On utilise les fonctions `fgets` pour lire une ligne (c'est-à-dire une chaîne jusqu'à un `'\n'` et `fputs` pour écrire une chaîne. Pour utiliser `fgets`, on définira une macro-constante `TAILLE_MAX_LIGNE` donnant la taille maximum d'une ligne (on fera attention au fait que `fgets` necessite de garder deux caractères supplémentaires `'\n'` et `'\0'` à la fin de la chaîne lue).

Question 7. Écrivez un programme `compte_lignes.c` qui prend en argument le nom d'un fichier et affiche le nombre de lignes de ce fichier.

Solution.

```
#include <stdio.h>
#include <stdlib.h>

#define TAILLE_MAX_LIGNE 100

int main(int argc, char *argv[])
{
```

```
  if (argc < 2) {
    printf("Veuillez donner un nom de fichier svp !\n");
    return EXIT_FAILURE;
  }
  FILE *f = fopen(argv[1], "r");
  if (f == NULL) {
    perror("Probl'eme 'a l'ouverture du fichier");
    return EXIT_FAILURE;
  }
  int compteur = 0;
  char buffer[TAILLE_MAX_LIGNE + 2];
  while (fgets(buffer, TAILLE_MAX_LIGNE + 2, f) != NULL)
    compteur++;
  printf("Ce fichier comporte %d lignes\n", compteur);
  fclose(f);
  return EXIT_SUCCESS;
}
```

Question 8. Écrivez un programme `melange_lignes.c` qui prend en argument le nom de trois fichiers, et qui écrit sur le troisième fichier une alternance de lignes provenant des deux premiers fichiers. Lorsque la fin du fichier le plus court est atteinte, on continue avec ce qui reste de l'autre fichier. Testez votre programme.

Solution.

```
#include <stdio.h>
#include <stdlib.h>

#define TAILLE_MAX_LIGNE 100

int main(int argc, char *argv[])
{
  if (argc < 4) {
    printf("Donnez 3 noms de fichiers !\n");
    return EXIT_FAILURE;
  }
  FILE *f1 = fopen(argv[1], "r");
  FILE *f2 = fopen(argv[2], "r");
  FILE *f3 = fopen(argv[3], "w");
  if (f1 == NULL || f2 == NULL || f3 == NULL) {
    perror("Probl'eme lors de l'ouverture d'un des fichiers.");
    return EXIT_FAILURE;
  }
  char b1[TAILLE_MAX_LIGNE + 2];
  char b2[TAILLE_MAX_LIGNE + 2];
  char *p1 = fgets(b1, TAILLE_MAX_LIGNE + 2, f1);
  char *p2 = fgets(b2, TAILLE_MAX_LIGNE + 2, f2);
  while (1) {
    if (!p1 && !p2) break;
    if (p1) {
      fputs(b1, f3);
      p1 = fgets(b1, TAILLE_MAX_LIGNE + 2, f1);
    }
```

```
    if (p2) {
      fputs(b2, f3);
      p2 = fgets(b2, TAILLE_MAX_LIGNE + 2, f2);
    }
  }
  fclose(f1), fclose(f2), fclose(f3);
  return EXIT_SUCCESS;
}
```

2.4 Entrée, sortie et erreur standard

Dans `<stdio.h>` on dispose de trois constantes de type `FILE *` pour désigner l'entrée, la sortie et l'erreur standard : `stdin`, `stdout`, `stderr`. On les utilise comme s'il s'agissait de fichiers.

Question 9. Réécrivez dans `fentree_fsortie.c` le tout premier programme `entree_sortie.c` en utilisant `fscanf` à la place de `scanf`, `fprintf` à la place de `printf` et de `perror`.

Solution.

```
#include <stdio.h>

int main()
{
  int i, j;
  fprintf(stdout, "Entrez 2 entiers:\n");
  if (fscanf(stdin, "%d%d", &i, &j) == 2)
    fprintf(stdout, "La somme est %d.\n", i + j);
  else
    fprintf(stderr, "Mauvais arguments de l'utilisateur\n");
  return 0;
}
```

Analogue à `realloc` pour le changement de taille d'une zone pointée, la fonction `freopen` permet de réaliser une redirection (c'est-à-dire de changer de fichier pointé), non plus en ligne de commande comme au début du TP, mais depuis l'intérieur d'un programme.

Question 10. Réécrivez dans `fredirection.c` le programme `entree_sortie.c` en redirigeant l'entrée standard vers le fichier `nombres`, la sortie standard vers le fichier `somme` et la sortie d'erreur standard vers le fichier `erreur`.

Solution.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  if (freopen("nombres", "r", stdin) == NULL) {
    perror("Redirection de stdin impossible");
    return EXIT_FAILURE;
  }
```

```
  if (freopen("somme", "w", stdout) == NULL) {
    perror("Redirection de stdout impossible");
    return EXIT_FAILURE;
  }
  if (freopen("erreur", "w", stderr) == NULL) {
    perror("Redirection de stderr impossible");
    return EXIT_FAILURE;
  }
  int i, j;
  printf("Entrez 2 entiers :\n");
  if (scanf("%d%d", &i, &j) == 2)
    printf("La somme est %d.\n", i + j);
  else
    printf("Mauvais arguments de l'utilisateur");
  return 0;
}
```