Prog. Or. Système - Correction série 04 : Entrées/Sorties

Exercice 1 : écriture dans un fichier
Voici un code relativement complet. La plus grosse difficulté est de gérer correctement les diverses anomalies possibles lors de la saisie de l'entrée. Si vous ne le faîtes pas, votre code sera évidemment bien plus court (mais beaucoup moins robuste). Regardez donc avec attention le corrigé ci-dessous.

La difficulté la plus subtile pour un déroulement correct est de supprimer le retour à la ligne ( ) qui traîne encore dans le buffer d'entrée après le `scanf` sur l'age.

(fichier src/ecriture.c)

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

/* taille maximale pour un nom */
#define TAILLE_NOM 1024

int main(void)
{
  char const nom_fichier[] = "data.dat"; /* le nom du fichier */
  FILE* sortie;
  int taille_lue;

  char nom[TAILLE_NOM]; /* pour stocker le "nom" à lire depuis le clavier */
  unsigned int age;     /* pour stocker l'"âge" à lire depuis le clavier */

  /* Ouverture de data.dat en écriture (w=write) */
  sortie = fopen(nom_fichier, "w");

  /* on teste si l'ouverture du flot s'est bien réalisée */
  if (sortie == NULL) {
    fprintf(stderr,
            "Erreur: le fichier %s ne peut etre ouvert en écriture !\n",
            nom_fichier);
    return 1; /* retourne un autre chiffre que 0 car il y a eu une erreur */
  }

  /* itération sur les demandes à entrer :
     on continue tant que stdin est lisible */
  while (! feof(stdin)) {

    /* tant qu'un nom vide est entré */
    do {
      printf("Entrez un nom (CTRL+D pour terminer) : "); fflush(stdout);
```

```
      fgets(nom, TAILLE_NOM, stdin);
      taille_lue = strlen(nom) - 1;
      if ((taille_lue >= 0) && (nom[taille_lue] == '\n'))
        nom[taille_lue] = '\0';
    } while (!feof(stdin) && (taille_lue < 1));

    if (! feof(stdin)) {
      /* L'utilisateur a bien saisi un nom, on peut donc lui demander
       * de saisir l'age.
       */
      printf("âge : "); fflush(stdout);
      taille_lue = scanf("%u", &age);

      if (taille_lue != 1) {
        printf("Je vous demande un age (nombre entier positif) pas "
               "n'importe quoi !\nCet enregistrement est annulé.\n");
        while (getc(stdin) != '\n'); /* vide le tampon d'entrée */
      } else {
        getc(stdin); /* récupère le \n résiduel */
        /* ecriture dans le fichier */
        fprintf(sortie, "%s %d\n", nom, age);
      }
        }
  }

  /* purisme : retour a la ligne pour finir proprement la question */
  putchar('\n');

  fclose(sortie); /* fermeture du fichier */

  return 0;
}
```

Exercice 2 : lecture depuis un fichier
Aucune difficulté ici.

(fichier src/lecture.c)

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

/* on affiche les noms sur 15 caractères, comme spécifié dans la donnée */
#define TAILLE_NOM 15

int main(void)
{
```

```
char const nom_fichier[] = "data.dat"; /* le nom du fichier */
FILE* entree;
int taille_lue;

char nom[TAILLE_NOM+1]; /* la donnée "nom" à lire depuis le fichier */
unsigned int age;       /* la donnée "age" à lire depuis le fichier */

/* variables nécessaires aux différents calculs */
unsigned int nb = 0;
unsigned int age_max = 0;
unsigned int age_min = (unsigned int) -1; /* truc : -1 sera toujours le plus
                                             grand nombre représentable */
double total = 0.0;

/* ouverture du fichier en lecture (r=read) */
entree = fopen(nom_fichier, "r");

/* on teste si l'ouverture du flot s'est bien réalisée */
if (entree == NULL) {
  fprintf(stderr,
          "Erreur: le fichier %s ne peut etre ouvert en lecture !\n",
          nom_fichier);
  return 1; /* retourne autre chose que 0 car ça s'est mal passé */
}

/* On commence par l'affichage du cadre */
printf("+-----------------+-----+\n");

/*
 * Et on boucle directement sur la condition de lecture correcte
 * du couple <nom,age> (en fait, sur la condition de lecture correcte
 * de 'age', mais comme il n'est pas possible de lire 'age' si la
 * lecture de 'nom' à échoué...)
 */

do {
  taille_lue = fscanf(entree, "%15s %u", nom, &age);

  if (taille_lue == 2) { /* la lecture s'est bien passée */
    ++nb; /* nombre de personnes + 1 */
    total += age; /* pour faire la moyenne plus tard */
    /* on vérifie si l'âge lu est le plus grand/petit lu jusqu'ici  */
    if (age_min > age) age_min = age; /* */
    if (age_max < age) age_max = age; /**/

    /* Affichage */
    /* le signe "-" permet d'aligner à gauche */
```

```
      printf("| %-15s | %3d |\n", nom, age);
    }
  } while (! feof(entree));

  /* Partie finale */

  fclose(entree);  /* ne pas oublier de fermer le  fichier ! */

  printf("+-----------------+-----+\n");
  printf("  âge minimum     : %3d\n", age_min);
  printf("  âge maximal     : %3d\n", age_max);

  printf("%d personnes, âge moyen : %4.1f ans\n", nb, total/nb);
  /* l'âge moyen est sur 4 chiffres dont un chiffre après la virgule */

  return 0;
}
```

Exercice 3 : fichiers binaires

1. Version initiale :

2.
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define TAILLE 1025

void demander_chaine(char* reponse, int taille)
{
  int taille_lue;

  do {
    fgets(reponse, taille, stdin);
    taille_lue = strlen(reponse) - 1;
    if ((taille_lue >= 0) && (reponse[taille_lue] == '\n'))
      reponse[taille_lue] = '\0';
  } while ((taille_lue < 1) && !feof(stdin));
}

int main(void) {

  char nom_fichier[FILENAME_MAX+1];
  char phrase[TAILLE];

  puts("Dans quel fichier voulez vous écrire ?\n");
```

```
  demander_chaine(nom_fichier, FILENAME_MAX+1);
  if (nom_fichier[0] == '\0') return 1;

  printf("Entrez une phrase (< %d caractères) :\n", TAILLE-1);
  demander_chaine(phrase, TAILLE);

  if (phrase[0] != '\0') {
    /* écriture et codage */
    int i;
    int taille;
    FILE* sortie;
    int ecrits;
    unsigned int code;

    /* On va écrire du binaire */
    sortie = fopen(nom_fichier, "wb");
    if (sortie == NULL) {
      fprintf(stderr,
              "Erreur : je ne peux pas ouvrir le fichier %s en écriture.\n",
              nom_fichier);
      return 1;
    }

    taille = strlen(phrase);
    for (i = 0; i < taille; ++i) {
      code = (unsigned char) phrase[i];
      code *= code;
      printf("%c -> %u -> %d\n", phrase[i], (unsigned char) phrase[i], code);
      ecrits = fwrite(&code, sizeof(int), 1, sortie);

      if (ecrits != 1)  {
        fprintf(stderr,
                "Erreur : je n'ai pas pu écrire plus que %d entiers (sur %d) !\n",
                i, taille);
        return 3;
      }
    }

    fclose(sortie);
  }

  return 0;
}
```

Exercice 4 : statistiques sur un fichier

Ce corrigé contient plusieurs aspects «pratiques» et devrait pour cela être étudié plus spécifiquement et bien

compris.

(fichier src/stat.c)

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

/* ===== CONSTANTES ===== */

/* nombre maximum de demandes en cas d'erreur */
#define NB_DEMANDES 3

/* taille maximum d'une Statistique : au plus 256 car il n'y a pas plus
   que 256 char. */
#define TAILLE 256

/* bornes sur les caractères à prendre en compte */
unsigned char start =  32;
unsigned char stop  = 253;


/* ===== DEFINITIONS DE TYPES ====== */

typedef unsigned long int Statistique[TAILLE];


/* ===== FONCTIONS ====== */
FILE* demander_fichier();

void initialise_statistique(Statistique a_initialiser);
/* Rappel : les tableaux sont toujours passés par référence. Pas
   besoin de pointeur supplémentaire ici */

unsigned long int collecte_statistique(Statistique a_remplir,
                                       FILE* fichier_a_lire);

void affiche(Statistique a_afficher, unsigned long int total,
             unsigned short int taille);

/* ================================================================== */
int main(void)
{
  FILE* fichier = demander_fichier();
  if (fichier == NULL) {
    printf("=> j'abandonne !\n");
    return 1;
  } else {
```

```
    Statistique stat;
    initialise_statistique(stat);
    affiche(stat, collecte_statistique(stat, fichier), stop - start + 1);
    fclose(fichier);
  }

  return 0;
}

/* ========================================================================
 * Fonction demander_fichier
 * ------------------------------------------------------------------------
 * In:   Un fichier (par référence) à ouvrir.
 * Out:  Ouvert ou non ?
 * What: Demande à l'utilisateur (au plus NB_DEMANDES fois) un nom de fichier
 *       et essaye de l'ouvrir en lecture.
 * ======================================================================== */
FILE* demander_fichier()
{
  FILE* f = NULL;
  char nom_fichier[FILENAME_MAX+1];
  size_t taille_lue = 0;
  unsigned short int nb = 0;

  do {
    ++nb;

    /* demande le nom du fichier */
    do {
      printf("Nom du fichier à lire : "); fflush(stdout);
      fgets(nom_fichier, FILENAME_MAX+1, stdin);
      taille_lue = strlen(nom_fichier);
      if ((taille_lue >= 1) && (nom_fichier[taille_lue-1] == '\n'))
        nom_fichier[--taille_lue] = '\0';
    } while ((taille_lue == 0) && !feof(stdin));

    if (nom_fichier[0] == '\0') {
      return NULL;
    }

    /* essaye d'ouvrir le fichier */
    f = fopen(nom_fichier, "r");

    /* est-ce que ça a marché ? */
    if (f == NULL) {
      printf("-> ERREUR, je ne peux pas lire le fichier \"%s\"\n",
             nom_fichier);
```

```
    } else {
      printf("-> OK, fichier \"%s\" ouvert pour lecture.",
             nom_fichier);
    }
  } while ((f == NULL) && (nb < NB_DEMANDES));

  /* la valeur de retour est le résultat du test entre (): 0 ou 1 */
  return f;
}


/* ======================================================================
 * Fonction initialiser_statistique
 * ----------------------------------------------------------------------
 * In:   Une Statistique à initialiser.
 * What: Initialiser tous les éléments d'une Statistique à zéro.
 * ====================================================================== */
void initialise_statistique(Statistique stat)
{
  int i;
  for (i = 0; i < TAILLE; ++i) {
    stat[i] = 0;
  }
}

/* ======================================================================
 * Fonction collecte_statistique
 * ----------------------------------------------------------------------
 * In:   Une Statistique à remplir et le fichier à lire.
 * Out:  Le nombre d'éléments comptés dans la Statistique.
 * What: Lit tous les caractères dans le fichier et compte dans la Statistique
 *       combien de fois chaque caractère apparait dans le fichier.
 * ====================================================================== */
unsigned long int collecte_statistique(Statistique stat, FILE* f)
{
  int c;                    /* le caractère lu             */
  unsigned long int nb = 0; /* le nombre d'éléments comptés */

  while ((c = getc(f)) != EOF) {
    /* est-ce que le caractère lu est dans l'intervalle qu'on étudie ? */
    if (( ((unsigned char) c) >= start) &&
        ( ((unsigned char) c) <= stop ) ) {
      ++(stat[c - start]); /* on incrément la statistique pour ce caractère */
      ++nb; /* on incrémente le nombre total d'éléments comptés */
    }
  }
```

```
  return nb;
}

/* ==========================================================================
 * Fonction affiche
 * --------------------------------------------------------------------------
 * In:   La Statistique à afficher, le nombre par rapport auquel on affiche
 *       les pourcentages (si 0 recalcule ce nombre comme la somme des
 *       éléments) et la taille du tableau.
 * What: Affiche tous les éléments d'une Statistique (valeurs absolue et
 *       relative).
 * ========================================================================== */
void affiche(Statistique stat, unsigned long int nb, unsigned short int taille)
{
  unsigned short int i;

  if (nb == 0) { /* on doit calculer la somme si nb == 0 */
    for (i = 0; i < taille; ++i)
      nb += stat[i];
  }

  printf("STATISTIQUES :\n");
  for (i = 0; i < taille; ++i) {
    /* on n'affiche que les résultats pour des statistiques supérieures à 0 */
    if (stat[i] != 0) {
      printf("%c : %10lu - %6.2f%%\n", (char) (i+start), stat[i],
             100.0 * stat[i] / (double) nb);
    }
  }
}
```

Dernière mise à jour : Dernière mise à jour le 8 mars 2016

Last modified: Tue Mar 8, 2016