Corrigé TP5

[a] `argc` représente le nombre d'arguments passés au programme. Le premier argument représente toujours le nom du programme.

[b] Tout dépend de votre implémentation

[c] Voici une implémentation possible.

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

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

int main (int argc, char *argv[]) {
   int i = 0, somme = 0;

   if (argc == 1) {
      printf ("USAGE: sum NUMBER ...\n");
      return 1;
   }

   for (i=1; i<argc; i++) {
      printf ("%s ", argv[i]);
      if (i < argc - 1) {
         printf ("+ ");
      }

      somme += atoi (argv[i]);
   }

   printf ("= %d\n", somme);
   return 0;
}
```

[d] Le message d'erreur en cas de nombre incorrect d'arguments dépend de votre implémentation. Si `fopen` n'arrive pas à ouvrire le fichier (fichier inexistent, droit de lecture manquant ...) alors la valeur renvoyée est `NULL`. On affiche donc le message d'erreur comme dans le programme original `lecture_fich.c`.
Voici une implémentation possible.

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>

int main (int argc, char *argv[]) {
   FILE *file = NULL;
   char *filename = NULL;
   char character;

   if (argc != 2) {
      printf ("USAGE my_cat FILE\n");
      return 1;
   }

   filename = argv[1];
   file = fopen (filename, "r");
```

```
    if (file == NULL) {
        printf ("%s n'a pas pu être ouvert en lecture\n", filename);
        return 1;
    }

    fscanf (file, "%c", &character);
    while ( !feof (file) ) {
        printf ("%c", character);
        fscanf (file, "%c", &character);
    }

    fclose (file);
    return 0;
}
```

[e] Une implémentation possible du programme copiant un fichier dans un autre (version sans l'exercice complémentaire).

```
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>

int main (int argc, char *argv[]) {
    char *source_filename = NULL, *destination_filename = NULL;
    FILE *source          = NULL, *destination          = NULL;
    char character;

    if (argc != 3) {
        printf ("USAGE my_cp SRC DEST\n");
        return 1;
    }

    source_filename      = argv[1];
    destination_filename = argv[2];

    source               = fopen (source_filename, "r");
    destination          = fopen (destination_filename, "w");

    if (source == NULL) {
        printf ("Impossible d'ouvrir en lecture : %s\n", source_filename);
        return 1;
    }

    if (destination == NULL) {
        printf ("Impossible d'ouvrir en écriture : %s\n", destination_filename);
        return 1;
    }

    fscanf (source, "%c", &character);
    while ( !feof (source) ) {
        fprintf (destination, "%c", character);
        fscanf (source, "%c", &character);
    }

    fclose (source);
    fclose (destination);
    return 0;
}
```

On rajoute maintenant le fichier de destination comme étant optionnel (exercice complémentaire).

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>

int main (int argc, char *argv[]) {
   char *source_filename = NULL, *destination_filename = NULL;
   FILE *source          = NULL, *destination          = NULL;
   char character;

   if (2 > argc || 3 < argc) {
      printf ("USAGE my_cp SRC [DEST]\n");
      return 1;
   }

   source_filename = argv[1];
   source          = fopen (source_filename, "r");

   if (source == NULL) {
      printf ("Impossible d'ouvrir en lecture : %s\n", source_filename);
      return 1;
   }

   if (argc == 3) {
      destination_filename = argv[2];
      destination          = fopen (destination_filename, "w");

      if (destination == NULL) {
         printf ("Impossible d'ouvrir en écriture : %s\n", destination_filename);
         return 1;
      }

   } else {
      destination = stdout;
   }

   fscanf (source, "%c", &character);
   while ( !feof (source) ) {
      fprintf (destination, "%c", character);
      fscanf (source, "%c", &character);
   }

   fclose (source);
   fclose (destination);
   return 0;
}
```

[f] La commande `cat *.c` affiche successivement le contenu de tous les fichiers correspondant au pattern `*.c`.

[g] La commande `cat` sans argument recopie dans le terminal les caractères que l'on tape au clavier. On remarque que les caractères ne s'affichent que lorsque l'on tape entrée. C'est parce que le terminal n'écrit effectiement les caractères que l'on a tapé dans `stdin` que lorsque l'on tape entrée.

[h] Voici une implémentation possible de cat simplifiée (sans l'exercice complémentaire).

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

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

void function_cat (char *filename) {
   char character;
   FILE *file = fopen (filename, "r");
```

```
    if (file == NULL) {
        printf ("Impossible d'ouvrir en lecture : %s\n", filename);
        exit (1);
    }

    fscanf (file, "%c", &character);
    while ( !feof (file) ) {
        printf ("%c", character);
        fscanf (file, "%c", &character);
    }

    fclose (file);
}

int main (int argc, char *argv[]) {
    int i = 1;

    if (2 > argc) {
        printf ("USAGE my_cat FILE ...\n");
        return 1;
    }

    while (i < argc) {
        function_cat (argv[i]);
        i++;
    }

    return 0;
}
```

Voici une implémentation possible de cat avec le même comportement lorsqu'aucun argument n'est entré (avec l'exercice complémentaire).

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

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

void function_cat (char *filename) {
   char character;
   FILE *file = fopen (filename, "r");

   if (file == NULL) {
      printf ("Impossible d'ouvrir en lecture : %s\n", filename);
      exit (1);
   }

   fscanf (file, "%c", &character);
   while ( !feof (file) ) {
      printf ("%c", character);
      fscanf (file, "%c", &character);
   }

   fclose (file);
}

void function_cat_without_argument (void) {
   char character;

   /* Tant que l'entrée standard est vide, le programme attend */
   fscanf (stdin, "%c", &character);

   /* feof ne considère que l'on a atteint la fin de l'entrée standard
    * que lorsque l'utilisateur a tapé Ctrl + D au début d'une ligne */
   while ( !feof (stdin) ) {

      /* Du point de vue du programme, on imprime immédiatement un
       * caractère lu. Toutefois, à l'exécution, les caractères ne
       * s'affichent que lorsqu'on appuie sur entrée. Cela est dû au
       * fait que le terminal n'écrit dans stdin que lorsqu'on
       * appuie effectivement sur entrée. */
      printf ("%c", character);
      fscanf (stdin, "%c", &character);
   }
}

int main (int argc, char *argv[]) {
   int i = 1;

   if (2 > argc) {
      function_cat_without_argument ();
      return 0;
   }

   while (i < argc) {
      function_cat (argv[i]);
      i++;
   }

   return 0;
}
```

[i] On retrouve l'intervalle des caractères imprimables à l'aide de la commande ascii ou grâce au manuel. Voici une implémentation possible du programme comptant le nombre d'occurence de chaque caractère.

```c
/**
 * Fichier de démonstration pour l'Université Grenoble Alpes
 *
 * @autor <corentin@marciau.fr>
 */

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

/* Caractères imprimables : de 33 ('!') à 126 ('~') soit 94 caractères */
#define NB_CARACTERES_IMPRIMABLES 94

int main (int argc, char *argv[]) {
   int i = 0;
   char character;
   char *filename = NULL;
   FILE *file     = NULL;

   /* Le tableau occurences est initialisé avec des 0
    *
    * Chaque case fait référence au caractère correspondant
    * à son indice dans la table ASCII à partir du caractère 33.
    *
    * Ex : occurence[0] ==> ASCII(33) '!'
    *      occurence[1] ==> ASCII(34) '"'
    *      occurence[2] ==> ASCII(35) '#'
    *      ...
    */
   int occurences[NB_CARACTERES_IMPRIMABLES] = {0};

   if (argc != 2) {
      printf("USAGE: wc FILE\n");
      return 1;
   }

   filename = argv[1];
   file     = fopen (filename, "r");

   if (file == NULL) {
      printf ("Impossible d'ouvrir en lecture : %s\n", filename);
      return 1;
   }

   fscanf (file, "%c", &character);
   while ( !feof (file) ) {
      if (32 < character && 127 > character) {
         occurences[character - 33]++;
      }

      fscanf (file, "%c", &character);
   }

   for (i=0; i<NB_CARACTERES_IMPRIMABLES; i++) {
      printf ("Nombre d'occurences du caractère '%c' : %d\n", i+33, occurences[i]);
   }

   return 0;
}
```

[j] Le nombre de fichiers contenant `gcc` dans `/opt` dépend de chaque ordinateur. Sur Turing, on trouve 20 versions de gcc dans `/opt`.

NB : Le répertoire `/opt` (pour optional) est un bon endroit pour installer manuellement un programme (c'est à dire sans passer par un gestionnaire de paquet comme `apt-get` ou `rpm`).