```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <errno.h>  /* Pour la variable systeme "errno" */
 4  #include <string.h> /* Pour strerror */
 5
 6  int main(int argc, char * argv[]) {
 7    FILE * f;
 8    int i;
 9    if(argc < 2) {
10      fprintf(stderr, "usage : %s file1 [file2 [...]]\n", argv[0]);
11      exit(1);
12    }
13    for(i = 1; i < argc; i++) {
14      if( (f = fopen(argv[i], "r")) == NULL)
15        fprintf(stderr, "Impossible d'ouvrir le fichier %s\nErreur %d : %s\n",
16                argv[i], errno, strerror(errno));
17      else {
18        fprintf(stderr, "Fichier %s ouvert et ... ", argv[i]);
19        fclose(f);
20        fprintf(stderr, "ferme\n");
21      }
22    }
23    return 0;
24  }
```

FIG. 1 – Ouverture et fermeture de fichiers

EXTRAIT DE **man fopen**

```
#include<stdio.h>
FILE *fopen (const char *path, const char *mode);
```

DESCRIPTION

La fonction fopen ouvre le fichier dont le nom est contenu dans la chaine pointée par path et lui associe un flux.
L'argument mode pointe vers une chaîne commençant par l'une des séquences suivantes (d'autres caractères peuvent suivre la séquence) :

- **r** Ouvre le fichier en lecture. Le pointeur de flux est placé au début du fichier.
- **r+** Ouvre le fichier en lecture et écriture. Le pointeur de flux est placé au début du fichier.
- **w** Ouvre le fichier en écriture. Le fichier est créé s'il n'existait pas. S'il existait déjà, sa longueur est ramenée à 0. Le pointeur de flux est placé au début du fichier.
- **w+** Ouvre le fichier en lecture et écriture. Le fichier est créé s'il n'existait pas. S'il existait déjà, sa longueur est ramenée à 0. Le pointeur de flux est placé au début du fichier.
- **a** Ouvre le fichier en ajout (écriture à la fin du fichier). Le fichier est créé s'il n'existait pas. Le pointeur de flux est placé à la fin du fichier.
- **a+** Ouvre le fichier en lecture et ajout (écriture en fin de fichier). Le fichier est créé s'il n'existait pas. Le pointeur de flux est placé à la fin du fichier.

En cas de réussite **fopen** renvoie un pointeur sur un fichier, de type **FILE**. Sinon, elle renvoie **NULL** et **errno** contient le code d'erreur.

EXTRAIT DE **man fread**

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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>  /* Pour la variable systeme "errno" */
4  #include <string.h> /* Pour strerror */
5
6  int parcourir(FILE * f) {
7    int taille = 0;
8    char c;
9    while ( fread (&c, 1, 1, f) ) taille++;
10   /* CECI EST EQUIVALENT A LA LIGNE PRECEDENTE
11     while ( !feof(f) ) { fread (&c, 1, 1, f); taille++;} */
12   return taille;
13 }
14
15 int main(int argc, char * argv[]) {
16   FILE * f;
17   int i;
18   if(argc < 2) {
19     fprintf(stderr, "usage : %s file1 [file2 [...]]\n", argv[0]);
20     exit(1);
21   }
22   for(i = 1; i < argc; i++) {
23     if( (f = fopen(argv[i], "r")) == NULL)
24       fprintf(stderr, "Impossible d'ouvrir le fichier %s\nErreur %d : %s\n",
25               argv[i], errno, strerror(errno));
26     else {
27       fprintf(stderr, "Fichier %s ouvert, parcouru (%d octets) et ... ",
28               argv[i], parcourir(f));
29       fclose(f);
30       fprintf(stderr, "ferme\n");
31     }
32   }
33   return 0;
34 }
```

FIG. 2 – Ouvrir, parcourir et fermer un fichier

```
size_t fread (void *ptr, size_t size, size_t nmemb, FILE *stream);
size_t fwrite (const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

DESCRIPTION

La fonction fread lit nmemb éments de données, chacun d'eux représentant size octets de long, depuis le flux pointé par stream, et les stocke à l'emplacement pointé par ptr.

La fonction fwrite écrit nmemb éléments de données, chacun d'eux représentant size octet de long, dans le flux pointé par stream, après les avoir lus depuis l'emplacement pointé par ptr.

VALEUR RENVOYÉE

fread et fwrite renvoient le nombre d'éléments correctement lus ou écrits (et non pas le nombre d'octets). Si une erreur se produit, ou si la fin du fichier est atteinte en lecture, le nombre renvoyé est plus petit que nmemb et peut même être nul.

fread traite la fin du fichier comme une erreur, et l'appelant devra appeler feof(3) ou ferror(3) pour distinguer ce cas.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>  /* Pour la variable systeme "errno" */
4  #include <string.h> /* Pour strerror */
5
6  #include <sys/types.h>
7  #include <sys/stat.h>
8  #include <unistd.h>
9
10 long parcourir(FILE * f, long taille) {
11   long i;
12   for(i = 0; i < taille; i++)
13     fseek(f, i, SEEK_SET);
14   return i;
15 }
16
17 int main(int argc, char * argv[]) {
18   FILE * f;
19   int i;
20   if(argc < 2) {
21     fprintf(stderr, "usage : %s file1 [file2 [...]]\n", argv[0]);
22     exit(1);
23   }
24   for(i = 1; i < argc; i++) {
25     if( (f = fopen(argv[i], "r")) == NULL)
26       fprintf(stderr, "Impossible d'ouvrir le fichier %s\nErreur %d : %s\n",
27               argv[i], errno, strerror(errno));
28     else {
29       long taille = 0;
30       struct stat buf;
31       stat(argv[i], &buf);
32       taille = (long) (buf.st_size);
33       fprintf(stderr, "Fichier %s ouvert, parcouru (%ld octets) et ... ",
34               argv[i], parcourir(f, taille));
35       fclose(f);
36       fprintf(stderr, "ferme\n");
37     }
38   }
39   return 0;
40 }
```

FIG. 3 – Ouvrir, parcourir et fermer un fichier V2 ...

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>  /* Pour la variable systeme "errno" */
4  #include <string.h> /* Pour strerror */
5
6  /* Pour stat */
7  #include <sys/types.h>
8  #include <sys/stat.h>
9  #include <unistd.h>
10
11 void remplacer(char * s, int l, char c1, char c2) {
12   int i;
13   for(i = 0; i < l; i++)
14     if(s[i] == c1) s[i] = c2;
15 }
16
17 int main(int argc, char * argv[]) {
18   FILE * f1, * f2;
19   struct stat buf;
20   char * txt = NULL;
21   if(argc != 3) {
22     fprintf(stderr, "usage : %s src_file dst_file\n", argv[0]);
23     exit(1);
24   }
25   if( (f1 = fopen(argv[1], "r")) == NULL) {
26     fprintf(stderr, "Impossible d'ouvrir le fichier %s\nErreur %d : %s\n",
27             argv[1], errno, strerror(errno));
28     exit(2);
29   }
30   if( (f2 = fopen(argv[2], "w")) == NULL) {
31     fprintf(stderr, "Impossible d'ouvrir (ou creer) le fichier %s\nErreur %d : %s\n",
32             argv[2], errno, strerror(errno));
33     exit(3);
34   }
35   stat(argv[1], &buf);
36   txt = malloc((int)(buf.st_size) * sizeof txt[0]);
37   fread(txt, 1, buf.st_size, f1);
38   remplacer(txt, buf.st_size, ' ', '\n');
39   fwrite(txt, 1, buf.st_size, f2);
40   fclose(f1);
41   fclose(f2);
42   return 0;
43 }
```

FIG. 4 – Un mini Rechercher-Remplacer