Programmation langage C
Section 11 : Fichiers

Présentation de **Kevin TRANCHO**

dispensé en classe de seconde année

à l'**ESGI** Paris
(Année scolaire 2022 - 2023)

Introduction

Comment sauvegarder des données et les récupérer au prochain lancement de notre programme ?

Proposition de solution :

- Utilisation de fichiers enregistrés dans la mémoire de stockage de la machine.
- Un fichier peut se gérer à l'aide d'un type `FILE *`.
- Un fichier s'ouvre par `fopen` de `stdio`.
- Un fichier se ferme par `fclose`.

fopen :

```
fopen(/* chemin du fichier */, /* mode d'ouverture */)
```

- Le premier argument est le chemin du fichier sur l'espace disque.
- Le second argument est le mode d'ouverture du fichier, ceci dépend si l'on souhaite lire, écrire, s'y déplacer et autre :

Mode	Lecture	Écriture	Création	Effacement	Ajout en fin
"r"	√				
"w"		√	√	√	
"a"		√	√		√
"r+"	√	√			
"w+"	√	√	√	√	
"a+"	√	√	√		√

Utilisation d'un fichier :

```c
FILE * fichier = NULL;
/* Tentative d'ouverture / création d'un fichier */
if((fichier = fopen(/* chemin */, /* mode */)) == NULL)
↪  {
  /* Gestion de l'impossibilité d'ouverture */
}
/* Opérations avec le fichier */
/* ... */
/* Fermeture du fichier */
fclose(fichier);
fichier = NULL;
```

Exemple pour créer un fichier vide $\star^1$:

```c
FILE * fichier = NULL;
if((fichier = fopen("MonFichier.txt", "w")) == NULL) {
  printf("Erreur de création de mon fichier.\n");
  exit(EXIT_FAILURE);
}
printf("Fichier créé avec succès.\n");
fclose(fichier);
fichier = NULL;
exit(EXIT_SUCCESS);
```

Lire et écrire dans un fichier

- Version formatée :
  - Écriture : `fprintf`.
  - Lecture : `fscanf`.
- Par caractères :
  - Écriture : `fputc`.
  - Lecture : `fgetc`.
- En binaire :
  - Écriture : `fwrite`.
  - Lecture : `fread`.

fprintf : écriture simple

```
FILE * fichier = fopen("test.txt", "w");

fprintf(fichier, "Hello fichier !\n");

fclose(fichier);
```

fprintf : écriture formatée

```c
int renseignerInfos(const char * pseudo) {
  int age;
  printf("Quel est votre âge ? ");
  scanf("%d", &age);
  FILE * infos = fopen(pseudo, "w");
  /* sauvegarde la valeur de age */
  fprintf(infos, "%d\n", age);
  fclose(infos);
}
```

fscanf

```c
int lireInfos(const char * pseudo, int * age) {
  FILE * infos = NULL;
  if((infos = fopen(pseudo, "r")) == NULL) {
    return 0; /* l'utilisateur est inconnu */
  }
  /* récupère la valeur de age */
  fscanf(infos, "%d", age);
  fclose(infos);
  return 1;
}
```

fgetc : lecture des caractères d'un fichier $\star^2$

```c
FILE * fichier = fopen("message.txt", "r");
int caractere;
/* tant qu'on lit un caractère dans le fichier */
while((caractere = fgetc(fichier)) != EOF) {
  putchar(caractere); /* on affiche le caractere */
}
fclose(fichier);
```

fgetc et fputc : chiffre de César $\star^3$

```c
FILE * input = fopen("message.txt", "r");
FILE * output = fopen("resultat.txt", "w");
int caractere;
const int cle = 5;
/* tant qu'on lit un caractère dans le fichier */
while((caractere = fgetc(input)) != EOF) {
  /* on le code s'il est alphabéthique */
  if(caractere >= 'a' && caractere <= 'z')
    caractere = (caractere - 'a' + cle) % 26 + 'a';
  else if(caractere >= 'A' && caractere <= 'Z')
    caractere = (caractere - 'A' + cle) % 26 + 'A';
  /* on l'écrit dans le fichier de sortie*/
  fputc(caractere, output);
}
fclose(input);
fclose(output);
```

fread et fwrite

- Passage mode binaire par ajout d'un 'b' dans le mode d'ouverture.
- Arguments de `fread` et `fwrite` :
  - Pointeur sur les données à lire / écrire.
  - Taille d'un élément.
  - Nombre d'éléments.
  - Pointeur sur le fichier où lire / écrire.

  Renvoient le nombre d'éléments lus / écrits.

fwrite : écrire tableau d'entiers $\star^4$

```
int sauvegarderListe(const char * filepath, const int * liste,
↪  int taille) {
  FILE * output = fopen(filepath, "wb");
  /* écriture de la taille : une variable */
  if(fwrite(&taille, sizeof(int), 1, output) != 1) {
    printf("Erreur écriture taille\n");
    return 0;
  }
  /* écriture de la liste : un tableau */
  if(fwrite(liste, sizeof(int), taille, output) != taille) {
    printf("Erreur écriture liste\n");
    return 0;
  }
  fclose(output);
  return 1;
}
```

fread : lire tableau d'entiers

```c
int chargerListe(const char * filepath, int ** liste, int * taille) {
  FILE * input = fopen(filepath, "rb");
  /* lecture de la taille : nécessaire à l'allocation */
  if(fread(taille, sizeof(int), 1, input) != 1) {
    printf("Erreur lecture taille\n");
    return 0;
  }
  /* allocation de la liste */
  if((*liste = (int *)malloc(sizeof(int) * *taille)) == NULL) {
    printf("Erreur allocation liste\n");
    return 0;
  }
  /* lecture des éléments de la liste */
  if(fread(*liste, sizeof(int), *taille, input) != *taille) {
    printf("Erreur lecture liste\n");
    return 0;
  }
  fclose(input);
  return 1;
```

Déplacement dans un fichier

- `ftell(FILE*)` indique la position actuelle du curseur dans un fichier.
- `rewind(FILE*)` rembobine le fichier au début.
- `fseek(FILE*, offset, repere)` déplace le curseur dans un fichier. `repere` peut prendre les valeurs :
  - `SEEK_SET` : début du fichier.
  - `SEEK_CUR` : position actuelle dans le fichier.
  - `SEEK_END` : fin du fichier.

Exemple : lire des phrases dans un fichier $\star^5$

```c
int lirePhrase(FILE * file, long * start, long * end) {
  int car;
  while((car = fgetc(file)) != EOF) {
    if(! carInChaine(car, "\t\n ")) {
      break;
    }
  }
  if(start) /* on récupère la position du début de la phrase */
    *start = ftell(file) - 1;
  do {
    if(car == '.') {
      break;
    }
  } while((car = fgetc(file)) != EOF);
  if(end) /* on récupère la position de la fin de la phrase */
    *end = ftell(file);
  return car != EOF;
}
```

Exemple : lire des phrases dans un fichier

```c
void afficherPortionFichier(FILE * file, long start, long end) {
  int car;
  /* on se replace dans le fichier à la position indiquée */
  fseek(file, start, SEEK_SET);
  while(ftell(file) != end) { putchar(fgetc(file)); }
}
int main() {
  FILE * file = fopen("message.txt", "r");
  int i; long start, end;
  for(i = 0; lirePhrase(file, NULL, NULL); ++i);
  printf("%d phrases.\n", i);
  rewind(file); /* on rembobine le fichier au début */
  for(i = 0; lirePhrase(file, &start, &end); ++i) {
    printf(" - Phrase %d (%ld caracteres): ", i + 1, end - start);
    afficherPortionFichier(file, start, end);
    printf("\n");
  }
  fclose(file);
  exit(EXIT_SUCCESS);
```

FILE* uniquement des fichiers ?

Plus généralement, `FILE*` permet de gérer des flux d'entrée / sortie. Vous en avez déjà utilisé sans le savoir :

- `stdout` : flux de sortie standard (déjà utilisé via `printf`).
- `stdin` : flux d'entrée standard (déjà utilisé via `scanf`).
- `stderr` : flux de sortie standard d'erreurs.

stdout

```c
printf("Par printf\n");
fprintf(stdout, "Par fprintf\n");
```

stdin

```c
int nombreScanf, nombreFscanf;
scanf("%d", &nombreScanf);
fscanf(stdin, "%d", &nombreFscanf);
printf("%d %d\n", nombreScanf, nombreFscanf);
```

stderr $\star^6$

```
FILE * fichier = NULL;
if((fichier = fopen("fichier_a_ne_pas_creer", "r"))
↪  == NULL) {
  fprintf(stderr, "Erreur main() :
  ↪  \"fichier_a_ne_pas_creer\" est introuvable\n");
  exit(EXIT_FAILURE);
}
fclose(fichier);
```

stderr : redirection dans fichier de logs

```
# gcc -o prog main.c
# ./prog
Erreur main() : "fichier_a_ne_pas_creer" est
 ↪  introuvable
# ./prog 2>log
# cat log
Erreur main() : "fichier_a_ne_pas_creer" est
 ↪  introuvable
```

Questions

Avez-vous des questions ?

Exercices

- Travailler sur les exercices sur les fichiers (section 11) du support de cours.
- Si les exercices de la section 11 sont terminés :
  - Avancer sur les sections 12 et 13.
  - Si cours terminé : Avancer sur le projet.
  - Si projet terminé avec certitude de 21 / 20 : le pousser plus loin.

Annexe


When you did all the work in a group project
PRODUCER
DIRECTOR
ACTOR
EDITOR