Algorithmique et programmation en C - Les fichiers.

François Delbot

Maître de conférences de l'Université Paris Ouest Nanterre la Défense
Membre de l'équipe de recherche opérationnelle du LIP6

April 4, 2018

Prérequis

1. Avoir suivi le premier cours sur les fichiers.
2. Comprendre comment formater une chaine de caractères et l'afficher grâce à la fonction printf.
3. Avoir suivi le cours sur les pointeurs.
4. Avoir suivi le cours sur les structures.

Besoin des fichiers

Jusqu'à présent, nous nous contentions d'utiliser les entrées depuis le clavier et les sorties vers l'écran.

Problème : les données n'étaient pas persistantes.

Nous allons utiliser les fichiers pour sauvegarder/restituer des données entre un **programme** et une **mémoire persistante**, par exemple un disque dur.

Mémoire de masse et mémoire vive.

Vitesse :

1. La mémoire vive est très rapide (RAM).
2. La mémoire de masse est très lente comparativement à la mémoire vive.

Taille :

1. La mémoire vive peut contenir une quantité de données limitée.
2. La mémoire de masse peut contenir une quantité de données très importante.

Coût :

1. La mémoire vive coûte très cher.
2. La mémoire de masse coûte peu cher.

Mémoire de masse et mémoire vive.

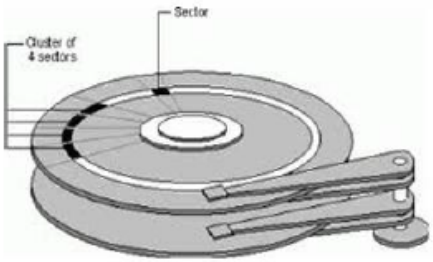
Donc...

Le prix d'un ordinateur doit rester raisonnable, mais nous avons besoin d'énormément de mémoire tout en calculant le plus rapidement possible.

1. La mémoire vive va servir pour les calculs.
2. La mémoire de masse pour le stockage.
3. Si on manque de mémoire vive, un mécanisme de swap se met en place.

Exemple d'interaction d'un programme avec un disque dur.

Disque Dur



Entrées/Sorties

Programme

- Ouvrir fichier X
- Lire D du fichier X
- Fermer fichier X
- R = Traitement (D)
- Ouvrir fichier Y
- Ecrire R dans fichier Y
- Fermer fichier Y

Mécanique de l'utilisation des fichiers

Temps d'accès à la mémoire L'accès à un fichier présent dans la mémoire de stockage de masse demande du temps :

1. Mécanique
2. Temps de transfert

Plus précisément, c'est l'accès (phases d'écriture ou de lecture) à la mémoire qui est couteux.

Mécanique de l'utilisation des fichiers
Taille d'un bloc

1. L'accès en écriture sur une mémoire persistante est **TRÈS** couteux.
2. L'écriture se fait bloc par bloc.
3. La taille minimum d'un bloc est d'un octet.
4. La taille maximum d'un bloc est donnée par la fonction stat (champs st_blksize). Sous Linux :

```
1     #include <sys/vfs.h>
```

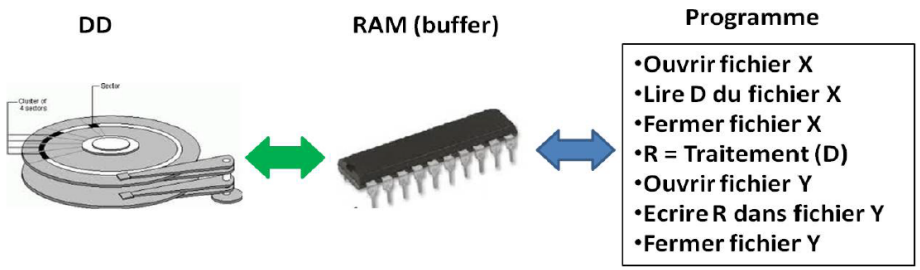
5. Il est donc préférable d'écrire des blocs les plus grands possibles pour **minimiser le nombre d'accès**.

Mécanique de l'utilisation des fichiers

minimiser le nombre d'accès à la mémoire de masse.

Buffer

Un buffer est une mémoire tampon, une zone de mémoire utilisée pour stocker temporairement des données.



On va utiliser un buffer (dont la taille est d'un bloc[1]) pour les opérations d'accès. Par exemple, lors d'une écriture, les données seront enregistrées dans le buffer. Lorsque le buffer est plein, une opération d'écriture est déclenchée, ne réalisant ainsi qu'un seul accès.

[1] mensonge pédagogique

Les données nécessaires à la manipulation d'un fichier

Le type FILE

En langage C, les informations nécessaires à maintenir l'association

programme $\Leftrightarrow$ buffer $\Leftrightarrow$ mémoire de persistante

sont décrites dans une structure de type FILE, définie dans stdio.h.

Parmi les informations stockées dans cette structure on trouve :

- Le numéro du fichier.
- Le type d'ouverture (lecture/écriture).
- l'adresse du buffer associé.
- La position du curseur de lecture.
- La position du curseur d'écriture.

Les données nécessaires à la manipulation d'un fichier

Pour utiliser un fichier, il faut donc commencer par déclarer une variable de type FILE, ou plus exactement un pointeur de type FILE. Exemple de fonction principale :

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

int main(void)
{
    FILE * nomDuPointeur = NULL;

    return EXIT_SUCCESS;
}
```

Ouverture d'un fichier
La fonction fopen

La fonction fopen

Située dans la bibliothèque stdio.h, la fonction fopen ouvre un fichier dont le nom (le chemin d'accès) est passé en argument sous la forme d'une chaine de caractères. Le second argument détermine le mode d'ouverture du fichier.

```
1   FILE *fopen(const char *pathname, const char *mode);
```

Si l'ouverture échoue, la fonction retourne NULL, sinon un pointeur de type FILE *.

Ouverture d'un fichier

Les modes d'ouverture de fichiers

Mode	Description
r	Ouverture du fichier en mode lecture. Le curseur est positionné au début du fichier.
r+	Ouverture du fichier en mode lecture ET écriture. Le curseur est positionné au début du fichier.
w	Ouvre un fichier en écriture. Si le fichier existe déjà, son contenu est supprimé. Si le fichier n'existe pas encore, il est créé. Le curseur est positionné au début du fichier.
w+	Ouvre un fichier en lecture ET en écriture. Si le fichier existe déjà, son contenu est supprimé. Si le fichier n'existe pas encore, il est créé. Le curseur est positionné au début du fichier.
a	Ouverture du fichier en mode écriture. Si le fichier n'existe pas encore, il est créé. Le curseur est positionné à la fin du fichier.

Vous trouverez plus d'informations ici :

`http://man7.org/linux/man-pages/man3/fopen.3.html`

Ouverture d'un fichier

Exemple de création de fichier

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

int main(void)
{
    FILE *fp = NULL;
    fp = fopen("miage.txt","w");
    if(fp == NULL)
    {
        printf("Erreur lors de la creation du fichier.");
        exit(EXIT_FAILURE);
    }
    printf("Creation du fichier reussie.");
    return EXIT_SUCCESS;
}
```

Erreurs les plus fréquentes :

1. Le fichier existe déjà et vous n'avez pas le droit d'y accéder.
2. Le chemin du fichier n'est pas correct.

Ouverture d'un fichier

Exemple d'ouverture de fichier en lecture seule

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

int main(void)
{
    FILE *fp;
    fp = fopen("miage.txt","r");
    if(fp == NULL)
    {
        printf("Erreur lors de l'ouverture du fichier.");
        exit(EXIT_FAILURE);
    }
    printf("Ouverture du fichier reussie.");
    return EXIT_SUCCESS;
}
```

Erreurs les plus fréquentes :

1. Le fichier n'existe pas.
2. Le chemin du fichier n'est pas correct.

Fermeture d'un fichier
La fonction fclose

La fonction fclose

Située dans la bibliothèque stdio.h, la fonction fclose ferme un fichier pointé par le pointeur passé en argument. Toutes les données présentes dans le buffer sont écrites.

```
1   int fclose(FILE *fp);
```

Si la fermeture réussie, alors la fonction retourne 0, sinon, elle retourne EOF.

Attention : Si vous tentez de fermer un fichier non ouvert à l'aide de la fonction fopen ou déjà fermé entrainera une erreur.

Fermeture d'un fichier

Exemple d'utilisation de la fonction fclose

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

int main(void)
{
    FILE *fp;
    fp = fopen("miage.txt","w");
    if(fp == NULL)
    {
        printf("Erreur lors de la creation du fichier.");
        exit(EXIT_FAILURE);
    }
    printf("Creation du fichier reussie.");
    fclose(fp);
    return EXIT_SUCCESS;
}
```

Fermeture d'un fichier

Le buffer

1. Lorsque nous écrivons dans un fichier, nous écrivons en fait dans le buffer.
2. Lorsque le buffer est suffisamment rempli, une opération d'écriture est réalisée. Le contenu du buffer est effectivement transféré dans le fichier sur la mémoire persistante et le buffer est vidé.
3. Si le programme se termine alors que le buffer contient des données, alors elles ne sont pas automatiquement transférées sur la mémoire persistante.
4. La fermeture du fichier force le transfert des données présentes dans le buffer vers le fichier dans la mémoire persistante.

Écriture dans un fichier
La fonction fprintf

La fonction fprintf

La fonction fprintf, définie dans stdio.h se comporte de la même manière que la fonction printf. Il suffit de lui ajouter, en premier argument, un pointeur vers une structure FILE retourné par la fonction fopen (et différent de NULL).

Pour plus d'informations sur les fonctions printf et fprintf, vous êtes invités à consulter la page suivante :
`http://manpagesfr.free.fr/man/man3/printf.3.html`

Écriture dans un fichier

Exemple de programme. Création d'un fichier HTML personnalisé.

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char chaine[255];
    int i,nb;
    FILE *fp;
    fp = fopen("miage.html","w");
    if(fp == NULL)
    {
        printf("Erreur lors de la creation du fichier.");
        exit(EXIT_FAILURE);
    }
    printf("Quel est votre nom ?");
    scanf("%s",chaine);
    printf("Combien d'items souhaitez vous ?");
    scanf("%d",&nb);
    fprintf(fp,"<html>\n\t<head>\n\t<title>Page d'exemple</title>\n\t</head>");
    fprintf(fp,"\n\t<body>\n\t\t<h1>Bonjour %s !</h1>\n\t\t<ul>\n",chaine);
    for(i=0;i<nb;i++)
    {
        fprintf(fp,"\t\t\t<li>item %d</li>\n",i);
    }
    fprintf(fp,"\t\t</ul>\n\t<body>\n</html>");
    fclose(fp);
    return EXIT_SUCCESS;
}
```

Exemple de programme

Création d'un fichier HTML personnalisé. Contenu du fichier généré.

```
<html>
    <head>
    <title>Page d'exemple</title>
    </head>
    <body>
        <h1>Bonjour delbot !</h1>
        <ul>
            <li>item 0</li>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
            <li>item 4</li>
            <li>item 5</li>
        </ul>
    <body>
</html>
```

Exemple de programme

Création d'un fichier HTML personnalisé. Ouverture avec Firefox.

Il suffit de double cliquer sur le fichier généré pour déclancher son ouverture par un navigateur web (du fait de l'extension .html)

Il s'agit d'un fichier texte, portant l'extension .html, dont le contenu est interprété par le navigateur web. A vous de vous amuser ;-)

Écriture dans un fichier

Autres fonctions.

En plus de la fonction fprintf, il existe d'autres fonction plus ou moins spécialisées. Par exemple :

1. La fonction fputc écrit un caractère c dans un fichier.

```
1    int fputc(int c, FILE *stream);
```

2. La fonction fputs écrit une chaine de caractères dans un fichier.

```
1    int fputs(const char *s, FILE *stream);
```

Pour plus d'informations sur ces fonction, rendez-vous sur la page manuel située à l'adresse suivante :
`https://linux.die.net/man/3/fputc`

Lire des données depuis un fichier

La fonction scanf

La fonction fscanf

La fonction fscanf, définie dans stdio.h se comporte de la même manière que la fonction scanf. Il suffit de lui ajouter, en premier argument, un pointeur vers une structure FILE retourné par la fonction fopen (et différent de NULL).

Pour plus d'informations sur les fonctions scanf et fscanf, vous êtes invités à consulter la page suivante :
`http://manpagesfr.free.fr/man/man3/scanf.3.html`

Lire des données depuis un fichier

Exemple de programme. Lecture d'un entier, d'un float et d'une chaine de caractères.

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

int main(void)
{
    char chaine[255];
    int nb;
    float moy;
    FILE *fp;
    fp = fopen("exemple.txt","r");
    if(fp == NULL)
    {
        printf("Erreur lors de l'ouverture du fichier.");
        exit(EXIT_FAILURE);
    }
    fscanf(fp,"%d",&nb);
    fscanf(fp,"%f",&moy);
    fscanf(fp,"%s",chaine);
    fclose(fp);
    printf("Les valeurs lues sont : %d puis %f puis %s\n", nb, moy, chaine);
    return EXIT_SUCCESS;
}
```

Lire des données depuis un fichier

La fonction scanf

Contenu du fichier exemple.txt :

```
42
3.14
coucou
```

Message affiché :

```
Les valeurs lues sont : 42
puis 3.140000 puis coucou
```

Lire des données depuis un fichier

Autres fonctions.

En plus de la fonction fscanf, il existe d'autres fonction plus ou moins spécialisées. Par exemple :

1. La fonction fgetc lit un caractère c depuis un fichier.

```
1    int fgetc(FILE *stream);
```

2. La fonction fgets lit une chaine de caractères de longeur au plus size depuis un fichier et la place dans un tableau.

```
1    char *fgets(char *s, int size, FILE *stream);
```

Pour plus d'informations sur ces fonction, rendez-vous sur la page manuel située à l'adresse suivante :
`https://linux.die.net/man/3/fgets`

Les flux standards

Il existe trois flux standards en langage C. Ces flux correspondent à des fichiers qu'il n'est pas nécessaire d'ouvrir, ni de fermer.

1. stdin (l'entrée standard) : le clavier
2. stdout (la sortie standard) : l'écran
3. stderr (l'erreur standard) : la sortie des messages d'erreur. Par défaut l'écran.

Les flux standards

Du sucre syntaxique

Il est tout à fait possible de manipuler les flux standards comme des fichiers. Exemple :

```
1   fprintf(stdout,"ca marche ! Trop fort !");
```

Plus précisément :

1. La fonction printf réalise un appel à la fonction fprintf sur stdout.
2. La fonction scanf réalise un appel à la fonction fscanf sur stdin.