Bases de la programmation

Séance 12

Lecture et écriture de fichiers

Sébastien Combéfis

mercredi 21 janvier 2015

Rappels du cours précédent

- Chaine de caractères
  - Déclaration, utilisation
  - Tableau de caractères
  - Opérations librairie standard
- Structure
  - Définition, initialisation
  - Paramètre et renvoi de structure
  - Structure de structures

Fichier

- Stockage permanent de données dans des fichiers

  Fichiers stockés sur le disque

- Deux types de fichiers
  - Suite de caractères stockée dans un fichier texte
  - Suite d'octets stockée dans un fichier binaire

- Deux opérations
  - Lecture de données depuis un fichier
  - Écriture de données dans un fichier

Ouvrir le fichier

- Ouverture d'un fichier avec fopen

```
FILE *file = fopen ("data.txt", "r");
if (file != NULL)
{
    // ...
}
```

- Descripteur de fichier utilisé pour les opérations

- Fichier ouvert en lecture ou écriture

  Peut être créé si non-existant

Fonction `fopen`

```
FILE* fopen (char *path, char *mode);
```

- Prend deux paramètres
  - Chemin du fichier sur le disque (absolu ou en relatif)
  - Mode d'ouverture
    - r : lecture seule
    - w : écriture (**fichier effacé** !)
    - a : écriture (ajout à la fin)
- Renvoie un pointeur vers un descripteur de fichier FILE$*$

  Pointeur NULL en cas d'erreur

Fermer le fichier

- Fermeture d'un fichier avec fclose

```
fclose (file);
```

  - Libère un descripteur de fichier ouvert dans l'OS
  - Force la sauvegarde sur disque (lors d'une écriture)

```
int fclose (FILE *file);
```

- Prend en paramètre le descripteur du fichier à fermer
- Renvoie 0 en cas de succès et EOF en cas d'échec

Ouvrir et fermer un fichier

```c
#include <stdio.h>

int main()
{
    FILE *file = fopen ("data.txt", "r");
    if (file != NULL)
    {
        printf ("Le fichier a été ouvert en lecture.\n");

        // lecture de données dans le fichier

        fclose (file);
    }

    return 0;
}
```

Lecture par caractère

- La fonction fgetc permet de lire un caractère dans un fichier

  *Conversion en **char** nécessaire*

```
char c = (char) fgetc (file);
```

- Renvoie EOF si fin du fichier ou erreur

```
int c;
while ((c = fgetc (file)) != EOF)
{
    printf ("%c", c);
}
```

Fonction `fgetc`

```
int fgetc (FILE *file);
```

- Prend en paramètre le descripteur du fichier où lire
- Renvoie le caractère suivant lu dans le fichier file

  EOF une fois la fin du fichier atteinte ou lors d'une erreur

Gestion des erreurs I

```
int feof (FILE *file);
```

- Permet de tester si la fin du fichier est atteinte
- Renvoie une valeur non-nulle si fin de fichier, et 0 sinon

```
int ferror (FILE *file);
```

- Permet de savoir si une erreur s'est produite
- Renvoie une valeur non-nulle si erreur, et 0 sinon

Gestion des erreurs II

```c
// Lecture du fichier caractère par caractère
int c;
while ((c = fgetc (file)) != EOF)
{
    printf ("%c", c);
}

// Vérification de l'état
if (feof (file))
{
    printf ("\n=> Lecture jusqu'au bout sans erreur.\n");
}
else if (ferror (file))
{
    printf ("\n=> Erreur produite pendant la lecture.\n");
}
```

Écriture par caractère

- La fonction fputc permet de lire un caractère dans un fichier

  *Conversion en **int** nécessaire*

```
char c = 'A';
fputc ((int) c);
```

- Renvoie EOF en cas d'erreur, le caractère écrit sinon

```
if ((fputc ((int) c)) == EOF)
{
    printf ("Une erreur d'écriture s'est produite.\n");
}
```

Fonction `fputc`

```
int fputc (int c, FILE *file);
```

- Prend deux paramètres
  - Le caractère à écrire dans le fichier
  - Le descripteur du fichier où écrire
- Écrit le caractère `c` dans le fichier `file`

  Renvoie EOF en cas d'erreur, sinon le caractère écrit

Copie d'un fichier

```c
#include <stdio.h>

int main()
{
    FILE *from = fopen ("data.txt", "r");
    FILE *to = fopen ("output.txt", "w");

    if (from != NULL && to != NULL)
    {
        int c;
        while ((c = fgetc (from)) != EOF)
        {
            fputc (c, to);
        }

        fclose (from);
        fclose (to);
    }

    return 0;
}
```

Lecture par ligne

- La fonction fgets permet de lire une ligne dans un fichier

  Nécessité d'allouer un buffer de taille suffisante

```
char *str = fgets (buffer, 80, file);
```

- La lecture s'arrête au premier \n et ajout d'un \0

- Renvoie NULL si fin du fichier ou erreur

```
char buffer[80];
while (fgets (buffer, 80, file) != NULL)
{
    printf ("%s", buffer);
}
```

Fonction `fgets`

```
char* fgets (char *str, int n, FILE *file);
```

- Prend trois paramètres
  - Un pointeur vers une zone mémoire **char***
  - Un nombre maximum de caractères à lire
  - Le descripteur du fichier où lire
- Lit maximum n − 1 caractères dans str depuis le fichier file
  - Renvoie str en cas de succès
  - S'il y a moins que n − 1 à lire, active EOF
  - Renvoie NULL en cas d'erreur ou s'il n'y a plus rien à lire

Lecture d'un fichier

```c
#include <stdio.h>

int main()
{
    FILE *file = fopen ("data.txt", "r");
    if (file != NULL)
    {
        char buffer[80];
        while (fgets (buffer, 80, file) != NULL)
        {
            printf ("%s", buffer);
        }

        fclose (file);
    }

    return 0;
}
```

Écriture par ligne

- La fonction fputs permet d'écrire une ligne dans un fichier

  Ne copie pas le caractère \0 et n'ajoute pas \n

```
char *str = "Hello";
fputs (str, file);
```

- Renvoie EOF en cas d'erreur, une valeur positive sinon

```
if (fputs ("Hello World", file) != EOF)
{
    printf ("L'écriture dans le fichier a réussi.\n");
}
```

Fonction `fputs`

```
int fputs (char *str, FILE *file);
```

- Prend deux paramètres
  - La chaine de caractères à écrire dans le fichier
  - Le descripteur du fichier où écrire
- Écrit la chaine de caractères str dans le fichier file

  Renvoie une valeur positive en cas de succès, et EOF sinon

Sortie formatée

- La fonction fprintf permet une sortie formatée dans un fichier

```
int i = 42;
fprintf (file, "%d", i);
```

- Renvoie le nombre de caractères écrits, ou un négatif en cas d'erreur

```
if (fprintf (file, "%d", i) >= 0)
{
    printf ("L'écriture dans le fichier a réussi.\n");
}
```

Fonction `fprintf`

```
int fprintf (FILE *file, char *format, ...);
```

- Prend deux paramètres + une liste
  - Le descripteur du fichier où écrire
  - Le format de la sortie (avec des balises)
  - Liste d'expressions pour les valeurs des balises
- Écrit format avec les balises remplacées dans le fichier file

  Renvoie le nombre de caractères écrits ou un négatif si erreur

Écrire une table de multiplication

```c
#include <stdio.h>

int main()
{
    FILE *file = fopen ("data.txt", "w");
    if (file != NULL)
    {
        int i;
        for (i = 1; i <= 10; i++)
        {
            fprintf (file, "%d x %d = %d\n", i, 3, i * 3);
        }

        fclose (file);
    }

    return 0;
}
```

Entrée formatée

- La fonction fscanf permet une sortie formatée dans un fichier

```
int i;
fscanf (file, "%d", &i);
```

- Renvoie le nombre de balises remplies, ou EOF en cas d'erreur

```
if (fscanf (file, "%d", &i) != EOF)
{
    printf ("La lecture dans le fichier a réussi.\n");
}
```

Fonction `fscanf`

```
int fscanf (FILE *file, char *format, ...);
```

- Prend deux paramètres + une liste
  - Le descripteur du fichier où écrire
  - Le format à lire en entrée (avec des balises)
  - Liste de pointeurs
- Lit format dans les pointeurs à partir du fichier file

  Renvoie le nombre de balises lues ou EOF si erreur

Calculer la somme d'entiers dans un fichier

```c
#include <stdio.h>

int main()
{
    FILE *file = fopen ("data.txt", "r");
    if (file != NULL)
    {
        int sum = 0;
        int i;
        while (fscanf (file, "%d", &i) != EOF)
        {
            sum += i;
        }

        printf ("La somme est %d.\n", sum);

        fclose (file);
    }

    return 0;
}
```

Écrire un fichier binaire

- Ouverture avec l'option b

```
FILE *file = fopen ("data.bin", "wb");
```

- La fonction fwrite écrit un tableau d'éléments

```
int N = 15;
int *tab;
// ...
fwrite (tab, sizeof (int), N, file);
```

- Renvoie le nombre d'éléments écrits et 0 en cas d'échec

```
if (fwrite (tab, sizeof (int), N, file) == N)
{
    printf ("L'écriture dans le fichier a réussi.\n");
}
```

Fonction `fwrite`

```
size_t fwrite (void *p, size_t s, size_t c, FILE *file);
```

- Prend quatre paramètres + une liste
  - Un pointeur vers un tableau
  - La taille des éléments du tableau
  - Le nombre d'éléments dans le tableau
  - Le descripteur du fichier où écrire
- Écrit le tableau `p` de taille `c` dans le fichier `file`

  Renvoie le nombre d'éléments écrits ou un négatif si erreur

Écrire des entiers dans un fichier binaire

```c
#include <stdio.h>

int main()
{
    FILE *file = fopen ("data.bin", "wb");
    if (file != NULL)
    {
        int i;
        for (i = 1; i <= 10; i++)
        {
            fwrite (&i, sizeof (int), 1, file);
        }

        fclose (file);
    }

    return 0;
}
```

Lire un fichier binaire

- Ouverture avec l'option b

```
FILE *file = fopen ("data.bin", "rb");
```

- La fonction fread lit un tableau d'éléments

```
int N = 15;
int *tab;
// ...
fread (tab, sizeof (int), N, file);
```

- Renvoie le nombre d'éléments lus et 0 en cas d'échec

```
if (fread (tab, sizeof (int), N, file) == N)
{
    printf ("La lecture depuis le fichier a réussi.\n");
}
```

Fonction `fread`

```
size_t fread (void *p, size_t s, size_t c, FILE *file);
```

- Prend quatre paramètres + une liste
  - Un pointeur vers un tableau
  - La taille des éléments du tableau
  - Le nombre d'éléments à lire
  - Le descripteur du fichier où lire
- Lit c éléments dans un tableau p depuis le fichier file

  Renvoie le nombre d'éléments lus ou un négatif si erreur

Lire des entiers depuis un fichier binaire

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

int main()
{
    FILE *file = fopen ("data.bin", "rb");
    if (file != NULL)
    {
        int N = 10;
        int *tab = (int*) malloc (N * sizeof (int));
        fread (tab, sizeof (int), N, file);
        free (tab);

        fclose (file);
    }

    return 0;
}
```

Format de fichier binaire