5 Fichiers et Structures

5.1 Bases sur les fichiers

Un fichier représente tout ce qui est enregistré sur votre disque dur ou presque, on va dire tout ce qui porte un nom. Il est possible de créer, de lire ou d'écrire dans des fichiers. Il faut noter cependant que certains fichiers par contre peuvent être protégés en lecture, en écriture ou les deux.

Afin de ne rien endommager dans notre système, créez à l'aide de NotePad ou de la commande Ms-Dos « edit » un fichier nommé test.txt, dans lequel vous taperez un texte de louanges pour ce superbe cours (à votre libre inspiration).

Voici un petit exemple de la lecture du fichier test.txt.

Exemple

```
#include <conio.h>
#include <io.h>
#include <fcntl.h>

int   main ()
{
      int   h_fic;
      char  ligne [80];
      int   nb_car_lus;
      int   i;

      /* effacement de l'écran */
      clrscr ();

      /* Ouverture du fichier */
      h_fic = open ("c:\\test.txt ", O_CREAT);

      /* Test si fichier ouvert */
      if (h_fic == -1)
      {
            printf ("Impossible d'ouvrir le fichier");
            getch ();
            return (1);
      }

      while (!eof (h_fic))
      {
            /* Lecture de 80 octets maximum */
            nb_car_lus = read (h_fic, ligne, 80);

            /* Ecriture de ce qui a ,t, lu */
            for (i=0; i<nb_car_lus; i++)
            {
                  printf ("%c",ligne [i]);
            }
            printf ("\n");
      }

      /* Fermeture du fichier */
      close (h_fic);
```

```
	/* Ecrire que c'est terminé */
	printf ("\n --- Fin ---");

	getch ();
	return (0);
}
```

Analysons l'ensemble

```
	/* Ouverture du fichier */
	h_fic = open ("c:\\test.txt", O_CREAT);
```

La fonction **open** permet d'ouvrir un fichier, le second paramètre est son mode d'ouverture, ici O_CREAT signifiant ouverture s'il existe ou création s'il n'existe pas. Par contre O_CREAT seul ne permet pas d'écrire dans le fichier offrant ainsi une sécurité supplémentaire. La fonction open renvoie –1 si l'on a pas réussi à ouvrir notre fichier et renvoie un handle (poignée) sur le fichier sinon.

What is an handle ?

Un lien sur quelque chose. On appelle ceci une poignée car comme avec un sac, vous le prenez par la poignée pour le porter.

Notez la présence de \\ au lieu de \ du fait que \ seul permet la composition de caractère spéciaux (ex \n).

```
	/* Test si fichier ouvert */
	if (h_fic == -1)
	{
		printf ("Impossible d'ouvrir le fichier");
		getch ();
		return (1);
	}
```

La fonction **eof** permet de savoir si l'on a atteint la fin du fichier. Regardez l'aide pour comprendre le while (!eof (…))

```
	while (!eof (h_fic))
	{
```

La fonction **read** permet de lire des octets (ici transformés en char). Elle prend pour paramètre le handle de fichier, la zone de mémoire dans laquelle on va recevoir les données et la taille maximale que l'on veut lire (attention au dépassement (cf. chapitre 2)).

```
		/* Lecture de 80 octets maximum */
		nb_car_lus = read (h_fic, ligne, 80);

		/* Ecriture de ce qui a été lu */
		for (i=0; i<nb_car_lus; i++)
		{
```

```
                printf ("%c",ligne [i]);
        }
        printf ("\n");
    }
```

A ne pas oublier, il faut fermer le fichier après l'avoir ouvert !

```
    /* Fermeture du fichier */
    close (h_fic);

    /* Ecrire que c'est terminé */
    printf ("\n --- Fin ---");

    getch ();
    return (0);
}
```

Très simple non !

5.2 Création d'un autre fichier

Reprenez l'exemple précédent et ajouter ceci après le open :

```
h_fic2= open ("c:\\ma_copie.bat", O_CREAT);
```

N'oubliez pas de déclarer h_fic2.

Ainsi nous allons créer une copie du fichier.

```
int         h_fic2;

/* Création d'une copie */
h_fic2 = open ("c:\\ma_copie.txt", O_CREAT);

/* Test si fichier bien créer */
if (h_fic2 == -1)
{
      printf ("Impossible de créer le nouveau fichier");
      close (h_fic2);
      getch ();
      return (1);
}

while (!eof (h_fic))
{
      /* Coupure */
      printf ("\n");

      /* Ecriture dans notre copie de fichier */
      write (h_fic2, ligne, nb_car_lus);
}

/* Fermeture du fichier */
close (h_fic);

/* Fermeture de ma copie de fichier */
close (h_fic2);
```

En **gras** sont signalés tous les ajouts effectués.

La fonction **write** permet d'écrire des octets sur le fichier. Les paramètres sont le handle de fichiers, les octets à écrire et la taille à écrire.

Essai, laissez 80 au lieu de nb_car_lus. Regardez, avec un peu de chance, vous allez trouver des morceaux de texte non désirés, eh oui, personne ne vous a dit que votre chaîne de départ était vide !

5.3 Ecriture dans un fichier : complément.

Insérer cette ligne après notre write :

```
/* Ecriture dans notre copie de fichier */
write (h_fic2, ligne, nb_car_lus);
```

oui la même ligne.

Regardez.

La fonction write a ajouté à la fin du fichier.

Attention, l'utilisation de O_CREAT permet de créer le fichier en mode lecture seul, pour l'ouvrir en plus avec des droits d'écriture il faut lui associer O_RDWR (lecture/écriture) ou O_WRONLY (écriture seule) de la façon suivante :

open ("c:\\test.txt", O_CREAT | O_RDWR).

5.4 Positionnement et taille d'un fichier

- La taille d'un fichier est donnée par la fonction **int filelength (handle)**
- La position en octets dans un fichier est donnée par la fonction **int tell (handle)**

Exemple

```
#include <conio.h>
#include <io.h>
#include <fcntl.h>

int   main ()
{
      int         position;
      int         h_fic;
      char  ligne [80];
      int         nb_car_lus;
      int         i;
      int         taille;

      /* effacement de l'écran */
      clrscr ();

      /* Ouverture du fichier */
      h_fic = open ("c:\\test.txt", O_CREAT);

      /* Test si fichier ouvert */
      if (h_fic == -1)
      {
            printf ("Impossible d'ouvrir le fichier");
```

```
            getch ();
            return (1);
      }

      /* Taille du fichier */
      taille =    filelength (h_fic);
      printf ("Taille du fichier : %d\n",taille);

      while (!eof (h_fic))
      {
            /* Position dans le fichier */
            position = tell (h_fic);
            printf ("Position dans le fichier : %d\n",position);

            /* Lecture de 80 octets maximum */
            nb_car_lus = read (h_fic, ligne, 80);

            /* Ecriture de ce qui a ,t, lu */
            for (i=0; i<nb_car_lus; i++)
            {
                  printf ("%c",ligne [i]);
            }

            printf ("\n");
      }

      /* Position dans le fichier */
      position = tell (h_fic);
      printf ("Position dans le fichier : %d\n",position);

      /* Fermeture du fichier */
      close (h_fic);

      printf ("\n --- Fin ---");

      getch ();
      return (0);
}
```

5.5 Structures

Nous avons vu, dans le chapitre précédent, l'utilisation des structures. Nous allons maintenant voir la définition et l'utilisation d'une structure.

5.5.1 Déclaration

2 choix se proposent à nous.

Solution 1 :

struct
{
    /* Définition de la structure */
} **nom de la variable** qui aura comme forme cette structure;

Solution 2 :

typedef struct
{
    /* Définition de la structure */
} **nom de la structure;**

<nom de la structure> nom de la variable qui aura comme structure cette structure.

Je préfère de loin la seconde solution.

Exemple

```
typedef struct
{
        char    nom [40];
        char    prenom [20];
        int     age;
} elt_fiche_nom;

elt_fiche_nom une_fiche ;
elt_fiche_nom une_seconde_fiche ;
```

elt_fiche_nom est le nom de la stucture, la macro typdef signifie définition d'un type, de ce fait là elt_fiche_nom se comporte exactement comme un autre type de données (int par exemple).

Par la suite, je n'utiliserai que la seconde méthode.

5.5.2 Utilisation

```
struct
{
        char    nom [40];
        char    prenom [20];
        int     age;
} fiche_nom;

fiche_nom       une_fiche;
fiche_nom       une_seconde_fiche;

une_fiche.age = 20;
```

5.5.3 Taille d'une structure

La taille d'une structure ou d'un type se détermine par la fonction **sizeof**.

Exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>

int main()
{
      typedef struct
      {
            char  nom [40];
            char  prenom [20];
```

```
            int   age;
      } fiche;

      fiche ma_fiche;

      printf ("Taille de la structure : %d\n",sizeof (fiche));
      sprintf (ma_fiche.nom, "%s","BERTHOMIER");
      sprintf (ma_fiche.prenom, "%s","Eric");
      ma_fiche.age = 30;
      printf ("%s %s est âgé de %d",ma_fiche.prenom, ma_fiche.nom,
             ma_fiche.age);

      getch ();
      return (0) ;
}
```

5.6 Fichier & structure

Ce programme ne fonctionne pas sous win98 avec le compilateur Borland C 2. (Compilateur 16 bits)
Il fonctionne par contre avec un compilateur 32 bits (Test avec VC++ 6.0).
L'explication existe mais est hors de portée de ce cours ou alors un bug de ma part. Peut être, peut être. Nul n'est parfait.

Je laisse tout de même celui-ci à titre d'exemple.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <fcntl.h>

int main(int argc, char* argv[])
{
      typedef struct
      {
            char  nom [40];
            char  prenom [20];
            unsigned char     age;
      } fiche;

      fiche ma_fiche;
      int   h_fic ;
      int         i;

      /* Création du du fichier -> le vider s'il existait */
      h_fic = open ("c:\\fiche.dat", O_CREAT | O_TRUNC | O_WRONLY);

      /* Test si fichier ouvert */
      if (h_fic == -1)
      {
            printf ("Impossible d'ouvrir le fichier");
            getch ();
            return (1);
      }

      printf ("Taille de la structure : %d\n\n",sizeof (fiche));

      sprintf (ma_fiche.nom, "%s","BERTHOMIER");
      sprintf (ma_fiche.prenom, "%s","Eric");
      ma_fiche.age = 30;
```

```
	printf ("Fiche 1 : %s %s est âgé de %d\n",ma_fiche.prenom,
	ma_fiche.nom, ma_fiche.age);
	write (h_fic, &ma_fiche, sizeof (fiche));

	sprintf (ma_fiche.nom, "%s","P'TIT");
	sprintf (ma_fiche.prenom, "%s","Luc");
	ma_fiche.age = 48;
	printf ("Fiche 2 : %s %s est âgé de %d\n",ma_fiche.prenom,
	ma_fiche.nom, ma_fiche.age);
	write (h_fic, &ma_fiche, sizeof (fiche));

	sprintf (ma_fiche.nom, "%s","Tolkien");
	sprintf (ma_fiche.prenom, "%s","JRR");
	ma_fiche.age = 89;
	printf ("Fiche 3 : %s %s est âgé de %d\n",ma_fiche.prenom,
	ma_fiche.nom, ma_fiche.age);
	write (h_fic, &ma_fiche, sizeof (fiche));

	/* Fermeture du fichier */
	close (h_fic);

	/* ------------------------------- */
	/*  Lecture des éléments du fichier  */
	/* ------------------------------- */
	/* Ouverture du fichier */
	h_fic = open ("c:\\fiche.dat", O_RDONLY);

	/* Test si fichier bien ouvert */
	if (h_fic == -1)
	{
		printf ("Impossible d'ouvrir le fichier");
		getch ();
		return (1);
	}

	for (i=0; i<3; i++)
	{
		/* Lecture de la fiche */
		read (h_fic, &ma_fiche, sizeof (fiche));

		/* Affichage des donn,es lues */
		printf ("Fiche n° %d : %s %s est âgé de %d\n", i,
		ma_fiche.prenom, ma_fiche.nom, ma_fiche.age);
	}

	/* Fermeture du fichier */
	close (h_fic);

	getch ();
	return 0;
}
```

5.7 Fin et commencement

Vous disposez maintenant d'à peu près toutes les connaissances nécessaires pour l'élaboration de petits programmes. D'autres cours suivront, en fonction de la demande.

Pour tout commentaire, suggestion de cours, écrivez moi :

eric.berthomier@free.fr ou lesouriciergris@free.fr