

# La bibliothèque standard du C

Avec les structures et l'allocation de la mémoire, nous nous rendons compte qu'un langage doit s'appuyer sur des librairies système pour construire des applications complètes. Le langage C++ possède sa propre librairie, mais nombre de programmeurs utilisent toujours les fonctions standards du langage C.

## 1. Les fonctions communes du langage C <stdlib.h>

La librairie standard `stdlib.h` contient des fonctions d'ordre général. Certaines fonctions peuvent être d'ailleurs déclarées dans d'autres en-têtes.

Voici une liste résumant quelques fonctions intéressantes pour le développement courant. Il est judicieux de consulter l'ouvrage de Kernighan et Ritchie ou bien une documentation fournie avec le compilateur pour connaître à la fois la liste complète des fonctions et en même temps leur signature.

Le Kernighan et Ritchie est l'ouvrage de référence écrit par les créateurs du langage C. Il est toujours édité et mis à jour à partir des évolutions du langage C.

Fonctions	Utilité
<code>atoi</code> , <code>atof</code> , <code>strtod</code> ...	Fonctions de conversion entre un type chaîne et un type numérique.
<code>getenv</code> , <code>setenv</code>	Accès aux variables d'environnement système.
<code>malloc</code> , <code>calloc</code>	Allocation de mémoire.
<code>rand</code> , <code>abs</code>	Fonctions mathématiques.

La bibliothèque standard `stdlib` contient aussi des macros instructions basées sur la syntaxe `#define` :

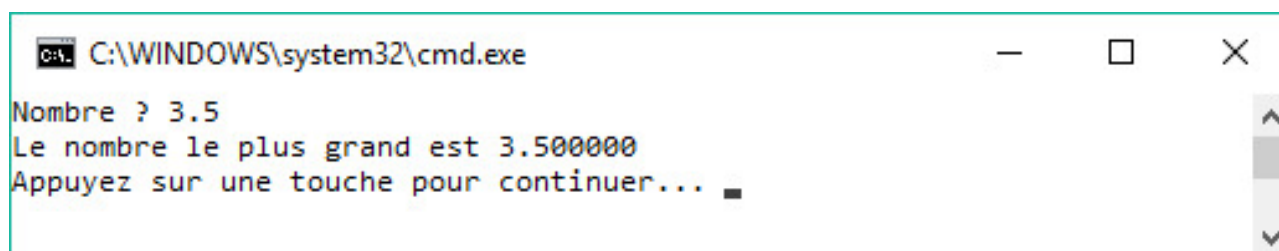
Macro	Utilité
<code>__min</code> , <code>__max</code>	Donne les valeurs min et max de deux arguments.
<code>NULL</code>	Littéralement <code>(void*)0</code> .

Un petit exemple montre comment utiliser la bibliothèque :

```
char* lecture = new char[500];
printf("Nombre ? ");
scanf("%s",lecture);

double nombre = atof(lecture);
double pi = 3.14159265358;

printf("Le nombre le plus grand est %2f\n",__max(nombre,pi));
```



```
C:\WINDOWS\system32\cmd.exe
Nombre ? 3.5
Le nombre le plus grand est 3.500000
Appuyez sur une touche pour continuer...
```

## 2. Chaînes <string.h>

Le langage C++ gère toujours ses littérales de chaîne au format `char*` défini par le langage C. Aussi est-il important de connaître les principales fonctions de la bibliothèque `string.h`.

Fonctions	Utilité
<code>memcpy</code> , <code>memcmp</code> , <code>memset</code>	Vision mémoire des chaînes de caractères : copie, comparaison, initialisation.
<code>strcpy</code> , <code>strcmp</code> , <code>strcat</code> , <code>strlen</code>	Copie, comparaison, concaténation, longueur.
<code>strchr</code> , <code>strstr</code>	Recherche de caractère, de sous-chaîne.
<code>strlwr</code> , <code>strupr</code>	Conversion minuscules/majuscules.

Prenons là encore un exemple pour illustrer l'emploi de cette librairie :

```

/*
 * Recherche
 */
printf("** Recherche\n");

// chaîne à analyser
char* chaine1 = "Amateurs de C++ et de programmation";

// motif à rechercher
char* motif = "C++";

// recherche : strstr fournit un pointeur sur la sous-chaîne
char* souschaine = strstr(chaine1,motif);

// résultat
if(souschaine != NULL)
    printf("Trouve le motif [%s] dans [%s] : \n[%s] a la position [%d]\n\n",
        motif,chaine1,souschaine,souschaine-chaine1);

/*

```

```

* Comparaison, concaténation
*/

printf("** Comparaisons, concatenations\n");

// réserver de la mémoire
char* chaine2 = new char[strlen(chaine1) + 50];

// construire la chaîne
strcpy(chaine2, chaine1);
strcat(chaine2, ", un nouvel ouvrage est disponible chez ENI");

// résultat
printf("chaine2 = %s\n", chaine2);

// initialiser une chaîne
char chaine3[50];
memset(chaine3, 'O', 3); // chaque octet est initialisé par le caractère O
chaine3[3] = 0; // 0 terminal (identique à '\0' à la 4ème position)

// résultat
if(strcmp(chaine3, "OOO"))
    printf("chaine3 = %s\n", chaine3);

// ne pas se tromper avec les longueurs
char* chaine4 = "ABC";
char* chaine5 = new char[ strlen(chaine4) + 1]; // +1 pour le 0 terminal
strcpy(chaine5, chaine4);

// passer en minuscules
char* chaine6 = strlwr(chaine5); // attention chaine5 est modifiée
printf("chaine5=%s chaine6=%s\n\n", chaine5, chaine6);

/*
* E/S
*/

printf("** E/S\n");

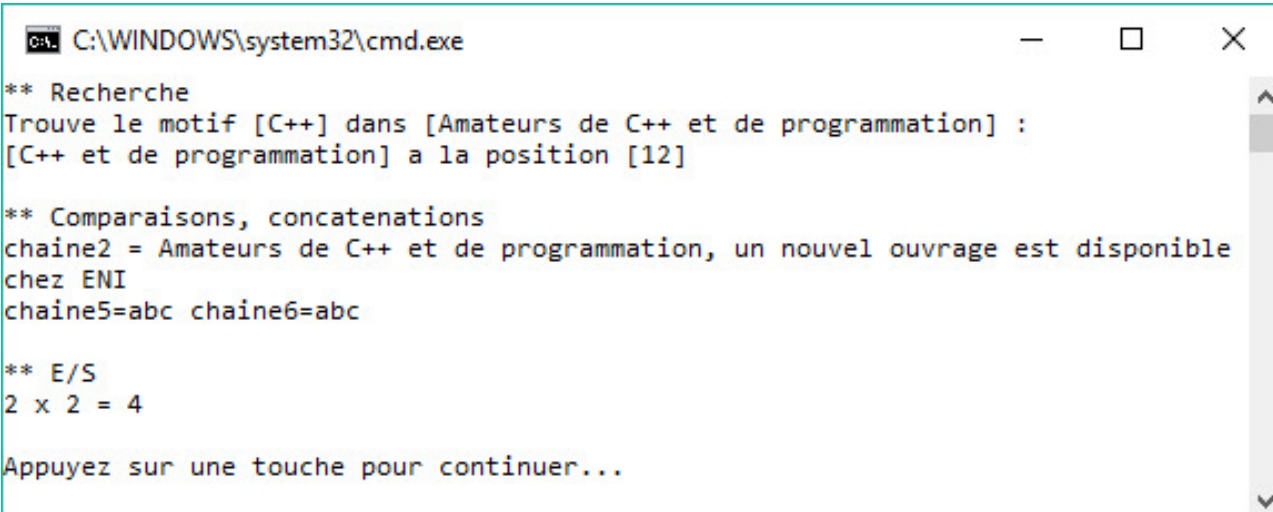
```

```
char message[500];

// écrire un message formaté dans une chaîne
sprintf(message,"2 x 2 = %d\n", 2*2);

printf("%s\n",message);
```

L'exécution de ce programme nous donne le résultat suivant :



```
C:\WINDOWS\system32\cmd.exe
** Recherche
Trouve le motif [C++] dans [Amateurs de C++ et de programmation] :
[C++ et de programmation] a la position [12]

** Comparaisons, concatenations
chaine2 = Amateurs de C++ et de programmation, un nouvel ouvrage est disponible
chez ENI
chaine5=abc chaine6=abc

** E/S
2 x 2 = 4

Appuyez sur une touche pour continuer...
```

### 3. Fichiers <stdio.h>

Les fonctions déclarées dans `stdio.h` sont consacrées aux opérations d'entrée-sortie, au sens large. On trouve deux types de prise en charge des fichiers : par `handle` système et par structure `FILE*`. Le premier type est plus ancien et fortement lié à Unix. Toutefois, la technique du `handle` a été généralisée dans d'autres parties du système d'exploitation : les sockets et le réseau, le graphisme...

Un handle est généralement assimilable à un entier. Il s'agit d'un identificateur géré par le système pour désigner différents types de ressources.

L'accès par la structure `FILE*` est quant à lui de plus haut niveau, donc plus spécialisé. Certains "fichiers" sont déjà déclarés, tels que `stdin`, `stdout`, `stderr` et `stdprn`.

Enfin, `stdio.h` propose des fonctions de formatage vers des chaînes de caractères, telles que `sprintf` (également présente dans `string.h`).

Fonctions	Utilité
<code>printf</code> , <code>scanf</code>	Impression sur <code>stdout</code> , lecture depuis <code>stdin</code> .
<code>fprintf</code> , <code>fscanf</code>	Impression et lecture depuis un <code>FILE*</code> .
<code>sprintf</code>	Impression sur une chaîne.
<code>open</code> , <code>close</code> , <code>read</code> , <code>write</code>	Ouverture, fermeture, lecture, écriture sur un <code>handle</code> (entier).
<code>fopen</code> , <code>fclose</code> , <code>fread</code> , <code>fwrite</code>	Ouverture, fermeture, lecture et écriture sur un <code>FILE*</code> .
<code>fflush</code>	Transfert du tampon ( <i>buffer</i> ) vers son périphérique ( <code>FILE*</code> ).
<code>putc</code> , <code>getc</code>	Écriture, lecture d'un seul caractère ( <code>handle</code> ).
<code>fputc</code> , <code>fgetc</code>	Écriture, lecture d'un seul caractère depuis un <code>FILE*</code> .
<code>feof</code>	Test de la fin d'un flux ( <code>FILE*</code> ).
<code>ftell</code> , <code>fseek</code>	Lecture et déplacement du pointeur de lecture /écriture dans un fichier <code>FILE*</code> .
<code>remove</code> , <code>rename</code> , <code>unlink</code> , <code>tempnam</code>	Gestion du nom d'un fichier.

L'exemple est ici un peu plus développé que précédemment. Il s'agit d'afficher le contenu d'un fichier en hexadécimal et en ASCII (on dit "dumper"). Le programme contient des méthodes utilitaires de déplacement dans le fichier ainsi qu'une fonction de lecture complète du fichier. Le lecteur pourra à titre d'exercice remplacer cette fonction par un menu proposant à l'utilisateur de lire ligne à ligne le fichier, de revenir au début...

```
// définit le point d'entrée pour l'application console.
```

```
#include "stdafx.h"
```

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

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

```
#include <string.h>
```

```
const int T_BLOC = 16;
```

```
// revient au début du fichier
```

```
void debut(FILE* f)
```

```
{
```

```
    fseek(f,0,SEEK_SET);
```

```
}
```

```
// va à la fin du fichier
```

```
void fin(FILE* f)
```

```
{
```

```
    // longueur du fichier
```

```
    fseek(f,0,SEEK_END);
```

```
    long taille = ftell(f);
```

```
    fseek(f,taille - (taille % T_BLOC), SEEK_SET);
```

```
}
```

```
// remonte dans le fichier
```

```
void haut(FILE* f)
```

```
{
```

```
    fseek(f, -T_BLOC, SEEK_CUR);
```

```
}
```

```
// descend dans le fichier
```

```
void bas(FILE* f)
```

```
{
```

```
    fseek(f, T_BLOC, SEEK_CUR);
```



```

}

// détermine la taille du fichier
long taille(FILE* f)
{
    // position actuelle
    long pos = ftell(f);

    // va à la fin et détermine la longueur à partir de la position
    fseek(f, 0, SEEK_END);
    long taille = ftell(f);

    // revient au point de début
    fseek(f, pos, SEEK_SET);

    return taille;
}

void afficher_bloc(FILE* f)
{
    int offset = ftell(f);
    char chaine_hexa[T_BLOC*3 + 2 + 1];
    char chaine_ascii[T_BLOC*2 + 2 + 1];

    strcpy(chaine_hexa, "");
    strcpy(chaine_ascii, "");

    int data;
    for(int i=0; i<16; i++)
    {
        int car = fgetc(f);
        if(car == -1)
            break;

        char concat[50];

        sprintf(concat, "%2x ", car);
        strcat(chaine_hexa, concat);

        sprintf(concat, "%c", car>=32?car:' ');
    }

```

```

    strcat(chaine_ascii,concat);

    if(i == T_BLOC/2)
    {
        strcat(chaine_hexa, " ");
        strcat(chaine_ascii, " ");
    }
}

// fprintf(stdout) = printf !
fprintf(stdout,"%4d | %s | %s\n",offset,chaine_hexa,chaine_ascii);
}

int main(int argc, char* argv[])
{

    char* nom_fichier="c:file://temp//briceip.txt";
    FILE*f = NULL;

    try {
        f = fopen(nom_fichier,"rb");

        /*
         * lecture continue du fichier
         */
        long p = 0;
        long t = taille(f);

        debut(f);
        while(p<t)
        {
            afficher_bloc(f);
            p+=T_BLOC;
        }

        fclose(f);
    }
    catch(...)
    {

```

```

    printf("erreur\n");
}

return 0;
}

```

Voici le résultat pour un fichier d'exemple :

```

C:\WINDOWS\system32\cmd.exe
0 | 2f 2a 20 d a 9 64 65 6d 6f 6e 73 74 72 61 74 | /* dem onstrat
16 | 69 6f 6e 20 66 6f 6e 63 74 69 6f 6e 73 20 d a | ion fonct ions
32 | 2a 2f d a 2f 2f 20 66 6f 6e 63 74 69 6f 6e 20 | */ // fo nction
48 | 73 6f 6d 6d 65 d a 66 6f 6e 63 20 73 6f 6d 6d | somme fo nc somm
64 | 65 28 61 2c 62 29 20 3a d a 64 65 62 75 74 d | e(a,b) : debut
80 | a 9 65 76 61 6c 20 22 61 20 2b 20 62 20 3d 22 | eval "a + b ="
96 | 20 5f 20 28 61 20 2b 20 62 29 3b d a 9 72 65 | _ (a + b ); re
112 | 74 6f 75 72 6e 65 72 20 61 20 2b 20 62 3b d a | tourner a + b;
128 | 66 69 6e 20 3b d a d a 2f 2f 20 66 6f 6e 63 | fin ; // fonc
144 | 74 69 6f 6e 20 73 65 75 69 6c d a 66 6f 6e 63 | tion seuil fonc
160 | 20 73 65 75 69 6c 28 61 29 20 3a d a 64 65 62 | seuil(a) : deb
176 | 75 74 d a 9 73 69 20 61 20 3c 20 31 30 20 61 | ut si a < 10 a
192 | 6c 6f 72 73 d a 9 9 72 65 74 6f 75 72 6e 65 | lors r etourne
208 | 72 20 61 20 3b d a 9 73 69 6e 6f 6e d a 9 | r a ; s inon
224 | 9 72 65 74 6f 75 72 6e 65 72 20 31 30 20 3b d | retourne r 10 ;
240 | a 9 66 69 6e 73 69 d a 66 69 6e 20 3b d a | fin si fin ;
256 | d a 2f 2f 20 70 72 6f 67 72 61 6d 6d 65 d a | // prog ramme
272 | 65 76 61 6c 20 73 6f 6d 6d 65 28 31 2c 32 29 3b | eval somm e(1,2);
288 | d a 73 6f 6d 6d 65 28 33 2c 34 29 3b d a 76 | somme(3 ,4); v
304 | 61 72 20 78 3b d a 78 20 3d 20 73 6f 6d 6d 65 | ar x; x = somme
320 | 28 31 30 30 2c 34 30 30 29 3b d a 65 76 61 6c | (100,400) ; eval
336 | 20 22 78 20 3d 20 22 20 5f 20 78 3b d a 65 76 | "x = " _ x; ev
352 | 61 6c 20 22 73 28 35 29 20 3d 20 22 20 5f 20 73 | al "s(5) = " _ s
368 | 65 75 69 6c 28 35 29 3b d a 65 76 61 6c 20 22 | euil(5); eval "
384 | 73 28 31 35 29 20 3d 20 22 20 5f 20 73 65 75 69 | s(15) = " _ seuil

```