

0.1 Les bibliothèques

Un certain nombre de fonctions sont en langage C regroupées dans des bibliothèques. Cela évite au programmeur de devoir pour chaque programme réécrire des fonctions déjà connus et communément utilisées. Ainsi il peut s'attaquer à son problème tout de suite. Ou bien encore, le développeur sait qu'il va réutiliser ses fonctions pour d'autres programmes, en ce cas il aura tout intérêt à créer des librairies.

Les bibliothèques peuvent être liées au programme de deux manières : statiquement ou bien dynamiquement. Dans le premiers cas, elle sera totalement incluse dans le programme, dans le second cas elle liée à l'exécutable que pendant l'exécution de celui-ci. Ainsi l'usage d'une librairie statique rend le programme plus lourd, oblige de recompiler le programme en cas de modification de celle-ci mais à l'avantage non négligeable de ne pas générer de problèmes de dépendance. L'utilisation d'une librairie dynamique à l'inverse à l'avantage de pouvoir être modifiée sans avoir pour conséquence de recompiler le programme, de rendre ce dernier plus léger, mais peut générer des problèmes de dépendance (ceux que nous rencontrons :-)) si elle n'est pas présente sur le système où est exécuté le programme. La distinction entre librairie statique et dynamique se fera lors de la compilation du programme, autrement dit de quelle façon elle sera liée à celui-ci.

0.1.1 1. Structure de la bibliothèque :

La librairie est composée de deux fichiers : le premier sera le fichier d'en tête, il est inscrit dans celui-ci la déclaration des fonctions, c'est à dire le type de paramètre que les fonctions utilisent, le type qu'elles renvoient, et rien de plus. Ce fichier aura pour extension **.h**. Dans le second fichier sera inscrit le corps des fonctions, en C cela va de soi. Ce dernier ressemblera en tout point à un programme en C à l'exception mais de taille de l'absence de la fonction principale **main**. Comme il est réalisé en C, il est tout naturel qu'il porte l'extension **.c**. Remarque : l'utilisation d'un fichier d'en tête **.h** pour déclarer ses fonctions dans un programme peut-être utilisé et est même souhaitable.

0.1.2 2. Les bibliothèques standards :

Pour réaliser cette partie, je me suis fortement inspiré du manuel **Introduction au langage C** réalisé par **Bernard CASSAGNE**. Je vous propose ici de vous donner les quelques bibliothèques les plus utilisées en C, avec un descriptif rapide de leur contenu.

0.1.3 2.1 Les entrées - sorties :

A inclure avec **#include <stdio.h>**. Cette bibliothèque, comme son nom l'indique, contient une multitude de fonction permettant des entrées dans le programme (exemple : entrée d'un nombre dans un terminal, entrée dans un fichier au cours de l'exécution du programme etc...) ou bien des sorties (exemple : affichage des résultats à l'écran ...).

Quelques fonctions usuelles :

- impression sur la sortie standard : **printf**. Cette impression est formatée, c'est à dire qu'il faut indiquer à **printf** le type de la variable qu'elle affiche. Les types les plus courants sont :
 - **%d** pour un entier
 - **%f** pour un double

- %e pour un double en écriture scientifique
 - %c pour un caractère
 - %s pour une chaîne (tableau) de caractères
 - %4f pour n'afficher que 4 chiffres après la virgule du double
 - \n pour le retour chariot
- usage : **printf ("Cette phrase d'affichera à l'écran"); printf ("La valeur de la variable est %5f", variable);**
- lecture sur l'entrée standard : **scanf (format, val1, ... , valn)** format indique le format du nombre lu (décimale, hexadécimale,...) :
 - %d pour un décimal
 - %lf pour un double
 - %x pour un hexadécimale
 - %c pour un caractère
 - %s pour une chaîne de caractères
 val1 etc sont les noms des variables qui seront affectée de la valeur. usage : **scanf ("%d %c",&i,&j);** Dans cette exemple, i sera affectée de la première valeur lue qui est un décimal, et j de la seconde qui est un caractère. Pour mieux comprendre ce symbole & devant le nom de la variable, il faut se reporter à la section pointeurs¹. Cela permet simplement de dire de ranger la valeur lue à l'adresse mémoire de la variable.
 - Ouverture d'un fichier durant l'exécution d'un programme : **fopen (fich, mode)**; fopen ouvre le fichier fich dans le mode que nous lui auront indiqué. Voici quelques modes :
 - r : ouvre le fichier en lecture seule, r suppose que le fichier existe autrement c'est une erreur.
 - w : ouvre le fichier en écriture, si le fichier n'existe pas il sera créé, autrement le fichier existant sera écrasé.
 - a : ouvre le fichier en écriture mais cette fois le fichier ne sera pas écrasé : toute écriture sera effectuée en fin de fichier.
 fopen renvoie comme valeur un pointeur de type FILE pointant sur le fichier, si l'ouverture est impossible elle renvoie la valeur NULL.
- Exemple d'utilisation :

```
#include <stdio.h>
FILE *fp;
if ((fp = fopen("donnees", "r")) == NULL)
{
    fprintf(stderr, "Impossible d'ouvrir le fichier données
    en lecture\n");
    exit(1);
}
```

- fermeture d'un fichier : **fclose (fp)**; fclose ferme le fichier qui est pointer par fp. Elle renvoie la valeur 0 si la fermeture est un succès, la valeur EOF en cas d'échec.

¹<http://www.trustonme.net/didactels/155.html>

0.1.4 2.2 Les fonctions mathématiques :

A inclure avec **#include <math.h>**. Leurs usages étant assez connue et du type **fonction (ma_variable)**, je ne développerai pas trop.

- Fonctions trigonométriques et hyperboliques.
 - **acos** : arc cosinus
 - **asin** : arc sinus
 - **atan** ou **atan2** : arc tangente
 - **cos** : cosinus
 - **sin** : sinus
 - **tan** : tangente
 - **cosh** : cosinus hyperbolique
 - **sinh** : sinus hyperbolique
 - **tanh** : tangente hyperbolique
- Fonctions exponentielles et logarithmiques
 - **exp** : exponentielle
 - **log** : logarithme népérien (de base e)
 - **log10** : logarithme décimale
 - **frexp** : étant donné x, calcul n et p tel que $x = n \cdot 2^p$
 - **modf** : calcule partie entière et décimale d'un nombre
- Autres fonctions
 - **ceil** : entier le plus proche par valeur supérieure
 - **floor** : entier le plus proche par valeur inférieure
 - **fabs** : valeur absolue
 - **fmod** : reste de la division entière
 - **pow** : puissance
 - **sqrt** : racine carrée

0.1.5 2.3 Utilitaires divers :

A inclure avec **#include <stdlib.h>**.

- Conversion de nombre
 - **atof** : conversion de chaîne vers double
 - **atoi** : conversion de chaîne vers int
 - **atol** : conversion de chaîne vers long int
 - **strtod** : conversion de chaîne vers double
 - **strtoul** : conversion de chaîne vers long int
 - **strtoul** : conversion de chaîne vers unsigned long int
- Génération de nombre aléatoire
 - **rand** : pour les nombres
 - **srand** : pour les chaînes de caractères

0.1.6 2.4 Manipulation de chaînes :

A inclure avec **#include <string.h>**.

- **memcpy**, **memmove**, **strcpy**, **strncpy** : pour copier
- **strcat**, **strncat** : pour concaténer
- **memcmp**, **strcmp**, **strcoll**, **strncmp** : pour comparer

- **strxfrm** : pour transformer
- **memchr, strchr, strcspn, strpbrk, strchr, strspn, strstr, strtok** : pour rechercher
- **memset** : pour initialiser
- **strlen** : pour calculer une longueur

0.1.7 2.5 Autres librairies :

Ne m'étant jamais servi de ces autres bibliothèques, je vous en donne seulement une description rapide...

- **<ctype.h>** : manipulation de caractères Ces fonctions testent une caractéristique du caractère passé en paramètre : **isalnum, isalpha, isctrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, tolower** : permet la conversion en miniscule **toupper** : permet la conversion en majuscule
- **<locale.h>** : permet de gérer les conventions nationales concernant le point dans les nombres, le symbole monétaire. Les deux fonctions associées sont : **setlocale** et **localeconv**.
- **<setjmp.h>** : branchement non locaux Elle permet d'effectuer des branchement avec goto à l'extérieur d'une procédure. Pour cela on utilise **setjmp** et **longjmp**.
- **<time.h>** : manipulation de la date et de l'heure **clock, difftime, mktime, time, asctime, ctime, gmtime, localtime, strftime**
- et encore d'autres...

«« Précédent² Suivant »»³

²<http://www.truostonme.net/didactels/152.html>

³<http://www.truostonme.net/didactels/154.html>