La Bibliothèque Standard :

petit résumé de la bibliothèque définie par la norme ANSI.

Table des matières

Introduction

Cette bibliothèque ne fait pas partie du langage C proprement dit, mais son utilité faisant l'unanimité, on la trouvera dans la plupart des environnements de programmation.

Notation :

Inclure dans le texte le fichier fichier_en_tete.h peut s'écrire de deux façons :

a) #include <fichier_en_tete.h>

=> fichier_en_tete.h sera cherché dans la liste des répertoires contenant des fichiers d'en-tête, et déclaré la plupart du temps dans l'environnement du shell (sous un unix quelonque)

b) #include "/chemin/ fichier_en_tete.h"

Cette seconde notation n'est pas conseillée, sauf cas particulier.

Dans ce cas, le fichier est cherché dans le répertoire indiqué. Si seul le nom du fichier figure, le répertoire courant est utilisé. Parfois, les répertoires classiques /usr/include , /usr/local/include sont scrutés, et cela peut fonctionner, mais rien n'est garanti.

Les cas que l'on décrit plus loin concernent les fichiers d'en-tête suivants :

- sdtio.h (entrées-sorties)
- stdlib.h (fontions utilitaires)
- string.h (traitement des chaines de caractères)
- math.h (les fonctions mathématiques)
- stdarg.h (les listes variables d'arguments)
- signal.h (les signaux)
- ctype.h (tests catégories de caractères)
- float.h (limites définies par l'implémentation)
- setjmp.h (les branchements hors fonction)
- limits.h (limites définies par l'implémentation)
- assert.h (les messages d'erreurs)
- time.h (les fonctions de traitement de la date et de l'heure)

1) **<stdio.h>** *: gestion des Entrées-Sorties*

Un flot est une source ou une destination, qui peut etre associée à n'importe quel périphérique. Quel que soit le mode (texte ou binaire), ce flot est géré de la même manière.

Opérations gérées :
- opérations sur les fichiers (" f " pour files) : fopen, fread, fwrite,fclose freopen, fflush
- sorties mises en forme : printf, fprintf, sprintf, vprintf vfprintf, vsprintf
- entrees mises en forme : scanf, fscanf, sscanf
- entrées-sorties de caractères : fgetc, fgets, fputs, getc, getchar, putc, putchar, puts, ungetc
- entrèes-sorties directes : fseek, ftell, rewind, fgetpos, fsetpos

2) **<stdlibh>** *: les fonctions utilitaires*

- conversions de nombres : atof (ascii to float) , atoi (ascii to integer), atol, strtoul ...etc
- allocation mémoire : calloc, malloc, realloc, free
- fonctions systèmes : abort, exit, system, getenv, bsearch
- tri et generation de valeurs : qsort, rand, srand, abs, labs, div, ldiv
- macros comme EXIT_SUCCESS ..etc

3) **<string.h>** *: traitement des chaines de caractères*

Deux groupes de fonctions sont définies dans ce fichier d'en-tête.

Les premières commencent par str (string) : strcpy, sctncpy, strcat, strncat, strcmp, strncmp, strchr, strrchr, strcspn ... strlen, strerror ...

Les secondes par mem, faites pour manipuler des objets en tant que tableaux de caractères : memcpy, memmove, memchr, memset,

4) **<math.h>** *: les fonctions mathématiques*

Ce fichier d'en-tête contient des déclarations de fonctions et de macros mathématiques.
Liste (incomplète) des principales macros mathématiques utilisables :

x et y sont de type *double*, n est un *int* et toutes les fonctions retournent un *double* :

sin(x) , cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x), sinh(x), cosh(x), tanh(x), exp(x), log(x)n pow(x,y), sqrt(x), ceil(x), floor(x), fabs(x), ldexp(x,n), frexp(x), modf(x)

Note : lors de la compilation, il faut passer l'option **-lm** (link avec la libmath) à gcc car sinon, le linking échouera.

5) **<stdarg.h>** *: les listes variables d'arguments*

Ce fichier d'en-tête permet à une fonction d'utiliser une liste d'arguments de nombre et de type inconnus.

Voir la page de man de : va_list , va_start, va_end

6) **<signal.h>** *: les signaux*

Contient des fonctions qui permettent de traiter les événements exceptionnels qui se produisent durant l'éxecution (signal d'interruption, erreur d'éxecution ..etc)

Signaux autorisés : SIGABRT , SIGFPE, SIGILL , SIGINT, SIGSEGV, SIGTERM

=> voir signal, raise et les pages de man correspondantes sous Unix.

7) **<ctypes.h>** *: les tests de catégories de caractères*

Contient des déclarations de fonctions destinées à tester les caractères : isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper , isxdigit ..etc

+ tolower (conversion en minuscules) et toupper (conversion en majuscules)

8) **<float.h>** *: limites définies par l'implémentation*

Définit des constantes pour les tailles des types réels (en notation virgule flottante). Les valeurs définies sont des minimum (des valeurs supérieures, dépendant de l'implémentation, pouvant être utilisées)

9) **<setjmp.h>** *: les branchements hors fonctions*

Les objets déclarés dans <setjmp.h> permettent d'éviter la séquence normale d'appel et de retour de fonction, ce qui s'utilise essentiellement pour revenir immédiatement d'une fonction profondément imbriquée.

=> voir les pages de man (sous Unix) de setjmp, longjmp

10) **<limits.h>** *: limites définies par l'implémentation*

Définit des constantes pour les tailles des types entiers. Les valeurs définies sont des minimum (des valeurs supérieures pouvant être utilisées selon l'implémentation)

Exemples : CHAR_BIT, CHAR_MAX, INT_MIN ... etc

11) **<assert.h>** *: les messages d'erreurs*

Définit la macro ***assert*** utilisée pour insérer des messages d'erreur dans les programmes.

12) **<time.h>** *: les fonctions de traitement de la date de de l'heure*

Contient des déclarations de types et de fonctions qui aident à manipuler la date et l'heure.
Fonctions : clock_t , time_t , asctime , ctime, strftime ...etc
La structure tm contient les composantes d'une heure calendaire