# Annexe A

# La librairie standard

Cette annexe donne la syntaxe des principales fonctions de la librairie standard. Une liste exhaustive de toutes les fonctions disponibles figure à l'annexe B de l'ouvrage de Kernighan et Richie [6]. Pour obtenir plus d'informations sur ces fonctions, il suffit de consulter les pages de `man` correspondant.

## A.1 Entrées-sorties <stdio.h>

### A.1.1 Manipulation de fichiers

L'usage des fonctions de manipulation de fichiers suivantes est détaillé au chapitre 6, page 81.

| fonction | action |
|---|---|
| `fopen` | ouverture d'un fichier |
| `fclose` | fermeture d'un fichier |
| `fflush` | écriture des buffers en mémoire dans le fichier |

### A.1.2 Entrées et sorties formatées

La syntaxe de ces fonctions et leur action sont décrites aux paragraphes 1.11 et 6.2-6.3.

| fonction | prototype | action |
|---|---|---|
| `fprintf` | `int fprintf(FILE *stream, char *format, ...)` | écriture sur un fichier |
| `fscanf` | `int fscanf(FILE *stream, char *format, ...)` | lecture depuis un fichier |
| `printf` | `int printf(char *format, ...)` | écriture sur la sortie standard |
| `scanf` | `int scanf(char *format, ...)` | lecture depuis l'entrée standard |
| `sprintf` | `int sprintf(char *s, char *format, ...)` | écriture dans la chaîne de caractères `s` |
| `sscanf` | `int sscanf(char *s, char *format, ...)` | lecture depuis la chaîne de caractères `s` |

### A.1.3 Impression et lecture de caractères

| fonction | prototype | action |
|---|---|---|
| `fgetc` | `int fgetc(FILE *stream)` | lecture d'un caractère depuis un fichier |
| `fputc` | `int fputc(int c, FILE *stream)` | écriture d'un caractère sur un fichier |
| `getc` | `int getc(FILE *stream)` | équivalent de `fgetc` mais implémenté par une macro |
| `putc` | `int putc(int c, FILE *stream)` | équivalent de `fputc` mais implémenté par une macro |
| `getchar` | `int getchar(void)` | lecture d'un caractère depuis l'entrée standard |
| `putchar` | `int putchar(int c)` | écriture d'un caractère sur la sortie standard |
| `fgets` | `char *fgets(char *s, FILE *stream)` | lecture d'une chaîne de caractères depuis un fichier |
| `fputs` | `int *fputs(char *s, FILE *stream)` | écriture d'une chaîne de caractères sur un fichier |
| `gets` | `char *gets(char *s)` | lecture d'une chaîne de caractères sur l'entrée standard |
| `puts` | `int *puts(char *s)` | écriture d'une chaîne de caractères sur la sortie standard |

## A.2 Manipulation de caractères <ctype.h>

Toutes les fonctions ci-dessous permettent de tester une propriété du caractère passé en paramètre. Elles renvoient la valeur 1 si le caractère vérifie la propriété et 0 sinon. Leur prototype est :

```
int fonction(char c)
```

| fonction | renvoie 1 si le caractère est |
|---|---|
| `isalnum` | une lettre ou un chiffre |
| `isalpha` | une lettre |
| `iscntrl` | un caractère de commande |
| `isdigit` | un chiffre décimal |
| `isgraph` | un caractère imprimable ou le blanc |
| `islower` | une lettre minuscule |
| `isprint` | un caractère imprimable (pas le blanc) |
| `ispunct` | un caractère imprimable qui n'est ni une lettre ni un chiffre |
| `isspace` | un blanc |
| `isupper` | une lettre majuscule |
| `isxdigit` | un chiffre hexadécimal |

On dispose également de deux fonctions permettant la conversion entre lettres minuscules et lettres majuscules :

| fonction | prototype | action |
|---|---|---|
| `tolower` | `int tolower(int c)` | convertit `c` en minuscule si c'est une lettre majuscule, retourne `c` sinon. |
| `toupper` | `int toupper(int c)` | convertit `c` en majuscule si c'est une lettre minuscule, retourne `c` sinon. |

## A.3 Manipulation de chaînes de caractères <string.h>

| fonction | prototype | action |
|---|---|---|
| `strcpy` | `char *strcpy(char *ch1, char *ch2)` | copie la chaîne `ch2` dans la chaîne `ch1` ; retourne `ch1`. |
| `strncpy` | `char *strcpy(char *ch1, char *ch2, int n)` | copie `n` caractères de la chaîne `ch2` dans la chaîne `ch1` ; retourne `ch1`. |
| `strcat` | `char *strcat(char *ch1, char *ch2)` | copie la chaîne `ch2` à la fin de la chaîne `ch1` ; retourne `ch1`. |
| `strncat` | `char *strncat(char *ch1, char *ch2, int n)` | copie `n` caractères de la chaîne `ch2` à la fin de la chaîne `ch1` ; retourne `ch1`. |
| `strcmp` | `int strcmp(char *ch1, char *ch2)` | compare `ch1` et `ch2` pour l'ordre lexicographique ; retourne une valeur négative si `ch1` est inférieure à `ch2`, une valeur positive si `ch1` est supérieure à `ch2`, 0 si elles sont identiques. |
| `strncmp` | `int strcmp(char *ch1, char *ch2, int n)` | compare les `n` premiers caractères de `ch1` et `ch2`. |
| `strchr` | `char *strchr(char *chaine, char c)` | retourne un pointeur sur la première occurence de `c` dans `chaine`, et `NULL` si `c` n'y figure pas. |
| `strrchr` | `char *strchr(char *chaine, char c)` | retourne un pointeur sur la dernière occurence de `c` dans `chaine`, et `NULL` si `c` n'y figure pas. |
| `strstr` | `char *strchr(char *ch1, char *ch2)` | retourne un pointeur sur la première occurence de `ch2` dans `ch1`, et `NULL` si `ch2` n'y figure pas. |
| `strlen` | `int strlen(char *chaine)` | retourne la longueur de `chaine`. |

## A.4 Fonctions mathématiques <math.h>

Le résultat et les paramètres de toutes ces fonctions sont de type `double`. Si les paramètres effectifs sont de type `float`, ils seront convertis en `double` par le compilateur.

| fonction | action |
|---|---|
| `acos` | arc cosinus |
| `asin` | arc sinus |
| `atan` | arc tangente |
| `cos` | cosinus |
| `sin` | sinus |
| `tan` | tangente |
| `cosh` | cosinus hyperbolique |
| `sinh` | cosinus hyperbolique |
| `tanh` | tangente hyperbolique |
| `exp` | exponentielle |
| `log` | logarithme népérien |
| `log10` | logarithme en base 10 |
| `pow` | puissance |
| `sqrt` | racine carrée |
| `fabs` | valeur absolue |
| `fmod` | reste de la division |
| `ceil` | partie entière supérieure |
| `floor` | partie entière inférieure |

## A.5 Utilitaires divers <stdlib.h>

### A.5.1 Allocation dynamique

Ces fonctions sont décrites au chapitre 3, paragraphe 3.4.

| fonction | action |
|---|---|
| calloc | allocation dynamique et initialisation à zéro. |
| malloc | allocation dynamique |
| realloc | modifie la taille d'une zone préalablement allouée par calloc ou malloc. |
| free | libère une zone mémoire |

### A.5.2 Conversion de chaînes de caractères en nombres

Les fonctions suivantes permettent de convertir une chaîne de caractères en un nombre.

| fonction | prototype | action |
|---|---|---|
| atof | double atof(char *chaine) | convertit chaine en un double |
| atoi | int atoi(char *chaine) | convertit chaine en un int |
| atol | long atol(char *chaine) | convertit chaine en un long int |

### A.5.3 Génération de nombres pseudo-aléatoires

La fonction rand fournit un nombre entier pseudo-aléatoire dans l'intervalle [0,RAND_MAX], où RAND_MAX est une constante prédéfinie au moins égale à $2^{15} - 1$. L'aléa fournit par la fonction rand n'est toutefois pas de très bonne qualité.

La valeur retournée par rand dépend de l'initialisation (germe) du générateur. Cette dernière est égale à 1 par défaut mais elle peut être modifiée à l'aide de la fonction srand.

| fonction | prototype | action |
|---|---|---|
| rand | int rand(void) | fournit un nombre entier pseudo-aléatoire |
| srand | void srand(unsigned int germe) | modifie la valeur de l'initialisation du générateur pseudo-aléatoire utilisé par rand. |

### A.5.4 Arithmétique sur les entiers

| fonction | prototype | action |
|---|---|---|
| abs | int abs(int n) | valeur absolue d'un entier |
| labs | long labs(long n) | valeur absolue d'un long int |
| div | div_t div(int a, int b) | quotient et reste de la division euclidienne de a par b. Les résultats sont stockés dans les champs quot et rem (de type int) d'une structure de type div_t. |
| ldiv | ldiv_t ldiv(long a, long b) | quotient et reste de la division euclidienne de a par b. Les résultats sont stockés dans les champs quot et rem (de type long int) d'une structure de type ldiv_t. |

### A.5.5 Recherche et tri

Les fonctions `qsort` et `bsearch` permettent respectivement de trier un tableau, et de rechercher un élément dans un tableau déjà trié. Leur syntaxe est détaillée au chapitre 4, page 71.

### A.5.6 Communication avec l'environnement

| fonction | prototype | action |
|---|---|---|
| `abort` | `void abort(void)` | terminaison anormale du programme |
| `exit` | `void exit(int etat)` | terminaison du programme ; rend le contrôle au système en lui fournissant la valeur `etat` (la valeur 0 est considérée comme une fin normale). |
| `system` | `int system(char *s)` | exécution de la commande système définie par la chaîne de caractères `s`. |

## A.6 Date et heure <`time.h`>

Plusieurs fonctions permettent d'obtenir la date et l'heure. Le temps est représenté par des objets de type `time_t` ou `clock_t`, lesquels correspondent généralement à des `int` ou à des `long int`.

| fonction | prototype | action |
|---|---|---|
| `time` | `time_t time(time_t *tp)` | retourne le nombre de secondes écoulées depuis le $1^{er}$ janvier 1970, 0 heures G.M.T. La valeur retournée est assignée à `*tp`. |
| `difftime` | `double difftime(time_t t1, time_t t2)` | retourne la différence `t1 - t2` en secondes. |
| `ctime` | `char *ctime(time_t *tp)` | convertit le temps système `*tp` en une chaîne de caractères explicitant la date et l'heure sous un format prédéterminé. |
| `clock` | `clock_t clock(void)` | retourne le temps CPU en microsecondes utilisé depuis le dernier appel à `clock`. |