

# Printf

<stdio.h>

```
int printf( const char * format, ... );
```

Écrit l'argument *format* sur la sortie standard *stdout*. Si *format* contient des spécificateurs (chaîne commençant par le caractère %), les arguments à la suite de *format* sont insérés à la place de leur spécificateur (il doit y avoir autant d'arguments après *format* que de "%..." dans *format*). De plus, le type des spécificateurs (%s, %c, %d, ...) détermine la façon dont est converti chaque argument.

Exemple:

```
printf( "%d est le code ASCII du caractère %c", 65, 65 )
```



Affiche sur *stdout*:

"65 est le code ASCII du caractère A"

Pour plus d'informations: *man printf*

## Principaux spécificateurs

Spécificateur	Type
%d	Entier (1, 2, -3, ...)
%f	Décimal (-1.0, 3.14, ...)
%s	Chaîne de caractères ("abc", ...)
%c	Caractère ('a', 'b', ...)

## Autres exemples

Code	Résultat
<pre>printf("hello world");</pre>	hello
<pre>printf("hello world\n");</pre>	hello world↵
<pre>printf("Il est %d heures et %d minutes", 10, 30);</pre>	Il est 10 heures et 30 minutes
<pre>int h=10; int m=30; printf("Il est %d heures et %d minutes", h, m);</pre>	Il est 10 heures et 30 minutes
<pre>printf("Je m'appelle %s", "Tom");</pre>	Je m'appelle Tom
<pre>printf("%s a %d ans", "Tom", 20);</pre>	Tom a 20 ans
<pre>char * nom = "Tom"; int age = 20; printf("%s a %d ans", nom, age);</pre>	Tom a 20 ans
<pre>printf("pi = %f", 3.14);</pre>	pi = 3.14
<pre>printf("%c = %f", 'x', 1.0);</pre>	x = 1.0

# Fprintf

<stdio.h>

```
int fprintf( fd, const char * format, ... );
```

Fait la même chose que *printf*, mais écrit dans le fichier *fd* plutôt que dans *stdout*.

Remarque:

*stdout* et *stderr* sont deux descripteurs de fichiers spéciaux (ouverts automatiquement). On peut les utiliser à la place de *fd*.

*fprintf( stdout, "hello" )* équivaut à *printf( "hello" )*.

Pour plus d'informations: *man fprintf*

## Exemples

Code	Résultat
<pre>FILE * fd = fopen("foo.txt", "w"); printf(fd, "hello world!"); fclose(fd);</pre>	Écrit "hello world!" dans le fichier "foo.txt"
<pre>printf(stdout, "Il est %d heures et %d minutes", 10, 30);</pre>	Écrit "Il est 10 heures et 30 minutes" sur la sortie standard <i>stdout</i> (ie. sur le terminal)
<pre>printf(stderr, "Il est %d heures et %d minutes", 10, 30);</pre>	Écrit "Il est 10 heures et 30 minutes" sur la sortie standard des erreurs <i>stderr</i> (ie. sur le terminal)

# Scanf

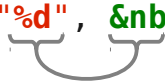
<stdio.h>

```
int scanf( const char * format, ... );
```

Lit, converti et stock les données saisies sur *stdin* (entrée standard du terminal) d'après l'argument *format*. Les spécificateurs contenus dans *format* sont les mêmes que pour *printf* (%d, %s, %c, %f, ...). Les arguments à la suite de *format* doivent être des **pointeurs** vers des variables. Il doit y avoir autant de pointeurs après l'argument *format* que de spécificateurs dans la chaîne *format* (généralement on préférera n'utiliser qu'un seul spécificateur/pointeur par appel à *scanf*). Le type des spécificateurs (%s, %c, %d, ...) détermine la façon dont est converti la valeur envoyée à chaque **pointeur**.

Exemple:

```
int nb;  
scanf( "%d", &nb )
```



Si l'utilisateur écrit les **caractères** 123 sur le terminal puis appuie sur la touche <Entrée>, alors cette suite de **caractères** '1','2','3' est convertie en un nombre **entier** valant 123 (car on a utilisé le descripteur %d) et cet entier est stocké dans la variable *nb*

Pour plus d'informations: *man scanf*

## Autres exemples

Code	Résultat
<pre>int nb; scanf("%d", &amp;nb);</pre>	Si l'utilisateur écrit 123 ↵  alors <i>nb</i> = 123
<pre>float x; scanf("%f", &amp;x);</pre>	Si l'utilisateur écrit 3.14 ↵  alors <i>x</i> = 3.14
<pre>char c; scanf("%c", &amp;c);</pre>	Si l'utilisateur écrit A ↵  alors <i>c</i> = 'A'
<pre>char str[80]; scanf("%s", str);</pre> <p><b>ATTENTION:</b> <i>str</i> est déjà un pointeur, on ne met pas de &amp; ici!</p>	Si l'utilisateur écrit hello ↵  alors <i>str</i> = "hello"
<pre>int n1; int n2; scanf("%d%d", &amp;n1, &amp;n2);</pre> <p><b>REMARQUE:</b> il vaut mieux éviter ça et se contenter de faire deux <i>scanf</i>...</p>	Si l'utilisateur écrit 123_456 ↵ ( _ est un espace)  alors <i>n1</i> = 123 et <i>n2</i> = 456

# Fscanf

<stdio.h>

```
int fscanf( fd, const char * format, ... );
```

Fait la même chose que *scanf*, mais lit dans le fichier *fd* plutôt que dans *stdin*.

Remarque:

*stdin* est un descripteur de fichier spécial (ouvert automatiquement). On peut l'utiliser à la place de *fd*.

*fscanf( stdin, "%d", &var )* équivaut à *scanf( "%d", &var )*.

Pour plus d'informations: *man fscanf*