printf

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

Print formatted data to stdout

Writes the C string pointed by *format* to the standard output.

A *format specifier* follows this prototype:
`%[flags][width][.precision][length]specifier`

Where the *specifier character* at the end is the most significant component, since it defines the type and the interpretation of its corresponding argument:

specifier	**Output**	**Example**
`d` *or* `i`	Signed decimal integer	`392`
`u`	Unsigned decimal integer	`7235`
`o`	Unsigned octal	`610`
`x`	Unsigned hexadecimal integer	`7fa`
`X`	Unsigned hexadecimal integer (uppercase)	`7FA`
`f`	Decimal floating point, lowercase	`392.65`
`F`	Decimal floating point, uppercase	`392.65`
`e`	Scientific notation (mantissa/exponent), lowercase	`3.9265e+2`
`E`	Scientific notation (mantissa/exponent), uppercase	`3.9265E+2`
`g`	Use the shortest representation: `%e` or `%f`	`392.65`
`G`	Use the shortest representation: `%E` or `%F`	`392.65`
`a`	Hexadecimal floating point, lowercase	`-0xc.90fep-2`
`A`	Hexadecimal floating point, uppercase	`-0XC.90FEP-2`
`c`	Character	`a`
`s`	String of characters	`sample`
`p`	Pointer address	`b8000000`
`n`	Nothing printed. The corresponding argument must be a pointer to a `signed int`. The number of characters written so far is stored in the pointed location.	
`%`	A `%` followed by another `%` character will write a single `%` to the stream.	`%`

Return Value

On success, the total number of characters written is returned.

If a writing error occurs, the *error indicator* (ferror) is set and a negative number is returned.

If a multibyte character encoding error occurs while writing wide characters, errno is set to `EILSEQ` and a negative number is returned.

Example

```
/* printf example */
#include <stdio.h>

int main()
{
   printf ("Characters: %c %c \n", 'a', 65);
   printf ("Decimals: %d %ld\n", 1977, 650000L);
   printf ("Preceding with blanks: %10d \n", 1977);
   printf ("Preceding with zeros: %010d \n", 1977);
   printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100,
100, 100);
   printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
   printf ("Width trick: %*d \n", 5, 10);
```

```
   printf ("%s \n", "A string");
   return 0;
}
```

Output:

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:       1977
Preceding with zeros: 0000001977
Some different radices: 100 64 144 0x64 0144
floats: 3.14 +3e+000 3.141600E+000
Width trick:    10
A string
```

scanf

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

Read formatted data from stdin

A *format specifier* for `scanf` follows this prototype:

```
%[*][width][length]specifier
```

Where the *specifier* character at the end is the most significant component, since it defines which characters are extracted, their interpretation and the type of its corresponding argument:

specifier	Description	Characters extracted
i	Integer	Any number of digits, optionally preceded by a sign (+ or −). Decimal digits assumed by default (0−9), but a 0 prefix introduces octal digits (0−7), and 0x hexadecimal digits (0−f). *Signed* argument.
d *or* u	Decimal integer	Any number of decimal digits (0−9), optionally preceded by a sign (+ or −). d is for a *signed* argument, and u for an *unsigned*.
o	Octal integer	Any number of octal digits (0−7), optionally preceded by a sign (+ or −). *Unsigned* argument.
x	Hexadecimal integer	Any number of hexadecimal digits (0−9, a−f, A−F), optionally preceded by 0x or 0X, and all optionally preceded by a sign (+ or −). *Unsigned* argument.
f, e, g a	Floating point number	A series of decimal digits, optionally containing a decimal point, optionally preceeded by a sign (+ or −) and optionally followed by the e or E character and a decimal integer (or some of the other sequences supported by `strtod`). Implementations complying with C99 also support hexadecimal floating-point format when preceded by 0x or 0X.
c	Character	The next character. If a *width* other than 1 is specified, the function reads exactly*width* characters and stores them in the successive locations of the array passed as argument. No null character is appended at the end.
s	String of characters	Any number of non-whitespace characters, stopping at the first whitespace character found. A terminating null character is automatically added at the end of the stored sequence.
p	Pointer address	A sequence of characters representing a pointer. The particular format used depends on the system and library implementation, but it is the same as the one used to format %p in fprintf.
[*characters*]	Scanset	Any number of the characters specified between the brackets. A dash (−) that is not the first character may produce non-portable behavior in some library implementations.
[^*characters*]	Negated scanset	Any number of characters none of them specified as *characters* between the brackets.
n	Count	No input is consumed. The number of characters read so far from `stdin` is stored in the pointed location.
%	%	A % followed by another % matches a single %.

Return Value

On success, the function returns the number of items of the argument list successfully filled. This count can match the expected number of items or be less (even zero) due to a matching failure, a reading error, or the reach of the *end-of-file*.

If a reading error happens or the *end-of-file* is reached while reading, the proper indicator is set (feof or ferror). And, if either happens before any data could be successfully read, EOF is returned.

If an encoding error happens interpreting wide characters, the function sets errno to EILSEQ.

Example

Edit & Run

```c
/* scanf example */
#include <stdio.h>

int main ()
{
  char str [80];
  int i;

  printf ("Enter your family name: ");
  scanf ("%79s",str);
  printf ("Enter your age: ");
  scanf ("%d",&i);
  printf ("Mr. %s , %d years old.\n",str,i);
  printf ("Enter a hexadecimal number: ");
  scanf ("%x",&i);
  printf ("You have entered %#x (%d).\n",i,i);

  return 0;
}
```

This example demonstrates some of the types that can be read with scanf:

```
Enter your family name: Soulie
Enter your age: 29
Mr. Soulie , 29 years old.
Enter a hexadecimal number: ff
You have entered 0xff (255).
```