scanf - Read formatted data from the standard input stream.

```c
int scanf ( const char* format [, argument]... ) ;
```

**Parameters**

- `format` - Format control string.
- `argument` - Optional arguments.

**Return Value**

The number of fields converted and assigned; the return value does not include fields that were read but not assigned.

A return value of 0 indicates that no fields were assigned.

The return value is `EOF` for an error or if the end-of-file character or the end-of-string character is encountered.

**Remarks**

The `scanf` function reads data from the standard input stream `stdin` and writes the data into the location given by `argument`.

Each argument must be a pointer to a variable of a type that corresponds to a type specifier in `format`.

**Type Characters for scanf functions**

<table>
<thead>
<tr>
<th>Character</th>
<th>Type of Input Expected</th>
<th>Type of Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>c</code></td>
<td>Single byte character.</td>
<td>Pointer to char.</td>
</tr>
<tr>
<td><code>d/ld</code></td>
<td>Decimal integer/long integer.</td>
<td>Pointer to int/long int.</td>
</tr>
<tr>
<td><code>f</code></td>
<td>Floating point value.</td>
<td>Pointer to float.</td>
</tr>
<tr>
<td><code>lf</code></td>
<td>Double value.</td>
<td>Pointer to double.</td>
</tr>
</tbody>
</table>

```c
#include <stdio.h>

int main()
{
    int   i, result;
    float fp;
    char  c;

    printf( "Enter an int, a float and a char:\n" );
    result = scanf( "%d %f %c", &i, &fp, &c);
    printf( "The number of fields input is %d.\n", result);
    printf( "The contents are: %d %f %c.\n", i, fp, c);
    return 0;
}
```

Enter an int, a float and a char:
71 98.6 h
The number of fields input is 3.
The contents are: 71 98.599998 h.
Always check for return value from scanf function!

```c
#include <stdio.h>

int main()
{
    char ch;
    int num1, num2;
    int result;

    printf("Reading two integer numbers...
");
    result = scanf("\d \d", &num1, &num2);
    if (result == 0) {
        printf("Read no integers.\n");
    } else if (result == EOF) {
        printf("Ops, the file is empty.\n");
        return 1; /* Return 1 for unsuccessful termination. */
    }
    printf("Number of successful inputs: %d.\n", result);

    printf("Reading a character...
");
    result = scanf("%c", &ch);
    if (result == -1) { /* EOF = -1 */
        printf("Read no character.\n");
        return 1;
    }
    printf("Number of successful inputs: %d.\n", result);

    return 0;
}
```

```
> test.exe < input.dat

<table>
<thead>
<tr>
<th>input.dat</th>
<th>result</th>
<th>variable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4 h</td>
<td></td>
<td>num1 = 3, num2 = 4, ch = 'h'</td>
</tr>
<tr>
<td></td>
<td>Reading two integer numbers...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of successful inputs: 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reading a character...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of successful inputs: 1.</td>
<td></td>
</tr>
<tr>
<td>2.3 3 h</td>
<td></td>
<td>num1 = 2, num2 = 4206636, ch = 'h'</td>
</tr>
<tr>
<td></td>
<td>Reading two integer numbers...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of successful inputs: 1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reading a character...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of successful inputs: 1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ops, the file is empty.</td>
<td></td>
</tr>
<tr>
<td>4 5</td>
<td></td>
<td>num1 = 4, num2 = 5, ch = ''</td>
</tr>
<tr>
<td></td>
<td>Reading two integer numbers...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of successful inputs: 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reading a character...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read no character.</td>
<td></td>
</tr>
</tbody>
</table>
```
printf - Print formatted output to the standard output stream.

```c
int printf (const char* format [, argument]...);
```

**Parameters**

- `format` - Format control (%[flags] [width] [.precision] type)
- `argument` - Optional arguments.

**Return Value**

The number of characters printed, or a negative value if an error occurs.

**Remarks**

The `printf` function formats and prints a series of characters and values to the standard output stream, `stdout`.

**Example:**

```c
#include <stdio.h>

int main()
{
    char gender = 'm';
    int age = 34;
    int weight_lb = 165;
    int height_f = 5;
    int height_i = 11;
    int height_cm = 180;
    double weight_kg = 74.8427411;

    printf("Athlete:Lance Armstrong\n");
    printf("Gender:\t%c\n", gender);
    printf("Age:\t%d\n", age);
    printf("Weight:\t%d lbs/%.2lf kgs\n", weight_lb, weight_kg);
    printf("Height:\t%d cms/%d feet %d inches\n", height_cm,
                  height_f, height_i);

    return 0;
}
```

Athlete:Lance Armstrong  
Gender: m  
Age: 34  
Weight: 165 lbs/74.84 kgs  
Height: 180 cms/5 feet 11 inches
printf Type Field Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Type</th>
<th>Output format</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>int</td>
<td>Specifies a single-byte character.</td>
</tr>
<tr>
<td>d</td>
<td>int</td>
<td>Signed decimal integer.</td>
</tr>
<tr>
<td>o</td>
<td>int</td>
<td>Unsigned octal integer.</td>
</tr>
<tr>
<td>u</td>
<td>int</td>
<td>Unsigned decimal integer.</td>
</tr>
<tr>
<td>x</td>
<td>int</td>
<td>Unsigned hexadecimal integer.</td>
</tr>
<tr>
<td>e</td>
<td>double</td>
<td>Signed value having the form [-]d.dddd e [sign]dd[d] where d is a single decimal digit, dddd is one or more decimal digits, dd[d] is two or three decimal digits depending on the output format and size of the exponent, and sign is + or -.</td>
</tr>
<tr>
<td>f</td>
<td>double</td>
<td>Signed value having the form [-]ddd.dddd, where dddd is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision.</td>
</tr>
<tr>
<td>p</td>
<td>Pointer to void</td>
<td>Prints the address of the argument in hexadecimal digits.</td>
</tr>
</tbody>
</table>

The first optional field of the format specification is flags. A flag directive is a character that justifies output and prints signs, blanks, decimal points, and octal and hexadecimal prefixes. More than one flag directive may appear in a format specification.

Flag Characters

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Left align the result within the given field width.</td>
<td>Right align.</td>
</tr>
<tr>
<td>+</td>
<td>Prefix the output value with a sign (+ or -) if the output value is of a signed type.</td>
<td>Sign appears only for negative signed values (-).</td>
</tr>
<tr>
<td>0</td>
<td>If width is prefixed with 0, zeros are added until the minimum width is reached. If 0 and - appear, the 0 is ignored.</td>
<td>No padding.</td>
</tr>
<tr>
<td>blank (' ')</td>
<td>Prefix the output value with a blank if the output value is signed and positive; the blank is ignored if both the blank and + flags appear.</td>
<td>No blank appears.</td>
</tr>
</tbody>
</table>
#include <stdio.h>

int main()
{
    char ch = 'k';
    int int1 = 10,
            int2 = -10;
    float f = 8.1234;
    double d1 = 12.123456789,
             d2 = -12.123456789,
             d3 = 0.123456789;

    printf("INTEGER PRINTING:\n");
    printf("=\n");
    printf("Printing char: %c\n", ch);
    printf("Printing decimal integer: %d\n", int2);
    printf("Printing decimal integer: %d\n", int1);
    printf("Printing unsigned octal integer: %o\n", int1);
    printf("Printing unsigned decimal integer: %u\n", int2);
    printf("Printing unsigned hexadecimal integer: %x\n\n", int1);

    printf("FLOATING POINT PRINTING:\n");
    printf("=\n");
    printf("Printing float: %f\n", f);
    printf("Printing double: %.9e\n", d1);
    printf("Printing double: %.9e\n", d2);
    printf("Printing double: %.9e\n", d3);
    printf("Printing double: %lf\n\n", d1);

    printf("ADDRESS PRINTING:\n");
    printf("=\n");
    printf("Printing address: %p\n\n", d1);

    printf("ALIGNMENT:\n");
    printf("=\n");
    printf("Right (default) align: *%7d*\n", int1);
    printf("Left align: *%-7d*\n\n", int1);

    printf("SIGN:\n");
    printf("=\n");
    printf("Sign prefix: +%d\n", int1);

    printf("ZERO PADDING:\n");
    printf("=\n");
    printf("Zero padding: 00010\n\n", int1);

    printf("BLANK PREFIX:\n");
    printf("=\n");
    printf("No blank prefix: %d\n", int1);
    printf("Blank prefix: *%+d*\n", int1);
    return 0;
}