Chapter Contents Previous Next

fscanf

fscanf

Read Formatted Input from a File

Portability: ISO/ANSI C conforming, UNIX compatible

SYNOPSIS

```
#include <stdio.h>

int fscanf(FILE *f, const char *format, loc1, loc2, ...);
```

DESCRIPTION

fscanf reads formatted input from the **FILE** designated by **f** according to the format specified by the string **format** . Following the format in the argument list may be one or more additional pointers (**loc1, loc2,..., locn**), addressing storage where the input values are stored.

format points to a string that contains zero or more of the following:

- white-space characters
- regular characters (not including %)
- conversion specifications.

The format string contains format specifiers or characters to be matched from the input. Format items have the following form:

```
%[*][ {OB} width {OBE} ][h | l | L | hh | z | t | ll | j]form
```

The specifiers have the following meanings:

- An asterisk (*) indicates that an input item is processed according to the format, but its value is not stored.

- If a value for **width** is present, **width** specifies the maximum width of the input item.
- An optional letter has the following meanings:
  - An **hh** before a **d**, **i**, or **n** conversion specifier indicates that the corresponding argument is a pointer to **char** instead of **int**.
  - An **h** before a **d**, **i**, or **n** conversion specifier indicates that the corresponding argument is a pointer to **short int** instead of **int**.
  - An **l**, **z**, or **t** before a **d**, **i**, or **n** conversion specifier indicates that the corresponding argument is a pointer to **long int** instead of **int**.
  - An **ll**, or **j** before a **d**, **i**, or **n** conversion specifier indicates that the corresponding argument is a pointer to **long long int** instead of **int**.
  - An **hh** before an **o**, **u**, or **x** conversion specifier indicates that the corresponding argument is a pointer to **unsigned char** instead of **unsigned int**.
  - An **h** before an **o**, **u**, or **x** conversion specifier indicates that the corresponding argument is a pointer to **unsigned short int** instead of **unsigned int**.
  - An **l**, **z**, or **t** before an **o**, **u**, or **x** conversion specifier indicates that the corresponding argument is a pointer to **unsigned long int** instead of **unsigned int**.
  - An **ll**, or **j** before an **o**, **u**, or **x** conversion specifier indicates that the corresponding argument is a pointer to **unsigned long long int** instead of **unsigned int**
  - An **l** before an **e**, **f**, or **g** conversion specifier indicates that the corresponding argument is a pointer to **double** instead of **float**.
  - An **L** before an **e**, **f**, or **g** conversion specifier indicates that the corresponding argument is a pointer to **long double** instead of **float**.
- **form** is one of the following characters, defining the type of the corresponding target object and the expected format of the input:

  c matches a sequence of characters specified by width. If no width is specified, one character is expected. A null character is not added. The corresponding argument should point to an array large enough to hold the sequence.

  d matches an optionally signed decimal integer whose format is the same as expected for the subject sequence of **strtol** with **base=10** . The corresponding argument should be **int *** .

  e , **E** , **f** , **g** , or **G** matches a floating-point number. The corresponding argument should be **float *** .

  i matches an optionally signed decimal integer, which may be expressed in decimal, in octal with a leading 0, or in hexadecimal with a leading 0x. The corresponding argument should be **int *** .

  n indicates that no input is consumed. The number of characters read from the input stream so far by this call to **fscanf** is stored in the object addressed by the corresponding **int *** argument.

  o matches an optionally signed octal integer. The corresponding argument should be **unsigned int *** .

p matches a pointer in the format written by the **%p printf** format. This implementation treats **%p** like **%x** . The corresponding argument should be **void **** .

s matches a sequence of nonwhite-space characters. A terminating null character is automatically added. The corresponding argument should point to an array large enough to hold the sequence plus the terminating null character.

u matches an optionally signed integer. The corresponding argument should be **unsigned int *** .

x , **X** matches a hexadecimal integer. The corresponding argument should be **unsigned int *** .

[ ] or < > matches a string comprised of a particular set of characters. A terminating-null character is automatically added. The corresponding argument should point to an array large enough to hold the sequence plus the terminating-null character. Note that you cannot use the two-character sequences (| and |) to replace the brackets in a **fscanf** format.

The format string is a C string. With the exception of the **c** and **[** or **<** specifiers, white-space characters in the format string cause white-space characters in the input to be skipped. Characters other than format specifiers are expected to match the next nonwhite-space character in the input. The input is scanned through white space to locate the next input item in all cases except the **c** and **[ ]** specifiers, where the initial scan is bypassed. The **s** specifier terminates on any white space.

The **fscanf** formats are described in more detail in the ISO/ANSI C standard. As an extension, uppercase characters may also be used for the format characters specified in lowercase in the previous list.

RETURN VALUE

fscanf returns **EOF** if end of file (or an input error) occurs before any values are stored. If values are stored, it returns the number of items stored; that is, the number of times a value is assigned with one of the **fscanf** argument pointers.

DIAGNOSTICS

EOF is returned if an error occurs before any items are matched.

IMPLEMENTATION

The format string can also contain multibyte characters. For details on how **fscanf** treats multibyte characters in the format string and in conversions, see Chapter 11 in the **SAS/C Library Reference, Volume 2**.

Because square brackets do not exist on some 370 I/O devices, the library allows the format **%[xyz]** to be replaced by the alternate form **%<xyz>** . This is not a portable format.

EXAMPLE

This example writes out the data stored in lines to a temporary file, and reads them back with **fscanf** :

```
#include <stdio.h>
#include <stdlib.h>

static char *lines[] = {
```

```
    "147.8 pounds\n"
    "51.7 miles\n",
    "4.3 light-years\n",
    "10000 volts\n",
    "19.5 gallons\n"
};

main()
{
   FILE *tmpf;
   int i;
   float amount;
   char unit[20];
   int count;

   tmpf = tmpfile();
   if (!tmpf){
         puts("Couldn't open temporary file.");
         exit(EXIT_FAILURE);
   }

   for (i = 0; i < sizeof(lines)/sizeof(char *); ++i){
      fputs(lines[i], tmpf);
   }
   rewind(tmpf);
   for(;;){
      count = fscanf(tmpf, "%f %s", &amount, unit);
      if (feof(tmpf)) break;
      if (count < 2){
         puts("Unexpected error in input data.");
         exit(EXIT_FAILURE);
      }
         printf("amount = %f, units = \"%s\"\n", amount, unit);
   }
   fclose(tmpf);
}
```

RELATED FUNCTIONS

fprintf , **scanf** , **sscanf**

SEE ALSO

- I/O Functions
- I/O Functions

Chapter Contents Previous Next Top of Page