# C printf / scanf Formatting Guide

*Complete quick reference for formatted output and input*

**Reference platform: 64-bit Ubuntu Linux with GCC**
C17-portable features are emphasized. C23 additions are identified separately.

## 1. Format String Anatomy

A conversion specification begins with % and tells printf or scanf how to interpret one argument. Ordinary characters in the format string are copied by printf or matched by scanf.

### printf family

```
%[flags][width][.precision][length]conversion
```

### scanf family

```
%[*][width][length]conversion
```

The printf and scanf families use similar-looking format strings, but their rules and argument types are not interchangeable.

## 2. printf Family

Functions include printf, fprintf, sprintf, snprintf, vprintf, vfprintf, vsprintf, and vsnprintf. They return the number of characters written, excluding the terminating null character. A negative result indicates an output or encoding error.

### printf Flags

| Flag | Meaning |
|---|---|
| - | Left-align within the field. The default is right alignment. |
| + | Always print a sign for signed numeric conversions. |
| space | Prefix a positive signed number with a space when + is absent. |
| # | Alternate form: prefixes octal/hex/binary, forces decimal point for floating output, and preserves trailing zeros for g/G. |
| 0 | Pad numeric output with leading zeros. Ignored when - is present; integer precision also overrides it. |

### Field Width and Precision

| Component | Meaning |
|---|---|
| width | Minimum field width. Output expands when needed; it is never truncated merely by width. |
| * | Take the width from the next int argument. A negative width implies left alignment. |
| .precision | Meaning depends on the conversion: digits, decimal places, significant digits, or maximum string bytes. |
| .* | Take precision from the next int argument. A negative precision is treated as omitted. |

```
printf("|%10d| |%-10d| |%08d|\n", 42, 42, 42);
printf("|%10.2f| |%.*f|\n", 3.14159, 4, 3.14159);
```

### printf Conversion Specifiers

| Specifier | Output | Expected argument |
|---|---|---|

| | | |
|---|---|---|
| %d, %i | Signed decimal integer | int |
| %u | Unsigned decimal integer | unsigned int |
| %o | Unsigned octal integer | unsigned int |
| %x, %X | Unsigned hexadecimal integer | unsigned int |
| %b, %B | Unsigned binary integer (C23) | unsigned int |
| %f, %F | Fixed-point floating notation | double |
| %e, %E | Scientific notation | double |
| %g, %G | Shortest of f/e style based on magnitude and precision | double |
| %a, %A | Hexadecimal floating notation | double |
| %c | Single character | int (converted to unsigned char) |
| %s | Null-terminated character string | char * |
| %p | Pointer value in implementation-defined form | void * |
| %n | Store characters written so far; produces no output | pointer matching length modifier |
| %% | Literal percent sign | none |

C23 note: %b and %B are standardized by C23, but compiler and C library support may lag. Check the Ubuntu/Glibc version used by the program before relying on them.

## printf Length Modifiers

| Length | printf meaning |
|---|---|
| hh | d/i: signed char; u/o/x/X/b/B: unsigned char (argument is promoted to int/unsigned int) |
| h | d/i: short; u/o/x/X/b/B: unsigned short (argument is promoted) |
| l | d/i: long; u/o/x/X/b/B: unsigned long; c: wint_t; s: wchar_t * |
| ll | d/i: long long; u/o/x/X/b/B: unsigned long long |
| j | intmax_t or uintmax_t |
| z | Signed counterpart of size_t for d/i; size_t for unsigned conversions |
| t | ptrdiff_t for d/i; corresponding unsigned type for unsigned conversions |
| L | f/e/g/a families: long double |

## Common printf Examples

```
int count = -42;
unsigned int mask = 255;
double pi = 3.141592653589793;
long double value = 1.0L / 3.0L;

printf("%d\n", count);
printf("%#x\n", mask);
printf("%.3f\n", pi);
printf("%12.5e\n", pi);
printf("%La\n", value);
printf("%p\n", (void *)&count);
```

## Integer Output Quick Matrix

| C type | printf format |
|---|---|
| signed char | %hhd |

| | |
|---|---|
| unsigned char | %hhu |
| short | %hd |
| unsigned short | %hu |
| int | %d or %i |
| unsigned int | %u |
| long | %ld |
| unsigned long | %lu |
| long long | %lld |
| unsigned long long | %llu |
| intmax_t | %jd |
| uintmax_t | %ju |
| size_t | %zu |
| ptrdiff_t | %td |

### Floating Output Quick Matrix

| C type | printf format |
|---|---|
| float expression | %f, %e, %g, or %a (promoted to double) |
| double | %f, %e, %g, or %a |
| long double | %Lf, %Le, %Lg, or %La |

## 3. scanf Family

Functions include scanf, fscanf, sscanf, vscanf, vfscanf, and vsscanf. They return the number of input items successfully assigned. EOF is returned when an input failure occurs before the first assignment.

### Whitespace and Literal Characters

Whitespace in a scanf format string consumes zero or more whitespace characters from the input. Most conversions skip leading whitespace automatically. The exceptions are %c, %[ and %n. A literal non-whitespace character in the format must match the next input character exactly.

```
scanf("%d %lf %c", &age, &gpa, &grade);
```

For the format above, input such as 25 3.75A is valid. Numeric conversion stops when the next character cannot belong to the number, and the space before %c then skips any whitespace before reading the character.

### scanf Options

| Option | Meaning |
|---|---|
| * | Assignment suppression. Input is consumed but not stored; no pointer argument is supplied. |
| width | Maximum number of input characters consumed for that conversion. For %s and %[ reserve room for the terminating null character. |
| length | Changes the type of pointer that scanf expects. |

### scanf Conversion Specifiers

| Specifier | Input conversion | Expected pointer |
|---|---|---|
| %d | Signed decimal integer | int * |

| | | |
|---|---|---|
| %i | Signed integer; base detected from prefix (decimal, octal, hexadecimal, and C23 binary where supported) | int * |
| %u | Unsigned decimal integer | unsigned int * |
| %o | Unsigned octal integer | unsigned int * |
| %x, %X | Unsigned hexadecimal integer | unsigned int * |
| %b, %B | Unsigned binary integer (C23) | unsigned int * |
| %f, %e, %g, %a | Floating value | float * |
| %c | Character(s); does not skip leading whitespace; no null terminator | char * |
| %s | Non-whitespace character sequence; appends null terminator | char * |
| %[...] | Scanset; reads matching characters; appends null terminator | char * |
| %p | Pointer value in the implementation format used by printf %p | void ** |
| %n | Store characters consumed so far; consumes no input | pointer matching length modifier |
| %% | Match a literal percent sign | none |

## scanf Length Modifiers

| Length | scanf meaning |
|---|---|
| hh | d/i: signed char *; u/o/x/X/b/B: unsigned char * |
| h | d/i: short *; u/o/x/X/b/B: unsigned short * |
| l | d/i: long *; unsigned conversions: unsigned long *; floating: double *; c/s/[: wide-character destinations |
| ll | d/i: long long *; unsigned conversions: unsigned long long * |
| j | intmax_t * or uintmax_t * |
| z | Signed counterpart of size_t * for d/i; size_t * for unsigned conversions |
| t | ptrdiff_t * for d/i; corresponding unsigned pointer for unsigned conversions |
| L | floating conversions: long double * |

## Scalar Input Quick Matrix

| Variable type | scanf format | Argument type |
|---|---|---|
| signed char | %hhd | signed char * |
| unsigned char | %hhu | unsigned char * |
| short | %hd | short * |
| unsigned short | %hu | unsigned short * |
| int | %d | int * |
| unsigned int | %u | unsigned int * |
| long | %ld | long * |
| unsigned long | %lu | unsigned long * |
| long long | %lld | long long * |

| | | |
|---|---|---|
| unsigned long long | `%llu` | `unsigned long long *` |
| float | `%f` | `float *` |
| double | `%lf` | `double *` |
| long double | `%Lf` | `long double *` |
| char | `%c` | `char *` |

### Why scanf Uses Addresses

scanf must modify a variable, so it receives the variable's address. The address-of operator supplies that location.

```
int age;
double gpa;
char grade;

printf("Enter age, GPA, and grade: ");
if(scanf("%d %lf %c", &age, &gpa, &grade) == 3)
{
    printf("Stored: %d %.2f %c\n", age, gpa, grade);
}
```

### Field Width for Safer Text Input

Although this guide focuses on all format features, remember that %s and %[ require a destination array. Always set a field width that leaves room for the terminating null character.

```
char word[20];
scanf("%19s", word);
```

### Scansets

| Pattern | Meaning |
|---|---|
| `%[abc]` | Read only a, b, or c. |
| `%[^\n]` | Read everything except a newline. |
| `%[0-9]` | Common implementation behavior for a digit range; ranges are implementation-defined by the standard. |
| `%[]a]` | Place ] first in the set to include it as a matching character. |

## 4. printf vs scanf: Critical Differences

| Topic | printf | scanf |
|---|---|---|
| float | `printf uses %f because float arguments are promoted to double.` | `scanf uses %f and expects float *.` |
| double | `printf uses %f.` | `scanf uses %lf and expects double *.` |
| long double | `printf uses %Lf.` | `scanf uses %Lf.` |
| Arguments | Values are passed. | Addresses/pointers are passed. |
| Width | Minimum output field width. | Maximum input characters consumed. |
| Whitespace | `Printed literally.` | Consumes zero or more input whitespace characters. |
| Return value | Characters written, or negative on error. | Successful assignments, or EOF before any assignment. |

## 5. Format Macros for Fixed-Width Integers

For types from <stdint.h>, use the macros from <inttypes.h> when exact portability matters. Concatenate the macro with adjacent string literals.

```
#include <inttypes.h>
#include <stdint.h>

int64_t value = INT64_C(123456789);
printf("Value: %" PRId64 "\n", value);
scanf("%" SCNd64, &value);
```

| Macro pair | Purpose |
|---|---|
| PRId8 / SCNd8 | signed decimal int8_t |
| PRIu16 / SCNu16 | unsigned decimal uint16_t |
| PRIx32 / SCNx32 | hexadecimal uint32_t |
| PRId64 / SCNd64 | signed decimal int64_t |
| PRIdMAX / SCNdMAX | `intmax_t` |
| PRIuPTR / SCNuPTR | uintptr_t |

## 6. Common Errors and Undefined Behavior

| Problem | Why it matters |
|---|---|
| Mismatched specifier and argument type | `printf("%d", someDouble)` or `scanf("%f", &someDouble)` where the destination is double. The behavior is undefined. |
| Missing & with scanf | `scanf("%d", age)` passes an integer value as though it were an address. |
| `Using & with an array name for %s` | An array name already converts to a pointer to its first element in this context. |
| `Unbounded %s or %[ input` | May write beyond the destination array. |
| `User-controlled printf format` | `printf(userText)` can expose memory or use %n. Prefer `printf("%s", userText)`. |
| Ignoring scanf return value | Variables may remain unchanged or uninitialized when conversion fails. |
| Trailing whitespace in scanf format | `Formats such as "%d\n" may wait for a later non-whitespace character.` |
| Assuming width truncates printf output | Width is a minimum, not a maximum. |

## 7. Compact Cheat Sheet

| Data | printf | scanf |
|---|---|---|
| Signed integers | `%d / %i` | `%d / %i` |
| Unsigned decimal | %u | %u |
| Octal | %o | %o |
| Hexadecimal | `%x / %X` | `%x / %X` |
| Binary (C23) | `%b / %B` | `%b / %B` |
| float | `%f` | `%f` |
| double | `%f` | `%lf` |
| long double | `%Lf` | `%Lf` |

| | | |
|---|---|---|
| Character | %c | %c |
| String | %s | %Ns, where N < array size |
| Pointer | %p with (void *)pointer | %p with void ** destination |
| Literal percent | %% | %% |

## 8. Complete Example

```
#include <stdio.h>

int main(void)
{
    int age;
    unsigned int count;
    double gpa;
    char grade;

    printf("Enter age, count, GPA, and grade: ");

    if(scanf("%d %u %lf %c", &age, &count, &gpa, &grade) != 4)
    {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }

    printf("Age:   %d\n", age);
    printf("Count: %u (0x%X)\n", count, count);
    printf("GPA:   %.2f\n", gpa);
    printf("Grade: %c\n", grade);

    return 0;
}
```

## Reference Notes

This guide follows the ISO C formatted input/output model and uses Ubuntu/GCC conventions for examples. Implementation-defined details, library-version support, locale behavior, wide-character conversions, and C23 rollout can vary. Compile with strong warnings and test the exact target environment.

Recommended compile command:

```
gcc -std=c17 -Wall -Wextra -Wpedantic -Wformat=2 program.c -o program
```

For C23 experiments, use the compiler mode supported by the installed GCC release, commonly -std=c23 or -std=c2x.