NAME

printf — format data

SYNOPSIS

printf *format* [*data*]...

DESCRIPTION

Writes *format* to the standard output stream, interpreting escape sequences and *data* conversions.

If there are any **%** conversion specifiers, this is repeated until all *data* are exhausted.

Escapes

\a, **\b**, **\t**, **\n**, **\v**, **\f**, **\r**, **\e**	The bell, backspace, tab, line feed, vertical tab, form feed, carriage return and escape characters, respectively.
\', ****	Just **'** and ****.
\c	If inside **%b**: exit instantly.
****_O_, ****_OO_, ****_OOO_	Byte corresponding to octal value _O_ (_OO_, _OOO_).
\x_X_, **\x**_XX_	Byte corresponding to hexadecimal value _X_ (_XX_).
\u_XXXX_, **\U**_XXXXXXXX_	A representation of the Unicode character corresponding to the hexadecimal value _XXXX_ (_XXXXXXXX_) in the current locale. If the character is out of range (**0** or greater than **0x0010'FFFF**) or the conversion failed, "**\u**_XXXX_" if <= **0xFFFF** ("**\U**_XXXXXXXX_" otherwise) is output instead.
All others	passed through.

Conversions

For an in-depth description of conversion specifiers, see printf(3). The format used is identical, except **%b** and **%q** are added, and size specifiers (like **ll**, **h**, **L** for *long long*, *short*, *long double*) have no effect: all integers are 64-bit [*unsigned*] *long long*s, and all floating-point numbers are 128-bit *long double*s.

If there are more conversion specifiers than *data*, **0** is used for numeric conversions (**%diouxXeEfFgGaAcC**) and the null string for string conversions (**%sSbq**).

Partial conversions yield a diagnostic, but processing continues. Numbers can also be specified as **'**_C_ or **"**_C_, in which case they're equal to the value of the character in the current locale following the **'/"** (the next byte if invalid, or **0** if there are none).

Numbered conversions are specified by starting a conversion with **%**_nth_**$** instead of just **%**: the *data* used will be the *nth* argument. In this mode, repetitions happen from after the highest *nth* each iteration. These may not be mixed with unnumbered conversions.

Variable width and precision (**%*.*d**, **%**_3_**$***_1_**$.***_2_**$d**) are supported, and those arguments are intepreted as *int*s.

%% A literal **%**; no *data*.

%d, **%i** Signed decimal integer.
%u Unsigned decimal integer.
%o Unsigned octal integer.
%x, **%X** Unsigned hexadecimal integer. With **%X** – upper-case letters.

%e, **%E** Floating-point number in exponent (1.234**e**±56) format. With **%E** – capital **E**, **NAN**, &c.
%f, **%F** Floating-point number decimal (123**.**456) format rounded to precision (default: **6**). With **%F** – capital **NAN**, **INF**, &c.
%g, **%G** Equivalent to **%f** (**%F**) for floating-point numbers if ≥ *0.0001* and < *10^precision* (default: **6**), otherwise **%e** (**%E**).
%a, **%A** Floating-point number in hexadecimal exponent (**0x**a**.**bcde±f) format. With **%A** – capital letters, **X**, **NAN**, &c.

%c, **%C** First byte of *data*, NUL byte if empty.
%s, **%S** *data*

%b *data* with **** escapes (and **\c**) intepreted, but octal *****O*[**O*[**O*]] escapes may also be prefixed with a **0** (like **\0***O*[**O*[**O*]]). If a precision is specified, limit output to that many bytes.

%q *data* in a format that can be used to fully recover it as a single token in a sh(1)-style shell — printable characters are wrapped in **'**, others as octal **$'** escapes, except for **\a**, **\b**, **\t**, **\n**, **\v**, **\f**, **\r**, and **\e**. If a precision is specified, format that many bytes.

EXIT STATUS

1 if no digits were specified for an **\x** escape, not enough digits for **\u** and **\U** escapes or no conversion, an invalid, or an unknown one was specified after a **%**; these conditions also immediately abort processing. Additionally, **1** is returned but processing continues if a non-**'**/" number had trailing data or parsing failed altogether,

EXAMPLES

Assuming a default UTF-8 locale:

```
$ printf '%02X;' 12 012 0x12 \"A  # no newline
0C;0A;12;41;
$ printf '%-7s: %gkg\t$%.2f\n' Bananas 3.5 4.51 Kiwis 2 3.19 Bread 20.21
Bananas: 3.5kg  $4.51
Kiwis  : 2kg    $3.19
Bread  : 20.21kg        $0.00

$ printf '\44\x9\U0001F629%0*d\n' 3 \"Q
$       😩081
$ LC_ALL=C printf '\44\x9\U0001F629%0*d\n' 3 \"Q
$       \U0001F629081

$ printf '%q\n' "$(printf '\44\x9\U0001F629%0*d' 3 \"Q)"
'$'$'\t''😩081'
$ LC_ALL=C printf '%q\n' "$(printf '\44\x9\U0001F629%0*d' 3 \"Q)"
'$'$'\t\360\237\230\251''081'

$ printf '%4$*5$s %3$*5$s %2$*5$s %1$*5$s\n' abcd abc ab a 5
   a   ab  abc abcd
```

SEE ALSO

printf(3), strtold(3), strtoull(3)

STANDARDS

Conforms to IEEE Std 1003.1-2024 ("POSIX.1"), except that floating-point numbers are parsed with **strtold**() rather than **strtod**() and are allowed a **'**/" prefix . POSIX specifies only the following escapes in the *format*: ****, **\a**, **\b**, **\f**, **\n**, **\r**, **\t**, **\v**, and *****O*[**O*[**O*]]; and in **%b**: ****, **\a**, **\b**, **\f**, **\n**, **\r**, **\t**, **\v**, **\0***O*[**O*[**O*]], and **\c**. The standard specifies floating-point (**%aAeEfFgG**) conversions as optional, and requires only **%diouxXcs** and **%b**. The behaviour of **\c** that was sliced out of **%b** with precision (for example "**%.2bX**" and "abcd\c") is left unspecified: *some* implementations produce "ab" (they peek the "**\c**" and this aborts processing), others, like this one, produce "abX", (**\c** wasn't seen and thus doesn't exist).

%CS are extensions, provided for exhaustion of IEEE Std 1003.1-2024 ("POSIX.1") against X/Open Systems Interfaces (XSI) printf(3); they're equivalent to **%lc** and **%ls**, and hence equivalent to **%c** and **%s**. Don't use them.

%q is an extension, originating from the GNU system, whose **printf** doesn't understand the precision argument to **%b** — this is a conformance bug, nor to **%q** — this implementation's is an extension.

Variable width and precision (**%*.*d**, **%*3*$*1*$.*2*$d**) are an extension, available universally. The standard recommends implementing it, and using variable substitution into *format* instead.

The behaviour of mixing numbered and serial conversions is unspecified: this implementation refuses the *format* outright; some other implementations start consuming *data*, starting at *some* position.

If a **%c** conversion gets an empty string it may either produce a NUL byte or nothing at all. The former is near-universal.

Beyond what's specified by the standard, most systems support a wild array of **** escapes and conversions; be wary.

HISTORY

Created by X/Open Portability Guide Issue 4 ("XPG4") to provide a portable way to mimic AT&T System V Release 3 UNIX **echo** with **%b**, in contrast to the incompatible Version 7 AT&T UNIX **echo**, only supporting first-argument **-n**, cf. echo(1).

IEEE Std 1003.1-2024 ("POSIX.1") adds numbered **Conversions**. The maximum *nth* is the same as for printf(3) (NL_ARGMAX), but most implementations do not have a limit.