## NAME

printf − Print output based off of the format string and proceeding arguments.

## SYNOPSIS

**printf** [**−−help**] [**−−version**] [*FORMAT*] [*ARGUMENT*]

## DESCRIPTION

Print output based off of the format string and proceeding arguments.

## OPTIONS

**−−help** Print help information

**−−version**
Print version information

[*FORMAT*]

[*ARGUMENT*]

## EXTRA

basic anonymous string templating:

prints format string at least once, repeating as long as there are remaining arguments output prints escaped literals in the format string as character literals output replaces anonymous fields with the next unused argument, formatted according to the field.

Prints the , replacing escaped character sequences with character literals and substitution field sequences with passed arguments

literally, with the exception of the below escaped character sequences, and the substitution sequences described further down.

### ESCAPE SEQUENCES

The following escape sequences, organized here in alphabetical order, will print the corresponding character literal:

− \" double quote

− \\ backslash

− \\a alert (BEL)

− \\b backspace

− \\c End−of−Input

− \\e escape

− \\f form feed

− \\n new line

− \\r carriage return

− \\t horizontal tab

– \\v vertical tab

– \\NNN byte with value expressed in octal value NNN (1 to 3 digits)
values greater than 256 will be treated

– \\xHH byte with value expressed in hexadecimal value NN (1 to 2 digits)

– \\uHHHH Unicode (IEC 10646) character with value expressed in hexadecimal value HHHH (4 digits)

– \\uHHHH Unicode character with value expressed in hexadecimal value HHHH (8 digits)

– %% a single %

### SUBSTITUTIONS

#### SUBSTITUTION QUICK REFERENCE

Fields

– %s: string – %b: string parsed for literals second parameter is max length

– %c: char no second parameter

– %i or %d: 64–bit integer – %u: 64 bit unsigned integer – %x or %X: 64–bit unsigned integer as hex – %o: 64–bit unsigned integer as octal
second parameter is min–width, integer
output below that width is padded with leading zeroes

– %q: ARGUMENT is printed in a format that can be reused as shell input, escaping non–printable characters with the proposed POSIX $" syntax.

– %f or %F: decimal floating point value – %e or %E: scientific notation floating point value – %g or %G: shorter of specially interpreted decimal or SciNote floating point value.
second parameter is
–max places after decimal point for floating point output
–max number of significant digits for scientific notation output

parameterizing fields

examples:

printf '%4.3i' 7

It has a first parameter of 4 and a second parameter of 3 and will result in ' 007'

printf '%.1s' abcde

It has no first parameter and a second parameter of 1 and will result in 'a'

printf '%4c' q

It has a first parameter of 4 and no second parameter and will result in ' q'

The first parameter of a field is the minimum width to pad the output to if the output is less than this

absolute value of this width, it will be padded with leading spaces, or, if the argument is negative, with trailing spaces. the default is zero.

The second parameter of a field is particular to the output field type. defaults can be found in the full substitution help below

special prefixes to numeric arguments

– 0: (e.g. 010) interpret argument as octal (integer output fields only) – 0x: (e.g. 0xABC) interpret argument as hex (numeric output fields only) – \': (e.g. \'a) interpret argument as a character constant

#### HOW TO USE SUBSTITUTIONS

Substitutions are used to pass additional argument(s) into the FORMAT string, to be formatted a particular way. E.g.

printf 'the letter %X comes before the letter %X' 10 11

will print

the letter A comes before the letter B

because the substitution field %X means 'take an integer argument and write it as a hexadecimal number'

Passing more arguments than are in the format string will cause the format string to be repeated for the remaining substitutions

printf 'it is %i F in %s \n' 22 Portland 25 Boston 27 New York

will print

it is 22 F in Portland it is 25 F in Boston it is 27 F in Boston

If a format string is printed but there are less arguments remaining than there are substitution fields, substitution fields without an argument will default to empty strings, or for numeric fields the value 0

#### AVAILABLE SUBSTITUTIONS

This program, like GNU coreutils printf, interprets a modified subset of the POSIX C printf spec, a quick reference to substitutions is below.

#### STRING SUBSTITUTIONS

All string fields have a 'max width' parameter %.3s means 'print no more than three characters of the original input'

– %s: string

– %b: escaped string – the string will be checked for any escaped literals from
the escaped literal list above, and translate them to literal characters.
e.g. \\n will be transformed into a newline character.
One special rule about %b mode is that octal literals are interpreted differently
In arguments passed by %b, pass octal–interpreted literals must be in the form of \0NNN
instead of \\NNN. (Although, for legacy reasons, octal literals in the form of \\NNN will

still be interpreted and not throw a warning, you will have problems if you use this for a literal whose code begins with zero, as it will be viewed as in \0NNN form.)

– %q: escaped string – the string in a format that can be reused as input by most shells.
Non−printable characters are escaped with the POSIX proposed '$"' syntax,
and shell meta−characters are quoted appropriately.
This is an equivalent format to ls −−quoting=shell−escape output.

#### CHAR SUBSTITUTIONS

The character field does not have a secondary parameter.

– %c: a single character

#### INTEGER SUBSTITUTIONS

All integer fields have a 'pad with zero' parameter %.4i means an integer which if it is less than 4 digits in length, is padded with leading zeros until it is 4 digits in length.

– %d or %i: 64−bit integer

– %u: 64−bit unsigned integer

– %x or %X: 64−bit unsigned integer printed in Hexadecimal (base 16)
%X instead of %x means to use uppercase letters for 'a' through 'f'

– %o: 64−bit unsigned integer printed in octal (base 8)

#### FLOATING POINT SUBSTITUTIONS

All floating point fields have a 'max decimal places / max significant digits' parameter %.10f means a decimal floating point with 7 decimal places past 0 %.10e means a scientific notation number with 10 significant digits %.10g means the same behavior for decimal and Sci. Note, respectively, and provides the shortest of each's output.

Like with GNU coreutils, the value after the decimal point is these outputs is parsed as a double first before being rendered to text. For both implementations do not expect meaningful precision past the 18th decimal place. When using a number of decimal places that is 18 or higher, you can expect variation in output between GNU coreutils printf and this printf at the 18th decimal place of +/− 1

– %f: floating point value presented in decimal, truncated and displayed to 6 decimal places by default. There is not past−double behavior parity with Coreutils printf, values are not estimated or adjusted beyond input values.

– %e or %E: floating point value presented in scientific notation
7 significant digits by default
%E means use to use uppercase E for the mantissa.

– %g or %G: floating point value presented in the shortest of decimal and scientific notation
behaves differently from %f and %E, please see posix printf spec for full details,
some examples of different behavior:
Sci Note has 6 significant digits by default
Trailing zeroes are removed
Instead of being truncated, digit after last is rounded

Like other behavior in this utility, the design choices of floating point behavior in this utility is selected to reproduce in exact the behavior of GNU coreutils' printf from an inputs and outputs standpoint.

### USING PARAMETERS

Most substitution fields can be parameterized using up to 2 numbers that can be passed to the field, between the % sign and the field letter.

The 1st parameter always indicates the minimum width of output, it is useful for creating columnar output. Any output that would be less than this minimum width is padded with leading spaces The 2nd parameter is proceeded by a dot. You do not have to use parameters

### SPECIAL FORMS OF INPUT

For numeric input, the following additional forms of input are accepted besides decimal:

Octal (only with integer): if the argument begins with a 0 the proceeding characters will be interpreted as octal (base 8) for integer fields

Hexadecimal: if the argument begins with 0x the proceeding characters will be interpreted will be interpreted as hex (base 16) for any numeric fields for float fields, hexadecimal input results in a precision limit (in converting input past the decimal point) of 10^−15

Character Constant: if the argument begins with a single quote character, the first byte of the next character will be interpreted as an 8−bit unsigned integer. If there are additional bytes, they will throw an error (unless the environment variable POSIXLY_CORRECT is set)

## VERSION

v(uutils coreutils) 0.8.0