Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'printf.1p' command

$ man printf.1p

PRINTF(1P) POSIX Programmer's Manual PRINTF(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

printf ? write formatted output

SYNOPSIS

printf format [argument...]

DESCRIPTION

The printf utility shall write formatted operands to the standard out? put. The argument operands shall be formatted under control of the for? mat operand.

OPTIONS

None.

OPERANDS

The following operands shall be supported:

format A string describing the format to use to write the remaining operands. See the EXTENDED DESCRIPTION section.

argument The strings to be written to standard output, under the con? trol of format. See the EXTENDED DESCRIPTION section.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of printf:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er? ror.

LC_NUMERIC

Determine the locale for numeric formatting. It shall affect the format of numbers written using the e, E, f, g, and G conversion specifier characters (if supported).

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

See the EXTENDED DESCRIPTION section.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

The format operand shall be used as the format string described in the Base Definitions volume of POSIX.1?2017, Chapter 5, File Format Nota? tion with the following exceptions:

1. A <space> in the format string, in any context other than a flag of a conversion specification, shall be treated as an ordinary charac? ter that is copied to the output.
2. A '' character in the format string shall be treated as a '' char? acter, not as a <space>.
3. In addition to the escape sequences shown in the Base Definitions volume of POSIX.1?2017, Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'), "\ddd", where ddd is a one, two, or three-digit octal number, shall be written as a byte with the numeric value specified by the octal number.
4. The implementation shall not precede or follow output from the d or u conversion specifiers with <blank> characters not specified by the format operand.
5. The implementation shall not precede output from the o conversion specifier with zeros not specified by the format operand.
6. The a, A, e, E, f, F, g, and G conversion specifiers need not be supported.
7. An additional conversion specifier character, b, shall be supported as follows. The argument shall be taken to be a string that can contain <backslash>-escape sequences. The following <backslash>-es? cape sequences shall be supported:
   -- The escape sequences listed in the Base Definitions volume of POSIX.1?2017, Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'), which shall be converted to the characters they represent
   -- "\0ddd", where ddd is a zero, one, two, or three-digit octal number that shall be converted to a byte with the numeric value specified by the octal number

-- '\c', which shall not be written and shall cause printf to ig?
nore any remaining characters in the string operand containing
it, any remaining string operands, and any additional charac?
ters in the format operand

The interpretation of a <backslash> followed by any other sequence of characters is unspecified.

Bytes from the converted string shall be written until the end of the string or the number of bytes indicated by the precision speci? fication is reached. If the precision is omitted, it shall be taken to be infinite, so all bytes up to the end of the converted string shall be written.

8. For each conversion specification that consumes an argument, the next argument operand shall be evaluated and converted to the ap? propriate type for the conversion as specified below.

9. The format operand shall be reused as often as necessary to satisfy the argument operands. Any extra b, c, or s conversion specifiers shall be evaluated as if a null string argument were supplied; other extra conversion specifications shall be evaluated as if a zero argument were supplied. If the format operand contains no con? version specifications and argument operands are present, the re? sults are unspecified.

10. If a character sequence in the format operand begins with a '%' character, but does not form a valid conversion specification, the behavior is unspecified.

11. The argument to the c conversion specifier can be a string contain? ing zero or more bytes. If it contains one or more bytes, the first byte shall be written and any additional bytes shall be ignored. If the argument is an empty string, it is unspecified whether nothing is written or a null byte is written.

The argument operands shall be treated as strings if the corresponding conversion specifier is b, c, or s, and shall be evaluated as if by the strtod() function if the corresponding conversion specifier is a, A, e, E, f, F, g, or G. Otherwise, they shall be evaluated as unsuffixed C

integer constants, as described by the ISO C standard, with the follow?
ing extensions:

* A leading <plus-sign> or <hyphen-minus> shall be allowed.
* If the leading character is a single-quote or double-quote, the value shall be the numeric value in the underlying codeset of the character following the single-quote or double-quote.
* Suffixed integer constants may be allowed.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a di? agnostic message shall be written to standard error and the utility shall not exit with a zero exit status, but shall continue processing any remaining operands and shall write the value accumulated at the time the error was detected to standard output.

It shall not be considered an error if an argument operand is not com? pletely used for a b, c, or s conversion.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

The floating-point formatting conversion specifications of printf() are not required because all arithmetic in the shell is integer arithmetic.

The awk utility performs floating-point calculations and provides its own printf function. The bc utility can perform arbitrary-precision floating-point arithmetic, but does not provide extensive formatting capabilities. (This printf utility cannot really be used to format bc output; it does not support arbitrary precision.) Implementations are encouraged to support the floating-point conversions as an extension.

Note that this printf utility, like the printf() function defined in the System Interfaces volume of POSIX.1?2017 on which it is based,

makes no special provision for dealing with multi-byte characters when using the %c conversion specification or when a precision is specified in a %b or %s conversion specification. Applications should be ex? tremely cautious using either of these features when there are multi-byte characters in the character set.

No provision is made in this volume of POSIX.1?2017 which allows field widths and precisions to be specified as '*' since the '*' can be re? placed directly in the format operand using shell variable substitu? tion. Implementations can also provide this feature as an extension if they so choose.

Hexadecimal character constants as defined in the ISO C standard are not recognized in the format operand because there is no consistent way to detect the end of the constant. Octal character constants are lim? ited to, at most, three octal digits, but hexadecimal character con? stants are only terminated by a non-hex-digit character. In the ISO C standard, the "##" concatenation operator can be used to terminate a constant and follow it with a hexadecimal character to be written. In the shell, concatenation occurs before the printf utility has a chance to parse the end of the hexadecimal constant.

The %b conversion specification is not part of the ISO C standard; it has been added here as a portable way to process <backslash>-escapes expanded in string operands as provided by the echo utility. See also the APPLICATION USAGE section of echo for ways to use printf as a re? placement for all of the traditional versions of the echo utility.

If an argument cannot be parsed correctly for the corresponding conver? sion specification, the printf utility is required to report an error. Thus, overflow and extraneous characters at the end of an argument be? ing used for a numeric conversion shall be reported as errors.

EXAMPLES

To alert the user and then print and read a series of prompts:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone number: "
```

```
    read phone
```

To read out a list of right and wrong answers from a file, calculate the percentage correctly, and print them out. The numbers are right-justified and separated by a single <tab>. The percentage is written to one decimal place of accuracy:

```
    while read right wrong ; do
        percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
        printf "%2d right\t%2d wrong\t(%s%%)\n" \
            $right $wrong $percent
    done < database_file
```

The command:

```
    printf "%5d%4d\n" 1 21 321 4321 54321
```

produces:

```
        1  21
      3214321
    54321   0
```

Note that the format operand is used three times to print all of the given strings and that a '0' was supplied by printf to satisfy the last %4d conversion specification.

The printf utility is required to notify the user when conversion er? rors are detected while producing numeric output; thus, the following results would be expected on an implementation with 32-bit twos-comple? ment integers when %d is specified as the format operand:

```
????????????????????????????????????????????????????????????????????????????
?           ?  Standard   ?                                    ?
? Argument   ?   Output    ?             Diagnostic Output             ?
????????????????????????????????????????????????????????????????????????????
?5a          ? 5           ? printf: "5a" not completely converted     ?
?9999999999  ? 2147483647  ? printf: "9999999999" arithmetic overflow  ?
?-9999999999 ? -2147483648 ? printf: "-9999999999" arithmetic overflow ?
?ABC         ? 0           ? printf: "ABC" expected numeric value      ?
????????????????????????????????????????????????????????????????????????????
```

The diagnostic message format is not specified, but these examples con?

vey the type of information that should be reported. Note that the value shown on standard output is what would be expected as the return value from the strtol() function as defined in the System Interfaces volume of POSIX.1?2017. A similar correspondence exists between %u and strtoul() and %e, %f, and %g (if the implementation supports floating-point conversions) and strtod().

In a locale using the ISO/IEC 646:1991 standard as the underlying code? set, the command:

```
printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"
```

produces:

3 Numeric value of constant 3

3 Numeric value of constant 3

-3 Numeric value of constant -3

51 Numeric value of the character '3' in the ISO/IEC 646:1991 stan? dard codeset

43 Numeric value of the character '+' in the ISO/IEC 646:1991 stan? dard codeset

45 Numeric value of the character '-' in the ISO/IEC 646:1991 stan? dard codeset

Note that in a locale with multi-byte characters, the value of a char? acter is intended to be the value of the equivalent of the wchar_t rep? resentation of the character as described in the System Interfaces vol? ume of POSIX.1?2017.

RATIONALE

The printf utility was added to provide functionality that has histori? cally been provided by echo. However, due to irreconcilable differ? ences in the various versions of echo extant, the version has few spe? cial features, leaving those to this new printf utility, which is based on one in the Ninth Edition system.

The EXTENDED DESCRIPTION section almost exactly matches the printf() function in the ISO C standard, although it is described in terms of the file format notation in the Base Definitions volume of POSIX.1?2017, Chapter 5, File Format Notation.

Earlier versions of this standard specified that arguments for all con? versions other than b, c, and s were evaluated in the same way (as C constants, but with stated exceptions). For implementations supporting the floating-point conversions it was not clear whether integer conver? sions need only accept integer constants and floating-point conversions need only accept floating-point constants, or whether both types of conversions should accept both types of constants. Also by not distin? guishing between them, the requirement relating to a leading single-quote or double-quote applied to floating-point conversions even though this provided no useful functionality to applications that was not al? ready available through the integer conversions. The current standard clarifies the situation by specifying that the arguments for floating-point conversions are evaluated as if by strtod(), and the arguments for integer conversions are evaluated as C integer constants, with the special treatment of leading single-quote and double-quote applying only to integer conversions.

FUTURE DIRECTIONS

None.

SEE ALSO

awk, bc, echo

The Base Definitions volume of POSIX.1?2017, Chapter 5, File Format No? tation, Chapter 8, Environment Variables

The System Interfaces volume of POSIX.1?2017, fprintf(), strtod()

COPYRIGHT

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .