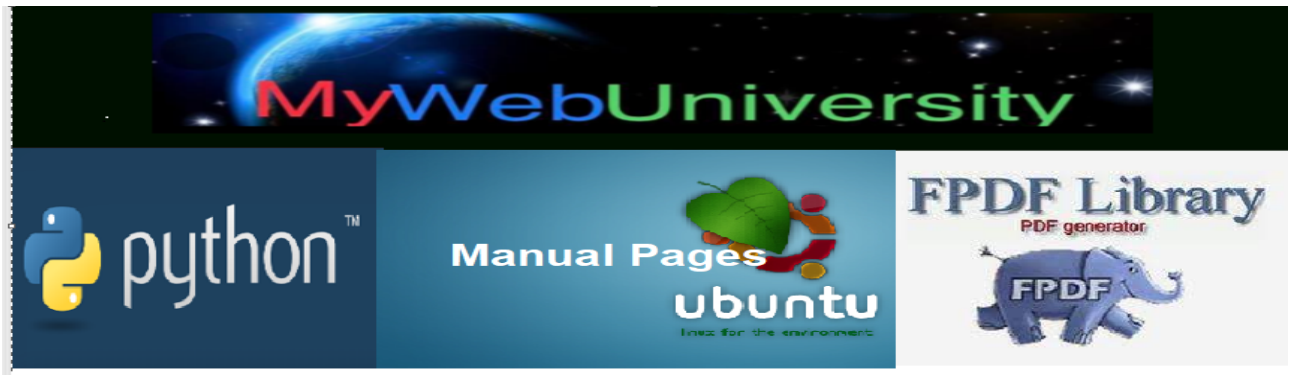

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'vsnprintf.3'

$ man vsnprintf.3

snprintf(3) Library Functions Manual snprintf(3)

NAME

snprintf, vsnprintf - string print formatted

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <stdio.h>

int snprintf(size_t size;
             char str[restrict size], size_t size,
             const char *restrict format, ...);
int vsnprintf(size_t size;
             char str[restrict size], size_t size,
             const char *restrict format, va_list ap);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

snprintf(), vsnprintf():

```
_XOPEN_SOURCE >= 500 || _ISOC99_SOURCE
    || /* glibc <= 2.19: */ _BSD_SOURCE
```

DESCRIPTION

These functions are similar to printf(3), except that they write to the character string str instead of a stream.

The functions snprintf() and vsnprintf() write at most size bytes (including the terminating null byte ('\0')) to str.

vsnprintf() is equivalent to snprintf(), except that it is called with a va_list instead of

a variable number of arguments. This function does not call the va_end macro. Because it invokes the va_arg macro, the value of ap is undefined after the call. See stdarg(3).

C99 and POSIX.1-2001 specify that the results are undefined if a call to snprintf() or vs? nprintf() would cause copying to take place between objects that overlap (e.g., if the tar? get string array and one of the supplied input arguments refer to the same buffer). See CAVEATS.

Format of the format string

See printf(3).

RETURN VALUE

Upon successful return, these functions return the number of bytes printed (excluding the null byte used to end output to strings).

The functions snprintf() and vsnprintf() do not write more than size bytes (including the terminating null byte ('\0')). If the output was truncated due to this limit, then the re? turn value is the number of characters (excluding the terminating null byte) which would have been written to the final string if enough space had been available. Thus, a return value of size or more means that the output was truncated. (See also below under CAVEATS.)

On error, a negative value is returned, and errno is set to indicate the error.

ERRORS

See printf(3).

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
snprintf(), vsnprintf()	Thread safety	MT-Safe locale

STANDARDS

C11, POSIX.1-2008.

HISTORY

SUSv2, C99, POSIX.1-2001.

Concerning the return value, SUSv2 and C99 contradict each other: when snprintf() is called with size=0 then SUSv2 stipulates an unspecified return value less than 1, while C99 allows str to be NULL in this case, and gives the return value (as always)

as the number of characters that would have been written in case the output string has been large enough. POSIX.1-2001 and later align their specification of snprintf() with C99.

CAVEATS

Some programs imprudently rely on code such as the following

```
snprintf(buf, countof(buf), "%s some further text", buf);
```

to append text to buf. However, the standards explicitly note that the results are unde? fined if source and destination buffers overlap when calling snprintf() and vsnprintf(). Depending on the version of gcc(1) used, and the compiler options employed, calls such as the above will not produce the expected results.

The glibc implementation of the functions snprintf() and vsnprintf() conforms to the C99 standard, that is, behaves as described above, since glibc 2.1. Until glibc 2.0.6, they would return -1 when the output was truncated.

BUGS

See printf(3).

EXAMPLES

To allocate a sufficiently large string and print into it (code correct for both glibc 2.0 and glibc 2.1):

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
char *
make_message(const char *fmt, ...)
{
    int n = 0;
    size_t size = 0;
    char *p = NULL;
    va_list ap;
    /* Determine required size.  */
    va_start(ap, fmt);
    n = vsnprintf(p, size, fmt, ap);
    va_end(ap);
    if (n < 0)
```

```
        return NULL;
    size = (size_t) n + 1;      /* One extra byte for '\0' */
    p = malloc(size);
    if (p == NULL)
        return NULL;
    va_start(ap, fmt);
    n = vsnprintf(p, size, fmt, ap);
    va_end(ap);
    if (n < 0) {
        free(p);
        return NULL;
    }
    return p;
}
```

If truncation occurs in glibc versions prior to glibc 2.0.6, this is treated as an error in? stead of being handled gracefully.

SEE ALSO

printf(1), asprintf(3), printf(3), puts(3), scanf(3), setlocale(3), strfromd(3), wcrtomb(3), wprintf(3), locale(5)