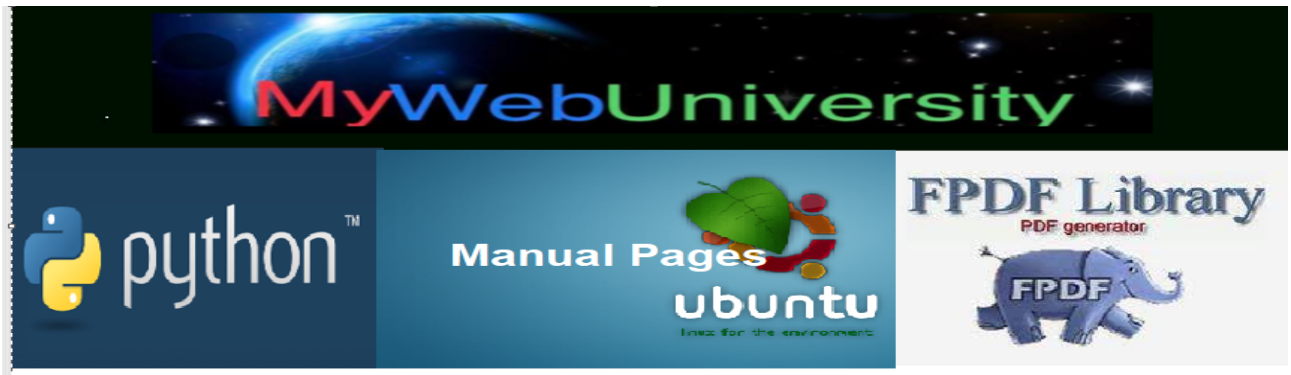

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man strncpy.3

stpncpy(3) Library Functions Manual stpncpy(3)

NAME

stpncpy, strncpy - fill a fixed-size buffer with non-null bytes from a string, padding with null bytes as needed

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <string.h>
char *strncpy(size_t dsize;
              char dst[restrict dsize], const char *restrict src,
              size_t dsize);
char *stpncpy(size_t dsize;
              char dst[restrict dsize], const char *restrict src,
              size_t dsize);
```

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

stpncpy():

Since glibc 2.10:

_POSIX_C_SOURCE >= 200809L

Before glibc 2.10:

_GNU_SOURCE

DESCRIPTION

These functions copy non-null bytes from the string pointed to by src into the array pointed to by dst. If the source has too few non-null bytes to fill the destination, the functions

pad the destination with trailing null bytes. If the destination buffer, limited by its size, isn't large enough to hold the copy, the resulting character sequence is truncated.

For the difference between the two functions, see RETURN VALUE.

An implementation of these functions might be:

```
char *
strncpy(char *restrict dst, const char *restrict src, size_t dsize)
{
    stpncpy(dst, src, dsize);
    return dst;
}
char *
stpncpy(char *restrict dst, const char *restrict src, size_t dsize)
{
    size_t  dlen;
    dlen = strnlen(src, dsize);
    return memset(mempcpy(dst, src, dlen), 0, dsize - dlen);
}
```

RETURN VALUE

strncpy()

returns dst.

stpncpy()

returns a pointer to one past the last non-null character written, that is, dest + strnlen(src, n).

ATTRIBUTES

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

Interface	Attribute	Value
stpncpy(), strncpy()	Thread safety	MT-Safe

STANDARDS

strncpy()

C11, POSIX.1-2008.

stpncpy()

POSIX.1-2008.

HISTORY

strncpy()

C89, POSIX.1-2001, SVr4, 4.3BSD.

stpncpy()

glibc 1.07. POSIX.1-2008.

CAVEATS

The name of these functions is confusing. These functions produce a null-padded character sequence, not a string (see string_copying(7)). For example:

```
strncpy(buf, "1", 5);       // { '1',   0,   0,   0,   0 }
strncpy(buf, "1234", 5);    // { '1', '2', '3', '4',   0 }
strncpy(buf, "12345", 5);   // { '1', '2', '3', '4', '5' }
strncpy(buf, "123456", 5);  // { '1', '2', '3', '4', '5' }
```

It's impossible to distinguish truncation by the result of the call, from a character se? quence that just fits the destination buffer; truncation should be detected by comparing the length of the input string with the size of the destination buffer.

If you're going to use this function in chained calls, it would be useful to develop a simi? lar function that accepts a pointer to the end (one after the last element) of the destina? tion buffer instead of its size.

EXAMPLES

```
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int
main(void)
{
    char    *p;
    char    buf1[20];
    char    buf2[20];
    size_t  len;
    if (sizeof(buf2) < strlen("Hello world!"))
```

```
            errx("strncpy: truncating character sequence");
        strncpy(buf2, "Hello world!", sizeof(buf2));
        len = strnlen(buf2, sizeof(buf2));
        printf("[len = %zu]: ", len);
        fwrite(buf2, 1, len, stdout);
        putchar('\n');
        if (sizeof(buf1) < strlen("Hello world!"))
            errx("stpncpy: truncating character sequence");
        p = stpncpy(buf1, "Hello world!", sizeof(buf1));
        len = p - buf1;
        printf("[len = %zu]: ", len);
        fwrite(buf1, 1, len, stdout);
        putchar('\n');
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

wcpncpy(3), string_copying(7)