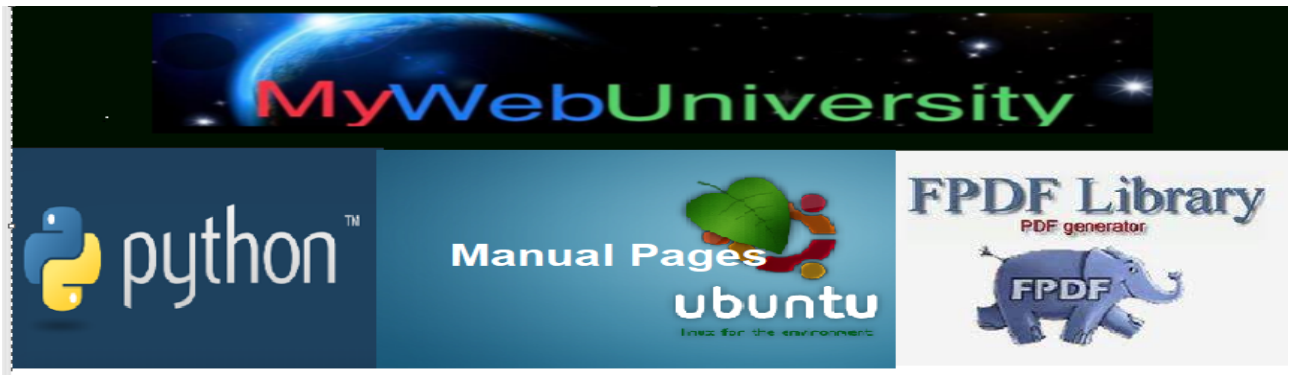

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man strcmp.3

strcmp(3) Library Functions Manual strcmp(3)

NAME

strcmp, strncmp - compare two strings

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

#include <string.h>

int strcmp(const char *s1, const char *s2);

int strncmp(const char s1[], const char s2[], size_t n);

DESCRIPTION

The strcmp() function compares the two strings s1 and s2. The locale is not taken into ac? count (for a locale-aware comparison, see strcoll(3)). The comparison is done using un? signed characters.

strcmp() returns an integer indicating the result of the comparison, as follows:

? 0, if the s1 and s2 are equal;

? a negative value if s1 is less than s2;

? a positive value if s1 is greater than s2.

The strncmp() function is similar, except it compares only the first (at most) n bytes of s1 and s2.

RETURN VALUE

The strcmp() and strncmp() functions return an integer less than, equal to, or greater than zero if s1 (or the first n bytes thereof) is found, respectively, to be less than, to match, or be greater than s2.

ATTRIBUTES

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

Interface	Attribute	Value
strcmp(), strncmp()	Thread safety	MT-Safe

VERSIONS

POSIX.1 specifies only that:

> The sign of a nonzero return value shall be determined by the sign of the difference between the values of the first pair of bytes (both interpreted as type unsigned char) that differ in the strings being compared.

In glibc, as in most other implementations, the return value is the arithmetic result of subtracting the last compared byte in s2 from the last compared byte in s1. (If the two characters are equal, this difference is 0.)

STANDARDS

C11, POSIX.1-2008.

HISTORY

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

EXAMPLES

The program below can be used to demonstrate the operation of strcmp() (when given two argu? ments) and strncmp() (when given three arguments). First, some examples using strcmp():

```
$ ./string_comp ABC ABC;
<str1> and <str2> are equal
$ ./string_comp ABC AB;      # 'C' is ASCII 67; 'C' - '\0' = 67
<str1> is greater than <str2> (67)
$ ./string_comp ABA ABZ;     # 'A' is ASCII 65; 'Z' is ASCII 90
<str1> is less than <str2> (-25)
$ ./string_comp ABJ ABC;
<str1> is greater than <str2> (7)
$ ./string_comp $'\201' A;   # 0201 - 0101 = 0100 (or 64 decimal)
<str1> is greater than <str2> (64)
```

The last example uses bash(1)-specific syntax to produce a string containing an 8-bit ASCII

code; the result demonstrates that the string comparison uses unsigned characters.

And then some examples using strncmp():

```
$ ./string_comp ABC AB 3;
<str1> is greater than <str2> (67)
$ ./string_comp ABC AB 2;
<str1> and <str2> are equal in the first 2 bytes
```

Program source

```
/* string_comp.c
   Licensed under GNU General Public License v2 or later.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int
main(int argc, char *argv[])
{
    int res;
    if (argc < 3) {
        fprintf(stderr, "Usage: %s <str1> <str2> [<n>]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if (argc == 3)
        res = strcmp(argv[1], argv[2]);
    else
        res = strncmp(argv[1], argv[2], atoi(argv[3]));
    if (res == 0) {
        printf("<str1> and <str2> are equal");
        if (argc > 3)
            printf(" in the first %d bytes\n", atoi(argv[3]));
        printf("\n");
    } else if (res < 0) {
        printf("<str1> is less than <str2> (%d)\n", res);
    } else {
```

```
            printf("<str1> is greater than <str2> (%d)\n", res);
        }
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

memcmp(3), strcasecmp(3), strcoll(3), string(3), strncasecmp(3), strverscmp(3), wcscmp(3), wcsncmp(3), ascii(7)