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 'strtok.3' command

$ man strtok.3

STRTOK(3) Linux Programmer's Manual STRTOK(3)

NAME

strtok, strtok_r - extract tokens from strings

SYNOPSIS

```
#include <string.h>

char *strtok(char *str, const char *delim);

char *strtok_r(char *str, const char *delim, char **saveptr);
```

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

strtok_r(): _POSIX_C_SOURCE

|| /* Glibc versions <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

The strtok() function breaks a string into a sequence of zero or more nonempty tokens. On the first call to strtok(), the string to be parsed should be specified in str. In each subsequent call that should parse the same string, str must be NULL.

The delim argument specifies a set of bytes that delimit the tokens in the parsed string. The caller may specify different strings in delim in successive calls that parse the same string.

Each call to strtok() returns a pointer to a null-terminated string containing the next token. This string does not include the delimiting byte. If no more tokens are found, strtok() returns NULL.

A sequence of calls to strtok() that operate on the same string main? tains a pointer that determines the point from which to start searching

for the next token. The first call to strtok() sets this pointer to point to the first byte of the string. The start of the next token is determined by scanning forward for the next nondelimiter byte in str. If such a byte is found, it is taken as the start of the next token.

If no such byte is found, then there are no more tokens, and strtok() returns NULL. (A string that is empty or that contains only delimiters will thus cause strtok() to return NULL on the first call.)

The end of each token is found by scanning forward until either the next delimiter byte is found or until the terminating null byte ('\0') is encountered. If a delimiter byte is found, it is overwritten with a null byte to terminate the current token, and strtok() saves a pointer to the following byte; that pointer will be used as the starting point when searching for the next token. In this case, strtok() returns a pointer to the start of the found token.

From the above description, it follows that a sequence of two or more contiguous delimiter bytes in the parsed string is considered to be a single delimiter, and that delimiter bytes at the start or end of the string are ignored. Put another way: the tokens returned by strtok() are always nonempty strings. Thus, for example, given the string "aaa;;bbb,", successive calls to strtok() that specify the delimiter string ";," would return the strings "aaa" and "bbb", and then a null pointer.

The strtok_r() function is a reentrant version of strtok(). The saveptr argument is a pointer to a char * variable that is used inter? nally by strtok_r() in order to maintain context between successive calls that parse the same string.

On the first call to strtok_r(), str should point to the string to be parsed, and the value of *saveptr is ignored (but see NOTES). In sub? sequent calls, str should be NULL, and saveptr (and the buffer that it points to) should be unchanged since the previous call.

Different strings may be parsed concurrently using sequences of calls to strtok_r() that specify different saveptr arguments.

RETURN VALUE

The strtok() and strtok_r() functions return a pointer to the next to?
ken, or NULL if there are no more tokens.

ATTRIBUTES

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

Interface	Attribute	Value
strtok()	Thread safety	MT-Unsafe race:strtok
strtok_r()	Thread safety	MT-Safe

CONFORMING TO

strtok()

POSIX.1-2001, POSIX.1-2008, C89, C99, SVr4, 4.3BSD.

strtok_r()

POSIX.1-2001, POSIX.1-2008.

NOTES

On some implementations, *saveptr is required to be NULL on the first
call to strtok_r() that is being used to parse str.

BUGS

Be cautious when using these functions. If you do use them, note that:

* These functions modify their first argument.

* These functions cannot be used on constant strings.

* The identity of the delimiting byte is lost.

* The strtok() function uses a static buffer while parsing, so it's not
thread safe. Use strtok_r() if this matters to you.

EXAMPLES

The program below uses nested loops that employ strtok_r() to break a
string into a two-level hierarchy of tokens. The first command-line
argument specifies the string to be parsed. The second argument speci?
fies the delimiter byte(s) to be used to separate that string into "ma?
jor" tokens. The third argument specifies the delimiter byte(s) to be

used to separate the "major" tokens into subtokens.

An example of the output produced by this program is the following:

```
$ ./a.out 'a/bbb///cc;xxx:yyy:' ':;' '/'
1: a/bbb///cc
        --> a
        --> bbb
        --> cc
2: xxx
        --> xxx
3: yyy
        --> yyy
```

Program source

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int
main(int argc, char *argv[])
{
    char *str1, *str2, *token, *subtoken;
    char *saveptr1, *saveptr2;
    if (argc != 4) {
        fprintf(stderr, "Usage: %s string delim subdelim\n",
                argv[0]);
        exit(EXIT_FAILURE);
    }
    for (int j = 1, str1 = argv[1]; ; j++, str1 = NULL) {
        token = strtok_r(str1, argv[2], &saveptr1);
        if (token == NULL)
            break;
        printf("%d: %s\n", j, token);
        for (str2 = token; ; str2 = NULL) {
            subtoken = strtok_r(str2, argv[3], &saveptr2);
            if (subtoken == NULL)
```

```
                break;
            printf(" --> %s\n", subtoken);
        }
    }
    exit(EXIT_SUCCESS);
}
```

Another example program using strtok() can be found in getad? drinfo_a(3).

SEE ALSO

index(3), memchr(3), rindex(3), strchr(3), string(3), strpbrk(3), strsep(3), strspn(3), strstr(3), wcstok(3)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.