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.3p' command

$ man strtok.3p

STRTOK(3P) POSIX Programmer's Manual STRTOK(3P)

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

strtok, strtok_r ? split string into tokens

SYNOPSIS

```
#include <string.h>
char *strtok(char *restrict s, const char *restrict sep);
char *strtok_r(char *restrict s, const char *restrict sep,
    char **restrict state);
```

DESCRIPTION

For strtok(): The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

A sequence of calls to strtok() breaks the string pointed to by s into a sequence of tokens, each of which is delimited by a byte from the string pointed to by sep. The first call in the sequence has s as its first argument, and is followed by calls with a null pointer as their first argument. The separator string pointed to by sep may be different

from call to call.

The first call in the sequence searches the string pointed to by s for the first byte that is not contained in the current separator string pointed to by sep. If no such byte is found, then there are no tokens in the string pointed to by s and strtok() shall return a null pointer. If such a byte is found, it is the start of the first token.

The strtok() function then searches from there for a byte that is con? tained in the current separator string. If no such byte is found, the current token extends to the end of the string pointed to by s, and subsequent searches for a token shall return a null pointer. If such a byte is found, it is overwritten by a NUL character, which terminates the current token. The strtok() function saves a pointer to the follow? ing byte, from which the next search for a token shall start.

Each subsequent call, with a null pointer as the value of the first ar? gument, starts searching from the saved pointer and behaves as de? scribed above.

The implementation shall behave as if no function defined in this vol? ume of POSIX.1?2017 calls strtok().

The strtok() function need not be thread-safe.

The strtok_r() function shall be equivalent to strtok(), except that strtok_r() shall be thread-safe and the argument state points to a user-provided pointer that allows strtok_r() to maintain state between calls which scan the same string. The application shall ensure that the pointer pointed to by state is unique for each string (s) being pro? cessed concurrently by strtok_r() calls. The application need not ini? tialize the pointer pointed to by state to any particular value. The implementation shall not update the pointer pointed to by state to point (directly or indirectly) to resources, other than within the string s, that need to be freed or released by the caller.

RETURN VALUE

Upon successful completion, strtok() shall return a pointer to the first byte of a token. Otherwise, if there is no token, strtok() shall return a null pointer.

The strtok_r() function shall return a pointer to the token found, or a null pointer when no token is found.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

Searching for Word Separators

The following example searches for tokens separated by <space> charac?ters.

```
#include <string.h>
...
char *token;
char line[] = "LINE TO BE SEPARATED";
char *search = " ";
/* Token will point to "LINE". */
token = strtok(line, search);
/* Token will point to "TO". */
token = strtok(NULL, search);
```

Find First two Fields in a Buffer

The following example uses strtok() to find two character strings (a key and data associated with that key) separated by any combination of <space>, <tab>, or <newline> characters at the start of the array of characters pointed to by buffer.

```
#include <string.h>
...
char    *buffer;
...
struct element {
    char *key;
    char *data;
} e;
...
// Load the buffer...
```

```
...
// Get the key and its data...
e.key = strtok(buffer, " \t\n");
e.data = strtok(NULL, " \t\n");
// Process the rest of the contents of the buffer...
...
```

APPLICATION USAGE

Note that if sep is the empty string, strtok() and strtok_r() return a pointer to the remainder of the string being tokenized.

The strtok_r() function is thread-safe and stores its state in a user-supplied buffer instead of possibly using a static data area that may be overwritten by an unrelated call from another thread.

RATIONALE

The strtok() function searches for a separator string within a larger string. It returns a pointer to the last substring between separator strings. This function uses static storage to keep track of the cur? rent string position between calls. The new function, strtok_r(), takes an additional argument, state, to keep track of the current position in the string.

FUTURE DIRECTIONS

None.

SEE ALSO

The Base Definitions volume of POSIX.1?2017, <string.h>

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 .