



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

\$ man getline.3

getline(3) Library Functions Manual getline(3)

NAME

getline, getdelim - delimited string input

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <stdio.h>
```

```
ssize_t getline(char **restrict lineptr, size_t *restrict n,  
                FILE *restrict stream);
```

```
ssize_t getdelim(char **restrict lineptr, size_t *restrict n,  
                 int delim, FILE *restrict stream);
```

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

getline(), getdelim():

Since glibc 2.10:

```
_POSIX_C_SOURCE >= 200809L
```

Before glibc 2.10:

```
_GNU_SOURCE
```

DESCRIPTION

getline() reads an entire line from stream, storing the address of the buffer containing the text into *lineptr. The buffer is null-terminated and includes the newline character, if one was found.

If *lineptr is set to NULL before the call, then getline() will allocate a buffer for storing the line. This buffer should be freed by the user program even if getline() failed.

`getdelim()` works like `getline()`, except that a line delimiter other than newline can be specified as the delimiter argument. As with `getline()`, a delimiter character is not added if one was not present in the input before end of file was reached.

If `*lineptr` was set to NULL before the call, then the buffer should be freed by the user program even on failure.

ENOMEM Allocation or reallocation of the line buffer failed.

[illegible]

```

#include <stdio.h>

#include <stdlib.h>

int
main(int argc, char *argv[])
{
    FILE *stream;

    char *line = NULL;

    size_t size = 0;

    ssize_t nread;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    stream = fopen(argv[1], "r");

    if (stream == NULL) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }

    while ((nread = getline(&line, &size, stream)) != -1) {
        printf("Retrieved line of length %zd:\n", nread);
        fwrite(line, nread, 1, stdout);
    }

    free(line);

    fclose(stream);

    exit(EXIT_SUCCESS);
}

```

SEE ALSO

[read\(2\)](#), [fgets\(3\)](#), [fopen\(3\)](#), [fread\(3\)](#), [scanf\(3\)](#)