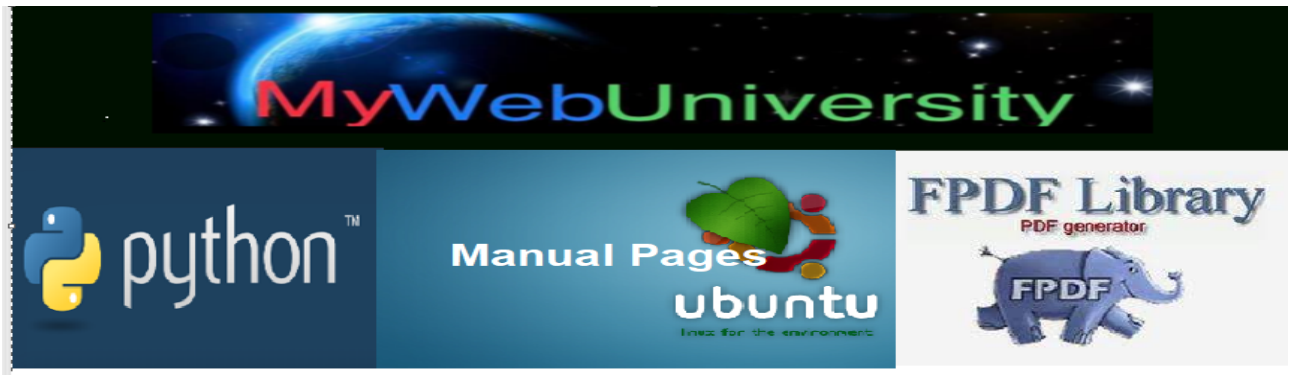

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man getdelim.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 stor? ing the line. This buffer should be freed by the user program even if getline() failed.

Alternatively, before calling getline(), *lineptr can contain a pointer to a malloc(3)-allo?
cated buffer *n bytes in size. If the buffer is not large enough to hold the line, get?
line() resizes it with realloc(3), updating *lineptr and *n as necessary.
In either case, on a successful call, *lineptr and *n will be updated to reflect the buffer
address and allocated size respectively.
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.

RETURN VALUE

On success, getline() and getdelim() return the number of characters read, including the de?
limiter character, but not including the terminating null byte ('\0'). This value can be
used to handle embedded null bytes in the line read.
At end of file, both functions return -1 with the file stream end-of-file indicator set. On
error, both functions return -1 with the file stream error indicator set, and errno is set
to indicate the error.
If *lineptr was set to NULL before the call, then the buffer should be freed by the user
program even on failure.

ERRORS

EINVAL Bad arguments (n or lineptr is NULL, or stream is not valid).
ENOMEM Allocation or reallocation of the line buffer failed.

ATTRIBUTES

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

??

? Interface ? Attribute ? Value ?

??

? getline(), getdelim() ? Thread safety ? MT-Safe ?

??

STANDARDS

POSIX.1-2008.

HISTORY

GNU, POSIX.1-2008.

EXAMPLES

#define _GNU_SOURCE

```
#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)