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 'getdelim.3p' command

$ man getdelim.3p

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

getdelim, getline ? read a delimited record from stream

SYNOPSIS

```
#include <stdio.h>
ssize_t getdelim(char **restrict lineptr, size_t *restrict n,
    int delimiter, FILE *restrict stream);
ssize_t getline(char **restrict lineptr, size_t *restrict n,
    FILE *restrict stream);
```

DESCRIPTION

The getdelim() function shall read from stream until it encounters a character matching the delimiter character. The delimiter argument is an int, the value of which the application shall ensure is a character representable as an unsigned char of equal value that terminates the read process. If the delimiter argument has any other value, the behav? ior is undefined.

The application shall ensure that *lineptr is a valid argument that could be passed to the free() function. If *n is non-zero, the applica?

tion shall ensure that *lineptr either points to an object of size at least *n bytes, or is a null pointer.

If *lineptr is a null pointer or if the object pointed to by *lineptr is of insufficient size, an object shall be allocated as if by malloc() or the object shall be reallocated as if by realloc(), respectively, such that the object is large enough to hold the characters to be writ? ten to it, including the terminating NUL, and *n shall be set to the new size. If the object was allocated, or if the reallocation operation moved the object, *lineptr shall be updated to point to the new object or new location. The characters read, including any delimiter, shall be stored in the object, and a terminating NUL added when the delimiter or end-of-file is encountered.

The getline() function shall be equivalent to the getdelim() function with the delimiter character equal to the <newline> character.

The getdelim() and getline() functions may mark the last data access timestamp of the file associated with stream for update. The last data access timestamp shall be marked for update by the first successful ex? ecution of fgetc(), fgets(), fread(), fscanf(), getc(), getchar(), get? delim(), getline(), gets(), or scanf() using stream that returns data not supplied by a prior call to ungetc().

RETURN VALUE

Upon successful completion, the getline() and getdelim() functions shall return the number of bytes written into the buffer, including the delimiter character if one was encountered before EOF, but excluding the terminating NUL character. If the end-of-file indicator for the stream is set, or if no characters were read and the stream is at end-of-file, the end-of-file indicator for the stream shall be set and the function shall return -1. If an error occurs, the error indicator for the stream shall be set, and the function shall return -1 and set errno to indicate the error.

ERRORS

For the conditions under which the getdelim() and getline() functions shall fail and may fail, refer to fgetc().

In addition, these functions shall fail if:

EINVAL lineptr or n is a null pointer.

ENOMEM Insufficient memory is available.

These functions may fail if:

EOVERFLOW

The number of bytes to be written into the buffer, including the delimiter character (if encountered), would exceed {SSIZE_MAX}.

The following sections are informative.

EXAMPLES

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;
    char *line = NULL;
    size_t len = 0;
    ssize_t read;
    fp = fopen("/etc/motd", "r");
    if (fp == NULL)
        exit(1);
    while ((read = getline(&line, &len, fp)) != -1) {
        printf("Retrieved line of length %zu :\n", read);
        printf("%s", line);
    }
    if (ferror(fp)) {
        /* handle error */
    }
    free(line);
    fclose(fp);
    return 0;
}
```

APPLICATION USAGE

Setting *lineptr to a null pointer and *n to zero are allowed and a

recommended way to start parsing a file.

The ferror() or feof() functions should be used to distinguish between an error condition and an end-of-file condition.

Although a NUL terminator is always supplied after the line, note that strlen(*lineptr) will be smaller than the return value if the line con? tains embedded NUL characters.

RATIONALE

These functions are widely used to solve the problem that the fgets() function has with long lines. The functions automatically enlarge the target buffers if needed. These are especially useful since they reduce code needed for applications.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5, Standard I/O Streams, fgetc(), fgets(), free(), malloc(), realloc()

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