NAME

fgetln - get a line from a stream

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdio.h>

*char **
fgetln(*FILE *stream, size_t *len*);

DESCRIPTION

The **fgetln**() function returns a pointer to the next line from the stream referenced by *stream*. This line is *not* a C string as it does not end with a terminating NUL character. The length of the line, including the final newline, is stored in the memory location to which *len* points. (Note, however, that if the line is the last in a file that does not end in a newline, the returned text will not contain a newline.)

RETURN VALUES

Upon successful completion a pointer is returned; this pointer becomes invalid after the next I/O operation on *stream* (whether successful or not) or as soon as the stream is closed. Otherwise, NULL is returned. The **fgetln**() function does not distinguish between end-of-file and error; the routines feof(3) and ferror(3) must be used to determine which occurred. If an error occurs, the global variable *errno* is set to indicate the error. The end-of-file condition is remembered, even on a terminal, and all subsequent attempts to read will return NULL until the condition is cleared with clearerr(3).

The text to which the returned pointer points may be modified, provided that no changes are made beyond the returned size. These changes are lost as soon as the pointer becomes invalid.

ERRORS

[EBADF] The argument *stream* is not a stream open for reading.

[ENOMEM] The internal line buffer could not be expanded due to lack of available memory, or because it would need to expand beyond INT_MAX in size.

The **fgetln**() function may also fail and set *errno* for any of the errors specified for the routines fflush(3), malloc(3), read(2), stat(2), or realloc(3).

SEE ALSO

ferror(3), fgets(3), fgetwln(3), fopen(3), getline(3), putc(3)

HISTORY

The **fgetln**() function first appeared in 4.4BSD.

CAVEATS

Since the returned buffer is not a C string (it is not NUL terminated), a common practice is to replace the newline character with ‘\0’. However, if the last line in a file does not contain a newline, the returned text won’t contain a newline either. The following code demonstrates how to deal with this problem by allocating a temporary buffer:

```
char *buf, *lbuf;
size_t len;

lbuf = NULL;
while ((buf = fgetln(fp, &len)) != NULL) {
        if (buf[len - 1] == '\n')
                buf[len - 1] = '\0';
        else {
                /* EOF without EOL, copy and add the NUL */
                if ((lbuf = malloc(len + 1)) == NULL)
                        err(1, NULL);
                memcpy(lbuf, buf, len);
                lbuf[len] = '\0';
                buf = lbuf;
        }
        printf("%s\n", buf);
}
free(lbuf);
if (ferror(fp))
        err(1, "fgetln");
```