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 'fgets.3p' command

$ man fgets.3p

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

fgets ? get a string from a stream

SYNOPSIS

#include <stdio.h>

char *fgets(char *restrict s, int n, FILE *restrict stream);

DESCRIPTION

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.

The fgets() function shall read bytes from stream into the array pointed to by s until n-1 bytes are read, or a <newline> is read and transferred to s, or an end-of-file condition is encountered. A null byte shall be written immediately after the last byte read into the ar? ray. If the end-of-file condition is encountered before any bytes are read, the contents of the array pointed to by s shall not be changed.

The fgets() function 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 execution of fgetc(), fgets(), fread(), fscanf(), getc(), getchar(), getdelim(), getline(), gets(), or scanf() using stream that returns data not sup? plied by a prior call to ungetc().

RETURN VALUE

Upon successful completion, fgets() shall return s. If the stream is at end-of-file, the end-of-file indicator for the stream shall be set and fgets() shall return a null pointer. If a read error occurs, the error indicator for the stream shall be set, fgets() shall return a null pointer, and shall set errno to indicate the error.

ERRORS

Refer to fgetc().

The following sections are informative.

EXAMPLES

Reading Input

The following example uses fgets() to read lines of input. It assumes that the file it is reading is a text file and that lines in this text file are no longer than 16384 (or {LINE_MAX} if it is less than 16384 on the implementation where it is running) bytes long. (Note that the standard utilities have no line length limit if sysconf(_SC_LINE_MAX) returns -1 without setting errno. This example assumes that sysconf(_SC_LINE_MAX) will not fail.)

```
#include <limits.h>
#include <stdio.h>
#include <unistd.h>
#define MYLIMIT 16384
char *line;
int line_max;
if (LINE_MAX >= MYLIMIT) {
    // Use maximum line size of MYLIMIT. If LINE_MAX is
    // bigger than our limit, sysconf() cannot report a
    // smaller limit.
```

```
        line_max = MYLIMIT;
    } else {
        long limit = sysconf(_SC_LINE_MAX);
        line_max = (limit < 0 || limit > MYLIMIT) ? MYLIMIT : (int)limit;
    }
    // line_max + 1 leaves room for the null byte added by fgets().
    line = malloc(line_max + 1);
    if (line == NULL) {
        // out of space
        ...
        return error;
    }
    while (fgets(line, line_max + 1, fp) != NULL) {
        // Verify that a full line has been read ...
        // If not, report an error or prepare to treat the
        // next time through the loop as a read of a
        // continuation of the current line.
        ...
        // Process line ...
        ...
    }
    free(line);
    ...
```

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5, Standard I/O Streams, fgetc(), fopen(), fread(), fscanf(), getc(), getchar(), getdelim(), gets(), ungetc()

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