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 'fmemopen.3p' command

$ man fmemopen.3p

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

fmemopen ? open a memory buffer stream

SYNOPSIS

```
#include <stdio.h>
FILE *fmemopen(void *restrict buf, size_t size,
   const char *restrict mode);
```

DESCRIPTION

The fmemopen() function shall associate the buffer given by the buf and size arguments with a stream. The buf argument shall be either a null pointer or point to a buffer that is at least size bytes long.

The mode argument points to a string. If the string is one of the fol? lowing, the stream shall be opened in the indicated mode. Otherwise, the behavior is undefined.

r Open the stream for reading.

w Open the stream for writing.

a Append; open the stream for writing at the first null byte.

r+ Open the stream for update (reading and writing).

w+ Open the stream for update (reading and writing). Truncate the buffer contents.

a+ Append; open the stream for update (reading and writing); the initial position is at the first null byte.

Implementations shall accept all mode strings allowed by fopen(), but the use of the character 'b' shall produce implementation-defined re? sults, where the resulting FILE * need not behave the same as if 'b' were omitted.

If a null pointer is specified as the buf argument, fmemopen() shall allocate size bytes of memory as if by a call to malloc(). This buffer shall be automatically freed when the stream is closed. Because this feature is only useful when the stream is opened for updating (because there is no way to get a pointer to the buffer) the fmemopen() call may fail if the mode argument does not include a '+'.

The stream shall maintain a current position in the buffer. This posi? tion shall be initially set to either the beginning of the buffer (for r and w modes) or to the first null byte in the buffer (for a modes). If no null byte is found in append mode, the initial position shall be set to one byte after the end of the buffer.

If buf is a null pointer, the initial position shall always be set to the beginning of the buffer.

The stream shall also maintain the size of the current buffer contents; use of fseek() or fseeko() on the stream with SEEK_END shall seek rela? tive to this size. For modes r and r+ the size shall be set to the value given by the size argument. For modes w and w+ the initial size shall be zero and for modes a and a+ the initial size shall be:

* Zero, if buf is a null pointer
* The position of the first null byte in the buffer, if one is found
* The value of the size argument, if buf is not a null pointer and no null byte is found

A read operation on the stream shall not advance the current buffer po? sition beyond the current buffer size. Reaching the buffer size in a read operation shall count as ``end-of-file''. Null bytes in the buffer

shall have no special meaning for reads. The read operation shall start at the current buffer position of the stream.

A write operation shall start either at the current position of the stream (if mode has not specified 'a' as the first character) or at the current size of the stream (if mode had 'a' as the first character). If the current position at the end of the write is larger than the current buffer size, the current buffer size shall be set to the current posi? tion. A write operation on the stream shall not advance the current buffer size beyond the size given in the size argument.

When a stream open for writing is flushed or closed, a null byte shall be written at the current position or at the end of the buffer, depend? ing on the size of the contents. If a stream open for update is flushed or closed and the last write has advanced the current buffer size, a null byte shall be written at the end of the buffer if it fits.

An attempt to seek a memory buffer stream to a negative position or to a position larger than the buffer size given in the size argument shall fail.

RETURN VALUE

Upon successful completion, fmemopen() shall return a pointer to the object controlling the stream. Otherwise, a null pointer shall be re? turned, and errno shall be set to indicate the error.

ERRORS

The fmemopen() function shall fail if:

EMFILE {STREAM_MAX} streams are currently open in the calling process.

The fmemopen() function may fail if:

EINVAL The value of the mode argument is not valid.

EINVAL The buf argument is a null pointer and the mode argument does not include a '+' character.

EINVAL The size argument specifies a buffer size of zero and the imple? mentation does not support this.

ENOMEM The buf argument is a null pointer and the allocation of a buf? fer of length size has failed.

EMFILE {FOPEN_MAX} streams are currently open in the calling process.

The following sections are informative.

EXAMPLES

```
#include <stdio.h>
#include <string.h>
static char buffer[] = "foobar";
int
main (void)
{
    int ch;
    FILE *stream;
    stream = fmemopen(buffer, strlen (buffer), "r");
    if (stream == NULL)
        /* handle error */;
    while ((ch = fgetc(stream)) != EOF)
        printf("Got %c\n", ch);
    fclose(stream);
    return (0);
}
```

This program produces the following output:

```
Got f
Got o
Got o
Got b
Got a
Got r
```

APPLICATION USAGE

None.

RATIONALE

This interface has been introduced to eliminate many of the errors en? countered in the construction of strings, notably overflowing of strings. This interface prevents overflow.

FUTURE DIRECTIONS

A future version of this standard may mandate specific behavior when

the mode argument includes 'b'.

A future version of this standard may require support of zero-length buffer streams explicitly.

SEE ALSO

fdopen(), fopen(), freopen(), fseek(), malloc(), open_memstream()

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

COPYRIGHT