setvbuf

Defined in header <stdio.h>

```
int setvbuf( FILE *         stream, char *         buffer,
             int mode, size_t size );                          (until C99)
int setvbuf( FILE *restrict stream, char *restrict buffer,
             int mode, size_t size );                          (since C99)
#define _IOFBF     /*unspecified*/
#define _IOLBF     /*unspecified*/
#define _IONBF     /*unspecified*/
```

Changes the buffering mode of the given file stream `stream` as indicated by the argument `mode`. In addition,

- If `buffer` is a null pointer, resizes the internal buffer to `size`.
- If `buffer` is not a null pointer, instructs the stream to use the user-provided buffer of size `size` beginning at `buffer`. The stream must be closed (with `fclose`) before the lifetime of the array pointed to by `buffer` ends. The contents of the array after a successful call to `setvbuf` are indeterminate and any attempt to use it is undefined behavior.

Parameters

stream - the file stream to set the buffer to

buffer - pointer to a buffer for the stream to use or null pointer to change size and mode only

mode - buffering mode to use. It can be one of the following values:

_IOFBF	full buffering
_IOLBF	line buffering
_IONBF	no buffering

size - size of the buffer

Return value

`0` on success or nonzero on failure.

Notes

This function may only be used after `stream` has been associated with an open file, but before any other operation (other than a failed call to `setbuf`/`setvbuf`).

Not all `size` bytes will necessarily be used for buffering: the actual buffer size is usually rounded down to a multiple of 2, a multiple of page size, etc.

On many implementations, line buffering is only available for terminal input streams.

A common error is setting the buffer of stdin or stdout to an array whose lifetime ends before the program terminates:

```
int main(void) {
    char buf[BUFSIZ];
    setbuf(stdin, buf);
} // lifetime of buf ends, undefined behavior
```

The default buffer size `BUFSIZ` is expected to be the most efficient buffer size for file I/O on the implementation, but POSIX fstat (http://pubs.opengroup.org/onlinepubs/9699919799/functions/fstat.html) often provides a better estimate.

Example

One use case for changing buffer size is when a better size is known.

Run this code

```
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <sys/stat.h>

int main(void)
{
    FILE* fp = fopen("/tmp/test.txt", "w+");
    if(fp == NULL) {
       perror("fopen"); return EXIT_FAILURE;
    }

    struct stat stats;
    int fileno(FILE*);
    if(fstat(fileno(fp), &stats) == -1) { // POSIX only
        perror("fstat"); return EXIT_FAILURE;
    }

    printf("BUFSIZ is %d, but optimal block size is %ld\n", BUFSIZ, stats.st_blksize);
    if(setvbuf(fp, NULL, _IOFBF, stats.st_blksize) != 0) {
       perror("setvbuf failed"); // POSIX version sets errno
       return EXIT_FAILURE;
    }

    int ch;
    while((ch=fgetc(fp)) != EOF); // read entire file: use truss/strace to
                                  // observe the read(2) syscalls used

    fclose(fp);
    return EXIT_SUCCESS;
}
```

Possible output:

```
BUFSIZ is 8192, but optimal block size is 65536
```

References

- C17 standard (ISO/IEC 9899:2018):
  - 7.21.5.6 The setvbuf function (p: 225)
- C11 standard (ISO/IEC 9899:2011):
  - 7.21.5.6 The setvbuf function (p: 308)
- C99 standard (ISO/IEC 9899:1999):
  - 7.19.5.6 The setvbuf function (p: 273-274)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.9.5.6 The setvbuf function

See also

setbuf	sets the buffer for a file stream (function)

C++ documentation for **setvbuf**