**Description:**

This C program improves file reading by using a buffered I/O system. It uses a file control block (FCB) to keep track of open files and their data. The program helps read files more efficiently by using a temporary storage space called a buffer. This buffer holds parts of the file, reducing the number of operations needed to access the file data. The program manages opening, reading, and closing files while ensuring users don't request more data than the file contains.

**Approuch/What I did:**

**Note:** I prefer the camelCase style more, but in this assignment, I switched to snake\_case style since the README mentioned snake case style functions, and the skeleton I was provided had more snake case style. Thus, I used snake\_case for better reading and consistency in this project.

First, I read the instructions multiple times and checked the provided template. I made notes on what was needed and then followed the instructions. Whenever I didn't understand something, I read the official libraries to learn more. Most of the time, I was reading and taking memos:

Implement functions in b\_io.c:

1. b\_io\_fd b\_open(char \*filename, int flags)
2. int b\_read(b\_io\_fd fd, char \*buffer, int count)
3. int b\_close(b\_io\_fd fd)

Use only the provided low-level APIs, i.e., LBAread and GetFileInfo.

b\_open function:

* Returns an integer file descriptor for tracking the file.
* Allocates a 512-byte buffer (B\_CHUNK\_SIZE) for read operations.
* Calls GetFileInfo to find the file size and location.
* If the file is not found, GetFileInfo returns NULL.

b\_read function:

* Reads B\_CHUNK\_SIZE byte chunks at a time from LBAread into your buffer.
* Copies the appropriate bytes from your buffer to the caller's buffer (as binary data).
* Handles read requests greater than B\_CHUNK\_SIZE.
* Keeps track of the file size and indicates when the end of the file is reached (returns 0).
* Returns the number of bytes transferred to the caller's buffer.

b\_close function:

* Frees any resources being used.

Purpose and logic:

 char \*buffer in the b\_fcb struct provides storage for the buffer used for the file, which allows for efficient read operations.

 int buf\_idx to the b\_fcb struct stores the index into the buffer where data can be read from. This allows for efficient tracking of the location of the next byte to read from the buffer.

 int block\_num to the b\_fcb struct stores the current block number of the file being read. This allows for efficient tracking of the current position in the file.

get\_current\_gpos function calculates the current get position of a file descriptor based on its buffer index and block number. This allows the code to keep track of where it is in the file, even if it has to read data in chunks.

 copy\_available\_bytes function copies the minimum of count and available bytes to the destination using a given file descriptor. This allows for efficient reads of the file, copying data from the buffer to the user's buffer as required.

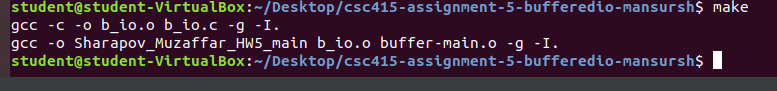
 b\_open allocates a buffer for the file and stores the buffer index and block number as -1 to indicate that the buffer is empty and no read operation has been called yet. This allows for efficient read operations later.

 b\_read reads the file in chunks of B\_CHUNK\_SIZE bytes and buffers the data in the FCB buffer. The function first reads directly into the user's buffer if the requested count is greater than B\_CHUNK\_SIZE. Then, it reads the file chunk by chunk, copying data from the buffer to the user's buffer using copy\_available\_bytes. It also keeps track of the current block number and buffer index in the FCB to allow for efficient reads.

 b\_close frees the buffer allocated for the file and resets the block\_num, buf\_idx, and fi fields of the FCB. This ensures that any allocated memory is properly freed and the FCB is ready for use with another file.

**Analysis**

**Screen shot of compilation of the program:**

****

**Screen shot of compilation and the execution of the program:**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated