Kurtis Chan ID: 918139175 Github: kchanw CSC415-03 Operating Systems

## Assignment 4 – Word Blast

**Description**: Write three functions, b\_open(), b\_read(), and b\_close(). b\_open() returns a file descriptor where it gets file info and allocates memory for the buffer. b\_read() reads the file in chunks of size 512 bytes using LBAread() for each block. It reads until the requested number of bytes is read, or until it reaches the end of the file while returning how many bytes have been read. The b\_close will free any resources used.

Approach / What I Did: At first for the struct b\_fcb, I did not understand why it said to add any other variables needed, but once I got to b\_read() I now understood why I needed a way to track where I am in the file, where I am in the buffer, and we need a pointer for our buffer. For b\_open() it will return the index given by b\_getFCB();. It will return -1 if the FCB array is full and when it fails to get a pointer. It also allocates memory of size B\_CHUNK\_SIZE which is 512 for that FCB's buffer. This will also initialize the file position, and buffer position to 0 so that it will start at the beginning of the file. I also did the first LBAread() for the FCB so that I won't have to do it in the b\_read().

For the b\_read() function, I have three different situations when reading. 1, where the count for reading is less than the size of the buffer, then it will just read and memcpy to their buffer. Each memcpy is followed by adding to the buffer position and file position to make sure we are in the right place for both the buffer and file. 2, when multiple blocks are needed like when you have 8 spaces left in your buffer, but the user calls to read 800 bytes. You will need multiple blocks, the current one you are in, one for a full chunk(512 bytes), and one for the remaining. 3, the last one is when you need to go to the next block. Each time to get a new block I needed to call LBAread() and to reset buffPos while updating filePos. At the end of the function, it will return bytesRead which is a variable that kept track of how many bytes you are reading. At the beginning of b\_read() there is an if statement to handle when the requested amount of bytes is more than the what you have left in the file to read, so to handle that I trimmed the count to what is left to read and that number is passed instead of the user requested count. Otherwise, just use the user requested count. For close it frees the buffer of the index in the FCB and makes the fi which is the pointer to the file information NULL.

Issues and Resolutions: A big issue I had was trying to make it into a loop for how many blocks I needed for the current call, but ended up confusing myself when doing this. I ended starting again and started with when you do have enough space in the buffer, then I worked on when you needed an extra block, then I worked on when it needed more than that. I do think that in this code it won't handle more than 3 blocks being read since the largest it does is the rest of a block, a full block, and the start of a block. Another thing I got stuck on was using LBAread() and how I only needed to call it when I needed a new block. At first, I kept calling it and incrementing after each b\_read() call and memcpy.

## Screen shots:





