

Assignment 5 – Buffered I/O

Description:

This project involves creating efficient buffered I/O functions in C to read data from files. Through this assignment to learn how to optimize file reading, manage multiple files, and handle end-of-file situations correctly. Using low-level APIs and ensure that the buffered I/O operations are performed efficiently.

Approach:

To ensure a thorough understanding of the existing structure and the functions that require implementation, I plan on carefully reviewing the provided skeleton code (`'b_io.c'`).

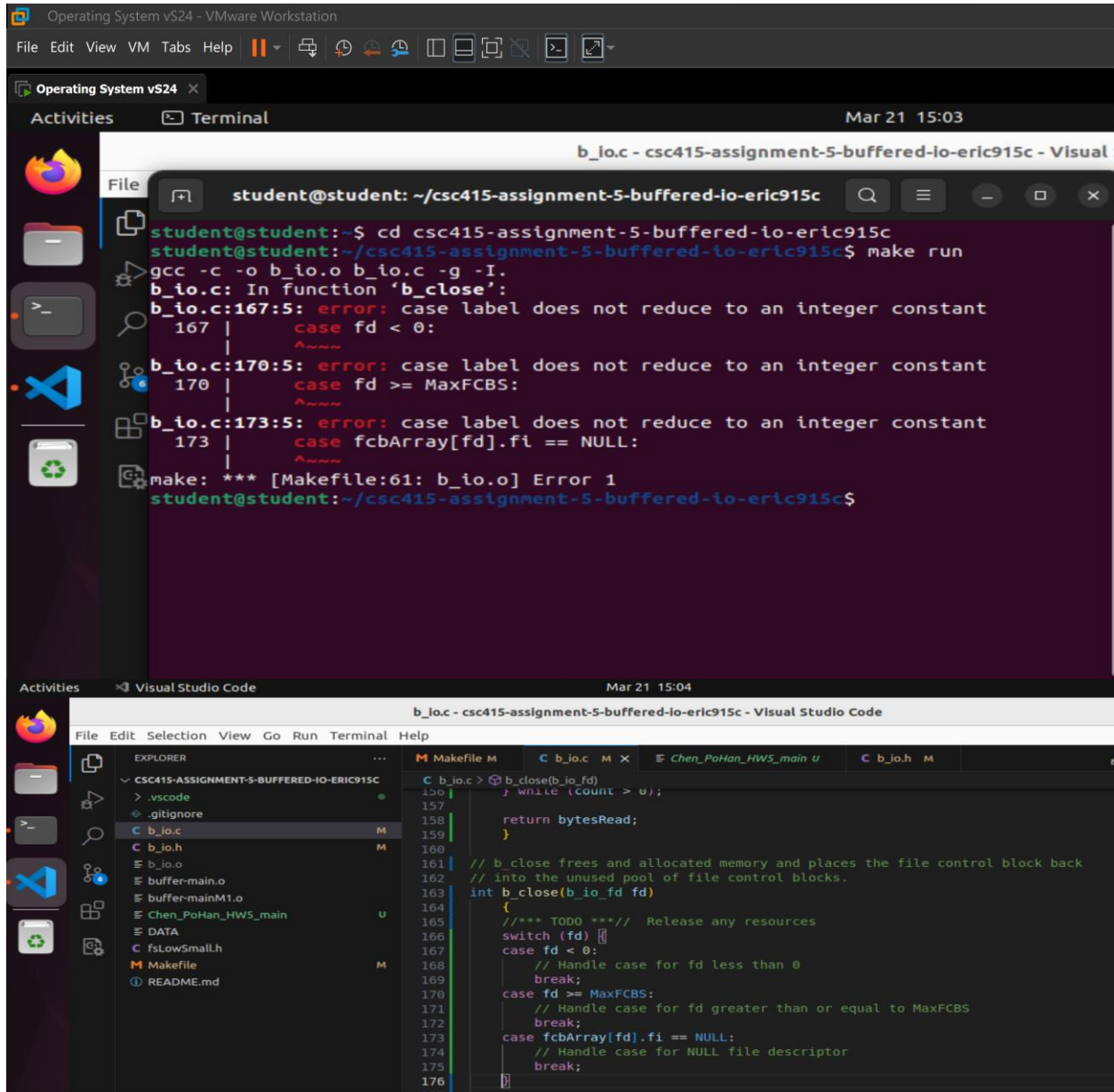
The `'b_open'` function starts by allocating a file descriptor and initializing the file control block (FCB) for the opened file. To retrieve file information and allocate a buffer of `'B_CHUNK_SIZE'` bytes if necessary, the function uses the `'GetFileInfo'` function. After obtaining file information, `'b_open()'` uses the `'b_getFCB()'` function to get a free file descriptor from the system. If there are no free descriptors available or the obtained descriptor exceeds the maximum allowable limit (`'MaxFCBS'`), the function returns an error code indicating that it is unable to open the file. Assuming that a valid file descriptor is obtained, the function associates the retrieved file information with the corresponding FCB, initializing its position to the beginning of the file.

The `'b_read'` function is the second part of the process. It uses `'LBaread'` to read data in chunks of `'B_CHUNK_SIZE'` and buffer it. The data is then copied to the caller's buffer while keeping track of the position in both the buffer and the file. To ensure proper handling of end-of-file conditions, the function `'b_read'` keeps track of the position in the buffer and file, ensuring a smooth transfer to the caller's buffer. This tracking mechanism allows for accurate management of data retrieval, ensuring that the correct parts of the file are read and transferred to the caller's buffer.

The `'b_close()'` function is responsible for releasing the resources associated with an archive. It checks for invalid archive descriptors and resets FCB. When `'b_close()'` is invoked, it starts by verifying the validity of the provided file descriptor. It performs a thorough check to ensure that the descriptor falls within the valid range and corresponds to an open file. This step is necessary to prevent errors and ensure system stability. Once the file descriptor is confirmed valid, `'b_close()'` proceeds to release the resources associated with the file. This includes deallocating any memory buffers or resources that were allocated during file operations. By releasing these resources, the function ensures efficient memory management and prevents memory leaks.

Issues and Resolutions:

My first issue was error in `'b_close()'` function: "case label does not reduce to an integer constant" when using a `'switch'` statement.



The screenshot shows a VMware Workstation window titled "Operating System vS24 - VMware Workstation". Inside the VM, there is a terminal window and a Visual Studio Code editor. The terminal window shows the following commands and output:

```
student@student:~$ cd csc415-assignment-5-buffered-io-eric915c
student@student:~/csc415-assignment-5-buffered-io-eric915c$ make run
gcc -c -o b_io.o b_io.c -g -I.
b_io.c: In function 'b_close':
b_io.c:167:5: error: case label does not reduce to an integer constant
167 |     case fd < 0:
    |     ^
b_io.c:170:5: error: case label does not reduce to an integer constant
170 |     case fd >= MaxFCBS:
    |     ^
b_io.c:173:5: error: case label does not reduce to an integer constant
173 |     case fcbArray[fd].fi == NULL:
    |     ^
make: *** [Makefile:61: b_io.o] Error 1
student@student:~/csc415-assignment-5-buffered-io-eric915c$
```

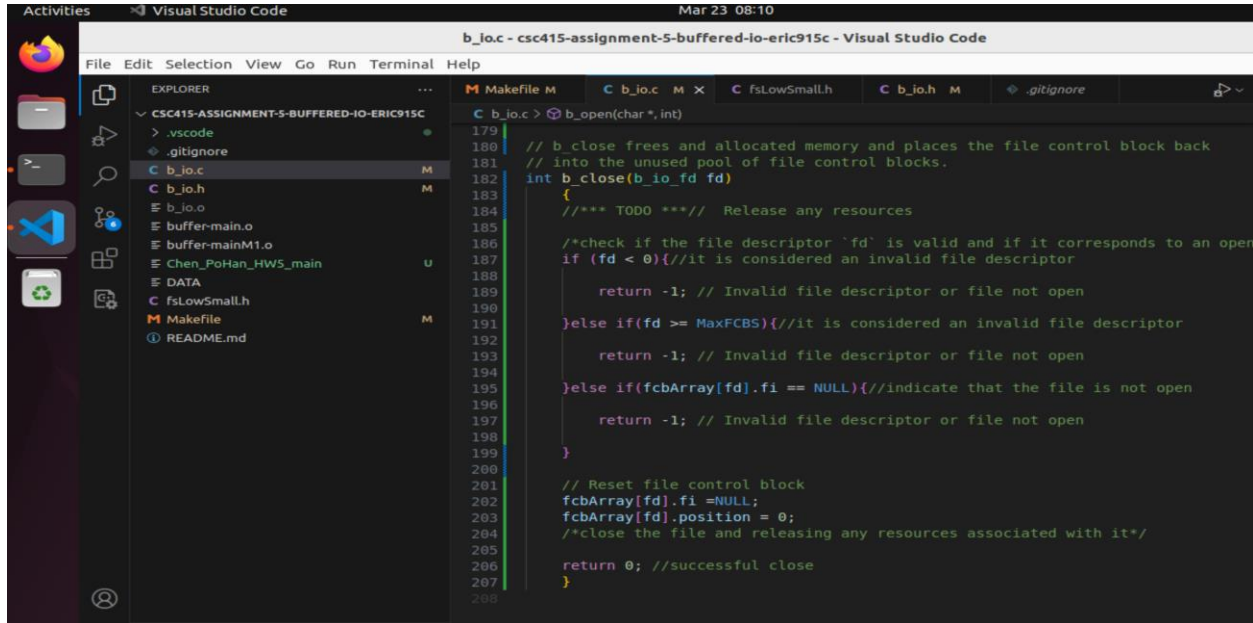
The Visual Studio Code editor shows the file explorer on the left with the following files:

- EXPLORER
- ▼ CSC415-ASSIGNMENT-5-BUFFERED-IO-ERIC915C
- .vscode
- .gitignore
- C b_io.c
- C b_io.h
- E b_io.o
- E buffer-main.o
- E buffer-mainM1.o
- E Chen_PoHan_HWS_main
- E DATA
- C fsLowSmall.h
- M Makefile
- README.md

The editor shows the following code in the file `b_io.c`:

```
156 |     } while (count > 0);
157 |
158 |     return bytesRead;
159 | }
160 |
161 | // b_close frees and allocated memory and places the file control block back
162 | // into the unused pool of file control blocks.
163 | int b_close(b_io_fd fd)
164 | {
165 |     /*** TODO ***/ Release any resources
166 |     switch (fd) {
167 |     case fd < 0:
168 |         // Handle case for fd less than 0
169 |         break;
170 |     case fd >= MaxFCBS:
171 |         // Handle case for fd greater than or equal to MaxFCBS
172 |         break;
173 |     case fcbArray[fd].fi == NULL:
174 |         // Handle case for NULL file descriptor
175 |         break;
176 |     }
```

I resolved it by checking the `case` labels of the `switch` statement do not reduce to constant integer values, which is a requirement in C. To resolve this, I modified the `b_close()` function to use `if-else` statements instead of a `switch` statement for handling different cases based on the value of `fd`.

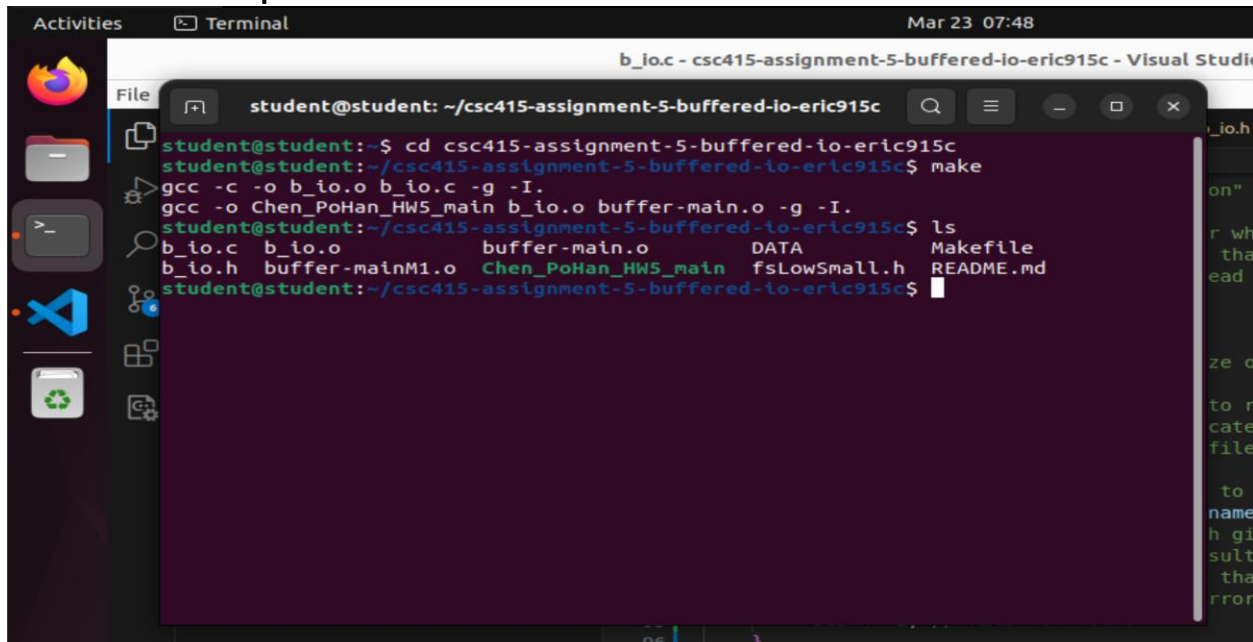


The screenshot shows the Visual Studio Code editor with the file `b_io.c` open. The file contains C code for a buffered I/O library. The code includes comments and function definitions for `b_open` and `b_close`. The `b_close` function is currently selected, showing its implementation which checks for valid file descriptors and releases resources.

```
179 // b_close frees and allocated memory and places the file control block back
180 // into the unused pool of file control blocks.
181 int b_close(b_io_fd fd)
182 {
183     /*** TODO ***/ Release any resources
184
185     /*check if the file descriptor 'fd' is valid and if it corresponds to an open
186     if (fd < 0){//it is considered an invalid file descriptor
187         return -1; // Invalid file descriptor or file not open
188     }else if(fd >= MaxFCBS){//it is considered an invalid file descriptor
189         return -1; // Invalid file descriptor or file not open
190     }else if(fcbArray[fd].fi == NULL){//indicate that the file is not open
191         return -1; // Invalid file descriptor or file not open
192     }
193
194     // Reset file control block
195     fcbArray[fd].fi =NULL;
196     fcbArray[fd].position = 0;
197     /*close the file and releasing any resources associated with it*/
198     return 0; //successful close
199 }
```

Analysis: (If required for the assignment)

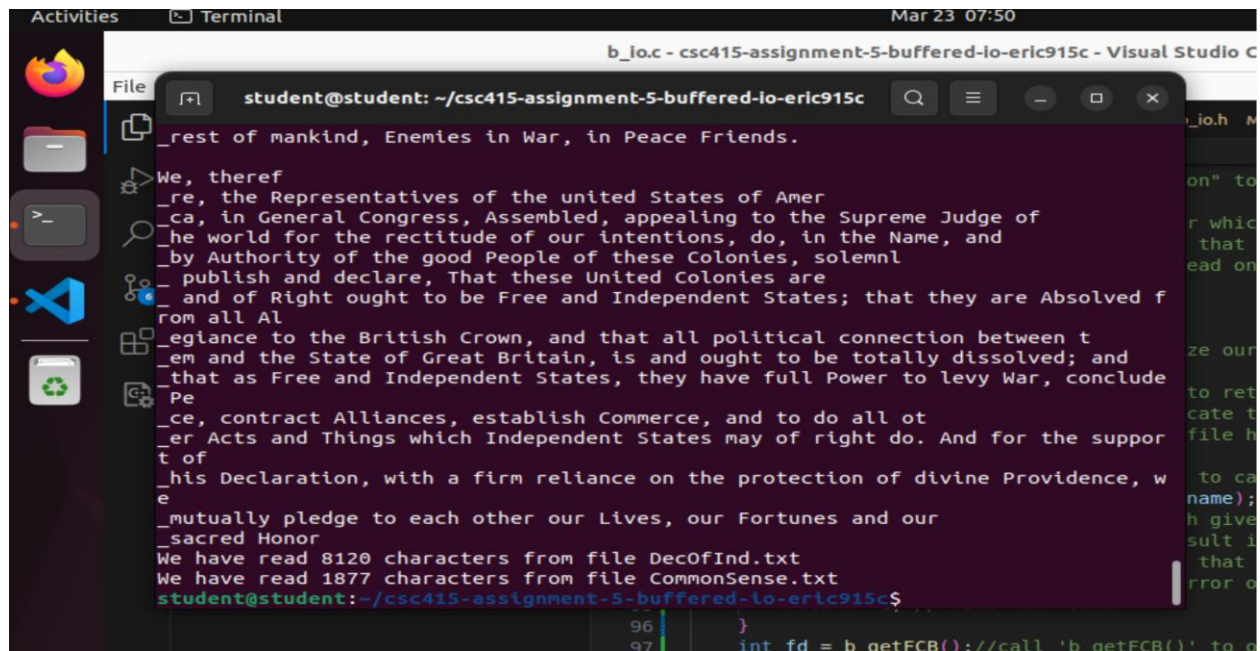
Screen shot of compilation:



The screenshot shows a terminal window with the following commands and output:

```
student@student: ~/csc415-assignment-5-buffered-io-eric915c
student@student:~$ cd csc415-assignment-5-buffered-io-eric915c
student@student:~/csc415-assignment-5-buffered-io-eric915c$ make
gcc -c -o b_io.o b_io.c -g -I.
gcc -o Chen_PoHan_HWS_main b_io.o buffer-main.o -g -I.
student@student:~/csc415-assignment-5-buffered-io-eric915c$ ls
b_io.c  b_io.o          buffer-main.o  DATA          Makefile
b_io.h  buffer-mainM1.o  Chen_PoHan_HWS_main  fsLowSmall.h  README.md
student@student:~/csc415-assignment-5-buffered-io-eric915c$
```

Screen shot(s) of the execution of the program:



```
Activities Terminal Mar 23 07:50
b_io.c - csc415-assignment-5-buffered-io-eric915c - Visual Studio C
File student@student: ~/csc415-assignment-5-buffered-io-eric915c
rest of mankind, Enemies in War, in Peace Friends.
We, theref
re, the Representatives of the united States of Amer
ca, in General Congress, Assembled, appealing to the Supreme Judge of
he world for the rectitude of our intentions, do, in the Name, and
by Authority of the good People of these Colonies, solemn
publish and declare, That these United Colonies are
and of Right ought to be Free and Independent States; that they are Absolved f
rom all Al
egiance to the British Crown, and that all political connection between t
em and the State of Great Britain, is and ought to be totally dissolved; and
that as Free and Independent States, they have full Power to levy War, conclude
Pe
ce, contract Alliances, establish Commerce, and to do all ot
er Acts and Things which Independent States may of right do. And for the suppor
t of
his Declaration, with a firm reliance on the protection of divine Providence, w
e
mutually pledge to each other our Lives, our Fortunes and our
sacred Honor
We have read 8120 characters from file DecOfInd.txt
We have read 1877 characters from file CommonSense.txt
student@student:~/csc415-assignment-5-buffered-io-eric915c$
96 }
97 int fd = b.getFCB();//call 'b.getFCB()' to o
```