Abbas Mahdavi ID: 918345420
Github: AbbasMahdavi021 CSC415 -03 Operating Systems

Assignment 5 – Buffered IO

Description:
This assignment uses the command line arguments to specify data file and the desired target file(s). The main program uses b_open, reads some variable number of characters at a time from the file using b_read, prints those characters to the screen (ending in a newline character), and loops until it has read the entire file, then b_close the file and exit. The program may open multiple files at one time and will request on chunk and print that one for each file, so the output will contain a line from file 1 than a line from file 2, etc until the end of file.

Approach / What I Did:
For this assignment, I first initialized a few new variables that we'll keep track of the fd, the index of the buffer, position in the file, and the current block we're buffering.
I then initialized these variables in b_open, using the fcbArray, and the file descriptor variable.
I checked for allocation errors, and returned a negative number when something failed.
b_open return an integer, fileDes.

Moving on to the b_read, two new int vars are need to keep track of the total number of bytes read in total, which we will return in this function, as well as keeping track of the index in the input buffer.
I created a while loop where, given the int count from the parameter, is bigger than 0, we would buffer the input buffer into the file buffer. There are a few scenarios that I need to take care of, and so I started with if statements to take care of all these possibilities.
First thing in the while loop is setting count to the file size, and the position we are in that file.

I populated the file buffer, by getting 1 block, from the fd, using LBAread, and using file information (fi), get the location of the block. This gets done only when buffer index is 0, hence a new block.

Then I created an if statement for when the count can fit in the file buffer, (count smaller than buffer index), and using memcpy, I copy the whole bytes from the file buffer into the input buffer. This if statement for this case, also offsets the index of the buffer, so that we do not overwrite any pervious bytes we read into the input buffer.

The other case I had to take care of was when count could not fit in the file buffer. In this case, I copied what we COULD fit in the file buffer. I know how much space we have, by subtracting Chunk Size by the index buffer. Then I copy the bytes of count into the file buffer. Update everything, the position in the file, number of total bytes read, ect, and setting the buffer index to 0, so that we can get at the start of a "new" block. And the loop continues until all the file size bytes are read and buffered.
The b_read function return the total number of bytes read.

I then freed all resources in the b_close function, and tested/debug my program.

Abbas Mahdavi ID: 918345420
Github: AbbasMahdavi021 CSC415 -03 Operating Systems

Issues and Resolutions:

- The main issue I had was properly keeping the track of everything, and implementing them properly in the if statements. It kept getting "over-written", and would not buffer properly.

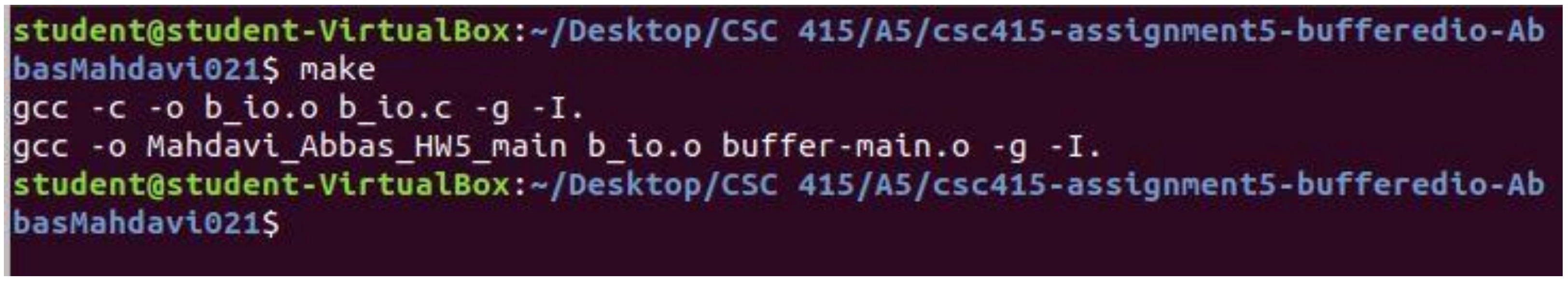
  I had to introduce new variables to store them in, and that made it easier to track and update. The issue was also the fact that I could not know then the end of the file is reached. I had to restructure my while loop so that it updates everything properly, not bytes are over-written, and the end of the file is not an issue.

- Another issue I had was how to deal with the count being bigger than file buffer.

  Using the B_CHUNK_SIZE and the current index of the buffer, I managed to see how much space we have left. 0-512. Then I managed to fill in the empty space available, update the count, the buffer, and file position, and reset the index to 0, so that in the next loop, a new block is read, and the process continues.

- The b_read was the main part of this assignment, I had many "small" issues why trying to write the code for, that I don't think is needed to mention, as writing, debugging, and improving of it was expanded over many days.

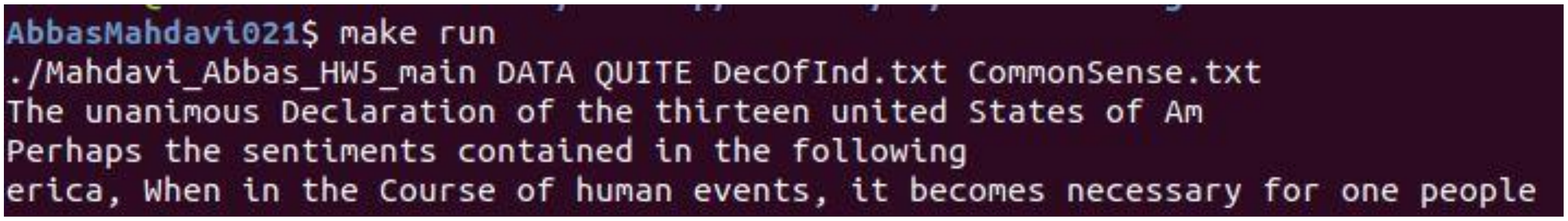
Screen shot of compilation:



```
student@student-VirtualBox:~/Desktop/CSC 415/A5/csc415-assignment5-bufferedio-Ab
basMahdavi021$ make
gcc -c -o b_io.o b_io.c -g -I.
gcc -o Mahdavi_Abbas_HW5_main b_io.o buffer-main.o -g -I.
student@student-VirtualBox:~/Desktop/CSC 415/A5/csc415-assignment5-bufferedio-Ab
basMahdavi021$
```

Abbas Mahdavi   ID: 918345420
Github: AbbasMahdavi021   CSC415 -03 Operating Systems

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

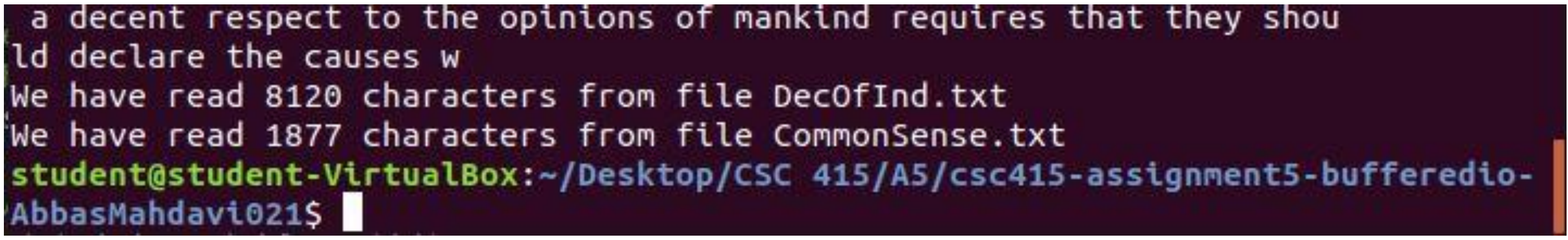


```
AbbasMahdavi021$ make run
./Mahdavi_Abbas_HW5_main DATA QUITE DecOfInd.txt CommonSense.txt
The unanimous Declaration of the thirteen united States of Am
Perhaps the sentiments contained in the following
erica, When in the Course of human events, it becomes necessary for one people
```

•

•

•



```
 a decent respect to the opinions of mankind requires that they shou
ld declare the causes w
We have read 8120 characters from file DecOfInd.txt
We have read 1877 characters from file CommonSense.txt
student@student-VirtualBox:~/Desktop/CSC 415/A5/csc415-assignment5-bufferedio-
AbbasMahdavi021$
```