Tomas Woldemichael
Program 3
May 25th, 2021

Standard I/O library

For this assignment I built a replica of the standard input and output library for C++. The purpose of this assignment is to provide an interface between the between the user and the underlying system calls. Calling a system call such as read() and write() can be very expensive in terms of time as systems call cause going to the disk for allocated memory space. Therefore we have implemented an fread() and fwrite() method that has a buffer mechanism that will reduce the amount of read and write calls to the under lying operating system.

The driver file takes in a text file (“hamlet.txt”/ “othello.txt”) and performs various operations on the text file such as but not limited to Create a new File class, fread(), fgets(), fgetc(), fwrite(),fputs(), fputc(). With this main function the class is able to interfaces read and writing to a file simply in the most efficient way possible by using the allocated buffer in the file object that only makes a system call such as read and write when the entire buffer is full or as full as is possible can be in terms of bytes.

fread()

fread is one of the main functions that filters the amount of read calls in to the underlying systems. One of the first thing I do when coming into the freads functions is checking if the file has a read flag. If the user is able to read then the user can either select a buffered mode or non-buffer mode. Fread read will identify how many bytes the user wants, and if the buffer is empty will make one read call to fill the buffer. Then each time the user request bytes is will copy over the memory from the buffer to the void ptr and moved the position pointer. Ultimately by filling up the buffer on the one read call the user can just copy memory from the buffer as they would like. One the position reaches the end You can clear the buffer and make another read call to fill the buffer back up to be used for the reading from the file.

The other reading functions such as fgets and fgetc reuse the freads buffer properties. For example in the fgetc that reads just a single character therefore what I do is create a user buffer with just a single character. Although the fread already takes in a large amount of charters into the buffer its only will copy a single character over and move the position pointer by one integer.

Tomas Woldemichael
Program 3
May 25th, 2021

fwrite()

fwrite is one of the main functions that filters out the amount write calls to the disk. The way that we were able to filter out the amount of write calls to the disk is by using the buffer within the stream. The first thing that the fwrite functions does is makes sure that the stream flag has write capabilities. Once I have identified that the stream has write capabilities the buffer will go ahead and load as much data as it can into the buffer. As the data get loaded into the buffer it will see how much data the user is requesting and move the bytes into the users queue and move the position of the location of the buffer. IF the buffer is filled it will then clear the buffer and add new data into the queue.

The smaller functions such as fputs and fputc use the functionality of the fwrite command so that they can use the already established buffer of the system. I just pass in the amount of data that is necessary for the action and carry that through.

Limitations & Extensions

Some of the possible limitation to this code most likely has to do with the buffer being able to both write and read. The reason being that the operations that need to take place are much more difficult. If a file has both the read and write possibility the buffer must be cleared anytime the flag has switched either through a fpurge or a fflush.

Some possible ways to extend this program is by making it more efficient in the amount of time is needs to write to the disk. By doing this it could be closer to the standard input output stream. Another way to potentially extend this program is by adding more function similar to the standard input output libaray. There are many more functions that we could potentially implement that have not bee incorporated such as overloading the >> and << operators.