One of the first things I fixed was my lock, when testing for assignment 2.2 I realized that I messed up my locking because I was creating it at every read, write, etc. instead of just at initialization and locking and release when needed. Without the correct locking, multithreaded race conditions could occur and lead to inaccurate output. I locked at the start of readat, writeat, undo, and close and then released at the end of the functions because they are all updating data fields in some way.

I also added in a length field to init to ensure that I was not bypassing SeekPastEndOfFileError. The directions from assignment 2.1 say you cannot readat for every write at, so the readat had to happen initially, then be updated after. I updated the length in writeat after writing to the LPfile to the max of the current length or the new write file length. After, I also checked if the offset provided was within range or produce a SeekPastEndOfFileError error. I checked for this before writing to LPfile so that a write would not occur if there was an invalid offset, and also outside of that condition in case there is just a write at with no pending data. I also updated the length in close because the file could be opened again for writes.

Another variable I added was a boolean for if the file was closed called self.LPfile.closed and set it to False initially then only set it to True after the close function is called. In each function call I check if the file is closed before proceeding and raise a FileClosedError if it is.

Some other errors I raised include RepyArgumentError if provided offset is < 0, this is because you cannot read or write to a negative value in memory. I made sure RepyArgumentError was before FileClosedError or SeekPastEndOfFileError because apparently there is a hierarchy. I also checked if bytes < 0 in readat because reads cannot happen to negative amounts of memory.