**Vulnerabilities existing before fixing in assignment 2.3:**

- Raising exceptions like FileCloseError on attempting to close an already closed file.
    - In case a file is closed and another call is made to write to the file, a FileCloseError exception should be thrown. Hence adding a global variable as a flag to store the file status was helpful in checking the state. Initially, it is set to false and updated to true once the file.close() is called.

- Reference monitors are expected to check the offset values at writeat() and throw exceptions accordingly like RepyArgumentError (offset < 0) and SeekPastEndOfFileError (offset > EOF). In my case, a mistake was made on calculating the length of file on reopening an existing file again (now fixed).
    - To calculate the length of the file, two global variables were initialized to 0 and updated to the length of the file if it existed. One stores the length of the file including the pending offset and the other has the length that is written to the file. When a new writeat operation is done, the variable storing the original length of the file is updated and checked for the offset of the incoming request if its out of bounds and an exception is raised.

- The exceptions thrown should be as per the precedence like RepyArgumentError has higher precedence than FileClosedError.
    - In case a file is closed and a writeat(“data”, -1) is called, it should throw a RepyArgumentError instead of FileClosedError due to the precedence. RepyArgumentError > FileClosedError > SeekPastEndOfFileError

**Vulnerabilities existing before fixing in assignment 2.1:**

- Offset check was added as explained above in point 2. Both buff and file length variable values are updated accordingly in each function.

- To avoid race conditions where in 2 processes access a common function block together, a global lock is created for the object and acquired and released during the start and end of each function block.

- At writeat(), pending offset is to be checked before writing to the file as `None` type cannot be passed as an arg.

- Pending_data and pending_offset is set to None when undo() is called and updated to values passed in writeat when it is called.

- At close(), file has to be written with any pending_data (check if it is None).