1. The time required to copy a file varies with the size of the buffer because the buffer specifies how many bytes can be read in from the file at one time. After a while, though, the speed of the copy no longer increases. This is because the overhead of the read and write system calls is considerably less significant time-wise than reading from the disk, which is an unavoidable process. Disk I/O will eventually become the *bottleneck*.

   The memmap implementation varies much less with regards to how you do the copy. This is because this implementation makes much fewer copies. While the read_write solution copied the contents of the source file from kernel-space to user-space and then back to kernel space in the final destination file, the mmap solution avoids this by directly mapping the kernel-space buffers allocated for the source and destination files directly to the user-level process' virtual address space.

2. The size of the destination file changes with the size of the buffer size. This is because the entire buffer is added to the destination file on each *write* call. Even if there is blank space in the buffer, it will get added into the file. This can be solved by checking for an incomplete buffer and changing the size of the data being written to the destination file accordingly.