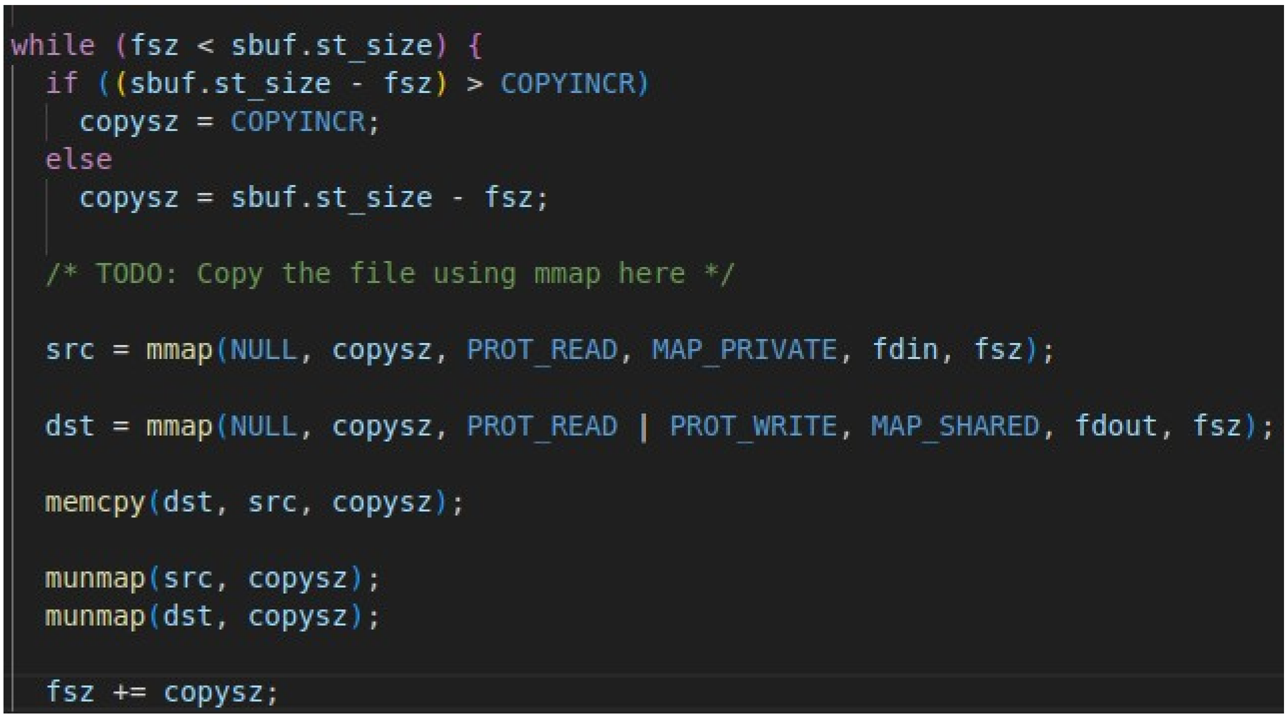
No. closing the file descriptor will not invalidate the memory-mapped I/O. The closure of file descriptor will not affect the data has already been mapped into memory. We can still access and operate the data in the memory-mapped area. The closure of file descriptor only disassociate from the file, but not affect the data has already been mapped into memory

First, we check whether fsz (file offset) is larger than sbuf.st_size (input file size). If fsz (file offset) smaller than sbuf.st_size (input file size) continue copy procedure, the copy procedure ends, and this means that the whole input file finishes the copy procedure to output file. Second, if input file size minus current file offset larger than COPYINCR set copysz to COPYINCR, set copysz to input file size minus current file offset. This means that we copy COPYINCR size from input file to output file each time; otherwise, copy the abundant content to output file. Third, we use mmap(2) to map source file and destination file to memory with copysz (the size of copy) and fsz (file offset). Using memcpy(3) copies src to dst with size of copysz. In the end, deallocate any mapping for the region starting at src and dst and extending copysz bytes and increase fsz by copysz.



```
while (fsz < sbuf.st_size) {
    if ((sbuf.st_size - fsz) > COPYINCR)
        copysz = COPYINCR;
    else
        copysz = sbuf.st_size - fsz;

    /* TODO: Copy the file using mmap here */

    src = mmap(NULL, copysz, PROT_READ, MAP_PRIVATE, fdin, fsz);

    dst = mmap(NULL, copysz, PROT_READ | PROT_WRITE, MAP_SHARED, fdout, fsz);

    memcpy(dst, src, copysz);

    munmap(src, copysz);
    munmap(dst, copysz);

    fsz += copysz;
```