# Assignment 10

(1) **(2 pt)** Complete the code and function correctly to copy a file, you can use source.txt to test it.

**Answer:** We use `mmap`, `memcpy` and `munmap` to copy the file. The size of memory required to b copied is automatically adjusted before we copy, so we need not do any extra calculation. `diff` shows that our program copies the data correctly, even when attempting to copy the `assignment12` binary itself.

(2) **(1 pt)** Try to close the input file after calling mmap and answer the following question in the report: Will closing the file descriptor invalidate the memory-mapped I/O?

**Answer:** We can close the file descriptor with `close` a. after the loop, and b. after `mmaping`, like so:

```
int err = close(fdin);
if(err != 0) puts("ErrIn");
err = close(fdout);
if(err != 0) puts("ErrOut");
```

The program behaviour remains unaffected:

```
$ ./assignment12 source.txt fd.txt
$ diff source.txt fd.txt
(no output)
```

Closing the file descriptor does not invalidate memory mapped I/O, and does not cause any residual errors as well. The descriptor is removed from association with our process but the file descriptor is still accessible.

(3) **(1 pt)** Describe your implementation in the report.

**Answer:** First, we get memory map pointers for the input file and output file. `MAP_SHARED` is used so that changes to memory at `src` and `dst` are reflected correctly, and so that the changes do not disappear after an unmap.

```
src = mmap(0, copysz, PROT_READ, MAP_SHARED, fdin, fsz);
dst = mmap(0, copysz, PROT_WRITE, MAP_SHARED, fdout, fsz);
```

The memory mapped areas are of size `copysz`, so we copy exactly that many bytes from source pointer location to destination pointer location.

```
memcpy(dst, src, copysz);
```

To prevent undesirable errors in later iterations, we unmap the memory areas. In many cases these steps can be ignored, since these regions will be unmapped when the process closes. However, if a larger file is used we may reach the number of mmap pointers we can create from one process.

```
munmap(dst, copysz);
munmap(src, copysz);
fsz += copysz;
```