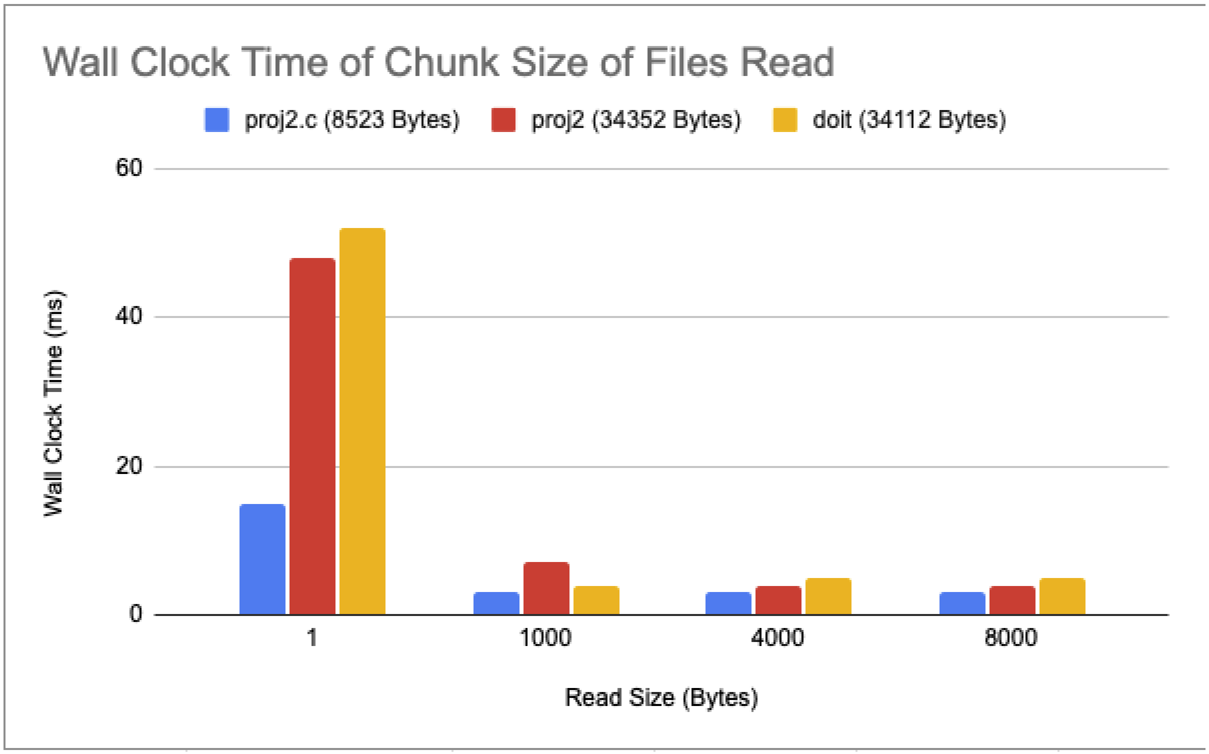
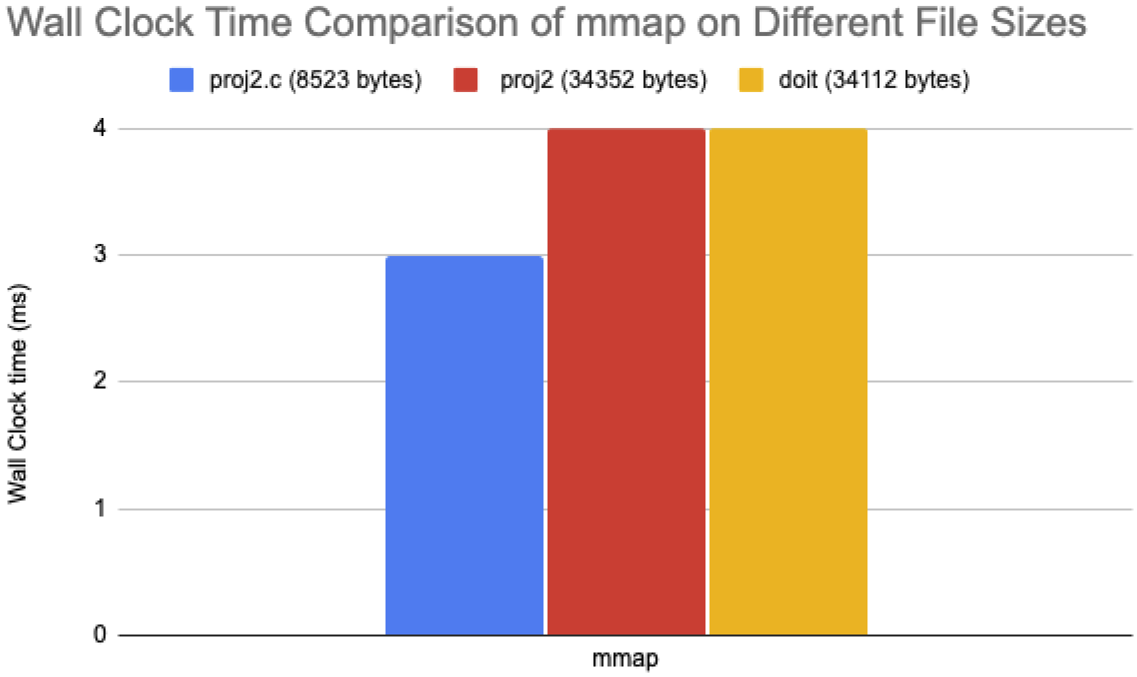
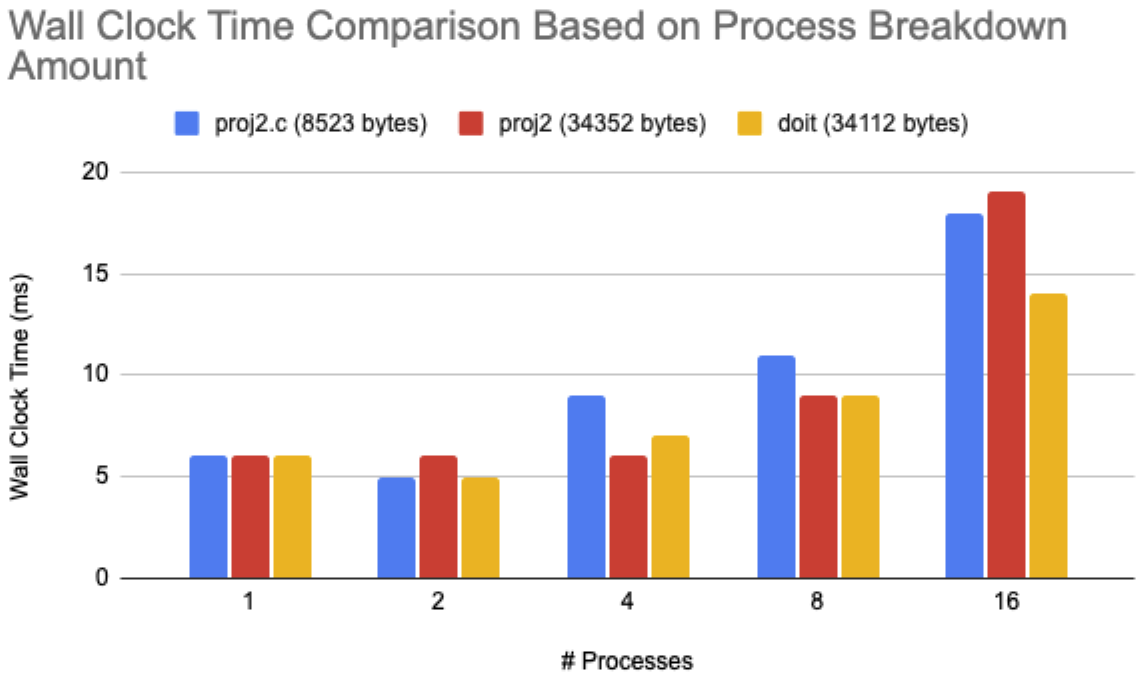

Wall Clock Time of Chunk Size of Files Read
proj2.c (8523 Bytes)
proj2 (34352 Bytes)
doit (34112 Bytes)
Wall Clock Time (ms)
60
40
20
0
1
1000
4000
8000
Read Size (Bytes)


Wall Clock Time Comparison of mmap on Different File Sizes
proj2.c (8523 bytes)
proj2 (34352 bytes)
doit (34112 bytes)
Wall Clock time (ms)
4
3
2
1
0
mmap



Environment - MacOS
It was clear that as chunk size for reading increased the wall clock time it took to process the overall file decreased. This showed that the closer the chunk size is to the overall size of the file the faster the file will be processed.

For memory mapping there was no real difference. While it did take less time to process the smaller file, as the size of the files increased there was little to no difference in the time it took to process them.

For the multiple processes portion of the assignment, it was clear that regardless of the file size, the more processes you tried to create the longer it took to finish the program. Maybe if we were creating and using threads it would be different however, since forking creates a brand new process, it is pointless to break down the file into child processes that each do their own part. The switching between the child and parent process, as well as the creation of the child process, is an unwise use of resources, even though they usually won't significantly affect the wall clock time.

In conclusion, as the chunk sizes increased, the wall clock time decreased, and as the number of processes increased, the wall clock time increased.