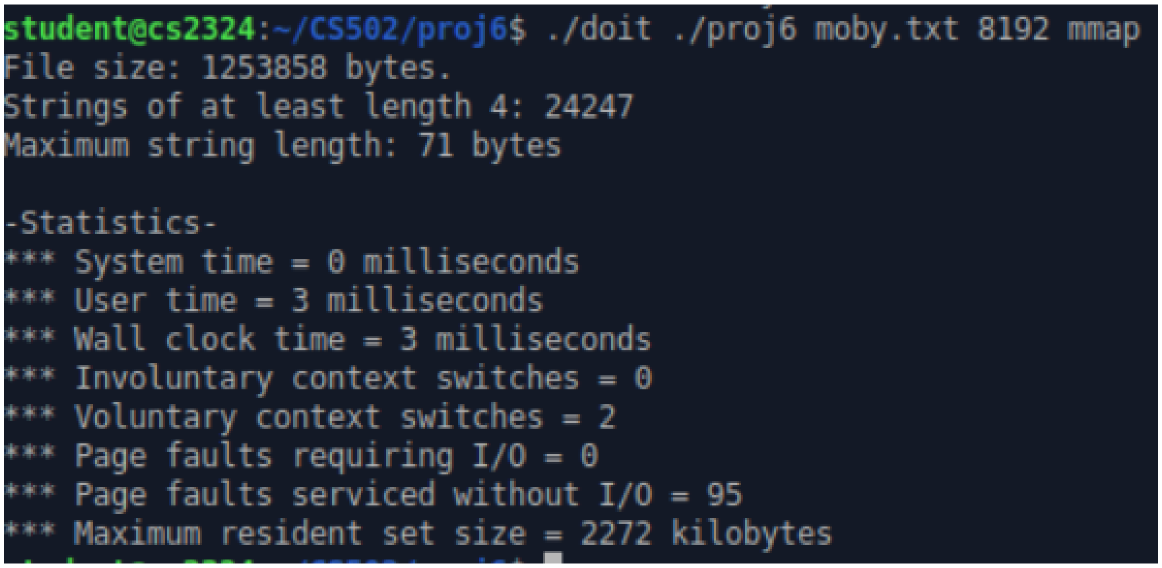
CS 502
Project 6 Writeup
Rakesh Veetekat

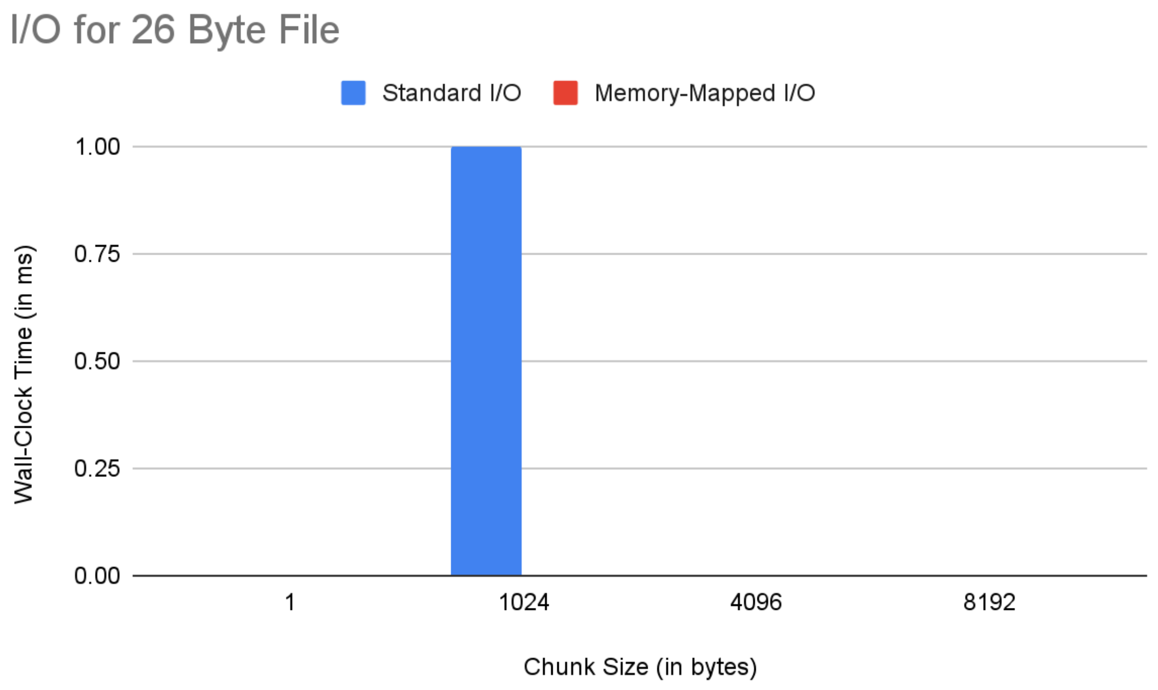
For my performance analysis, I gathered data on the major page faults and wall-clock time returned by my proj6 executable when run on 3 files of varying sizes. The sizes of these files are 26 bytes, 6254 bytes, and 1253858 bytes. I ran the program with both standard I/O as well as memory-mapped I/O, and I also changed the chunk size for each of these runs. Here's a screenshot of how I've run my program on each of the files:

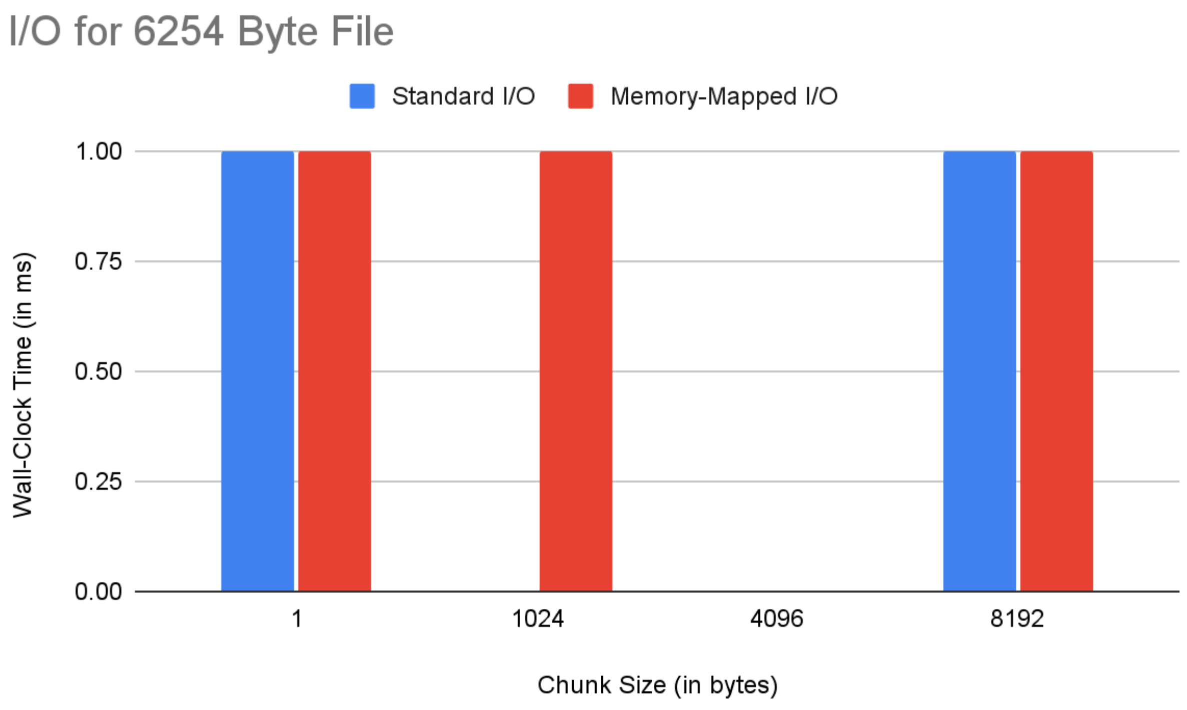


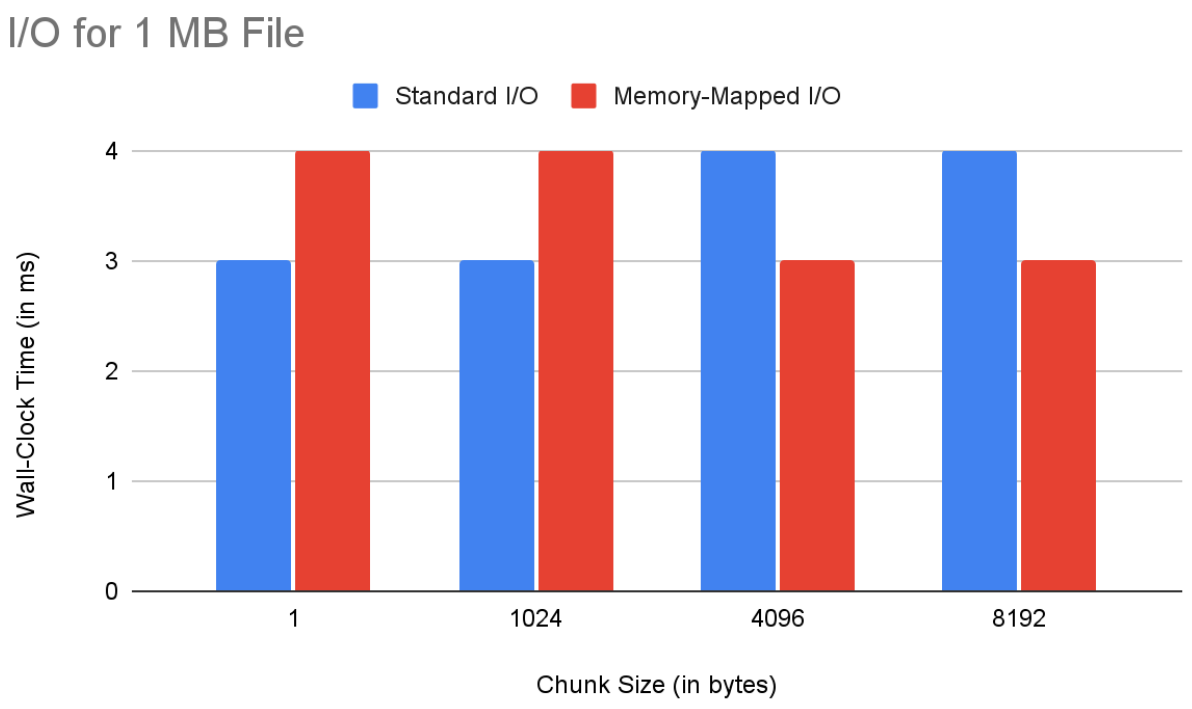
```
student@cs2324:~/CS502/proj6$ ./doit ./proj6 moby.txt 8192 mmap
File size: 1253858 bytes.
Strings of at least length 4: 24247
Maximum string length: 71 bytes

-Statistics-
*** System time = 0 milliseconds
*** User time = 3 milliseconds
*** Wall clock time = 3 milliseconds
*** Involuntary context switches = 0
*** Voluntary context switches = 2
*** Page faults requiring I/O = 0
*** Page faults serviced without I/O = 95
*** Maximum resident set size = 2272 kilobytes
```

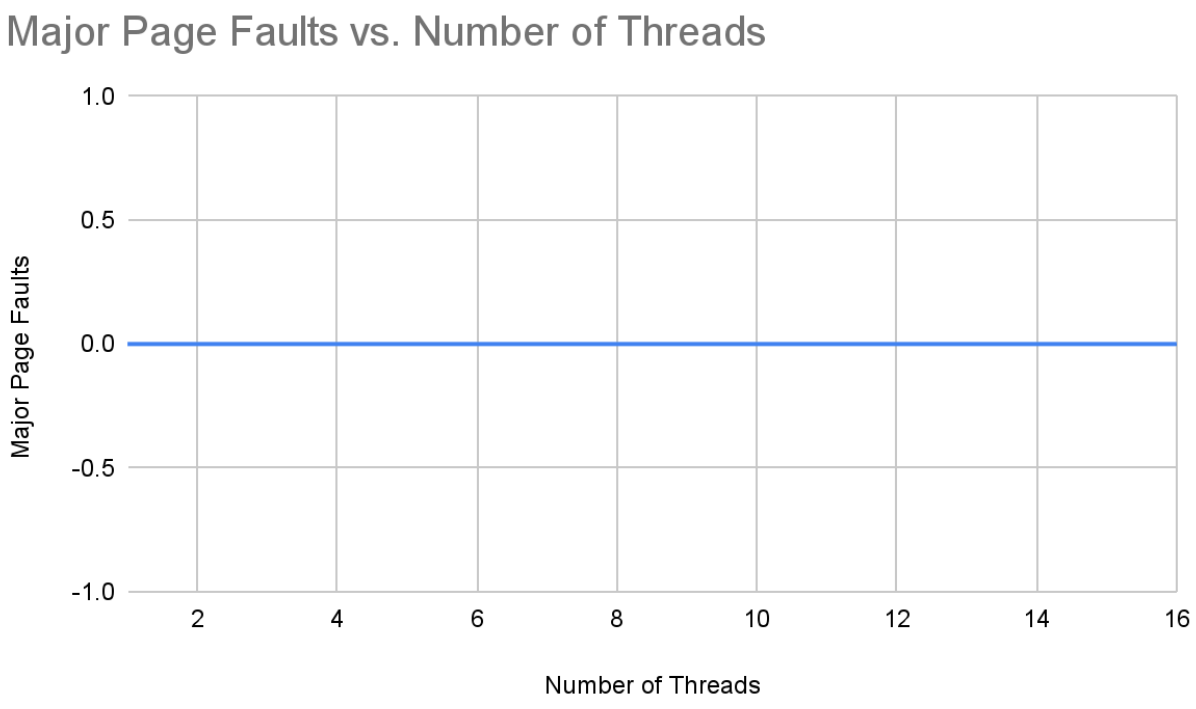
The graphs of the standard I/O vs memory-mapped I/O results are shown below:

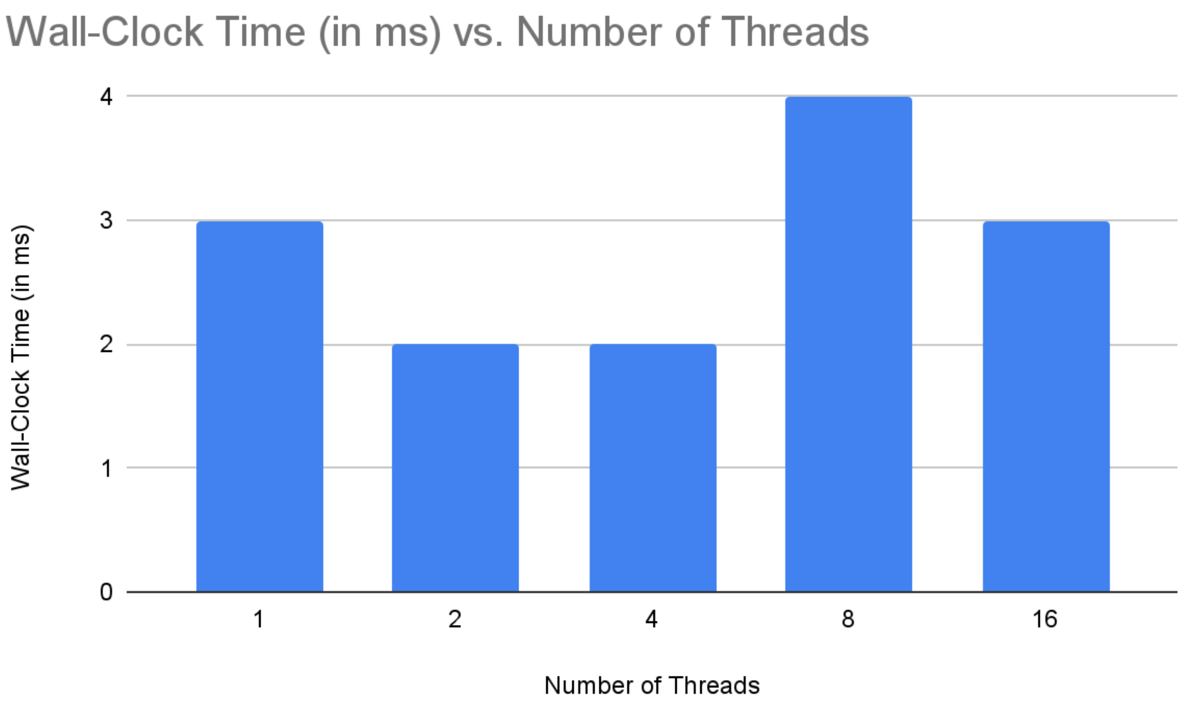





I/O for 1 MB File
Standard I/O
Memory-Mapped I/O
Wall-Clock Time (in ms)
0
1
2
3
4
1
1024
4096
8192
Chunk Size (in bytes)

To understand the performance of the different thread options, I’ve only run the program on the 1 MB file since it provides the most meaningful results. I’ve pasted the graph of these results below:





For all of my tests, my proj6 program never ran into any major page faults according to the statistics acquired from my doit program. The wall-clock time for both the 26 byte and 6254 byte files was always 0 or 1, which may be because of the severely small size of both files. As for the 1 MB file I tested with, the wall-clock time always hovered around 3 to 4 ms, and I think this may be because of the power of the system I am testing with.As for testing with the multi-threaded portion of my program, I wasn’t able to gain meaningful results in this area either. Although I don’t see much loss or improvement in performance for utilizing more or less threads in my tests, I believe that using higher thread counts for smaller files would probably increase the wall-clock time while higher thread counts for larger files would probably decrease the wall-clock time. My reasoning behind this is that the overhead required for more threads on a smaller file will probably decrease performance in that scenario, while increasing performance if used on a larger file.