Mert Ilgınoğlu, 26774

CS307 HW4 Report

In this report we are going to analyze and compare 3 different file reading methods, which 1 of them is in C++ language and 2 of them are in C language. The C++ method that we are going to discuss is fstream and the C methods are fopen and mmap.

mmap() is used for mapping between process address space and file or device. It returns a pointer to the location of virtual memory. Hence, it is the lowest level method amongst the others. In this method we don't read the data sequentially, we are mapping the data to main memory. As a result, this method has an advantage compared to the other ones if we are random accessing the file and not reading it sequentially. With mmap, the file's content can be accessed in an array. Disadvantage of this method is that memory mappings must fit into the process' address space. If we have 32-bit address space, this might not be enough for files with very large sizes. [1]

fopen() allocates a FILE object on the heap and initializes it with the amount of buffered data that hasn't been flushed yet (which is 0 at first). Following that it makes a call to the filesystem driver, then the driver opens the file and provides a handle to the file. Filesystem driver is dependent on the HDD driver to do the reads and writes to the HDD. A cache may also be allocated in RAM for the file. For example if a user requests 1 byte to read, it can be allocated to RAM to make the later reads faster. Then the FILE pointer ist returned.[2] As a result, this method is more high level compared to mmap and easier to handle for programmers compared to mmap. It is also good when we are trying to read a file sequentially and performs badly if we are trying to do random access on a file. Since we are reading from HDD, it is slower compared to mmap. In addition to that fopen is not exception safe, which is a disadvantage compared to fstream.

File reading in C++ using ifstream is similar to fopen in C. If the optimizations are open, there is no significant difference between running times of ifstream and fopen. Else ifstream is slower than fopen.[3] Also it is more high level compared to fopen, it is exception safe, is a better encapsulation and has higher level concepts. Also it is a stream and can be treated generically as a stream.[4]

In the flow server all 3 methods were tried. Among 3 methods, mmap is the one running fastest. It has a mean running time of 1.636 seconds. In the second place, we have fopen of the C language. It has a mean running time of 3.824 seconds. fstream is the slowest method amongst all, with mean of 4.516 seconds. I have also tried to run it with optimisations on but it did not change the results. This is a %18 percent difference compared to fopen. It might have been because of a problem in the flow server. Results are given in the table below.

[1] https://stackoverflow.com/questions/258091/when-should-i-use-mmap-for-file-access
[2] https://stackoverflow.com/a/5130577/10922395
[3] https://stackoverflow.com/a/17468320/10922395
[4] https://stackoverflow.com/questions/5570361/file-vs-fstream

	Mean	**Standard Deviation**	**Run 1 (seconds)**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
fstream	4.516	0.013	4.51	4.54	4.51	4.51	4.51
fopen	3.824	0.005	3.83	3.82	3.83	3.82	3.82
mmap	1.636	0.013	1.62	1.65	1.65	1.63	1.63

Since our file is 270 Megabytes, we can see differences in running time of the algorithms. However, if it was a very small text file, running time differences of the algorithms would had been negligible.

To sum up, if we are experienced enough so that we can handle the low level, we want to do random access on the file and our file is not so big, we should use mmap to read files. For sequential reads, we can use any of the methods, although I would have chosen fopen or fstream since they are higher level compared to mmap. If file size is very large that our page size can't handle, then we have no other choice than fopen or fstream. Between fopen and fstream, if the running times are the same, I would have preferred to use fstream since it has extra security measures for errors. If running times are not the same as if it were in our case, then I would have chosen fopen.