CS307 HW4 Report – İlker Gül 26352

In this homework we have written 3 different code 2 of them in C and one of them in C++. In two of the code files, we did not use the technique of memory mapping. We just open the text file that we need to read to count the total occurrence of the character ‘a’ in it. In each code file execution, we get the same number for total occurrence of ‘a’ which is equal to 19082160.

The text file(“loremipsum.txt”) approximately has a 275 MB size. Therefore, I was expecting that in each code file the iteration over the characters of the whole file would take more than 1 second. After implementing each file my expectation was correct, because the size of text file was impressively large.

In each of the executions after printing the total occurrences, I also wanted to print out the total time that taken for the execution.

- In CPP file, it was 5.67 seconds
- In C file, it was 3.61 seconds
- In memory mapped C file, it was 1.40 seconds

The time that, I am giving as a result might change in a small interval. To clarify, , in some executions I get 3.59, 3.60 seconds for the execution of the C file. In my last test, I get 3.61 seconds. Hence, the results may vary but will be very close to each other.

In CPP file, I used ifstream to open the file. Then, I used get() function to read the characters one by one to count the occurrence of ‘a’.
In C file, I used fopen() in read mode. Then, I used getc() function to read the characters of the text file one by one to count the total occurrence of ‘a’
In memory mapped C file, I use mmap() function to map the file to access like an array by also specifying its size. Then, since I do not need to read with a getc() function but access the file as an array, I iterated over the indexes of the array one by one to count the occurrence of ‘a’.

For time efficiency, since the size of the file is approximately 275 MB, I think the total execution time for each file was not so bad. Having approximately 5.67 seconds(if I add eof() check in the conditions of for loop it would take 6.83 seconds) should be improved(it can be

improved by reading it line by line, but to have the same approach as in C, I am reading char by char) but when we use the same technique in C, we get 3.58 seconds. Thus, without having memory mapping C performed better than CPP. It can be an indicator to select between languages when we need to perform reading files. It is most probably related to language design, functionality of the language and compiler. Moreover, as far as I have concerned, when we get bigger files than loremipsum.txt the time for execution will also increase. Therefore, we need to find a technique to decrease the time consumption for abstraction of reading, we should find a way to access the file with a less costly technique.

Memory mapping is actually a great performance saver. We can also see it in the results. We are mapping a file into a memory location by this technique, so we do not need to make system calls to open the file in the system. Since we also mapped the file in memory, we also performed paging. This gives us a prevention of unnecessary kernel/user mode switches and an ability to use the advantage of paging. Since we can easily access the file in memory and do not need to system calls for opening and reading file, the advantage of access gives us less execution time. So, we can see it by having total 1.40 second execution to count the occurrence.

It is clear that, when we have big files like loremipsum.txt or bigger than it, to easily access the file and have less time computation, we should perform memory mapping. Furthermore, it is also easy to understand and implement. However, we need to be careful about usage protection and flag bits. If we do not implement the necessary conditions, it can cause unwanted errors. Also, as I have stated before we use paging in mmap(). Hence, when the file size is smaller than the page, it will cause wasted memory space which should be prevented.

In conclusion, memory mapping is very useful when we have large files. Reading files by necessary function for each language will also give us the wanted result. However, time is a precious value for performance. Hence, we must concentrate on how to increase performance and choose memory mapping for our case and try to research and find further developments that can be implemented to decrease time for execution to have a better performance.

Yours Sincerely
İlker Gül