Bilkent University

Department of Computer Engineering

CS - 342 Operating Systems

Project 4

4 January 2020

Name: Osman Burak INTISAH
Id: 21602430

Name: Rumeysa OZAYDIN
Id: 21601558

Report

The structure of the entities we provide is as follows:

- Superblock is kept in block 0 in order to store the empty block number. Since the block size is 1024 its entities should be 1024 B too. To make the size 1024 B, a dummy array is kept in the struct of the superblock.
- A struct called fileInfo is created in order to store the file information such as name, being used, mode, size and head. Also, a dummy array is kept to make the struct 128B since we only allowed to store 8 files in a block of size 1024 B. An array of 'fileInfo' is created to store in blocks.
- A struct called fileDes is created to store the entities of a file allocation table. The size of the struct is 8 B and a block can contain 128 'fileDes'. An array of 'fileDes' is created to store in blocks
- A struct called opened is created to store the information of an open file. It keeps the name, the file descriptor, and the offset. Offset is stored to manage the reading of the file. A global array of 'opened' is created to keep track of the opened files.

In the sfs_format function, all the arrays explained above are initialized to their default values and written into the disk according to the clarifications of the project.

In the sfs_read function, we are reading whole the data from the data segments and give back to the buffer from the beginning of the offset. And its size is equal to n, however, if this n is larger than the data remained in the data segment, we are just reading that remaining data and returning this remaining number of bytes.

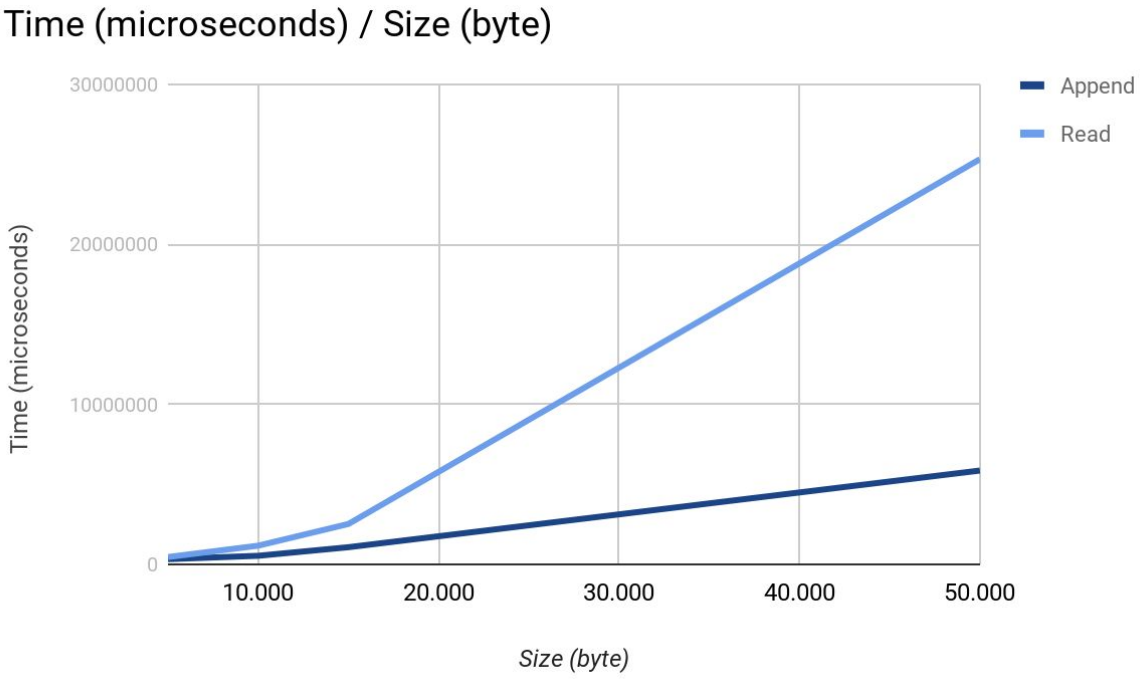
Stressing the Library

SIZE (byte)	TIME (microseconds)
5.000	326143 µs
10.000	536744 µs
15.000	1073496 µs
50.000	5863349 µs

Table 1: Results of append function stress

SIZE (byte)	TIME (microseconds)
5.000	459532 µs
10.000	1169387 µs
15.000	2526397 µs
50.000	25325117 µs

Table 2: Results of read function stress



Graph 1: Comparing append and dead

As it can be seen in the plot, the read function takes much more than the append function. Since all the read operations are continues and starts reading from the remaining unread part, an offset is kept to keep track of this. In the read function, first, we are reading the whole data and then, return the value from the offset. Reading the whole data is the reason why read function takes more time.

File Count	Time (microseconds)
5	1075 µs
25	5937 µs
50	13265 µs

Table 3: Results of create function stress