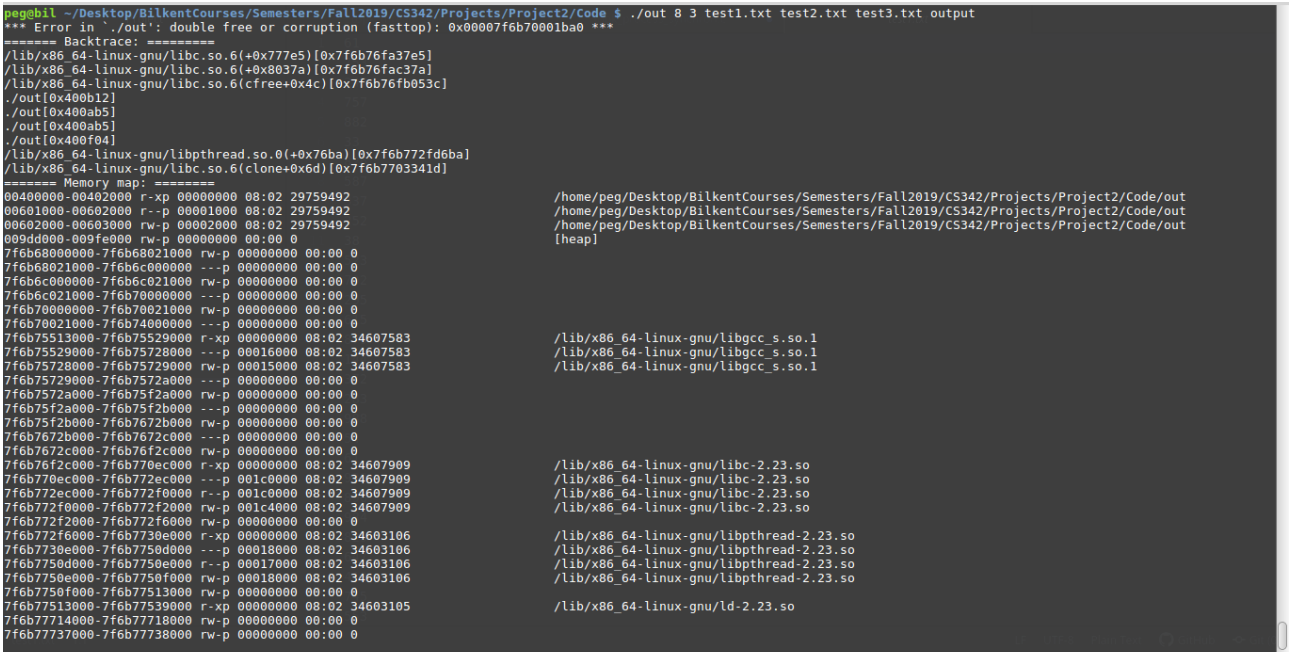
In the first part of the project I remove the critical path from our code. In other words, I removed the mutex locks from the code, up until two threads the code works and I saw an output. However, when I increment the number of threads I get an error saying :double free or corruption (fasttop): 0x00007f6b70001ba0 ***. Below is attached the screen shot of the terminal screen. This error must be due to each node trying to access the tree and insert and remove from the tree at the same time. Handling it with two number of threads might be possible but the more threads we have the more complicated the code gets and the more probable we are in getting errors as such.



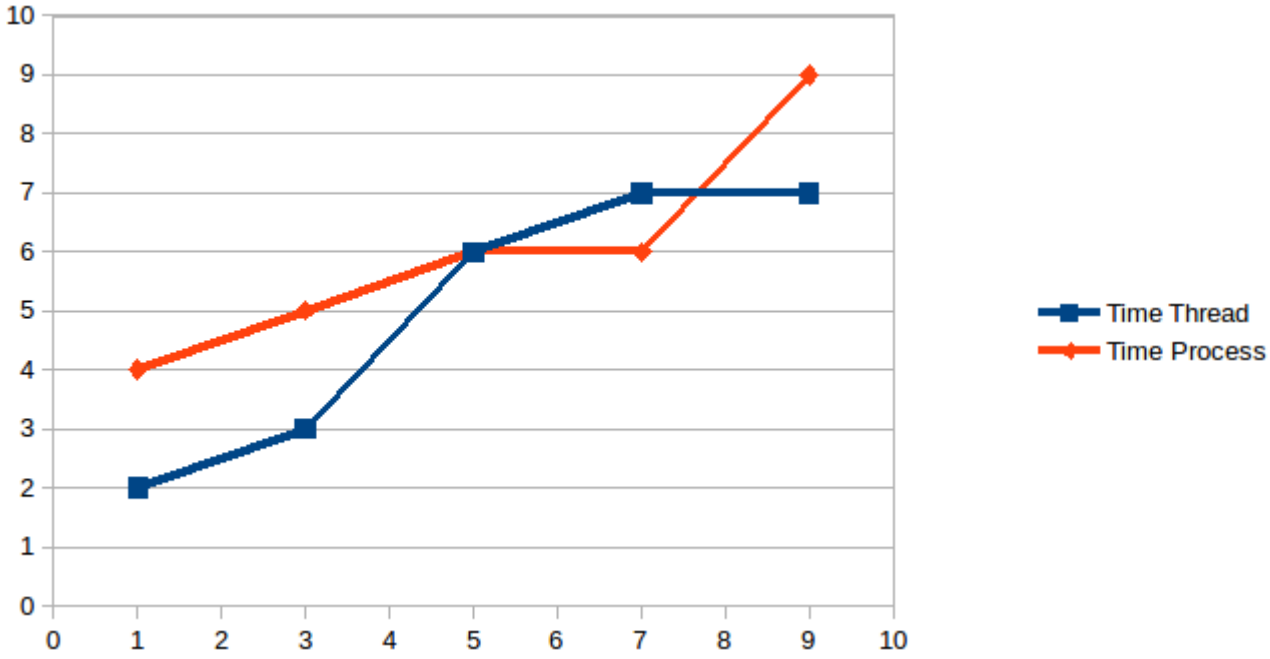
Removing the semaphore from the second part of the project does not result in an error. However, the result obtained from the program is not correct and different from the result that I obtained from the code when I had the semaphore locks on. We do not free any objects in this second part we are simply using arrays so inserting and removing in the array should not cause any errors. But of course the result is wrong.

Analyzing time for topk_process_synchron:

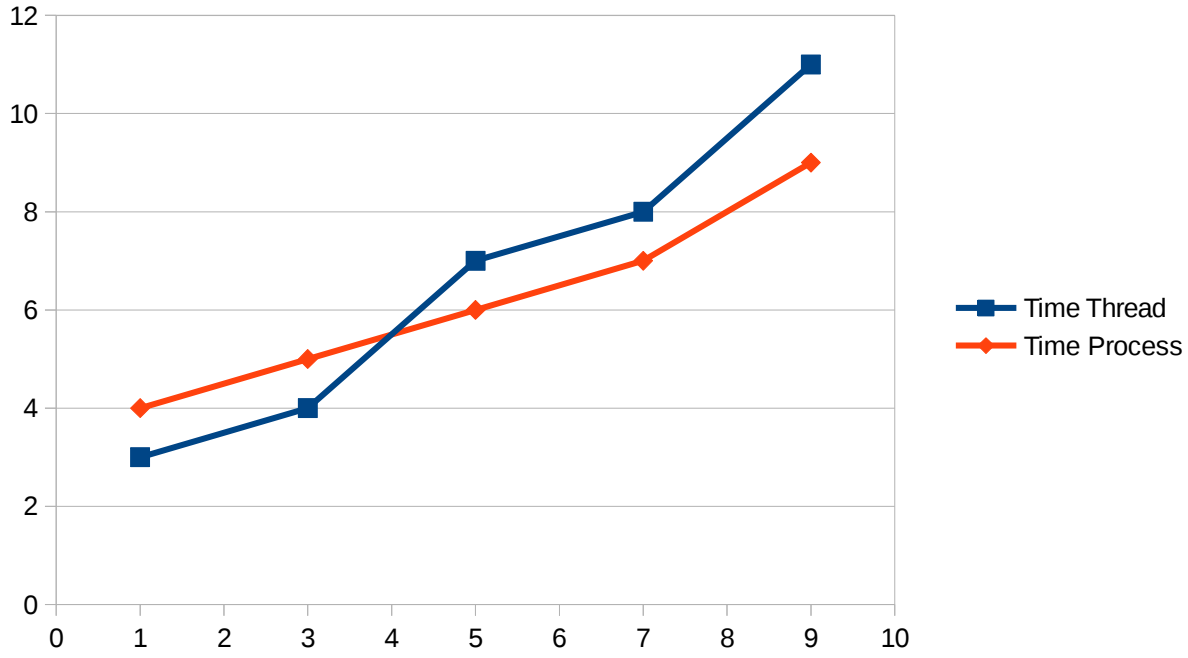
N	K	Running Time in ms Threads	Running Time in ms Processes
1	100	2	4
3	100	3	5
5	100	6	6

7	100	7	6
9	100	7	9
1	200	3	4
3	200	4	5
5	200	7	6
7	200	8	7
9	200	11	9

The graph below demonstrates a comparison between the running time of the processes and the threads. There might be some errors in the calculations but from the overall view shows that the more files we have to process the slower the program gets both in threads and processes. In this graph the K value is fixed which is 100.



The second graph below shows the same information as above but this time the K value has changed from 100 to 200. In this case we can see that after a certain point processes are faster than threads when they get same amount of input.



As a conclusion, we can see from the data in the table and the graphs that using processes when the number of files are increasing will leave us with a more efficient and fast result. Additionally, I do not reckon that this comparison will give us sufficient and accurate information since the task of the threads and the processes are not the same. In the thread part we implemented a binary tree and did tree operations on the files. However, we created an array of integers in the second part of the project which is relevant to the processes. Hence I do not think that these numbers will give us consistent information regarding threads and processes.