**Brian Jimenez**

**1001388904**

**Part 3: Evaluation of report**

* The whole point of the experiment is for us to learn more about how multiple process works. This comes along with how fork and the exec functions work. This part of the lab was more practice learning how to use fork and the whole purpose of using them. Another reason is that we also need to be able to use the exec function. This was the whole point of doing part one in my opinion. In terms of the lab, the point of creating mandelseries was to create multiple pictures of the same image but zoomed in more. We had to create a total of 50 images. Not only that but the user will provide with how many pictures they want to see at a time. By doing that, we need to implement forks and exec.

The way to execute mandelseries is to do this:

ex: ./mandelseries n (n=number of process you want to run at a time)

By doing this, instead of running one pic at a time you can do more pics at a time. The n is what the user will type as to the number of processes they want to run simultaneously. If no n is given, the program will not run because it will not have any number of process for it to run so it will crash. We need to have the number of processes. It will keep producing at that rate until it gets to 50. If for some reason the number of process is not divisible by 50, then it will run at the same pattern you implement until that pattern is about go over 50 and then it will run one by one until it gets to 50. For example this will be example of how it will do it.

Ex: ./mandelseries 10 ======== 10(images will first appear), then another 10 will be produced, will keep happening until get to 50

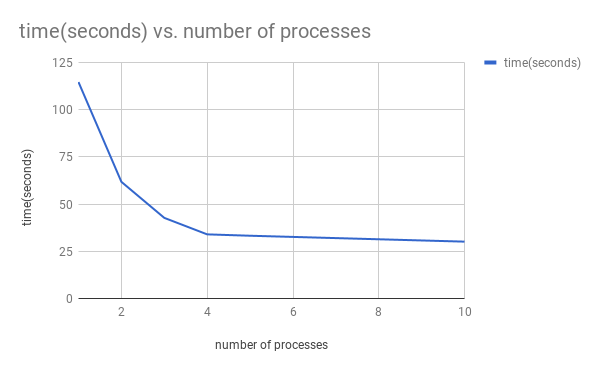
Ex: ./mandelseries 15 =========(15) will first appear, then another 15(30 total), then another 15 (45), then 5 will run but one by one.

* The whole point of part 2 is to do multiple threads. We had never worked with threads, so this was a great example of how we could implement threads. In terms of the project, we need to find a faster way to produce one image. In order to do that, we have to we have to use threads. The threads will divide and conquer. An example of running this is:

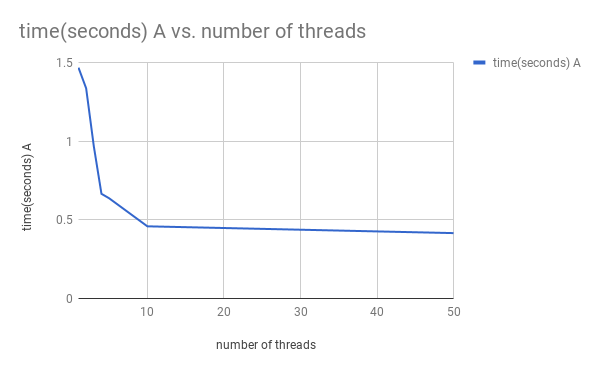
Ex: ./mandel – n 5 -x 0.286932 -y 0.014287 -s .0005 -m 1000\n\n

In the example above, the –n and the number after that is to determine how many threads you want, the rest is all information need to create the image. If no n is provided, the threads will be 1.

* The following is a graph that illustrates all the information illustrating the graph of 1,2,3,4,5,10 process running for mandelseries. All the information of the picture that I used is:
* a="-x 0.286932 –y 0.014287 –s 2 –m 2000 –W 1024 11 –H 1024 –o mandel3.bmp” with S changing every picture ;

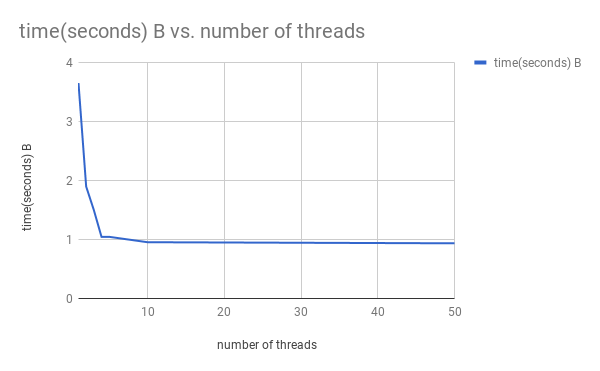


* The shape the curve is going down as the number of processes go up. Based on the chart and the data, the more process you have, the faster and less time it will take to produce 50 images with different zoom. I feel like the best more optimal number of process is having 5 process running at the same time. The reason being is because the transition from 5 to 10 process is not that much different. Only about 3 seconds of difference. I feel as you increase the process, the time will continue to go down, but it won’t be as drastic as going from one process to two process or two to three. Those were the biggest change in time. Going from 10 process to 50 process is only about a 2 second difference. So, based on that the more process you have the faster it is, but it won’t be that much of a difference after 5 process running simultaneously. As far to having too many process, I think if you have A LOT of process, then yes it would affect in the way that each process will in a way slow down the other ones. They might get in each other’s ways. For the case of the project, as the number of processes go up, the time decreases, which is good.
* The graph below is a graph of the threads working with A: mandel -x -.5 -y .5 -s 1 -m 2000



The shape of the curve is a negative slope. As the number of threads increases, the time in seconds decreases. Based on the data and the graph, you could say that the optimal number of threads is 10 threads. The reason being is because at 10 threads, the time drastically decreased compared to the other previous amount of threads. Going on to 50 threads, the time seems to not change much, so the most optimal number of threads is 10.

The graph below is a graph of the threads working with B: mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000



The graph and date below show that it takes longer to produce this image with one thread. As the number of threads increase, the time in seconds decrease. This graph illustrates a downward trend. The time is affected by the number of threads. I would say that the best optimal number of threads is 2 to 3 threads. Based on my number that is when the time drastically changed and after that the time still decreases, but at a slower rate.

* The reason why A and B create a different shape is because B has W and H as 1024 which increases the size of the picture and x and y are more specific than A which is he standard one. They are similar in the way that they both have a negative slope. The only thing that is different is because of the resolution so that’s what made it different.

A combination of the two types of parameters are here for comparison:

