|  |
| --- |
| **Project 1: Fork and Pipe** |
| CSCI 474: Operating Systems Concept |
|  |
| The summary report about the fork and pipe project for the course of CSCI 474: Operating Systems Concept |
|  |
| **Akshat Sharma** |
| **10/28/2013** |
|  |

**1.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | File 1 | File 2 | File 3 |
| 1 Process | 0.001127 seconds | 0.003140 seconds | 0.013463 seconds |
| 2 Processes | 0.000589 seconds | 0.001694 seconds | 0.005486 seconds |
| 4 Processes | 0.000507 seconds | 0.001135 seconds | 0.003200 seconds |

**2**. In this project, fork function was used to create child processes so that the numbers in the file could be processed in parallel. The fork function was called depending on the number of child processes to be created.

**FILE 1:**   
 As we can see, from the above table, when the file 1 was processed by using just a single process, when compared to multiple processes, it took almost the double amount of time as it took when multiprocessing was used to process the file. File 1 when ran with a single process took 0.001127 seconds. This happened as the File 1 contained 1000 numbers in it, and when single processing was used, just a single process acted on those 1000 numbers. This led to the time consumption. As soon as 2 or more processes were used, the file got split up, and the numbers were divided to be assigned to different child processes, which then ran parallel to each other thus reducing the time to complete the whole task at least by half. When 2 processes were used, each child process handled 500 numbers are processed them in parallel. File 1 with 2 processes used 0.000589 seconds of execution time. Similarly, when 4 processes were used, each child processes handled 250 numbers from file 1 and then 4 child processes processed in parallel, thus reducing the execution time even more as file 1 execution using 4 processes took 0.000507 seconds.

**FILE 2:**  
 Similar was the case in File 2, which contained 10,000 numbers. The File 2 took 0.003140 seconds of execution time when using a single process. But when the user selection was 2 Processes for File 2, the file was divided into sections of 5000 numbers, each handled by the 2 child processes, thus reducing the time of execution to 0.001694 seconds. Further, when 4 child processes executed File 2 simultaneously, each child process handled 2500 numbers, and the time of execution reduced even more to 0.001135 seconds.

**FILE 3:**  
 Next, when File 3 is processed using a single process, the time of execution is 0.013463 seconds to process 100,000 numbers present in the File 3. Whereas, as soon as multiprocessing is introduced in execution of File 3, we can see the time of execution again go down by nearly one-half. When 2 child processes are used, each process takes 50,000 numbers from the file and the two processes process in parallel. The time of execution in this case is reduced to 0.05486. Further, when 4 child processes were used, each child handled 25,000 numbers in processed them in parallel, which reduced the time of execution even more to 0.003200 seconds

**3.** This project helped me gain knowledge on multiprocessing using fork and pipes. Using the fork function call, we create child processes, and every time a fork function is called, a new child process is created. With this program, I learned how effectively creating child processes using fork function call helps in reducing the time of execution of a program drastically.

Further, I learnt how to use multiple pipes to send data from parent to child and from child to parent simultaneously. I also gained knowledge on time functions in C, since we had to take out the time of execution for the program.

This project also made me learn many file operation functions. I gained knowledge on reading the file to get the total amount of lines present in the file. I also gained knowledge on certain file functions such as the fgets(), fseek(), and fgetc(). It was interesting to learn how a dat file contained a total of 5 characters and thus, by using fseek(), the pointer should be offset by 5 characters. I also learned a new programming language called C!

REFERENCES:

http://www.tutorialspoint.com/c\_standard\_library/c\_function\_fgets.htm

http://www.tutorialspoint.com/c\_standard\_library/c\_function\_fseek.htm

http://www.dreamincode.net/forums/topic/195643-runtime-with-fork-and-pipe/

http://www.qnx.com/developers/docs/6.3.2/neutrino/lib\_ref/c/clock\_gettime.html