**Part – 1**

**2. (15 Points) Provide following:**

**(a) Describe the parent/child process synchronization approach implemented in the**

**program.**

The parent/child process is synchronized using the fork() & pipe() to execute the main program’s function. In part 1 of the program, child process 1 executes the first line of the .txt file and child process 2 executes the second line of txt. Then both child processes send data to the parent process.

**(b) Describe the inter-process communication (IPC) approach you have selected to**

**exchange data (both input and output) among parent and child processes. Provide**

**reasoning for your selection.**

The Inter Process Communication(IPC) was a cooperative approach between two child’s and one parent process. Using direct Communication in this program using ordinary pipes. Pipes from child processes sent messages to the parent process.

**(c) Briefly describe a different inter-process communication approach that can be used in**

**this program. Provide reasoning for NOT implementing it in current program.**

Another method of Inter process communication is shared memory. A disadvantage of shared memory is the need to synchronize access. I felt that counting alphabets and printing their count through pipes was faster and easier. The main function would initialize the data, then the child would count the alphabetical chars. Finally, the parent would add both child counts that were sent and output total alphabet char count.

**Part – 2**

**2. (15 Points) Provide following:**

**(a) Describe the parent/child thread synchronization approach implemented in the**

**program.**

The parent child thread synchronization approach in this program is user threads. User threads include pthreads which are Linux only threads. Threads are responsive, share resources, and scalable.

**(b) Describe the approach you have selected to exchange data (both input and output)**

**among parent and child threads. Provide reasoning for your selection.**

The approach I used in part II threads was creating two void functions. One function would process the input line and the other process would count the total alphabetic chars in a txt file. These functions would send the data to the new and child thread. These threads would then execute the functions and output their information. The reason I chose this approach was because this would separate the functions, meaning the threads would independently fetch data hence making them faster.

**Part – 3**

**1. Compare and contrast the two implementations: Part – 1 (process-based) and Part – 2**

**(Thread-based). Is one approach (always) preferable over other? Provide reasoning for**

**your views.**

Processes are single addressed spaced. Meaning that only one global address space per process. While threads provide memory sharing. Threads also separate notion of execution from the rest of the definition of a process. This in turn makes threads more efficient while synchronizing between threads. This is the reason why in this certain program threads are preferable than processes.