Name: Vishal Vijay Devadiga

Roll Number: CS21BTECH11061

Code Flow

- Opens input file in main and gets values of n,k
- Passes n,k to the solve function
- Create k child processes that execute the childsolve function. Parent stores the PID of the children in an array.
  - Child process creates shared memory and opens a log file to print the output of its calculations
  - The child process finds perfect numbers among the numbers assigned by the formula: $i \times \frac{n}{k} + procNo$ where i is the i^{th} iteration of the loop in the child process and procNo is the number of the process(Not PID).
  - Child process then prints the perfect numbers found to the shared memory.
- While the child process are executing the childsolve function at their pace, parent process is in a loop that waits for a specific child process(identified by the PID stored)
- If that child finishes executing(exits), then parent process accesses the shared memory created by the child process and then reads all the perfect numbers produced by the child.
- The parent process prints this to the output file, and repeats the process for all of the children.

Reason for some decisions

The child process is creating the shared memory. In my code, parent does not write to the shared memory, only reads from it. Parent process does not waste time creating shared memory, instead now goes forward in the loop and creates more child processes, saving some time, as child processes create the shared memory in that saved time.
In the end of the iteration, the parent process unlinks the SHM of that child process.

The allocation of numbers to calculate is based on the formula $i \times \frac{n}{k} + procNo$ where i^{th} iteration of the loop in the child process and procNo is the number of the process(Not PID). This balances the load between the processes, compared to the sequential allocation(1 to $\frac{n}{k}$ for process 1, $\frac{n}{k} + 1$ to $\frac{2 \times n}{k}$ for process 2 and so on).
The producer-consumer like queue for the processes looked good for some small amount of child processes and balanced the load even more. However, parent process bottlenecks the calculations when the number of processes produced were a lot more(depends on the system)

Some Observations

Printing the log files takes the most amount of time in the program. The actual program without printing the log files takes a lot less time.
Increasing the number of processes after a certain amount actually increased the execution time of the program.