Task 1 [15 minutes]

Create a program that launches multiple child processes and monitors their execution. Use pipes to collect and display information about the processes, such as their start and end times. Every child process will write its start and end time to the pipe and the parent process will print this information along with printing the most CPU bound process.

Task 2 [20 minutes]

Consider a scenario where two types of processes, named “P”, and “C”, share a common, finite-sized buffer (or queue) as a means of communication. The challenge is to ensure that “P” does not produce items into the buffer when it's full and that “C” does not consume items from the buffer when it's empty. Can this issue occur if we use pipes? Use code to answer the question.

Task 3 [25 minutes]

Implement a sorting algorithm (merge sort) using multiple processes and pipes for merging sorted subarrays. The parent process creates three child processes, each process will sort 1/3rd of the array and then will write the sorted array to pipe. The parent process will collect the sorted sub-arrays sent by the child processes and will print the final sorted array.

Task 4 [30 minutes]

Write a program that builds a chat application between two processes using pipes. You have to create a child process. Both the processes will read and write data through pipes. The communication should stop as soon as any of the processes has written “bye” in the pipe.