**Task 2**: First-Come, First-Serve (FCFS) Scheduler Simulation

**Task Description:**

The task involves simulating a simple First-Come, First-Serve (FCFS) process scheduler in **C**. The program should allow input of process details and display key scheduling outputs.

* Input Requirements:
  + Number of processes to be scheduled.
  + Each process's arrival time and burst time.
* Output Requirements:
* A Gantt chart (or equivalent representation) showing the scheduling of processes.
* “Waiting time” and “Turnaround time” for each process.
* “Average waiting time” and “Turnaround time” across all processes.

**Libraries/Header Files Used**:

* **stdio.h**: For standard input and output functions like printf**()** and scanf**()**

**Detailed Explanation**:

* **Struct Process definition**:
* A structure **Process** is defined to hold process-specific data:
  + int pid: Process ID, a unique identifier.
  + int arrivalTime: stores time at which the process arrives in the queue.
  + int burstTime: stores the duration of the process’s execution.
  + int waitingTime: stores the value calculated as turnaroundTime -burstTime.
  + int turnaroundTime: stores the value calculated as completionTime -arrivalTime.
  + int completionTime: Stores the time when the process finishes execution at runtime.
* **Input**:
  + - The user has to choose between a custom sequence of process or using a predefined example to apply FCFS algorithm.

A screenshot of a computer program

Description automatically generated

* + - If the user wishes to provide custom sequence of processes then he must:
      * 1. Provide Number of Processes
        2. Arrival time for each process
        3. Burst time for each process

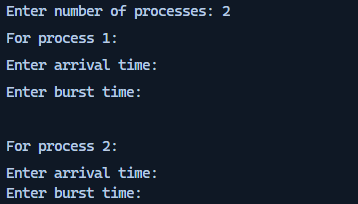


Figure 1: Custom Sequence of Processes

* **Functionality**:

1. **swap()**:
   * The program calls swap**()** function and based on the arrivalTime of the processes, the processes are sorted in ascending order with respect to their arrival times using a nested loop.
2. **FCFS\_Scheduling()**:
   * The function FCFS\_Scheduling() iterates through the processes[] and calculates the metrics based on the constraints of the FCFS:
     + **Waiting Time**: Time a process spends waiting in the queue of the processes[].
     + **Turnaround Time**: Total time of each process from its arrival to completion.
     + **Completion Time**: Calculated incrementally using currentTime.
   * Updates avgWaitingTime and avgTurnAroundTime to compute averages for the all the processes in the processes[].
3. **DisplayResults()**:
   * DisplayResults() outputs a formatted table showing all relevant process data.
   * Average Waiting Time and Turnaround Time are displayed as floating-point values.
4. **GanttChart()**:
   * The function GanttChart() visually represents process execution, highlighting their IDs and respective times on a timeline.
5. **CustomProcesses() and ExampelProcesses():**
   * CustomProcesses(): Prompts the user for process details ( Includes the number of processes, arrival time and burst time for each process). *Validates input to ensure at least two processes are entered*.
   * ExampleProcesses(): Uses a predefined set of processes for simulation. Has a set of 3 Processes with predefined metrics.

* **Program Flow**:
* The user selects is provided a menu to select an option to define custom processes or use predefined data for simulation of FCFS Scheduling.
* The program invokes appropriate functions to schedule processes, calculate metrics, and display results.
* Same functions will be called for both Predefined sequence of processes and Custom sequence of processes
* A formatted table showing all relevant process data is printed after the FCFS algorithm is applied
* The Gantt chart is also drawn which provides a visual representation of the scheduling.
* Average waiting and turnaround times are computed and displayed.

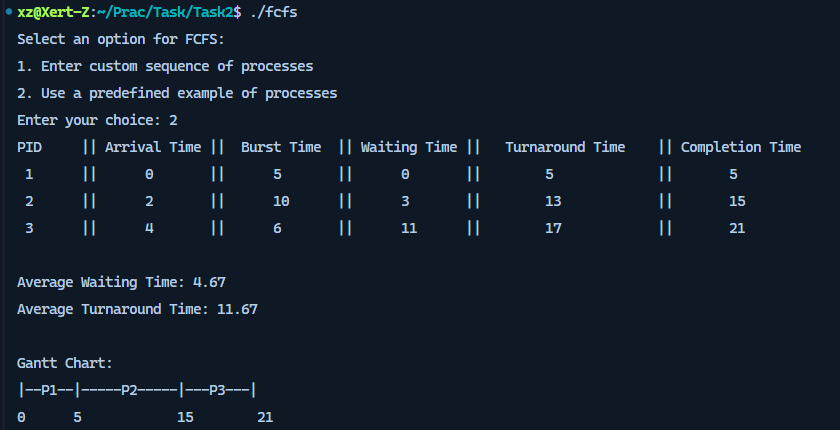


Figure 2 Complete Execution of FCFS.c using Predefined sequence of processes

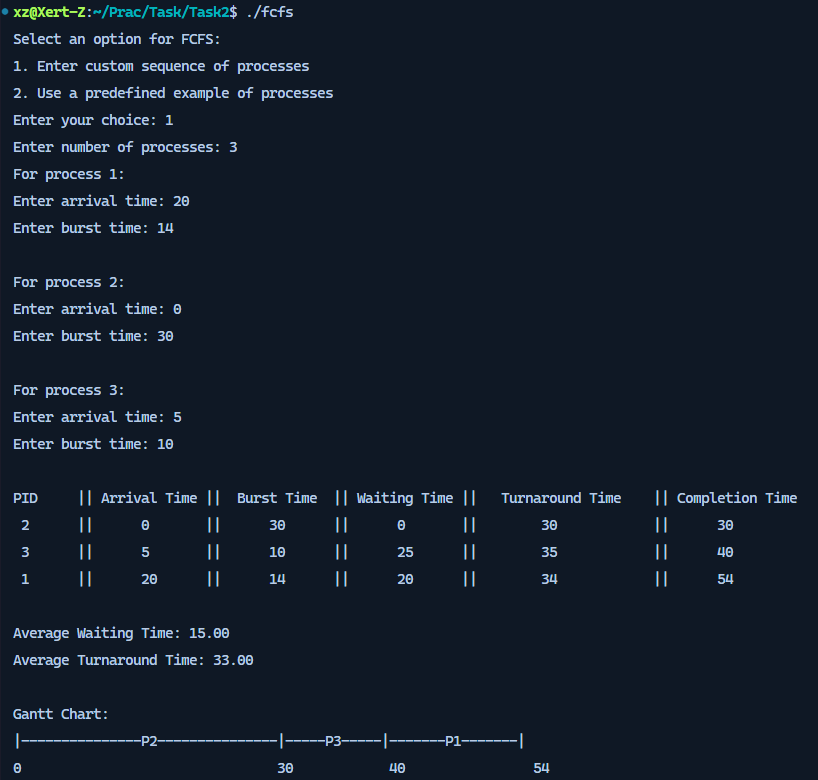


Figure 3: Complete execution of FCFS.c using Custom Sequence of Processess