**Project 4 Report: CPU Scheduler**

This project implements three different CPU scheduling algorithms:

* **First-Come, First-Served (FCFS)**: Processes tasks in the order they arrive
* **Priority Scheduling**: Selects the highest priority task for execution
* **Round Robin**: Uses time slicing to allocate CPU time to each process in rotation

Each implementation shares common components:

* **Task Structure**: Contains task name, priority, burst time, and task ID
* **Task List**: Manages the collection of tasks using a linked list
* **CPU Interface**: Simulates task execution

### **First-Come, First-Served (FCFS)**

**void add(char \*name, int priority, int burst):**

* Creates a new task with the provided parameters
* Assigns a unique task ID
* Inserts the task into the task list

**Task \*pickNextTask():**

* Follows FIFO principle by selecting the oldest task in the list (the one inserted first)
* Removes the selected task from the list
* Returns the task to be executed

**void schedule():**

* Implements the FCFS scheduling algorithm
* Continuously selects the next task until all tasks are executed
* Runs each task for its full burst time without interruption
* Prints completion messages for each task

### **Priority Scheduling**

**void add(char \*name, int priority, int burst):**

* Similar to FCFS implementation
* Creates a task and adds it to the task list

**Task \*pickNextTask():**

* Selects the task with the highest priority value
* Traverses the list to find the task with maximum priority
* Removes the selected task from the list
* Returns the highest priority task

**void schedule():**

* Implements priority-based scheduling
* Continuously selects the highest priority task until all tasks are executed
* Runs each task for its full burst time
* Prints completion messages for each task

### **Round Robin**

**void add(char \*name, int priority, int burst):**

* Creates a new task with provided parameters
* Assigns a unique task ID
* Adds the task to the tail of the queue

**void schedule(int quantum):**

* Implements Round Robin scheduling with the specified quantum
* Executes each task for a time slice equal to the quantum or the remaining burst time (whichever is smaller)
* If a task still has remaining burst time after its time slice, it's added back to the end of the queue
* Continues until all tasks are completed
* Prints completion messages for finished tasks

## **Main Function**

In all three implementations, the main function:

1. Validates command-line arguments
2. Opens and reads the input file containing task information
3. Parses each line to extract task name, priority, and burst time
4. Adds tasks to the scheduler
5. Invokes the scheduling algorithm
6. For Round Robin, allows specifying a custom time quantum (defaults to 10 if not provided)

**Video**

<https://youtu.be/GZifwHjJ2Qs>