Cairo University

kernel-sim-bootstrappers

Name	Code	Section
Ahmed Fathy	9230162	1
Ammar Yasser	9230593	2
Esraa Hassan	9230209	1
Tasneem Ahmed	9230295	1

Workload Distribution and Time Estimation

Name	Workload	Estimated Time
Ahmed Fathy	• Reading and parsing the processes.txt input file. • Take the user input to select the scheduling algorithm (HPF, SRTN, or RR). • Forking and managing the Clock and Scheduler processes. • Creating and managing the Arrival • Dispatching processes at the correct simulation time via IPC (message queues). • Handling cleanup of IPC resources and queues upon termination. • Integrating all components for system startup and shutdown • Synchronization between process_generator.c and scheduler.c with semaphores .	48 HOURS
Ammar Yasser	• Implementing Round Robin scheduling with time quantum support. • Managing process preemption and resumption using SIGSTOP and SIGCONT. • Maintaining the shared ready queue and PCB synchronization. • Calculating system metrics • Generating scheduler.log and scheduler.perf output files. • Signaling the scheduler upon process completion using SIGUSR2.	48 HOURS
Esraa Hassan	• Developing process.c to simulate a CPU-bound process. • Managing process behavior based on SIGCONT (resume) and SIGSTOP (pause). • Updating and tracking remaining time in shared memory.	48 HOURS

	• Implementing the Shortest Remaining TimeNext (SRTN) scheduling logic, including insertion based on remaining time. • Synchronize process.c with scheduler.c	
Tasneem Ahmed	• Implementing the Highest Priority First (HPF) scheduling logic. • Setting up the Process Control Block (PCB) structure for each process. • Managing shared memory attachments for PCBs and the ready queue. • Forking new processes and managing their states (start, stop, resume). • Implementing data Structures explained below	48 HOURS

Data Structures Used

Several specialized data structures were implemented to efficiently manage process scheduling and IPC communication. Each was chosen to suit the operational needs of the scheduling algorithms.

PCBQueue (Round Robin Ready Queue)

Purpose: Circular dynamic queue to manage ready processes in Round Robin scheduling.

Operations: enqueue, dequeue, resize.

Advantages: Fast O(1) enqueue and dequeue; scalable.

PriorityQueue (Process Arrival Management)

Purpose: Priority queue used by the process generator to organize processes by arrival time.

Operations: push, pop, top.

Sorting Modes: Arrival time only.

Advantages: Guarantees correct scheduling order on arrival.

MinHeap (Priority and Remaining Time Management)

Purpose: Min-heap for selecting processes with the highest priority (HPF) or shortest remaining time (SRTN).

Operations: insert, extract_min, update_remaining_time.

Comparator Functions: compare_priority, compare_remaining_time.

Advantages: O(log n) efficient selection of the next process to run.

PCB (Process Control Block)

Purpose: Structure representing a process, storing process metadata such as PID, arrival time, execution time, priority, remaining time, and IPC-related fields.

Advantages: Centralizes all scheduling-related information for each process.

Algorithm Explanation and Results

This project implements three CPU scheduling algorithms:

- Highest Priority First (HPF)
- Shortest Remaining Time Next (SRTN)
- Round Robin (RR)

The scheduler is responsible for managing processes according to the selected algorithm, handling context switches, and collecting performance metrics.

Highest Priority First (HPF)

Scheduling Type: Non-preemptive

Selection Criteria: Process with the highest priority (smallest numerical value).

Operation: When the CPU is idle, the scheduler selects the process with the highest priority from the ready queue.

The selected process runs until completion without interruption.

If multiple processes have the same priority, the process that arrived first is selected.

Data Structure Used: MinHeap sorted by priority.

Handling Ties: Arrival time is used to break ties between processes with equal priority.

Shortest Remaining Time Next (SRTN)

Scheduling Type: Preemptive

Selection Criteria: Process with the shortest remaining execution time.

Operation: At each clock tick or arrival of a new process, the scheduler compares the currently running process with the new candidates.

If a newly arrived process has a shorter remaining time than the running process, preemption occurs.

The running process is stopped (SIGSTOP) and the new process is resumed (SIGCONT).

Data Structure Used: MinHeap sorted by remaining time.

Handling Ties: Arrival time is used to resolve ties if remaining times are equal.

Round Robin (RR)

Scheduling Type: Preemptive with time slicing

Selection Criteria: First-Come, First-Served with time quantum.

Operation: Each process is assigned a fixed time slice (quantum).

If a process does not finish within its quantum, it is preempted and moved to the back of the queue.

The next process in the queue is resumed.

Data Structure Used: Circular dynamic queue (PCBQueue).

Quantum: User-defined at runtime.

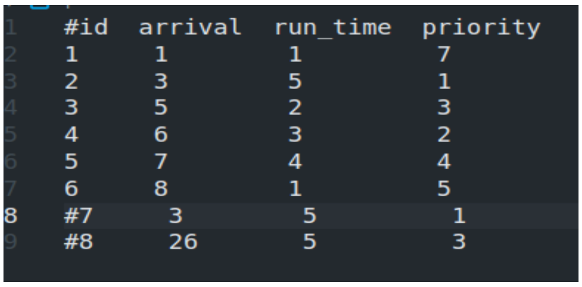
General Scheduling Process

1. **Initialization:** The user selects the scheduling algorithm; the corresponding ready queue (MinHeap for HPF/SRTN or Queue for RR) is initialized.
2. **Process Arrival:** The process generator reads processes.txt and sends processes to the scheduler via a message queue; processes are inserted into the ready queue upon arrival.
3. **Process Management**: Processes are forked and initially stopped (SIGSTOP); the scheduler resumes (SIGCONT) or preempts (SIGSTOP) processes as required.
4. **Context Switching:** In preemptive algorithms (SRTN, RR), the running process is stopped, its state is updated, and another process is scheduled; all state transitions are logged.
5. **Completion:** When a process finishes, it signals the scheduler using SIGUSR2; the scheduler updates metrics and releases process resources.

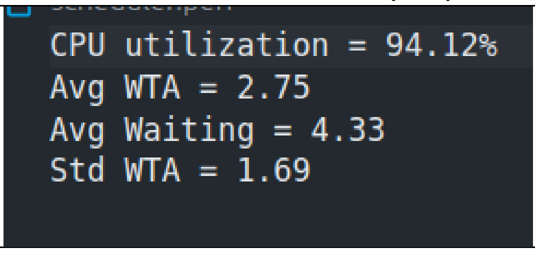
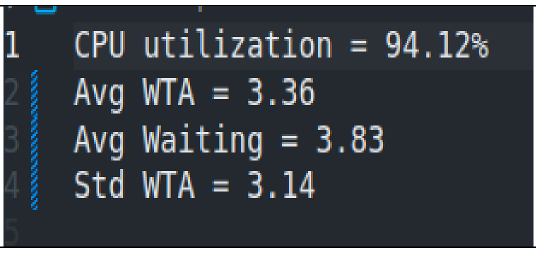
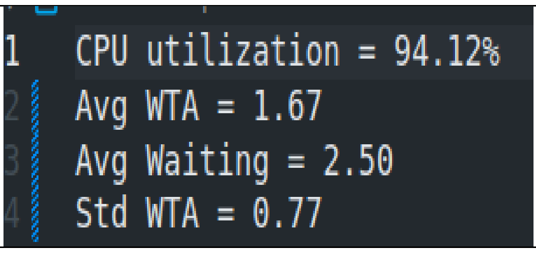
6. **Termination:** After all processes complete, the scheduler generates two output files: scheduler.log (event history) and scheduler.perf (performance summary).

Results

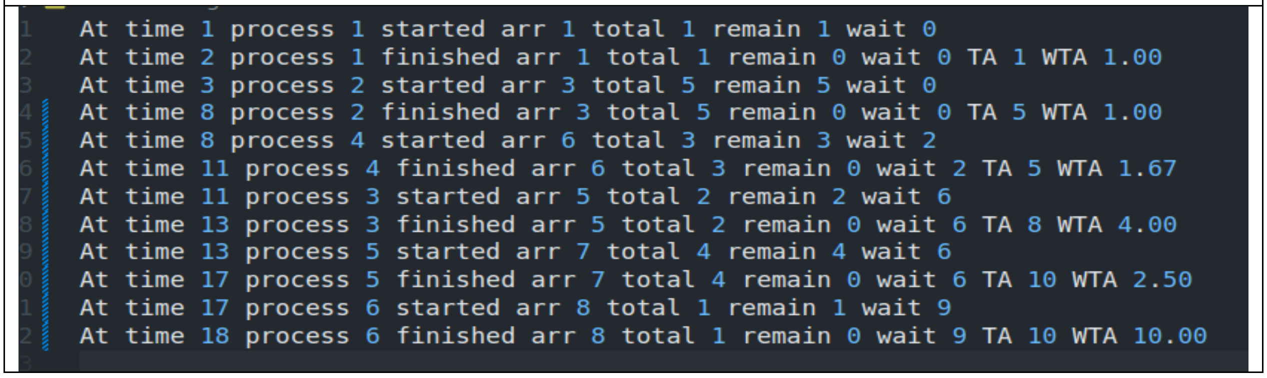
These are the results on this processes.txt test case:-



```
#id  arrival  run_time  priority
1    1        1         7
2    3        5         1
3    5        2         3
4    6        3         2
5    7        4         4
6    8        1         5
#7    3        5         1
#8    26       5         3
```

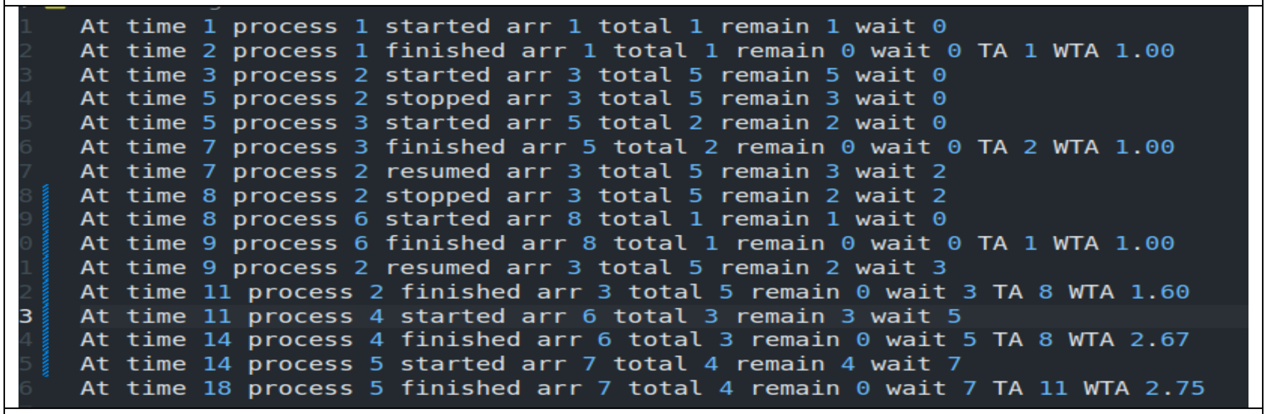
	Round Robin (RR)
scheduler.perf	 CPU utilization = 94.12% Avg WTA = 2.75 Avg Waiting = 4.33 Std WTA = 1.69
	Highest Priority First (HPF)
scheduler.perf	 CPU utilization = 94.12% Avg WTA = 3.36 Avg Waiting = 3.83 Std WTA = 3.14
	Shortest Remaining Time Next (SRTN)
scheduler.perf	 CPU utilization = 94.12% Avg WTA = 1.67 Avg Waiting = 2.50 Std WTA = 0.77

HPF



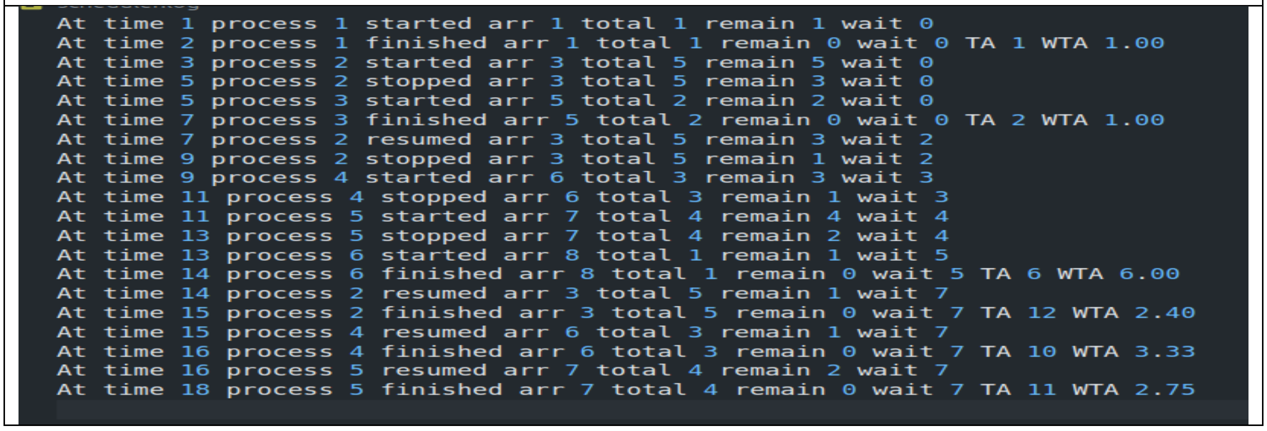
```
At time 1 process 1 started arr 1 total 1 remain 1 wait 0
At time 2 process 1 finished arr 1 total 1 remain 0 wait 0 TA 1 WTA 1.00
At time 3 process 2 started arr 3 total 5 remain 5 wait 0
At time 8 process 2 finished arr 3 total 5 remain 0 wait 0 TA 5 WTA 1.00
At time 8 process 4 started arr 6 total 3 remain 3 wait 2
At time 11 process 4 finished arr 6 total 3 remain 0 wait 2 TA 5 WTA 1.67
At time 11 process 3 started arr 5 total 2 remain 2 wait 6
At time 13 process 3 finished arr 5 total 2 remain 0 wait 6 TA 8 WTA 4.00
At time 13 process 5 started arr 7 total 4 remain 4 wait 6
At time 17 process 5 finished arr 7 total 4 remain 0 wait 6 TA 10 WTA 2.50
At time 17 process 6 started arr 8 total 1 remain 1 wait 9
At time 18 process 6 finished arr 8 total 1 remain 0 wait 9 TA 10 WTA 10.00
```

SRTN



```
At time 1 process 1 started arr 1 total 1 remain 1 wait 0
At time 2 process 1 finished arr 1 total 1 remain 0 wait 0 TA 1 WTA 1.00
At time 3 process 2 started arr 3 total 5 remain 5 wait 0
At time 5 process 2 stopped arr 3 total 5 remain 3 wait 0
At time 5 process 3 started arr 5 total 2 remain 2 wait 0
At time 7 process 3 finished arr 5 total 2 remain 0 wait 0 TA 2 WTA 1.00
At time 7 process 2 resumed arr 3 total 5 remain 3 wait 2
At time 8 process 2 stopped arr 3 total 5 remain 2 wait 2
At time 8 process 6 started arr 8 total 1 remain 1 wait 0
At time 9 process 6 finished arr 8 total 1 remain 0 wait 0 TA 1 WTA 1.00
At time 9 process 2 resumed arr 3 total 5 remain 2 wait 3
At time 11 process 2 finished arr 3 total 5 remain 0 wait 3 TA 8 WTA 1.60
At time 11 process 4 started arr 6 total 3 remain 3 wait 5
At time 14 process 4 finished arr 6 total 3 remain 0 wait 5 TA 8 WTA 2.67
At time 14 process 5 started arr 7 total 4 remain 4 wait 7
At time 18 process 5 finished arr 7 total 4 remain 0 wait 7 TA 11 WTA 2.75
```

RR



```
At time 1 process 1 started arr 1 total 1 remain 1 wait 0
At time 2 process 1 finished arr 1 total 1 remain 0 wait 0 TA 1 WTA 1.00
At time 3 process 2 started arr 3 total 5 remain 5 wait 0
At time 5 process 2 stopped arr 3 total 5 remain 3 wait 0
At time 5 process 3 started arr 5 total 2 remain 2 wait 0
At time 7 process 3 finished arr 5 total 2 remain 0 wait 0 TA 2 WTA 1.00
At time 7 process 2 resumed arr 3 total 5 remain 3 wait 2
At time 9 process 2 stopped arr 3 total 5 remain 1 wait 2
At time 9 process 4 started arr 6 total 3 remain 3 wait 3
At time 11 process 4 stopped arr 6 total 3 remain 1 wait 3
At time 11 process 5 started arr 7 total 4 remain 4 wait 4
At time 13 process 5 stopped arr 7 total 4 remain 2 wait 4
At time 13 process 6 started arr 8 total 1 remain 1 wait 5
At time 14 process 6 finished arr 8 total 1 remain 0 wait 5 TA 6 WTA 6.00
At time 14 process 2 resumed arr 3 total 5 remain 1 wait 7
At time 15 process 2 finished arr 3 total 5 remain 0 wait 7 TA 12 WTA 2.40
At time 15 process 4 resumed arr 6 total 3 remain 1 wait 7
At time 16 process 4 finished arr 6 total 3 remain 0 wait 7 TA 10 WTA 3.33
At time 16 process 5 resumed arr 7 total 4 remain 2 wait 7
At time 18 process 5 finished arr 7 total 4 remain 0 wait 7 TA 11 WTA 2.75
```

Assumptions

During the design and implementation, several practical assumptions were made to simplify scheduling, process management, and synchronization:

Process Behavior

Assumption: All processes are CPU-bound and do not perform I/O or voluntarily block themselves.

Reason: Process simulation decrements remaining time in a tight loop without sleeping, blocking, or waiting for any external events.

Unique Process IDs

Assumption: Each process generated has a unique ID as specified in processes.txt.

Reason: Process management (identification, logging, shared memory) relies on process IDs (PID and id_from_file) being unique.

Preemptive Models (SRTN and RR)

Assumption: Preemption occurs instantly upon context switching without overhead or delay.

Reason: When a process is preempted, it is immediately stopped with SIGSTOP, and a new process is resumed. No extra time is counted for context switch delays.

Input File Format

Assumption: The input file (processes.txt) strictly follows the required format:

# ID   ArrivalTime   RunTime   Priority

(Fields separated by tabs '\t')

Reason: The process generator expects properly formatted input and does not perform deep validation.

Shared Memory Handling

Assumption: Shared memory segments are properly attached, updated, and detached without corruption.

Reason: Each process creates its own shared memory segment to update its remaining time. There is no protection mechanism (like semaphores) since only one process writes to its own memory.

Process Arrival Synchronization

Assumption: Multiple processes arriving at the same clock tick are dispatched atomically before the scheduler proceeds.

Reason: A binary semaphore is used to synchronize the Process Generator and Scheduler. The Process Generator locks the semaphore before dispatching all processes of the current time unit and releases it afterward, ensuring the scheduler only schedules after all arrivals are inserted.