Implemented Scheduling Algorithms

FCFS Scheduling Policy Implementation

- Used `#ifdef FCFS` to conditionally compile this scheduling policy.
- The scheduler selects processes based on their creation time (`ctime`).
- It iterates through all runnable processes to find the process with the earliest `ctime`.
  Initialized a pointer (procFirst) to keep track of the process with the earliest arrival time.

In selecting a Process, For each runnable process:

If procFirst is not assigned (indicating that this is the first runnable process encountered), procFirst is set to point to the current process.

If procFirst is already assigned:

Compare the arrival time (ctime) of the current process with the ctime of procFirst.

If the current process arrived earlier, update procFirst to point to the current process.

For executing the Process,If a runnable process was found (procFirst is assigned):

Acquire the lock associated with procFirst to ensure exclusive access.
Change the state of procFirst to RUNNING to indicate that it's currently being executed.
Assign the CPU to procFirst for execution.
Perform a context switch to start the execution of procFirst.
After execution, release the CPU and release the lock of procFirst.

- The FCFS policy ensures that the process with the earliest `ctime` is given priority for execution. There is no preemption. Once a process starts executing, it continues until it voluntarily relinquishes the CPU or completes its execution.

Multi-Level Feedback Queue (MLFQ)

Priority Queue Configuration: Created four priority queues (0 to 3), where queue 0 has the highest priority and queue 3 has the lowest priority.

Each queue has different time slices:

Priority 0: 1 timer tick
Priority 1: 3 timer ticks
Priority 2: 9 timer ticks
Priority 3: 15 timer ticks

When a process is initiated, it's added to the end of the highest priority queue (priority 0).

Aging of Processes: To prevent processes from being stuck in lower priority queues indefinitely, an aging mechanism was implemented.
If a process spends too much time in a lower priority queue, it is periodically moved to a higher priority queue.

The scheduler iterates through all processes to check if they've been in the current queue for too long (ticks - p->arrival_inqueue > WAITING_LIMIT) and haven't used up their time slice (p->present_inqueue > 0).

If so, it moves the process to the next queue with higher priority and updates the time statistics.

For Selecting the Process to be Scheduled: Loops through all runnable processes and compares priorities to select the highest priority process.
CurProc is updated with the selected process.

If a process is selected (CurProc != 0), it proceeds to schedule.Updates process state, arrival time, and executes it.Records process statistics after execution.

Demotion of Processes in kernel/trap.c

Processes are demoted to lower priority queues if they exceed their time slice in the current queue.
The demotion logic is as follows:
If the process is in queue 0, it will be demoted to queue 1.
If the process is in queue 1, it will be demoted to queue 2.
If the process is in queue 2, it will be demoted to queue 3.

Checks if a process exceeded its time limit in the current queue.If so, it moves the process to a lower priority queue.

Performance Comparison

Setup:
- Test Environment: 1 CPU

Results:
- RR:
  - Average Running Time: `11`
  - Average Waiting Time: `142`

- FCFS:
  - Average Running Time: `21`
  - Average Waiting Time: `93`

- MLFQ:
  - Average Running Time: `13`
  - Average Waiting Time: `144`

Conclusion:
The average run time of round robin and MLFQ are almost equal.
Fcfs has the highest running time and is not suitable,RR and MLFQ are better than FCFS.