Process Scheduling

Code Submission

https://github.com/baralsamrat/MSCS630_Project

Deliverable 2

Process Scheduling

Samrat Baral

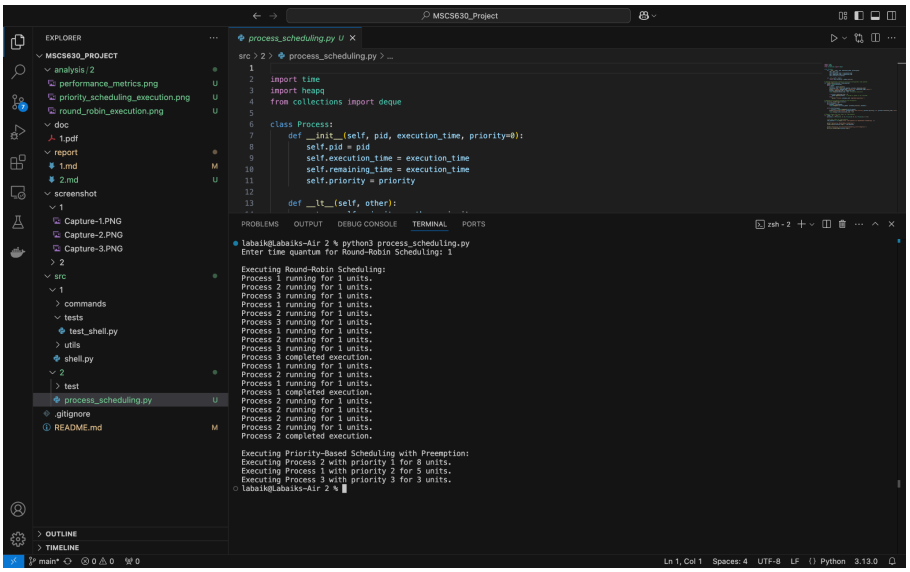
Feb 02, 2025

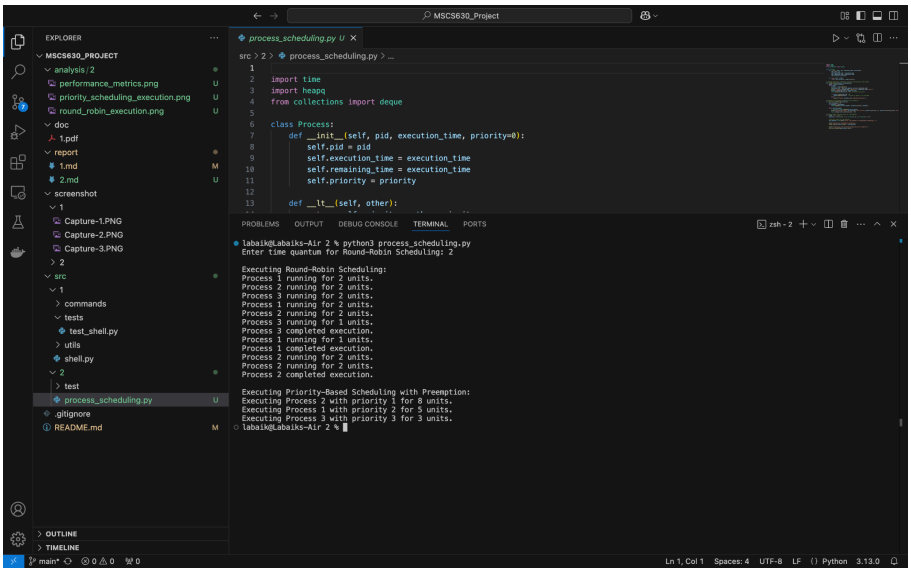
University of the Cumberlands

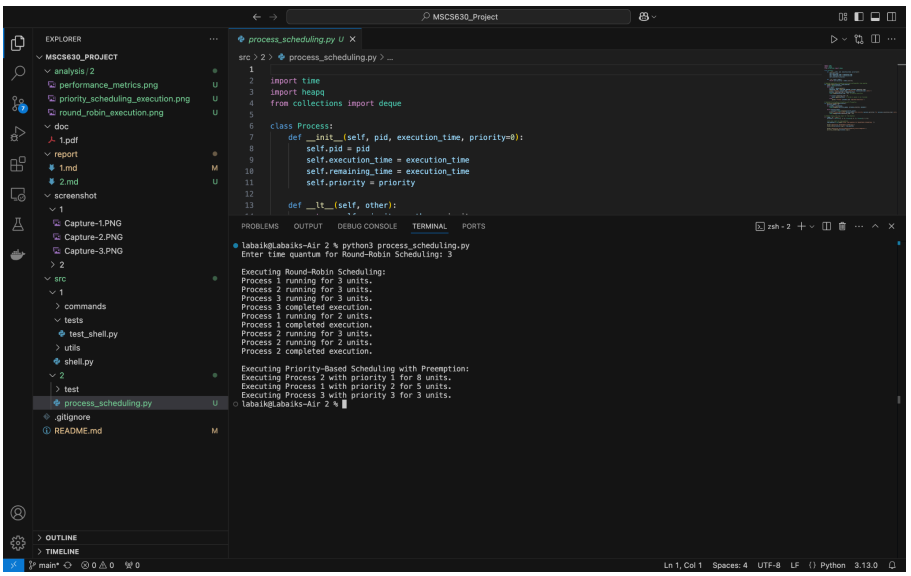
Advanced Operating Systems (MSCS-630-A01)

Primus Vekuh

Screenshots:







Process Management:

Scheduling Algorithm	Description	Specifics and Assignment
Round-Robin Scheduling	Distributes processor time equally among all active processes using a fixed quantum	Each process receives a user-configurable time slice (quantum). If the process does not complete within its allocated time slice, it is moved to the end of the ready queue. The `time.sleep()` function is used to simulate execution.
Priority-Based Scheduling with Preemption	The scheduler selects the highest-priority process for execution	Processes are assigned priorities, with a lower numerical value representing a higher priority. The highest-priority process is always executed first. If multiple processes share the same priority, they are scheduled in a First-Come, First-Served (FCFS) manner. If a higher-priority process enters the ready queue while a lower-priority process is executing, the lower-priority process is preempted, and the higher-priority process takes control of the processor.

Performance Analysis

Round Robin Scheduling

- **Core Principle:** This scheduling algorithm operates on the principle of time-sharing. Each process in the ready queue is assigned a fixed time quantum (time slice).
- **Fairness:** Round Robin ensures a fair allocation of CPU time to all processes. No process is left waiting indefinitely.
- **Turnaround Time:** While generally good, turnaround time can increase under Round Robin due to context switching overhead. Each switch incurs the cost of saving and restoring the process's state, which can add up with a large number of processes or short time slices.
- **No Prioritization:** Round Robin treats all processes equally, regardless of their priority or urgency. This can be a drawback in scenarios where some processes are more critical than others.

Priority Based Scheduling

- **Core Principle:** Processes are assigned priorities, and the CPU is allocated to the process with the highest priority at any given time.
- **Minimized Waiting Time:** Priority scheduling reduces the waiting time for high-priority processes, ensuring they receive prompt attention.
- **Starvation:** A major issue with priority scheduling is the potential for starvation of low-priority processes. If high-priority processes keep arriving, low-priority processes may never get a chance to execute.
- **Preemption:** Priority scheduling can be preemptive or non-preemptive. In preemptive priority scheduling, a running process can be interrupted if a higher-priority process arrives. Non-preemptive priority scheduling allows a running process to complete its time slice even if a higher-priority process arrives.
- **Priority Inversion:** A situation known as priority inversion can occur where a high-priority process is indirectly blocked by a low-priority process. This can happen if the low-priority process holds a resource that the high-priority process needs.

Additional Considerations on both Round Robin and Priority Based scheduling are widely used in operating systems. The choice between them depends on the specific requirements of the system and the characteristics of the processes being scheduled.

- **Hybrid Approaches:** Many modern operating systems employ hybrid scheduling algorithms that combine elements of both Round Robin and Priority Based scheduling to leverage their advantages and mitigate their drawbacks.
- **Dynamic Priorities:** Some systems use dynamic priority scheduling, where process priorities can change over time based on their behavior or external factors.
- **Multi-Level Queues:** Multi-level queue scheduling uses multiple queues with different scheduling algorithms for each queue. Processes can be moved between queues based on their priority or resource requirements.

Scheduling Algorithm	Challenge	Solution	Improvement
Priority Scheduling	Preemption can result in the starvation of lower-priority processes	Utilize a priority queue to manage processes. When a higher-priority process arrives, preempt the current process and return it to the priority queue. The higher-priority process then assumes control of the CPU.	Aging can be implemented to periodically increase the priority of lower-priority processes.
Round-Robin Scheduling	The default time slice may not be equitable for all processes	Implement user-configurable quantum switching, enabling users to specify the time slice for each process. This ensures that all processes receive an equitable allocation of CPU time and allows users to fine-tune the scheduler for their workload.	Dynamically adjust time slices based on process behavior or system load.

Conclusion

The shell now simulates process scheduling, mimicking OS process management. This lays the groundwork for future enhancements like memory management and synchronization.