REPORT COVERING PERFORMANCE COMPARISON BETWEEN THE DEFAULT AND 3 IMPLEMENTED POLICIES.

First Come First Serve (FCFS)

Algorithm:
In FCFS policy, the process which arrives first is executed first. It will work best when incoming processes are short and there is no need for the processes to execute in a specific order.

Advantages:
- FCFS Algorithm doesn’t include any complex logic, it just puts the process requests in a queue and executes it one by one.
- Hence, FCFS is pretty simple and easy to implement.
- Eventually, every process will get a chance to run, so starvation doesn't occur.

Disadvantages:
- There is no option for pre-emption of a process. If a process is started, then CPU executes the process until it ends.
- Because there is no pre-emption, if a process executes for a long time, the processes in the back of the queue will have to wait for a long time before they get a chance to be executed.

Round Robin (RR)[DEFAULT]

Algorithm:
Round Robin is used as default policy in xv6. In this policy, each process gets executed for certain ticks and then next process is executed for certain ticks and so on it continues till the end of the queue and then again it starts from the beginning. It will work best when processes are a mix of long and short processes and task will only be completed if all the processes are executed successfully in a given time.

Advantages:
- Each process is served by the CPU for a fixed time quantum, so all processes are given the same priority.
- Starvation doesn't occur because for each round robin cycle, every process is given a fixed time to execute. No process is left behind.

Disadvantages:
- The throughput in RR largely depends on the choice of the length of the time quantum. If time quantum is longer than needed, it tends to exhibit the same behavior as FCFS.
- If time quantum is shorter than needed, the number of times that CPU switches from one process to another process, increases. This leads to decrease in CPU efficiency.

Priority based Scheduling

Algorithm:
In PRIORITY based policy the process having the least priority is executed first. It will work best when processes are a mix of user based and kernel based

processes because generally kernal based processes have higher priority.

Advantages:

- The priority of a process can be selected based on memory requirement, time requirement or user preference. For example, a high end game will have better graphics, that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.

Disadvantages:

- A second scheduling algorithm is required to schedule the processes which have same priority.
- In preemptive priority scheduling, a higher priority process can execute ahead of an already executing lower priority process. If lower priority process keeps waiting for higher priority processes, starvation occurs.

MULTILEVEL QUEUE SCHEDULING

Algorithm:

In this policy 5 different types of queues are kept. The one having more CPU burst time is pushed in the last priority queue which leaves behind interactive processes in the top priority queues.

Advantages:

- Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics.
- If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.
- Throughput is increased as the top priority queue contains the processes which requires minimum CPU time.

Disadvantages:

- We need to check starvation condition for each process which takes time.
- Moving the process around queue produce more CPU overhead.