NAME: Aditya Raj

Reg No.: 11804514

GitHub link: <https://github.com/Aditya11100/CSE316-Operating-System>

Priority Scheduling Algorithm

Priority Scheduling could be a method of scheduling processes that's supported priority. during this algorithm, the scheduler selects the tasks to figure as per the priority.

The processes with higher priority should be disbursed first, whereas jobs with equal priorities are disbursed on a round-robin or FCFS basis. Priority depends upon memory requirements, time

Types of Priority Scheduling:

Preemptive Scheduling

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it's important to run a task with a better priority before another lower priority task, whether or not the lower priority task remains running. The lower priority task holds for a few time and resumes when the upper priority task finishes its execution.

Non-Preemptive Scheduling

In this kind of scheduling method, the CPU has been allocated to a selected process. the method that keeps the CPU busy, will release the CPU either by switching context or terminating. it's the sole method that may be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

Characteristics of Priority Scheduling

• A CPU algorithm that schedules processes supported priority.

• It utilized in Operating systems for performing batch processes.

• If two jobs having the identical priority are READY, it works on a primary COME, FIRST SERVED basis.

• In priority scheduling, variety is assigned to every process that indicates its priority level.

• Lower the amount, higher is that the priority.

• In this kind of scheduling algorithm, if a more moderen process arrives, that's having a better priority than the currently running process, then the currently running process is preempted.

Alogorithm:

Priority Scheduling Algorithm: during this, each process is assigned a priority and processes are

executed on the idea of their priority. we are able to either opt to set priority of very cheap number to

be the primary priority or the other way around. No other process can execute until the method with the best

priority has fully executed. If two processes have same priority, then process is executed on the

basis of their point in time.

3. Calculate complexity of implemented algorithm. (Student must specify complexity of

each line of code together with overall complexity)

Description (purpose of use):

- Complexity of Priority Scheduling Algorithm: O(n)

**- Priority Scheduling Algorithm** (written in Python3):

def waitingTime(processes, n, wt):

wt[0] = 0

for i in range(1, n):

wt[i] = processes[i - 1][1] + wt[i - 1]

def turnAroundTime(processes, n, wt, tat):

for i in range(n):

tat[i] = processes[i][1] + wt[i]

def findavgTime(processes, n):

wt = [0] \* n

tat = [0] \* n

waitingTime(processes, n, wt)

turnAroundTime(processes, n, wt, tat)

print("\nProcesses Burst Time Waiting", "Time Turn-Around Time")

total\_tat = 0

for i in range(n):

total\_wt = total\_wt + wt[i]

total\_tat = total\_tat + tat[i]

print(" ", processes[i][0], "\t\t",

processes[i][1], "\t\t",

wt[i], "\t\t", tat[i])

print("\nAverage waiting time = %.5f "%(total\_wt /n))

print("Average turn around time = ", total\_tat / n)

def priorityScheduling(proc, n):

proc = sorted(proc, key = lambda proc:proc[2],

reverse = True);

print("Order in which processes gets executed")

for i in proc:

print(i[0], end = " ")

findavgTime(proc, n)

if \_\_name\_\_ =="\_\_main\_\_":

process = [[1, 10, 1],

[2, 20, 0],

[3, 15, 1],

[4, 11, 2]]

n = 4

priorityScheduling(process, n)

**Constraints for Priority Scheduling Algorithm:**

1. Process with highest priority gets preference first.

2. If two or more processes have same priority, then process if executed on their arrival time.

3. If there is already a process running and another process comes with higher priority, then the

running process is pre-empted.

**Test Case:**

Priority Scheduling Algorithm:

1. Process 1 comes -> starts getting executed

2. Next process comes with lower priority -> Waits till execution finishes

3. Another process comes with higher priority -> Pre-empts running process and starts

executing

4. If two process have similar priority -> process that arrives first, gets executed