**Exp No.** 9 **Name:** PONNURI ANIRUDDHA

**Date**:03/10/2022 **Reg No:** RA2112704010015

**Priority and Round robin scheduling**

**Aim: --**

To implement Priority Type and Round Robin type of CPU Scheduling

**Procedure: --**

**Priority CPU Scheduling**

Priority scheduling is one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on. Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Round Robin CPU Scheduling**: --

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique. The period of time for which a process or job is allowed to run in a pre-emptive method is called time quantum. Each process or job present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will end else the process will go back to the waiting table and wait for its next turn to complete the execution.

**Code (Priority Scheduling): --**

*def findWaitingTime(processes, n, wt):*

*wt[0] = 0*

*# calculating waiting time*

*for i in range(1, n):*

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

*# Function to calculate turn around time*

*def findTurnAroundTime(processes, n, wt, tat):*

*# Calculating turnaround time by*

*# adding bt[i] + wt[i]*

*for i in range(n):*

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

*# Function to calculate average waiting*

*# and turn-around times.*

*def findavgTime(processes, n):*

*wt = [0] \* n*

*tat = [0] \* n*

*# Function to find waiting time*

*# of all processes*

*findWaitingTime(processes, n, wt)*

*# Function to find turn around time*

*# for all processes*

*findTurnAroundTime(processes, n, wt, tat)*

*# Display processes along with all details*

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

*total\_wt = 0*

*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):*

*# Sort processes by priority*

*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)*

*# Driver code*

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

*# Process id's*

*proc = [[1, 10, 1],*

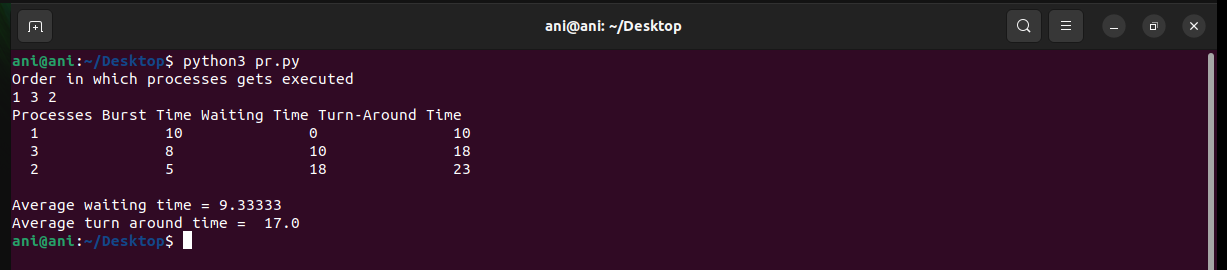
*[2, 5, 0],*

*[3, 8, 1]]*

*n = 3*

*priorityScheduling(proc, n)*

**OUTPUT (Priority Scheduling): --**



**CODE (Round Robin CPU Scheduling): --**

*def findWaitingTime(processes, n, bt, wt, quantum):*

*rem\_bt = [0] \* n*

*# Copy the burst time into rt[]*

*for i in range(n):*

*rem\_bt[i] = bt[i]*

*t = 0 # Current time*

*# Keep traversing processes in round*

*# robin manner until all of them are*

*# not done.*

*while(1):*

*done = True*

*# Traverse all processes one by*

*# one repeatedly*

*for i in range(n):*

*# If burst time of a process is greater*

*# than 0 then only need to process further*

*if (rem\_bt[i] > 0) :*

*done = False # There is a pending process*

*if (rem\_bt[i] > quantum) :*

*# Increase the value of t i.e. shows*

*# how much time a process has been processed*

*t += quantum*

*# Decrease the burst\_time of current*

*# process by quantum*

*rem\_bt[i] -= quantum*

*# If burst time is smaller than or equal*

*# to quantum. Last cycle for this process*

*else:*

*# Increase the value of t i.e. shows*

*# how much time a process has been processed*

*t = t + rem\_bt[i]*

*# Waiting time is current time minus*

*# time used by this process*

*wt[i] = t - bt[i]*

*# As the process gets fully executed*

*# make its remaining burst time = 0*

*rem\_bt[i] = 0*

*# If all processes are done*

*if (done == True):*

*break*

*# Function to calculate turn around time*

*def findTurnAroundTime(processes, n, bt, wt, tat):*

*# Calculating turnaround time*

*for i in range(n):*

*tat[i] = bt[i] + wt[i]*

*# Function to calculate average waiting*

*# and turn-around times.*

*def findavgTime(processes, n, bt, quantum):*

*wt = [0] \* n*

*tat = [0] \* n*

*# Function to find waiting time*

*# of all processes*

*findWaitingTime(processes, n, bt, wt, quantum)*

*# Function to find turn around time*

*# for all processes*

*findTurnAroundTime(processes, n, bt, wt, tat)*

*# Display processes along with all details*

*print("Processes Burst Time Waiting","Time Turn-Around Time")*

*total\_wt = 0*

*total\_tat = 0*

*for i in range(n):*

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

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

*print(" ", i + 1, "\t\t", bt[i],*

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

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

*print("Average turn around time = %.5f "% (total\_tat / n))*

*# Driver code*

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

*# Process id's*

*proc = [1, 2, 3]*

*n = 3*

*# Burst time of all processes*

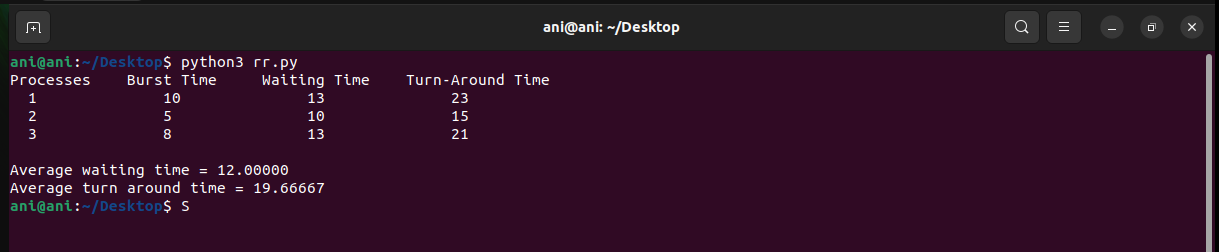
*burst\_time = [10, 5, 8]*

*# Time quantum*

*quantum = 2;*

*findavgTime(proc, n, burst\_time, quantum)*

**OUTPUT (Round Robin CPU Scheduling): --**



**RESULT: --**

The priority based and round robin based types of CPU scheduling algorithms are implemented and tested.