**Student Name: Jaya Prakash Reddy. P**

**Student ID: 11705253**

**Email Address: jayaprakashreddyp2000@gmail.com**

**GitHub Link:**

# QUESTION NO 15:

# A uniprocessor system has n number of CPU intensive processers has its own requirements of CPU burst. The process with CPU burst is given the highest priority.

# A late-arrival highest priority process can preempt a current running process with lower priority is never starved due to the execution of lower priority process. What should be its average waiting due to the execution of lower priority process. What should be its average turnaround time if no two processes are arriving are arriving at same time.

# Descsription:

# A multiprogramming system is a basic form of parallel processing in which multiple programs are run at the same time on a uniprocessor. As a result of having only a single processor, concurrent execution of multiple programs is impossible.

# ALGORITHM:-

# 1. Take n no.of processes.

# 2. In this algorithm the process with small Amount of time running until completion is selected to execute.

# 3. The compiler works on the less arrival time and also compares the burst time of the arrival time.

# 4. If the burst time of arrival time is higher among remaining processes then it will not be taken first.

# 5. The processes executes first which has less arrival and less burst time as comparing with others.

# 6. The n processes compiles n times and enter a for loop.

# 7. In this loop, it checks the processes and assign the values to them and ends the loop when process no is greater than.

# 8. The process will continue until all the processes completed their execution.

# Entire code:

# #include<stdio.h>

# int n;

# struct process

# {

# int p\_no;

# int arrival\_t,burst\_t,ct,wait\_t,taround\_time,p;

# int flag;

# }

# p\_list[100];

# void Sorting()

# {

# struct process p;

# int i, j;

# for(i=0;i<n-1;i++)

# {

# for(j=i+1;j<n;j++)

# {

# if(p\_list[i].arrival\_t > p\_list[j].arrival\_t)

# {

# p = p\_list[i];

# p\_list[i] = p\_list[j];

# p\_list[j] = p;

# }

# }

# }

# }

# int main()

# {

# int i,t=0,b\_t=0,peak;

# int a[10];

# float wait\_time = 0, taround\_time = 0, avg\_w\_t=0, avg\_taround\_time=0;

# printf("enter the no. of processes: ");

# scanf("%d",&n);

# for(i = 0; i < n; i++)

# {

# p\_list[i].p\_no = i+1;

# printf("\nEnter Details For P%d process:-\n", p\_list[i].p\_no);

# printf("Enter Arrival Time: ");

# scanf("%d", &p\_list[i].arrival\_t );

# printf("Enter Burst Time: ");

# scanf("%d", &p\_list[i].burst\_t);

# p\_list[i].flag = 0;

# b\_t = b\_t + p\_list[i].burst\_t;

# }

# Sorting();

# for(int i=0;i<n;i++)

# {

# a[i]=p\_list[i].burst\_t;

# }

# p\_list[9].burst\_t = 9999;

# for(t = p\_list[0].arrival\_t; t <= b\_t+1;)

# {

# peak = 9;

# for(i=0;i<n;i++)

# {

# if(p\_list[i].arrival\_t <= t && p\_list[i].burst\_t < p\_list[peak].burst\_t && p\_list[i].flag != 1)

# {

# peak = i;

# }

# if(p\_list[peak].burst\_t==0 && p\_list[i].flag != 1)

# {

# p\_list[i].flag = 1;

# p\_list[peak].ct=t;p\_list[peak].burst\_t=9999;

# printf("P%d completes in %d\n",p\_list[i].p\_no,p\_list[peak].ct);

# }

# }

# t++;

# (p\_list[peak].burst\_t)--;

# }

# for(i=0;i<n;i++)

# {

# p\_list[i].taround\_time=(p\_list[i].ct)-(p\_list[i].arrival\_t);

# avg\_taround\_time=avg\_taround\_time+p\_list[i].taround\_time;

# p\_list[i].wait\_t=((p\_list[i].taround\_time)-a[i]);

# avg\_w\_t=avg\_w\_t+p\_list[i].wait\_t;

# }

# printf("PNO\tAT\tCT\tTA\tWTt\n");

# for(i=0;i<n;i++)

# {

# printf("P%d\t%d\t%d\t%d\t%d\n",p\_list[i].p\_no,p\_list[i].arrival\_t,p\_list[i].ct,p\_list[i].taround\_time

# ,p\_list[i].wait\_t);

# }

# printf("Average Turn around Time: %f\t\n\n",avg\_taround\_time);

# printf("Average Waiting Time :\t %f\t\n",avg\_w\_t);

# }

# Test cases:

**Process ArrivalTime BurstTime Turn. A. T Waiting. T**

P1 0 2 2 0

P2 1 14 13 7

P3 2 5 5 0

Average Turn Around Time :- 6

Average Waiting Time :- 2

**Reference: -**

# TEST CASE:-1

# 

# D:\WhatsApp Image 2019-04-01 at 22.09.00.jpeg

# t

# te

# te

**TEST CASE:-2**

**Process ArrivalTime BurstTime Turn. A. T Waiting. T**

P1 0 88 2

P2 1 14 13 7

P3 2 0 -2 -10

P4 2 3 1 0

P5 4 21 17 10

Average Turn Around Time :- 7

Average Waiting Time :- 1

**Reference:**

# te

# afradffefgrga

# D:\WhatsApp Image 2019-04-01 at 22.21.53.jpeg

# D:\WhatsApp Image 2019-04-01 at 22.22.05.jpeg

# t

# te

# te

**Process ArrivalTime BurstTime Turn. A. T Waiting. T**

P1 0 22 0

P2 1 24 23 17

P3 2 7 5 0

P4 3 17 14 9

P5 4 30 26 19

P6 6 11 5 1

Average Turn Around Time :- 12

Average Waiting Time :- 7

TEST CASE:-3

# D:\WhatsApp Image 2019-04-01 at 22.33.50.jpeg

# D:\WhatsApp Image 2019-04-01 at 22.34.07.jpeg

# Constrains:-

# Some constrains in my program are:-

# =>the process goes up to n-1 CPU intensive processors arrival time of [i] list is greater than

# the arrival time of [j] list.

# for(i=0;i<n-1;i++)

# {

# for(j=i+1;j<n;j++)

# {

# if(p\_list[i].arrival\_t > p\_list[j].arrival\_t)

# {

# p = p\_list[i];

# p\_list[i] = p\_list[j];

# p\_list[j] = p;

# }

# }

# }

# =>to check whether all processes are completed their execution or not

# for(int i=0;i<n;i++)

# {

# a[i]=p\_list[i].burst\_t;

# }

# p\_list[9].burst\_t = 9999;

# for(t = p\_list[0].arrival\_t; t <= b\_t+1;)

# {

# peak = 9;

# for(i=0;i<n;i++)

# {

# if(p\_list[i].arrival\_t <= t && p\_list[i].burst\_t < p\_list[peak].burst\_t && p\_list[i].flag != 1)

# {

# peak = i;

# }

# if(p\_list[peak].burst\_t==0 && p\_list[i].flag != 1)

# {

# p\_list[i].flag = 1;

# p\_list[peak].ct=t;p\_list[peak].burst\_t=9999;

# printf("P%d completes in %d\n",p\_list[i].p\_no,p\_list[peak].ct);

# }

# }

# t++;

# (p\_list[peak].burst\_t)--;

# }

**Boundary Condition: -**

Here the main boundary condition is to execute the process maximum of 2 units time and to then it holds the process after the 2 units. In this period of time the other process will execute which has less arrival time as compare to previous holding process and it will go on with this boundary condition. If the time limit exceeds for a single process among all processes(i.e. 2 units will be as 3 or 4 units ) then the program or the output will become error or incorrect.