SJF

#include <stdio.h>

int main() {

int n;

// Step 1: Input number of processes

printf("Enter number of processes: ");

scanf("%d", &n);

// Declare arrays to store process data

int pid[20], at[20], bt[20], ct[20], tat[20], wt[20], completed[20] = {0};

int time = 0, completed\_count = 0;

// Step 2: Input PID, Arrival Time (AT), and Burst Time (BT)

printf("Enter Process ID, Arrival Time and Burst Time:\n");

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

scanf("%d %d %d", &pid[i], &at[i], &bt[i]);

}

// Step 3: Loop until all processes are completed

while (completed\_count < n) {

int idx = -1;

// Step 4: Find the process with the shortest burst time among the arrived and not completed

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

if (at[i] <= time && completed[i] == 0) {

if (idx == -1 || bt[i] < bt[idx]) {

idx = i; // Store index of the process with the shortest BT

}

}

}

// Step 5: If a process is found to execute

if (idx != -1) {

ct[idx] = time + bt[idx]; // Completion Time = current time + burst time

tat[idx] = ct[idx] - at[idx]; // Turnaround Time = Completion - Arrival

wt[idx] = tat[idx] - bt[idx]; // Waiting Time = Turnaround - Burst

time = ct[idx]; // Move current time to process completion

completed[idx] = 1; // Mark this process as completed

completed\_count++; // Increase completed process count

} else {

time++; // If no process is ready, move time forward (CPU idle)

}

}

// Step 6: Print the results in a table

printf("\nPID\tAT\tBT\tCT\tTAT\tWT\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\n", pid[i], at[i], bt[i], ct[i], tat[i], wt[i]);

}

return 0;

}

START

1. Input number of processes (n)

2. For each process i from 0 to n-1:

a. Input Process ID (PID[i])

b. Input Arrival Time (AT[i])

c. Input Burst Time (BT[i])

d. Set Completed[i] = 0 // Mark all as not completed

3. Set time = 0

4. Set completed\_count = 0

5. While completed\_count < n:

a. Set idx = -1 // Index of selected process

b. For each process i from 0 to n-1:

i. If AT[i] <= time AND Completed[i] == 0:

- If idx == -1 OR BT[i] < BT[idx]:

→ Set idx = i // Select process with shortest burst

c. If idx ≠ -1: // A process is selected

i. Completion Time CT[idx] = time + BT[idx]

ii. Turnaround Time TAT[idx] = CT[idx] - AT[idx]

iii. Waiting Time WT[idx] = TAT[idx] - BT[idx]

iv. Set time = CT[idx] // Move time forward

v. Mark Completed[idx] = 1

vi. Increment completed\_count

d. Else:

i. Increment time by 1 // CPU is idle

6. Print table of PID, AT, BT, CT, TAT, WT for all processes

END