U18CO018

Shubham Shekhaliya

Assignment – 5

Operating System

Write a c/Java program for simulation of

(1). Shortest Job First (SJF)

(2). Shortest Remaining Time First (SRTF) CPU scheduler.

Program should maintain Ready\_Q using process pointers. Each Process should have cpu\_time and arrival\_time . Cpu\_time and arrival\_time should be generated randomly.

Demonstrate processes context switch according to SRTF Scheduling.

**Struct process**

{

Int pid;

Int cpu\_time;

Int arrival\_time;

Struct process \* next;

}

Void main()

{

Initialize

/\*

Generate n no of process.

Assign **process\_id** serially to each process.

Randomly generate **cpu\_time** and **arrival\_time** for each process.

Create &amp; initialize **Ready\_Q** with n no of processes.

\*/

Untill Ready\_Q <> empy

{

select Queue **Ready\_Q**

Randomly generate **current\_time** requirement out of **total\_time** requirement for

corresponding process present on the front of corresponding queue.

Call delay for current\_time no of times.

Subtract current\_time from total\_cpu\_time &amp; update total\_cpu\_time

Shift process to another queue.

displyQueue.

}

**SJF Algorithm :-**

#include <bits/stdc++.h>

using namespace std;

class process {

public:

    int pid;

    int cpuTime;

    int arrivalTime;

};

void SJF(vector<process> &p) {

    int n, cur;

    int curr\_time = 0, cur\_cpu\_time = INT\_MAX;

    vector<process> ans;

    while (p.size() > 0) {

        n = p.size();

        cur\_cpu\_time = INT\_MAX, cur = -1;

        for (int i = 0; i < n; i++) {

            if (p[i].arrivalTime <= curr\_time && p[i].cpuTime < cur\_cpu\_time) {

                cur\_cpu\_time = p[i].cpuTime;

                cur = i;

            }

        }

        if (cur != -1) {

            ans.push\_back(p[cur]);

            p.erase(p.begin() + cur);

            curr\_time += cur\_cpu\_time;

        } else {

            curr\_time++;

        }

    }

    cout << "PID\tARRIVAL\_TIME\tCPU\_TIME\tCOMP\_TIME" << endl;

    int comp\_time = 0;

    for (int i = 0; i < ans.size(); i++) {

        comp\_time += ans[i].cpuTime;

        cout << ans[i].pid << '\t' << ans[i].arrivalTime << "\t\t" << ans[i].cpuTime << "\t\t" << comp\_time << endl;

    }

}

int main() {

    int n;

    cout << "Enter the value of n : ";

    cin >> n;

    vector<process> init(n);

    cout << "PID\tARRIVAL\_TIME\tCPU\_TIME" << endl;

    for (int i = 0; i < n; i++) {

        init[i].pid = i + 1;

        init[i].cpuTime = rand() % (n \* 3) + 1;

        init[i].arrivalTime = rand() % (n \* 3) + 1;

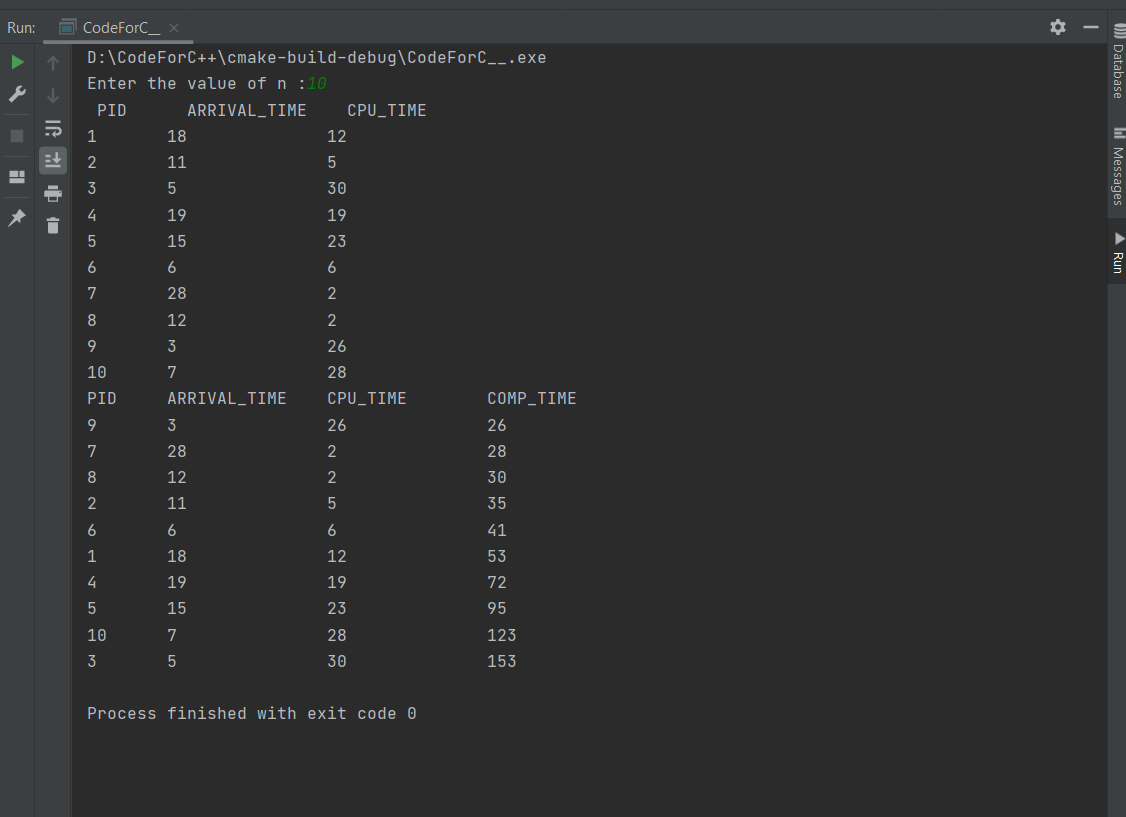
        cout << init[i].pid << "\t" << init[i].arrivalTime << "\t\t" << init[i].cpuTime << endl;

    }

    SJF(init);

}

**Output:-**

****

**SRTF Algorithm:-**

#include <bits/stdc++.h>

using namespace std;

class process {

public:

    int pID;

    int cpuTime;

    int arrivalTime;

};

void SRTF\_Algo(vector<process>  p) {

    int n, cur;

    int curr\_time = 0, cur\_cpu\_time = INT\_MAX;

    vector<pair<int, int> > ans;

    while (p.size() > 0) {

        n = p.size();

        cur\_cpu\_time = INT\_MAX, cur = -1;

        for (int i = 0; i < n; i++) {

            if (p[i].arrivalTime <= curr\_time && p[i].cpuTime < cur\_cpu\_time) {

                cur\_cpu\_time = p[i].cpuTime;

                cur = i;

            }

        }

        if (cur != -1) {

            ans.push\_back({curr\_time, p[cur].pID});

            p[cur].cpuTime -= 1;

            if (p[cur].cpuTime == 0)

                p.erase(p.begin() + cur);

            curr\_time++;

        } else {

            curr\_time++;

            ans.push\_back({curr\_time, -1});

        }

    }

    cout << endl << "CPU\_TIME\tPID" << endl;

    cout << ans[0].first << "\t\t" << ans[0].second << endl;

    for (int i = 1; i < ans.size(); i++) {

        if (ans[i - 1].second != ans[i].second)

            cout << ans[i].first << "\t\t" << ans[i].second << endl;

    }

    cout << "End -->-->--> " << ans[ans.size() - 1].first + 1 << endl;

}

int main() {

    int n;

    cout << "Enter the value of n : \n";

    cin >> n;

    vector<process> init(n);

    cout << "PID\tARRIVAL\_TIME\tCPU\_TIME" << endl;

    for (int i = 0; i < n; i++) {

        init[i].pID = i + 1;

        init[i].cpuTime = rand() % (n \* 5) + 1;

        init[i].arrivalTime = rand() % (n \* 5) + 1;

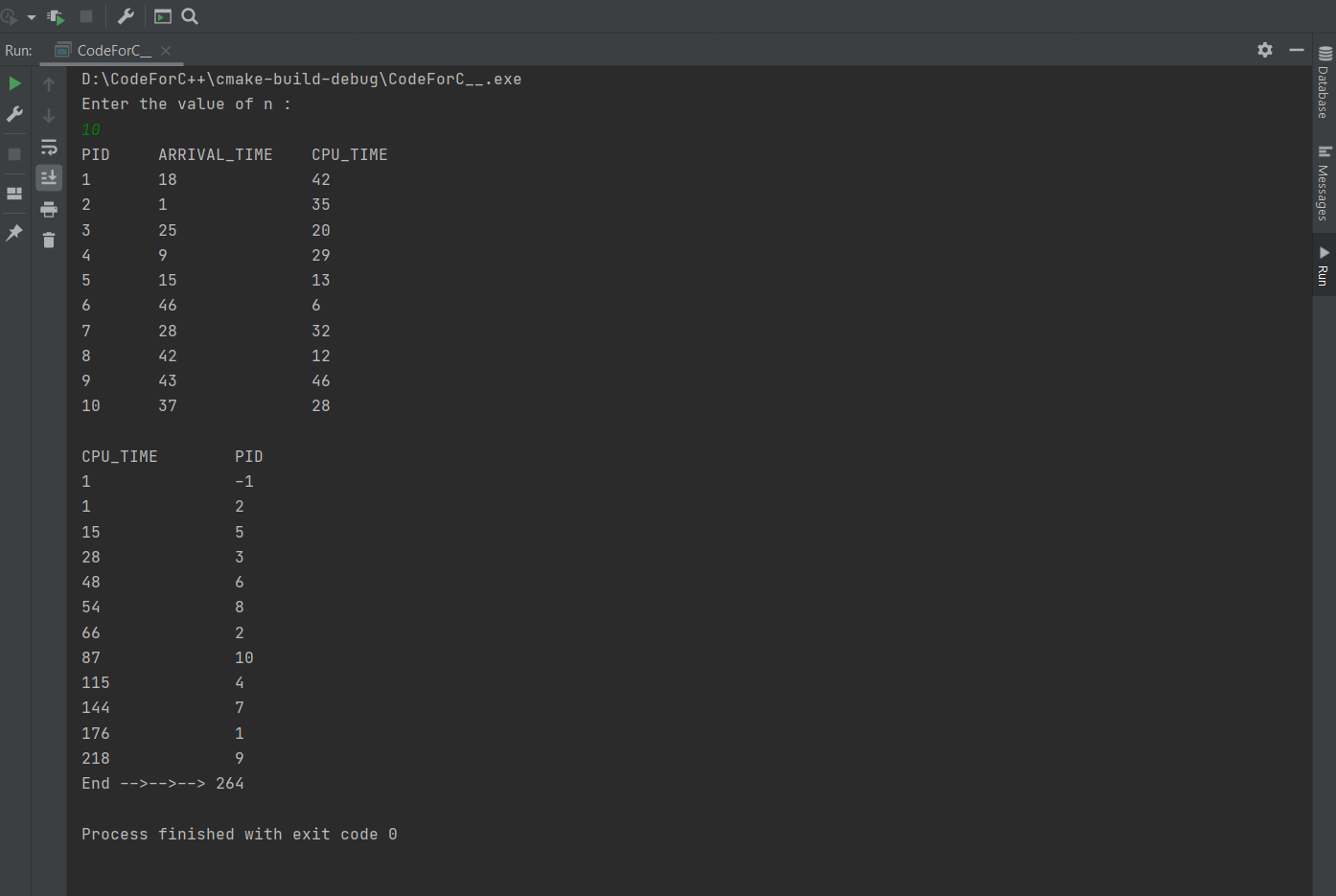
        cout << init[i].pID << '\t' << init[i].arrivalTime << "\t\t" << init[i].cpuTime << endl;

    }

    SRTF\_Algo(init);

}

**Output:-**

****