LAB Manual

|  |
| --- |
| **Name of Student: Vidhi Binwal PRN: 22070122249** |
| **Semester: IV Year AY 23-24** |
| **Subject Title: Operating Systems Lab** |
| **EXPERIMENT No: 11 Assignment No : 8** |
| **TITLE: Scheduling Algorithm** **DoP : 30/3/2024** |

***Aim:*** Implement C program demonstrate **Shortest Remaining Job algorithm**

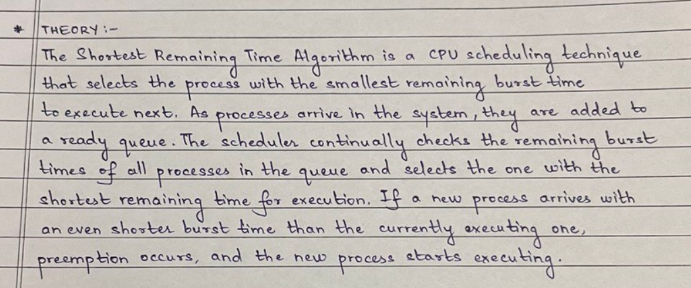
***Learning Outcomes:*** *1. To understand the* **scheduling algorithm**

*2. To Demonstrate the* Shortest Remaining Job algorithm

***Hardware/Software*:**

****

***Theory:***

******

***Algorithm:***

1. Initialize: Set the current time to 0 and the set of ready processes to empty.

2. Arrival of Process: As processes arrive in the system, add them to the ready queue.

3. Selecting the Next Process:

- If the ready queue is empty, the CPU remains idle.

- If the ready queue is not empty, find the process with the smallest remaining burst time.

4. Executing the Process:

- Execute the selected process for a predefined time quantum or until it completes its execution, whichever comes first.

5. Preemption:

- If a new process arrives with a smaller remaining burst time than the currently executing process, preempt the current process and execute the newly arrived process.

6. Repeat:

- Repeat steps 3-5 until all processes have completed their execution.

7. Completion:

- When a process completes execution, remove it from the system.

8. Calculate Metrics:

- Calculate performance metrics such as waiting time, turnaround time, and CPU utilization.

***Program:***

#include <stdio.h>

#include <stdbool.h>

struct Process {

    int id;

    int arrival\_time;

    int burst\_time;

    int remaining\_time;

    int start\_time;

    int completion\_time;

};

void srt(struct Process processes[], int n) {

    int current\_time = 0;

    bool is\_completed[n];

    int total\_completed = 0;

    for (int i = 0; i < n; i++)

        is\_completed[i] = false;

    while (total\_completed < n) {

        int shortest\_remaining = -1;

        int shortest\_burst = 999999;

        for (int i = 0; i < n; i++) {

            if (!is\_completed[i] && processes[i].arrival\_time <= current\_time &&

                processes[i].burst\_time < shortest\_burst) {

                shortest\_burst = processes[i].burst\_time;

                shortest\_remaining = i;

            }

        }

        if (shortest\_remaining == -1) {

            current\_time++;

            continue;

        }

        processes[shortest\_remaining].remaining\_time--;

        current\_time++;

        if (processes[shortest\_remaining].remaining\_time == 0) {

            processes[shortest\_remaining].completion\_time = current\_time;

            total\_completed++;

            is\_completed[shortest\_remaining] = true;

        }

    }

}

int main() {

    int n;

    printf("Enter the number of processes: ");

    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {

        printf("Enter arrival time and burst time for process %d: ", i + 1);

        scanf("%d %d", &processes[i].arrival\_time, &processes[i].burst\_time);

        processes[i].id = i + 1;

        processes[i].remaining\_time = processes[i].burst\_time;

    }

    printf("Shortest Remaining Time (SRT) scheduling:\n");

    srt(processes, n);

    int total\_waiting\_time = 0;

    int total\_turnaround\_time = 0;

    for (int i = 0; i < n; i++) {

        processes[i].start\_time = processes[i].completion\_time - processes[i].burst\_time;

        total\_waiting\_time += processes[i].start\_time - processes[i].arrival\_time;

        total\_turnaround\_time += processes[i].completion\_time - processes[i].arrival\_time;

    }

    double avg\_waiting\_time = (double) total\_waiting\_time / n;

    double avg\_turnaround\_time = (double) total\_turnaround\_time / n;

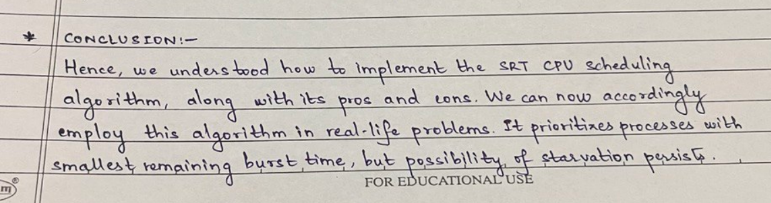
    printf("Average Waiting Time: %.2lf\n", avg\_waiting\_time);

    printf("Average Turnaround Time: %.2lf\n", avg\_turnaround\_time);

    return 0;

}

***Conclusion:***



***Output:***

