Ex no: 3 M. Rohith
Date: 03-04-2023 3122 21 5001 085

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)
SSN College of Engineering

Department of Computer Science and Engineering

UCS2412 – Operating Systems Laboratory

II Year CSE –A &B Section (IV Semester)

Academic Year 2022-23

Exercise – 3 – Scheduling Algorithms -FCFS, SJF

Lab Exercise 3 Implementation of Scheduling Algorithms (FCFS, SJF)

Aim:

To develop a C program to implement the Scheduling Algorithms.

Scenario:

Consider a hospital scenario, where the patients are waiting for doctor consultation. Identify the suitable scheduling algorithm for this scenario and develop a solution to schedule the patients with respect to their token number. Arrival time of the patient is used to decide the token number, consultation time with the doctor is the burst time.

a. Compute the waiting time, turnaround time and response time of each patient and tabulate the results with the Gantt chart.

In the above scenario, let us include the following cases that take less consultation time:

- Vaccination
- Showing test reports only
- Medical representatives

b. Identify the suitable scheduling algorithm for this scenario by giving preferences to the above cases and develop a solution to schedule the consultations.
c. Compute the waiting time, turnaround time and response time of each patient and tabulate the results with the Gantt chart.
d. Analyze and report the best scheduling algorithm based on the scheduling criteria.

M. Rohith
3122 21 5001 085

Program code using FCFS Scheduling:

```
//FCFS Scheduling
#include <stdio.h>

void print_gantt_chart(int *at, int *bt, int n, int *tat)
{
    int pid[n];
    for (int i = 0; i < n; i++)
    {
        pid[i] = i + 1;
    }
    int i, j;
    // print top bar
    printf(" ");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < bt[i]; j++)
        printf("--");
        printf(" ");
    }
    printf("\n|");

    // printing process id in the middle
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < bt[i] - 1; j++)
            printf(" ");
        printf("P%d", pid[i]);
        for (j = 0; j < bt[i] - 1; j++)
            printf(" ");
        printf("|");
    }
    printf("\n ");
    // printing bottom bar
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < bt[i]; j++)
            printf("--");
        printf(" ");
    }
```

M. Rohith
3122 21 5001 085

```
    printf("\n");

    // printing the time line
    printf("0");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < bt[i]; j++)
            printf(" ");
        if (tat[i] > 9)
            printf("\b"); // backspace : remove 1 space
        printf("%d", tat[i]);
    }
    printf("\n");
}

int main()
{
    int n, i, j, ct;
    float avwt = 0, avtat = 0;
    printf("Enter the number of patients: ");
    scanf("%d", &n);
    int bt[n], at[n], wt[n], tat[n], rt[n], st[n], et[n];
    for (i = 0; i < n; i++)
    {
        printf("Enter the arrival time and consultation time for
process %d: ", i + 1);
        scanf("%d %d", &at[i], &bt[i]);
    }
    for (i = 0; i < n; i++)
    {
        if (i == 0)
        {
           st[i] = at[i];
        }
        else
        {
            if (at[i] > et[i - 1])
            {
                st[i] = at[i];
            }
            else
            {
```

M. Rohith
3122 21 5001 085

```
                st[i] = et[i - 1];
            }
        }
        et[i] = st[i] + bt[i];
        wt[i] = st[i] - at[i];
        tat[i] = et[i] - at[i];
        avwt += wt[i];
        avtat += tat[i];
    }
    print_gantt_chart(at, bt, n, tat);
    printf("\nPatient\tAT\tBT\tST\tET\tWT\tTAT\tRT\n");
    for (i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[i],
bt[i], st[i], et[i], wt[i], tat[i], rt[i]);
    }
    avwt /= n;
    avtat /= n;
    printf("\nAverage waiting time of patients= %.2f", avwt);
    printf("\nAverage turnaround time patients= %.2f\n", avtat);
    return 0;
}
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Operating Systems lab\Exercise-3> gcc a.c -o run
PS C:\Rohith\Backup\Desktop\SEM 4\Operating Systems lab\Exercise-3> ./run
Enter the number of patients: 5
Enter the arrival time and consultation time for process 1: 0 5
Enter the arrival time and consultation time for process 2: 0 3
Enter the arrival time and consultation time for process 3: 3 4
Enter the arrival time and consultation time for process 4: 10 1
Enter the arrival time and consultation time for process 5: 12 3
 ---------- ------ -------- -- ------
|    P1    |  P2  |   P3   |P4|  P5  |
 ---------- ------ -------- -- ------
0     5   8    9 3   4

Patient AT      BT      ST      ET      WT      TAT     RT
P1      0       5       0       5       0       5       12068272
P2      0       3       5       8       5       8       6422020
P3      3       4       8       12      5       9       2002618229
P4      10      1       12      13      2       3       2002949172
P5      12      3       13      16      1       4       32

Average waiting time of patients= 2.60
Average turnaround time patients= 5.80
```

M. Rohith
3122 21 5001 085

Program code using Non Preemptive Shortest Job First:

```
//Non Preemptive SJF Scheduling
#include <stdio.h>

int main()
{
    int n, i, j, ct, min, minimumat = 9999999;
    float averagewait = 0, averageturn = 0;
    printf("Enter the number of patients: ");
    scanf("%d", &n);
    int bt[n], at[n], wt[n], tat[n], rt[n], st[n], et[n], done[n];
    for (i = 0; i < n; i++)
    {
        printf("Enter the arrival time and burst time for each
patient %d: ", i + 1);
        scanf("%d%d", &at[i], &bt[i]);
        if (at[i] < minimumat)
        {
            minimumat = at[i];
        }
        done[i] = 0;
    }
    ct = minimumat;
    while (1)
    {
        min = -1;
        for (i = 0; i < n; i++)
        {
            if (!done[i] && at[i] <= ct && (min == -1 || bt[i] <
bt[min]))
            {
                min = i;
            }
        }
        if (min == -1)
        {
            break;
        }
        if (done[min] == 0)
        {
```

M. Rohith
3122 21 5001 085

```
            if (ct < at[min])
            {
                ct = at[min];
            }
            st[min] = ct;
            et[min] = ct + bt[min];
            ct = et[min];
            wt[min] = st[min] - at[min];
            tat[min] = et[min] - at[min];
            rt[min] = st[min] - at[min];
            averagewait += wt[min];
            averageturn += tat[min];
            done[min] = 1;
        }
     }
    printf("\nProcess\tAT\tBT\tST\tET\tWT\tTAT\tRT\n");
    for (i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[i],
bt[i], st[i], et[i], wt[i], tat[i], rt[i]);
    }
    averagewait /= n;
    averageturn /= n;
    printf("\nAverage waiting time = %.2f", averagewait);
    printf("\nAverage turnaround time = %.2f\n", averageturn);
    return 0;
}
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Operating Systems lab\Exercise-3> gcc b.c -o run
PS C:\Rohith\Backup\Desktop\SEM 4\Operating Systems lab\Exercise-3> ./run
Enter the number of patients: 5
Enter the arrival time and burst time for each patient 1: 0 10
Enter the arrival time and burst time for each patient 2: 3 7
Enter the arrival time and burst time for each patient 3: 4 4
Enter the arrival time and burst time for each patient 4: 5 1
Enter the arrival time and burst time for each patient 5: 7 9

Process AT      BT      ST      ET      WT      TAT     RT
P1      0       10      0       10      0       10      0
P2      3       7       15      22      12      19      12
P3      4       4       11      15      7       11      7
P4      5       1       10      11      5       6       5
P5      7       9       22      31      15      24      15

Average waiting time = 7.80
Average turnaround time = 14.00
```

Ex no: 3 M. Rohith
Date: 03-04-2023 3122 21 5001 085

FCFS Algorithm:

- Considering the hospital scenario, the patients are scheduled to meet the doctor with respect to their token number.
- Since the arrival time of the patient is used to decide the token number FCFS Algorithm best suits this case as the FCFS algorithm schedules the job according to the first come first serve model that is the patient who arrives first will be scheduled with the doctor first.

Non-Preemptive SJF Algorithm:

- In the above hospital scenario when various cases such as vaccination, showing test reports only, medical representatives, etc are included in the manner that it takes less consultation time, Non-Preemptive SJF Algorithm best suits for this purpose.
- Since the consultation time with the doctor is the burst time , the job with shortest job is given higher priority and hence as the name SJF suggests Shortest Job First is best fit for this case.

Learning Outcomes:

- Application of FCFS Algorithm and Non_Preemptive SJF Algorithm
- Computation of waiting time, turnaround time, starting and end time of the process along with their averages.
- Limitations of FCFS and Non preemptive SJF Algorithm.