Lab Assignment-09

ROLL: 2005535 | NAME: SAHIL SINGH | DATE: 15/03/22

QUES 1: WAP in C to implement the FCFS scheduling algorithm without considering the arrival time.

SOLUTION:

#include <stdio.h>

*void* displayTable(*int* *n*, *int* *wt*[], *int* *at2*[], *int* *bt*[], *int* *tat*[])

{

*float* total\_tat = 0, total\_wt = 0;

    printf("\nProcesses\tArrival time\tBurst time\tWaiting time\tTurn around time\n");

    for (*int* i = 0; i < *n*; i++)

    {

        total\_wt = total\_wt + *wt*[i];

        total\_tat = total\_tat + *tat*[i];

        printf("    P%d\t\t", i);

        printf("    %d\t\t", *at2*[i]);

        printf("    %d\t\t", *bt*[i]);

        printf("    %d\t\t", *wt*[i]);

        printf("    %d\t\t\n", *tat*[i]);

    }

    printf("\nAverage waiting time = %.2f\n", (total\_wt / (*float*)*n*));

    printf("Average turn around time = %.2f\n", (total\_tat / (*float*)*n*));

}

*void* FCFS(*int* *n*, *int* *at*[], *int* *at2*[], *int* *bt*[])

{

*int* start, pos, max = 0, min, k = 0;

*int* seq[10], re[10], ex[10];

*int* wt[10], tat[10];

    start = *at*[0];

    for (*int* i = 1; i < *n*; i++)

    {

        if (start > *at*[i])

        {

            start = *at*[i];

        }

    }

    printf("\nSequence of execution: \n");

    for (*int* i = 0; i < *n*; i++)

    {

        if (max < *at*[i])

        {

            max = *at*[i];

        }

    }

    max = max + 1;

    for (*int* i = 0; i < *n*; i++, k++)

    {

        min = max;

        for (*int* j = 0; j < *n*; j++)

        {

            if (*at*[j] != -1)

            {

                if (*at*[j] < min)

                {

                    min = *at*[j];

                    pos = j;

                }

            }

        }

        printf("[P%d]  ", pos);

        seq[k] = pos;

        if (start < *at*[pos])

        {

            re[pos] = start;

            start = *at*[pos];

            start += *bt*[pos];

*at*[pos] = -1;

            ex[pos] = start;

        }

        else

        {

            re[pos] = start;

            start += *bt*[pos];

*at*[pos] = -1;

            ex[pos] = start;

        }

    }

    printf("\n");

    for (*int* i = 0; i < *n*; i++)

    {

        tat[i] = ex[i] - *at2*[i];

        wt[i] = tat[i] - *bt*[i];

    }

    displayTable(*n*, wt, *at2*, *bt*, tat);

}

*void* getProcessesInput(*int* *n*, *int* *at*[], *int* *at2*[], *int* *bt*[])

{

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

    scanf("%d", &*n*);

    for (*int* i = 0; i < *n*; i++)

    {

*at*[i] = 0;

*at2*[i] = *at*[i];

    }

    printf("Enter burst time for processes: ");

    for (*int* i = 0; i < *n*; i++)

    {

        scanf("%d", &*bt*[i]);

    }

    FCFS(*n*, *at*, *at2*, *bt*);

}

*int* main()

{

*int* n, at[10], at2[10], bt[10];

    getProcessesInput(n, at, at2, bt);

}

OUTPUT:

Enter the number of process: 3

Enter burst time for processes: 24 3 3

Sequence of execution:

[P0]  [P1]  [P2]

Processes       Arrival time    Burst time      Waiting time    Turn around time

    P0              0               24              0               24

    P1              0               3               24              27

    P2              0               3               27              30

Average waiting time = 17.00

Average turn around time = 27.00

QUES 2: WAP in C to implement the FCFS scheduling algorithm with considering the arrival time.

SOLUTION:

#include <stdio.h>

*void* displayTable(*int* *n*, *int* *wt*[], *int* *at2*[], *int* *bt*[], *int* *tat*[])

{

*float* total\_tat = 0, total\_wt = 0;

    printf("\nProcesses\tArrival time\tBurst time\tWaiting time\tTurn around time\n");

    for (*int* i = 0; i < *n*; i++)

    {

        total\_wt = total\_wt + *wt*[i];

        total\_tat = total\_tat + *tat*[i];

        printf("    P%d\t\t", i);

        printf("    %d\t\t", *at2*[i]);

        printf("    %d\t\t", *bt*[i]);

        printf("    %d\t\t", *wt*[i]);

        printf("    %d\t\t\n", *tat*[i]);

    }

    printf("\nAverage waiting time = %.2f\n", (total\_wt / (*float*)*n*));

    printf("Average turn around time = %.2f\n", (total\_tat / (*float*)*n*));

}

*void* FCFS(*int* *n*, *int* *at*[], *int* *at2*[], *int* *bt*[])

{

*int* start, pos, max = 0, min, k = 0;

*int* seq[10], re[10], ex[10];

*int* wt[10], tat[10];

    start = *at*[0];

    for (*int* i = 1; i < *n*; i++)

    {

        if (start > *at*[i])

        {

            start = *at*[i];

        }

    }

    printf("\nSequence of execution: \n");

    for (*int* i = 0; i < *n*; i++)

    {

        if (max < *at*[i])

        {

            max = *at*[i];

        }

    }

    max = max + 1;

    for (*int* i = 0; i < *n*; i++, k++)

    {

        min = max;

        for (*int* j = 0; j < *n*; j++)

        {

            if (*at*[j] != -1)

            {

                if (*at*[j] < min)

                {

                    min = *at*[j];

                    pos = j;

                }

            }

        }

        printf("[P%d]  ", pos);

        seq[k] = pos;

        if (start < *at*[pos])

        {

            re[pos] = start;

            start = *at*[pos];

            start += *bt*[pos];

*at*[pos] = -1;

            ex[pos] = start;

        }

        else

        {

            re[pos] = start;

            start += *bt*[pos];

*at*[pos] = -1;

            ex[pos] = start;

        }

    }

    printf("\n");

    for (*int* i = 0; i < *n*; i++)

    {

        tat[i] = ex[i] - *at2*[i];

        wt[i] = tat[i] - *bt*[i];

    }

    displayTable(*n*, wt, *at2*, *bt*, tat);

}

*void* getProcessesInput(*int* *n*, *int* *at*[], *int* *at2*[], *int* *bt*[])

{

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

    scanf("%d", &*n*);

    printf("Enter arrival time for processes: ");

    for (*int* i = 0; i < *n*; i++)

    {

        scanf("%d", &*at*[i]);

*at2*[i] = *at*[i];

    }

    printf("Enter burst time for processes: ");

    for (*int* i = 0; i < *n*; i++)

    {

        scanf("%d", &*bt*[i]);

    }

    FCFS(*n*, *at*, *at2*, *bt*);

}

*int* main()

{

*int* n, at[10], at2[10], bt[10];

    getProcessesInput(n, at, at2, bt);

}

OUTPUT:

Enter the number of process: 5

Enter arrival time for processes: 2 5 1 0 4

Enter burst time for processes: 6 2 8 3 4

Sequence of execution:

[P3]  [P2]  [P0]  [P4]  [P1]

Processes       Arrival time    Burst time      Waiting time    Turn around time

    P0              2               6               9               15

    P1              5               2               16              18

    P2              1               8               2               10

    P3              0               3               0               3

    P4              4               4               13              17

Average waiting time = 8.00

Average turn around time = 12.60

-----------------------------------------------------------------------------------------