Sowmya.R

185001159

UCS1411 - OPERATING SYSTEMS LAB

Lab Exercise:3

Implementation of CPU Scheduling Policies: FCFS and SJF (Nonpreemptive and Preemptive)

```c
// Program to perform fcfs,SJF(preemptive and non preemptive scheduling)

#include<stdio.h>
#include <stdlib.h>
#include<string.h>

//DEFN. of Process structure
typedef struct Process
{
    char pid[2];
    int at;
    int bt;
    int tt;
    int wt;
    int rt;
    int flag;
    int visited;

}Process;


void sjf_nonpreemptive(Process *p[20],int n);
void fcfs_scheduling(Process *p[20],int n);
void sjf_preemptive(Process *p[20],int n);


void main()
{
    Process *p[20],*ptr[20];
    int ch,n,i;


    for(i=0;i<20;i++)
    {

        p[i]=(Process*)malloc(sizeof(Process));
        ptr[i]=(Process*)malloc(sizeof(Process));
```

```
        ptr[i]->flag=0;
        p[i]->flag=0;
        p[i]->visited=0;
    }


    printf("MENU\n");
    printf("1.FCFS\n");
    printf("2.SJF\n");
    printf("Enter choice:");
    scanf("%d",&ch);

    //Read process details

    printf("\nEnter the no. of processes:");
    scanf("%d",&n);


    for(i=0;i<n;i++)
    {
        printf("\nEnter Process ID:");
        scanf("%s",p[i]->pid);
        strcpy(ptr[i]->pid,p[i]->pid);
        printf("Enter arrival time:");
        scanf("%d",&p[i]->at);
        ptr[i]->at=p[i]->at;
        printf("Enter burst time:");
        scanf("%d",&p[i]->bt);
        ptr[i]->bt=p[i]->bt;

        p[i]->rt=0;
        p[i]->tt=0;
        p[i]->wt=0;

        ptr[i]->rt=0;
        ptr[i]->tt=0;
        ptr[i]->wt=0;

    }


    switch(ch)
    {
        case 1:fcfs_scheduling(p,n);
                break;
        case 2:sjf_nonpreemptive(p,n);
                sjf_preemptive(ptr,n);
                break;
    }
```

```
}

Process* shortest_at(Process *p[20],int n)//Sorts the processes acc. to increa
sing arrival time and returns the processes in that order
{

    int i,j;
    Process *temp;
    Process *p1[20];
    for(i=0;i<n;i++)
    {
        p1[i]=p[i];
    }

    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(p1[i]->at>p1[j]->at)
            {
                temp=p1[i];
                p1[i]=p1[j];
                p1[j]=temp;
            }
        }
    }
    i=0;
    while(p1[i]->flag!=0)
    {
        i++;
    }

    return p1[i];
}

int remaining_process(Process *p[20],int n)//returns 1 if there are processes
yet to finish
{
    int i;
    for(i=0;i<n;i++)
    {
        if(p[i]->flag==0)//flag is set to 1 if process is finished
            return 1;
    }
    return 0;
}
void printChart(Process *p[20],int n)// function to print Gantt chart
```

```
{

    int i,t;
    int time=0,total_time=0;

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

        p[i]->flag=0;

    }
    printf("-------------------------------------------------------\n");
    for(i=0;remaining_process(p,n)!=0;i++)
    {

        if(time<p[i]->at)
        {
            printf("   %s    ","idle");//Print IDLE when CPU is idle
            t=(p[i]->at-time);
            time+=t;
            continue;
        }

        printf("   %s    ",p[i]->pid);
        time+=p[i]->bt;
        p[i]->flag=1;
    }
    printf("\n");
    printf("-------------------------------------------------------\n");
    for(i=0;i<n;i++)
    {

        p[i]->flag=0;

    }
    time=0;

    for(i=0;remaining_process(p,n)!=0;i++)
    {
        if(time<p[i]->at)
        {
            t=(p[i]->at-time);
            time+=t;
        }
        printf("%d       ",time);
        time+=p[i]->bt;
        p[i]->flag=1;
    }
```

```
    printf("%d",time);
}

void wait_time(Process *p[20],int n)// function to calculate wait time for non
 preemptive scheduling
{
    int i,time=0,t;

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

        p[i]->flag=0;

    }

    for(i=0;remaining_process(p,n)!=0;i++)
    {
        p[i]->wt=time-p[i]->at;
        if(time<p[i]->at)
        {
            t=(p[i]->at-time);
            time+=t;
        }

        time+=p[i]->bt;
        p[i]->flag=1;
    }
}

void turaround_time(Process *p[20],int n)// function to calculate turnaround t
ime
{
    int i,time=0;

    for(i=0;i<n;i++)
    {
        p[i]->tt=p[i]->wt+p[i]->bt;
    }

}

void response_time(Process *p[20],int n)// function to calculate response time
 for non preemptive scheduling
{
    int i;
    for(i=0;i<n;i++)
    {
        p[i]->rt=p[i]->wt;
```

```
    }
}
void print_Table(Process *p[20],int n)// function to print the table containin
g wait time, turaround time and response time
{

    int i;
    float avg_wt=0,avg_rt=0,avg_tt=0;

    turaround_time(p,n);



    printf("\n\n-----------------------------------------------------------------
------------------------------\n");
    printf("Process ID  Arrival Time  Burst Time   TurnaroundTime   Waiting Ti
me   Response Time\n");
    printf("-----------------------------------------------------------------
--------------------------\n");
    for(i=0;i<n;i++)
    {
        printf("%-11s%14d%13d%17d%15d%13d\n",p[i]->pid,p[i]->at,p[i]->bt,p[i]-
>tt,p[i]->wt,p[i]->rt);
        avg_wt+=p[i]->wt;
        avg_rt+=p[i]->rt;
        avg_tt+=p[i]->tt;
    }
    avg_wt/=n;
    avg_rt/=n;
    avg_tt/=n;
    printf("-----------------------------------------------------------------
--------------------------\n");
    printf("                                Average           %.2f
 %.2f          %.2f\n",avg_tt,avg_wt,avg_rt);
}

void fcfs_scheduling(Process *p[20],int n)//program to sort the processes acc.
 to fcfs algo.
{
    int i,j;
    Process *temp;

    printf("FCFS SCHEDULING\n");
    // sort acc. to arrival time
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(p[i]->at>p[j]->at)
```

```
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
            }
        }
    }

    int bTime,aTime;
    //sorting based on burst time for processes having same arrival time

    for(i=0,j=1;i<n-1;i++,j++)
    {
        aTime = p[i]->at;
        bTime = p[i]->bt;

        if(aTime == p[j]->at && bTime > p[j]->bt)
        {
            temp=p[j];
            p[j]=p[i];
            p[i]=temp;
        }
    }

    printChart(p,n);
    wait_time(p,n);
    response_time(p,n);
    print_Table(p,n);

}

void sjf_preemptive(Process *p[20],int n)
{
int burst_time[n];
int gantt_time[30];
int process[30];
int gantt_chart[30];
int count=0,c=0;
int i,j,time;

    // Copy the burst time into burst_time[]
    for (i = 0; i < n; i++)
        burst_time[i] = p[i]->bt;

    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    int check = 0;
```

```
    // process until all proesses gets
    // completed
    while (complete != n) {

        // Find process with minimum
        // remaining time among the
        // process that arrives till the
        // current time`
        for (int j = 0; j < n; j++) {
            if ((p[j]-
>at<= t) && (burst_time[j] < minm) && burst_time[j] > 0)
            {
                if(p[j]->visited!=1)
                {
                    p[j]->rt=t-p[j]->at;
                }
                p[j]->visited=1;
                process[count++]=j;
                minm = burst_time[j];
                shortest = j;
                check = 1;

            }
        }

        if (check == 0)
        {
            t++;
            continue;
        }

        // Reduce remaining time by one
        burst_time[shortest]--;

        // Update minimum
        minm = burst_time[shortest];
        if (minm == 0)
            minm = INT_MAX;

        // If a process gets completely
        // executed
        if (burst_time[shortest] == 0) {

            // Increment complete
            complete++;
            check = 0;

            // Find finish time of current
```

```
            // process
            finish_time = t + 1;

            // Calculate waiting time
            p[shortest]->wt = finish_time - p[shortest]->bt - p[shortest]-
>at;

            if (p[shortest]->wt < 0)
                p[shortest]->wt = 0;
        }
        // Increment time
        t++;
    }


    int flag=0;
    int curr_time=0;
    for(i=0;i<count-1;i++)
    {

        if(process[i]==process[i-1] && i!=0)
            continue;
        j=i+1;
        time=1;
        while(process[i]==process[j] && j<count)
        {

            j++;
            time++;
        }

        curr_time+=time;
        gantt_chart[c]=process[i];
        gantt_time[c++]=curr_time;


    }

    int index;
    printf("SJF PREEMPTIVE SCHEDULING\n");
    printf("------------------------------------------------------------\n");

    //Printing the gantt chart
    for(i=0;i<c;i++)
    {
        index=gantt_chart[i];
        printf("   %s    ",p[index]->pid);
    }
    printf("\n");
```

```
    printf("0      ");
    for(i=0;i<c;i++)
    {
        printf("%d       ",gantt_time[i]);
    }


    print_Table(p,n);




}

void sjf_nonpreemptive(Process *p[20],int n)
{

    int time=0,j=0,i,k,t;
    Process *ptr[20];


    Process *temp;

    printf("SJF NON PREEMPTIVE SCHEDULING\n");

    //sort acc. to burst time
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(p[i]->bt>p[j]->bt)
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
            }
        }
    }

    for(i=0;i<20;i++)
    {

        ptr[i]=(Process*)malloc(sizeof(Process));
        ptr[i]->flag=0;

    }

    j=0;
```

```
    //sort based on arrival time if no processes have arrived at current time
or when two processes have same burst time

    for(i=0;remaining_process(p,n)!=0;)
    {

        if(time<p[i]->at)
        {

            ptr[j]=shortest_at(p,n);


            for(k=0;k<n;k++)
            {
                if(strcmp(ptr[j]->pid,p[k]->pid)==0)
                {
                    p[k]->flag=1;
                    break;
                }
            }

            time+=ptr[j]->bt;

            if(time<ptr[j]->at)
            {
                t=(ptr[j]->at-time);
                time+=t;

            }


            j++;
        }

        else
        {

            if(p[i]->flag==0)
            {

                ptr[j++]=p[i];
                p[i]->flag=1;
                time+=p[i]->bt;

            }
            i++;
        }
```

```
    }

    printChart(ptr,n);
    wait_time(p,n);
    response_time(p,n);
    print_Table(ptr,n);



}


/*
C:\Users\Sowmya\Desktop\Sowmya\Lab\OS\A3>gcc -o a scheduling.c

C:\Users\Sowmya\Desktop\Sowmya\Lab\OS\A3>a
MENU
1.FCFS
2.SJF
Enter choice:1

Enter the no. of processes:5

Enter Process ID:p1
Enter arrival time:0
Enter burst time:8

Enter Process ID:p2
Enter arrival time:1
Enter burst time:6

Enter Process ID:p3
Enter arrival time:2
Enter burst time:1

Enter Process ID:p4
Enter arrival time:3
Enter burst time:9

Enter Process ID:p5
Enter arrival time:4
Enter burst time:3
FCFS SCHEDULING
------------------------------------------------------------
   p1       p2       p3       p4       p5
------------------------------------------------------------
0       8       14       15       24       27
```

```
--------------------------------------------------------------------------------
--------------
Process ID      Arrival Time  Burst Time   TurnaroundTime   Waiting Time   Res
ponse Time
--------------------------------------------------------------------------------
--------------
p1                      0           8                8              0
    0
p2                      1           6               13              7
    7
p3                      2           1               13             12
   12
p4                      3           9               21             12
   12
p5                      4           3               23             20
   20
--------------------------------------------------------------------------------
--------------
                            Average           15.60              10.20
     10.20


MENU
1->FCFS
2->SJF
Enter choice:2

Enter the no-> of processes:5

Enter Process ID:p1
Enter arrival time:0
Enter burst time:8

Enter Process ID:p2
Enter arrival time:1
Enter burst time:6

Enter Process ID:p3
Enter arrival time:2
Enter burst time:1

Enter Process ID:p4
Enter arrival time:3
Enter burst time:8

Enter Process ID:p5
Enter arrival time:4
Enter burst time:3
SJF NON PREEMPTIVE SCHEDULING
```

```
-----------------------------------------------------
   p1       p3       p5       p2       p4
-----------------------------------------------------
0       8       9       12       18       27

-------------------------------------------------------------------------------
---------------
Process ID      Arrival Time  Burst Time   TurnaroundTime   Waiting Time   Res
ponse Time
-------------------------------------------------------------------------------
---------------
p1                      0            8                8              0
    0
p3                      2            1                7              6
    6
p5                      4            3                8              5
    5
p2                      1            6               17             11
   11
p4                      3            8               23             15
   15
-------------------------------------------------------------------------------
---------------
                            Average            12.60             7.40
    7.40

SJF PREEMPTIVE SCHEDULING
-----------------------------------------------------
   p1       p2       p3      p2       p5      p2       p1       p4
0       1       2       3       4        7       11       18        27

-------------------------------------------------------------------------------
---------------
Process ID      Arrival Time  Burst Time   TurnaroundTime   Waiting Time   Res
ponse Time
-------------------------------------------------------------------------------
---------------
p1                      0            8               18             10
    0
p2                      1            6               10              4
    0
p3                      2            1                1              0
    0
p4                      3            9               24             15
   15
p5                      4            3                3              0
    0
```

```
--------------------------------------------------------------------------------
---------------
                              Average            11.20               5.80
    3.00

*/
```