2024/05/15 Thursday

Lab->3

* Write a c program to simulate the following non-preemptive cpu scheduling algorithm to find turnaround and waiting time.

① FCFS ② SJF(non-preemptive)

```
#include<stdio.h>
int main(){
    int process[10][2];
    int n;
    printf("Enter the total no. of process\n");
    scanf("%d",&n);
    printf("Enter the respective burst and arrival time\n");
    for(int i=0;i<n;i++){
        printf("Enter the burst time and arrival time of
                %d process:",i+1);
        scanf("%d%d",&process[i][0],&process[i][1]);
    }
//completion time
    int completion[10];
    completion[0]=process[0][0];
    for(int i=1;i<n;i++){
        completion[i]=completion[i-1]+process[i][0];
    }
//turnaround time
    int turnaround[10];
    int turn_sum=0;
    printf("The turn around time of n process are:\n");
    int idea=0;
    for(int i=0;i<n;i++){
        turnaround[i]=completion[i]-process[i][1]+idea;
        if(i<n-1&&completion[i]<process[i+1][1])idea
            =process[i+1][1]-completion[i];
        printf("%d:%d\n",i+1,turn-around[i]);
        turn_sum+=turn_around[i];
    }
```

```
printf("The average turn-around time of n processes:
    %f\n", (float) turn_sum/n);

//waiting time
    int waiting[10];
    int wait_sum=0;
    printf("The waiting time of n processes:\n");
    for(int i=0;i<n;i++){
        waiting[i]= turn_around[i]-process[i][0];
        printf("P%d:%d\n",i+1, waiting[i]);
        wait_sum+=waiting[i];
    }
printf("The average waiting time of n processes:
        %f\n", (float) wait_sum/n);
printf("Throughput:%f", (float)completion[n-1]/n);
}
```

Output:

Enter the total no of process: 4

Enter the respective burst and arrival time

3	0
6	1
4	4
2	6

The turnaround time of n processes:

P1:3

P2:8

P3:9

The average turn.around time of n process: 7.2500

The waiting times of n processe:

P1: 0
P2: 2
P3: 5
P4: 7

The avg. waiting-time of n processe: 3.500

Throughput: 3.7560

SJF

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void swap(int *x,int *y)
{
    int temp=*x;
    *x=*y;
    *y=temp;
}
void sortat(int p[],int at[],int bt[],int n)
{
    int i,j;
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(at[i]>at[j])
            {
                swap(&p[i],&p[j]);
                swap(&at[i],&at[j]);
                swap(&bt[i],&bt[j]);
            }
            else if(at[i]==at[j])
            {
                if(bt[i]>bt[j])
                {
                    swap(&p[i],&p[j]);
                    swap(&at[i],&at[j]);
                    swap(&bt[i],&bt[j]);
                }
            }
        }
    }
}
void tatwt(int ct[],int at[],int bt[],int tat[],int wt[],
           int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        tat[i]=ct[i]-at[i];
        wt[i]=tat[i]-bt[i];
```

```
}
}
int main()
{ int *p, *at, *bt, *bt1, *wt, *ct, pos, i, j, min=1000, n;
  float awt=0, atat=0;
  printf("\n enter the number of process:");
  scanf("%d", &n);
  p=(int*) malloc(n*sizeof(int));
  at=(int*) malloc(n*sizeof(int));
  bt=(int*) malloc(n*sizeof(int));
  ct=(int*) malloc(n*sizeof(int));
  wt=(int*) malloc(n*sizeof(int));
  tat=(int*) malloc(n*sizeof(int));
  printf("enter the process\n");
  for(i=0; i<n; i++)
  {
    scanf("%d", &p[i]);
  }
  printf("enter the arrival time\n");
  for(i=0; i<n; i++)
  {
    scanf("%d", &at[i]);
  }
  printf("enter the burst time\n");
  for(i=0; i<n; i++)
  {
    scanf("%d", &bt[i]);
  }
  sort(p, at, bt, n);
  ct[0]=at[0]+bt[0];
  for(i=1; i<n; i++)
  {
    for(j=i; j<n; j++)
    {
      if(at[j]<=ct[i-1])
      {
        if(bt[j]<min)
        { min=bt[j];
```

```
        }
      }
    }
  swap(&p[i],&p[pos]);
  swap(&at[i],&at[pos]);
  swap(&bt[i],&bt[pos]);
  min=1000;
  ct[i]=ct[i-1]+bt[i];
}
tatwt(ct,at,bt,tat,wt,n);
printf("\n\nPt\tAt\tBt\tCt\tTat\tWt\n");
for(i=0;i<n;i++)
{
  printf("\n%d\t%d\t%d\t%d\t%d\t%d",p[i],at[i],bt[i],
      tat[i],wt[i]);
  }
  for(i=0;i<n;i++)
{
  atat+=tat[i];
  awt+=wt[i];
  }
  atat=atat/n;
  awt=awt/n;
printf("\n\n avg tat=%.2f\t avg wt=%.2f",atat,awt);
return 0;
}
```

Output:

```
enter the number of process->4
enter the process.
1
2
3
4
enter the arrival tim
 0
```

1
4
6

enter the burst time

3
6
4
2

p	at	bt	ct	tat	wt
1	0	3	3	3	0
2	1	6	9	8	2
3	6	2	11	5	3
4	4	4	15	11	7

avg tat = 6.75

avg wt = 3.0.

16/5/2024