2024/06/12 Wednesday Lab > 6

1) Write a C program to simulate Real time cpu scheduling

a) Rate-Monotonic

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void
sort(int proc[],int b[],int p[],int n)
{
    int temp=0;
    for(int i=0;i<n;i++)
    {
        for(int j=i;j<n;j++)
        {
            if(p[j]<p[i])
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
                temp=b[j];
                b[j]=b[i];
                b[i]=temp;
                proc[i]=proc[j];
                proc[j]=temp;
            }
        }
    }
}
int
gcd(int a,int b)
{
    int r;
    while(b>0)
    {
        r=a%b;
        a=b;
```

```
return a;
}
int
lcmul(int p[], int n)
{
  int lcm=p[0];
  for(int i=1; i<n; i++)
    {
      lcm=(lcm*p[i])/gcd(lcm,p[i]);
    }
  return lcm;
}
void main()
{int n;
 printf("enter the no. of processes:");
 scanf("%d",&n);
 int proc[n],b[n],pt[n],tem[n];
 printf("Enter the CPU burst times:\n");
 for(int i=0;i<n;i++);
   {
    scanf("%d",&b[i]);
    tem[i]=b[i];
   }
 printf("enter the time periods:\n");
 for(int i=0; i<n; i++)
 scanf("%d",&pt[i]);
 for(int i=0;i<n;i++)
  proc[i]=i+1;
 sort(proc,b,pt,n);
  int l= lcmul(pt,n);
 printf("Lcm=%d\n",l);
 printf("P/Rate monotone scheduling:\n");
 printf("PID\t Burst\t period\n");
  for(int i=0;i<n;i++);
   printf("%d\t%d\t%d\t%d\n",proc[i],b[i],pt[i]);
```

```
double sum=0.0;
for(int i=0;i<n;i++)
{
sum+=(double)b[i]/p[i];
}
double rhs=n*(pow(2.0,(1.0/n))-1.0);
printf("%f<=%f=>%s\n",sum,rhs,(sum<=rhs)?
        "true":"false");
if(sum>rhs)
exit(0);
printf("Scheduling occurs for rate monotonic\n\n",1);
//RMS
int time=0,prev=0,x=0;
while(time<l)
{
int f=0;
for(int i=0;i<n;i++)
{
if(time%p[i]==0)
rem[i]=b[i];
if(rem[i]>0)
{
if(prev!=proc[i])
{
printf("%d:%d occurs:process%d runn
in",time,proc[i]);
prev=proc[i];
}
rem[i]--;
f=1;
break;
x=0
}}
if(!f)
{if(x!=1)
{printf("%d:CPU is idle\n",time);
```

```
    }
  }
  timeff;
}
}
```

output:

```
Enter the number of processes: 4
Enter the Cpu burst times:
1
2
3
4
Enter the time period:
1
5
6
3
```

LCM = 30

Rate monotone scheduling:

PID	Burst	period
1	1	1
4	4	3
2	2	5
3	3	6

3.233 <= 0.75682 => False

Sun
4/27/24

b) Earliest Deadline First

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void sort(int proc[],int d[],int b[],int pt[],int n)
{
    int temp=0;
    for(int i=0;i<n;i++)
    {
        for(int j=i;j<n;j++)
        {
            if(d[j]<d[i])
            {
                temp=d[j];
                d[j]=d[i];
                d[i]=temp;
                temp=pt[i];
                pt[i]=pt[j];
                pt[j]=temp;
                temp=b[j];
                b[j]=b[i];
                b[i]=temp;
                proc[i]=proc[j];
                proc[j]=temp;
            }
        }
    }
}

int gcd(int a,int b)
{
    int r;
    while(b>0)
    {
        r=a%b;
        a=b;
        b=r;
    }
    return a;
}
int
```

```
lcmul(int p[], int n)
{
 int lcm = p[0];
 for(int i=1; i<n; i++)
  {lcm=(lcm*p[i])/gcd(lcm,p[i]);
   }
 return lcm;
}
void main()
{
 int n;
 printf("Enter the no. of processes:");
  scanf("%d",&n);
  int proc[n], b[n], pt[n], d[n], rem[n];
  printf("Enter the CPU burst times:\n");
   for(int i=0; i<n; i++)
     { scanf("%d", &b[i]);
       rem[i]=b[i];
     }
 printf("Enter the deadlines:\n");
  for(int i=0; i<n; i++)
  scanf("%d", &pt[i]);
   for(int i=0; i<n; i++)
   proc[i]=i+1;

 Sort(proc, d, b, pt, n);
 //Lcm
 int t= lcmul(pt,n);
printf("\nEarliest Deadline Scheduling:\n");
printf("PID\t Burst\tdeadline\t Period\n");
 for(int i=0; i<n; i++)
  printf("%d\t%d\t%d\t%d\n", proc[i], b[i], d[i], pt[i]);
  printf("Scheduling occurs for %d ms\n\n", t);
  //EDF
```

```
int Deadline[i]= d[i];
For(int i=0; i<n; i++)
 {
   nextDeadline[n];
  For(int i=0; i<n; i++)
  {
   nextDeadline[i]=d[i];
    rem[i]=bt[i];
  }

   if (taskToExecute!=-1)
   {
    printf("\n\nTime: Task%d is running.\n",
   time, proc[taskToExecute]);
    rem[taskToExecute]--;
   }

  else
   {
    printf("\n\nTime: CPU is Idle.\n", time);
   }

  time++;
 }
}
```

Output:-

Enter the number of process: 4

Enter the CPU burst times:

2 1 5 6

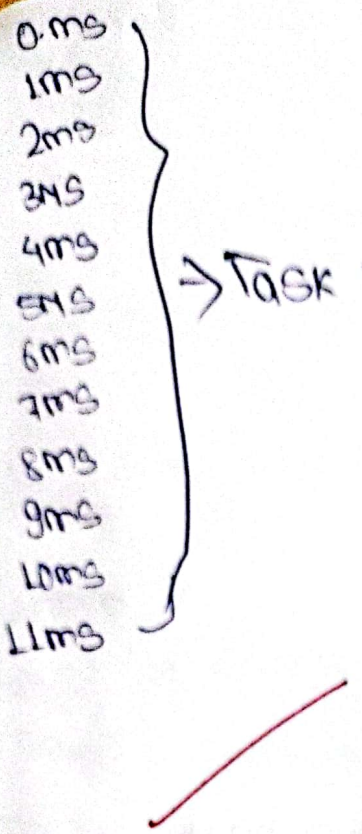
Enter the deadline: 1 4 5 6

Enter the time periods: 1 4 2 3

EArliest Deadline scheduling.

PID	BUST	Deadline	perid
1	2	1	1
2	1	4	4
3	5	5	2
4	6	6	3

CPU is free in 12ms


0.ms
1ms
2ms
3MS
4ms
5MS
6ms
7ms
8ms
9ms
10ms
11ms
→Task is running:

6) Proportional Scheduling

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main(){
    srand(time(NULL));
    int n;
    printf("Enter no. of processes:\n");
    scanf("%d",&n);
    int p[n],t[n],cum[n],m[n];
    int c=0,int total=0, count=0;
    printf("enter tickets of the processes:\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&t[i]);
        c+=t[i];
        cum[i]=c;
        p[i]=i+1;
        m[i]=0;
        total+=t[i];
    }
    while(count<n)
    {
        int wt=rand()%total;
        for(int i=0;i<n;i++)
        {
            if(wt<cum[i]&&m[i]==0)
            {
                printf("the winning is%d & winning
participants:%d\n",wt,p[i]);
                m[i]=1; count++;
}}}
```

```
printf("\n\nProbability:\n");
for(int i=0;i<n;i++)
{
printf("\n Probability of %d winning :%.2f %%\n",
p[i],((double)t[i]/total*100));
}
}
```

Output:

Enter the no. of processes: 3
Enter the tickets of the processes: 2 3 5

The winning no is 9 and winning participant is: 3
The winning no is 3 and winning participant is: 2
The winning no is 1 & winning participant is: 1

Probabilities:

The Probability of P1 winning: 20.00
The Probability of P2 winning: 30.00
The Probability of P3 winning: 50.00.

2) WAP to simulate producer-consumer problem using semaphores.

```
#include<stdio.h>
void main()
{
int buffer[10], bufsize, in, out, produce, consume, choice;
in=0;
out=0;
bufsize=10;
while(choice!=3)
{
printf("\n1. produce \t 2. consume \t 3. Exit");
printf("\n Enter your choice:");
scanf("%d", choice);
switch(choice)
{
case 1: if((in+1)%bufsize==out)
printf("\n Buffer is full");
else
{
printf("\n Enter the value:");
scanf("%d", &produce);
buffer[in]=produce;
in=(in+1)%bufsize;
}
break;
case 2: if(in==out)
printf("\n Buffer is Empty");
else
{consume=buffer[out];
```

```
printf("\n The consumed value is %d", consume);
out=(out+1)%bufsize;
}
break;
}
}
}
```

Output:

1. Produce 2. consume 3. Exit.

Enter your choice 2

Buff is empty

1. produce 2. consume 3. Exit.

Enter the value: 1

Enter the value 3

The consumed value is 3.

1. produce 2. consume 3 Exit.

Enter your choice. 3.

4/7/24