12/6/24

SURYA Gold
Date Page

Write a C program to simulate Real Time CPU scheduling Algorithms:

a) Rate - Monotonic

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void sort (int proc[], int b[], int pt[], int n)
{
    int temp=0;
    for(int i=0; i<n; i++)
    {
        for(int j=i; j<n; j++)
        {
            if (pt[j]<pt[i])
            {
                temp=pt[i];
                pt[i]=pt[j];
                pt[j]=temp;
                temp=b[j];
                b[j]=b[i];
                b[i]=temp;
                temp=proc[i];
                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 lcmu1 (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 number of processes :");
    scanf("%d", &n);
    int proc[n], b[n], pt[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 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("\nRate Monotonic scheduling:\n");
printf("PID\t Burst\t Period\n");
for(int i=0; i<n; i++)
    printf("%d\t\t%d\t\t%d\n", proc[i], b[i],
           pt[i]);
double sum = 0.0;
for(int i=0; i<n; i++)
{
    sum += (double) b[i] / pt[i];
}
double rhs = n*(pow(2.0, (1.0/n)) - 1.0);
printf("\n%lf <= %lf => %s\n", sum, rhs,
       (sum <= rhs) ? "true" : "false");
if(sum > rhs)
    exit(0);
printf("scheduling occurs for %d ms\n\n", l);
int time=0, prev=0, x=0;
while(time < l)
{
    int f=0;
    for(int i=0; i<n; i++)
    {
        if(time % pt[i] == 0)
            rem[i] = b[i];
        if(rem[i] > 0)
        {
            if(prev != proc[i])
```

```
{
    printf("%d ms onwards: Process %d running\n",
            time, proc[i]);
    prev = proc[i];
}
rem[i]--;
f = 1;
break;
x = 0;
}
}
if(!f)
{
if(x!=1)
{
printf("%d ms onwards: CPU is idle\n",
        time);
x = 1;
}
}
time++;
}
}
```

output :-

Enter the number of processes : 2

Enter the CPU burst times :

20 35

Enter the time period :

50 100

LCM = 100

Rate Monotonic scheduling :

PID	Burst Period	Time Period
1	20	50
2	35	100

0.750000 <= 0.828427 >= true

scheduling occurs for 100ms

0ms onwards: Process 1 running

20ms onwards: Process 2 running

50ms onwards: Process 1 running

70ms onwards: Process 2 running

75ms onwards: CPU is idle

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;
            temp = proc[i];
            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 lcmu (int p[], int n)
{
    int lcm = p[0];
    for (int i = 1; i<n; i++)
    {
        lcm = (lcm * p[i])/gcm (lcm, p[i]);
    }
    return lcm;
}
void main()
{
    int n;
```

```
printf("Enter the number 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", &d[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, d, b, pt, n);
int l = lcmu1(pt, n);
printf("\nEarliest Deadline Scheduling:\n");
printf("PID\t Burst\t Deadline\t Period\n");
for (int i=0; i<n; i++)
    printf("%d\t%d\t\t%d\t\t%d\n",
           proc[i], b[i], d[i], pt[i]);
printf("scheduling occurs for %d ms\n\n", l);
int time=0, prev=0, x=0;
int nextDeadlines[n];
for(int i=0; i<n; i++)
{
    nextDeadlines[i] = d[i];
    rem[i] = b[i];
}
```

```
while (time < l)
{
    for (int i=0; i<n; i++)
    {
        if (time % pt[i] == 0 && time != 0)
        {
            nextDeadlines[i] = time + d[i];
            rem[i] = b[i];
        }
    }
    int minDeadline = l + 1;
    int taskToExecute = -1;
    for (int i=0; i<n; i++)
    {
        if (rem[i] > 0 && nextDeadlines[i] < minDeadline)
        {
            minDeadline = nextDeadlines[i];
            taskToExecute = i;
        }
    }
    if (taskToExecute != -1)
    {
        printf("%d ms: Task %d is running.\n", time,
               proc[taskToExecute]);
        rem[taskToExecute]--;
    }
    else
    {
        printf("%d ms: CPU is idle.\n", time);
    }
    time++;
}
```

SURYA Gold
Date Page

Output:-

Enter the number of processes : 3
Enter the CPU burst times :
3
2
2
Enter the deadlines :
7
4
8
Enter the time periods :
20
5
10

Earliest Deadline scheduling

PID	Burst	Deadline	Period
2	2	4	5
1	3	7	20
3	2	8	10

scheduling occurs for 20ms
0ms : Task 2 is running.
1ms : Task 2 is running.
2ms : Task 1 is running.
3ms : Task 1 is running.
4ms : Task 1 is running.
5ms : Task 3 is running.
6ms : Task 3 is running.
7ms : Task 2 is running.
8ms : Task 2 is running.
9ms : CPU is idle.
10ms : Task 2 is running
11ms : Task 2 is running
12ms : Task 3 is running

13ms : Task 2 is running.
14ms : CPU is idle.
15ms : Task 2 is running
16ms : Task 2 is running.
17 ms : CPU is idle
18ms : CPU is idle.
19ms : CPU is idle.

c) Proportional Scheduling

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    srand(time(NULL));
    int n;
    printf("Enter number of processes :");
    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]);
        ct = 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 number is %d and
                winning participant is : %d \n", wt,
                    p[i]);
            m[i] = 1;
            count++;
        }
    }
}

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

Output :-

Enter the number of processes : 3
Enter tickets of the processes : 10 20 30
The winning number is 10 & winning participant is : 2
The winning number is 10 & winning participant is : 3
The winning number is 3 & winning participant is : 1
Probabilities :
The probability of P1 winning : 16.67
The probability of P2 winning : 33.33
The probability of P3 winning : 50.00

write a C program to simulate producer - consumer problem using semaphores.

```
#include <stdio.h>
void main()
{
  int buffer[10], bufsize, in, out, produce, consume,
      choice = 0;
  in = 0;
  out = 0;
  bufsize = 10;
  while(choice != 3)
  {
    printf("\n 1. 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("\nThe consumed value is %d", consume);
            out = (out+1)%bufsize;
        }
        break;
    }
  }
}
```

Output :-

1. Produce 2. Consume 3. Exit
Enter your choice : 1
Enter the value : 1
1. Produce 2. consume 3. Exit
Enter your choice : 1
Enter the value : 2
1. Produce 2. Consume 3. Exit
Enter your choice : 2
The consumed value is 1
1. Produce 2. Consume 3. Exit
Enter your choice : 2
The consumed value is 2.
1. Produce 2. Consume 3. Exit
Enter your choice : 2
Buffer is empty
1. Produce 2. consume 3. Exit
Enter your choice : 3