Tribhuvan University
Faculty of Humanities and Social Sciences

Lab report on:
Operating System Lab 5:
Producer-consumer problem & Round Robin scheduling algorithm

Submitted to:
Mr. Roshan Maharjan,
Er. Himal Chand Thapa ,
Department of Computer Application,
Himalaya College of Engineering,
Chyasal,Lalitpur

Submitted by:
Sujal Gurung
Roll no: 34
BCA II/IV
September 20, 2023

1 Objectives

- to implement a basic solution to producer-consumer problem
- to implement round robin process scheduling algorithm

2 Introduction

Producer-consumer problem is a classical scheduling problem where a producer and consumer process share same memory space(buffer). Producer produces a value & consumer uses (consumes) it. Error occurs when producer tries producing values when buffer is full, & consumer tries consuming when buffer is empty. Both conditions should be prevented.

Round Robin scheduling algortihm aims to provide equal attention to every process. It does so by setting a `quantum time` amount. Each process in queue is run for that amount of time & then swapped out to run the next one.

3 Lab Work

3.1 Write a program to implement producer consumer problem

```
#include<stdio.h>
int main() {
    int buffer[5],bufsize,in,out,produce,consume,choice=0;
    in = 0;    // current buffer index that will be produced
    out = 0;   // current buffer index that will be consumed
    bufsize = 5;
    while(1) {
        printf("\n1. Produce \t 2. Consume \t3. Exit");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice) {
            case 1:
                if((in+1)%bufsize == out)
                    printf("\nBuffer is Full so producer can't produce");
                else {
                    printf("\nEnter the value: ");
                    scanf("%d", &produce);
                    buffer[in] = produce;
                    in = (in+1)%bufsize;
                    printf("\nThe produced value is %d", produce);
                }
                break;
            case 2:
                if(in == out)
                    printf("\nBuffer is empty so consumer can't consume");
                else {
                    consume = buffer[out];
                    printf("\nThe consumed value is %d", consume);
                    out = (out+1)%bufsize;
                }
                break;
            default:
```

```
                return 0;
        }
    }
    return 0;
}
```

Output:

```
1. Produce       2. Consume     3. Exit
Enter your choice: 1
Enter the value: 1
The produced value is 1

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
Buffer is empty so consumer can't consume
    1. Produce       2. Consume     3. Exit
    Enter your choice: 3
```

3.2 Write a program to implement Round Robin algorithm

```
#include<stdio.h>
void main()
{
    int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
    float awt=0,att=0,temp=0;
    printf("Enter the no of processes -- ");
    scanf("%d",&n);
    for(i=0;i<n;i++) {
        printf("\nEnter Burst Time for process %d -- ", i+1);
        scanf("%d",&bu[i]);
        ct[i]=bu[i];
    }
    printf("\nEnter quantum time -- ");
    scanf("%d",&t);
    max=bu[0];
    for(i=1;i<n;i++) {
        if(max<bu[i]) max=bu[i];
    }
    for(j=0;j<(max/t)+1;j++) {
        for(i=0;i<n;i++) {
            if(bu[i]==0) break;
            if(bu[i]<=t) {
                tat[i]=temp+bu[i];
                temp=temp+bu[i];
                bu[i]=0;
            }
            else {
                bu[i]=bu[i]-t;
                temp=temp+t;
            }
        }
```

```
    }
    for(i=0;i<n;i++) {
        wa[i]=tat[i]-ct[i];
        att+=tat[i];
        awt+=wa[i];
    }
    printf("\nThe Average Turnaround time is -- %f",att/n);
    printf("\nThe Average Waiting time is -- %f",awt/n);
    printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
    for(i=0;i<n;i++) {
        printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
    }
}
```

Output:

```
Enter the no of processes -- 3
Enter Burst Time for process 1 -- 1
Enter Burst Time for process 2 -- 3
Enter Burst Time for process 3 -- 2
Enter quantum time -- 2

The Average Turnaround time is -- 1.666667
The Average Waiting time is -- 0.333333
PROCESS  BURST TIME      WAITING TIME    TURNAROUND TIME
1        1               0                1
2        3               4                1
3        2               3                5
```

4 Conclusion

Thus, we were able to implement a basic solution to the producer-consumer problem, as well as the Round robin scheduling algorithm.