Tribhuvan University
Faculty of Humanities and Social Sciences

Lab report on:
Operating System Lab 4:
Process Scheduling algorithms continued

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

- implement SJF process scheduling algorithm
- implement priority process scheduling algorithm
- understand ideal use-cases for both

2 Introduction

As discussed in the last lab, there are many popular process scheduling algorithms for deciding which process should be currently run. 2 other algorithms that OS developers use are:

- **Shortest Job First (SJF)**: Scheduler runs process in wait queue with shortest burst time.
- **Priority Scheduling**: Processes can be allocated priority level & the one with highest priority is run.

Both algorithms can have either pre-emptive or non pre-emptive implementations. In pre-emptive versions, scheduler can interrupt running process & run another process that it deems more important based on the algorithm.

3 Lab Work

NOTE: *For simplicity, we consider arrival time to be 0*

3.1 Write a program to implement SJF Scheduling algorithm and analyze the result

```
#include<stdio.h>
int main() {
  int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
  float avg_wt,avg_tat;
  printf("Enter number of process: ");
  scanf("%d",&n);

  printf("Enter Burst time for\n");
  for(i=0;i<n;i++) {
    printf("p%d: ",i+1);
    scanf("%d",&bt[i]);
    p[i]=i+1;
  }

  //sorting of burst times
  for(i=0;i<n;i++) {
    pos=i;
    for(j=i+1;j<n;j++) {
      if(bt[j]<bt[pos]) pos=j;
    }

    temp=bt[i];
    bt[i]=bt[pos];
```

```
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
  }

  wt[0]=0;
  for(i=1;i<n;i++) {
    wt[i]=0;
    for(j=0;j<i;j++) wt[i]+=bt[j];
    total+=wt[i];
  }

  avg_wt=(float)total / n;
  total=0;

  printf("Processes sorted based on order of execution\n");
  printf("Process\t Burst Time\t Waiting Time\t Turnaround Time");
  for(i=0;i<n;i++) {
    tat[i]=bt[i]+wt[i];
    total+=tat[i];
    printf("\np%d \t\t%d  \t\t%d \t\t%d",p[i],bt[i],wt[i],tat[i]);
  }

  avg_tat=(float)total / n;
  printf("\nAverage Waiting Time: %f",avg_wt);
  printf("\nAverage Turnaround Time: %f",avg_tat);
}
```

Output:

```
Enter number of process: 3
Enter Burst time for
p1: 7
p2: 4
p3: 2

Processes are sorted based on order of execution
Process  Burst Time      Waiting Time    Turnaround Time
p3              2               0               2
p2              4               2               6
p1              7               6               13
Average Waiting Time: 2.666667
Average Turnaround Time: 7.000000
```

3.2 Write a program to implement priority scheduling algorithm and analyze the result.

```
#include <stdio.h>
void swap(int *a,int *b) {
  int temp=*a;
  *a=*b;
  *b=temp;
}

int main() {
  int n;
```

```
  printf("Enter Number of Processes: ");
  scanf("%d",&n);

  // b is array for burst time, p for priority and index for process id
  int b[n], p[n], index[n];

  for(int i=0;i<n;i++) {
    printf("Enter Burst Time and Priority Value for Process %d: ",i+1);
    scanf("%d %d",&b[i],&p[i]);
    index[i]=i+1;
  }

  for(int i=0;i<n;i++) {
    int a=p[i],m=i;

    //Finding out highest priority element and placing it at its desired position
    for(int j=i;j<n;j++) {
      if(p[j] > a) {
        a=p[j];
        m=j;
      }
    }

    //Swapping processes
    swap(&p[i], &p[m]);
    swap(&b[i], &b[m]);
    swap(&index[i],&index[m]);
  }

  float total_wait=0, total_turnAround=0;
  printf("Processes sorted based on order of execution\n");
  printf("Process Id  \tBurst Time \tWait Time \tTurnaround Time\n");
  int wait_time=0;
  for(int i=0;i<n;i++) {
    printf("P%d \t\t %d \t\t %d \t\t %d\n",index[i],b[i],wait_time,wait_time +
    ↪  b[i]);
    total_wait += (float)wait_time;
    wait_time += b[i];
    total_turnAround += (float)wait_time;
  }
  printf("Average wait time: %f\n", total_wait / n);
  printf("Average turn around time: %f", total_turnAround / n);
  return 0;
}
```

Output:

```
Enter Number of Processes: 3
Enter Burst Time and Priority Value for Process 1: 7 1
Enter Burst Time and Priority Value for Process 2: 4 3
Enter Burst Time and Priority Value for Process 3: 2 2

Processes sorted based on order of execution:
Process Id      Burst Time      Wait Time       Turnaround Time
P2               4               0               4
P3               2               4               6
P1               7               6               13
Average wait time: 3.333333
```

```
Average turn around time: 7.666667
```

4 Conclusion

Both algorithms are good for specific purposes. Shortest job 1st priorotizes processes that can be completed quickly while Priority scheduling runs those that are deemed important/urgent. Although SJF has shorter average times here using same values, it depends on a lot of factors like arrival times, burst time, and priority. With a different combination of values, Priority Scheduling may have lower average times. As such, one cannot be said to be definitively better than the other. Each has its own strengths & cases where it is most appropriate.