OS Lab Assignment 3

CPU Scheduling Algorithms

1. First Come First Serve Algorithm

#include<iostream>

using namespace std;

int main(){

  cout<<"\t\"First come First Serve\" CPU Scheduling Algorithm\n\n";

  int numberOfProcesses;

  cout<<"Enter number of processes: ";cin>>numberOfProcesses;

  int burstTime[numberOfProcesses], arrivalTime[numberOfProcesses], processId[numberOfProcesses];

  for(int i=0;i<numberOfProcesses;i++){

    cout<<"Enter burst time for process["<<i+1<<"]: ";

    cin>>burstTime[i];

    cout<<"Enter arrival time for process["<<i+1<<"]: ";

    cin>>arrivalTime[i];

    processId[i] = i;

  }

  // Sort the processes based on arrival time

  for(int i=0;i<numberOfProcesses;i++){

    bool swap = false;

    for(int j=0;j<numberOfProcesses-i-1;j++){

      if(arrivalTime[j] > arrivalTime[j+1]){

        int temp = arrivalTime[j];

        arrivalTime[j] = arrivalTime[j+1];

        arrivalTime[j+1] = temp;

        temp = burstTime[i];

        burstTime[j] = burstTime[j+1];

        burstTime[j+1] = temp;

        temp = processId[j];

        processId[j] = processId[j+1];

        processId[j+1] = temp;

        swap = true;

      }

    }

    if(swap) break;

  }

  // Implementing the logic

  int finishTime[numberOfProcesses], waitingTime[numberOfProcesses], turnAroundTime[numberOfProcesses];

  finishTime[0] = arrivalTime[0] + burstTime[0];

  turnAroundTime[0] = finishTime[0] - arrivalTime[0];

  waitingTime[0] = turnAroundTime[0] - burstTime[0];

  float waitingTimeSum = waitingTime[0];

  float turnAroundTimeSum = turnAroundTime[0];

  for(int i=1;i<numberOfProcesses;i++){

    finishTime[i] = burstTime[i] + finishTime[i-1];

    turnAroundTime[i] = finishTime[i] - arrivalTime[i];

    turnAroundTimeSum += turnAroundTime[i];

    waitingTime[i] = turnAroundTime[i] - burstTime[i];

    waitingTimeSum += waitingTime[i];

  }

  waitingTimeSum = (1.0\*waitingTimeSum)/numberOfProcesses;

  turnAroundTimeSum = (1.0\*turnAroundTimeSum)/numberOfProcesses;

  // printing the output

  cout<<"Process ID\t Burst Time\t Arrival Time\t TurnAround Time\t Finish Time\n";

  for(int i=0;i<numberOfProcesses;i++){

    cout<<processId[i]<<"\t\t "<<burstTime[i]<<"\t\t "<<arrivalTime[i]<<"\t\t "<<turnAroundTime[i]<<"\t\t\t "<<finishTime[i]<<"\n";

  }

  cout<<"Avg. Waiting time: "<<waitingTimeSum<<"\n";

  cout<<"Avg. TurnAround Time: "<<turnAroundTimeSum<<"\n";

  return 0;

}

1. Priority Based Scheduling Algorithm

#include<iostream>

using namespace std;

int main(){

  cout<<"\t\tPriority Based Scheduling Algorithm\n";

  int numberOfProcesses;

  cout<<"Enter number of processes: ";cin>>numberOfProcesses;

  int burstTime[numberOfProcesses], priority[numberOfProcesses], processId[numberOfProcesses];

  for(int i=0;i<numberOfProcesses;i++){

    cout<<"Enter bursttime for process["<<i+1<<"]: ";

    cin>>burstTime[i];

    cout<<"Enter priority  for process["<<i+1<<"]: ";

    cin>>priority[i];

    processId[i] = i;

  }

  // soting the values based on priority of the process

  for(int i=0;i<numberOfProcesses;i++){

    bool swap = false;

    for(int j=0;i<numberOfProcesses-i-1;j++){

      if(priority[j] < priority[j+1]){

        int temp = burstTime[i];

        burstTime[j] = burstTime[j+1];

        burstTime[j+1] = temp;

        temp = priority[j];

        priority[j] = priority[j+1];

        priority[j+1] = temp;

        temp = processId[j];

        processId[j] = processId[j+1];

        processId[j+1] = temp;

        swap = true;

      }

    }

    if(swap) break;

  }

  int waitingTime[numberOfProcesses], turnAroundTime[numberOfProcesses];

  float avgWaitingTime = 0.0, avgTurnAroundTime = 0.0;

  for(int i=0;i<numberOfProcesses;i++){

    for(int j=0;j<i;j++){

      waitingTime[i] += burstTime[j];

    }

    avgWaitingTime += waitingTime[i];

    turnAroundTime[i] = burstTime[i] + waitingTime[i];

    avgTurnAroundTime += turnAroundTime[i];

  }

  avgTurnAroundTime = (1.0\*avgTurnAroundTime)/numberOfProcesses;

  avgWaitingTime = (1.0\*avgWaitingTime)/numberOfProcesses;

  cout<<"ProcessId\t Waiting Time\t Turn around time\t burst time\n";

  for(int i=0;i<numberOfProcesses;i++){

    cout<<processId[i]<<"\t\t"<<waitingTime[i]<<"\t\t\t"<<turnAroundTime[i]<<"\t\t\t"<<burstTime[i]<<"\n";

  }

  cout<<"Avg Turn Around Time: "<<avgTurnAroundTime<<"\n";

  cout<<"Avg Waiting Time: "<<avgWaitingTime;

  return 0;

}

1. Round Robin Scheduling Algorithm

#include<iostream>

using namespace std;

int main(){

  cout<<"\t\tRound Robin CPU Scheduling Algorithm\n";

  int numberOfProcesses;

  cout<<"Enter number of processes: ";cin>>numberOfProcesses;

  int arrivalTime[numberOfProcesses], burstTime[numberOfProcesses], dup\_burstTime[numberOfProcesses], processId[numberOfProcesses];

  for(int i=0;i<numberOfProcesses;i++){

    cout<<"Enter burst time of the process["<<i+1<<"]: ";

    cin>>burstTime[i];

    dup\_burstTime[i] = burstTime[i];

    cout<<"Enter arrival time of the process["<<i+1<<"]: ";

    cin>>arrivalTime[i];

  }

  int quantum;cout<<"Enter quantum size: ";cin>>quantum;

  // implementing the algorithm

  int time = 0, waitingTime[numberOfProcesses], turnAroundTime[numberOfProcesses];

  float avgWaitingTime = 0, avgTurnAroundTime = 0;

  while(1){

    bool isDone = true;

    for(int i=0;i<numberOfProcesses;i++){

      if(dup\_burstTime[i] > 0){

        isDone = false;

        if(dup\_burstTime[i] > quantum){

          time += quantum;

          dup\_burstTime[i] -= quantum;

        }

        else{

          time += dup\_burstTime[i];

          waitingTime[i] = time - burstTime[i];

          avgWaitingTime += waitingTime[i];

          turnAroundTime[i] = waitingTime[i] + burstTime[i];

          avgTurnAroundTime += turnAroundTime[i];

          dup\_burstTime[i] = 0;

        }

      }

    }

    if(isDone)break;

  }

  avgTurnAroundTime = (1.0\*avgTurnAroundTime)/numberOfProcesses;

  avgWaitingTime = (1.0\*avgWaitingTime)/numberOfProcesses;

  cout<<"ProcessId\t Waiting Time\t Turn around time\t burst time\n";

  for(int i=0;i<numberOfProcesses;i++){

    cout<<processId[i+1]<<"\t\t"<<waitingTime[i]<<"\t\t"<<turnAroundTime[i]<<"\t\t\t"<<burstTime[i]<<"\n";

  }

  cout<<"Avg Turn Around Time: "<<avgTurnAroundTime<<"\n";

  cout<<"Avg Waiting Time: "<<avgWaitingTime;

  return 0;

}

1. Shortest Job First Scheduling Algorithm

#include<iostream>

using namespace std;

int main(){

  cout<<"\t\tShortest Job First CPU Scheduling Algorithm\n\n";

  int numberOfProcesses;

  cout<<"Enter number of processes: ";cin>>numberOfProcesses;

  int processesTrack[numberOfProcesses][6];

  for(int i=0;i<numberOfProcesses;i++){

    cout<<"Enter burst   time for process["<<i+1<<"]: ";

    cin>>processesTrack[i][2];

    cout<<"Enter arrival time for process["<<i+1<<"]: ";

    cin>>processesTrack[i][1];

    processesTrack[i][0] = 1;

  }

  // Sort the processes based on arrival time

  for(int i=0;i<numberOfProcesses;i++){

    for(int j=0;j<numberOfProcesses-i-1;j++){

      if(processesTrack[j][1] > processesTrack[j+1][1]){

        for(int z=0;z<5;z++){

          int temp = processesTrack[j][z];

          processesTrack[j][z] = processesTrack[j+1][z];

          processesTrack[j+1][z] = temp;

        }

      }

    }

  }

  // implementing the algorithm

  // choosing the process based on min burst time

  processesTrack[0][3] = processesTrack[0][1] + processesTrack[0][2];

  processesTrack[0][5] = processesTrack[0][3] - processesTrack[0][1];

  processesTrack[0][4] = processesTrack[0][5] - processesTrack[0][2];

  int p;

  for(int i=1;i<numberOfProcesses;i++){

    int minBurstTime = processesTrack[i][2];

    int temp = processesTrack[i-1][3];

    for(int j=0;i<numberOfProcesses;j++){

      if(temp >= processesTrack[j][1] && minBurstTime >= processesTrack[j][2]){

        minBurstTime = processesTrack[j][2];

        p = j;

      }

    }

    processesTrack[p][3] = temp + processesTrack[p][2];

    processesTrack[p][5] = processesTrack[p][3] - processesTrack[p][1];

    processesTrack[p][4] = processesTrack[p][5] - processesTrack[p][2];

    for(int j=0;i<6;j++){

      int val = processesTrack[p][j];

      processesTrack[p][j] = processesTrack[i][j];

      processesTrack[i][j] = val;

    }

  }

  float turnAroungTimeAvg = 0.0, waitingTimeAvg = 0.0;

  for(int i=0;i<numberOfProcesses;i++){

    turnAroungTimeAvg += processesTrack[i][5];

    waitingTimeAvg += processesTrack[i][4];

  }

  turnAroungTimeAvg = (1.0\*turnAroungTimeAvg)/numberOfProcesses;

  waitingTimeAvg = (1.0\*waitingTimeAvg)/numberOfProcesses;

  // print the output

  cout<<"Process ID\t Burst Time\t Arrival Time\t TurnAround Time\t Waiting Time\n";

  for(int i=0;i<numberOfProcesses;i++){

    cout<<processesTrack[i][0]<<"\t\t"<<processesTrack[i][2]<<"\t\t"<<processesTrack[i][1]<<"\t\t\t"<<processesTrack[i][5]<<"\t\t"<<processesTrack[i][4]<<"\n";

  }

  cout<<"Avg waiting time: "<<waitingTimeAvg<<"\n";

  cout<<"Avg turn around time: "<<turnAroungTimeAvg<<"\n";

  return 0;

}