NAME: Arundarasi Rajendran

PRN:18070122081

BATCH: C4

Write a pseudocode and implement a C program to simulate the following CPU scheduling algorithms to find turnaround time and waiting time for the above problem.

a) FCFS

b) SJF (Preemptive & Non Preemptive )

c) Round Robin (quantum=2)

d) Priority (Non Preemptive )

DESCRIPTION

Assume all the processes arrive at the same time.

FCFS CPU SCHEDULING ALGORITHM

For FCFS scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. The scheduling is performed on the basis of arrival time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. Calculate the waiting time and turnaround time of each of the processes accordingly.

PSEUDOCODE:

In main function

Program start

Initialize variable avtat to 0

Initialize avwt to 0

Declare variable n,i,j

Declare array bt[20], wt[[20], tat[20]

Input number of processes

For i=0 to n

Print process [i]

Input burst time in ‘bt’ array

End for

Initialize wt[0] to 0

For i=1 to n

Initialize wt[i]=0

For j=0 to i

Add burst time to wait time wt[i]=wt[i]+bt[i]

End for

End for

Print process , burst time, waiting time

For i=1 to n

Calculate burst time and wait time then store in ‘tat’ array

tat[i]=bt[i]+wt[i];

Add wait time with variable ‘avwt’ (Average waiting time)

avwt=avwt+wt[i];

Add tat time with variable ‘avtat’ (Average turnaround time)

avtat=avtat+tat[i];

End for

Divide avwt by i

Avwt = avwt/i

Divide avtat by i

Avtat = avtat/i

Print average waiting time and average turnaround time

Stop

SJF CPU SCHEDULING ALGORITHM

For SJF scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. Arrange all the jobs in order with respect to their burst times. There may be two jobs in the queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly.

PSEUDOCODE:

In main function

Program start

Declare variable n,i,j ,pos,temp,total of type int

Declare variable avwt, avtat of type float

Declare array bt[20], wt[[20], tat[20], p[20]

Input number of processes

For i=0 to n

Print process [i]

Input burst time in bt array

Increment i in array ‘p’

End for

For i=0 to n

pos=i

For j=i+1 to n

If bt[j] > bt[pos]

pos=j

End if

End for

Swap value bt[i] with bt[pos]

temp=p[i]

p[i]=p[pos]

p[pos]=temp;

End for

Initialize wt[0]=0

For i=1 to n

Initialize wt[i]=0

For j=0 to i

wt[i]=wt[i]+bt[j];

End for

total = total + wt[i];

End for

Calculate average waiting time

avwt=(float)total/n

Initialize total to 0;

Print process burst time waiting time

For i=0 to n

tat[i]=bt[i]+wt[i]

total = total + tat[i];

Print process, burst time, and waiting time values

End for

avtat=(float)total/n

Print average waiting time and average turnaround time

ROUND ROBIN CPU SCHEDULING ALGORITHM

For a round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the size of the time slice. Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

PSEUDOCODE:

In main function

Program start

Declare variables i,x,time\_quantum, limit of type int

Initialize variables wt=0, tat=0 of type int

Declare arrays at[10], bt[10], temp[10] of type int

Declare variables awt, avtat of type float

Input total number of processes

Assign variable x to limit

For i=0 to limit

Print process [i]

Input arrival time into at[]

Input burst time into bt[]

Assign bt[i] inside temp[i]

End for

Input quantum time

For total = 0 to i=0 to x!=0

If temp[i] <= time\_quantum && temp[i] > 0

total = total + temp[i];

temp[i] = 0;

Initialize counter = 1;

End if

Else if temp[i] > 0

Temp[i] = temp[i] - time\_quantum

Total = total + time\_quantum

End else if

if temp[i] == 0 && counter == 1

Decrement x;

Print bt[i] and total - at[i];

Calculate wt = wt + total - at[i] - bt[i]

Calculate tat = tat + total - at[i]

Initialize counter = 0

End if

If i == limit - 1

Initialize i=0

End if

Else if at[i+1] <= total

i++

End if

Else

Initialize i = 0

End if

End if

Calculate average waiting time and average turnaround time

awt = wt\*(1.0)/limit

avtat = tat\*(1.0)/limit

Print average waiting time and average turnaround time

PRIORITY CPU SCHEDULING ALGORITHM

For priority scheduling algorithms, read the number of processes/jobs in the system, their CPU burst times, and the priorities. Arrange all the jobs in order with respect to their priorities. There may be two jobs in the queue with the same priority, and then FCFS approach is to be performed. Each process will be executed according to its priority. Calculate the waiting time and turnaround time of each of the processes accordingly.

PSEUDOCODE:

In main function

Program start

Declare variables temp,n,i,j of type int

Initialize variables totwt=0, totta=0 of type int

Declare arrays et[10], at[10], p[10] , st[10], ft[10], wt[10], ta[10] of type int

Declare variables awt, avtat of type float

Declare arrays pn[10][10], t[10] of type char

Input total number of processes

For i=0 to n

Input process name, arrival time, execution time, and priority

End for

For i=0 to n

For j=0 to n

If p[i] < p[j]

Swap p[i] and p[j]

temp=p[i]

p[i]=p[j]

p[j]=temp

Swap at[i] and at[j]

temp=at[i]

at[i]=at[j]

at[j]=temp

Swap et[i] and et[j]

temp=at[i]

at[i]=at[j]

at[j]=temp

strcpy(t,pn[i])

strcpy(pn[i]. pn[j])

strcpy(pn[j], t)

End if

End if

End if

For i=0 to n

If i==0

st[i]=at[i]

wt[i]=st[i]-at[i]

ft[i]=st[i]+et[i]

ta[i]=ft[i]-at[i]

End if

Else

st[i]=ft[i-1]

wt[i]=st[i]-at[i]

ft[i]=st[i]+et[i]

ta[i]=ft[i]-at[i]

End else

totwt=totwt+wt[i]

totta=totta+ta[i]

awt=(float)totwt/n

ata=(float)totta/n

Print process name, arrival time, execution time, priority, waiting time, and ttatime.

Average waiting time and average turnaround time