**LAPORAN TUGAS SISTEM OPERASI**

**ALGORITMA PENJADWALAN**



Oleh

1. Andaru Bagus Gandi N. (E31181929)
2. Dita Shafira Wahyudi (E31181830)
3. Salsabila Eksananda (E31181971)
4. Muhammad Alif M. (E31181991)
5. Muhammad Nazil Z. (E31182056)

**PROGRAM STUDI MANAJEMEN INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI JEMBER**

**2019**

1. FCSF

* Source Code

#include<stdio.h>

#include<string.h>

main(){

int n, ar[100], b[100], i, j, tmp, wt[100], ta[100], time[100];

int totWT=0, totTA=0;

float AvWT, AvTA;

char name[25][25], tmpName[25];

printf("\t======== Penjadwalan CPU Metode First Come First Serve ========\n");

puts("");

printf("Masukan Jumlah Proses (Angka)\t = "); scanf("%d", &n);

puts("");

for(i=0; i<n; i++){

fflush(stdin);

printf("Nama Proses\t = "); gets(name[i]);

printf("Arrival Time\t = "); scanf("%d", &ar[i]);

printf("Burst Time\t = "); scanf("%d", &b[i]);

puts("");

}

for(i=0; i<n; i++){

for(j=i+1; j<n; j++)

if(ar[i]>ar[j]){

strcpy(tmpName, name[i]);

strcpy(name[i], name[j]);

strcpy(name[j], tmpName);

tmp=ar[i];

ar[i]=ar[j];

ar[j]=tmp;

tmp=b[i];

b[i]=b[j];

b[j]=tmp;

}}

time[0]=ar[0];

puts("\n\t- Tabel Proses -");

puts("==========================================");

printf("| No | Proses\t | Time Arrival\t | Burst |\n");

puts("""""""""""""""");

for (i=0; i<n; i++){

printf("| %2d | %s\t | \t%d\t | %d\t |\n", i+1, name[i], ar[i], b[i]);

time[i+1]=time[i]+b[i];

wt[i]=time[i]-ar[i];

ta[i]=time[i+1]-ar[i];

totWT+=wt[i];

totTA+=ta[i];

}

puts("==========================================");

printf("\tWaiting Time\t= %d \n", totWT);

printf("\tTurn Arround Time\t= %d \n", totTA);

puts("\n\t- Tabel Waktu Proses -");

puts("==================================================");

printf("| No | Proses\t | Waiting Time\t | Turn Arround\t |\n");

puts("""""""""""""""""""");

for(i=0; i<n; i++){

printf("| %2d | %s\t | \t%d\t | \t%d\t |\n", i+1, name[i], wt[i], ta[i]);

}

puts("==================================================");

puts("\n");

puts("\t- Diagram Gant -\n");

for(i=0; i<n; i++){

printf(" %s\t ", name[i]);

}

puts("");

for(i=0; i<n; i++){

printf("|=========");

}

printf("|\n");

for(i=0; i<=n; i++){

printf(" %d\t ", time[i]);

}

printf("(Dari Penjumlahan Burst)");

puts("\n");

AvWT=(float)totWT/n;

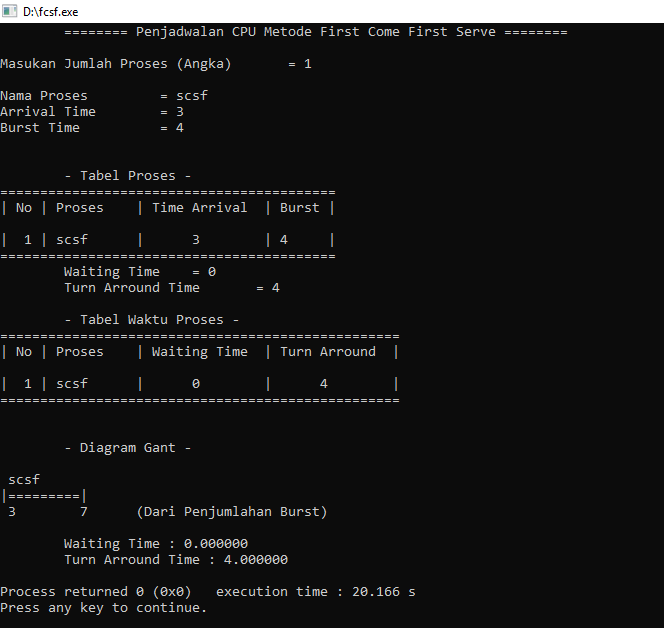
AvTA=(float)totTA/n;

printf("\tWaiting Time : %f\n", AvWT);

printf("\tTurn Arround Time : %f\n", AvTA);

}

* Output



1. Shortest Job

* Source Code

#include <stdio.h>

#include <string.h>

main()

{

int i,j,n,brust\_time[10],start\_time[10],end\_time[10],wait\_time[10],temp,tot;

float avg;

printf("Enter the No. of jobs:\n\n");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

printf("\n \n Enter %d process burst time:\n",i);

scanf("%d",&brust\_time[i]);

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

if(brust\_time[i]>brust\_time[j])

{

temp=brust\_time[i];

brust\_time[i]=brust\_time[j];

brust\_time[j]=temp;

}}

if(i==1)

{

start\_time[1]=0;

end\_time[1]=brust\_time[1];

wait\_time[1]=0;

}

else

{

start\_time[i]=end\_time[i-1];

end\_time[i]=start\_time[i]+brust\_time[i];

wait\_time[i]=start\_time[i];

}}

printf("\n\n BURST TIME \t STARTING TIME \t END TIME \t WAIT TIME\n");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for(i=1;i<=n;i++)

{

printf("\n %5d %15d %15d %15d",brust\_time[i],start\_time[i],end\_time[i],wait\_time[i]);

}

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for(i=1,tot=0;i<=n;i++)

tot+=wait\_time[i];

avg=(float)tot/n;

printf("\n\n\n AVERAGE WAITING TIME=%f",avg);

for(i=1,tot=0;i<=n;i++)

tot+=end\_time[i];

avg=(float)tot/n;

printf("\n\n AVERAGE TURNAROUND TIME=%f",avg);

for(i=1,tot=0;i<=n;i++)

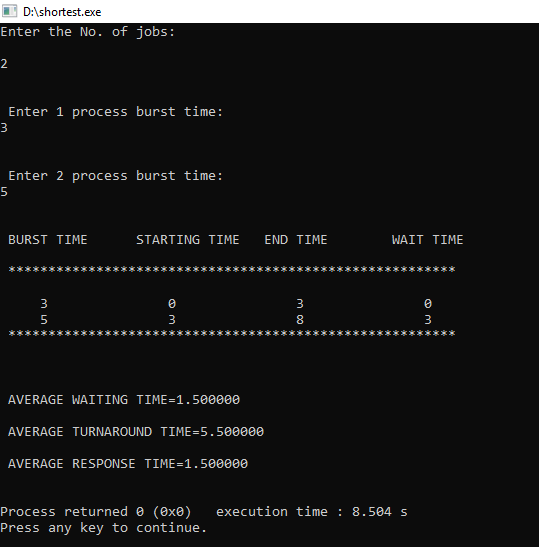
tot+=start\_time[i];

avg=(float)tot/n;

printf("\n\n AVERAGE RESPONSE TIME=%f\n\n",avg);

}

* Output



1. Round robin

* Source Code

#include<stdio.h>

#include<conio.h>

main()

{

int st[10],bt[10],wt[10],tat[10],n,tq;

int i,count=0,swt=0,stat=0,temp,sq=0;

float awt=0.0,atat=0.0;

clrscr();

printf(“Enter number of processes:”);

scanf(“%d”,&n);

printf(“Enter burst time for sequences:”);

for(i=0;i<n;i++)

{

scanf(“%d”,&bt[i]);

st[i]=bt[i];

}

printf(“Enter time quantum:”);

scanf(“%d”,&tq);

while(1)

{

for(i=0,count=0;i<n;i++)

{

temp=tq;

if(st[i]==0)

{

count++;

continue;

}

if(st[i]>tq)

st[i]=st[i]-tq;

else

if(st[i]>=0)

{

temp=st[i];

st[i]=0;

}

sq=sq+temp;

tat[i]=sq;

}

if(n==count)

break;

}

for(i=0;i<n;i++)

{

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

swt=swt+wt[i];

stat=stat+tat[i];

}

awt=(float)swt/n;

atat=(float)stat/n;

printf(“Process\_no Burst time Wait time Turn around time

“);

for(i=0;i<n;i++)

printf(“%d %d %d %d

“,i+1,bt[i],wt[i],tat[i]);

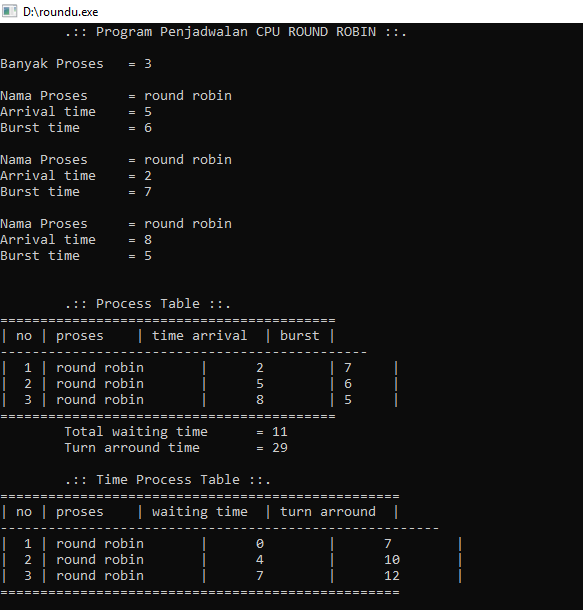
printf(“Avg wait time is %f

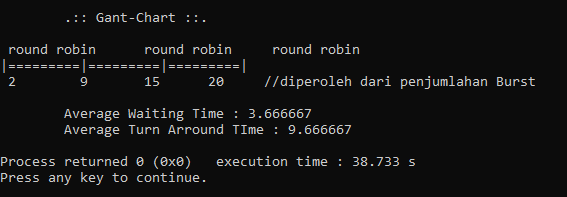
Avg turn around time is %f”,awt,atat);

getch();

}

* Output





D. Short Preemptive

-source code:

#include<bits/stdc++.h>

using namespace std;

struct Process

{

int pid; // process ID

int bt; // burst Time

};

/\*

this function is used for sorting all

processes in increasing order of burst time

\*/

bool comparison(Process a, Process b)

{

return (a.bt < b.bt);

}

// function to find the waiting time for all processes

void findWaitingTime(Process proc[], int n, int wt[])

{

// waiting time for first process is 0

wt[0] = 0;

// calculating waiting time

for (int i = 1; i < n ; i++)

{

wt[i] = proc[i-1].bt + wt[i-1] ;

}

}

// function to calculate turn around time

void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])

{

// calculating turnaround time by adding bt[i] + wt[i]

for (int i = 0; i < n ; i++)

{

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

}

}

// function to calculate average time

void findAverageTime(Process proc[], int n)

{

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

// function to find waiting time of all processes

findWaitingTime(proc, n, wt);

// function to find turn around time for all processes

findTurnAroundTime(proc, n, wt, tat);

// display processes along with all details

cout << "\nProcesses "<< " Burst time "

<< " Waiting time " << " Turn around time\n";

// calculate total waiting time and total turn around time

for (int i = 0; i < n; i++)

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

cout << " " << proc[i].pid << "\t\t"

<< proc[i].bt << "\t " << wt[i]

<< "\t\t " << tat[i] <<endl;

}

cout << "Average waiting time = "

<< (float)total\_wt / (float)n;

cout << "\nAverage turn around time = "

<< (float)total\_tat / (float)n;

}

// main function

int main()

{

Process proc[] = {{1, 21}, {2, 3}, {3, 6}, {4, 2}};

int n = sizeof proc / sizeof proc[0];

// sorting processes by burst time.

sort(proc, proc + n, comparison);

cout << "Order in which process gets executed\n";

for (int i = 0 ; i < n; i++)

{

cout << proc[i].pid <<" ";

}

findAverageTime(proc, n);

return 0;

}

Output:

