LAB : Priority Scheduling- Pre-emptive by considering High No High Priority

Name of the Student: **Rahebar Shaikh** Div. **AI & DS - B**  Roll No.**72** PRN:**12120069**

**Aim**: Implementation of Priority Scheduling- Pre-emptive by considering High No High Priority

Code-

#include <stdio.h>

// structure representing a structure

struct priority\_scheduling

{

// name of the process

char process\_name;

// time required for execution

int burst\_time;

// waiting time of a process

int waiting\_time;

// total time of execution

int turn\_around\_time;

// priority of the process

int priority;

};

int main()

{

// total number of processes

int number\_of\_process;

// total waiting and turnaround time

int total = 0;

// temporary structure for swapping

struct priority\_scheduling temp\_process;

// ASCII numbers are used to represent the name of the process

int ASCII\_number = 65;

// swapping position

int position;

// average waiting time of the process

float average\_waiting\_time;

// average turnaround time of the process

float average\_turnaround\_time;

printf("Enter the total number of Processes: ");

// get the total number of the process as input

scanf("%d", &number\_of\_process);

// initializing the structure array

struct priority\_scheduling process[number\_of\_process];

printf("\nPlease Enter the Burst Time and Priority of each process:\n");

// get burst time and priority of all process

for (int i = 0; i < number\_of\_process; i++)

{

// assign names consecutively using ASCII number

process[i].process\_name = (char)ASCII\_number;

printf("\nEnter the details of the process %c \n", process[i].process\_name);

printf("Enter the burst time: ");

scanf("%d", &process[i].burst\_time);

printf("Enter the priority: ");

scanf("%d", &process[i].priority);

// increment the ASCII number to get the next alphabet

ASCII\_number++;

}

// swap process according to high priority

for (int i = 0; i < number\_of\_process; i++)

{

position = i;

for (int j = i + 1; j < number\_of\_process; j++)

{

// check if priority is higher for swapping

if (process[j].priority < process[position].priority)

position = j;

}

// swapping of lower priority process with the higher priority process

temp\_process = process[i];

process[i] = process[position];

process[position] = temp\_process;

}

// First process will not have to wait and hence has a waiting time of 0

process[0].waiting\_time = 0;

for (int i = 1; i < number\_of\_process; i++)

{

process[i].waiting\_time = 0;

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

{

// calculate waiting time

process[i].waiting\_time += process[j].burst\_time;

}

// calculate total waiting time

total += process[i].waiting\_time;

}

// calculate average waiting time

average\_waiting\_time = (float)total / (float)number\_of\_process;

// assigning total as 0 for next calculations

total = 0;

printf("\n\nProcess\_name \t Burst Time \t Waiting Time \t Turnaround Time\n");

printf("------------------------------------------------------------\n");

for (int i = 0; i < number\_of\_process; i++)

{

// calculating the turn around time of the processes

process[i].turn\_around\_time = process[i].burst\_time + process[i].waiting\_time;

// calculating the total turnaround time.

total += process[i].turn\_around\_time;

// printing all the values

printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process\_name, process[i].burst\_time, process[i].waiting\_time, process[i].turn\_around\_time);

printf("\n-----------------------------------------------------------\n");

}

// calculating the average turn\_around time

average\_turnaround\_time = (float)total / (float)number\_of\_process;

// average waiting time

printf("\n\n Average Waiting Time : %f", average\_waiting\_time);

// average turnaround time

printf("\n Average Turnaround Time: %f\n", average\_turnaround\_time);

return 0;

}

Output:

Text

Description automatically generated